IO-Link Integration
Part 1

Technical Specification

for PROFIBUS
and PROFINET

related to
IO-Link specification V1.0

Version 1.00
December 2007

Order No: 2.812

Document Identification: TC3-07-0004a
File name: IO-Link-Integration-P1_2812_V100_Dec07

Prepared by the PROFIBUS Working Group WG16 "IO-Link" within Technical Committee 3 "Application Profiles".

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

may:	indicates flexibility of choice with no implied preference.
should:	indicates flexibility of choice with a strongly preferred implementation.
shall:	indicates a mandatory requirement. Designers **shall** implement such mandatory requirements to ensure interoperability and to claim conformance with this specification.

Publisher:
PROFIBUS Nutzerorganisation e.V.
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 721 / 96 58 590
Fax: +49 721 / 96 58 589
E-mail: info@profibus.com
Web site: www.profibus.com

Content

List of figures

List of tables

Revision Log

Version	Originator	Date	Change Note / History / Reason
0.1	Subteam PB integration	22-May-07	Initial version (Framework)
0.2	Subteam PB integration	02-July-07	Incorporation of the system description
0.3	Subteam PB integration	10-July-07	Merging of system description and specification
0.4	Subteam PB integration	26-Aug-07	Covering change requests CRs 1…36 in project DB
0.9	Subteam PB integration	13-Oct-07	Covering change requests CRs 37…48 in project DB
0.95	Subteam PB integration	30-Oct-07	Covering change requests CRs 49…67 in project DB
0.99	Subteam PB integration	31-Oct-07	Draft version for PI-Review after Subteam telecon
1.00	Subteam PB integration	30-Dec-07	Final version, covering CRs 68…84 in project DB

1 Management summary - scope of this document

The IO-Link consortium within PROFIBUS & PROFINET International has been developing a point-to-point digital communication technology mainly for sensors and actuators in factory automation, which are using nowadays many more optical, magnetic, capacitive, radio-based, and other effects in conjunction with very small and cost-effective micro-controllers. With the help of this new communication technology the obstacles of traditional signal connection technologies such as switching 24V, analog 0...10V, etc. can be overcome. These obstacles are

- Limited signal size (no data structures)
- No centralized and system-conform parameterization (tool and program controlled)
- Very limited diagnosis capabilities
- Error-prone analog signal transmission

The new communication technology IO-Link requires more interdependency between the sensor and actuator manufacturers and the diverse fieldbus communication organisations as opposed to the traditional signal connection technologies. Digital communication always needs mutual conformance agreements in order to achieve sufficient reliability. However, this is not enough for a complete integration in an existing fieldbus system with for example its device description and parameterization methodologies, I/O data structure conventions, device models and addressing schemes, diagnosis definition and reporting systems, just to name a few.

The IO-Link device manufacturers are striving for a general integration strategy for the most important fieldbus systems with the smallest impact on the logistics of their IO-Link devices. Therefore support for several types of device descriptions or proxy function blocks for PLCs shall be omitted. IO-Link system handling at least should be similar from one fieldbus system to the other.

It is the objective of this specification to describe an examplary integration solution into PROFIBUS and PROFINET for the counterpart of an IO-Link device, the IO-Link master. The model behind shall be generic and comprehensive enough to be acceptable for the most important other fieldbusses as well. Thus this specification starts with a detailed description of the integration strategy, the system architectures, the features, and the behavior. It is followed by the typical detailed mapping descriptions, which can be found in other PI integration documents such as HART or fieldbus integration into PROFINET. This specification incorporates the results of several IO-Link integration subteams, for example "port configuration", "data storage", "device description", and others.

2 List of affected patents

There is no affected patent known by the members of the IO-Link integration working groups. Thus the list is empty. No patent search – neither external nor internal – has been executed by the members of the IO-Link Integration working groups so far. PROFIBUS International does not guarantee the completeness of this list.

3 Related documents and references

3.1 Related documents

The following documents are the basis for the designs in this document herein.

- IO-Link Communication Specification V1.0, PNO-Order No. 2.802

- IO-Link System Integration V0.99
- IO-Link Device Description Language V0.1
- IEC 61158-5-3:FDIS, *Industrial communication networks – Fieldbus specifications – Part 5-3: Application layer service definition*
- IEC 61158-5-10:FDIS, *Industrial communication networks – Fieldbus specifications – Part 5-10: Application layer service definition*
- IEC 61158-6-3:FDIS, *Industrial communication networks – Fieldbus specifications – Part 4-3: Application layer protocol specification*
- IEC 61158-6-10:FDIS, *Industrial communication networks – Fieldbus specifications – Part 4-10: Application layer protocol specification*
- PNO specification, *Tool Calling Interface*, V1.0, Order-No. 2.602
- FDT Group, *FDT Interface Specification*, V1.2.1
- FDT Group, *Annex to FDT Specification: Field Device Tool for IO-Link*, Draft V0.8

Figure 1 shows the documents required for the implementation of particular IO-Link components. It is intended to incorporate the currently independent "System Integration" and "Device Description" into one of the next versions of the "IO-Link Specification".

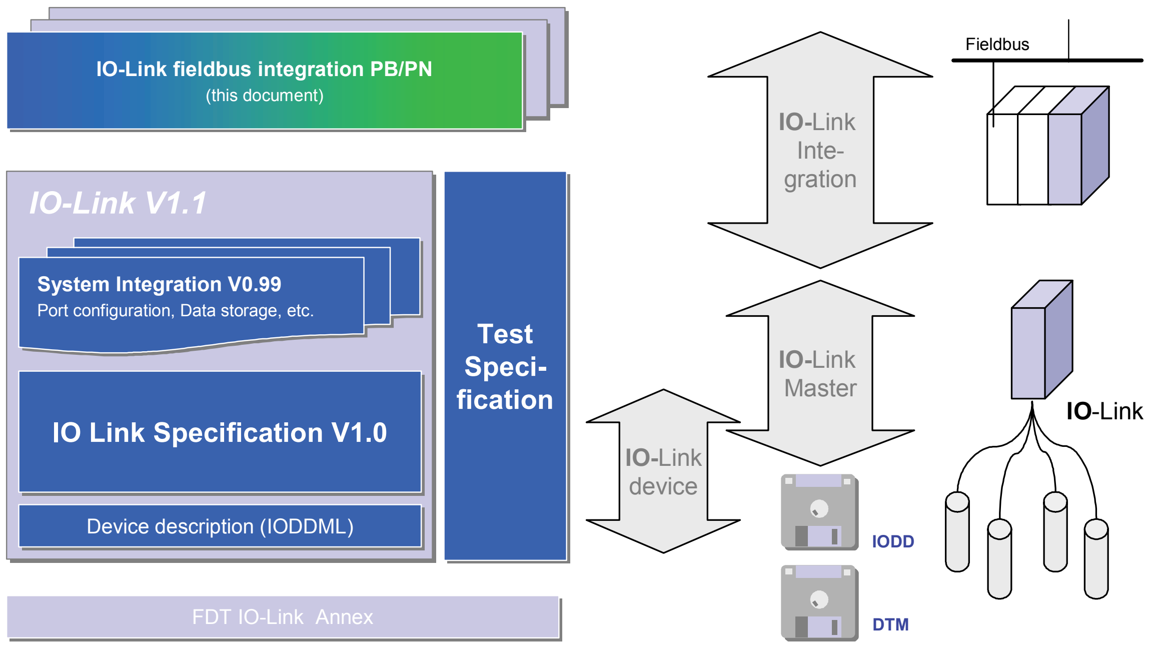


Figure 1 — Overview of the IO-Link specifications

3.2 References

- Specification for PROFIBUS Device Description and Device Integration, Volume 1: GSD, V5.04; Order No. 2.122
- GSDML Specification for PROFINET IO, V2.1; Order No. 2.352
- Profile Guidelines, Part 1: Identification & Maintenance Functions, V1.1; Order No. 3.502
- Profile Guidelines, Part 2: Data Types, Programming Languages, and Platforms, V1.0; Order No. 3.512
- Profile Guidelines, Part 3: Diagnosis, Alarms, and Time Stamping, V1.0; Order No. 3.522
- Communication Function Blocks on PROFIBUS DP and PROFINET IO, V2.0; Order No. 2.182

4 Definitions and Abbreviations

4.1 Definitions

This specification relies on the definitions in [8], [9], [13], [14], [15], and [16].

4.2 Abbreviations

AI	Analog Input
AO	Analog Output
AR	Application Relationship (IEC 61158)
ASE	Application Service Element (IEC 61158)
CPF3	Communication Profile Family 3 in IEC 61158 (PROFIBUS)
CPF10	Communication Profile Family 10 in IEC 61158 (PROFINET)
C-R-D	Channel-Related-Diagnosis
Comm-FB	Communication Function Blocks (according IEC 61131-3)
DCS	Distributed Control System
DI	Digital Input
DO	Digital Output
DP-V1	(PROFIBUS) Decentralized Peripherals Version 1 (acyclic services MS1, MS2 among others)
DTM	Device Type Manager
GSD	General Station Description
FDT	Field Device Tool
FI_Index	Function Invocation Index (CALL mechanism of PB-DP; IEC 61158)
IM0	I&M record 0 (example; others are IM1, IM2...IM16, IM17, etc.)
IOCS	Input Output object Consumer Status (IEC 61158-6-10, PROFINET IO)
IODD	IO-link Device Description
IODML	IO-Link Device description Markup Language
IOL-CM	IO-Link Compact Master (fieldbus gateway device, e.g. DP-Slave or IO-Device)
IOL-D	IO-Link Device (sensor or actuator)
IOL-M	IO-Link Master (IOL-CM or IOL-MM)
IOL-MM	IO-Link master module (of a physically modular remote I/O)
IOPD	IO-link Parameterization and Diagnosis tool
IOPS	Input Output object Provider Status (IEC 61158-6-10, PROFINET IO)
IOxS	IOCS or IOPS (x stands for "C" or "P")
iPar	Individual Parameters of a field device application
I&M	Identification and Maintenance (functions)
LR	Load Region (PROFIBUS DP; IEC 61158)
MS0	Cyclic Master class 1 to slave Services
MS1	Acyclic Master class 1 to slave Services
MS2	Acyclic Master class 2 to slave Services
NC	Not connected or Numerical Controller
OP	Operator Panel
OSSD	Output Signal Switching Device
PB-DP	PROFIBUS DP
PC	Personal Computer

PCT	Port Configurator Tool
PDU	Protocol Data Unit
PG	Programmer (specialized PC with engineering and programming software)
PLC	Programmable Logic Controller
PN-IO	PROFINET IO
Prm block	Parameter Block (Initial parameter message of PROFIBUS DP)
SIO	Standard Input Output (mode of IO-Link communication channel)
TCI	Tool Calling Interface
URL	Universal Resource Locator (Internet)

4.3 Conventions

In this specification UML2 notation is used as well as the recommendations of the IEC 62390 for state and transition descriptions.

All "reserved" bits herein shall be set "0". Reserved text strings shall be set "20h" (blank).

5 IO-Link master integration strategy (system description)

In this clause a comprehensive view on all the aspects of integration are presented, from the classic way of sensor and actuator connection, aspects of data structure formats and processing within programmable logic controllers to the entire architectures, topologies, components and their interactions.

5.1 Classic connection of sensors and actuators and new challenges

Usually, the classic sensors and actuators are connected to a fieldbus system via so-called remote I/O devices. These devices may provide variable configurations of digital and/or analog input and output modules (modular remote I/O) or fixed number of I/O channels (compact remote I/O). The ingress protection of modular remote I/O usually is IP20 and for compact remote I/O it is IP67.

5.1.1 Binary sensors and actuators

The majority of sensors in factory automation are binary switches or devices behaving like a binary switch. Figure 2 is showing the classic integration in PB-DP or PN-IO via remote I/O, addressed by node numbers. In case of a modular remote I/O, modules may be plugged in so-called slots, here a digital input module (DI Module) in slot 4. The minimum data structure per slot that can be transmitted across the bus is one byte. According to the conventions of PB-DP and PN-IO the eight inputs of the DI Module in the example are packed as data type "Unsigned8".

The particular switch sensor of Figure 2 is connected to input "7". The programmable logic controller is cyclically receiving (e.g. MS0 services) the input information via its process image. The location information of the particular switch in this case is transparent in the user program ("I4.7").

The only necessary convention between a switch sensor manufacturer and the system manufacturer is the coding of a "closed" (24V = true = "1") and an "open" switch (0V = false = "0").

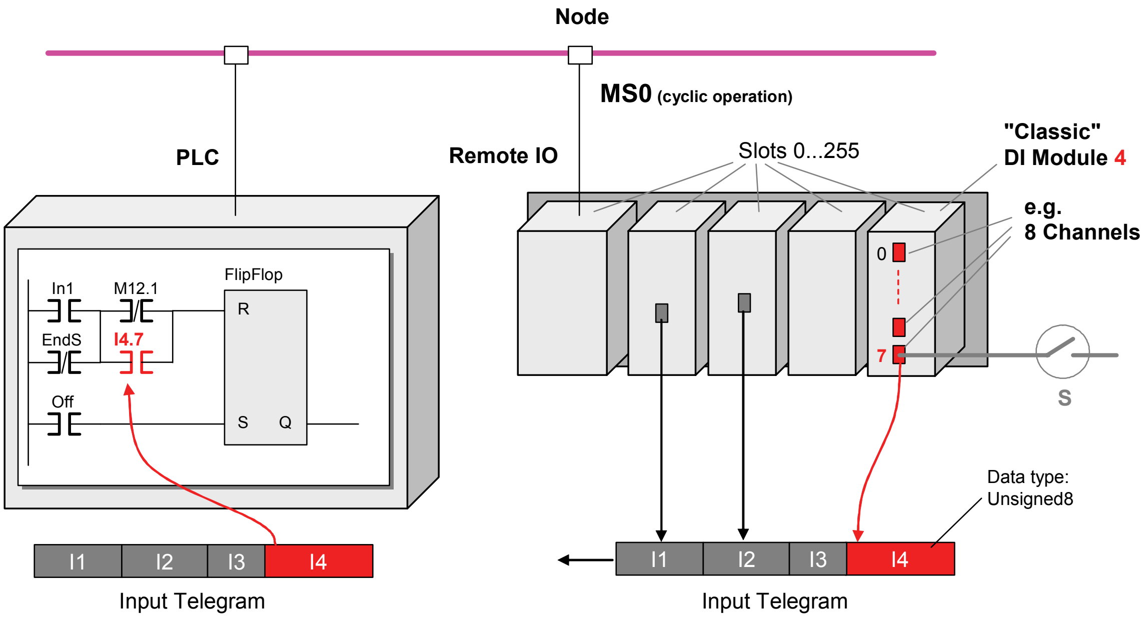


Figure 2 — Classic integration of binary switches

Unfortunately it was not possible to keep the one-to-one mapping of a module to a particular slot. It was necessary to introduce an indirection via the byte number of the transmitted telegrams according Figure 3.

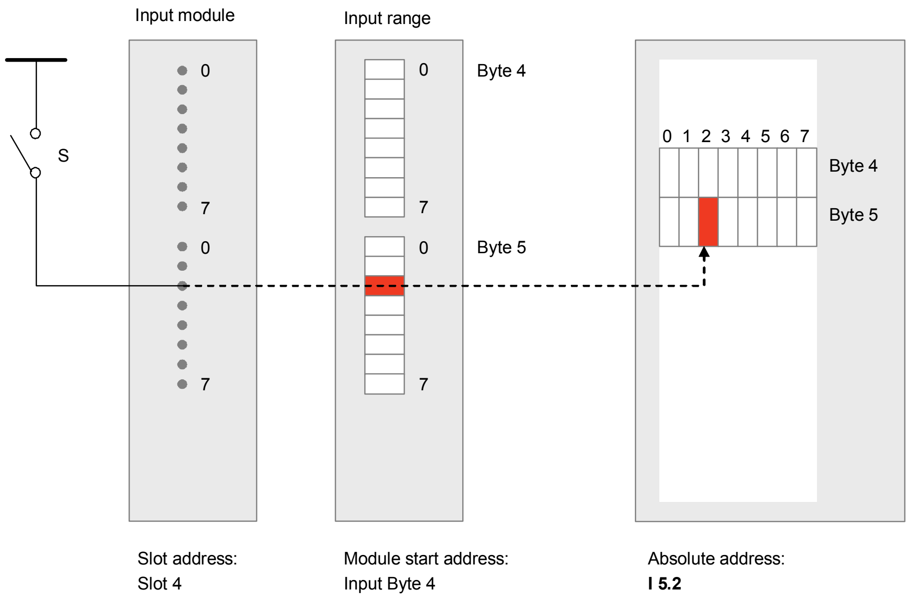


Figure 3 — Address indirection of I/O data

5.1.2 Analog sensors and actuators

The information transmission of analog sensors in factory automation is based on voltage ranges such as ± 80 mV, ± 2,5 V, ± 5 V, and ± 10 V as opposed to the 4…20 mA current loop generally used in process automation. "Voltage modulation" fits well to fast signaling, short distances, and "unlimited" power supply. "Current modulation" fits well to slow signaling, long distances, and explosion proof areas with power supply limitations.

The transformation of the analog sensor values into digital signals usually resulted in the "classic world" in a 16 bit coding as shown in Figure 4 independent of the resolution of an analog input module.

Resolution of a **15** Bit analog input module																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Resolution of a **12** Bit analog input module																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	SN	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	**0**	**0**	**0**

Resolution of an **8** Bit analog input module																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Figure 4 — Digital representation of analog values

Bit 15 (SN) represents the sign of the value. Usually the data type is Integer16.

Figure 5 is showing how complex analog sensors with additional binary switch signals can be connected to different types of standard digital (slot 3) and analog (slot 4) input modules of a remote I/O.

This can be done without integration conflicts for a PLC system and for a programmer to assign individual symbolic names to the variables (e.g. I3.6, I3.7, IW10). See also [7].

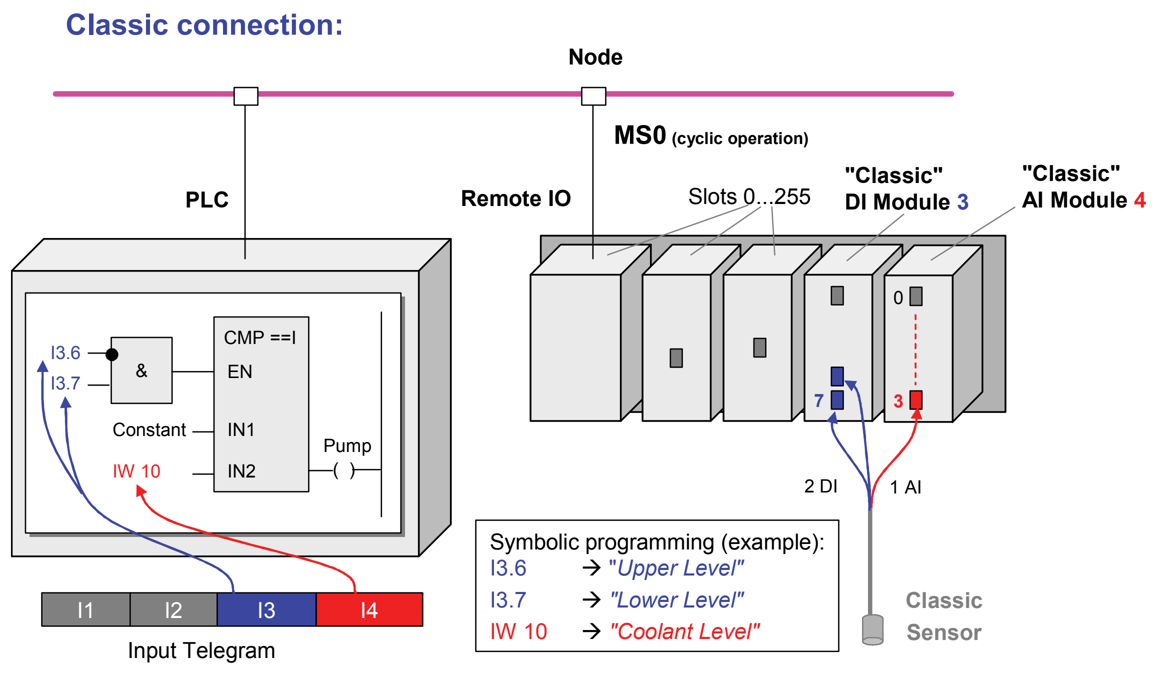


Figure 5 — Connecting complex analog sensors (example)

5.1.3 Considerations on data structures for IO-Link

IEC 61158-5:2003 defines in Clause 5 data types for all the fieldbusses incorporated in IEC 61158. An unambiguous data type class identifier is assigned to all of these data types across all of the fieldbusses. For PROFIBUS the valid subset of the defined data types is presented in Table 405 in Clause 6.2.5.6 of IEC 61158-6:2003. In the meantime some PROFIBUS application profiles defined their own special "constructed" data types to support checking and automatic assignment and invocation of device driver function blocks. The most current set of data types for PROFIBUS and PROFINET is officially released by PI in [4].

For IO-Link, the expected data structures to be conveyed to a programmable controller (PLC) hardly fit to the existing data type scheme of PROFIBUS and PROFINET. The IO-Link system constraints may lead to "packed" data structures as shown in Figure 6.

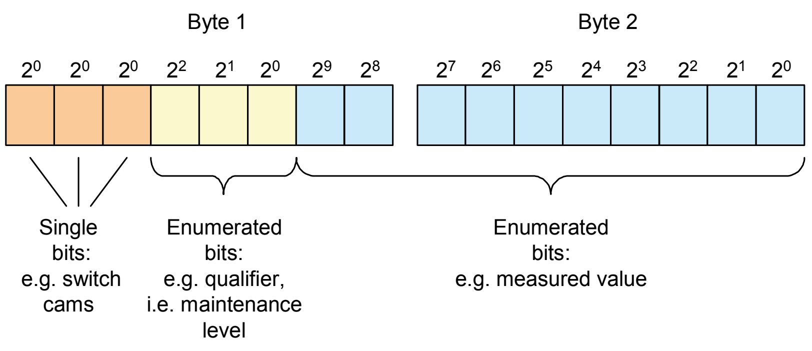

Byte 1
Byte 2
2^0 2^0 2^0 2^2 2^1 2^0 2^9 2^8
2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
Single bits: e.g. switch cams
Enumerated bits: e.g. qualifier, i.e. maintenance level
Enumerated bits: e.g. measured value

Figure 6 — Unfavorable I/O data structure of an IO-Link device

In the following it is shown how inappropriate layouts of these I/O structures may have negativ impact on system features, human interfaces, and user programming efforts.

The "packed" data structures can only be incorporated as a byte sequence in the processing data unit (PDU) of a PROFIBUS or PROFINET message. It is the responsibility of an IO-Link device designer to consider the byte order to be transmitted, which is defined in the IO-Link specification. In case of a modular remote I/O, a module can contribute a manufacturer specific total sum of bytes (e.g. 32), i.e. this is the limit for a so-called single slot IO-Link master (7.3.3). In case of a compact remote I/O, i.e. for a multi slot IO-Link master, this limit extends to *244 bytes* (7.3.3.1).

It is one of the objectives of this specification to provide appropriate means for the programmer

- to inform about the details of the data structures, the absolute addresses and the offsets,
- to support the effort of decomposing the structures into programm variables with symbolic names.

One precondition is the provision of corresponding IO-Link device information in electronic form (e.g. via IO-Link device description "IODD" or I&M functions).

The designer of IO-Link devices should take special care on the issues of "value qualifier" and "diagnosis information". Qualifiers should be used to intervene immediately in the automated system e.g. by switching to default process values instead of the measured values. Thus, qualifiers should be conveyed cyclically. In contrast, diagnosis information should only be conveyed acyclically when an error or failure occurred. The diagnosis mechanisms mainly address operators or the maintenance staff who should resolve the problem and thus need foreign language support. PI provides a guideline dealing especially with these issues [5].

An exception from this rule may be the case where only one particular light indicator shall be switched on in case of an undifferentiated fault.

An I/O data structure such as in Figure 6 causes a PLC programmer to implement a rather complex user program as he can only access it via the process image byte by byte, word by word, or bit by bit. The right aligned measured value would need to be shifted to the left, enumerated bits somewhere in between need to be picked as single bits. Assigning symbolic names is hard to achieve. Figure 7 is showing exemplary a more favorable design of the I/O data structure. The measured value is treated as one byte even though the least signficant bits are outside the resolution range. These are not important for the data processing within the user program and can easily be used for binary switching signals.

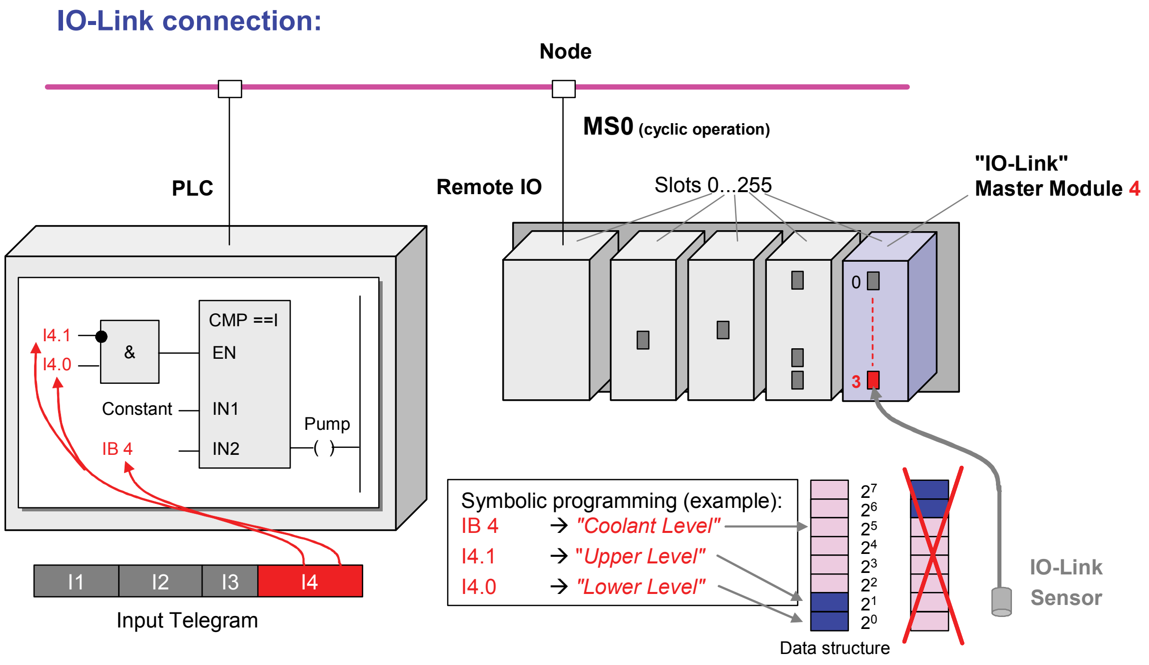


Figure 7 — Recommended I/O data structuring (example)

5.2 Concepts, components and constraints

In this clause the main architectures of the IO-Link master integration are developped along the PROFIBUS and PROFINET system constraints.

5.2.1 Basic requirements

There are two main types of fieldbus devices to be considered for the IO-Link master integration:

- modular devices such as remote I/O with classic digital and analog in/outputs
- compact devices solely designed to connect IO-Link devices to the fieldbus ("Gateway")

Both device types are following the same addressing schemes and communication basics (device model). The modular device is preferred by those users intending to deploy a mix of classic in/outputs and IO-Link devices and who enjoy the flexibility to migrate or switch as needed. However, this type has some restrictions due to the resource sharing with other modules on the local backplane bus.

In contrast the compact device ("Gateway") can use the entire resources of PROFIBUS and PROFINET such as the available PDU length and the mapping of ports to slots.

It is the goal of this specification to provide one common integration strategy for both the modular and compact devices in accordance with the current and long term technical policy of PI and the IO-Link consortium. Table 1 contains a collection of the most important decisions on the requirements.

Table 1 — Requirements for the PROFIBUS and PROFINET integration

Number	Requirement	Consequence	Remark
R1	Merging of IO-Link device descriptions ("IOD") into an existing GSD file of a fieldbus device shall be omitted.	Mandatory	PNO Advisory Board, 02-Mar-2007
R2	The concept of a "Port Configurator" in combination with the "iPar-Server" shall be specified.	Highly recommended	PNO Advisory Board, 02-Mar-2007
R3	The concept of a limited number (e.g. 10) of anonymous IO-Link device parameters for IO-Link master GSD or "Port Configurator" shall be specified.	Mandatory	IO-Link WG16
R4	The concept of a "Parameter Channel" within the cyclic fieldbus communication may be specified in an annex.	Not recommended for DP-V1 devices	PNO Advisory Board, 02-Mar-2007
R5	For modular remote I/O, the ports of an IO-Link master shall be mapped into one slot.	Highly recommended	

5.2.2 IO-Link master types

The possible IO-Link masters need further differentiation than just "master modules" and "compact masters".

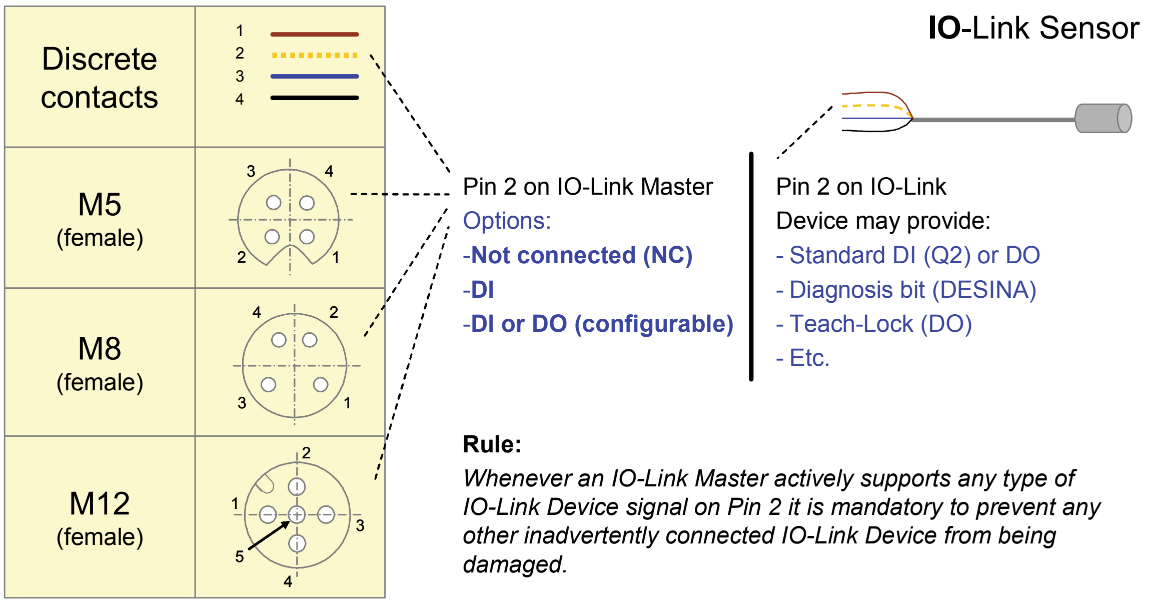


Figure 8 — Connection and operation possibilities for IO-Link master

Figure 8 is showing the entire connection possibilities for IO-Link masters. The IO-Link specification [8] defines several connection types for "compact masters" such as M5, M8, and M12. However, it disregards the normal discrete contacts for modules in remote I/O (IP20).

In order to prevent the vendors and the customers from damage to either devices due to unknown signals or wrong configuration or unexpected and unwanted device behavior during commissioning this specification defines several types of IO-Link masters and specifies their configuration possibilities and behavior.

Pin 4 of the connectors can be configured as a Digital Input (DI), Digital Output (DO), or an IO-Link Communication line. In case of IO-Link communication, the signal on Pin 4 can be switched (*Fallback* operational mode of the IOL-M) to DI or DO after an initial communication phase (e.g. parameterization).

Pin2 of the connectors can provide 3 options:

- Not connected (NC),
- Digital Input (DI) only, or
- Digital Input (DI) configurable to Digital Output (DO).

Table 2 is showing the electrical characteristics of the connector pin configurations.

Table 2 — Electrical characteristics of the connector pin configurations

Pin	Digital Input	Digital Output	SIO (Fallback)		Remarks
			DI	DO	
4	IEC 61131-2, *Type 1*, 24V DC	IEC 61131-2, 24V DC, *0.1A*, short-circuit protection	IEC 61131-2, *Type 1*, 24V DC	IEC 61131-2, 24V DC, *0.1A*, short-circuit protection	
2	IEC 61131-2, *Type 1*, 24V DC	IEC 61131-2, 24V DC, *0.5A*, short-circuit protection	-	-	Options: NC, DI, or DI and configurable DO

The connection of power actuators up to 3.5 A is shown in Figure 9. These actuators need separate 24V power supply, which can only be provided through Pin 2 and Pin 5. In case of an IO-Link master module in a remote I/O (IP20) the necessary provision of the 24V power supply can be provided independently from the IO-Link master module through appropriate terminals.

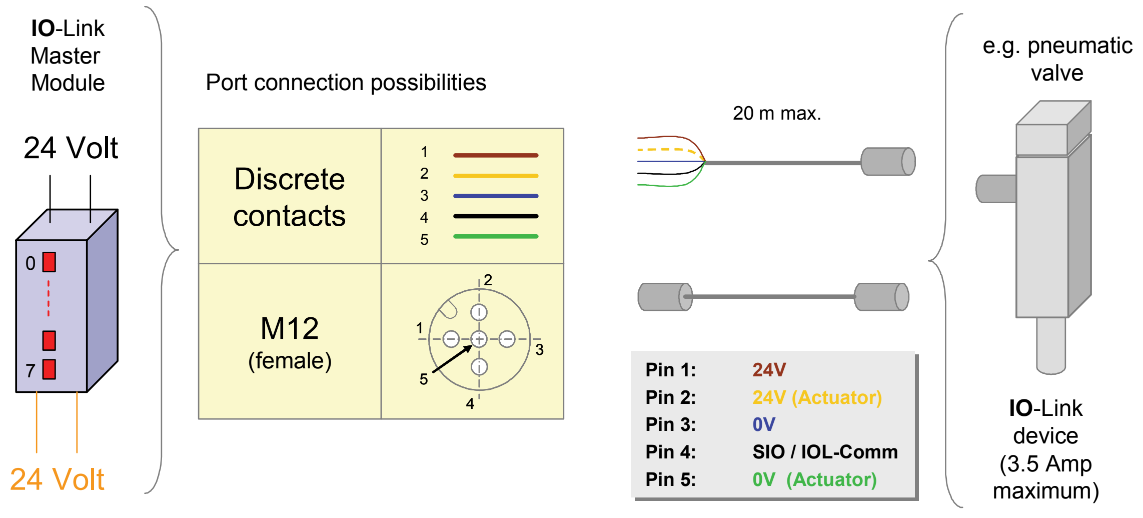


Figure 9 — IO-Link master types for power actuators

Table 3 specifies the characteristics of the three IO-Link master types A, B, and C.

Table 3 — IO-Link master type definitions (classes)

Master type	IP	Connection	Pin 2	Pin 5	Remark
Class A	67	M5, M8, M12	Options: NC, DI, or DI and configurable DO	M5, M8: not available; M12: not connected	Table 2
Class B	67	M12	+24V (separate)	0V (separate)	Figure 11
Class C	20	Discrete	Separate module	Separate module	Figure 10
NOTE Column IP (Ingress protection) represents typical values.					

Type C represents a typical IP20 IO-Link master module of a modular remote I/O with discrete wiring. Any IO-Link device signal on Pin 2 – be it DI or DO – can be served by separate DI or DO modules within the remote I/O. The identical IO-Link master module can be used to support power actuators, even though its ports are marked with the standard IO-Link logo (Figure 10). In this case Pin 2 is connected to a separate +24V source and Pin 5 to the associated 0V of this source via appropriate terminal blocks (Figure 10). For safety reasons de-energizing of the power actuator is possible via separate PROFIsafe power modules or discrete means.

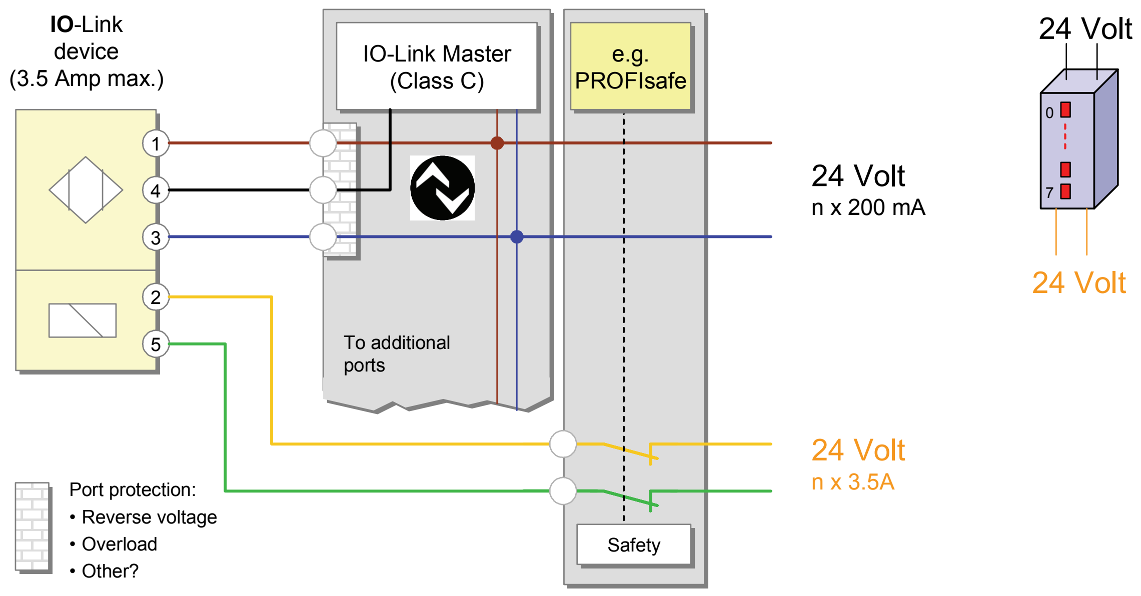


Figure 10 — IO-Link master module for power actuators (e.g. IP20)

Type B is represented by a typical IP67 IO-Link compact master with connector type M12 (Figure 11). Class B ports shall be marked in a significant manner, e.g. by a ring around the IO-Link symbol as shown in Figure 11. Pin 2 provides +24V of a separate power supply and Pin 5 the associated 0V of this power supply (Figure 11). Usually type B master will only provide the separate 24V power supply. However, configuration of Pin 2 similar to type A is possible if safety and protection constraints can be guaranteed. For functional safety reasons de-energizing of the power actuators is possible via separate or integrated safety means.

The wire colors of the IOL-D cables shall be according IEC 60947-5-2: pin 1 = brown (L+), pin 2 = white, pin 3 = blue (M), pin 4 = black. Pin 5 is not specified in IEC 60947-5-2. It is recommended to use the color gray or green. For better visibility the color "orange" instead of "white" is used in Figure 8, Figure 9, Figure 10, and Figure 11.

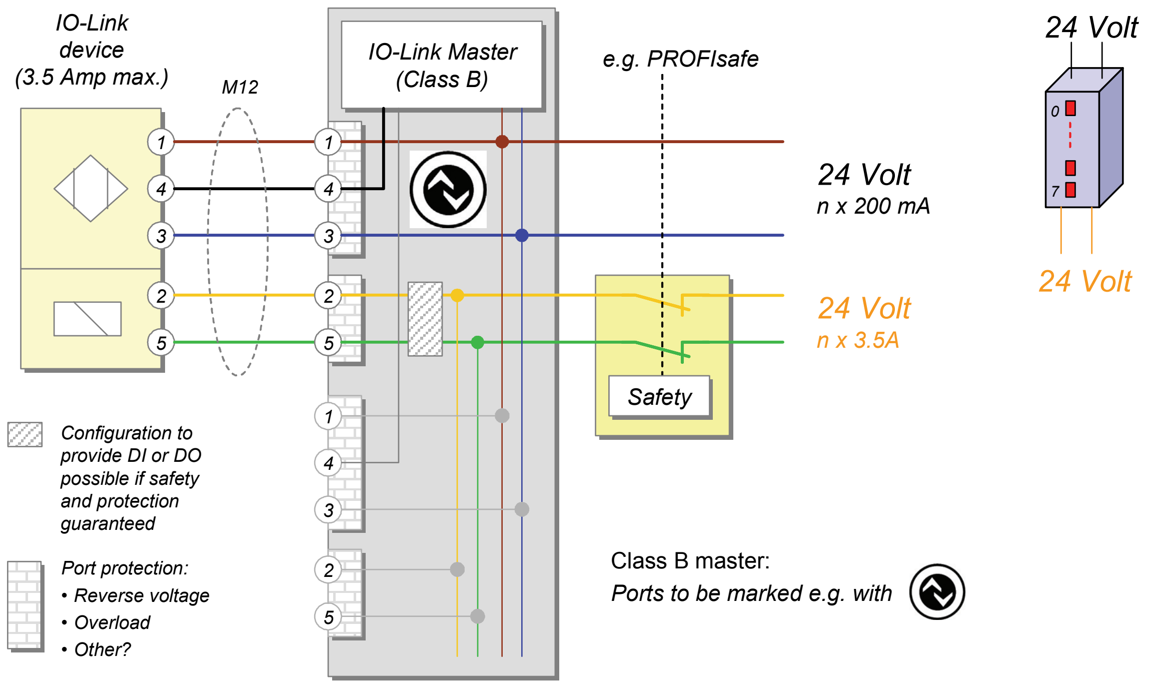


Figure 11 — IO-Link compact master for power actuators (e.g. IP67)

5.2.3 LED indications

IOL-CMs usually are incorporating the PB-DP or PN-IO interfaces. For these interfaces it is recommended to provide the LED indications specified in PROFIBUS Profile Guidelines [5]. For PN-IO these guidelines are in progress.

IOL-MMs in modular remote I/Os usually are one element out of several providing the integration of various communication types such as HART, AS-I, etc. It is therefore necessary to define common look and behaviors across the LED indications of all these types to avoid confusion of the users.

The IO-Link specification already defined LED colors and indications for IOL-D. For the ports of IOL-CM and IOL-MM it is recommended to use the following definitions:

- Two LEDs per port for Pin 4 and Pin 2
- One LED only if Pin 2 is "not available/not connected"
- If horizontally ordered: Pin 4 LED shall be positioned left from Pin 2 LED
- If vertically ordered: Pin 4 LED shall be positioned above Pin 2
- Recommended LED indications according Table 4

Table 4 — Recommended LED indications for IO-Link master port signals

Signal	DI mode	DO mode	IO-Link communication or SIO mode	
Pin 4 "Channel 1"	🟩 Green (on = 1) ■ Dark (off = 0)	🟩 Green (on = 1) ■ Dark (off = 0)	🟩 Green 2.0Hz	(Signaling startup and parameterization phase, blinking duration >= 3 sec)
			🟩 Green	(Data exchange ok or in SIO mode "on")
			■ Dark	(Not connected or in SIO mode "off")
			🟥 Red 0.5Hz	(Fault; blinking due to red-color-blind persons; 1s on, 1s off)
Pin 2 (opt.)	🟩 Green (on = 1)	🟩 Green (on = 1)	Not defined for non-safety devices	

Signal	DI mode	DO mode	IO-Link communication or SIO mode
"Channel 2"	■ Dark (off = 0)	■ Dark (off = 0)	
NOTE Instead of the color green 🟩 the color yellow 🟨 can be used optionally			

5.2.4 Integration components and concepts

A user shall be able to use both IO-Link master modules and classic digital and analog I/O modules as a mix in the same remote I/O. This constellation is the most ambitious of all and thus it is used as the main model for the integration. All the other configurations can easily be derived from this main model. The already mentioned maximum of process I/O data per module or IO-Link master (5.1.3), the degree of hardware integration of the intended *microcontroller*, the *total power consumption* of all the connected IO-Link devices of *n x 200 mA* and the *available space* and *heat dissipation* may be some of the more important challenges for the developer. Figure 7 is showing how an IO-Link master module is replacing classic I/O modules.

In the easiest case all the connected IO-Link devices are sensors using the SIO mode, i.e. the process data per sensor is just one bit which leads to the same configuration like in Figure 2 and to no changes in the user program.

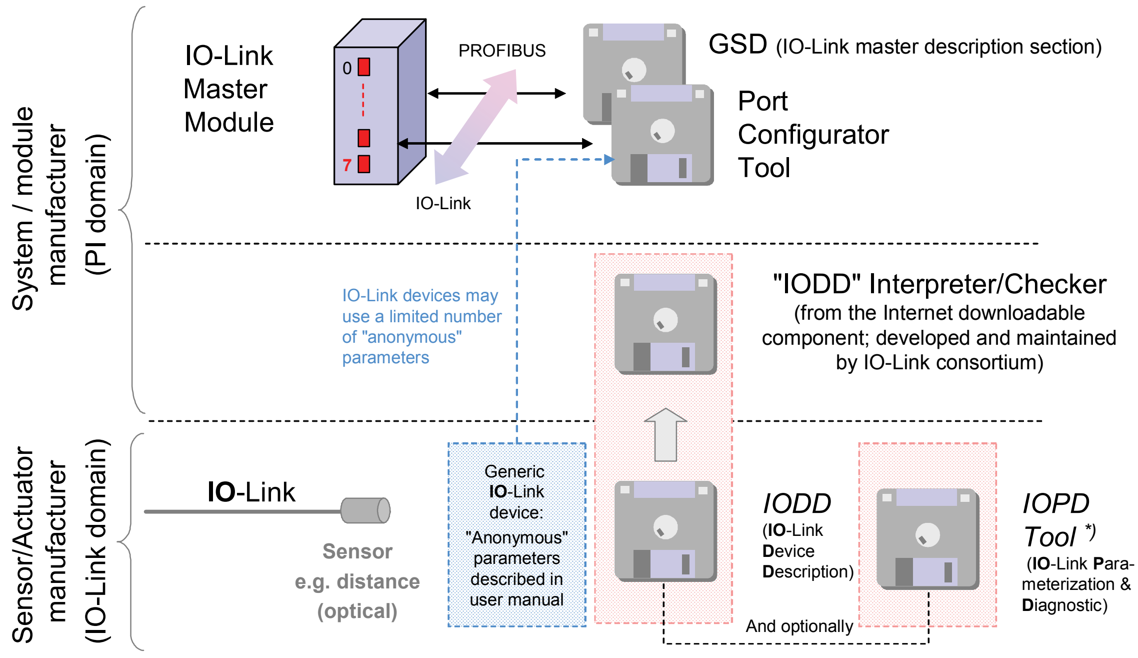

System / module manufacturer (PI domain)
Sensor/Actuator manufacturer (IO-Link domain)
IO-Link Master Module
0
7
PROFIBUS
IO-Link
GSD (IO-Link master description section)
Port Configurator Tool
IO-Link devices may use a limited number of "anonymous" parameters
"IODD" Interpreter/Checker (from the Internet downloadable component; developed and maintained by IO-Link consortium)
IO-Link
Sensor e.g. distance (optical)
Generic IO-Link device: "Anonymous" parameters described in user manual
IODD (IO-Link Device Description)
And optionally
IOPD Tool *) (IO-Link Parameterization & Diagnostic)

Figure 12 — IO-Link integration components

More sophisticated IO-Link devices with data structures such as in Figure 6 will lead to rather complex data structures "packed" by the IO-Link master. This specification defines "packing rules" for the IO-Link master.

Figure 12 is presenting the necessary components for the IO-Link integration. It starts with the IO-Link master module itself which needs a description section in the overall GSD file of a remote I/O device. As the IO-Link device parameters are not permitted to be merged into the overall GSD file a special *Port Configurator Tool (PCT)* takes over the tasks of configuring the ports and parameterizing the IO-Link devices with the help of user manual information (maximum of 10 "anonymous" parameters) or electronic *IO-Link device description (IODD).* The PCT reads the IODD with the help of an *IODD interpreter/checker*, allows the user to assign the desired values to the parameters, and performs up and downloads to the IO-Link devices.

The *IODD interpreter/checker* component shall ensure

- conformance of a particular IODD with a valid version of an IODML schema
- no corruption of the PCT by invalid IODD
- support of all valid versions of IODML icluding the most actual version

One solution for these requirements could be an IODD Interpreter/checker component that is downloadable from the IO-Link webserver. It is developped and maintained by the IO-Link consortium.

In addition it is possible for IO-Link device manufacturers to provide *IO-Link parameterization and diagnosis tools* (IOPD) facilitating the interactions with their particular IO-Link device for example via wizzards.

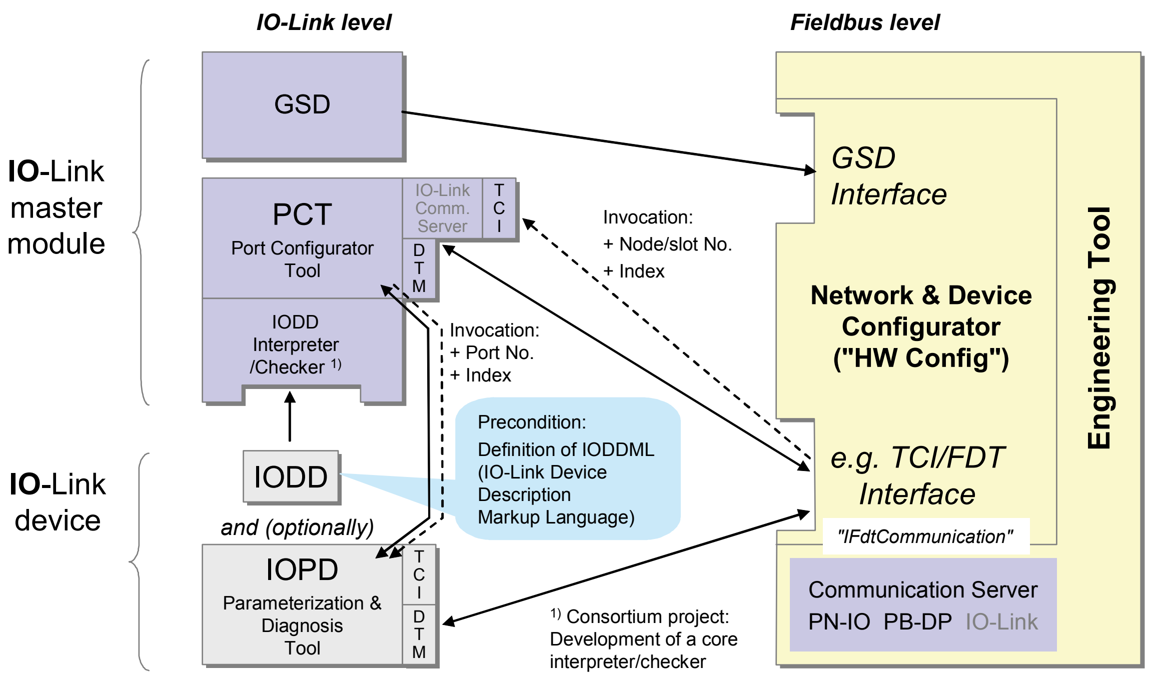

IO-Link level
Fieldbus level
IO-Link master module
IO-Link device
GSD
PCT
Port Configurator Tool
IO-Link Comm. Server
TCI
DTM
IODD Interpreter /Checker 1)
IODD
and (optionally)
IOPD
Parameterization & Diagnosis Tool
TCI
DTM
Invocation: + Node/slot No. + Index
Invocation: + Port No. + Index
Precondition: Definition of IODDML (IO-Link Device Description Markup Language)
1) Consortium project: Development of a core interpreter/checker
GSD Interface
Network & Device Configurator ("HW Config")
e.g. TCI/FDT Interface
"IFdtCommunication"
Communication Server
PN-IO PB-DP IO-Link
Engineering Tool

Figure 13 — System interfaces for the integration components

Figure 13 is showing the system interfaces for the integration components. The GSD file is using the existing interface of an engineering tool.

The PCT is using one of the standard fieldbus engineering interfaces such as TCI [11] or FDT/DTM [12] in case of PROFIBUS or PROFINET. IODDs are installed in a particular path of a programming device from where the integrated IODD interpreter / checker of the PCT reads the data into an internal database for further use. The optional additional IOPD tool may use the same system interfaces TCI or FDT/DTM as the PCT.

NOTE TCI and FDT/DTM are different solutions for tool integration. However, both are using the same communication interface "IFdtCommunication". The extension of this interface for the access of IO-Link devices is currently being developed as an annex by the FDT group (Figure 1). The corresponding specification is related to this specification [19]. While FDT is using the IO-Link communication server as a separate component in an open context, TCI is using an integrated approach within the PCT.

Both, TCI and FDT/DTM are supporting the storage of device instance data within a project database.

5.2.5 Configuration steps using PCT

It is necessary for an IO-Link master module to go into default data exchange right after it is plugged into the remote I/O and after switching on the power supply. This shall be possible just by using the GSD section describing the IO-Link master module.

Figure 14 is depicting the configuration steps of an IO-Link master module. In step ① the entire start-up parameterization block (Prm_telegram) is generated by the engineering tool with the help of the GSD file data. The default configuration of an IO-Link master module for each and every port is the SIO input mode (1 bit per port) i.e. for example one byte of process input data for 8 ports. In this phase the IO-Link master module is already able to support classic switching input devices. In step ② the PCT is connected to the IO-Link master module and takes care of the further configuration of the ports and parameterization of the IO-Link devices.

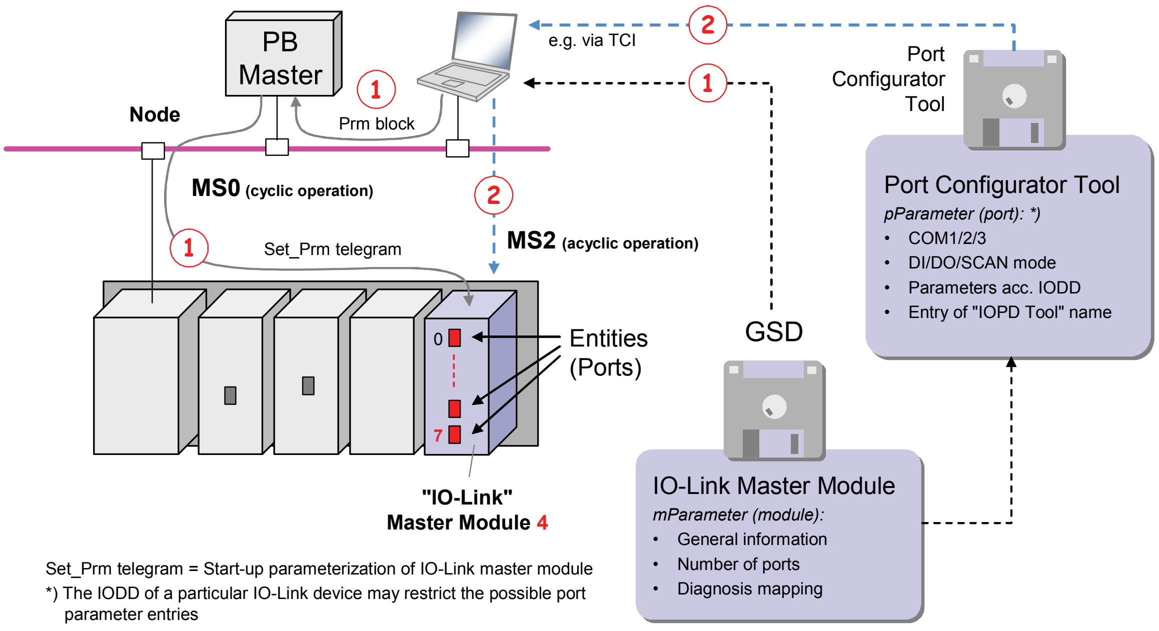


Figure 14 — Configuration steps

5.2.6 Cyclic and acyclic communications

Normally, the term "MS0 services" for cyclic data exchange used so far in the specification is not causing any comprehension problems. Figure 14 introduces an additional term "MS2" for acyclic communication.

Figure 15 demonstrates how the additional services MS1 and MS2 are operating. Normally, the bus cycle is divided into two time frames. In the first time frame a master class 1, for example inside a PLC uses the MS0 services for the fast cyclic exchange of process data with its associated slaves. The second time frame is a configured "spare time" and is used for acyclic access. Within this time the master class 1 is able to access one of its slaves when needed, for example to change parameters of a slave device.

During the following cycles, a master class 2, for example a PC can also access one of the slaves if needed after it received a token from its master class 1. For example to read out detailed diagnosis information. This procedure optimizes the efficiency of cyclic and acyclic communications.

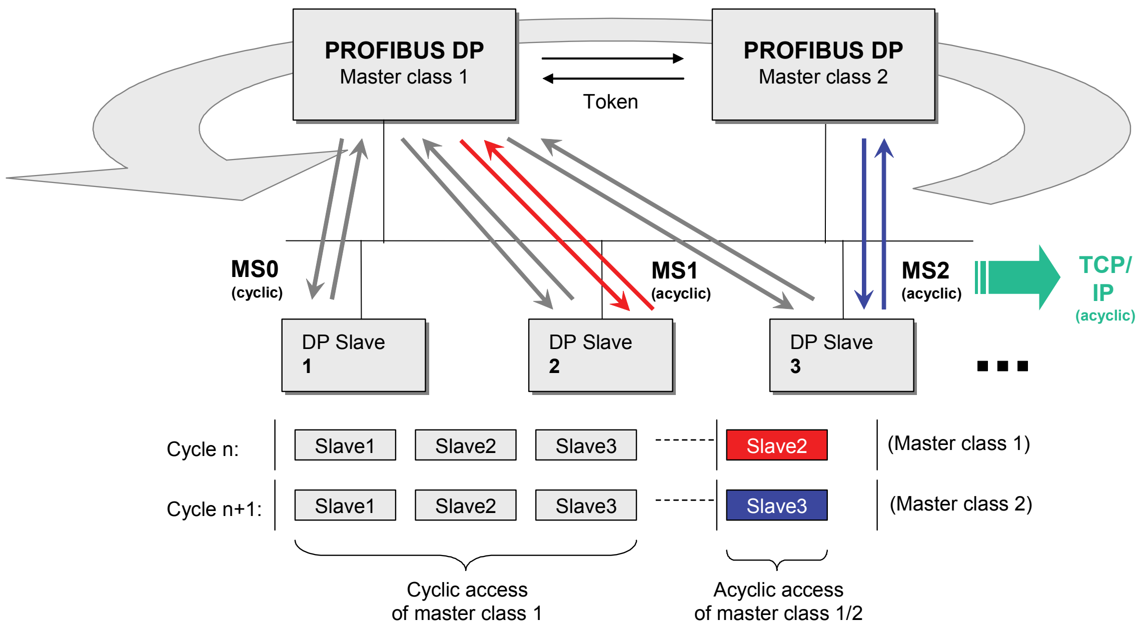


Figure 15 — Basics of cyclic and acyclic PROFIBUS communications

5.2.7 Acyclic port access via CALL mechanisms

In order for the PCT to communicate with a particular IO-Link device the acyclic CALL mechanism is used with its possibility to address the ports of an IO-Link master via "Entities". Figure 16 is showing the principle.

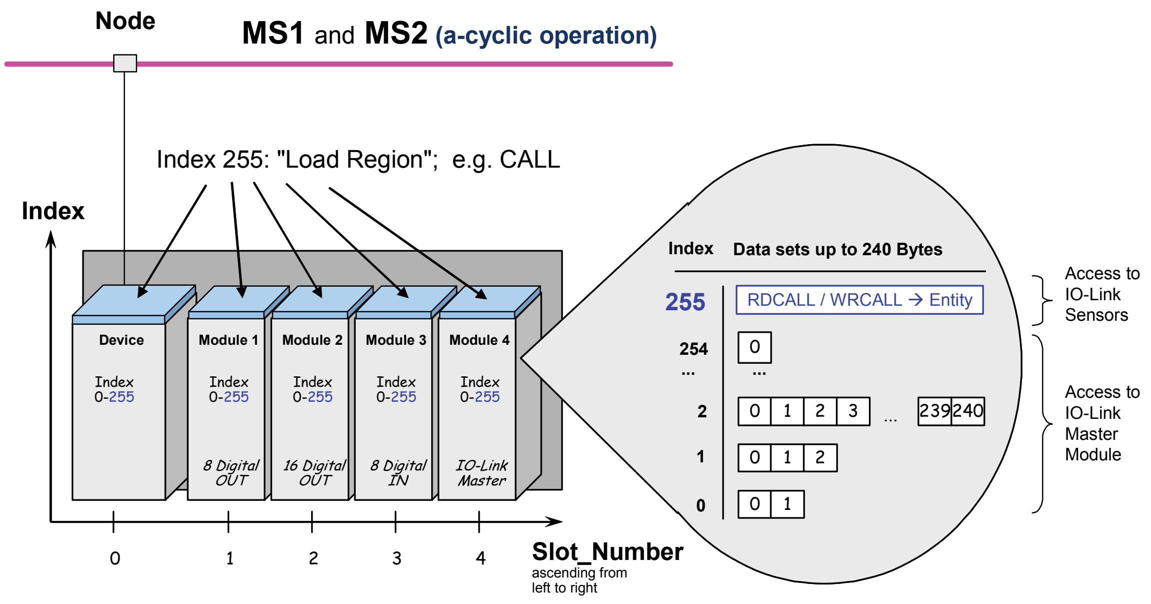


Figure 16 — Addressing schema of the CALL mechanism in principle

IEC 61158-6-3 (CPF3) specifies acyclic stateless read and write access to the index 255 (RDCALL, WRCALL). This CALL mechanism defines an additional address space, so-called "Entities" (0…255), which can be used to address channels or in case of IO-Link the ports.

Figure 17 is presenting exemplary the header structure of a specialized IOL_CALL.

Content of an IOL_CALL	Size	Coding	Notes	
Function_Num	1 Octet	5Fh	Indicates "write", fix	DP-V1 Header
Slot_Number	1 Octet	0 … 255	variable	
Index	1 Octet	255	fix	
Length (of net data)	1 Octet	0 … 240	Length of data set (with body)	
Extended_Function_Num	1 Octet	08h	Indicates "CALL", fix	Call Header
Entity_Number	1 Octet	0 … 63 *)	IO-Link-Port	
FI_Index	2 Octets	65098	IO-Link profile specific	
IO-Link specific extensions	236 Octets	IO-Link	Extensions comprise an IOL header with "State", "IOL_Index", and "IOL_Subindex"	Body

*) Due to the maximum number of channels with diagnosis According to IEC 61158-6-3

Figure 17 — Header structure of an IO-Link CALL (IOL_CALL)

This specification defines a method on how to access the IOL_Index and the IOL_Subindex of an IO-Link device connected to a particular port ("entity") and an additional method on how to access ports simultanuously without conflicts (9.2). The principle uses I&M information of the IO-Link master which defines individual indices per port instead of index 255 ("redirection"). Index 255 is only used for the access of the I&M information.

5.2.8 Example of a PCT user interface

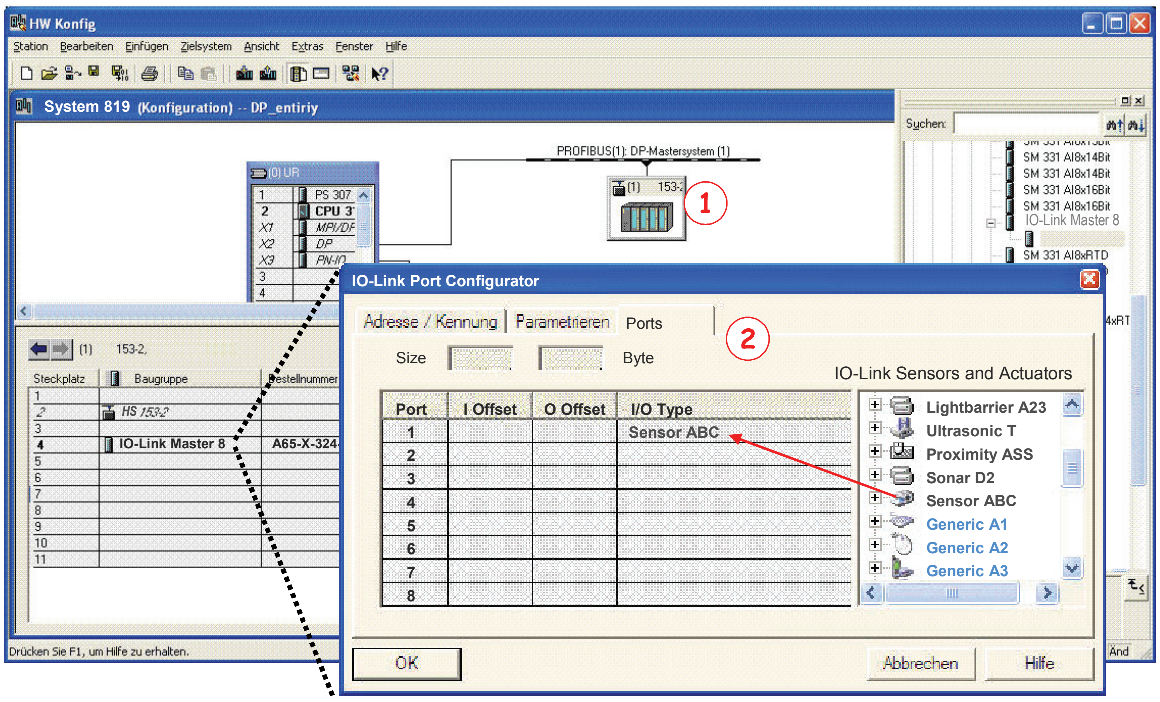
The architecture discribed so far is splitting the IO-Link and the PROFIBUS/ PROFINET configuration into two nearly independent levels. It will be the task of the user interfaces of the engineering tool and the PCT to keep the user within a clear paradigm on which level he/she is operating.

The upside of this strict separation is that there is minimal impact on the existing fieldbus mechanisms (valid not only for PROFIBUS and PROFINET but for other fieldbusses as well) and that it allows the entire features of IO-Link to flourish. It is not forcing two independent technologies to map information into one another and thus causing confusion for the user.

The downside is that the IO-Link technology cannot be mapped into the same configuration level as the classic I/O devices.

In order to visualize how the two levels may present themselves to the user this clause contains some examples of humble human interfaces. Guidance for the layout and human interface design can be acquired from [17].

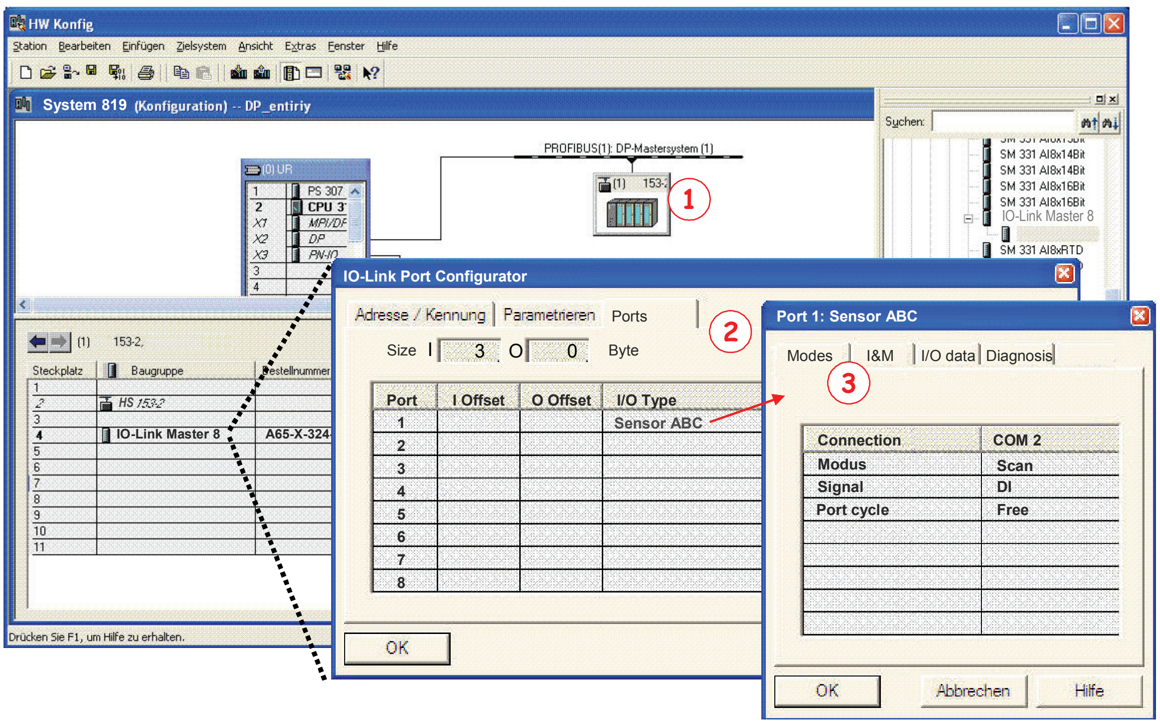
Figure 18 shows a PROFIBUS configuration window where an IO-Link master module has been taken from a module library on the left side and placed into a remote I/O. With the help of a right mouse click it is possible to open the PCT window. The IO-Link master shows here the available ports and the library of installed IO-Link devices (IODD). The user is able to pick a device from the library and to place it on any port. Two additional display fields are presenting the necessary number of input and ouput bytes within the PROFIBUS or PROFINET PDU.



Port	I Offset	O Offset	I/O Type
1			Sensor ABC
2			
3			
4			
5			
6			
7			
8			

Figure 18 — Example of a PROFIBUS configuration and PCT

After placing the particular IO-Link device onto a port, a sub-window can be opened for that particular IO-Link device. Figure 19 is depicting the sub-window.


HW Konfig
Station Bearbeiten Einfügen Zielsystem Ansicht Extras Fenster Hilfe
System 819 (Konfiguration) -- DP_entiriy
PROFIBUS(1): DP-Mastersystem (1)
1
2
3
IO-Link Port Configurator
Adresse / Kennung
Parametrieren
Ports
Size I 3 O 0 Byte
OK
Port 1: Sensor ABC
Modes
I&M
I/O data
Diagnosis
OK
Abbrechen
Hilfe
Drücken Sie F1, um Hilfe zu erhalten.

Port	I Offset	O Offset	I/O Type
1			Sensor ABC
2			
3			
4			
5			
6			
7			
8			

Connection	COM 2
Modus	Scan
Signal	DI
Port cycle	Free

Figure 19 — Example of a PCT port sub-window

Within this sub-window the following information may be presented and entries and modifications enabled:

- Port configuration (IODD defines the permitted port features of the IOL-D)

- I&M functions of the IO-Link device (provided by the IOL-D)
- Structure of the I/O data to support the programmer (provided by IODD)
- Parameterization of the IOL-D (individual or anonymous)
- Display of the current I/O process values (representation supported by IODD)
- Other manufacturer specific services (optional)

For those generic IO-Link devices without IODD or IOPD expecting entry capabilities for the 10 "Anonymous Parameters" the PCT provides a "Generic device" in the library that can be placed into a port field. The corresponding PCT sub-window in Figure 20 allows the entry of data sheet parameters via values in hexadecimal form. PCT uses a generic IODD in this case.

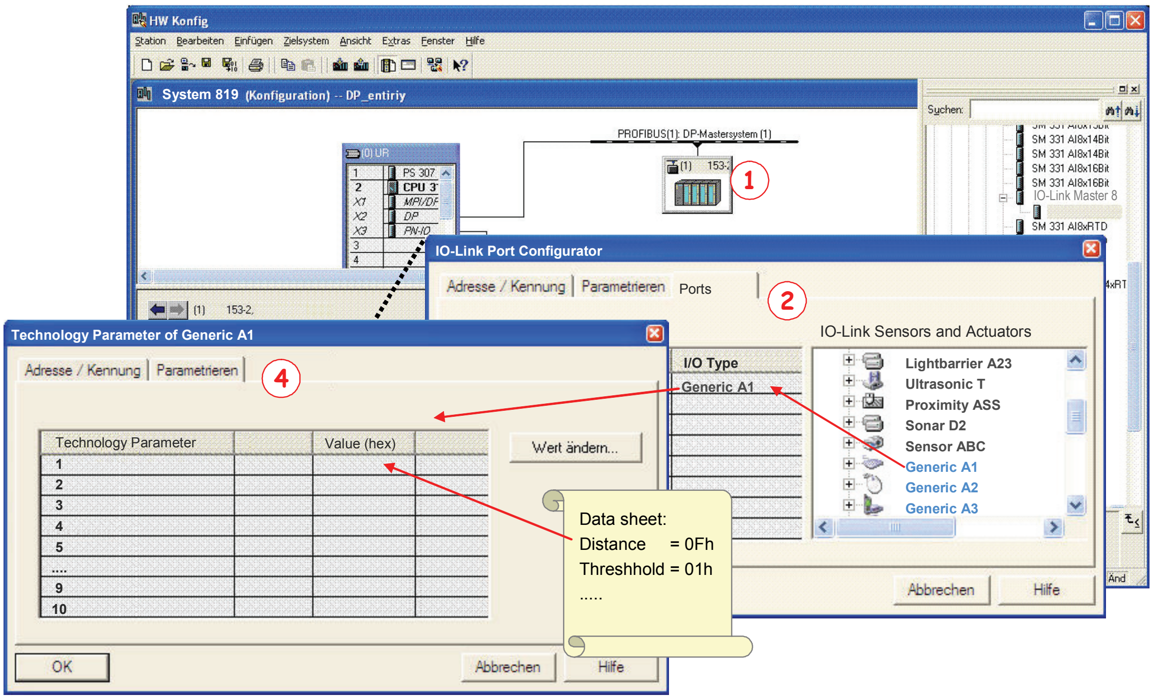


Figure 20 — Sub-window for the entry of "Anonymous Parameters"

5.2.9 Assigning the size of the I/O process data configuration

One major obstacle for integration concepts is the data exchange between the two configuration levels. PROFIBUS needs information about the I/O data structure for the Chk_Cfg-telegram (Check Configuration). The size of the final I/O data structure is only known after the configuration of the ports via PCT. This information has to be assigned manually from the display fields in a PCT window to the IO-Link master in the configuration window of an engineering tool as there may not be a standardized fieldbus related way of electronic data exchange between a PCT and an engineering tool.

The GSD section of the IO-Link master provides a reasonable selection of possible I/O structures in bytes such as 8/2, 32/8, etc. This specification contains a proposal for such a selection. The data types and their numeric identifiers are not relevant here and they are not included in the Chk_Cfg-telegram.

5.2.10 User program controlled parameterization

Modern production processes in factory automation are characterized by the flexibility to change between different production phases. The parameters of IO-Link devices need to be changed rapidly from one phase to the other by program-controlled parameterization. Usually the parameters comprise setpoint values (e.g. distance, level) and units (e.g. °C or °F, cm or inch). This specification provides two methods to put in place such functionality:

- "Parameter Channel" as add-on to the cyclically exchanged process data, or
- Standard "Comm-FB" using acyclic communcation.

Figure 21 is showing the realization of the "Parameter Channel" and its design implications. 8 additional bytes (or more) are accompanying the input and output data structures of an IO-Link master. The "Parameter Channel" ① is using the cyclic communication services MS0 of PROFIBUS and PROFINET. Its protocol is implemented in the IO-Link master and on the controller side through a special ParChannel-FB, which needs to be supported by the IO-Link master manufacturers for the different programmable controllers on the market.

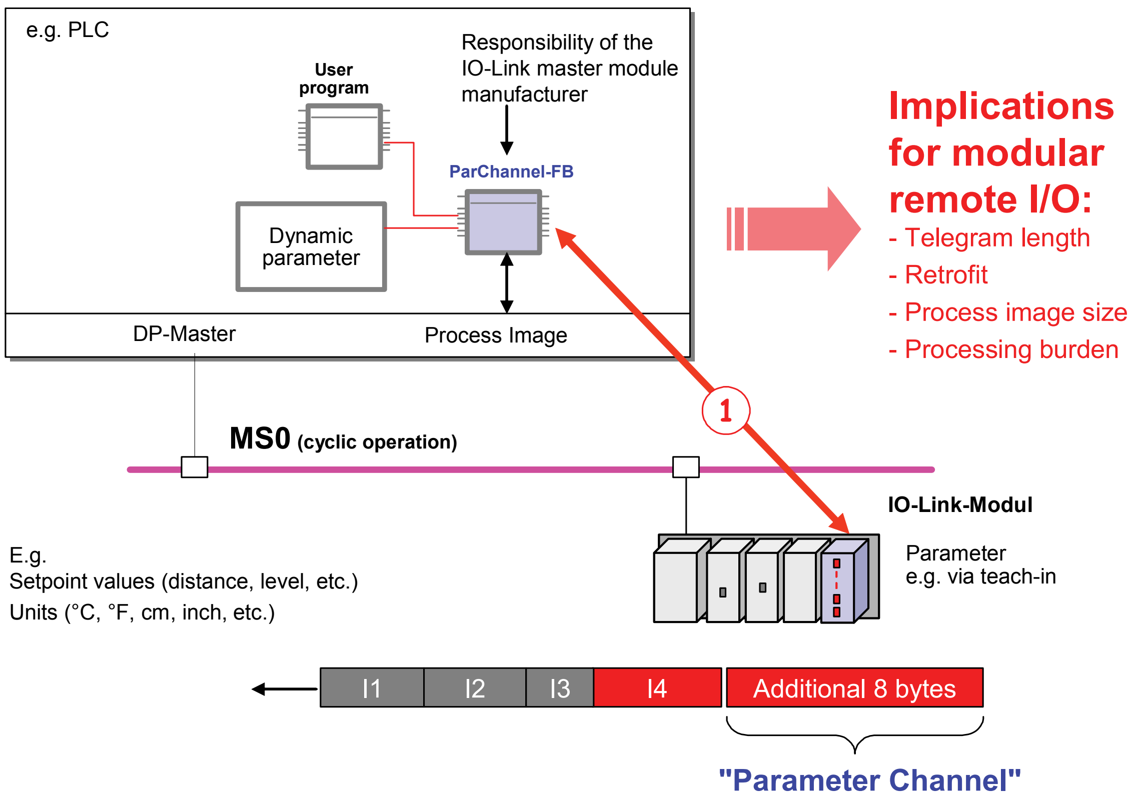


Figure 21 — The cyclically operating "Parameter Channel"

Figure 22 is showing the realization via standard "Comm-FB" according to [6] and the CALL access method described in 5.2.7. These IOL_CALL, RDREC, and WRREC function blocks are using acyclic communication services MS1 of PROFIBUS or similar services of PROFINET as a precondition.

The IOL_CALL function blocks are implicitely invoking RDREC and WRREC function blocks specified in [6]. It is the responsibility of system manufacturers (PLC) to provide and support these communication function blocks. The IOL_CALL function block can be seen as a shell around the RDREC and WRREC function blocks providing the CALL headers and the IO-Link headers.

Working group 4 in technical committee 2 of the PNO is responsible for specifying general purpose IOL_CALL function blocks that can be used by other profile groups also.

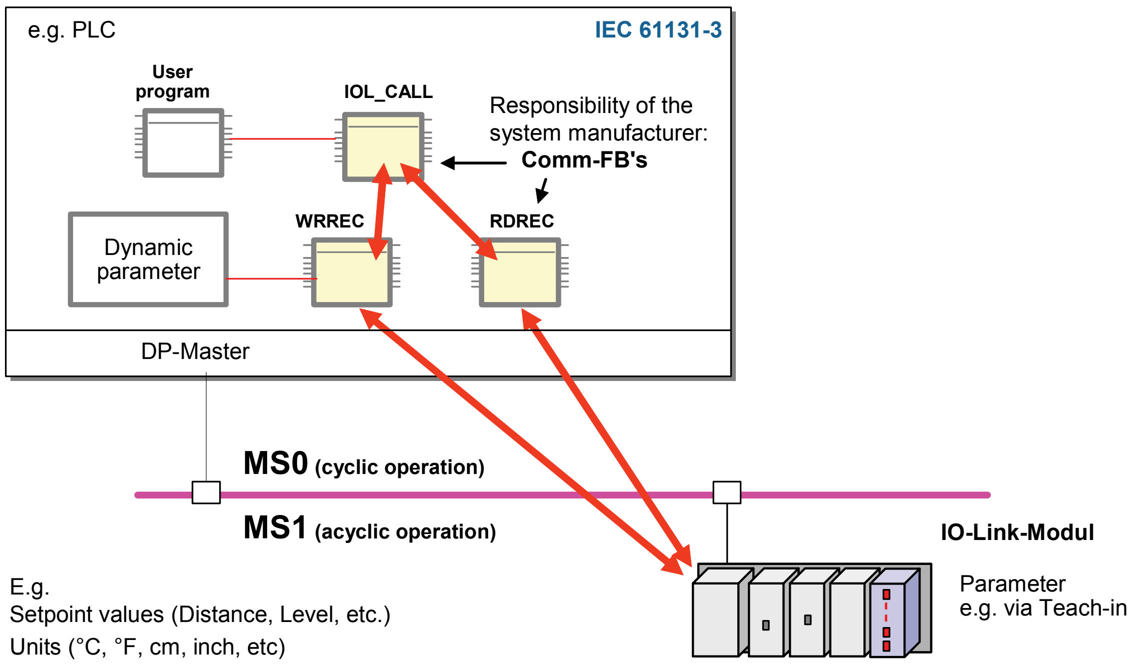


Figure 22 — The acyclically operating "Comm-FB"

Figure 23 is presenting the outline of an IOL_CALL function block using WRREC and RDREC function blocks. The extensions are depicted in blue color.

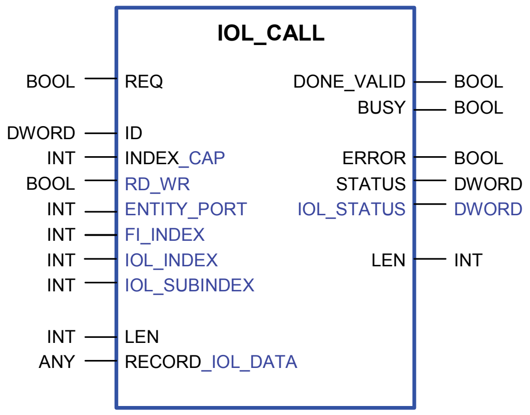


Figure 23 — Specialized IOL_CALL FB for IO-Link access

ERROR indicates any IOL_CALL error. For STATUS see [6], IOL_STATUS see 9.3.6.2.

5.2.11 The parameter server (iPar)

The IO-Link master is dealing with the parameters of the IO-Link devices according to the *IO-Link System Integration V0.9* specification and keeps a record of the entire parameters. For the quick replacement of the IO-Link master module in case of failure without the need of tools to restore the entire parameters, it is highly recommended to establish a new mechanism, the so-called iPar-Server. It is the responsibility of system manufacturers to provide this capability, be it realized within a PLC or a controlled subsystem such as an industrial computer on the same network.

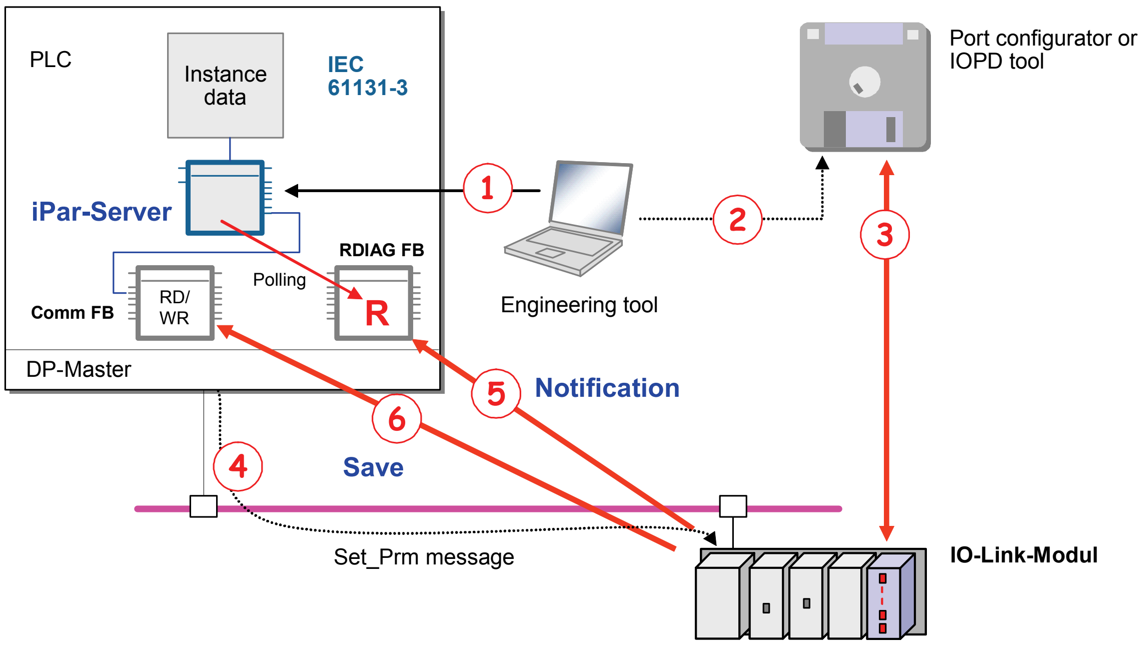


Figure 24 — The iPar Server

Figure 24 demonstrates the principle steps of the iPar-Server mechanism via an example.

Together with the network configuration of an IO-Link master module, an associated iPar-Server function is instantiated (1). The PCT can be launched via an appropriate interface (2) such as TCI (Tool Calling Interface) or FDT (Field Device Tool) from the engineering tool, propagating at least the node address and the slot of the configured IO-Link master module. Parameterization, commissioning, testing, etc., can be executed with the help of the PCT or separate IOPD tools (3). After final verification and release (4), the IO-Link master module is enabled to initiate an upload notification (5) to its iPar-Server instance. It thereby utilizes the standard diagnosis mechanism. The iPar-Server polls the diagnosis information (e.g. RDIAG FB) to interpret the request (R) and to establish the upload process (6), storing the iParameters as instance data within the iPar-Server host using acyclic services (Read Record).

After the replacement of a defective IO-Link master module, the new device initiates a download notification to its iPar-Server instance, again using the standard diagnosis mechanism. The iPar-Server polls the diagnosis information to interpret the request (R) and to establish the download process (Write Record). Through this transfer the IO-Link master module is enabled to provide the original functionality without the PCT or further IOPD tools.

5.2.12 Diagnosis

Figure 25 presents an overview on all the diagnosis methods for PROFIBUS DP specified in the IEC 61158 standards [13], and [15]. Additional conceptual details can be retrieved from [5].

With the advent of PROFIBUS DP-V1, the original Device-Related-Diagnosis has been split into two different methods: the ALARM model and the STATUS model. The ALARM model requests an acknowledgement by the user application prior to further diagnosis messages of a slave. The STATUS model allows a slave to send any diagnosis messages even when the master was not able to process the previous diagnosis messages. For PROFIBUS DP it is not mandatory to use the ALARM model.

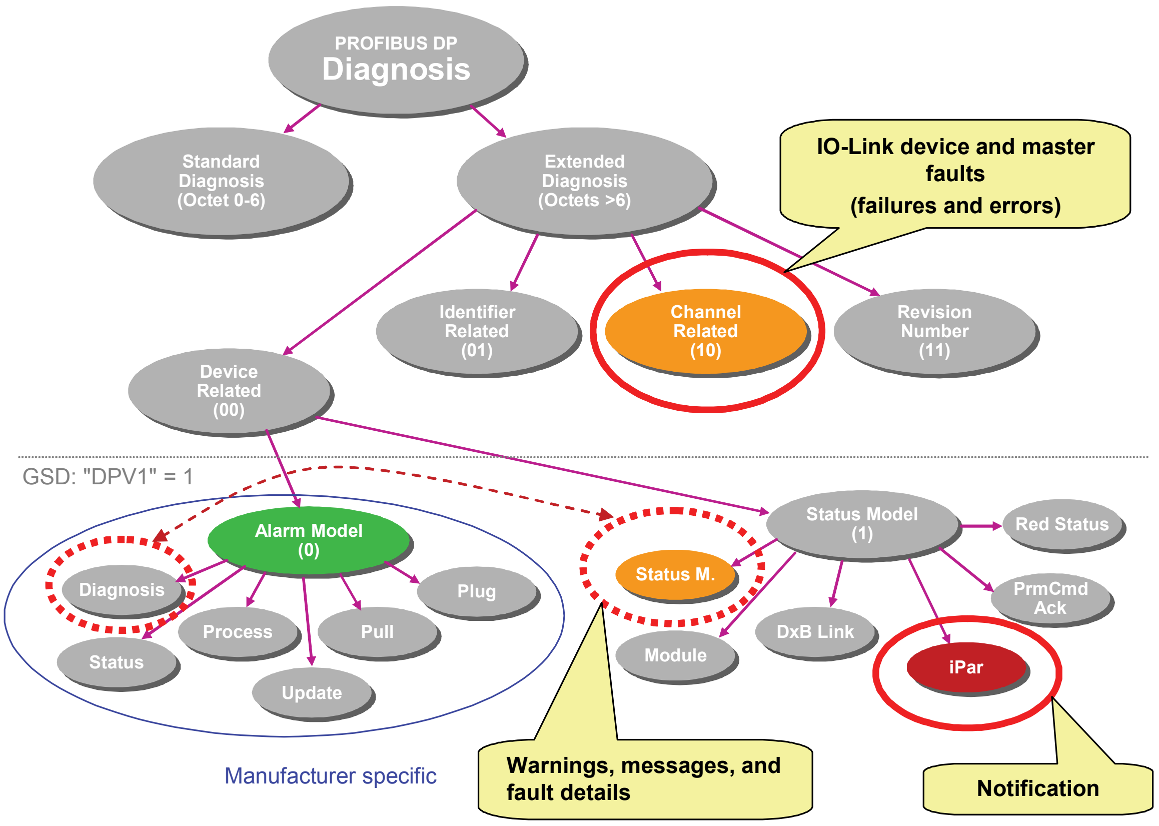


Figure 25 — Diagnosis methods of PROFIBUS DP

Thus, the IO-Link master integration focuses on

- *Channel-Related-Diagnosis* (C-R-D), mainly for IO-Link device and master faults
- *Status Message*, preferred for warnings, messages and fault details
- *iPar notification* for the parameter server mechanism (iPar server)

The Status Message provides a distinct possibility for the error reports of PROFIBUS devices such as IO-Link compact masters. The individual codes can be defined in the GSD file together with its associated error text and help text in several languages [1], and [5]. The coding of *Diagnosis Alarms* and *Status Messages* correspond to each other (dotted arrow in Figure 25).

Figure 26 presents an overview on all the diagnosis methods for PROFINET IO specified in the IEC 61158 standards [14], and [16]. PROFINET IO took over the ALARM model as a basic diagnosis reporting method. Thus, the C-R-D now follows the ALARM model also as well as the iPar notification. Status messages should be mapped into diagnosis alarms.

The definitions of the Channel-Related-Diagnosis are the same as with PROFIBUS DP and are now extended. Table 5 is showing the list of C-R-D codes. Code 1 to 10 is defined for PB-DP and PN-IO with the identical semantics. Codes 15 and above are specified in PB-DP as "manufacturer specific". This is the same with PN-IO. However, for PN-IO recommended semantics are specified in addition and should be used whenever possible. Visualization systems will display corresponding diagnosis text in any chosen language without definitions in the GSD file. Otherwise, individual GSD assignments are overwriting the default system assignments.

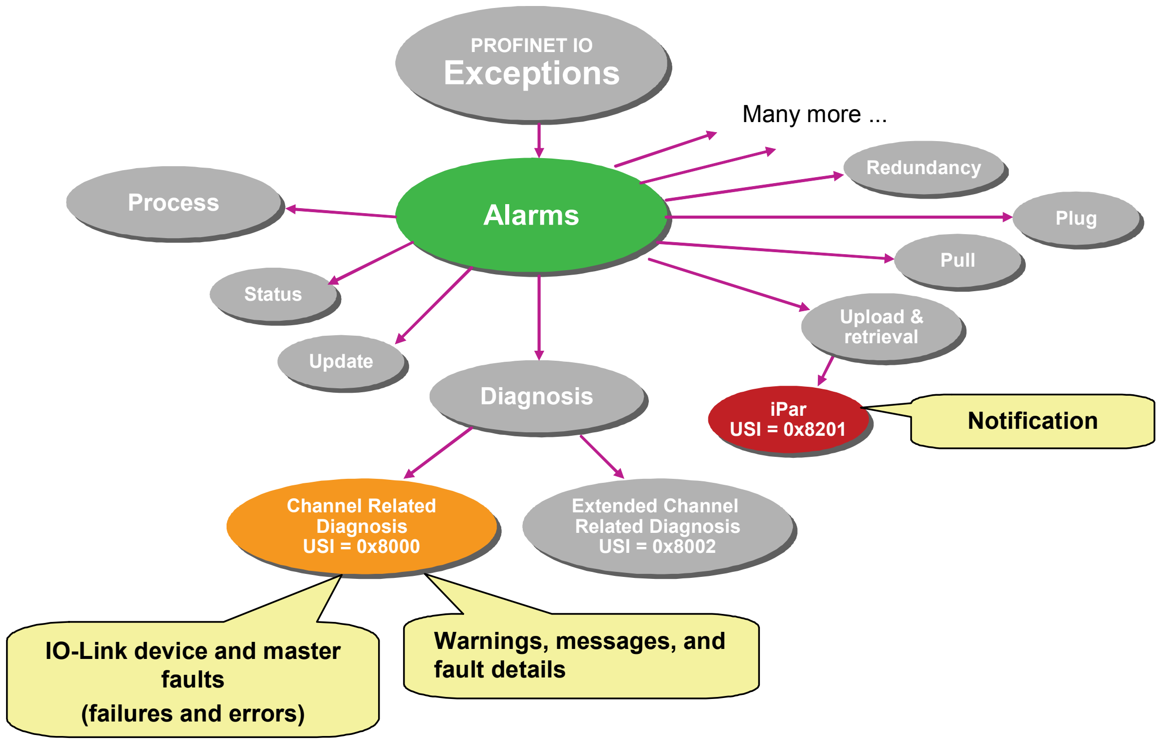

PROFINET IO
Exceptions
Many more ...
Redundancy
Process
Alarms
Plug
Pull
Status
Upload & retrieval
Update
Diagnosis
iPar USI = 0x8201
Notification
Channel Related Diagnosis USI = 0x8000
Extended Channel Related Diagnosis USI = 0x8002
IO-Link device and master faults (failures and errors)
Warnings, messages, and fault details

Figure 26 — Diagnosis methods of PROFINET IO

C-R-D codes can be mapped to specific error codes of the IO-Link devices whenever an unambiguous association is possible (Table 5 and Table 13).

Table 5 — Error types of Channel-Related-Diagnosis (IEC 61158-6-10)

C-R-D code	Error definition	IO-Link use
0	reserved	
1	Short circuit	Yes
2	Undervoltage	Yes
3	Overvoltage	No
4	Overload	Yes
5	Overtemperature	Yes
6	Line break	Yes
7	Upper limit value exceeded	Yes
8	Lower limit value underrun	Yes
9	Error	Yes. All remaining IOL-M or IOL-D faults can be mapped into this code. It can be used as indication to invoke PCT or IOPD.
10	Simulation active	No
11	reserved	
12	reserved	
13	reserved	
14	reserved	
15	Parameter missing	No
16	Parametrization fault	Yes, optional
17	Power supply fault	Yes, optional

C-R-D code	Error definition	IO-Link use
18	Fuse blown / open	Yes, optional
19	Manufacturer specific	
20	Ground fault	No
21	Reference point lost	No
22	Process event lost / sampling error	No
23	Threshold warning	No
24	Output disabled	No
25	Safety event	No
26	External fault	No
27	Manufacturer specific	
28	Manufacturer specific	
29	Manufacturer specific	
30	Manufacturer specific	
31	Temporary fault	No

NOTE The codings from 15 to 31 are defined "manufacturer specific" in previous versions of IEC 61158. The corresponding error and help texts are to be defined in the GSD of a PB-DP or PN-IO device. The new release of IEC 61158 recommends the use of the above listed definitions.

5.2.13 I&M functions

The PROFIBUS I&M functions are standardized down to an IO-Link master level (IOL-MM or IOL-CM). [3] provides for functional details. Figure 27 is showing the principle.

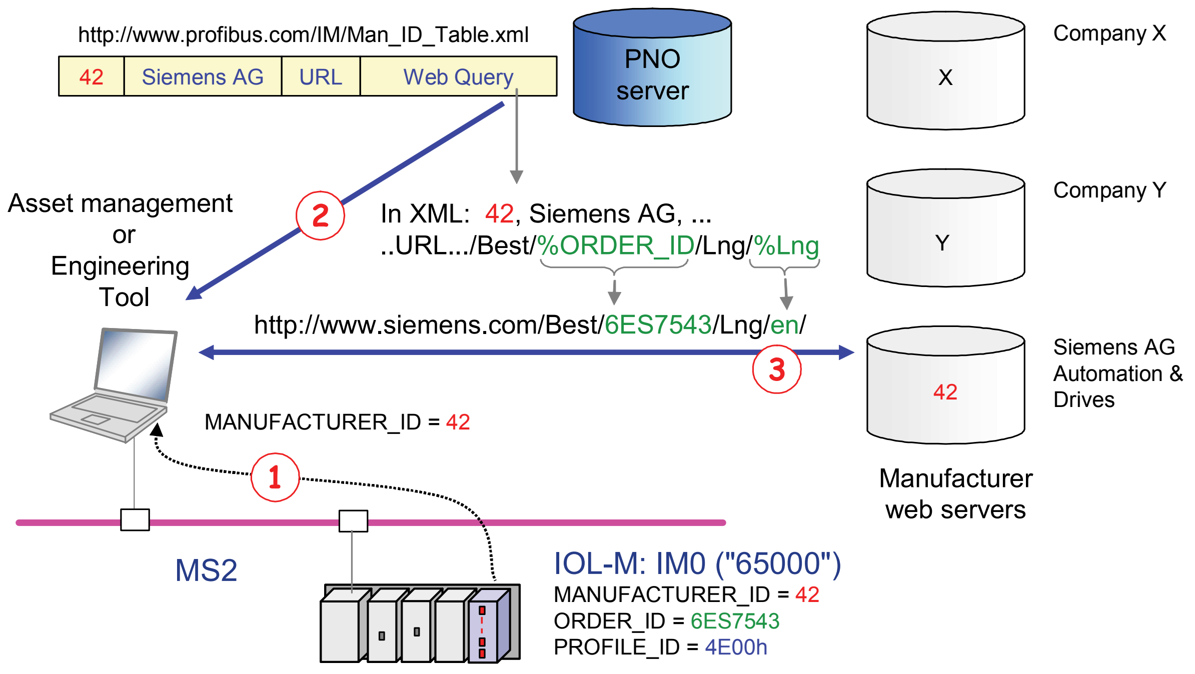

http://www.profibus.com/IM/Man_ID_Table.xml
42
Siemens AG
URL
Web Query
PNO server
X
Company X
Y
Company Y
Asset management or Engineering Tool
2
In XML: 42, Siemens AG, ...
..URL.../Best/%ORDER_ID/Lng/%Lng
http://www.siemens.com/Best/6ES7543/Lng/en/
3
42
Siemens AG Automation & Drives
MANUFACTURER_ID = 42
1
Manufacturer web servers
MS2
IOL-M: IM0 ("65000")
MANUFACTURER_ID = 42
ORDER_ID = 6ES7543
PROFILE_ID = 4E00h

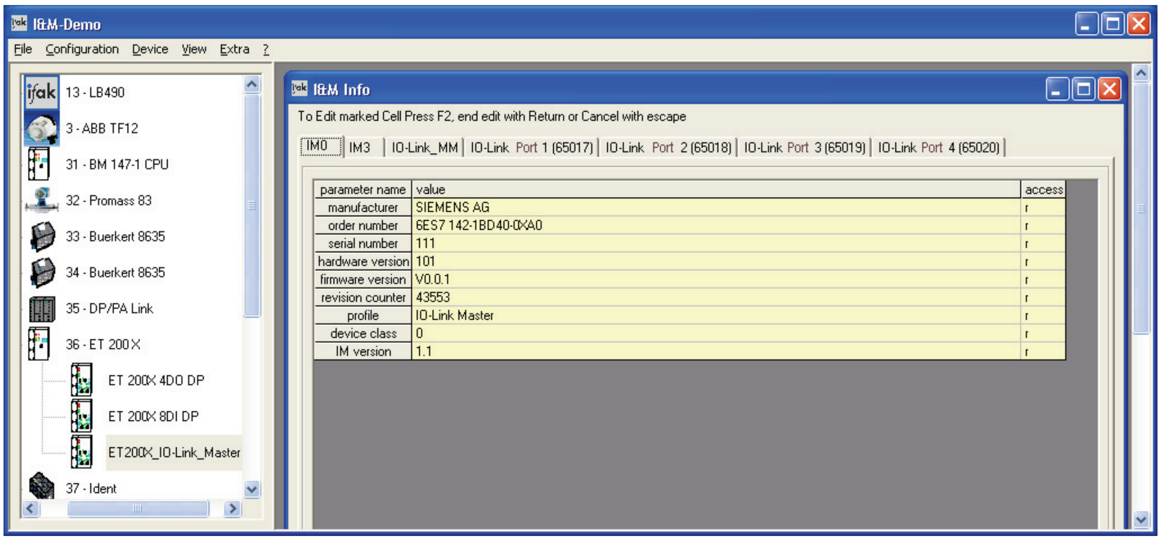
Figure 27 — Principle of referencing the PNO web server for I&M functions

Table 6 is presenting the essential coding of the mandatory IM0 record for any device on PROFIBUS DP or PROFINET IO. This record can be accessed via the CALL mechanism described in 5.2.7.

Table 6 — Coding of the mandatory IM0 record

Content	Size	Coding (IO-Link)
MANUFACTURER_ID	2 Bytes	Unsigned16
ORDER_ID	20 Bytes	VisibleString
SERIAL_NUMBER	16 Bytes	VisibleString
HARDWARE_REVISION	2 Bytes	Unsigned16
SOFTWARE_REVISION	4 Bytes	Char, Unsigned8
REVISION_COUNTER	2 Bytes	Unsigned16
PROFILE_ID	2 Bytes	Unsigned16 (→ 4E00h)
PROFILE_SPECIFIC_TYPE (Device class)	2 Bytes	Unsigned16
IM_VERSION	2 Bytes	2 x Unsigned8
IM_SUPPORTED	2 Bytes	Unsigned16 used as Boolean

Figure 28 is showing a visualization example of IM0 for an IO-Link master module. The items in the top window correspond to the content of the IM0 record. The MANUFACTURER_ID is dissolved by the name of the company through the PNO server.



To Edit marked Cell Press F2, end edit with Return or Cancel with escape

parameter name	value	access
manufacturer	SIEMENS AG	r
order number	6ES7 142-1BD40-0XA0	r
serial number	111	r
hardware version	101	r
firmware version	V0.0.1	r
revision counter	43553	r
profile	IO-Link Master	r
device class	0	r
IM version	1.1	r

Figure 28 — Example of IM0 for an IO-Link master module

IO-Link is a PROFIBUS or PROFINET independent technology. Its integration tasks are similar to those of the PROFINET integration of other existing fieldbus technologies. For example, it is defining its own Vendor_IDs, Device_IDs, Function_IDs, a.s.f. Thus, the easiest way to provide IO-Link I&M functions of IO-Link devices to the user is on the IO-Link related PCT level. Figure 29 is showing the principle.

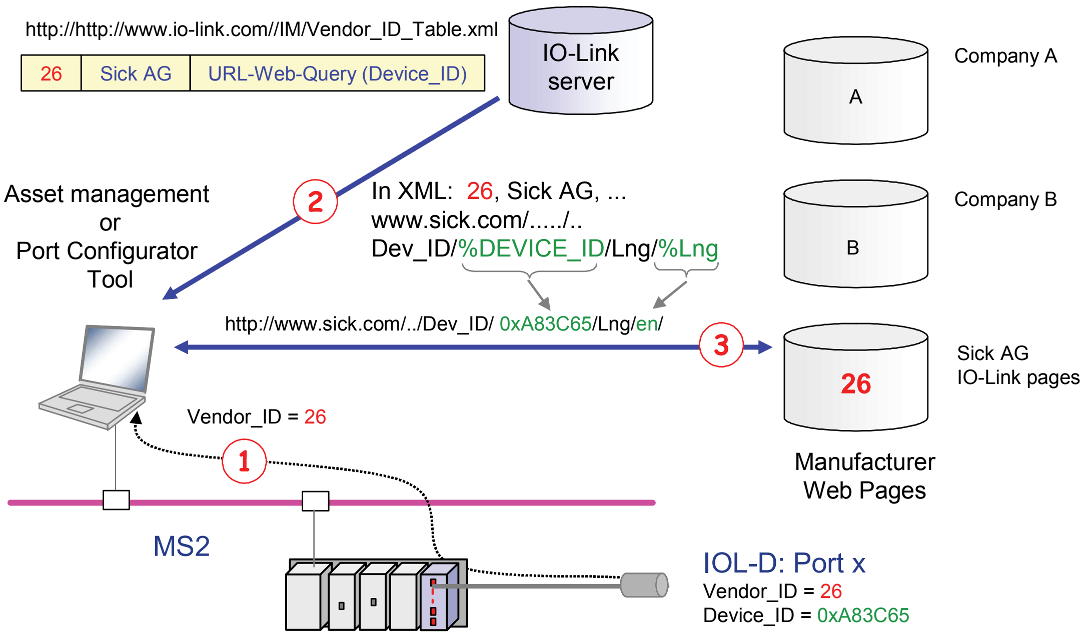
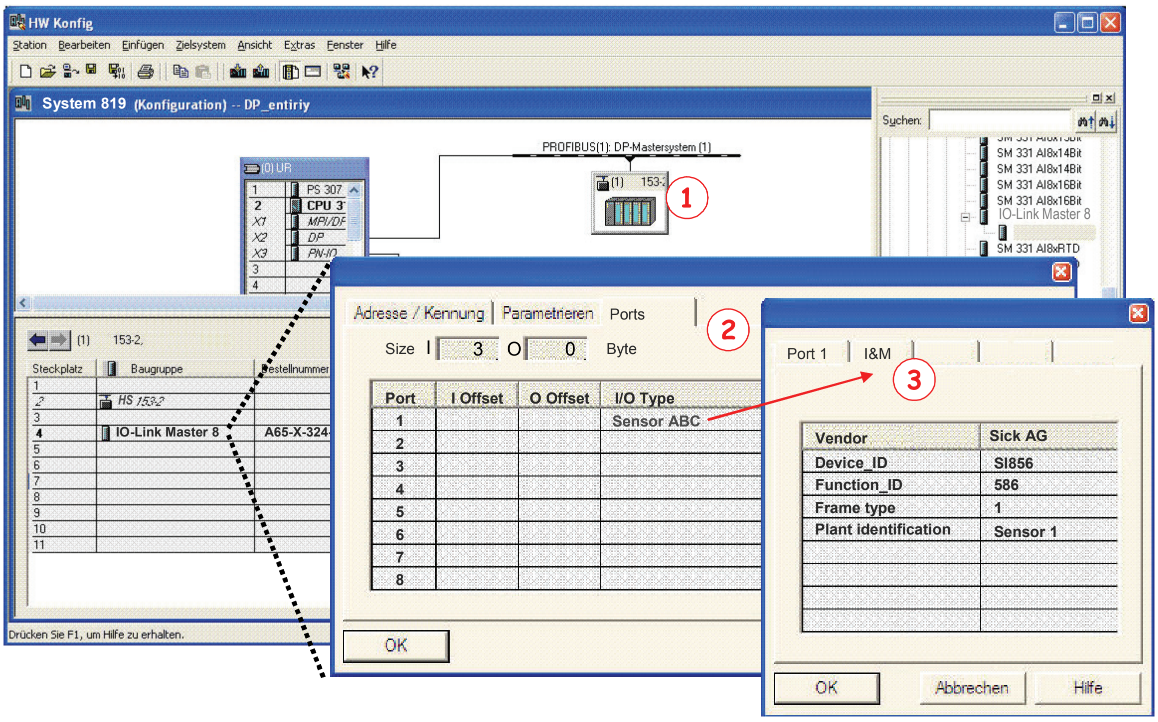

http://http://www.io-link.com//IM/Vendor_ID_Table.xml
26 | Sick AG | URL-Web-Query (Device_ID)
IO-Link server
Company A
A
Company B
B
Asset management or Port Configurator Tool
2
In XML: 26, Sick AG, ...
www.sick.com/...../..
Dev_ID/%DEVICE_ID/Lng/%Lng
http://www.sick.com/../Dev_ID/ 0xA83C65/Lng/en/
3
26
Sick AG IO-Link pages
Vendor_ID = 26
1
Manufacturer Web Pages
MS2
IOL-D: Port x
Vendor_ID = 26
Device_ID = 0xA83C65

Figure 29 — I&M functions on the IO-Link level (Port Configurator Tool)

Figure 30 is showing a visualization example of I&M functions for an IO-Link device on port 1 via the Port Configurator Tool (PCT).



Port	I Offset	O Offset	I/O Type
1			Sensor ABC
2			
3			
4			
5			
6			
7			
8			

Vendor	Sick AG
Device_ID	SI856
Function_ID	586
Frame type	1
Plant identification	Sensor 1

Figure 30 — Example of IO-Link I&M functions via PCT

The only way to provide I&M functions of IO-Link devices on PROFIBUS or PROFINET level is via the PROFILE_ID in IM0 (Table 6), which ranges from 0x4E00 to 0x4EFF for IO-Link. With the help of this PROFILE_ID it is possible to define further records within the profile-specific parts of the FI-Index, for example IM17 = 65017 for port 1 (see Figure 17). Figure 31 shows how an additional *IM_Runtime_IOLinkMaster.xml* file supports the interpretation of the profile specific records. The IO-Link device information of port 1 such as Vendor_ID, Device_ID, Function_ID, etc. is mapped into IM17 = 65017. The IO-Link specific I&M space can be reached via the IO-Link server URL (http://www.io-link.com/IM/Vendor_ID_Table.xml) to resolve the Vendor_ID.

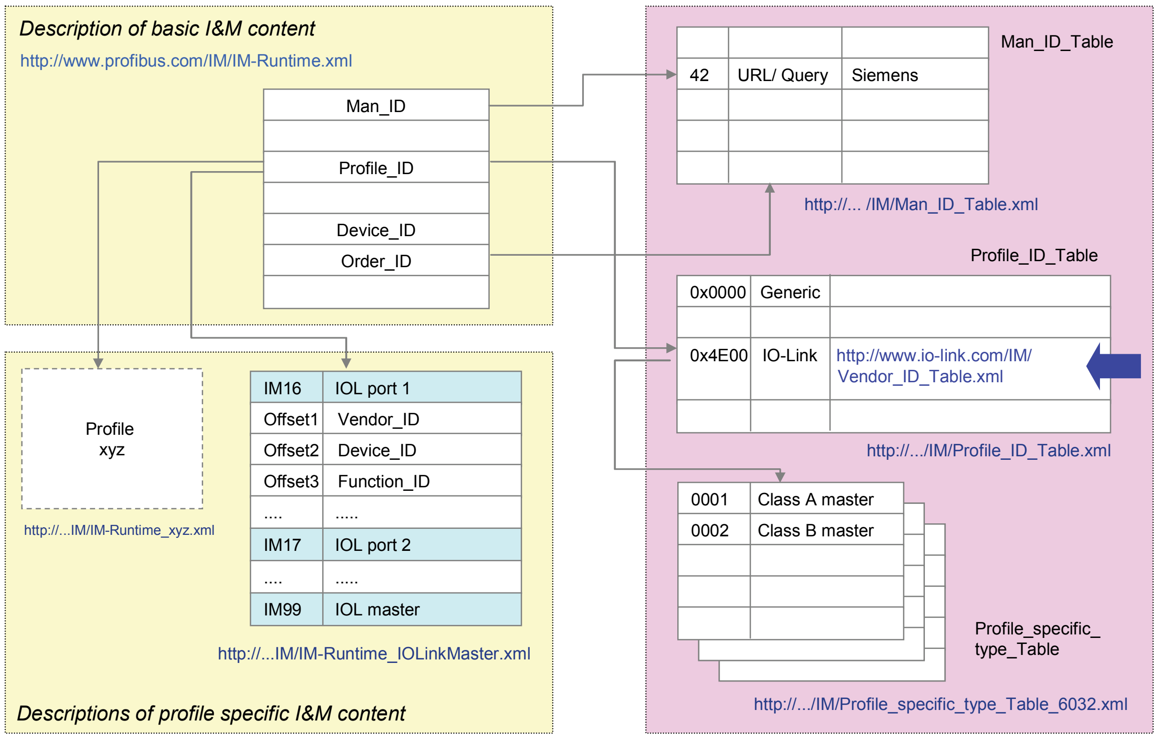
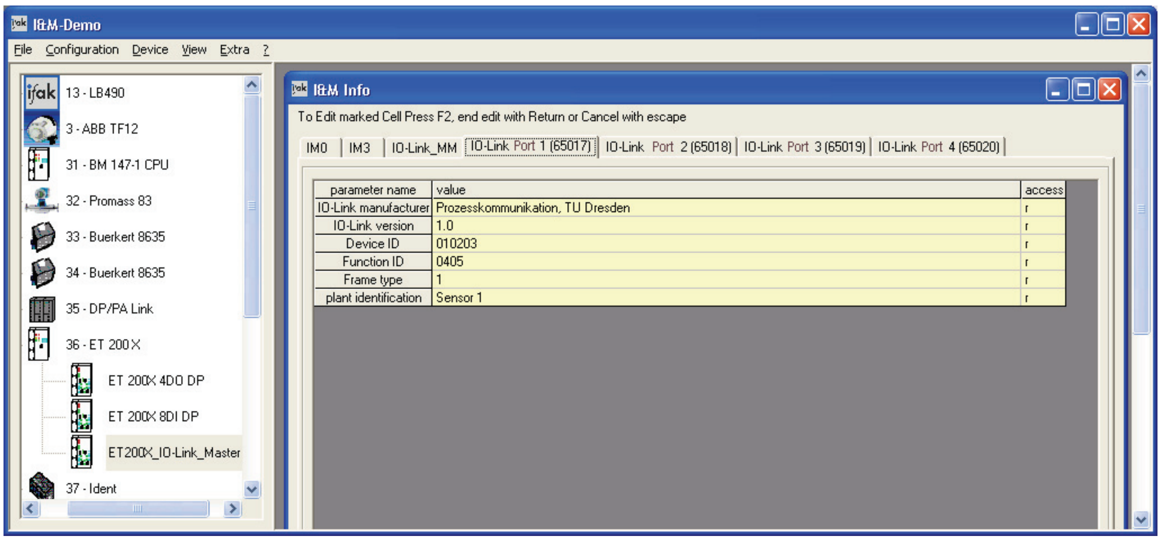

Description of basic I&M content
http://www.profibus.com/IM/IM-Runtime.xml
Man_ID
Profile_ID
Device_ID
Order_ID
Profile xyz
http://...IM/IM-Runtime_xyz.xml
IM16 IOL port 1
Offset1 Vendor_ID
Offset2 Device_ID
Offset3 Function_ID
....
IM17 IOL port 2
....
IM99 IOL master
http://...IM/IM-Runtime_IOLinkMaster.xml
Descriptions of profile specific I&M content
Man_ID_Table
42 URL/ Query Siemens
http://... /IM/Man_ID_Table.xml
Profile_ID_Table
0x0000 Generic
0x4E00 IO-Link http://www.io-link.com/IM/Vendor_ID_Table.xml
http://.../IM/Profile_ID_Table.xml
0001 Class A master
0002 Class B master
Profile_specific_type_Table
http://.../IM/Profile_specific_type_Table_6032.xml

Figure 31 — IO-Link profile specific I&M functions

Figure 32 is showing a visualization example of mapped IO-Link I&M information. It provides the mandatory IM0 as well as an optional IM3 (maintenance or commissioning comment) followed by the IOL-MM and four port informations. I&M information of port 1 is on top.



To Edit marked Cell Press F2, end edit with Return or Cancel with escape

parameter name	value	access
IO-Link manufacturer	Prozesskommunikation, TU Dresden	r
IO-Link version	1.0	r
Device ID	010203	r
Function ID	0405	r
Frame type	1	r
plant identification	Sensor 1	r

Figure 32 — Example of IO-Link specific device information at port 1

5.2.14 Functional safe communication

It is not yet the task of this specification to define the connectivity of safety devices such as defined in IEC 61496 (e.g. light curtains) with redundant and self monitoring so-called OSSD signals. These devices are suffering from the same lacks as the IO-Link devices: missing parameterization and diagnosis support. Therefore, it should be shown that the suggested system features are necessary and sufficient enough to build the basis for the safety extensions. Design objectives are performance and simplicity.

The necessary system features are:

- Pin 4 for parameterization/diagnosis and subsequent fallback to safe digital input according to safety standards
- Pin 2 as the second safe digital input according to safety standards

With this basic design concept, a fieldbus independent connectivity of safety devices should be possible. Further safety fieldbus communication can be performed by the IOL-MM according to individual functional safety communication protocols such as PROFIsafe or others such as those defined in IEC 61784-3-x.

In case of more complex safety data it should be investigated whether a redundant standard IO-Link communication on both channels (Pin 4 and Pin 2) can be sufficient for functional safety communication. Implications for this solution are data integrity level (CRC polynomial), performance and simplicity. However, it may become attractive with the advent of inexpensive IO-Link ASICs in the future.

5.3 Use Cases for PROFIBUS and PROFINET

In this clause several main scenarios are presented, starting with normal operation of a correctly configured and parameterized IOL-M system. This scenario is followed by an off-line session, a commissioning phase with plug-in of an IOL-M with online configuration and parameterization, and the parameter server functions. More specific scenarios are the user program-controlled parameterization of sensors and actuators, the diagnosis capabilities and special situations such as pull and plug of the IOL-M and of the IOL-Ds.

5.3.1 Normal operations of an IOL-M

Preconditions: IOL-M and IOL-Ds are configured and parameters are correctly assigned to both. The IOL-D parameters are either hold locally (remanent) within the IOL-D or within the IOL-M. Power-on occurs on both the PB/PN system and the IO-Link system.

For performance reasons it is recommended to design the IOL-M such that the PB/PN start-up and the IOL-M start-up can run concurrently. If IO-Link parameters such as port configurations are conveyed via GSD and for example a Prm Block in the Prm message, the IOL-M is waiting on the arrival of these parameters prior to the configuration of the IO-Link ports. The implications of this approach are cost and effort for parameter remanence within the IOL-M.

In case the IOL-M is not configured specifically ("default DI ports"), it automatically starts as DI (Table 2) on all its ports. The GSD section contains a default assignment (for example 8 bit or 16 bit input) for the fieldbus PDU. The IOL-M reads all DI in one step to achieve high performance and passes over the data to the fieldbus layer stack in a cyclic manner. Faults occuring on the ports or within the IOL-M are transmitted preferably via C-R-D codes (5.2.12).

In case the IOL-M is configured specifically ("non default DI ports"), it starts with adjusting the ports and the communication features (modes, bitrates, frame types, fallback, etc.). The IOL-M reads the parameter CRC signature out of IOL-Ds and in case of mismatch downloads the corresponding parameter set into the IOL-D. IOL-Ds without remanence are directly provided with their parameter sets. After this part of the start-up phase the IOL-M starts exchanging IO process values with the IOL-Ds and reading/writing the DI/DO signals that are availbale and assigned. This can be done with a fixed cycle rate or asynchronously. The IO process values are packed for the fieldbus PDU according to the rules of this specification and to the IO configuration structure chosen by the user from the GSD selections.

An IOL-M without remanence starts notifying the iPar-Server and causes the download of the parameter set after the fieldbus start-up similar to the IOL-M replacement procedure.

5.3.1.1 Start-up of ports with inadvertently permuted IOL-D cables

Inadvertently permuted IOL-D cables can only be detected electronically when the IOL-M is able to read the parameter CRC signature and uses inspection level 4, i.e. inclusion of the serial number as defind in 11.4.

Inadvertently permuted IOL-Ds operating on a fixed DI/DO port or running in SIO mode can only be detected through a "Reconfiguration" command (11.9) and inspection level 5 (11.4).

5.3.1.2 IOL-D replacement

A defect IOL-D can be replaced with another IOL-D either at runtime ("hot swap") or at power-off. The second situation corresponds to normal operation, when the IOL-M returns to power-on.

"Hot swap" or a defect IOL-D can cause the IOL-M to interrupt the communication and to cast a diagnosis message across the fieldbus (*diagnosis coming*). The IOL-M continuously tries to reestablish the IO-Link communication. After the replacement with another IOL-D the IOL-M resumes communication and checks the CRC signature. Due to the mismatch it downloads the parameter set. When running correctly again, the IOL-M casts another diagnosis message across the fieldbus to indicate the remedy of the defect (*diagnosis going*).

The IOL-M cannot detect the "hot swapping" of an IOL-D running in SIO mode. This can lead to destructive situations if the IOL-D starts running with its default parameter set instead of an expected parameter set conveyed through a regular IOL-M start-up.

Unfavorable solutions could be an (expensive) current detection of the IOL-D power supply or periodic "wake-ups". In the latter case the IOL-M is holding the signal value sampled at the beginning of the wake-up sequence for a certain period of time (about 100ms).

The compromise solution is the provision of a special "reconfiguration" command for the entire IOL-M or individual ports via the function block *WR_IOL_CALL* (9.3.2). The user needs to program a connection between this "command" and a "reconfiguration button" for maintenance and repair. This specification covers only the compromise solution.

5.3.1.3 IOL-M replacement

A defect IOL-M can be replaced with another IOL-M either at runtime ("hot swap") or at power-off. These situations correspond to normal operation, when the IOL-M returns to power-on (5.3.1).

5.3.2 Offline configuration and parameterization session

One objective of this IO-Link integration design is to provide offline engineering of both the IOL-M and the IOL-Ds as much as possible.

5.3.2.1 PB-DP and PN-IO configuration (GSD based)

An IOL-M is selected in the fieldbus configuration tool and "connected" to a particular fieldbus segment or to a slot within a remote I/O. The associated GSD should only contain the necessary information for the fieldbus communication and a selection of viable I/O data structures and particular diagnosis definitions for the IO-Link support.

5.3.2.2 Generic IOL-D with anonymous parameters

It is possible but not recommended for the GSD of an IOL-M to carry the definitions of port configurations or any IOL-D parameters even those for IOL-Ds with so-called anonymous parameters (10 bytes). A specially adapted IODD shall be specified and provided for those "generic" IOL-Ds that are not providing their own particular IODD (or IOPD).

5.3.2.3 Port configuration tool (PCT)

Out of the fieldbus configuration tool and from the IOL-M information (icon) it is possible to launch the associated port configuration tool (PCT) via TCI or FDT means. The PCT provides the necessary views and entries for the port configuration, the installation and handling of IODDs, the IOL-D parameterization, the associated I/O data structures, the retrieval of I&M information, the diagnosis information and more (5.2.8).

It shall be possible to configure and parameterize the IOL-M and its ports as well as the IOL-Ds as long as their IODDs are available. It is not necessary for a vendor of an IOL-M to provide an associated IODD. The generated IO-Link project data (IOL-M and IOL-Ds) shall be stored offline within the appropriate project storage of the fieldbus configuration or engineering tool.

5.3.2.4 IODD installations

The IODD shall be provided in XML format as defined in [10]. The PCT supports the installation procedure of these IODD in a commonly defined computer and operating system location (registry). Thus, other PCT or instances of PCT can retrieve this IODD "database" without reinstallation of the IODD.

5.3.2.5 Port configuration

Port configuration is achieved with the help of the PCT. It is not necessary to specify a particular IODD for the IOL-M. It is the responsibility of the vendor of an IOL-M and its associated PCT to guarantee the interoperability between both. When "offline", the PCT writes to and reads from a "virtual" IOL-M (local database).

5.3.2.6 Parameterization via IODD or IOPD

When "offline", the PCT writes to and reads from "virtual" ports (local database).

5.3.2.7 Process I/O data configuration

The IOL-M is mapping and packing the process data of the IOL-D into the fieldbus PDU according to the rules of [9] and this specification (0).

5.3.2.8 Storage of project data

The total data of the "virtual" IOL-M (5.3.2.5) and the "virtual" ports (5.3.2.6) are forming an "IOL-M project" that can be stored and retrieved. The integration means TCI [11] and FDT [12] are both defining possibilities for the storage of IOL-M project data within a fieldbus automation project.

5.3.2.9 Programming support

In order to change parameters within the IOL-D at runtime, the user can deploy and instantiate special IO-Link communication function blocks as defined in 5.2.10 and normally provided by the system vendor or a "ParChannel" function block as defined in A.1 and provided by the IOL-M vendor. This normally is done with the help of the system engineering or programming tool.

Via these function blocks "Commands" and "Status" information can be exchanged with the IOL-M also.

5.3.2.10 PDU configuration (GSD based)

The PCT shall provide information for the necessary number of input and output bytes for a particular configuration. On the fieldbus configuration level the user chooses an appropriate combination of input and output bytes out of the selection within the GSD.

5.3.3 Online session (commissioning)

Preconditions: The fieldbus system is switched on and running, the IOL-Ds are connected to the IOL-M ports, the ports of the IOL-M are all running in "default DI" mode, acyclic communication on the fieldbus down to the IOL-M is possible via TCI or FDT, an engineering tool with fieldbus configuration capabilities is connected and the fieldbus configuration for the IOL-CM or IOL-MM is available.

A user launches the PCT and can either open a project from an offline session or start a new project. An available project can be copied to the IOL-M and its ports. After the adjustment of an appropriate combination of input and output bytes out of the selection (GSD) on the fieldbus configuration level, the fieldbus system needs to be restarted in order to provide the necessary parameters to the fieldbus device and to reconfigure the IOL-M. The system is now ready for testing.

In case of a new project, the user is going through the same steps as with the offline session. However, the assignments are stored directly in the IOL-M or IOL-D and saved within the "virtual" local database. After the online session the identical configuration and parameter set can be saved within the fieldbus automation project (TCI/FDT).

5.3.3.1 Communication and addressing models

The basic communication and addressing models to achieve the online session objectives are all defined within this specification to the necessary and sufficient extend. It demonstrates available fieldbus features that can be used. When used they shall be used in the decribed manner to guarantee availability in existing systems and interoperability.

5.3.3.2 PCT connection

The principles of PCT connections and communications are described in 5.2.4 and 5.2.5. Refined "Read/Write Record" services (CALL) to individually assigned indices are defined in this specification. Additional information is available in [11] and [12].

5.3.3.3 Generic IOL-Ds

Generic IOL-Ds are those not providing an individual IO-Link device description (IODD) file. The PCT holds a generic IODD file for these devices, which allows defining the values of up to 10 bytes anonymous parameters. These values can be assigned manually from an IOL-D data sheet in hexadecimal representation.

5.3.3.4 IOL-D parameterization tool (IOPD)

The PCT is the only entity to know the port address of a particular IOL-D and the index of the IOL-M to be used for conflict-free communication via the refined "Read/Write Record" services (CALL). Thus, individual IOPD tools can only be launched from a PCT using the invocation variables "port number" and "index for the CALL services" amongst others.

5.3.3.5 Alternating parameter sets

It may be necessary for an IOL-D to switch between several alternative parameter sets during production. The solution in favor here is to feed the alternative parameter sets one after the other into the IOL-D and to save and restore the complete block of parameter sets as bulk data. The command for the IOL-D to switch to a particular parameter set comes via process data from the user program.

5.3.3.6 External parameterization (teach-in)

Many existing sensors being candidates to become an IOL-D provide mechanisms for external parameterization such as a teach-in button attached to the sensor or a USB adapter. For those devices it is necessary to indicate a change of the parameters via a "Dirty Flag". Otherwise, the IOL-M will not be able to distinguish between the different parameter sets to save and restore after a restart of the IOL-M.

5.3.4 Parameter server

This specification defines a two-level parameter server concept and describes its dynamics: Between an IOL-M and its IOL-D and between an IOL-M and the fieldbus host system (iPar server).

5.3.5 User program-controlled parameterization

Modern flexibel manufacturing requires IOL-Ds to accept parameter changes at runtime. This specification defines means and methods to provide this capability. See also 5.3.2.9.

5.3.6 Diagnosis

The IO-Link specification defines many different and specialized diagnosis types. It is common understanding in the fieldbus world, that service personnel nowadays can only handle major and simple diagnosis messages. Thus, this specification suggests two levels of diagnosis "granularity". At the fieldbus level it should be "coarse" and thus the IO-Link diagnosis types are mapped to a few C-R-D types. At the PCT level it should be possible to obtain the "fine" individual IO-Link diagnosis code and display the associated text out of an IODD file.

5.3.7 Special situations

In this section special use cases are defined.

5.3.7.1 IOL-M w/o cyclic fieldbus operation

An IOL-M shall be able to operate stand-alone without cyclic fieldbus communication to a controller (PLC, host, etc.). However, the acyclic communication for a PCT shall be enabled.

5.3.7.2 Class A and class B masters

A class B master usually provides extra 24V on Pin 2 and Pin 5. Initial configuration after power-on shall be DI. Second configuration step is IO-Link communication and read out of the requested IOL-D port configuration. Final configuration step is the adjustment of the port according to this request. This procedure is only necessary for class B master with configurable Pin 2 (extra 24V, DI, or DO).

5.3.7.3 Fallback, the extra operational mode of an IOL-M

One of the major highlights of IO-Link is the possibility for an IOL-D to receive parameters at start-up (or per user request) and then switch to the DI mode and thus combine flexibility and high performance (switching signals). The corresponding term *fallback* is not a port mode and thus unknown to the IOL-D. It rather is an operational mode of the IOL-M consisting of the following subsequent steps on a particular port (11.9):

- *Scan mode*: Wake-up procedure, communication, and parameterization (PCT or FB)
- *SIO mode*: Change port to DI or DO mode, no communication, standard switching signal from or to the IOL-D

Implications of the fallback operational mode are:

- IOL-D indicates "fallback" capability by a property in its IODD file
- User is provided with means in PCT or FB to command the fallback operational mode during commissioning or at runtime
- IO data is one bit input
- IO data exchange on the fieldbus level is "disabled" during the fallback sequence (="0")
- IOL-D exchange (e.g. for repair) cannot be discovered by the IOL-M in DI mode
- IOL-D exchange for repair needs special user interaction ("reconfiguration" command in 9.3.2) to go through the fallback sequence and to save or restore parameters

6 Overview of PB-DP and PN-IO

Independent from IO-Link the following sections provide a brief overview on important terms and concepts of PROFIBUS DP and PROFINET IO.

6.1 PROFIBUS DP

The information herein is based on the international standard IEC 61158. Further details may be retrieved from [13] and [15].

PROFIBUS DP is designed to support the communication of the following device types:

- DP master class 1
- DP master class 2
- DP slave

Figure 33 demonstrates these three device types and their relationships. DP master class 1 is the fieldbus controlling component that also is responsible for the start-up parameterization and the diagnosis handling. This master may be hosted by a PLC, NC, or PC. DP master class 2 is an additional fieldbus component running concurrently with the master class 1 with the possibility to acyclically communicate with other components. This master usually is hosted by a PG, PC, OP, or process monitoring system (HMI). DP slave is a component conveying process I/O data, usually in form of a remote I/O. Here, the DP slave represents one or more IO-Link master systems (IOL-CM or IOL-MM).

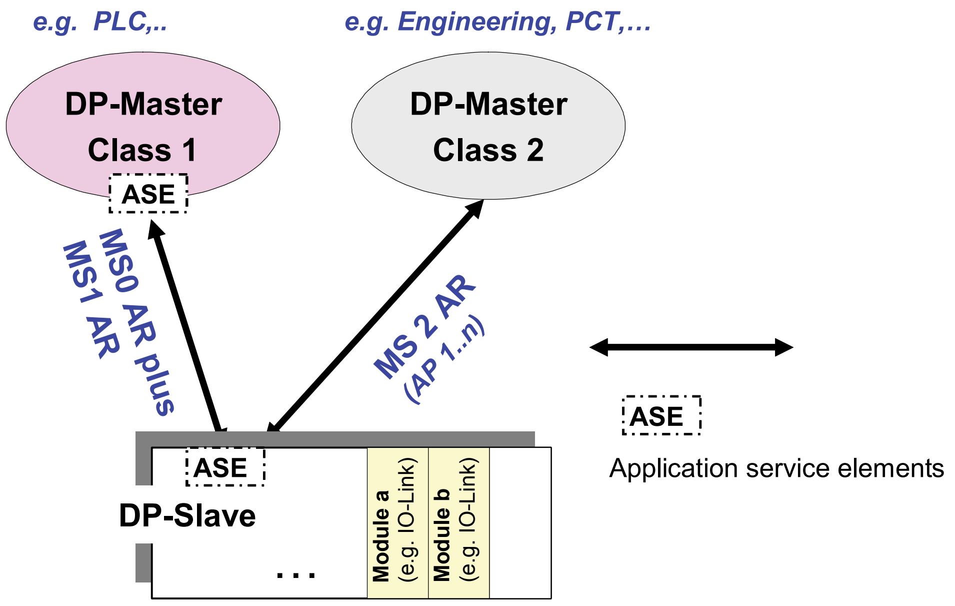


Figure 33 — Overview of PB-DP relationships

Basically there are several types of application relationships MS0 up to MS3 (6.1.2) to connect so-called application processes. Relevant for this specification are MS0, MS1, and MS2. For MS0 and MS1 only a maximum of *one* application process (API=0) can be supported. For MS2 several different application processes can be supported.

An AP may be distributed across several slots as shown in Figure 34. The figure also shows the context of application process, the data elements (ASE), and the slots of a DP slave.

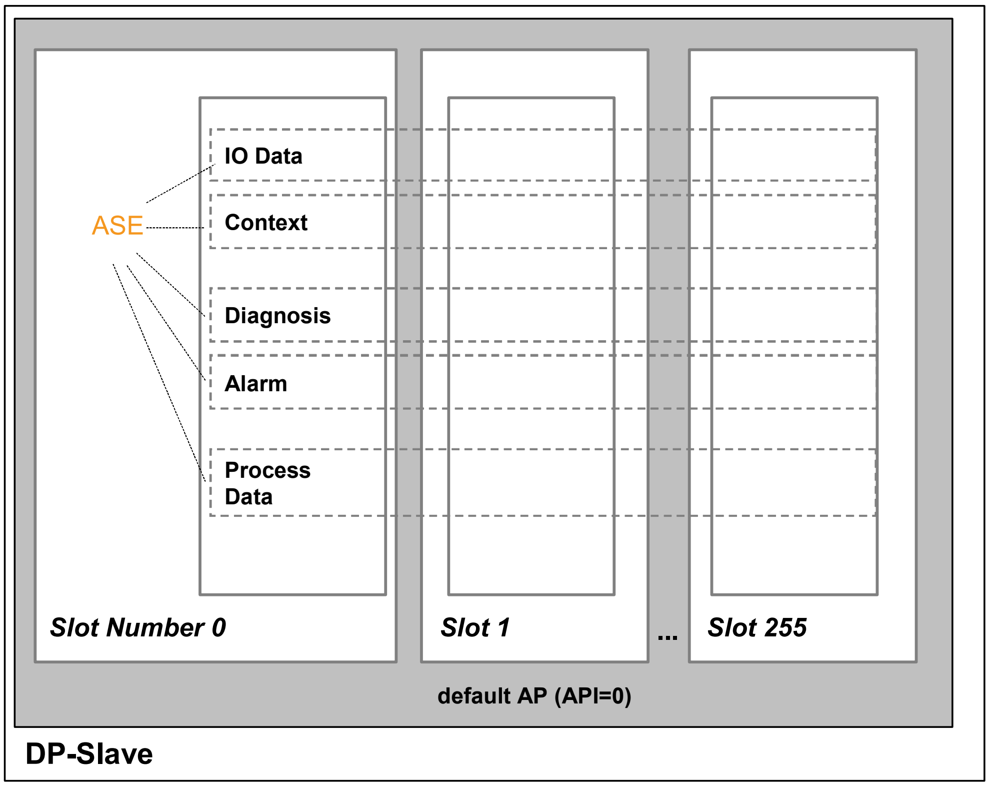


Figure 34 — DP slave with AP, slots, and ASEs

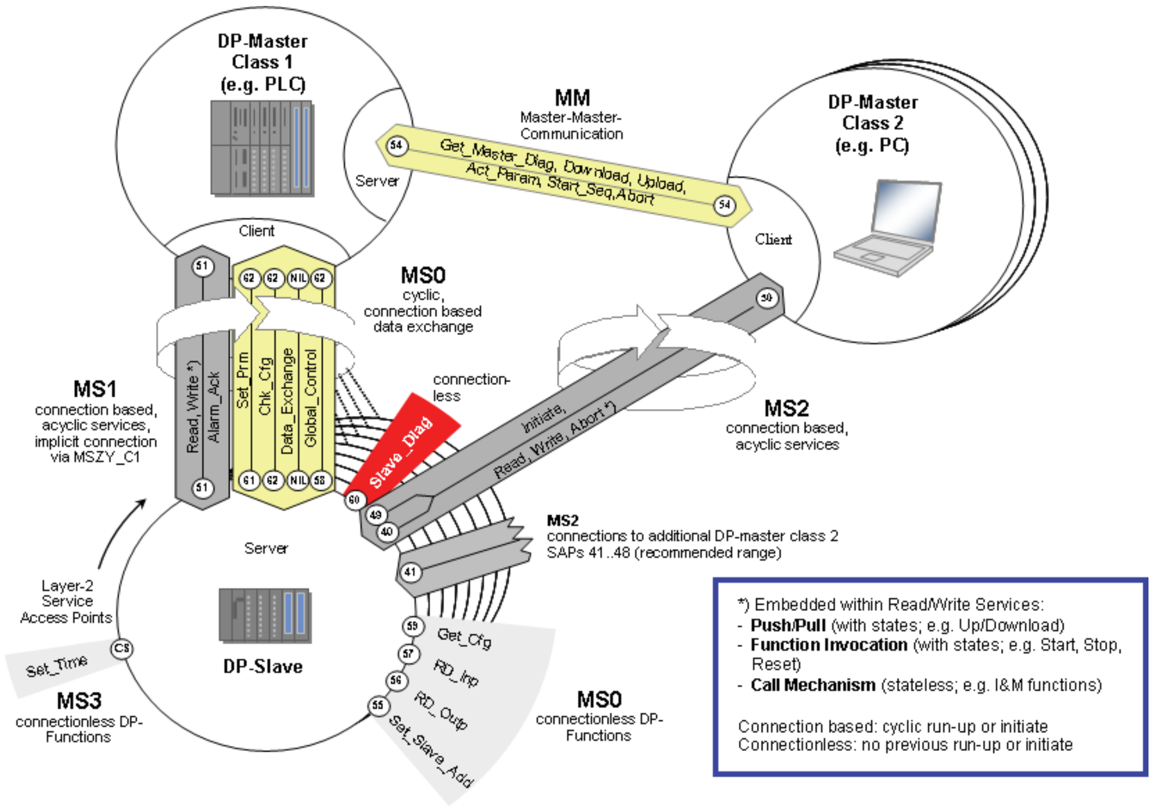
6.1.1 DP-V0 and DP-V1

Within the PROFIBUS system several terms describing DP-slave communication functionality are common. DP-Vx describes a specific subset of the over all PROFIBUS DP communication functionalities (protocol and services). DP-V0 indicates that a device is able to communicate cyclically, a minimum functional requirement of each DP-slave. DP-V1 characterises the extended functionality of acyclic communication. In addition the extended specification in DP-V1 provides an advanced parameterization and diagnosis model (alarm and status). To indicate acorresponding device functionality, e.g. the extended diagnosis model, the DP-V1 terms are persistent (e.g. GSD key word “DPV1_SLAVE”).

The DP-Vx terminology indicates packages of historical extensions of PROFIBUS DP-functions (DP-V0, DP-V1, and DP-V2). Unfortunately, these terms are unsuitable to characterize a device's communication capability. It therefore is recommended to classifiy communication functionality of PROFIBUS DP devices using the MSx terminology (MS0-3), considering the participating communication systems. The terms MSx describe specific communication functionalities (protocol and services) which are related to a master/slave communication relation (also known as application relation (AR).

6.1.2 MS0, MS1, and MS2

In IEC 61158 [13], [15] the joint fulfillment of communication tasks between controlling devices (e.g. PLC or DCS) and field devices is modelled by means of distributed application processes (AP). The communication between the distributed application processes is described by several types of application relationships (AR) (Figure 33). The purpose of the AR´s is the exchange of information and coordination of the application process's joint operations. In other words: the enabling of “communication channels”. Only the application relationships MS0-2 are in the scope of this specification (Figure 35).


DP-Master Class 1 (e.g. PLC)
Server
Client
MM
Master-Master-Communication
54
Get_Master_Diag, Download, Upload, Act_Param, Start_Seq,Abort
54
DP-Master Class 2 (e.g. PC)
Client
50
51
62 62 NIL 62
Read, Write *)
Alarm_Ack
Set_Prm
Chk_Cfg
Data_Exchange
Global_Control
51
61 62 NIL 58
MS0
cyclic, connection based data exchange
connection-less
Slave_Diag
60
initiate, Read, Write, Abort *)
49
40
MS2
connection based, acyclic services
MS2
connections to additional DP-master class 2
SAPs 41..48 (recommended range)
41
MS1
connection based, acyclic services, implicit connection via MSZY_C1
Layer-2 Service Access Points
Server
DP-Slave
CS
Set_Time
MS3
connectionless DP-Functions
59 Get_Cfg
57 RD_Inp
56 RD_Outp
55 Set_Slave_Add
MS0
connectionless DP-Functions

*) Embedded within Read/Write Services:
- **Push/Pull** (with states; e.g. Up/Download)
- **Function Invocation** (with states; e.g. Start, Stop, Reset)
- **Call Mechanism** (stateless; e.g. I&M functions)

Connection based: cyclic run-up or initiate
Connectionless: no previous run-up or initiate

Figure 35 — Overview of MS services of PROFIBUS DP

MS0: application relationship between the application process of one DP-master (Class 1) and all related DP-slaves and optional between the AP of one or several DP-master (Class 2) and all related DP-slaves and optional between the AP of one or several DP-slaves with all related DP-slaves for the following purposes amongst others:

- cyclic exchange of the I/O data with the DP-master (Class 1)
- acyclic data transfer for parameterisation, configuration and diagnosis (DP-master (Class1)

MS1: connection-oriented relationship between the application processes of one DP-master (Class 1) and of one related DP-slave for the following purposes amongst others:

- acyclic read and write of records /variables
- acyclic transfer of alarms
- invocation of stateless and/or state-oriented functions

MS2: connection-oriented relationship between the application processes of one DP-master (Class 2) and of one related DP-slave for the purpose amongst others:

- Initialization of connections
- acyclic read and write of records /variables
- invocation of stateless and/or state-oriented functions

6.1.3 Application Service Elements (ASE)

An application service element (ASE), as defined in ISO/IEC 9545, is a set of application functions providing a capability for the interworking of application processes for a specific purpose. ASEs provide a set of services for conveying requests and responses to and from application processes and their objects.

The application layer of PROFIBUS DP as defined in IEC 61158 [13] and [15] offers the following ASEs (Figure 34):

IO Data ASE
The IO Data ASE provides a set of services to convey IO data cyclically. These data always belong to those slots that have been configured in terms of the Context ASE.

Process Data ASE
The Process Data ASE provides a set of services to convey data acyclically. The application of the DP Master (Class1, 2) requests each transmission individually whenever needed. The Process Data ASE can be related to all APs of a DP Slave.

Diagnosis ASE
The Diagnosis ASE provides services for the DP Master (class1, 2) to read Diagnosis information from a DP Slave.

Alarm ASE
The Alarm ASE provides a set of services to convey alarms issued by the DP Slave. The assigned DP Master (class1) acknowledges the alarm.

Context ASE
The Context ASE provides a set of services to

- convey device parameter and configuration according to a device description
- set or check I/O data configurations

6.2 PROFINET IO

The information herein is based on the international standard IEC 61158. Further details may be retrieved from [14] and [16].

PROFINET IO is designed to support the communication of the following device types:

- IO Controller
- Supervisor
- PN-IO Device

Figure 36 demonstrates these three device types and their relationships. IO Controller is the fieldbus controlling component that also is responsible for the start-up parameterization and the diagnosis handling. This IO Controller may be hosted by a PLC, NC, or PC. The Supervisor is an additional fieldbus component running concurrently with the IO Controller with the possibility to acyclically communicate with other components. This Supervisor usually is hosted by a PG, PC, OP, or process monitoring system (HMI). PN-IO Device is a component conveying process I/O data, usually in form of a remote I/O. Here, the PN-IO Device represents one or more IO-Link master systems (IOL-CM or IOL-MM).

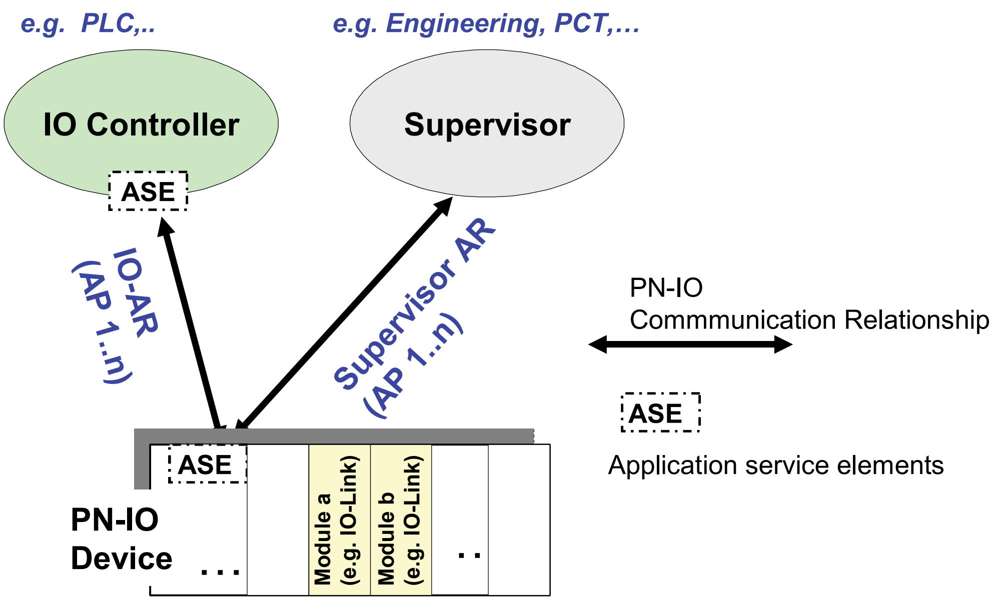


Figure 36 — Overview of PN-IO relationships

6.2.1 Application relationships

Basically there are two types of application relationships: IO-AR and Supervisor AR. Any application relationship may maintain one or more application processes (AP) to PN-IO Devices.

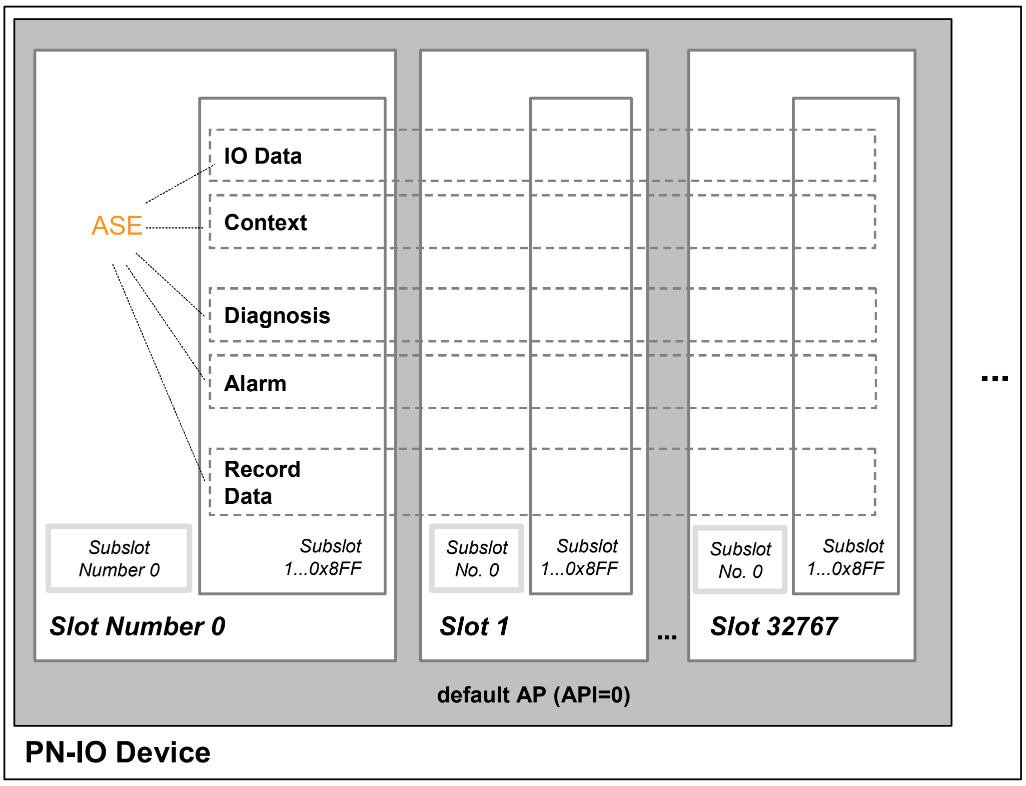


Figure 37 — PN-IO Device with APs, slots, subslots, and ASEs

In the application process environment, an application may be partitioned and distributed to a number of devices on the network. Each of these partitions is referred to as an application process (AP). In this case each individual AP is uniquely identified by an AP Identifier (API) as shwon in Figure 37. Each PN-IO device shall have a Default AP with API = 0. The Default APs shall provide device related information. Other APs are reserved to be uniquely assigned to device profiles and further standardized use.

An AP may be distributed to several slots and subslots. Figure 37 shows the relationship between the APs, the data elements, the slots, and subslots of a PN-IO device. The gray color framed boxes illustrate that there exist no “real” subslot 0. Subslot 0 shall not contain e.g. IO data.

6.2.2 Application Service Elements (ASE)

An application service element (ASE), as defined in ISO/IEC 9545, is a set of application functions that provide a capability for the interworking of application processes for a specific purpose. ASEs provide a set of services for conveying requests and responses to and from application processes and their objects.

The application layer of PROFINET IO as defined in IEC 61158 [14] and [16] offers the following ASEs (Figure 37):

IO Data ASE
The IO Data ASE provides a set of services to convey IO data cyclically. These data always belong to those slots/subslots that have been configured in terms of the Context ASE. IO Data Objects contain a status for transmission. Optionally the IO Data ASE offers the possibility to share the Input Data of one PN-IO Device with other PN-IO Devices. IO Data may also be read and written by an IO supervisor acyclically.

Record Data ASE
The Record Data ASE provides a set of services to convey data acyclically. The application of the IO Controller or IO Supervisor requests each transmission individually whenever needed. The Record Data ASE can be related to all APs of a PN-IO device.

Diagnosis ASE
The Diagnosis ASE provides services for the IO Controller or IO Supervisor to read Diagnosis information from a PN-IO Device.

Alarm ASE
The Alarm ASE provides a set of services to convey alarms issued by the PN-IO Device or IO Controller. The assigned IO Controller acknowledges the alarm.

Context ASE
The Context ASE provides a set of services to

- convey device parameter and configuration according to a device description
- identify AR endpoints
- maintain AR and CR parameters (timeouts, modes, ..)
- and to establish or release an association between individual APs

7 Mapping model

The following considerations focus only on the mapping of the IO-Link functions to the relevant slot/subslot definitions. The mapping is such that there exists no restriction in respect to the design or modeling of the PB-DP slave or PN-IO Device respectively.

The physical structure of a remote I/O system or of an IO-Link system (IOL-M and IOL-D) is mapped here to the "logical object model" of PB-DP and PN-IO. This object model supports two types of remote I/O systems (Figure 38).

- *Modular remote I/O system (type A)*: Discrete physical modules (e.g. DI, DO, or IOL-MM) can be added to or removed from the "backplane" communication system of a remote I/O on an individual basis thus permitting variable and extensible configurations.
- *Compact remote I/O system (type B):* All functions are integral part of a PB-DP slave or PN-IO device. However, the functions can be separated into "virtual" slots/subslots. Compact devices usually provide a fixed scope of operation that can be variied to some extend through inclusion/exclusion of functions and parameterization within limits.

One or more IOL-MM can be part of a modular remote I/O. An IOL-CM is part of a compact remote I/O. IOL-D are connected point-to-point to one of the ports of an IOL-MM or IOL-CM.

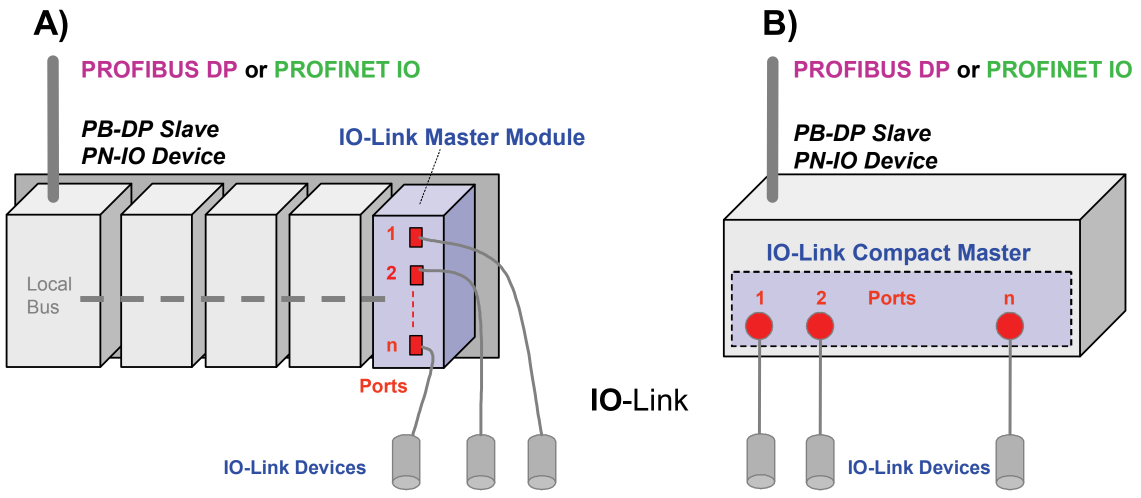


Figure 38 — IO-Link master modules and compact masters

7.1 Address model of PROFIBUS DP

Basis for the mapping of IO-Link functions onto PB-DP is the node address/slot/index address model, permitting access to the PROFIBUS objects DP slave, module, and channel. The IO-Link integration (IOL-M with its associated IOL-Ds) is using this model.

- All the IO-Link functions can be addressed via slot number and index.
- All the diagnosis information can be traced back through slot and channel number.

7.1.1 Single-slot mapping

The complete IO-Link system (IOL-M with its associated IOL-Ds) is mapped onto one *single slot* of the PB-DP slave. This slot is the only access point to the IOL-M.

The single-slot mapping is the *preferred* and *highly recommended* method for the integration of IO-Link. It is particularly suitable for physically modular slave devices. Basically, the single-slot mapping can be used for both the physically modular and the compact PB-DP slave.

Figure 39 is showing the mapping principle of an IO-Link system onto a single PB-DP slot. It is the basis for the further descriptions in this specification containing details of aspects.

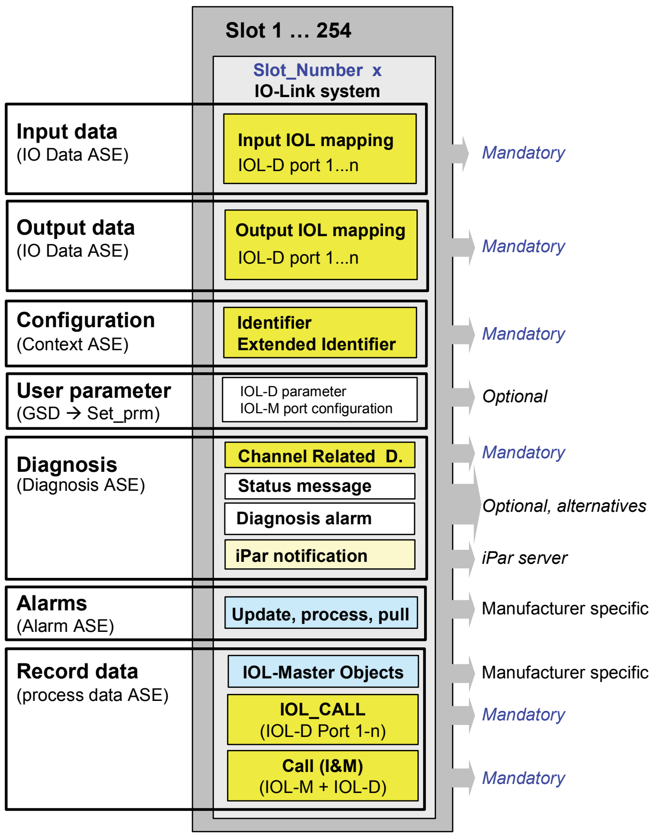


Figure 39 — Single-slot mapping (PB-DP)

Mapping rules:

- An IO-Link system can be mapped onto any slot from 1 to 254 within the address schema of the PB-DP slave. Slot 0 is not permitted.
- Output data of all the IOL-Ds and of the IOL-M are mapped onto the output address space of the particular slot of the PB-DP slave (IO Data ASE). See details in 7.3.3.
- Input data of all the IOL-Ds and of the IOL-M are mapped onto the input address space of the particular slot of the PB-DP slave (IO Data ASE). See details in 7.3.3.
- A configuration check is executed during start-up of the PB-DP slave (Context ASE).
- GSD based User Parameters are transmitted during start-up of the PB-DP slave. It is highly recommended not to transmit IO-Link specific User Parameters (IOL-M and IOL-D). Optionally it is possible to transmit the port configurations of an IOL-M and the 10 bytes anonymous parameters for IOL-Ds.
- Diagnosis information of the IOL-Ds or the IOL-M respectively is mapped to C-R-D and optionally to "Status messages" (Diagnosis ASE). See details in 7.4.
- Usage of the "Alarm model" for diagnosis information is manufacturer specific and is not standardized herein (Alarm ASE).
- Access to IOL-Ds (including I&M data) as well as to the I&M data of the IOL-M is realized through "Record data" (CALL and IOL_CALL in Process data ASE). See details in 9.
- Access to IO-Link master objects is possible through special IOL-M "Record data" (Process data ASE). See details in 8.

7.1.2 Multi-slot mapping

The IOL-M itself is mapped to a particular slot x. The IOL-Ds connected to the ports are mapped to the subsequent slots. Slot number of an IOL-D results from the slot number of the IOL-M plus the port number.

The multi-slot mapping differs from the single-slot mapping essentially by the IOL-D-wise mapping of input/output data (cyclic data transfer) and diagnosis or alarms. Record data are only mapped onto the slot of the IOL-M.

Anonymous Parameter: In cases where optionally 10 bytes of IOL-D anonymous parameters are transmitted through the start-up (Set_Prm) of the PB-DP slave these parameters are directly propagated to the corresponding IOL-D address (AL service "Write-Request (index 1, subindex 0)").

Figure 40 is showing the mapping principle of an IO-Link system onto "n" subsequent PB-DP slots. It is the basis for the further descriptions in this specification containing details of aspects.

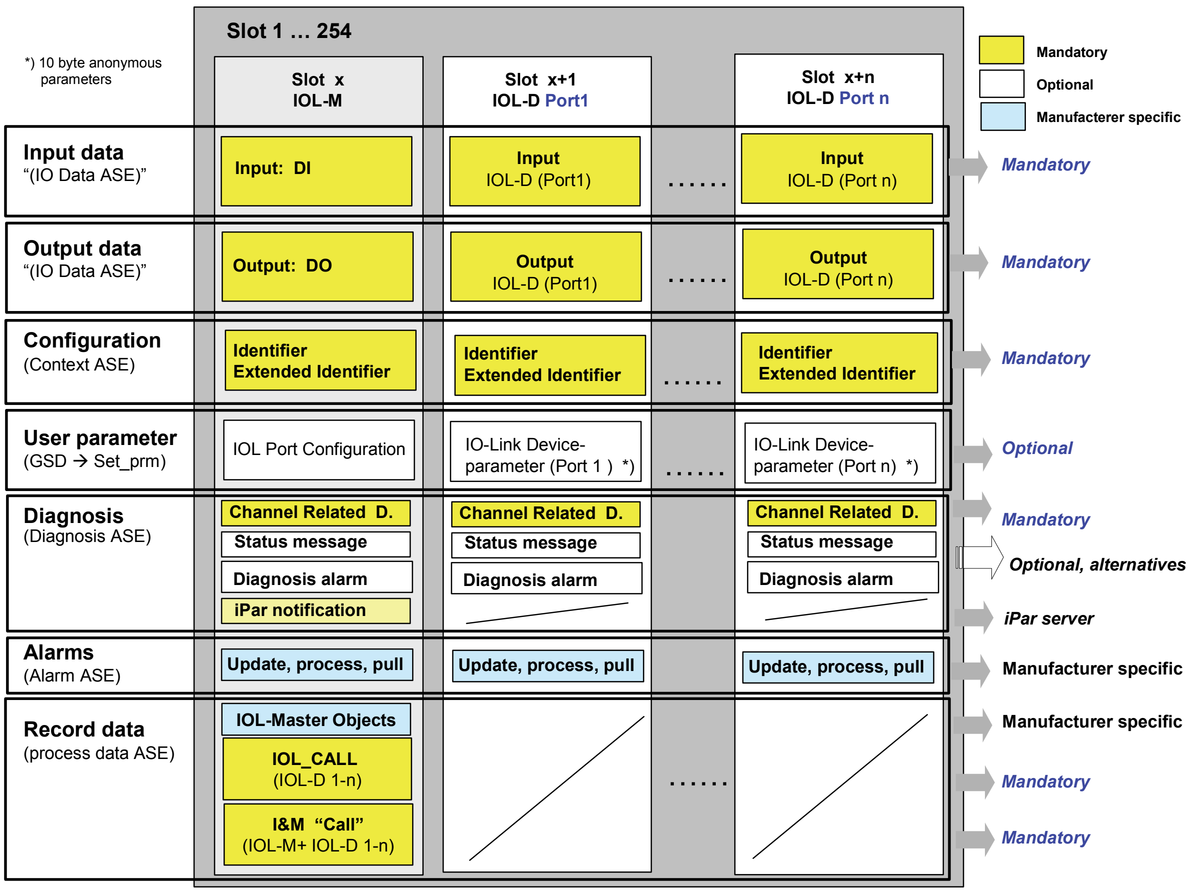


Figure 40 — Multi-slot mapping (PB-DP)

Mapping rules:

- An IO-Link system can be mapped onto any slots according to the algorithm shown in Figure 40. Slot 0 is not permitted.
- Output data of the IOL-M (Digital output and DO) are mapped onto the output address space of the particular IOL-M slot. Output data of the IOL-D are mapped onto the associated slot (IO Data ASE). See details in 7.3.4.

- Input data of the IOL-M (Digital Input and DI) are mapped onto the input address space of the particular IOL-M slot. Input data of the the IOL-D are mapped onto the associated slot (IO Data ASE). See details in 7.3.4.
- A configuration check for the particular slot is executed during start-up of the PB-DP slave (Context ASE).
- GSD based User Parameters are transmitted during start-up of the PB-DP slave (Context ASE). It is highly recommended not to transmit IO-Link specific User Parameters (IOL-M and IOL-D). Optionally it is possible to transmit the port configurations of an IOL-M and the 10 bytes anonymous parameters for IOL-Ds.
- Diagnosis information of the IOL-Ds or the IOL-M respectively is mapped to C-R-D and optionally to "Status messages" (Diagnosis ASE) of the corresponding slot. See details in 7.4.
- Usage of the "Alarm model" for diagnosis information is manufacturer specific and is not standardized herein (Alarm ASE).
- Access to IOL-Ds (including I&M data) as well as to the I&M data of the IOL-M is realized through "Record data" (CALL and IOL_CALL in Process data ASE). See details in 9.
- Access to IO-Link master objects is possible through special IOL-M "Record data" (Process data ASE). See details in 8.

From an IOL-D point of view the following behavior is defined:

- The IOL-M (AL service "Get_Input") is reading cyclically the input data of an IOL-D and inserts them into the input data area of the corresponding slot (IO data ASE).
- Output data within a particular slot of the PB-DP slave (IO Data ASE) are cyclically transfered to the corresponding IOL-D (AL service "Set_Output").

7.1.3 Check Configuration

The configuration items (Configuration Identifier) are specifying the input / output data structure types per slot to be exchanged cyclically (MS0). The PB-DP master class 1 is managing the cyclic communication based on the Configuration Identifiers.

The following formats in Table 7 can be used for the integration of IO-Link systems. See details in the IEC 61158 ([13] [15]).

Table 7 — Configuration Identifier

Configuration Identifier	Structure
Identifier_Format	Cfg_Identifier
Special_Identifier_Format	Special_Cfg_Identifier, [Length_Octet] , [Length_Octet], [Manufacturer_Specific_Data*]
Extended_Special_Identifier_Format	Special_Cfg_Identifier,[Extended_Length_Octet], [Extended_Length_Octet], [Data_Type*], [Manufacturer_Specific_Data*] Extended_Length_Octet fields shall always start with fields that indicate output data if present.

All the formats are describing the following attributes/features:

- Length of input/output data per slot
- Type (Input, Output) or format extension
- Structural units (byte/ word)
- Consistency checking mode
- Extensions (data types, manufacturer specific data)

Configuration check: At every start-up (Check Configuration), the PB-DP slave is checking the consistency of the selected Configuration Identifiers of a particular project. In case of inconsistencies it casts a diagnosis message "IOLConfigurationFault" (7.4.1.3.1).

NOTE Inconsistencies with the Configuration Identifier of the I/O data length (Identifier_Format, Special_Identifier_Format, and Extended_Special_Identifier_Format) and the actual IO-Link configuration lead to a diagnosis message "IOLConfigurationFault" by the IOL-M.

7.2 Address model of PROFINET IO

Basis for the mapping of IO-Link functions onto PN-IO is the node address/API/slot/subslot/index address model, permitting access to the PROFINET objects IO-Device, API, Slot, Subslot, and channel. The IO-Link integration (IOL-M with its associated IOL-Ds) is using this model.

- All the IO-Link functions can be addressed via subslot number and index.
- All the diagnosis information can be traced back through subslot and channel number.

7.2.1 Single-subslot mapping

The complete IO-Link system (IOL-M with its associated IOL-Ds) is mapped onto one *single subslot* of the PN-IO device. This subslot is the only access point to the IOL-M.

The single-slot mapping is the *preferred* and *highly recommended* method for the integration of IO-Link. It is particularly suitable for physically modular slave devices. Basically, the single-slot mapping can be used for both the physically modular and the compact PN-IO device.

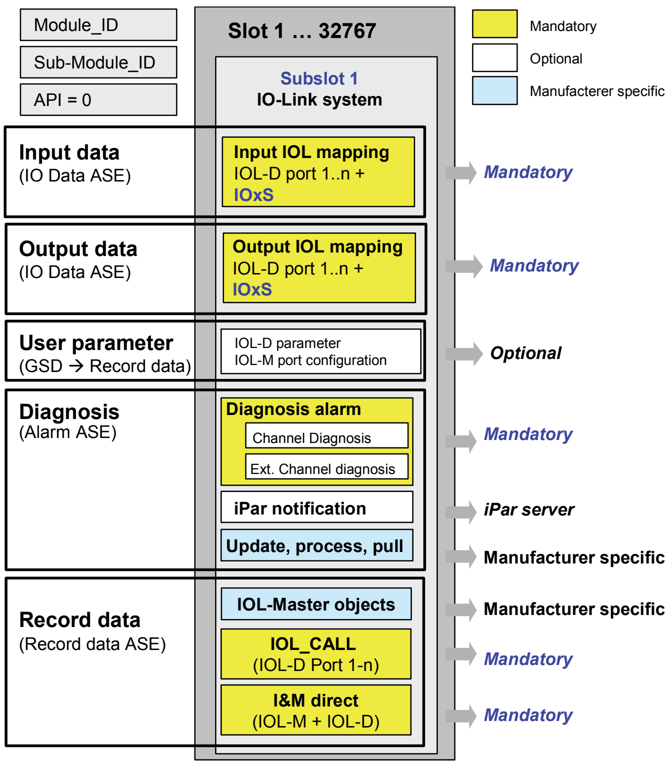


Figure 41 — Single-subslot mapping (PN-IO)

Mapping rules:

- An IO-Link system can be mapped onto any slot from 1 to 32767 within the address schema of the PN-IO device. Slot 0 is not permitted.
- Within this slot the subslot 1 is reserved for the IO-Link system.
- The standard API (API=0) shall be used for the IO-Link integration.
- Output data of all the IOL-Ds and of the IOL-M are mapped onto the output address space of subslot 1 of the PN-IO device (IO Data ASE). See details in 7.3.3.
- Input data of all the IOL-Ds and of the IOL-M are mapped onto the input address space of subslot 1 of the PN-IO device (IO Data ASE). See details in 7.3.3.
- A configuration check is executed during start-up of the PN-IO device. The consistency of Vendor_ID, Device_ID, API, Modul_Ident_Number and Submodul_Ident_Number is checked while establishing the Application Relation (Context ASE).
- GSD based User Parameters (records) are transmitted during start-up of the PN-IO device. It is highly recommended not to transmit IO-Link specific User Parameters (IOL-M and IOL-D). Optionally it is possible to transmit the port configurations of an IOL-M and the 10 bytes anonymous parameters for IOL-Ds.
- Diagnosis information of the IOL-Ds or the IOL-M respectively is mapped to ChannelDiagnosis (C-R-D) or ExtChannelDiagnosis and casted via alarms (Alarm ASE). This diagnosis information is readable via records also. See details in 7.4.
- Access to IOL-Ds (IOL_CALL) is realized through "Records" (Record data ASE). See details in 9.
- Access to I&M data of the IOL-M or IOL-D is realized through "Records" (Record data ASE). See details in 7.5.2
- Access to IO-Link master objects is possible through special IOL-M "Records" (Record data ASE). See details in 8.

7.2.2 Multi-subslot mapping

The IOL-M itself is mapped to a particular slot x/subslot 1. The IOL-Ds connected to the ports are mapped to the subsequent subslots. The subslot number of an IOL-D results from the subslot number of the IOL-M plus the port number.

The multi-slot mapping differs from the single-subslot mapping essentially by the IOL-D-wise mapping of input/output data (cyclic data transfer) and diagnosis or alarms. Record data are only mapped onto the subslot of the IOL-M.

Anonymous Parameter: In cases where optionally 10 bytes of IOL-D anonymous parameters are transmitted through the start-up of the PN-IO device these parameters are directly propagated to the corresponding IOL-D address (AL service "Write-Request (index 1, subindex 0)").

Figure 42 is showing the mapping principle of an IO-Link system onto "n" subsequent PB-DP slots. It is the basis for the further descriptions in this specification containing details of aspects

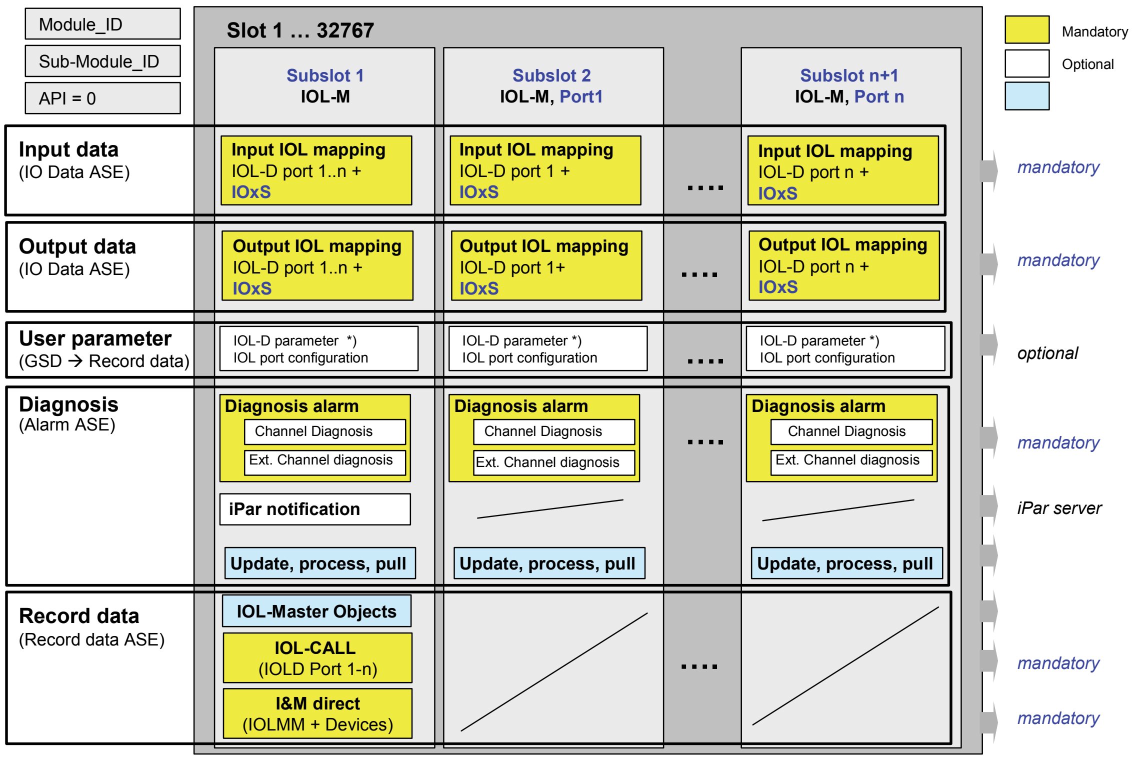


Figure 42 — Multi-subslot mapping (PN-IO)

Mapping rules:

- An IO-Link system can be mapped onto any slot from 1 to 32767 within the address schema of the PN-IO device. Slot 0 is not permitted.
- Within this slot the subslot 1 is reserved for the IOL-M.
- The standard API (API=0) shall be used for the IO-Link integration.
- Output data of the IOL-M are mapped onto the output address space of subslot 1 of the PN-IO device (IO Data ASE). See details in 7.3.4.
- Output data of the IOL-Ds are mapped onto the output address space of subslot (1+ port number) of the PN-IO device (IO Data ASE). See details in 7.3.4.
- Input data of the IOL-M are mapped onto the input address space of subslot 1 of the PN-IO device (IO Data ASE). See details in 7.3.4.
- Input data of the IOL-Ds are mapped onto the input address space of subslot (1+ port number) of the PN-IO device (IO Data ASE). See details in 7.3.4.
- A configuration check is executed during start-up of the PN-IO device. The consistency of Vendor_ID, Device_ID, API, Modul_Ident_Number and Submodul_Ident_Number is checked while establishing the Application Relation (Context ASE).
- GSD based User Parameters (records) are transmitted during start-up of the PN-IO device. It is highly recommended not to transmit IO-Link specific User Parameters (IOL-M and IOL-D). Optionally it is possible to transmit the port configurations of an IOL-M and the 10 bytes anonymous parameters for IOL-Ds.
- Diagnosis information of the IOL-Ds or the IOL-M respectively is mapped to ChannelDiagnosis (C-R-D) or ExtChannelDiagnosis and casted via alarms (Alarm ASE). This diagnosis information is readable via records also. See details in 7.4.
- Access to IOL-Ds is realized through "Records" (Record data ASE). See details in 9.
- Access to the I&M data of the IOL-M or IOL-D is realized through "Records" (7.5.2).

- Access to IO-Link master objects is possible through special IOL-M "Records" (Record data ASE). See details in 8.

From an IOL-D point of view the following behavior is defined:

- The IOL-M (AL service "Get_Input") is reading cyclically the input data of an IOL-D and inserts them into the input data area of the corresponding subslot (IO data ASE).
- Output data within a particular subslot of the PN-IO device (IO Data ASE) are cyclically transfered to the corresponding IOL-D (AL service "Set_Output").

7.2.3 Module Ident Number and API

API: There is no special API defined for IO-Link submodules. This means that addressing submodules not conforming to a dedicated (elsewhere defined) API *shall* be carried out with API = 0.

Module Ident Number: This number is manufacturer specific and thus no definitions exist in this specification. A Module Ident Number references an IO-Link system.

Submodule Ident Number: This number is manufacturer specific and thus no definitions exist in this specification.

Within the GSD file of a remote I/O system every IO-Link system describes a "subset" of submodules (Submodule Ident Number).

NOTE During start-up, the PN-IO device will check the consistency of the Modul Ident Number and the Submodule Ident Number.

7.3 I/O data mapping

The input and/or output data of the IOL-D at a particular port are inserted in the input and/or output fieldbus PDU of a PB-DP slave or PN-IO device (IO data ASE) according to the configured or implicite mapping information (port configuration) known to the IOL-M.

7.3.1 IOPS/ IOCS of PN-IO

Together with both input data and output data a *qualifier* (IOPS, IOCS) is transmitted per sub-module, which indicates the validity of the data. The mapping application within an IOL-M transfers the input/output data consistently from IOL-Ds to PN-IO controllers or vice versa. Hence there is consistency of input/output data per submodule for the entire transmission path. IOPS stands for "*Input Output object Provider Status*" and IOCS for "*Input Output object Consumer Status*".

Input Data and initial value concept

Input data of an IOL-D are continuously acquired and transmitted to the PN-IO Controller according to the actual mapping information.

The IOPS of the inputs shows the availability of cyclic data of a submodule from an IOL-M's point of view. With IOPS = “Good”, the last received input data block shall be forwarded to the IO controller. The IOCS of the inputs *should* be ignored.

Output Data and substitute values

Output data are continuously transmitted from the IOL-M "transfer memory" to the IOL-D.

The status of IOPS shall be transmitted to the IOL-Ds in the following manner:

- When IOPS changed from "BAD" to "GOOD", an IOL-M sends the "Master Command" = 0x98 (Process output data valid) to all associated IOL-D.

- When IOPS changed from "GOOD" to "BAD", an IOL-M sends the "Master Command" = 0x99 (Process output data invalid or not available) to all associated IOL-D.

These "Master Commands" are executed via IO-Link write services within the application layer: port =x, index =2, subindex =0, data =0x98 or 0x99.

The PN-IO substitute value concept shall not be used in the context of IO-Link as IO-Link provides its own substitute value concept via IODD and the PCT (11.10).

7.3.2 Clear state of PB-DP

PROFIBUS DP in its state "Clear" is transmitting zero output data (="0") to the DP slaves and thus to the IOL-M. These data are propagated to the IOL-D as well. In addition, the respective status (Clear / Operate) of PB-DP is signalled to the IOL-D via an event message as follows:

- When "Clear" changed to "Operate", an IOL-M sends the "Master Command" = 0x98 (Process output data valid) to all associated IOL-D.
- When "Operate" changed to "Clear", an IOL-M sends the "Master Command" = 0x99 (Process output data invalid or not available) to all associated IOL-D.

These "Master Commands" are executed via IO-Link write services within the application layer: port =x, index =2, subindex =0, data =0x98 or 0x99.

Input data of an IOL-D are continuously acquired and transmitted to the DP master according to the actual mapping information. Output data are continuously transmitted from the IOL-M "transfer memory" to the IOL-D. Bumpless changes from substitute to actual data cannot be assured due to missing substitute features with PB-DP.

7.3.3 Single-(sub)slot I/O

Figure 43 is demonstrating the principles of the mapping procedure. In fact there is no difference between PB-DP und PN-IO. The mapping application of an IOL-M copies the entire input and/or output data of all the IOL-D to the user defined areas of the fieldbus PDU of PB-DP (slot x) or PN-IO (subslot 1). Its mapping information is created by the configuration tool (PCT) and conveyed to the IOL-M.

The corresponding information coding is manufacturer specific and not defined in this specification.

Features of the mapping application:

- Pin 2 signals and /or Port Qualifiers (PQ) may be mapped onto the input data (11.4)
- Possible extra bytes within the PB-DP or PN-IO input PDU shall be padded by the IOL-M with 0x00.
- Possible extra space within the PB-DP or PN-IO output PDU is not relevant.
- The internal data types of an IOL-D data structure are not known to the mapping application in the IOL-M. The structure is just copied.
- Without any mapping information the IOL-M supposes default digital input (DI) configuration. I.e. *one* bit per port.
- The input/output data of an IOL-D can be freely mapped to a position within the fieldbus PDU space (Byteoffset.Bitoffset) via the PCT.
- The IO-Link process data services (Application Layer: Get_Input, Set_Output) are linked to the "IO data ASE" services of PB-DP or PN-IO.

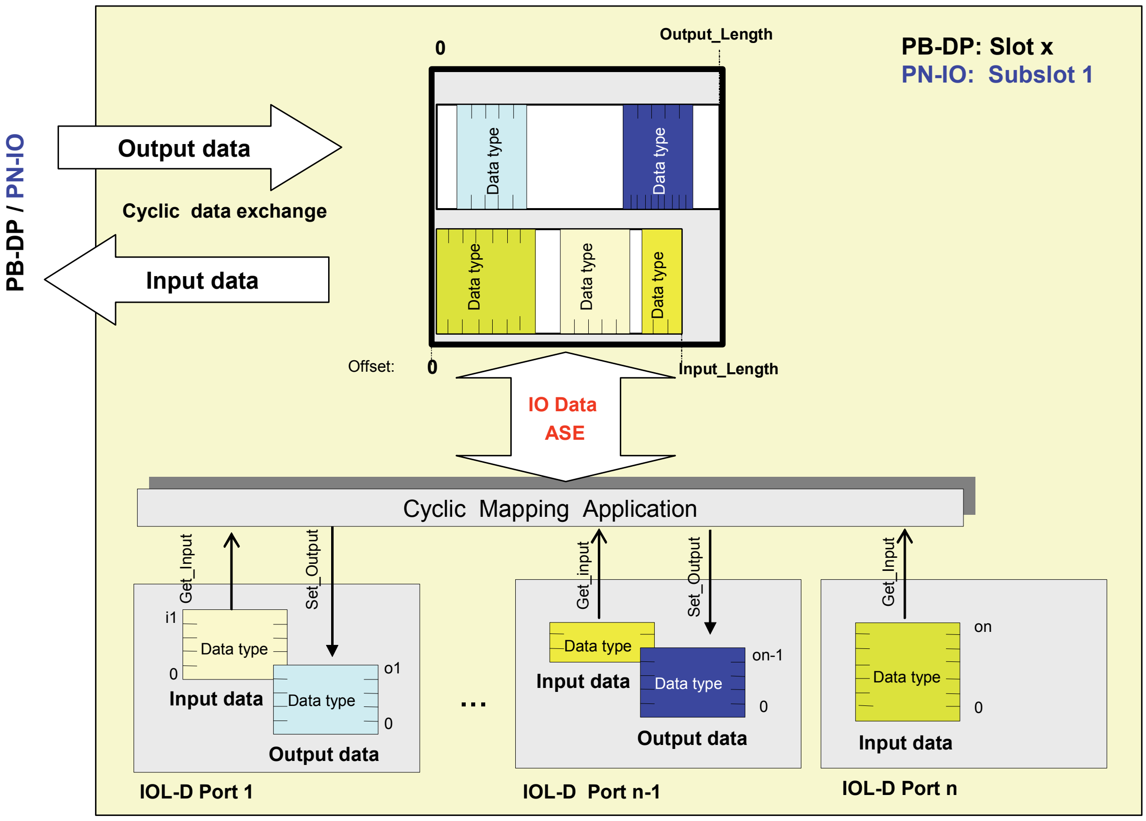


Figure 43 — Cyclic data mapping "single slot"

7.3.3.1 IOxS of PN-IO (single slot)

The rules for the creation of IOPS for the input data of a submodule are defined in the following Table 8.

Table 8 — IOPS definitions for input data

Situation	IOPS	Definition	Source /Agent
IOL-MM pulled	“Bad”	IOL-MM is pulled out or not available	Pull: IOPS “Bad” is generated by slot or IO-Device or IO-Controller IOL-M during start-up of IOL-Ds
IOL-MM in operation	“Good”	IOL-Ds are started. Cyclic data exchange of input/output data is running.	IOL-M as soon as IOL-D startups are terminated

The rules for the creation of IOPS for the output data of a submodule are defined in the following Table 9.

Table 9 — IOPS definitions for output data

Situation	IOPS	Reaction	IOCS
IO-Controller provides IOPS	“Bad”	Provision of substitute values (manufacturer specific)	"Good“
IO-Controller provides IOPS	“Good”	Output data are transmitted to the IOL-D	“Good”
IOl_MM pulled	“Bad”	IOL-MM not available	“Bad”

7.3.4 Multi-(sub)slot I/O

Figure 44 is demonstrating the principles of the direct mapping procedure. With PB-DP or PN-IO a particular I/O address space is provided for each slot or subslot. The I/O data of IOL-D ports are inserted into individual slots or subslots (x, x+1, x+2, etc.) without an offset (byte offset = "0", bit offset = "0").

- Pin 2 signals and /or Port Qualifiers (PQ) may be mapped onto the input data (11.4)
- Possible extra space within the PB-DP or PN-IO input PDU will be padded by the IOL-M with 0x00.
- Possible extra space within the PB-DP or PN-IO output PDU is not relevant.
- The internal data types of an IOL-D data structure are not known to the mapping application in the IOL-M. The structure is just copied.
- No explicite mapping information (configuration) is required.
- The IO-Link process data services (Application Layer: Get_Input, Set_Output) are linked to the "IO data ASE" services of PB-DP or PN-IO.

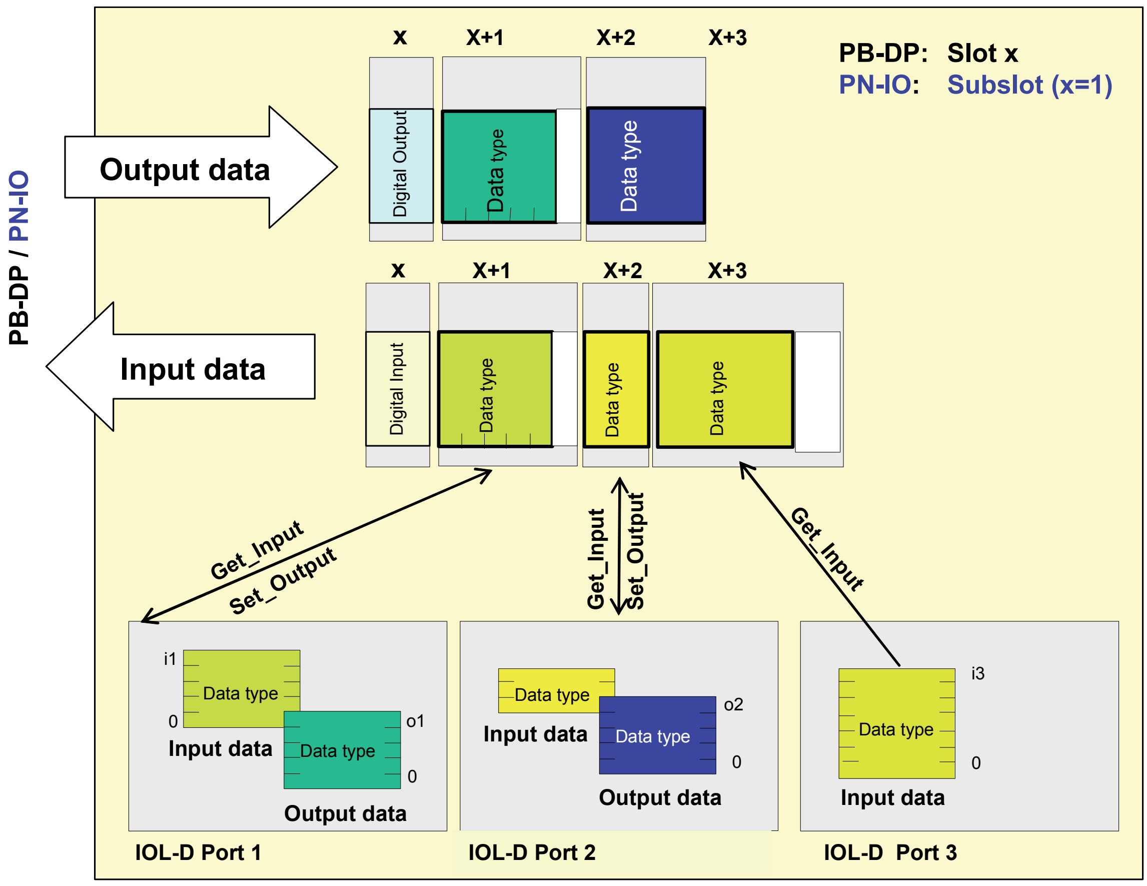


Figure 44 — Cyclic data mapping "multi (sub)slot"

7.3.4.1 IOxS of PN-IO (multi slot)

The rules for the creation of IOPS for the input data of a submodule are defined in the following Table 10. During start-up of the IOL-M, IOPS is set "bad" until the first correct mapping and valid input data could be achieved.

Table 10 — IOPS definitions for input data

Situation	IOPS	Definition	Source /Agent
IOL-MM pulled	“Bad”	IOPS of all the relevant submodules are set "Bad".	Pull: IOPS “Bad” is generated by slot or IO-Device or IO-Controller
Input data of an IOL-D are communicated and mapped correctly	“Good”	IOPS of the corresponding submodule is set "Good"	IOL-M
Input data of an IOL-D are not communicated or mapped correctly	“Bad”	IOPS of the corresponding submodule is set "Bad"	IOL-M (submodule)

The rules for the creation of IOPS for the output data of a submodule are defined in the following Table 11.

Table 11 — IOPS definitions for output data

Situation	IOPS	Reaction	IOCS
IO-Controller provides IOPS	“Bad”	Provision of substitute values (manufacturer specific)	"Good“
IO-Controller provides IOPS	“Good”	Output data are transmitted to the IOL-D	“Good”
IOl_MM pulled	“Bad”	IOL-MM not available	“Bad”

7.4 Diagnosis / events

IOL-Ds and the IOL-M can create "events" such as *errors*, *warnings*, or *messages* that are mapped here onto PB-DP or PN-IO mechanisms respectively. These "events“ indicate a change of static diagnosis information within an IOL-D (mode = coming or going) or convey a singular information ("single shot").

7.4.1 PB-DP diagnosis

The following PB-DP methods are used with IO-Link to transmit diagnosis /event information.

- Channel related diagnosis (C-R-D)
- Status Message
- iPar notification (7.4.1.4.2)

Figure 25 presents an overview of the PB-DP possibilities and the methods in use for IO-Link. The "alarm" model is not specified here but may be used manufacturer specific.

"Coming" or "Going" IO-Link events

In these cases the IOL-M generates and transmits the corresponding PB-DP diagnosis message for a "coming" event and waits until the associated "going" event occurred or a failure (permanent malfunction) of the IOL-D has been recognized.

Singular IO-Link events ("single shots")

In case of a singular IO-Link event of an IOL-D, the IOL-M generates and transmits a PB-DP diagnosis message for a limited period of time.

Example: An IOL-D sends an event with ErrorCode = "xyz" and Mode = "single shot". The IOL-M then generates the corresponding PB-DP diagnosis message for a holding time of some seconds. The holding time is manufacturer specific.

7.4.1.1 Concurrent diagnosis information

Several blocks of C-R-Ds or Status Messages can be incorporated in the body of one PB-DP diagnosis message. They can represent the entire state of the IOL-M and its IOL-Ds and usually are interpreted / processed block by block.

Thus several different active IO-Link events or states can be transmitted in form of a PB-DP diagnosis message at one point in time as long as the PDU size is not exceeded.

7.4.1.2 Hierarchical diagnosis

IO-Link "Errors", "Warnings", and "Messages" are mapped in the following manner.

- It is mandatory to map all the "*Errors*" of IOL-D or IOL-M onto “Channel Related Diagnosis” of PB-DP. For each "*Error*" a simplified code (ErrorType) is transmitted (7.4.1.3).
- Optionally, all the "*Warnings*" of IOL-D or IOL-M can be mapped onto "Status Message" of PB-DP (7.4.1.4).
- Optionally, all the "*Messages*" of IOL-D or IOL-M can be mapped exclusively onto "Status Message" of PB-DP (7.4.1.4).
- Optionally, details of "error" information can be mapped onto "Status Message" of PB-DP. Included is a detailed IO-Link "*Event*" information (7.4.1.4).

7.4.1.3 Channel related diagnosis (C-R-D)

The following structure in Table 12 represents the mapping of an "*Error*" event. In case of an "active" C-R-D (device faults) the Ext_Diag flag in the first standard part (6 bytes) of the diagnosis PDU shall be set. Usually this flag is used in DP slaves for LED indicators. See [5] for details. The structure comprises

- Identifier Number (slot number)
- ChannelNumber ("0" for the IOL-M, "1…63" for theIO-Link ports)
- ErrorType (code)

Table 12 — Structure of a Channel-Related-Diagnosis block

7	6	5	4	3	2	1	0	**Octet n** (header)
1	**0**							Selection: 10: Channel related diagnosis (C-R-D)
		Identifier Number						Range (0-63) → coresponds to slot number
7	6	5	4	3	2	1	0	**Octet n+1** (channel)
1	**1**							Input Output Selection (0): reserved (1): Input (2): Output (3): Input und Output → IO-Link
		ChannelNumber						Range (0-63) → "0" for the IOL-M, "1…63" corresponds to IO-Link port
7	6	5	4	3	2	1	0	**Octet n+2** (type of diagnosis)
0	**0**	**0**						Channel type (0): unspecific → IO-Link (4): octet (1): 1 Bit (5): word (2): 2 Bit (6): 2 word (3): 4 Bit (7): reserved
			ErrorType					Range (0-31): Table 13 and Table 14

Use of the ChannelNumber:

- ChannelNumber represents the IOL-D port number and thus the connected IOL-D. Possible ChannelNumbers are 1...63.
- ChannelNumber "0" is assigned to the IOL-M. I.e. the attached C-R-D relates to the IOL-M.

7.4.1.3.1 Mapping to Channel-Related-Diagnosis

Table 13 shows the mapping of IO-Link port related incidents to PB-DP C-R-D (ChannelNumber 1...63).

Table 13 — Mapping to Channel-Related-Diagnosis (IOL-D)

ErrorType	ErrorType definition	IO-Link error codes	Source
0	Reserved	----	
1	Short circuit	Short circuit on IO-Link pins	Local
2	Undervoltage	Supply low voltage (0x5110-0x5119)	Remote
3	Overvoltage		
4	Overload	Power section – output stages (0x5410)	Remote
5	Overtemperature	Excess ambient temperature (0x4110) Excess temperature device (0x4210) Excess temperature periphery (0x4310)	Remote
6	Line break	Wrong or missing device at this port: Port configuration does not match the actual device (type) at this port, i.e. there is a device configured for this port but no actual device was detected – or the detected device type does not match the configuration. As a consequene, the process data for this port are set invalid.	Local / port configurat ion
7	Upper limit value exceeded	Excess process variable range (0x8C10) Excess measurement range (0x8C20)	Remote
8	Lower limit value underrun	Too low process variable range (0x8C30)	Remote
9	Error	All IO-Link error codes not shown in the other parts of this table and created by the IOL-M or IOL-Ds shall be mapped to this ErrorType.	Local or remote
10	Simulation active	------	
11	Reserved	-----	
12	Reserved	------	
13	Reserved	-----	
14	Reserved	------	
15	Parameter missing	-----	
16	Parametrization fault	Optional: Parameter Error (no details) (0x6230) (if not "single shot" as mentioned in [8]) Device software - data record - loss of parameter (0x6310) Device software - data record - parameter error (0x6320) Device software - data record - parameter not initialized (0x6330) Device software - data record - parameter non specific (0x6340)	Remote
17	Power supply fault	Optional: Problem with power supply of the IO-Link port (voltage	Local

ErrorType	ErrorType definition	IO-Link error codes	Source
		at IO-Link Pin "L+" too low or too high)	
18	Fuse blown / open	Optional: Power section – fuses (0x5450-0x5459)	Remote
19	Manufacturer specific	----	
20	Ground fault	----	
21	Reference point lost	----	
22	Process event lost / sampling error	---	
23	Threshold warning	---	
24	Output disabled	---	
25	Safety event	---	
26	External fault	---	
27	Manufacturer specific	---	
28	Manufacturer specific	---	
29	Manufacturer specific	----	
30	Manufacturer specific	----	
31	Temporary fault	----	
--- not used in IO-Link context			

IOL-M related diagnosis

Table 14 shows the mapping of IOL-M related incidents. These are mapped to C-R-D with ChannelNumber "0".

Actually there are only few IO-Link specific errors which are not port related. All of these shall be mapped to ErrorType "9". It is suggested to use a PCT for detailed IO-Link related problems.

The mapping of other, not typical IO-Link error and status issues onto C-R-D (with ChannelNumber = "0") is not within the scope of this document.

Table 14 — Mapping to Channel-Related-Diagnosis (IOL-M)

ErrorType	ErrorType definition	IO-Link error code	Source
6	Wire break	IOLConfigurationFault	IOL-M
9	Error	IO-Link collective fault (created by IOL-M)	IOL-M

IOLConfigurationFault

Deviations of the IO address space of an IOL-D from the preconfigured IO address space of PB-DP are leading to an ErrorType "6".

In case of single-slot mapping according Figure 43 the following criterias apply:

- Length of input PDU of PB-DP $\geq$ total length of input data of all the configured IOL-D
- Length of output PDU of PB-DP $\geq$ total length of output data of all the configured IOL-D

"IOLConfigurationFault“ shall be generated if any of these conditions is not fullfilled.

In case of multi-slot mapping according Figure 44 the following criterias apply:

- Length of the configured input (e.g. slot x): ≥ digital inputs (DI) of the IOL-Ds
- Length of the configured input (e.g. slot x+1): ≥ length of the input data of the IOL-Ds
- Length of the configured output (e.g. slot x): ≥ digital outputs (DO) of the IOL-Ds
- Length of the configured output (e.g.slot x+1): ≥ length of the output data of the IOL-Ds

"IOLConfigurationFault“ shall be generated if any of these conditions is not fullfilled.

7.4.1.4 Status Model

The data structure in Table 15 shows the coding prescription of the diagnosis status model of PB-DP. Status_message and iPar notification are using the same model. They differ in the StatusType (1: Status_message, 7: iPar notification) and in the status data:

Status_Type =1(Status_message)
Status data contains the IOL_StatusDataDescription (IO-Link Event)

Status_Type =7(iPar notification)
Status data contains the iPar notification data

Table 15 — Structure of a Status_message block

7	6	5	4	3	2	1	0	**Octet n (Header)**
0	**0**							Selection: 00: Device related diagnosis
		Block_Length						Range (0-63) → number of octets of the following Extended diagnosis block including the header octet
7	6	5	4	3	2	1	0	**Octet n+1 (Status_Type)**
1								Status Model
	Status_Type							Status_Type (0): Reserved (8-29): Reserved (1): **Status_message** (30): PrmCmdAck (2): Modul_Status (31): Red_Stat (3): DxB_Link_Status (32-125): manufacturer specific (4-6): Reserved (126): PA-Profile (7): **iPar notification** (127): Reserved
7	6	5	4	3	2	1	0	**Octet n+2 (Slot_Number)**
								Slot_Number Range (0-254) → data type: unsigned8
7	6	5	4	3	2	1	0	**Octet n+3 (Status_Specifier)**
Reserved (set to 0)								Status_Specifier (0): no further differentiation (2): status disappeared (1): status appeared (3): reserved
7	6	5	4	3	2	1	0	**Octet n+4 ... (Status data description)**
								(Status_Type =1) IOL_StatusDataDescription (Status_Type =7) iPar notification

7.4.1.4.1 IOL_StatusDataDescription

An IOL_StatusDataDescription block describes an IO-Link event and its attributes in an unambiguous manner (Table 16):

- EventCode
- ChannelNumber
- EventQualifier

Table 16 — IOL_StatusDataDescription block

IOL_StatusDataDescription	Size	Coding	Notes
EventCode	2 Octets	(0…65535)	IO-Link EventCode according to the IO-Link specification [8]
ChannelNumber	1 Octet	(0): IOL-M (1…63): Port No. (IOL-D)	Channel/ port selector
EventQualifier	1 Octet	See Figure 45	Event classification according to the IO-Link specification [8]

MODE		TYPE		Res.	INSTANCE		
7	6	5	4	3	2	1	0

Figure 45 — EventQualifier

In the following the bit structure of Figure 45 is defined:

Bits 0 to 2: INSTANCE

These bits indicate the instance triggered by the event (event source) and, therefore, the instance forming the basis for the evaluation of the event (Table 17).

Table 17 — EventQualifier (INSTANCE)

Value	Definition
0	unknown
1	Phy
2	DL
3	AL
4	Application
5…7	reserved

Bit 3: reserved

This bit is reserved and shall be set "0".

Bits 4 to 5: TYPE

These bits indicate the event category (Table 18).

Table 18 — EventQualifier (TYPE)

Value	Definition
0	reserved
1	Information
2	Warning
3	Error

Bits 4 to 5: MODE

These bits indicate the event mode. Permitted values for MODE are listed in Table 19.

Table 19 — EventQualifier (MODE)

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

7.4.1.4.2 iPar notification

Die iPar notification consists of 4 blocks with 4 Octets (iPar0, iPar1, iPar2, and iPar3). The Coding of the iPar notification is shown in Table 20.

Table 20 — Coding of the iPar notification

iPar specifier	Name	Octet 3	Octet 2	Octet 1	Octet 0	Definition
iPar0	iPar_Req_Header	SR_Version	Reserved	Reserved	SR_Type	Type of iPar-Server request (Unsigned32)
iPar1	Max_Segm_Size	0x00h	0x00h	0x00h	0…234	Maximum permitted net size of a segment in octets (Unsigned32)
iPar2	Transfer_Index	0x00h	0x00h	0x00h	0…254 (255)	Index for the read/write record transfer (Unsigned32)
iPar3	Total_iPar_Size					Total length of iParameter octets (Unsigned32)
NOTE 1: Reserved: Shall be coded "0".						
NOTE 2: The parameter "Max_Segm_Size" may be larger than 234 octets with PN-IO. It can comprise up to 2^{22}-1 octets.						
NOTE 3: A "Transfer_Index" of 255 may conflict with other services such as a CALL of I&M functions.						
NOTE 4: The parameter "Transfer_Index" may be larger than 255 with PN-IO (up to 65535).						
NOTE 5: A replacement device may not know the correct size of iParameter of its predecessor. In this case the notification for Restore may contain "Total_iPar_Size = 0", which means the iPar-Server shall download the complete iParameter data set.						

The parameter "SR_Version" shall be set to 0x01h. The parameter "SR_Type" shall be coded as shown in Figure 46.

7	6	5	4	3	2	1	0	
*	*	*	*	*	*	0	0	Reserved (4.3 shall be observed)
*	*	*	*	*	*	0	1	Save (Upload)
*	*	*	*	*	*	1	1	Restore (Download)
*	*	*	*	↑	↑			Reserved (4.3 shall be observed)
*	*	*	0	*	*	*	*	Transfer per one read/write record
*	*	*	1	*	*	*	*	Segmented transfer per push/pull mechanism
↑	↑	↑						Reserved (4.3 shall be observed)

Figure 46 — Coding of SR_Type

After sending an iPar-Server request the IOL-M is waiting 2^{18} ms (approximately 4,4 minutes) for the "Save" or "Restore" service to be executed completely. After the expiration of this time, it launches an appropriate diagnosis message.

7.4.1.5 PB-DP Alarms

The mapping of IO-Link events onto the PB-DP alarm model is manufacturer specific and not defined herein.

7.4.2 PN-IO Diagnosis

Figure 26 presents an overview of all the diagnosis methods for PROFINET IO specified in the IEC 61158 standards [14] and [16]. PN-IO took over the alarm model as a basic diagnosis reporting method. Thus, the C-R-D now follows the alarm model also as well as the iPar notification. Status_messages should be mapped into diagnosis alarms.

The definitions of the ErrorTypes are the same as with PB-DP but extended. The IOL-D events (errors, warnings, messages) are converted into PN-IO alarms.

"Appearing" and "disappearing" IO-Link events

In IO-Link appearing events are directly converted to "coming" alarms of PN-IO (Channel-Related-Diagnosis or Extended-Channel-Related-Diagnosis). This is valid for disappearing events also.

"Singular" IO-Link events ("single shot")

The IOL-M generates a "coming" alarm of PN-IO in case of a "singular" event from an IOL-D ("single shot"). After a holding time of some seconds the IOL-M generates a corresponding "going" alarm.

7.4.2.1 Mapping alternatives

The following mapping alternatives for IO-Link events can be used optionally.

1. Mapping onto Channel-Related-Diagnosis (USI =0x8000). This generates an identical view as with PB-DP. I.e. only IO-Link errors are converted onto ErrorTypes 0...31.
2. Mapping to Extended-Channel-Related-Diagnosis (USI =0x8002). I.e. all the IO-Link specific events can be conveyed to the IO Controller using (Channel)ErrorTypes 0...31 and ExtChannelAddValue vor detailed information.

7.4.2.2 PN-IO diagnosis alarm mechanisms

The PN-IO Alarm ASE shall be used for the alarm mechanism. The Alarm-PDU for the PN-IO diagnosis alarm mechanism is defined in Table 21 below.

Table 21 — PN-IO alarm structure

Attribute	Definitions
AlarmType	“Diagnosis” if a diagnosis event (from IO-Link) appears or disappears Special case: When all diagnosis disappears the “Diagnosis disappears” offers an optimized way to transmit this information.
API	0x0000 (Default API)
SlotNumber	Slot number of the IOL-MM
SubslotNumber	Subslot number of that submodule, the diagnosis is related to
ModuleIdentNumber	ModuleIdentNumber related to the IOL-MM
SubmoduleIdentNumber	SubmoduleIdentNumber related to the IOL-D / IOL-MM
AlarmSpecifier	Diagnosis state info
UserStructureIdentifier	0x8000 (ChannelDiagnosisData) 0x8002 (ExtendedChannelDiagnosisData)
AlarmData	Structure depends on UserStructureIdentifier USI 0x8000 : see 7.4.2.2.1 USI 0x8002: see 7.4.2.2.2

The structure of the AlarmData is depending on the UserStructureIdentifier and is defined in the following sections.

7.4.2.2.1 Channel-Related-Diagnosis (USI=0x8000)

When using the mapping alternative 1 (USI=0x8000) the coding shall be according Table 22.

Table 22 — Coding of Channel-Related-Diagnosis (USI=0x8000)

Attribute	Definitions
ChannelNumber	(1...63) IO-Link port number 0x8000 IOL-M
ChannelProperties.Type	0x00 or according [16]
ChannelProperties.Reserved	0
ChannelProperties.maintenanceRequired	Table 23 — Valid combinations of "ChannelProperties"
ChannelProperties.maintenanceDemanded	
ChannelProperties.Specifier	
ChannelProperties.Direction	0x00 or according [16]
ChannelErrorType	Corresponds to ErrorTypes in Table 13

In respect to the mapping of IO-Link events onto ChannelErrorTypes the definitions in 7.4.1.3.1 apply. "ErrorType" represents a subset of "ChannelErrorType" ([15] versus [16]).

The dependencies of the "ChannelProperties" fields are specified in Table 23.

Table 23 — Valid combinations of "ChannelProperties"

Maintenance-Required Bit 9:	Maintenance-Demanded Bit 10:	Specifier	Definition	IO-Link Representation
0x00	0x00	0x00	All subsequent Diagnosis, MaintenanceRequired, MaintenanceDemanded, and QualifiedDiagnosis disappear	---
		0x01	Diagnosis appears	ChannelErrorType 0…x
		0x02	Diagnosis disappears	---
		0x03	Diagnosis disappears but others remain	ErrorType 0…x
0x01	0x00	0x00	Reserved	Reserved
		0x01	MaintenanceRequired appears	IO-Link Warning appears ChannelErrorType 9
		0x02	MaintenanceRequired disappears	MaintenanceRequired disappears
		0x03	MaintenanceRequired disappears but other remain	IO-Link Warning appears ChannelErrorType 9
0x00	0x01	0x00	Reserved	Reserved
		0x01	MaintenanceDemanded appears	IO-Link message appears ChannelErrorType 9
		0x02	MaintenanceDemanded disappears	IO-Link message disappears ChannelErrorType 9
		0x03	MaintenanceDemanded disappears but others remain	MaintenanceDemanded disappears but others remain

7.4.2.2.2 Extended-Channel-Related-Diagnosis (USI=0x8002)

When using the mapping alternative 2 (USI=0x8002) the coding shall be according Table 24. In addition to the mapping onto ChannelErrorTypes the information of IO-Link events can be transferred transparently via the field *ExtChannelAddValue* (4 Octets).

Table 24 — Coding of Extended-Channel-Related-Diagnosis (USI=0x8002)

Attribute	Definitions
ChannelNumber	(1…63) IO-Link port number 0x8000 IOL-M
ChannelProperties.Type	0x00 or according [16]
ChannelProperties.Reserved	0
ChannelProperties.maintenanceRequired	Table 23 — Valid combinations of "ChannelProperties"
ChannelProperties.maintenanceDemanded	
ChannelProperties.Specifier	
ChannelProperties.Direction	0x00 or according [16]
ChannelErrorType	Corresponds to ErrorTypes in Table 13
ExtchannelErrorType	Type: IO-LInk
ExtChannelAddValue	See 7.4.1.4.1 IOL_StatusDataDescription

7.4.2.3 iPar notification (USI=0x8201)

The PN-IO Alarm ASE shall be used for the alarm mechanism. The iPar notification is a subelement of the more general "Upload & Retrieval" alarm. The coding of this alarm type is presented in Table 25.

Table 25 — PN-IO alarm structure for Upload &Retrieval

Attribute	Definitions
AlarmType	Upload&Retrieval
API	0x0000 (Default API)
SlotNumber	Slotnumber of the IOL-MM
SubslotNumber	Subslotnumber of the submodule, the diagnosis is related to
ModuleIdentNumber	ModuleIdentNumber related to the IOL-MM
SubmoduleIdentNumber	SubmoduleIdentNumber related to the IOL-D / IOL-MM
AlarmSpecifier	Reserved
UserStructureIdentifier	0x8201 iPar-Server 0x8200 reserved
AlarmData	Structure depends on UserStructureIdentifier USI 0x8201 iPar notification (see 7.4.1.4.2.) USI 0x8200 reserved

7.5 I&M Functions

It is the main objective of the I&M functions described herein to support the operator of automation systems at various scenarios such as configuration, parameterization, commissioning, trouble shooting, updating, etc. within the lifecycle of fieldbus devices [3].

There are no differences between the I&M data of PB-DP and PN-IO in respect of content, structure, and semantics.

Basic functions (records IM0...IM15)

- The mandatory record IM0 (read only) contains the identification of the IOL-M. It contains the parameter MANUFACTURER_ID, ORDER_ID, HARDWARE_REVISION, SOFTWARE_ REVISION, REV_COUNTER, PROFILE_ID, PROFILE_SPECIFIC_TYPE, IM_VERSION, and IM_SUPPORTED.
- Records IM1 up to IM15 are optional for IOL-M. Usually they are write-enabled. Amongst others there are parameters such as TAG_FUNCTION, TAG_LOCATION, DESCRIPTION, etc.

Profile specific functions (records IM16…IM99)

The parameter PROFILE_ID in IM0 (IO-Link = 4E00h) enables the specific usage of the records IM16 up to IM99 for IO-Link.

- IOL-D identification data such as Vendor_ID, Device_ID, Function_ID can be mapped optionally to records starting with IM16 = port 0, IM17 = port 1, etc.
- It is mandatory for an IOL-M to provide "directory" information (features) as a center point for access coordination.

Manufacturer specific functions (records IM100…IM199)

Manufacturer specific I&M functions are not specified herein. PB-DP and PN-IO are supporting the same I&M functions. However, the access methods are depending on the particular fieldbus and therefore they are described in the following sections. Also described are the profile (IO-Link) specific definitions.

7.5.1 I&M mapping with PB-DP

In order to provide a uniform access method for all the PB-DP devices, the "CALL" mechanism with its index address extensions is used for I&M. This "CALL" is generally defined in the IEC standard only for index 255 that had been reserved in previous versions.

Figure 47 is showing beyond index 255 a (sub)-index space which is called FI_Index (0…65535). I&M is using a subset hereof (65000…65199) and names it IM_INDEX.

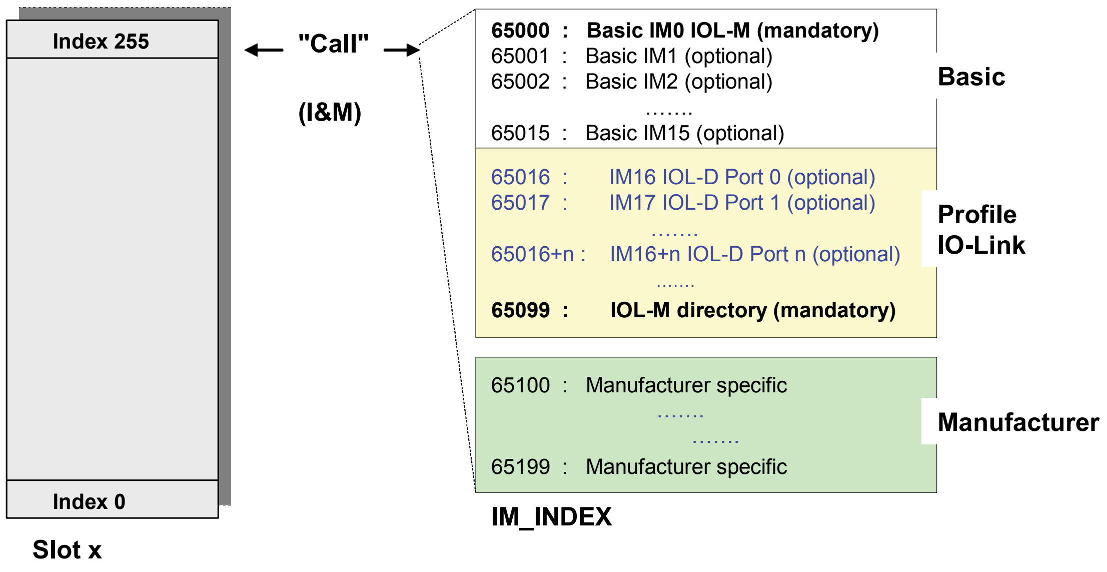

Index 255
Index 0
Slot x
"Call"
(I&M)
65000 : Basic IM0 IOL-M (mandatory)
65001 : Basic IM1 (optional)
65002 : Basic IM2 (optional)
…….
65015 : Basic IM15 (optional)
Basic
65016 : IM16 IOL-D Port 0 (optional)
65017 : IM17 IOL-D Port 1 (optional)
…….
65016+n : IM16+n IOL-D Port n (optional)
…….
65099 : IOL-M directory (mandatory)
Profile IO-Link
65100 : Manufacturer specific
…….
…….
65199 : Manufacturer specific
Manufacturer
IM_INDEX

Figure 47 — (Sub)-index space for I&M

7.5.1.1 Standard CALL procedure of IEC 61158

An I&M client is accessing the I&M functions of an IOL-M via the standard CALL procedures. The Write Call and Read Call are defined in [15]. See also [3] for additional details.

7.5.1.2 I&M profile extensions

Some special definitions are needed for profile specific use of I&M.

a) PROFILE_ID

Table 26 contains the PROFILE_ID range assigned to the IO-Link profile.

Table 26 — Profile ID for IO-Link

PROFILE_ID	Assigned to
4E00 – 4EFF	Profile "IO-Link"

b) IM_SUPPORTED (within IM0)

Bit 0 ="1" indicates the availability of profile specific I&M data, here IO-Link.

7.5.2 I&M mapping with PN-IO

With PN-IO the I&M data are directly mapped onto the index address space of the particular subslot 1 (Figure 48). I.e. it is possible with PN-IO to directly read and write IM records via *record data.submodule.index*. Following some PN-IO specifics:

- The index range for I&M data starts at 0xAFF0.
- The basic I&M data of the IOL-M is available at subslot 1 (IM0 mandatory).
- The profile specific I&M data of the IOL-M and its IOL-Ds is available at subslot 1 starting at 0xB000.
- The IOL-M directory is available at subslot 1, index 0xB063

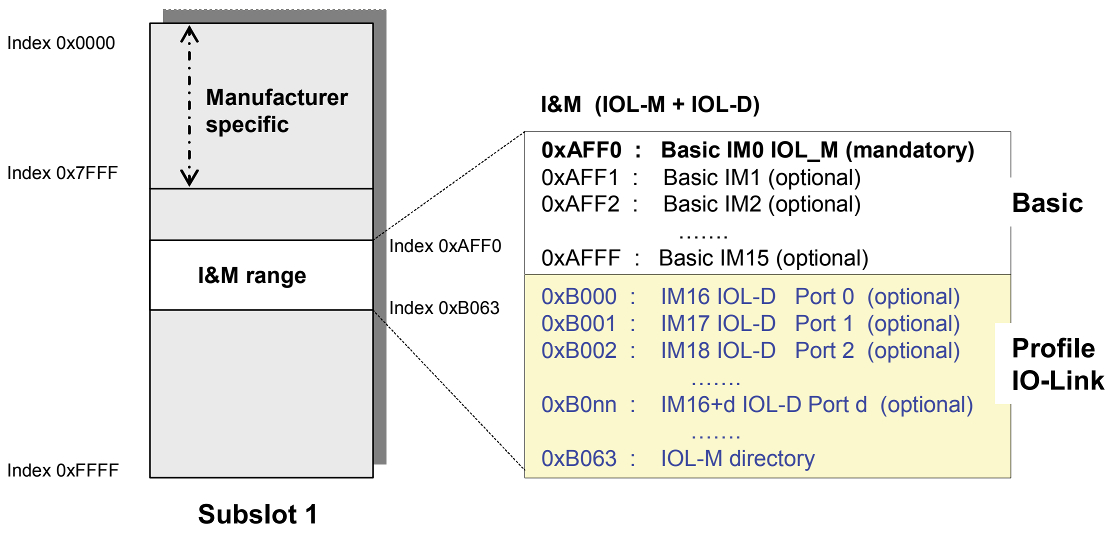

Index 0x0000
Manufacturer specific
Index 0x7FFF
I&M range
Index 0xAFF0
Index 0xB063
Index 0xFFFF
Subslot 1
I&M (IOL-M + IOL-D)
0xAFF0 : Basic IM0 IOL_M (mandatory)
0xAFF1 : Basic IM1 (optional)
0xAFF2 : Basic IM2 (optional)
....
0xAFFF : Basic IM15 (optional)
Basic
0xB000 : IM16 IOL-D Port 0 (optional)
0xB001 : IM17 IOL-D Port 1 (optional)
0xB002 : IM18 IOL-D Port 2 (optional)
....
0xB0nn : IM16+d IOL-D Port d (optional)
....
0xB063 : IOL-M directory
Profile IO-Link

Figure 48 — IM records for IO-Link with PN-IO

7.5.3 IM records for IOL-D

Figure 48 is showing how the identification objects of the connected IOL-Ds can be retrieved via the profile specific extensions IM16_IOL_D up to IM16+d ("d" = number of ports). The standard identification objects Vendor_ID, Device_ID and Function_ID of a particular IOL-D are packed into one IMx_IOL-D record and made accessible through PB-DP or PN-IO.

Table 27 — IM record representing an IOL-D

IM16 ... IM16+d	Size	IOL-D parameter	Definition
VENDOR_ID	2 Octets	Vendor_ID1, Vendor_ID2	VENDOR_ID is composed of Vendor_ID1 (low byte) and Vendor_ID2. Data type is Unsigned16.
DEVICE_ID	4 Octets	Device_ID1, Device_ID2, Device_ID3	DEVICE_ID is composed of Device_ID1 (low byte) up to Device_ID3 and of a padding byte (0x00). Data type is Unsigned32.
FUNCTION_ID	2 Octets	Function_ID1, Function_ID 2	FUNCTION_ID is composed of Function_ID1 (low byte) and Function_ID2. Data type is Unsigned16.
Reserved	10 Octets	---	---

7.5.4 IOL-M directory

The IOL-M directory object establishes a central access point to an IOL-M. For an IOL-M it is mandatory to provide this object at the fixed IM_INDEX "65099".

Figure 49 is demonstrating the principle. The object contains important information for example for the access coordination of multiple clients. It also provides references to additional information such as port configuration parameters, IO mapping structures, or iParameter logistics.

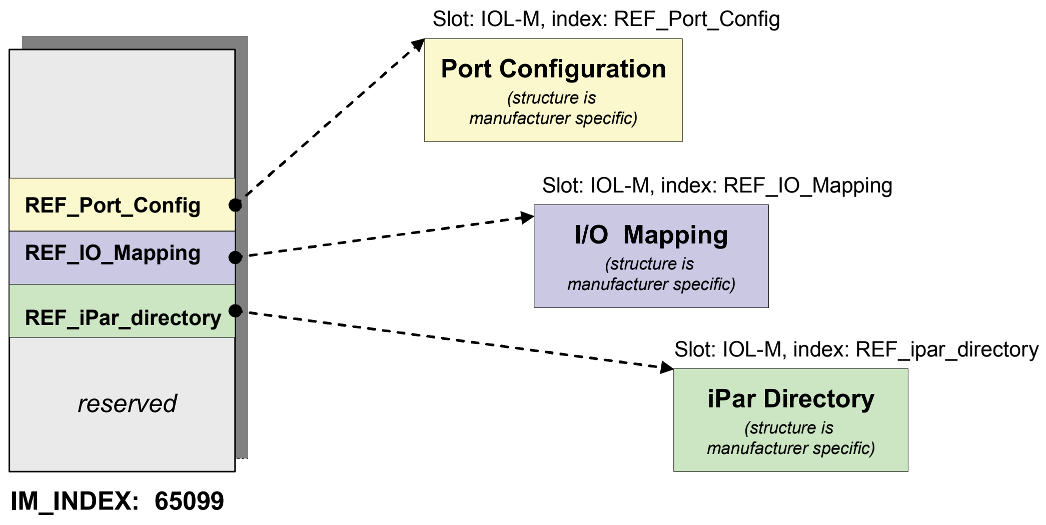

Slot: IOL-M, index: REF_Port_Config
Port Configuration
(structure is manufacturer specific)
Slot: IOL-M, index: REF_IO_Mapping
I/O Mapping
(structure is manufacturer specific)
Slot: IOL-M, index: REF_ipar_directory
iPar Directory
(structure is manufacturer specific)
REF_Port_Config
REF_IO_Mapping
REF_iPar_directory
reserved
IM_INDEX: 65099

Figure 49 — IOL-M directory concept

The coding of the IOL-M directory object is defined in Table 28. The structure of the records in REF_xxx is manufacturer specific.

Table 28 — Coding of the IOL-M directory object

Content	Size	Coding	Definitions
IOL-Link_Version	Unsigned8	0..255	IO-Link communication version
IO-Link_Profile_Version	Unsigned8	0..255	IO-Link profile version
IO-Link_Feature_Support	Unsigned32	……	Bits are indicating available features
NumberofPorts	Unsigned8	0…255	Number of supported ports
REF_Port_Config	Unsigned8	0…255	0: No "Port Configuration" data 1…255: Index of the record
REF_IO_Mapping	Unsigned8	0…255	0: No "I/O Mapping" data 1…255: Index of the record
REF_iPar_directory	Unsigned8	0…255	0: No "iPar Directory" data 1…255: Index of the record
REF_IOL_M	Unsigned8	0…255	0: No "IOL-M Parameter" data 1…255: Index of the record
Number_of_cap	Unsigned8	0…255	0…255: Number of client access points
Index_cap1	Unsigned8	1…255	Index of the client acces point 1
……..			……
Index_capn	Unsigned8	1…255	Index of the client acces point n

8 IOL-M records

An IOL-M keeps a number of records supporting the different tasks. The records are generated, maintained, and interpreted by different engineering applications or tools as shown in Figure 50.

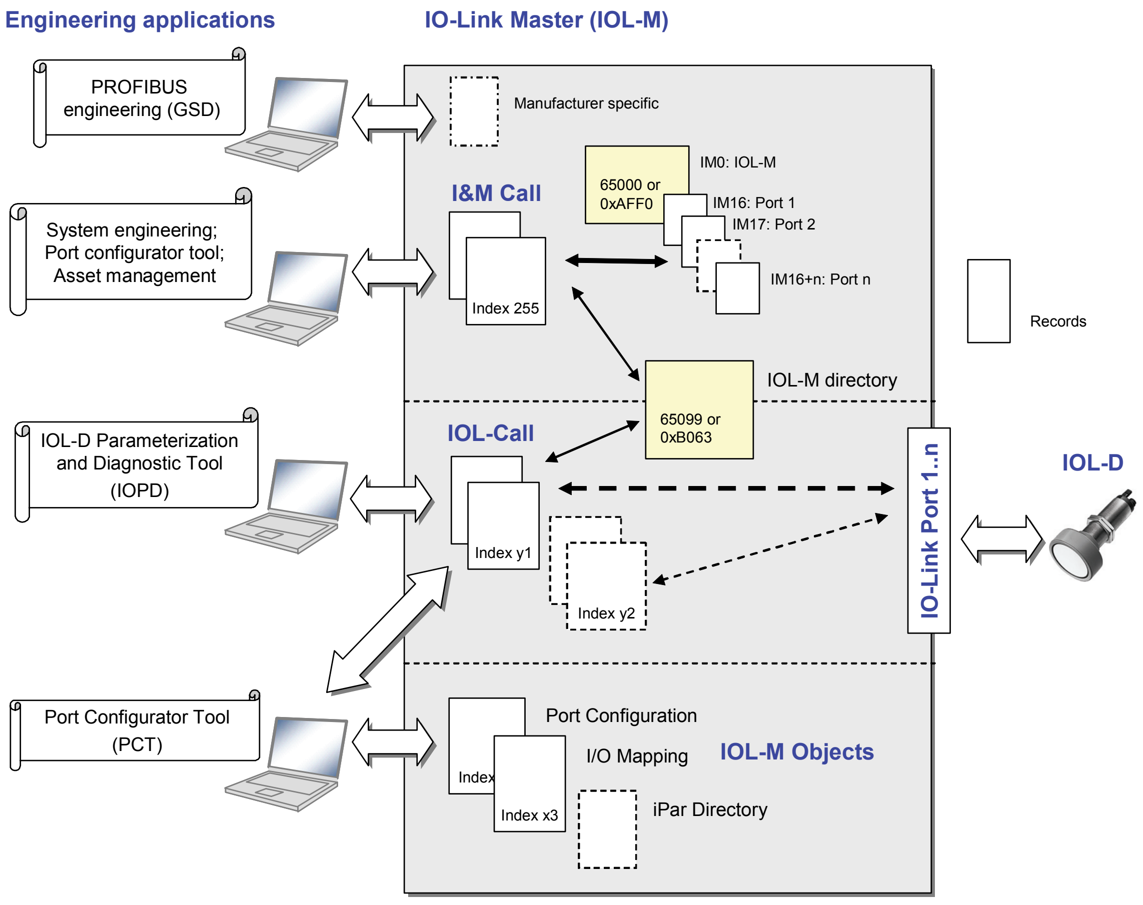


Figure 50 — IO-Link master records

I&M functions and the significant IOL-M directory object as a main entrance are specified in 7.5.

The access to IOL-Ds via PCT or individual IOL-D tools (IOPD) via IOL_CALL is specified in detail in 9.

The parameterization of the ports and the mapping of I/O data are carried out by the Port Configurator Tool (PCT). The corresponding data objects (records) are described in the following sections 8.1 and 8.2.

8.1 Record "Port Configuration"

The configuration data of all the ports 1...n is conveyed to the IOL-M via this record. The index to be used is defined in 7.5.4. The following information shall be provided:

- *Identification:* VENDOR_ID, DEVICE_ID, FUNCTION_ID
- *Physical layer*: Physics 1
- *Operational mode:* DI, DO, Scan Mode, Fallback, inactive

- *Cycle type:* free, synchronous, fixed (value)
- *Inspection level:* No check, Profile device, Vendor specific, Type equivalent, Identical device
- Others to be defined as necessary (manufacturer)

A nominal/actual comparison between a configured IOL-D and the actually connected IOL-D can be performed with the help of the *Identification* parameters.

The structure and coding of these "Port Configuration" parameters are manufacturer specific and not specified herein.

8.2 Record "I/O Mapping"

The detailed mapping (data structuring) of the I/O data onto the PB-DP and PN-IO input and output PDUs is retained in this record and conveyed to the IOL-M. The IOL-M uses this "prescription" to build-up the PDU data structures for the cyclic data exchange across the fieldbus (7.3).

The structure and coding of these "I/O Mapping" parameters are manufacturer specific and not specified herein.

8.3 Record "iPar Directory"

This record provides information about manufacturer specific I&M information mapping. This way an IOL-M manufacturer is able to propagate further IOL-D information such as vendor name, vendor URL, etc.

The structure and coding of these "iPar Directory" parameters are manufacturer specific and not specified herein.

9 Acess to IO-Link data objects and port functions

Client applications such as PCT, IOPD, and user programs in PLCs can access IO-Link data objects or initiate port functions using PB-DP and PN-IO mechanisms.

The characteristics of these IO-Link data objects are:

- Access through *IOL_Index* 0x0000 – 0x7FFF and *IOL_Subindex* 0x00 – 0xFF
- Client applications can read or write IO-Link data objects
- The maximum size of any (Index.Subindex) IO-Link data object is < 232 Byte

9.1 Applications and communication paths

The access to IO-Link data objects and port functions is necessary for the following applications and use cases:

- "Online“ parameterization of IOL-Ds via PCT, IOPDs, or other applications
- User-program-controlled parameterization of IOL-D via function blocks in PLC systems
- Service and monitoring functions (diagnosis, identification, etc.)
- Ports can be managed in many ways (initialization, wake-up, fallback, SIO-mode, etc.)

The corresponding communication paths are demonstrated in Figure 51.

- 1 to n client applications can have access to one IOL-D.

- 1 to n client applications can have access to different IOL-Ds "behind" one IOL-M.
- 1 to n client applications can have access to different IOL-Ds "behind" different IOL-M

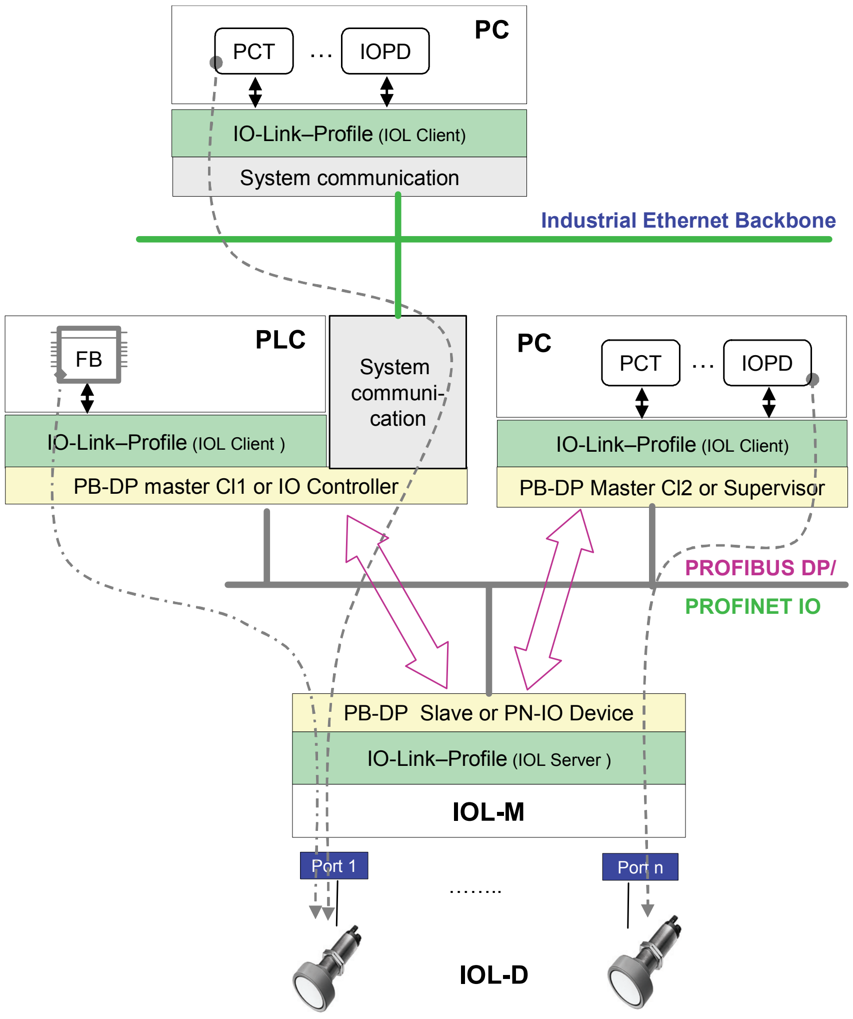

PC
PCT
…
IOPD
IO-Link–Profile (IOL Client)
System communication
Industrial Ethernet Backbone
PLC
FB
IO-Link–Profile (IOL Client)
PB-DP master Cl1 or IO Controller
System communi-cation
PC
PCT
…
IOPD
IO-Link–Profile (IOL Client)
PB-DP Master Cl2 or Supervisor
PROFIBUS DP/
PROFINET IO
PB-DP Slave or PN-IO Device
IO-Link–Profile (IOL Server)
IOL-M
Port 1
……..
Port n
IOL-D

Figure 51 — Access to ports and IOL-Ds

The following requirements result from the structures and applications:

- An IO-Link client system (engineering) can be operated directly from PB-DP or PN-IO.
- An IO-Link client system is "tunneling" IO-Link data objects to a requested IOL-D.
- Optionally several clients can concurrently access an IOL-M or an IOL-D (9.2)
- The *IOL-Client* communicates with the *IOL-Server* in an IOL-M.

The set of communication services and protocols between the *IOL-Client* and its *IOL-Server* is called *IOL-CALL*.

9.2 Client management

Actually several methods exist to achieve concurrency, i.e. several clients are communicating concurrently with one or more different IOL-Ds connected to an IOL-M. A method is described and recommended herein, which represents a good compromise between a basic (for example any communication through index 255 with PB-DP) and a comfortable but rather complex method (coordination via connection IDs). Figure 52 demonstrates the *index redirection* principle of this method.

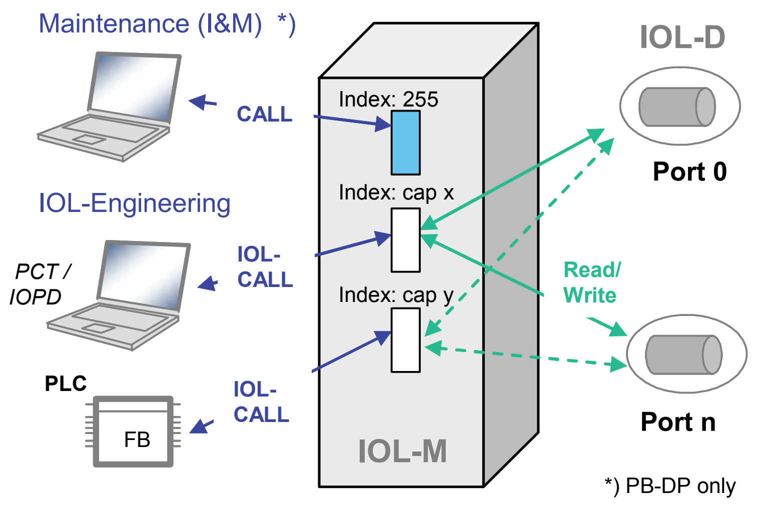


Figure 52 — Client handler with Fixed Index Redirection

The accesses of the client applications for maintenance, IOL-Engineering (PCT, IOPD), or PLC program function blocks (FB) are carried out through disjoint access points (PB-DP Index 255, Index_capx, y...). Concurrent accesses are dissolved within an IOL-M. A fixed order is recommended: e.g. IOL-Engineering → Indec_capx und PLC-FB → Index_capy (Figure 52). Basically any IOL-D can be accessed through any disjoint access point. For each port an individual access point for the access of PLC-FBs can be assigned if necessary.

The details of the method supported by a particular IOL-M (for example the *Fixed Index Redirection* method described herein) is "lodged" within the associated PCT as well as within the IOL-M directory object (7.5.4). The PCT owns the knowledge about assigned access points. Upon invocation of an IOPD tool the PCT passes over the corresponding access point ("cap") besides other address information to the IOPD.

From a programmer's point of view the corresponding access point ("cap") shall be attached to the access function block (FB).

9.3 IO-Link CALL (IOL_CALL)

The communication protocol between an IOL-Client and an IOL-Server is called *IOL-CALL* and is specified within the following sections. Characteristic is the usage of the *Client Access Points* ("cap") as specified in 9.2.

Characteristics of the IOL_CALL:

- Access to the IOL-Server is carried out through a *Client Access Point*. This "cap" is represented by an index of the IOL-M.
- The *Client Access Point*s ("cap") are defined within the IOL-M Directory Object.
- Each IOL-M provides at least one *Client Access Point* ("cap").
- IOL_CALL is based on the CALL specification in [13] and [15], which for the first time was used by the I&M functions [3]. IOL-CALL is a non-exclusive extension to the standard CALL. It can be used by other integration technologies even though the name suggests exclusive usage for IO-Link.
- The IOL-CALL deploys
  - in case of PB-DP the DP-V1 services "Process Data ASE READ/ WRITE" supported by MS1 and MS2
  - in case of PN-IO the "Record Data ASE READ/ WRITE"
- In essence the corresponding procedures on PB-DP and PN-IO are nearly identical.

- Any read / write of IOL data objects is commenced by the IOL-Client with an IOL_CALL_REQ PDU at the assigned "cap" index (9.3.3)
- The IOL-Server executes the request and provides the required data or transfers the data to the corresponding IOL-D.
- The IOL-Client reads an IOL_CALL_RES PDU at the assigned "cap" index and receives the required data or acknowledges the completion of the request.

9.3.1 IOL-Header part for IOL-D

With the help of the specific IO-Link CALL (IOL_CALL) the IOL data objects can be directly "tunneled" to an IOL-D ("Index.subindex") as shown in Table 29. The IOL_Index range from 0...32767 is directly assigned to the index of the IOL-D. The IOL_Subindex range is directly assigned to the subindex of the IOL-D.

Table 29 — IOL-Header part for IOL-D

	Content	Size	Coding	Remarks
IOL-Header	Control	1 Octet	0…255	"Coding of Control" (9.3.3)
	IOL_Index	2 Octet	0…32767	IOL-D data: Index
	IOL_Subindex	1 Octet	0…255	IOL-D data: Subindex

9.3.2 IOL-Header part for port functions

With the help of the specific IO-Link CALL (IOL_CALL) additional port specific functions can be invoked on the IOL-M such as *Wakeup* or *Fallback* (11.9). Port functions can be addressed through the IOL_Index 65535. A particular port function is coded via the IOL_Subindex.

Table 30 — IOL-Header part for port functions

	Content	Size	Coding	Remarks
IOL-Header	Control	1 Octet	0…255	"Coding of Control" (9.3.3)
	IOL_Index	2 Octet	65535	Port Function Invocation
	IOL_Subindex	1 Octet	0…255	Port Function 0: Reserved 1: Fallback 2: Wakeup 3: Reconfiguration 4..63: Reserved 64..255: Manufacturer specific

The IOL_Index range from 32767 through 65534 is reserved, i.e. each and every IOL_CALL using one of these indices will be rejected with an error message "not supported" (Table 37).

9.3.3 IOL_CALL request PDU

The coding of an IOL_CALL_REQ (request) PDU is defined in Table 31. Presented in this table is the coding example of a "Write-REQ" PDU with a DP-V1-Header, Call-Header, IOL-Header, and Body.

Table 31 — Coding of an IOL_CALL request PDU

Section	Content	Size	Coding	Definitions
DP-V1-Header	Function_Num	1 Octet	5FH	Fix ("Write Record")
	Slot_Number	1 Octet	x	Slot: IOL-M
	Index	1 Octet	255 0…254	Regular index (CALL) Index redirection (cap)
	Length	1 Octet	0…241	Length of data including header
Call -Header	Extended_Function_Num	1 Octet	08h	Indicates "CALL“, fix
	Entity_Port	1 Octet	0 1…63 64…255	IOL-M Port number Reserved
	FI_Index	2 Octets	65098	IOL-Header is following
IOL-Header	Control	1 Octet	0…255	"Coding of Control" (9.3.3)
	IOL_Index	2 Octet	0…32767 65535	IOL-D data: Index Port function invocation
	IOL_Subindex	1 Octet	0…255	IOL-D data: Subindex or Port function
Body	IOL_Data_Object	232 Octets maximum		"Write" Data
NOTE The coding of the DP-V1-Header within this table is exemplary for PB-DP. The PN-IO-Header is characterized by API /Slot /Subslot /Index.				

The activities of the IOL_CALL are carried out with the help of the variable "Control".

Coding of Control (Octet)

7	6	5	4	3	2	1	0		Definitions
0	0	0	0	0	0	0	0	=	Cancel / Release IOL_CALL
0	0	0	0	0	0	0	1	=	IDLE Sequence
0	0	0	0	0	0	1	0	=	Write Data
0	0	0	0	0	0	1	1	=	Read Data
other								=	Reserved

9.3.4 IOL_CALL response PDU

The coding of an IOL_CALL_RES (response) PDU is defined in Table 32. Presented in this table is the coding example of a "Read-RES" PDU with a DP-V1-Header, Call-Header, IOL-Header, and Body.

Table 32 — Coding of an IOL-CALL response PDU

Section	Content	Size	Coding	Description
DP-V1-Header	Function_Num	1 Octet	5EH	Fix ("Read Record")
	Slot_Number	1 Octet	x	Slot: IOL-MM
	Index	1 Octet	255 0…254	Regulra index (CALL) Index redirection (cap)
	Length	1 Octet	0…241	Length of data including header
Call Header	Extended_Function_Num	1 Octet	08h	Indicates "CALL“, fix
	Entity_Port	1 Octet	0 1…63	IOL-M Port number

Section	Content	Size	Coding	Description
			64…255	Reserved
	FI_Index	2 Octets	65098	IOL-Header is following
IOL-Header	State	1 Octet	0..255	"Coding of State" (9.3.4)
	IOL_Index	2 Octet	0…32767 65535	IOL-D data: Index Port function invocation
	IOL_Subindex	1 Octet	0…255	IOL-D data: Subindex or Port function
Body	IOL_Data_Object	232 Octets maximum		"Read" Data
NOTE	The coding of the DP-V1-Header within this table is exemplary for PB-DP. The PN-IO-Header is characterized by API /Slot /Subslot /Index.			

The activities of the IOL_CALL are monitored with the help of the variable "State".

Coding of State (Octet)

7	6	5	4	3	2	1	0		Definitions
0	0	0	0	0	0	0	0	=	Done /Transfer terminated
0	0	0	0	0	0	0	1	=	IDLE sequence
0	0	0	0	0	0	1	0	=	Reserved
0	0	0	0	0	0	1	1	=	Reserved
1	0	0	0	0	0	0	0	=	IOL Error PDU (Table 36 and Table 37)
other								=	Reserved

9.3.5 IOL_CALL sequence

The following Figure 53 and Figure 54 are depicting the transfer sequences of the IOL_CALL protocol exemplary for PB-DP. Client access points ("cap") are already supposed at this point.

For easier reading the following term replacements are used in the figures and in the corresponding text:

- Read-REQ-PDU = IOL_CALL_REQ "read" PDU
- Write-REQ-PDU = IOL_CALL_REQ "write" PDU
- Read-RES-PDU = IOL_CALL_RES "read" PDU
- Write-RES-PDU = IOL_CALL_RES "write" PDU
- Read-NRS-PDU = IOL_CALL_RES "busy" PDU

In case an IOL-Server is unable to immediately provide response (flow control) after a Write-REQ-PDU it can answer with a Write-NRS-PDU ("Resource busy"). The IOL-Client will then repeat the Write-REQ-PDU. A timeout is monitored independently (9.3.6.2.2).

In case an IOL-Server is unable to immediately provide response data after a Read-REQ-PDU it can

- Answer with a Read-NRS-PDU ("Resource busy"). The IOL-Client will then repeat the Read-REQ-PDU. This method is shown in example 1.
- Hold up the Read-REQ-PDU until a Read-RES-PDU is available. This method is shown in example 2.

In the following the two examples are presented and explained.

The two methods can be used for both reading and writing of IOL data objects. Example 1 is demonstrating the case "Resource busy" whereas example 2 is demonstrating the case "Delayed response".

Example 1: Reading an IOL data object (PB-DP)

A read IOL data object task is transfered with the help of a Write-REQ-PDU. The header contains the address information "IOL port /index /subindex" of the IOL data object.

The IOL-Client keeps polling with Read-REQ-PDUs until a Read-RES-PDU with the requested IOL data object is returned to the IOL-Client. The Read-RES-PDU can contain error information when necessary.

Figure 53 is demonstrating exemplary the activities of an IOL-MM within a physically modular remote I/O system and its local bus system.

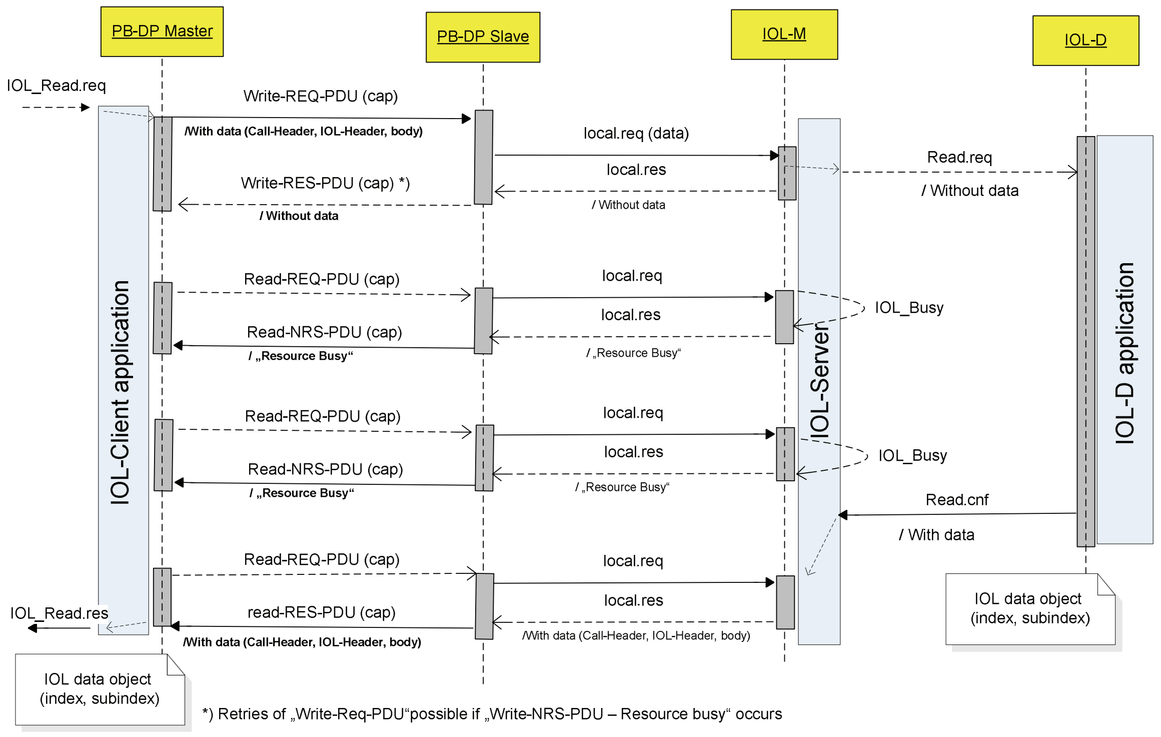


Figure 53 — Read IOL data object

Example 2: Writing an IOL data object (PB-DP)

A write IOL data object task is transfered with the help of a Write-REQ-PDU. The header contains the address information "IOL port /index /subindex" of the IOL data object. The Body contains the IOL data object.

The IOL-Client casts a Read-REQ-PDU and is waiting until a Read-RES-PDU indicates the completion of the write sequence to the IOL-D. Die Read-RES-PDU is being delayed until the response data are available.

Figure 54 is demonstrating exemplary the activities of an IOL-CM.

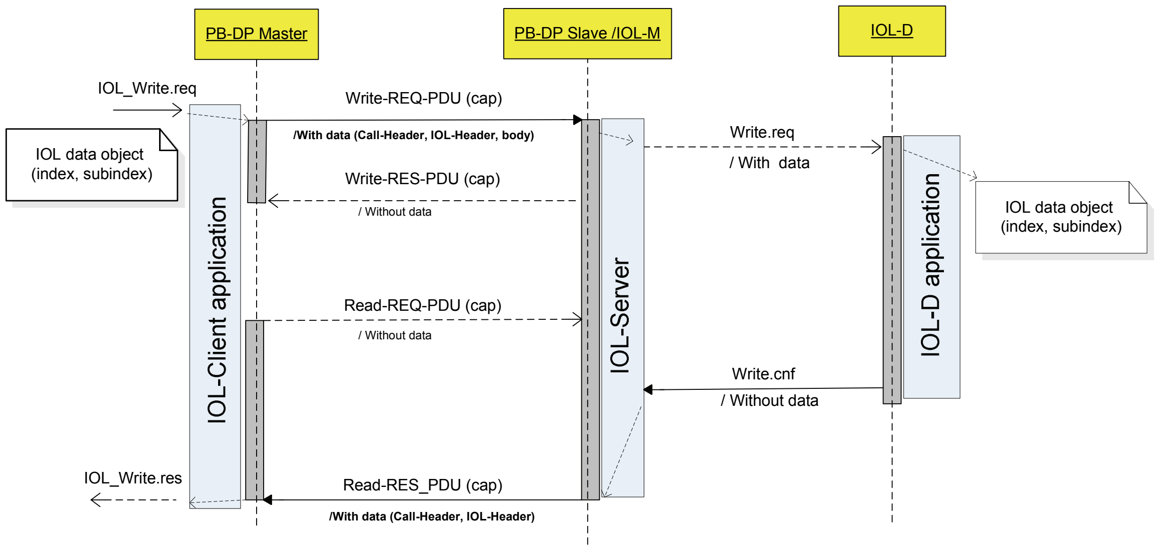


Figure 54 — Write IOL data object

9.3.6 Errors and coding

While executing the IOL_CALL sequences the following error situations can occur.

- *ERROR indication:* errors caused by the IOL_CALL communication protocol or which are related to the PB-DP or PN-IO mechanisms are reported using the PB-DP or PN-IO mechanisms shown in Table 33. Errors related to the IO-Link extensions are reported via a corresponding IOL error PDU as shown in Table 36 and Table 37.
- *Sequence ERROR:* For example when several IOL-Clients are trying to use IOL_CALL through one client access point ("cap"). For details see 9.3.6.2.1.
- *Client Timeout:* For example in case an IOL-D did not deliver a response within the defined watchdog time. For details see 9.3.6.2.2.

9.3.6.1 PB-DP and PN-IO error coding

The IO-Link relevant errors and the associated coding are presented in Table 33 and in the subsequent sections.

Table 33 — PB-DP and PN-IO error coding

	Content	Size	Coding	Remarks
e.g. Record Data.res(-)	Function_Num	1 Octet	0xDF 0xDE	Fix (Error Write) Fix (Error Read)
	Error_Decode	1 Octet	0x80	fix
	Error_Code_1	1 Octet	0…255	Error_Class, Error_Code
	Error_Code_2	1 Octet	0 1…255	Don´t care Additional code

Error_Code_1 (Error_Decode 0x80):

Error_Class				Error_Code			
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

The coding of Error_Class is defined in Table 34, the Error_Code in Table 35.

Table 34 — Error_Class "Access"

Error cause	Error_Class	Error_Code	Action
Access of too many IOL-Clients (Sequence Error)	0xB =Access	0x5 =State conflict	Abort IOL_CALL
See [6] for further possible error codes.			

Table 35 — Error_Class "Resource"

Error cause	Error_Class	Error_Code	Action
IOL data objects not yet available (request pending)	0xC =Resource	0x2 =Resource busy	Repeat REQ-PDU
See [6] for further possible error codes.			

9.3.6.2 IO-Link error coding

Errors that are generated directly by the IO-Link application layer services IOL_CALL read or write are directly returned via an IOL Error PDU to the client application (Table 36 and Table 37). Generally, an IO-Link error type consists of four octets, the IOL-M Error_Code, the IOL Error_Code and the IOL Add_Error_Code. Table 36 defines the error coding in case of IOL data object transfers.

Table 36 — IOL_CALL_RES PDU in case of errors

	Content	Size	Coding	Definitions
IOL-Header	State	1 Octet	0x80	= IOL Error PDU
	IOL_Index	2 Octets	0…65535	IOL Index
	IOL_Subindex	1 Octet	0…255	IOL Subindex
Body (IOL Error PDU)	IOL-M Error_Code	2 Octets	0…65535	Coding see Table 37
	IOL Error_Code	1 Octet	0…255	0x00 or Error Code [8]
	IOL Add_Error_Code	1 Octet	0…255	0x00 or Additional Error Code [8]

Inconsistent parameters such as an IOL_Index > 32767 and errors related to an invocation of port functions are coded within the IOL error ranges defined in Table 37.

Table 37 — IOL-M Error_Codes

IOL-M Error_Code	Coding	Definitions
No error	0x0000	No IOL-M or IOL-Server errors
-	0x0001…0x06FF	Reserved for future use
IOL_CALL conflict	0x7000	Unexpected Write req instead of Read req
Wrong IOL_CALL	0x7001	Decode error
Port blocked	0x7002	Port occupied by another task
-	0x7003…0x7FFF	Reserved for future use
Timeout	0x8000	Timeout when IOL-Ds or IOL-M ports are busy
Wrong index	0x8001	IOL_Index >32767 or <65535
Wrong port address	0x8002	Port address beyond defined maximum
Wrong port function	0x8003	Port function not supported
-	0x8004…0xFFFF	Reserved for future use

9.3.6.2.1 Sequence error

Sequence errors can occur because of the nature of an IOL-CALL that consists of a sequence of write and read transactions. Basically sequence errors are caused by several IOL-Clients trying to communicate with an IOL-Server through one client access point ("cap"). Several reaction types exist:

- Two or more IOL-Clients are trying to communicate with an IOL-Server through one client access point ("cap") at one point in time. The IOL-Server will start processing one of them (write.req) and usually will wait on a read.req but receives the write.req of the second IOL-Client. No matter how this (very unlikely) conflict is resolved, the results may be unsatisfactory. The PB-DP-Slave or PN-IO-Device or IOL-M respectively accepts only to process the newest ("last") IOL_CALL request. No error message occurs.
- Read-REQ-PDU without an IOL-Client request (Figure 53): The PB-DP Slave or PN-IO-Device responds with a Read-NRS-PDU (Error_Class =0xB, Error_Code =0x05).
- Two or more correct Read-REQ-PDUs after an IOL-Client request (Figure 53): The PB-DP Slave or PN-IO-Device responds with a Read-NRS-PDU (Error_Class =0xB, Error_Code =0x05).

9.3.6.2.2 Client timeout

Each IOL-Client maintains a watchdog timer to monitor the IOL_CALL sequence. The time duration is manufacturer specific but always > 1 second. It is not necessary for an IOL-Server to maintain a watchdog timer. However, the underlying IOL communication needs time monitoring.

9.3.7 IOL-CALL function block definitions

In 5.2.10 and Figure 23 the outline of a dedicated IOL_CALL function block for user program controlled parameterization is presented which is detailed in the following. Table 38 is defining the input and output variables.

Table 38 — IOL_CALL function block variables

Variable	Data type	Definition
REQ	BOOL	Request function
ID	DWORD	Identification of a Field Device or a slot / subslot of it
INDEX_CAP	INT	Identifier of an IOL-D or IOL-M port data object
RD_WR	BOOL	Read or write access
ENTITY_PORT	INT	IOL-M port address
FI_INDEX	INT	Indicating an IOL-Header via fix 65098
IOL_INDEX	INT	IO-Link index 0…32767 (IOL-D) or 65535 (port functions)
IOL_SUBINDEX	INT	IO-Link subindex 0…255
LEN	INT	Actual data length of an IOL data record (read or write)
RECORD_IOL_DATA	ANY	Array of IOL data object bytes
DONE_VALID	BOOL	Flag indicating the successful completion of the function (and in case of a read function the validity of received data)
BUSY	BOOL	Flag indicating the function is still performing its task. The function block is not ready to perform a new task.
ERROR	BOOL	Flag indicating the abortion of the function with an error
STATUS	DWORD	Completion or bus error code. Busy =0xFFFFFFFF (in [6] coded as "-1").
IOL_STATUS	DWORD	Completion or IOL-M (Table 37) and IOL-D error code [8]

The IOL data object and error information is passed by input-output variables to the IOL_CALL function block. Typically these data can be described as an array of bytes. The length of the byte arrays may vary from instance to instance of one IOL_CALL function block. It shall also be possible to use a structured data type if the used data is structured. Further information is available in [6].

The input variable LEN specifies the count of bytes which shall be read as a maximum. The size of the variable RECORD_IOL_DATA shall be at least LEN bytes. The output variable DONE_VALID indicates the availability of a received data record in the variable RECORD_IOL_DATA. The output variable LEN contains the length of the data record in byte. In case of an error, the ERROR output variable is set true (=1) and the IOL_STATUS output variable contains the error code (9.3.6).

The function block is invoked when the input REQ is true (=1). The output variables of the IOL_CALL function block are showing the state of the last requested service (task). They will remain unchanged until a new service is requested from the same function block instance.

9.3.8 State diagrams

The IOL-CALL protocol is realized by a finite state machine on both the IOL-Client and the IOL-Server side. One finite state machine is necesssary per client access point ("cap"). The finite state machine(s) of the IOL-Server are instantiated on the IOL-M.

9.3.8.1 State diagram of the IOL-Client

The finite state machine for an IOL-Client as shown exemplary as IOL_CALL function block in Figure 55 is coined through four states and their responsibilities. It is generic enough to be used in other client applications such as an IOPD tool. A new IOL-CALL task can be started from the state *1 IDLE* via variable REQ = 1. The tasks will be aborted via IOL-Client timeout within the states *2 SEND_REQUEST* or *3 WAIT_ON_RES* (for example if an IOL-Server fails).

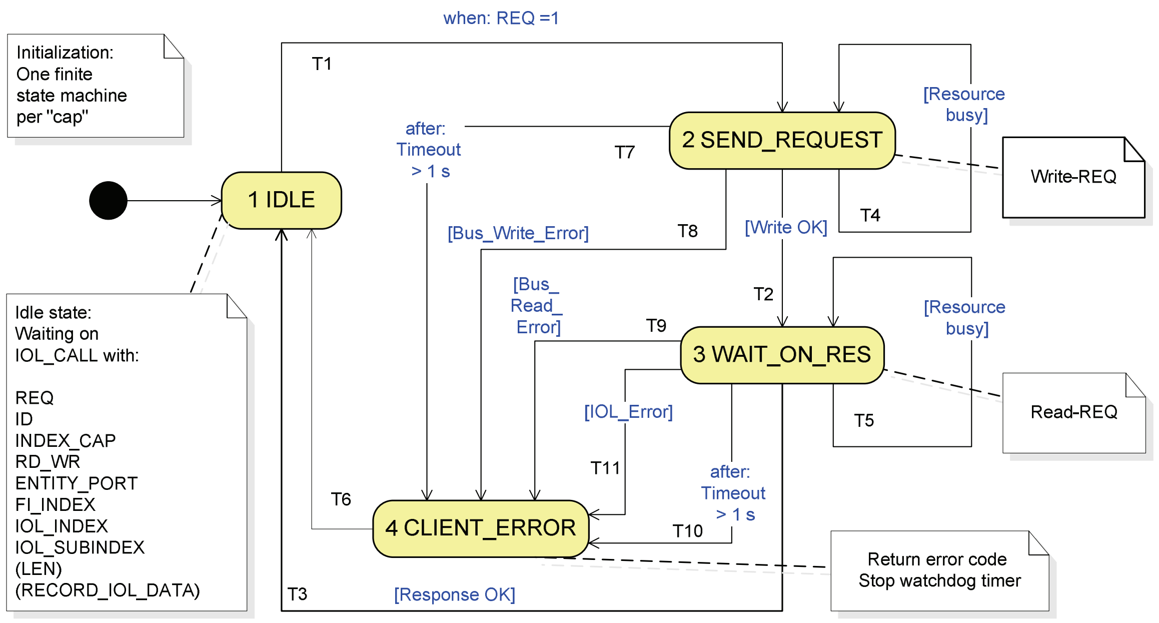


Figure 55 — State diagram of an IOL-Client

The terms used in Figure 55 are specified as follows:

REQ =1	User program launches the IOL_CALL function
[Write OK]	PB-DP or PN-IO write service has been carried out correctly (Figure 53)

[Bus_Write_Error]	PB-DP or PN-IO write service failed (for error codes see Table 33)
[Bus_Read_Error]	PB-DP or PN-IO read service failed (for error codes see Table 33)
[IOL_Error]	IOL services failed (for error codes see Table 36 and Table 37)
[Response OK]	PB-DP or PN-IO services have been carried out correctly; no RES PDU with error codes recognized
[Resource busy]	IOL services not completed (for error code see Table 35)
Timeout >1s	IOL_CALL Timeout expired after 1s

States and transitions are defined in detail in Table 39.

Table 39 — State transition table of an IOL-Client

STATE NAME	STATE DESCRIPTION
1 IDLE	Initialization already completed after startup. Idle state, no activities. Reset and de-activate the IOL-Client watchdog timer. Waiting on IOL_CALL request (REQ =1).
2 SEND_REQUEST	IOL_CALL request was sent via Write-REQ PDU to the IOL-M through a "cap". At the same time the watchdog timer is started. IOL-M executes request (IOL-D or port function). Waiting on Write-RES PDU (OK) or Write-NRS PDU (error).
3 WAIT_ON_RES	Read-REQ PDU to read the IOL-M response through a "cap". Read-REQ PDUs to be repeated, when "Resource busy" (0xC2) is returned (Table 35). A consistency check is performed between both the headers of the IOL_CALL_REQ PDU and the IOL_CALL_RES PDU. Waiting on Read-RES PDU (OK) or Read-NRS PDU (error).
4 CLIENT_ERROR	Indicate a "faulty“ abort of the protocol and stop the IOL-Client watchdog timer. Indicate error. Set output variables.

TRAN-SITION	SOURCE STATE	TARGET STATE	ACTION
T1	1	2	DONE_VALID =0, BUSY =1, ERROR =0, STATUS =-1, IOL_STATUS =-1, LEN =0. Start timer. Send Write-REQ PDU.
T2	2	3	Received Write-RES PDU: DONE_VALID =0, BUSY =1, ERROR =0, STATUS =-1, IOL_STATUS =-1, LEN =0.
T3	3	1	Received Read-RES PDU: Stop timer; DONE_VALID =1, BUSY =0, ERROR =0, STATUS =0, IOL_STATUS =-1, LEN =record length. Invocation
T4	4	4	Received Write-NRS PDU with "Resource busy" (=0xC2 from Table 35): Send Write-REQ PDU again. Continue timer; DONE_VALID =0, BUSY =1, ERROR =0, STATUS =-1, IOL_STATUS =-1, LEN =0.
T5	3	3	Received Read-NRS PDU with "Resource busy" (=0xC2 from Table 35): Send Read-REQ PDU again. Continue timer; DONE_VALID =0, BUSY =1, ERROR =0, STATUS =-1, IOL_STATUS =-1, LEN =0.
T6	4	1	DONE_VALID =0, BUSY =0, ERROR =1, STATUS =error code, IOL_STATUS =error code, LEN =0. Invocation
T7	2	4	Did not receive Write-RES PDU or Write-NRS PDU in time: Reset timer; DONE_VALID =0, BUSY =0, ERROR =1, STATUS =0, IOL_STATUS =error code, LEN =0.
T8	2	4	Received Write-NRS PDU with error: Stop timer; DONE_VALID =0, BUSY =0, ERROR =1, STATUS =error code, IOL_STATUS =0, LEN =0.
T9	3	4	Received Read-NRS PDU with error: Stop timer; DONE_VALID =0, BUSY =0, ERROR =1, STATUS =error code, IOL_STATUS =0, LEN =0.
T10	3	4	Did not receive Read-RES PDU or Read-NRS PDU in time: Stop timer; DONE_VALID =0, BUSY =0, ERROR =1, STATUS =0, IOL_STATUS =error code, LEN =0.
T11	3	4	Received Read-RES PDU with IOL error:

TRAN-SITION	SOURCE STATE	TARGET STATE	ACTION
			Stop timer; DONE_VALID =0, BUSY =0, ERROR =1, STATUS =0, IOL_STATUS =error code, LEN =0.

INTERNAL ITEMS	TYPE	DEFINITION
Timer		IOL-Client activities are monitored by a watchdog timer (9.3.6.2.2)
Error_Code		In case of errors or failures an Error_Code is returned. Possible errors: Table 33, Table 34, Table 35, Table 36, and Table 37.
Resource busy		If IOL-M cannot execute: Table 35

9.3.8.2 State diagram of an IOL-Server

The finite state machine of the IOL-Server is presented in Figure 56. The IOL-Server maintains a watchdog timer for the IOL services. The finite state machine can always be restarted by an incoming new IOL_CALL from the same or another IOL-Client. All current IOL processes of the associated port will be aborted.

States and transitions are defined in detail in Table 40.

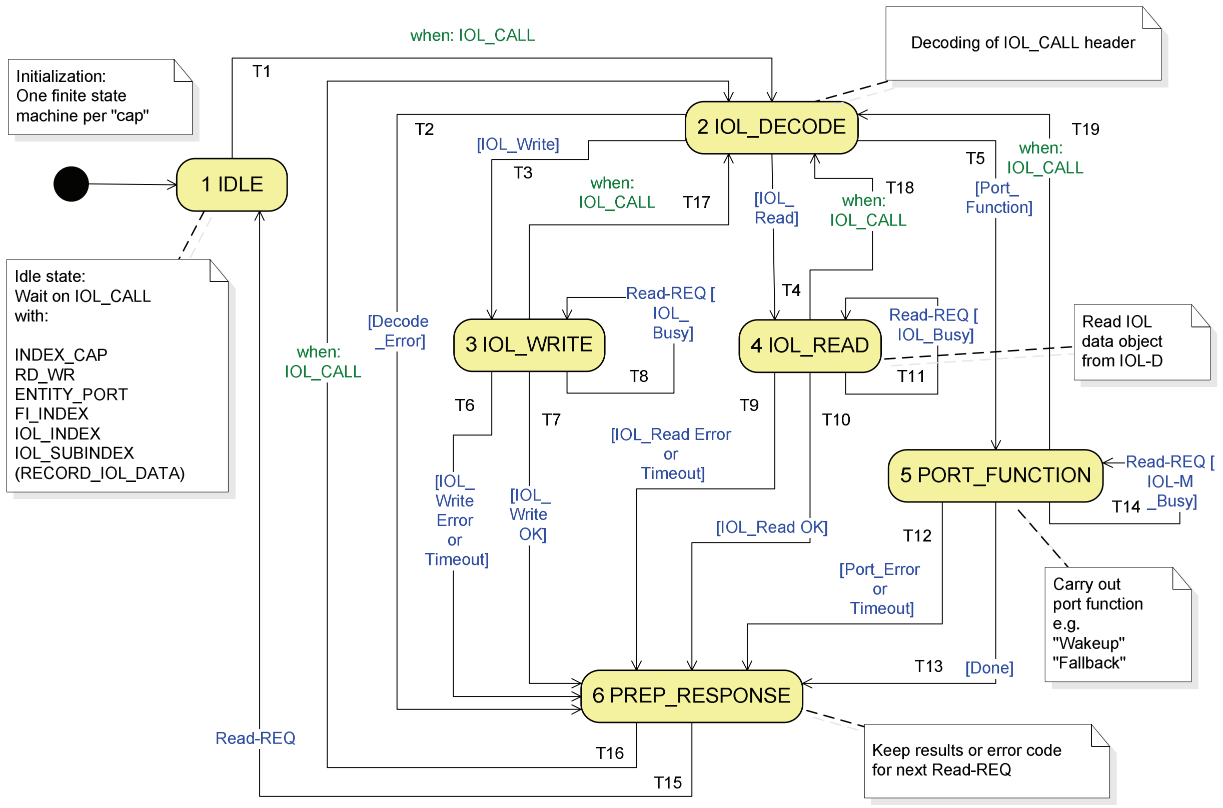


Figure 56 — State diagram of an IOL-Server

The terms used in Figure 56 are defined within the "Internal Items" section of Table 40.

States and transitions are defined in detail in Table 40.

Table 40 — State transition table of an IOL-Server

STATE NAME	STATE DESCRIPTION
1 IDLE	Idle state, no activities
2 IOL_DECODE	IOL_CALL request received either from the *1 IDLE* state or any other. Decoding and checking for correctness of the IOL-CALL header.
3 IOL_WRITE	Write service to the IOL-D to be performed (address: IOL_Index, IOL_Subindex) as activity according to the flowchart in Figure 57. Unexpected new IOL_CALL will cause the transition to the *2 IOL_DECODE* state. All other processes are aborted.
4 IOL_READ	Read service to the IOL-D to be performed (address: IOL_Index, IOL_Subindex) as activity similar to the flowchart in Figure 57. Unexpected new IOL_CALL will cause the transition to the *2 IOL_DECODE* state. All other processes are aborted.
5 PORT_FUNCTION	A specified port function such as "Wakeup" or "Fallback" to be performed (address: IOL_Index, IOL_Subindex) as activity similar to the flowchart in Figure 57. It is assumed that time monitoring is provided by the generic IOL_M layer. Unexpected new IOL_CALL will cause the transition to the *2 IOL_DECODE* state. All other processes are aborted.
6 PREP_RESPONSE	Prepare either Read-NRS PDU in case of error (incorporate error codes) or Read-RES PDU with or without data. Waiting on next Read-REQ PDU.

TRAN-SITION	SOURCE STATE	TARGET STATE	ACTION
T1	1	2	-
T2	2	6	Return Write-RES PDU and prepare error code for Read-RES PDU "Wrong IOL_CALL" (Table 37)
T3	2	3	Return Write-RES PDU
T4	2	4	Return Write-RES PDU
T5	2	5	Return Write-RES PDU
T6	3	6	Prepare error codes for Read-RES PDU
T7	3	6	Prepare empty Read-RES PDU
T8	3	3	Respond with Read-NRS PDU "Resource busy"
T9	4	6	Prepare error codes for Read-NRS PDU
T10	4	6	Prepare IOL data for Read-RES PDU
T11	4	4	Respond with Read-NRS PDU "Resource busy"
T12	5	6	Prepare error codes for Read-NRS PDU
T13	5	6	Prepare empty Read-RES PDU
T14	5	5	Respond with Read-NRS PDU "Resource busy"
T15	6	1	Send either Read-RES or Read-NRS PDU
T16	6	2	Abort all IOL processes
T17	3	2	Abort all IOL processes
T18	4	2	Abort all IOL processes
T19	5	2	Abort all IOL processes

INTERNAL ITEMS	TYPE	DEFINITION
Decode_Error		Error condition containing "0x7001"
Done		OK condition for the execution of a port function
IOL_Busy		IOL-D did not accomplish a task from a particular port yet
IOL_CALL		Write or read request for an IOL-D or an IOL-M port
IOL-M_Busy		IOL-M did not accomplish a task yet

INTERNAL ITEMS	TYPE	DEFINITION
IOL_Read		Task to read IOL data from an IOL-D
IOL_Read Error		Error occurred while reading IOL data from an IOL-D
IOL_Read OK		Read task accomplished successfully
IOL_Write		Task to write IOL data to an IOL-D
IOL_Write Error		Error occurred while writing IOL data to an IOL-D
IOL_Write OK		Write task accomplished successfully
Port_Error		Error occurred while performing port functions
Port_Function		Task to execute a particular port function
Response		Send either Read-RES or Read-NRS PDU
Timeout		Time period for monitoring the execution of IOL_WRITE, IOL_READ, or PORT_FUNCTION. Monitoring is provided by the generic IOL_M layer defined in [8]. A special error code is returned by this layer indicating "Timeout". The individual value for the timer shall be provided by the IODD of the particular IOL-D connected to the port.

The principle of the activity in the IOL_WRITE state is shown in Figure 57. T3, T6, T7, T8, and T17 refer to the transitions in Figure 56.

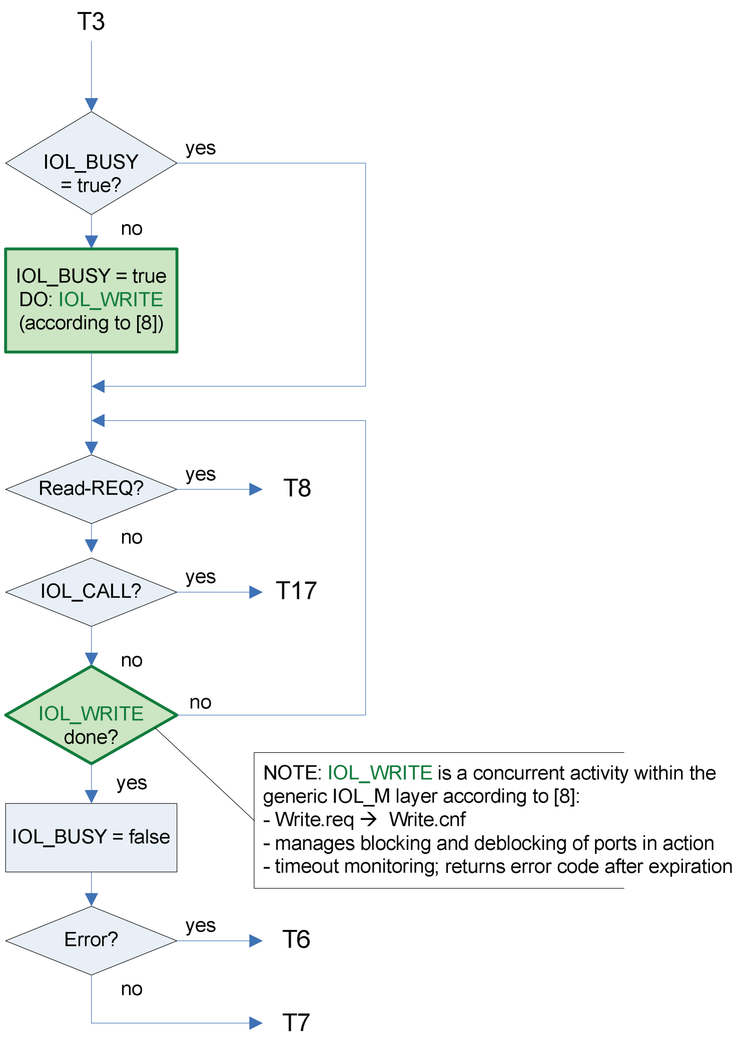

T3
IOL_BUSY = true?
yes
no
IOL_BUSY = true
DO: IOL_WRITE
(according to [8])
Read-REQ?
yes
T8
no
IOL_CALL?
yes
T17
no
IOL_WRITE done?
no
yes
IOL_BUSY = false
NOTE: IOL_WRITE is a concurrent activity within the generic IOL_M layer according to [8]:
- Write.req → Write.cnf
- manages blocking and deblocking of ports in action
- timeout monitoring; returns error code after expiration
Error?
yes
T6
no
T7

Figure 57 — Activity within the IOL_WRITE state

10 Parameter server

The method described in the following sections provides mechanisms for IOL-D and IOL-M parameters to be stored on and retrieved from a central place (iPar-Server).

NOTE IOL-M port parameter and up to 10 bytes anonymous IOL-D parameter defined via GSD entries are explicitely excluded here. With PB-DP and PN-IO a central start-up parameterization model for these GSD based parameters already exists. These parameters are retained in the PB-DP Master Class 1 or PN-IO Controller.

10.1 Introduction /use cases

Usually the IOL-D needs adjustments prior to deployment through parameterization. This activity is carried out with the help of engineering tools and device description files such as PCT and IODD or individual designated IOPD tools. The tools create IOL-D parameter blocks that are to be downloaded into an IOL-D via the IO-Link interfaces. From an IOL-D point of view the parameterization narrows down to the download of one or more parameter blocks (objects). The parameterization tool shall notify the IOL-D about the end of a parameterization session, usually after function verification.

IOL-Ms are also candidates to receive parameter blocks such as port configuration from the PCT.

The following use cases are covered by the parameter server model described in subsequent sections:

- *IOL-D replacement (hot swap):* In case of a defective device it shall be possible to replace an IOL-D even "at runtime" without an engineering tool.
- *Parameter change online (flexible production):* Generally it shall be possible to adapt or change parameters at runtime. There are two ways to realize this use case.
  - Changed parameters are only valid temporarily, i.e. for a particular phase of an automation process.
  - Changed parameters are retained after power-off, i.e. after power-on the IOL-D is ready without reparameterization.
- *Offline configuration and parameterization (project planning /development):* Configuration and parameterization is carrried out offline with the help of an engineering system. The generated parameter blocks (objects) are stored within the project database of the engineering system. From there it shall be possible to download the generated parameter blocks (objects) into the corresponding IOL-M and IOL-Ds once they are connected and online.
- *Online configuration and parameterization (commissioning):* Configuration and parameterization is carried out online with the help of the engineering system that is communicating directly with the IOL-M and its IOL-Ds.

10.2 Parameter server models

Generally there are different applications in today's automation processes that have access to the parameters of an IOL-M and its IOL-Ds. The access methods are specified in 9.

Figure 58 is showing the typical applications such as PLC, fieldbus engineering systems, process monitoring systems (HMI), IO-Link tools (PCT, IOPD). It shall be taken into account that IOL-Ds can also receive or establish parameters via local teach-in, PC adapters (e.g. USB), or other handheld devices.

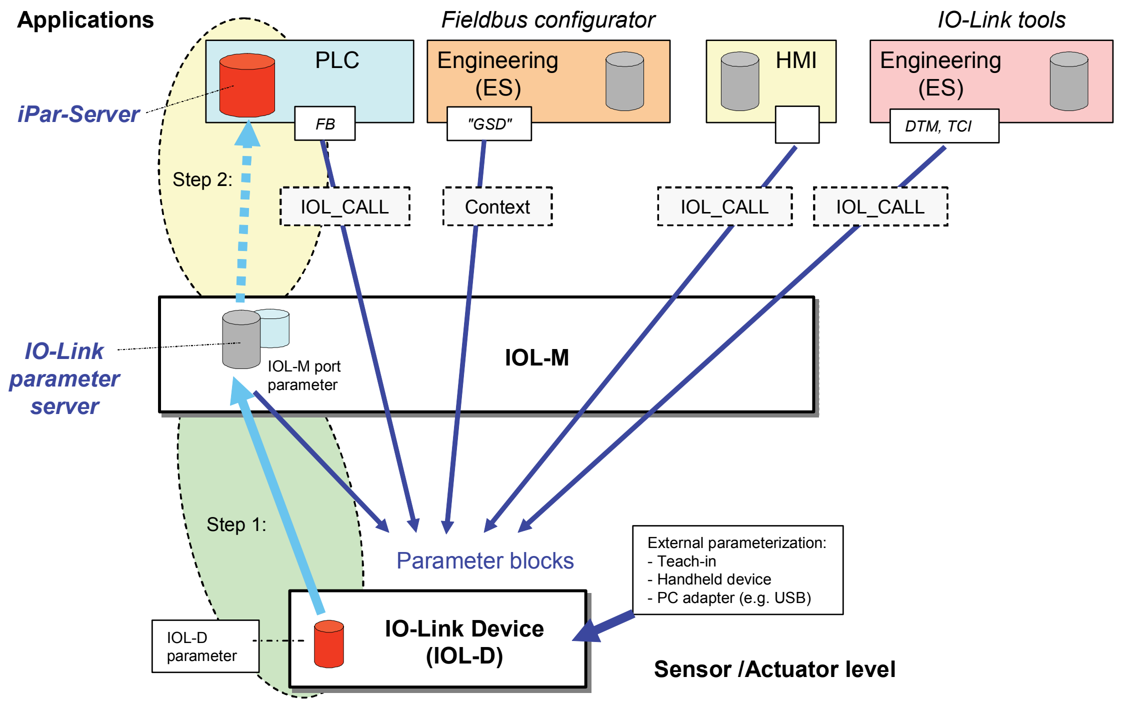


Figure 58 — Parameter server concept

The parameter server model is built-up as a two-tier solution (levels). Level 1 is an IO-Link specific functionality supporting the permanent storage of IOL-D parameters within an IOL_M.

- *Level 1:* At this level the IOL-D parameters can be saved in an IO-Link parameter server within the IOL-M. The IOL-M is actively involved in checking the integrity /version of IOL-D parameters and restoring parameters at start-up if needed. This method is specified in the "System Integration" part [9] of the future version V1.1 of the IO-Link Specification [8] according to Figure 1.
- *Level 2:* At this level the complete set ("bulk data") of IOL-M and IOL-D parameters from inside an IOL-M can be saved in a standard "iPar-Server" usually located in a system that provides the GSD-based fieldbus start-up parameterization ("Context"). Upon request from the IOL-M the "iPar-Server" delivers the complete set back to the IOL-M for restoration, e.g. in case of IOL-M replacement. PLC or other control systems hosting PB-DP Master Class 1 or PN-IO Controller are obliged to provide the "iPar-Server" services.

The IOL-D is the solely "owner" of the *actual valid* parameter set after correct commissioning or external parameterization. All the applications and instances such as the IOL-M may or may not have an identical parameter set. Thus, it is the responsibility of the IOL-D to save the Parameter set when certain criteria are fulfilled. This general rule is necessary due to the many ways the IOL-D can receive or establish new parameters, e.g. teach-in. One criterion should be the completion of a parameterization session with engineering tools. A corresponding "valid identifier" triggers the IOL-D to either save or only keep the parameter set temporarily.

The IOL-D notifies its IOL-M via an event to upload (save) the parameter set. The "IO-Link parameter server" within the IOL-M is responsible to execute the request. An IOL-M notifies its iPar-Server to upload (save) the complete parameter sets of all the IOL-Ds and its own parameter set in form of "bulk data" (stream of bytes). The activities are marked as bright blue arrows in Figure 58.

The IOL-M is actively involved in checking the integrity /version of IOL-D parameters and restoring parameters within the IOL-D at start-up if needed. An IOL-M recognizing that its own stored parameter set ("bulk data") is invalid, for example through a CRC signature check, notifies its iPar-Server to download (restore) the saved "bulk data".

10.2.1 IO-Link parameter server

This mechanism is specified in the "System Integration" part [9] of the future version V1.1 of the IO-Link Specification [8] according to Figure 1.

The basic concept can be described as follows:

- A change of the parameter set or a "save" request is indicated by the IOL-D through an *event*.
- The IOL-M is reading and retaining the complete parameter set from the IOL-D ("save").
- Upon each IOL-D start-up (power-on or reconfiguration) the IOL-M is checking the validity of the parameter set through CRC signature or comparison with its retained corresponding parameter set. In case of mismatch the IOL-M writes ("restore") the retained parameter set into the IOL-D.
- Specialty: IOL-D without "Service PDU supported“ [8] are always supplied with the retained corresponding parameter set.

10.2.2 iPar-Server

The mechanism iPar-Server provides an interface to

- save (store) a data object ("bulk data") per slot or subslot at a central place such as a PLC
- restore this data object ("bulk data" = parameters sets) into a slot or subslot, here IOL-M

The initiative to "save" or "restore" data objects is always coming from an IOL-M. The IOL-M initiates a "save" request through a dedicated notification or a "restore" request through another dedicated notification. Basically the structure and coding of the "bulk data" is purely manufacturer specific and needs no standardization. However, it is highly recommended to provide a 32 bit CRC signature within the first four bytes of the "bulk data". A proven-in-use CRC polynomial for this purpose is 1*F4ACFB13*h.

10.2.2.1 Notification

See 7.4.1.4.2 for details.

10.2.2.2 Read/ Write services

The iPar-Server application is looking for two kinds of requests "save" and "restore" (7.4.1.4.2). In order to execute these requests it uses the standard "read record" and "write record" acyclic services as shown in Table 41 and Table 42. For small amounts of "bulk data" an ordinary unsegmented version per single "read record" and "write record" is sufficient.

For amounts of "bulk data" exceeding the limits of one record an extended version of the "read record" and "write record" acyclic services can be used, specified in [13] as the so-called "Pull" and "Push" services. The corresponding PDU codings are shown in Table 43 and Table 44.

Table 41 is presenting the coding of a "read record" PDU (Save).

Table 41 — Structure of the Read_RES_PDU ("read record")

Structure of the Read_RES_PDU	Size	Coding	Definitions	
Function_Num	1 octet	0x5E	Indicates "read", fix	Header
Slot_Number	1 octet	0 … 255	Location of module	
Index	1 octet	0 … 254	"Transfer_Index"	
Length of net data	1 octet	0 … 240	Length of iPar segment	
iParameter (segment)	n octets	-	n = 240 maximum per record	Data
NOTE Header represents a PB-DP example				

Table 42 is presenting the coding of a "write record" PDU (Restore).

Table 42 — Structure of the Write_REQ_PDU ("write record")

Structure of the Write_REQ_PDU	Size	Coding	Definitions	
Function_Num	1 octet	0x5F	Indicates "write", fix	Header
Slot_Number	1 octet	0 … 255	Location of module	
Index	1 octet	0 … 254	"Transfer_Index"	
Length of net data	1 octet	0 … 240	Length of iParameter segment	
iParameter	n octets	-	n = 240 maximum	Data
NOTE Header represents a PB-DP example				

A description of the segmented transmission of "bulk data" exceeding the size of one record is contained in [18]. Table 43 is presenting the coding of a "pull record" PDU (Save).

Table 43 — Structure of the Pull_RES_PDU

Structure of the Pull_RES_PDU	Size	Coding	Definitions	
Function_Num	1 octet	0x5E	Indicates "Read", fix	Header
Slot_Number	1 octet	0 … 255	Location of module	
Index	1 octet	0 … 254 (255)	"Transfer_Index"	
Length of net data	1 octet	0 … 240	Length of iPar segment + Load Region header	
Extended_Function_Num	1 octet	0x02	Indicates "Pull"	Load Region
Options	1 octet	Unsigned8	Flow control, see 6.2.17.2 in [13]	
Sequence_Number	4 octets	Unsigned32	…of current iPar segment	
Parameter (segment)	n octets	Octet String	n = 234 maximum per record	Data
NOTE 1 A "Transfer_Index" of 255 complies in this case with [13]. However, access conflicts with other services such as a CALL of I&M functions shall be considered in the design and implementation phase. All other indices can be used for the "Pull" and "Push" services. NOTE 2 Header represents a PB-DP example				

Table 44 is presenting the coding of a "push record" PDU (Restore).

Table 44 — Structure of the Push_REQ_PDU

Structure of the Push_REQ_PDU	Size	Coding	Notes	
Function_Num	1 octet	0x5F	Indicates "Write", fix	Header
Slot_Number	1 octet	0 … 255	Location of module	
Index	1 octet	0 … 254 (255)	"Transfer_Index"	
Length of net data	1 octet	0 … 240	Length of iPar segment + Load Region header	
Extended_Function_Num	1 octet	0x01	Indicates "Push"	Load Region
Options	1 octet	Unsigned8	Flow control, see 6.2.17.2 in [13]	
Sequence_Number	4 octets	Unsigned32	…of current iPar segment	
iParameter (segment)	n octets	Octet String	n = 234 maximum per record	Data
NOTE 1 A "Transfer_Index" of 255 complies in this case with [13]. However, access conflicts with other services such as a CALL of I&M functions shall be considered in the design and implementation phase. All other indices can be used for the "Pull" and "Push" services.				
NOTE 2 Header represents a PB-DP example				

10.2.3 iPar_IOL mapping within the IOL-M

From an iPar-Server's point of view exactly one parameter object ("bulk data") can be saved or restored. The criteria for the initiation of the corresponding activities are defined in the following sections.

NOTE The structure and coding of parameter objects ("bulk data") is manufacturer specific and need not be standardized in this specification. A useful exemplary mapping is presented herein for better comprehension.

10.2.3.1 iPar_IOL mapping onto single-slot /subslot

All the parameter objects of the IOL-Ds and the IOL-M of an IO-Link system are incorporated in an "iPar_IOL" data object. Depending on its size this data object ("bulk data") is transferred via

- Read record /write record ("bulk data" size up to 244 bytes)
- Pull data record /push data record ("bulk data" size >244 bytes).

The internal structure of the "iPar_IOL" data object is manufacturer specific and not standardized herein.

Criteria for the initiation of "save" and "restore" activities:

- Detected and acknowledged change of the parameter set of IOL-Ds or the IOL-M leads automatically to "save" activities of the "bulk data". The IOL-M is responsible to create the notification for the iPar-Server.
- Upon power-on of the IOL-M a partial download of the "bulk data" (the first four bytes CRC signature only) is carried out in order to check the actuality of local "bulk data" against the remote "bulk data" in the iPar-Server (Option).
- Upon power-on of the IOL-M a complete download of the "bulk data" is carried out and parameter sets are restored locally in the IOL-M.

10.2.3.2 iPar_IOL mapping onto multi-slot /subslot

For the base slot x /subslot x=1 (Figure 44) all the parameter objects of the IOL-Ds and the IOL-M of an IO-Link system are incorporated in an "iPar_IOL" data object, i.e. only the base slot x /subslot x=1 are maintaining an "iPar_IOL" data object. Depending on its size this data object ("bulk data") is transferred via

- Read record /write record ("bulk data" size up to 244 bytes)
- Pull data record /push data record ("bulk data" size >244 bytes).

The internal structure of the "iPar_IOL" data object is manufacturer specific and not standardized herein.

Criteria for the initiation of "save" and "restore" activities are the same as with iPar_IOL for single slot /subslot.

10.3 Parameter quantities

In order to facilitate the layout of parameter servers, this version of the specification is defining minimum quantities of IOL-D parameters an IOL-M shall be able to store and the minimum quantity of parameters an iPar-Server shall be able to store.

- Per port these are at least 512 bytes in total (including address information) and
- Per IOL-M (IOL-MM or IOL-CM) these are at least 32 kB in total for all the ports and the IOL-M itself.

11 Dynamic behavior

11.1 Introduction

This clause deals with the start-up procedures of an IOL-M. It is the task of the IOL-M to coordinate the fieldbus start-up of PB-DP or PN-IO with the start-up of the ports and IOL-Ds.

The user is faced with several dynamic behaviors due to several different ways of parameterization via GSD, PCT, or IOPD.

It is highly recommended to use remanent storage within the IOL-M for port configuration data created by the PCT and for the IOL-D parameter objects hold by the IO-Link parameter server. This way a fast regular start-up can be achieved for the IO-Link system.

11.2 IOL-M power-on

Typical behavior examples are

- *Default start-up:* I/O configuration defined by GSD is default "DI". All ports are set to DI mode. IOL-M acts like a normal DI module.
- *GSD driven start-up:* GSD carries definitions for port parameters, 10 bytes anonymous IOL-D parameters, and the I/O data lengths. Input data of IOL-Ds are inactive. Output data are not issued to the IOL-Ds. After the PB-DP /PN-IO start-up the IOL-M received the necessary information to configure the ports and to parameterize the IOL-D. Subsequently the input and output data of IOL-Ds are effectiv for the fieldbus cyclic data exchange.
- *Regular start-up:* Valid port configurations and IOL-D parameter sets available upon PB-DP /PN-IO start-up due to remanent storage. Input and output data of IOL-Ds are effectiv immediately without diagnosis messages.
- *No remanence but iPar-Server:* No valid port configurations and IOL-D parameter sets available upon PB-DP /PN-IO start-up. I/O configuration through GSD is not default DI for all the ports. Input data of IOL-Ds are inactive. Output data are not issued to the IOL-Ds. IOL-M restores the actual parameter sets per request (notification) from the iPar-Server. After an IOL-M "reconfiguration" a regular start-up takes place.
- *Neither remanence nor iPar-Server:* No valid port configurations and IOL-D parameter sets available upon PB-DP /PN-IO start-up. I/O configuration through GSD is not default DI for all the ports. Input data of IOL-Ds are inactive. Output data are not issued to the IOL-Ds. A

diagnosis message "wire break" is casted. The IO-Link system waits on PCT to get configuration and parameterization data.

11.3 IOL-D start-up

The start-up behavior of an IOL-D is described in [8]. It has the following impact on the PB-DP/ PN-IO communications:

- During start-up of an IOL-D no input or output data are mapped onto the PB-DP/ PN-IO Slot /Subslot.
- Input data of IOL-Ds are inactive.
- The port qualifier information ("PDinvalid" flag) shall be set "invalid" if necessary.
- During start-up of an IOL-D a nominal /actual comparison between a configured and a connected IOL-D is carried out.
- Diagnosis messages are casted on PB-DP /PN-IO as necessary.

11.4 IOL-D configuration check

At each and every IOL-D start-up a nominal /actual comparison between a configured and a connected IOL-D is carried out. This check is a fieldbus independent function and is to be performed by the port configuration unit as will be described in the future version 1.1 of the IO-Link specification (Figure 1).

The user is able to set the inspection level during port configuration. The following comparison or inspection levels are defined:

- *Level 1:* "No check" → No inspection takes place
- *Level 2:* "Profile device" → The inspection includes the IO-Link *Function_ID* only. IOL-Ds from different vendors and with different Device_IDs are accepted. This is possible for IOL-Ds conforming to a certain device profile.
- *Level 3:* "Vendor specific" → The inspection includes the IO-Link *Function_ID* and the *Vendor_ID*. Vendor is able to distinguish IOL-Ds.
- *Level 4:* "Type equivalent" → The inspection includes the IO-Link *Vendor_ID* and the *Device_ID*. The IOL-Ds may differ but parameterization is the same.
- *Level 5:* "Identical Device" → The inspection includes the IO-Link *Vendor_ID*, the *Device_ID*, and the serial number. This way inadvertently permuted IOL-D on the ports can be detected.

The result "mismatch" shall always be indicated by the diagnosis message "wire break“.

11.5 Change of port configurations or I/O mapping

A change of the port configuration or I/O mapping is usually carried out with the help of a PCT. Optionally a GSD based PB-DP /PN-IO startup is possible.

- Any change of the port configuration or I/O mapping causes a port and IOL-D start-up.
- During the IOL-D start-up no I/O mapping for this port is carried out. Substitute values (IODD) are used instead and a diagnosis message for this port is casted.

11.6 Plug-in of an IOL-D

Plug-in of an IOL-D is recognized by the IO-Link mechanism and causes a start-up of this port and of the IOL-D. The definitions of 11.3 apply. Exception: an IOL-D operating in SIO mode after "fallback". Pull and plug of such an IOL-D cannot be detected automatically and thus manual interaction for example via a "reconfiguration" command is needed.

11.7 PB-DP /PN-IO start-up

The PB-DP /PN-IO start-up occurs independently from the start-up of the ports and the IOL-Ds. The IOL-M receives the configured I/O configuration via GSD means and prepares the corresponding I/O mapping per slot /subslot. Input PDUs are initialized with manufacturer specific substitute values, for example "0".
A nominal /actual check for IOL-Ds is not performed.

11.8 Port qualifier (PQ)

Normally it is not possible to detect the validity of I/O data (IOL-D available). If needed a port specific additional qualifier information (PQ) can be added to the input PDU. Usually the IO-Link flag "PDinvalid" reflects the validity of the port input data.

11.9 Port functions

The future version 1.1 of the IO-Link specification (Figure 1) will incorporate some *port functions* an IOL-M shall be able to perform upon adequate requests. These port functions are fieldbus independent and thus are subject of the above mentioned specification. The corresponding function invocations (commands) via PROFIBUS / PROFINET are specified herein in clause 9.3.2.

The port functions comprise but are not limited to the following examples:

- *Wakeup*: a master port currently in SIO mode sends a particular signal to an IOL-D, prompting its change to the IOL communication mode. A standard sensor or actuator without IO-Link functionality will not respond as expected and thus the port will remain in DI mode.
- *Fallback:* a master port currently in IOL communication mode sends a particular message to an IOL-D, prompting its change to DI or DO mode ("SIO").
- *Reconfiguration:* a master port currently in SIO mode will use a combination of *Wakeup* to temporarily communicate with the IOL-D (identification, parameterization, etc.) and *Fallback* to eventually switch back to DI or DO mode.

11.10 IOL-D substitute values

IO-Link established its own substitute value concept. The substitute values for a particular IOL-D (actuator) can be defined in the associated IODD and transferred to the IOL-D through parameterization (PCT). There are no retroactive effects to an IOL-M. Two main possibillities exist:

- Individual substitute value defined via IODD or
- Special parameter "Hold last PD value" alternatively

Upon a Master Command "0x99" (Process output data invalid or not available) the IOL-D will activate the parameterized substitute value.

12 IO-Link device description (IODD)

An XML based IO-Link device description language is specified in [10].

13 Conformance test & certification

No test and certification for IOL-M on PB-DP or PN-IO defined yet.

Bibliography

[1] Specification for PROFIBUS Device Description and Device Integration, Volume 1: GSD, V5.04; Order No. 2.122

[2] GSDML Specification for PROFINET IO, V2.1; Order No. 2.352

[3] PNO Profile Guidelines, Part 1: *Identification & Maintenance Functions*, V1.1; Order No. 3.502

[4] PNO Profile Guidelines, Part 2: *Data Types, Programming Languages, and Platforms*, V1.0; Order No. 3.512

[5] PNO Profile Guidelines, Part 3: *Diagnosis, Alarms, and Time Stamping*, V1.0; Order No. 3.522

[6] PNO specification: *Communication Function Blocks on PROFIBUS DP and PROFINET IO*, V2.0; Order No. 2.182

[7] PROFIBUS Profile: *Remote I/O for Process Automation*, V1.0; Order No. 3.132

[8] IO-Link Communication Specification V1.0, PNO-Order No. 2.802

[9] IO-Link System Integration V0.9

[10] IO-Link Device Description Language V0.1

[11] PNO specification: *Tool Calling Interface*, V1.0, Order-No. 2.602

[12] FDT Group specification: *FDT Interface Specification*, V1.2.1, March 2005, FDT Group order No. 0001-0001-002 (www.fdt-jig.org)

[13] IEC 61158-5-3:2007, *Industrial communication networks – Fieldbus specifications – Part 5-3: Application layer service definition*

[14] IEC 61158-5-10:2007, *Industrial communication networks – Fieldbus specifications – Part 5-10: Application layer service definition*

[15] IEC 61158-6-3:2007, *Industrial communication networks – Fieldbus specifications – Part 4-3: Application layer protocol specification*

[16] IEC 61158-6-10:2007, *Industrial communication networks – Fieldbus specifications – Part 4-10: Application layer protocol specification*

[17] FDT Group specification: *Device Type Manager (DTM) Style Guide*, V1.0; FDT Group order No. 0001-0008-000 (www.fdt-jig.org)

[18] PNO specification: *PROFIsafe – Profile for Safety Technology on PROFIBUS DP and PROFINET IO*, V2.4, Order-No. 3.192b

[19] FDT Group specification: *Annex: Field Device Tool for IO-Link*, Draft V0.8

Annex A Optional integration features

A.1 Parameter channel for IO-Link device parameters (DP-V0)

A.1.1 Motivation and objectives

Nowadays all the relevant PROFIBUS DP related control systems are supporting the acyclic MS1 services (former "C1" communication) between a master and its slaves and many of them even the mandatory associated communication function blocks according IEC 61131-3 (RD/WR_RECORD, GET/SET_IO, and RDIAG) [6]. This part of the so-called DP-V1 technology is forming a perfect transition to the acyclic services of PROFINET IO based on Ethernet's TCP/IP. However, in order to support the retrofit (modifications or extensions) of existing facilities based on DP-V0 (without MS1 services) an emulation of the acyclic services based on the cyclic services (MS0) is commonly in use. This emulation, called "parameter channel", uses a number of especially reserved process data bytes to exchange data such as parameter values between a user program and a field device, here an IO-Link master and its IO-Link devices. The drawbacks of such a solution, mentioned in 5.2.10 should be considered prior to an implementation.

There are many possible ways to realize such a parameter channel. This specification describes a reasonable procedure especially adapted for IO-Link. The parameter channel can be realized as control software in a PLC. This software is called here "VC1-Driver".

A.1.2 Parameterization channel through cyclic process data

Transmission performs via a so-called virtual C1 module (VC1 module). A C1 module should be selected in the engineering tool (PROFIBUS hardware configuration) in the same way as "normal" I/O devices and then specified for the parameter and configuration telegrams.

The VC1 module is only a virtual device because the process data can be used to transmit communication data and is not linked to a specific module. During active process data exchange, it is possible to assign the VC1 module sequentially to different modules with communication objects and to exchange parameter data parallel to the process data.

*The VC1 module may be configured at any position of the PB-Slave. We recommend configuring the VC1 device in the last position (*Figure 59*). In this way the configured slot and the actual slot occupied by the other IOL-D will always be the same. It is not linked to any hardware, so a module is not actually inserted.*

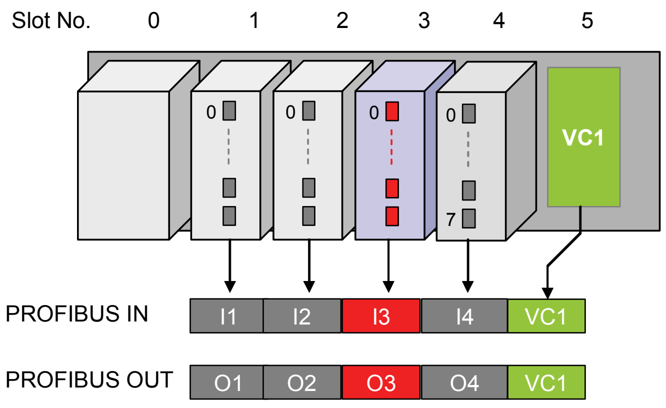


Figure 59 — Recommended position of VC1

A.1.2.1 Process data length of VC1

The process data width occupied by the VC1 module in the process data channel can be selected from 4 (this is reasonable the lower value for IO-Link communication) to 16 words PB IN/OUT data in increments of 2 words. This means that communication objects can be used even if resources are limited. If there are sufficient free resources, a data width of up to 16 words can be used, providing the same ease of operation as for DP/V1 communication.

To avoid incompatibilities, the process data of the VC1-Slave should be consistent over its whole length.

As the data width of the VC1 module is between 4 and 16 words, but the user data can be up to 120 bytes (240 words) per communication, it may be necessary to split the data and transmit it in several steps. This leads to:

- Start fragment
- Continue fragment
- End fragment
- Abort/error fragment

Each fragment contains a service byte, which is used for the precise assignment of the fragment. The individual fragments and the service byte are explained in detail in the following.

A.1.2.2 Start fragment

The start fragment comprises the following structure:

Byte 1:	Service
Byte 2:	Slot number
Byte 3:	Port number
Byte 4:	Index high
Byte 5:	Index low
Byte 6:	Subindex
Byte 7:	Length, if required
Byte 8:	Data Byte, if required
…	
Byte n:	Data Byte, if required

Table 45 — Structure of the service byte 1

Byte 1							
7	6	5	4	3	2	1	0
Request/response	0	0	Fragmentation	Action			

Bit 7:	Request / response 0 = Request 1 = Response
Bit 5 to 6:	Fragment type 00 = Start fragment
Bit 4:	Fragmentation 0 = Not fragmented 1 = Fragmented
Bit 3 to 0:	Action

Table 46 — Action coding

Code	Action	Convention
0x00	No Action (clear)	Mandatory
0x01	Reserved	Mandatory
0x02	Reserved	Mandatory
0x03	Read (IOL-MM)	Optional
0x04	Write(Fast)[1] (IOL-MM)	Optional
0x05	Reserved	Mandatory
0x06	Read SPDU (IOL-D)	Mandatory
0x07	Write SPDU (IOL-D)	Mandatory
0x08	Read and write simultaneously (special objekts only)	Optional
0x09…0x0F	Reserved	Mandatory

A.1.2.3 Continue fragment

The continue fragment comprises the following structure:

Byte 1:	Service
Byte 2:	Data Byte, if required
…	
Byte n:	Data Byte, if required

Table 47 — Service byte of the continue fragment

Byte 1							
7	6	5	4	3	2	1	0
Request/response	0	1	Fragment number (0x01 – 0x1F)				

Bit 7:	Request / response 0 = Request 1 = Response
Bit 5 to 6:	Fragment type 01 = Continue fragment
Bit 4 to 0:	Counter 0x01 to 0x1F fragment number

A.1.2.4 End fragment

The end fragment comprises the following structure:

Byte 1:	Service
Byte 2:	Data Byte, if required
…	
Byte n:	Data Byte, if required

Table 48 — Service byte of the end fragment

Byte 1							
7	6	5	4	3	2	1	0

[1] This service makes a “special quick” Write-Request to IO-Link Devices (more than one SPDU could be buffered in the IOL_CM) possible.

Request/response	1	0	Reserved

Bit 7: Request / response
0 = Request
1 = Response

Bit 5 to 6: Fragment type
10 = End fragment

Bit 4 to 0: Reserved

A.1.2.5 Abort / Error fragment

The abort / error fragment comprises the following structure:

Byte 1: Service
Byte 2: Error code, if required
…
Byte n: Error code, if required

Table 49 — Service byte of the abort / error fragment

Byte 1							
7	6	5	4	3	2	1	0
Request/response	1	1	Reserved				

Bit 7: Request / response
0 = Request
1 = Response

Bit 5 to 6: Fragment type
11 = Abort / error fragment

Bit 4 to 0: Reserved

A.1.3 Description of the transfer

A response is sent after every request. This response indicates that the request has been received and shows its current status.

A.1.3.1 Response structure

Table 50 — Response structure

Byte No.	Content	Meaning
1	*Service*	„mirrored“ of request with set response bit
2	*Status /* Data Byte	*Status* only on first read response, end or error fragment, data if required
3	*Length* / Data Byte	*Length* only on first read response only, Data Byte if required
4	Data Byte	if required
…	Data Byte	if required
n	Data Byte	if required

A.1.3.1.1 Response status

The status is indicated when local IO-Link transmission is complete and in the event of an error. In the event of an error, the data can provide details. An error has occurred if the value of the status byte does not equal 0x00.

A.1.3.1.2 Error types

Table 51 — Error codes

Error_Code_1	Error_Code_2	Meaning
0x00	0	No error, transfer ok.
0x44	0	Error during communication
0xB0	0	Index Invalid
0xB1	0	Invalid data length when valid
0xB2	0	Invalid device number
0xB5	0	Status conflict, last read/write not finished yet
0xB6	0	Access to device or index not permitted
0xB7	0	Invalid parameter
0xC3	0	(Internal) resource not available
other	other	to be defined (see “IO-Link Specification”)

A.1.3.2 Request structure

The meaning of the VC1 parameters is defined in Table 52:

Table 52 — Request structure

Byte No.	Content	Meaning
1	*Service*	Depends on Read- or Write-Request
2	*Slot number* / Data Byte	*Slot number* only on Start fragment, Data Byte if required
3	*Port number* / Data Byte	*Port number* only on Start fragment, Data Byte if required
4	*Index High* / Data Byte	*Index High* only on Start fragment, Data Byte if required
5	*Index Low* / Data Byte	*Index Low* only on Start fragment, Data Byte if required
6	*Subindex* / Data Byte	*Subindex* only on Start fragment, Data Byte if required
7	*Length* / Data Byte	*Length* only on Start fragment, Data Byte if required
8	Data Byte	if required
…	Data Byte	if required
n	Data Byte	if required

A.1.3.2.1 Slot number

The length for the *Slot Number* is one byte. The *Slot Number* is always ”0“ in case of a IOL-CM (Figure 60).

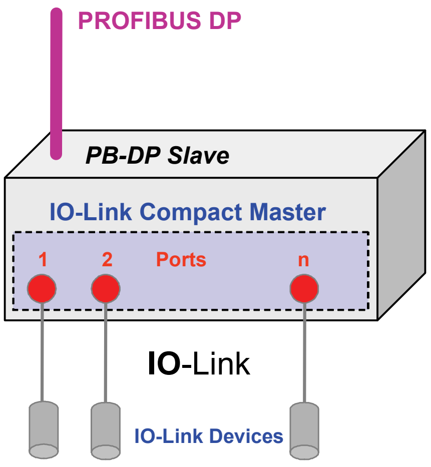


Figure 60 — Slot number in a compact slave device

The *Slot Number* ranges from "1" to "n" in case of an IO-Link master module in a physically modular remote I/O (Figure 61).

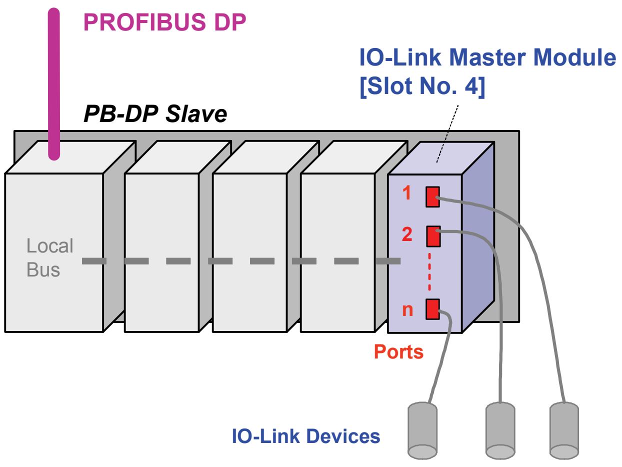


Figure 61 — Slot numbers in a modular remote I/O device

A.1.3.2.2 Port (Start fragment, byte 3)

The *Port* parameter is one byte, contains a number, and thus specifies the port of an IO-Link master (Figure 60) connected to a particular IO-Link device:

0 addresses the IO-Link master itself (e.g. for peripheral control, or port configuration)

1 addresses the IO-Link device connected to port 1

….

n addresses the IO-Link device connected to port n

A.1.3.2.3 Index High and Index Low (Start fragment, byte 4+5)

This parameter specifies the object index of the addressed object of the IOL-D in two bytes. This is true for objects of the IOL-MM. Table 53 is showing some examples.

Table 53 — High / low index examples

Example	Index [dec]	Index High [hex]	Index Low [hex]

1	244	00	F4
2	1702	06	A6

A.1.3.2.4 Subindex (Start fragment, byte 6)

The *Subindex* parameter has a length of one byte. When working with an object, the *Subindex* can be used to select a specific element from an array or record. Therefore the *Subindex* should be specified when accessing IOL-D.

Subindex 0 = all elements of the object (of Index) (max 240 byte)

Subindex 1 = first element of the object (of Index)

…

Subindex n = nth element of the object (of Index)

A.1.3.2.5 Length (Start fragment, byte 7)

This value specifies how many bytes of data (object contents) follows. Depending on the addressed *Port* number, this may be IOL-D data or IOL-MM data.

A.1.3.2.6 Data

This is only the content of an object. The length and scope of the data is already described by the *Length* parameter.

The next two clauses are providing examples of data objects during transmission processes.

A.1.3.2.7 Example 1: Reading (VC1 Length 8 words)

In this case the connecting-state of the IO-Link ports of an IOL-MM are read back.

As a precondition, the IOL-MM may have the objects in ….

Table 54 — Example objects of an IOL-MM

Index No.	Content / Function	Access
5	Status of IO-Link connections, per port = 3 bytes Byte 1 = Port number Byte 2 = Status of connection 0x00 = port is in SIO Mode 0x01 = port is in Com Mode (scan was successful and IO-Link communication “runs”) 0xFF = error, port is in Scan Mode but no connection possible (maybe no IOL-D?) Byte 3 = reserved (0x00)	Read only

Read request (Master -> Slave)

PB OUT Data of VC1 (8 words [hex])	Data structure
03 / 00 / 00 / 00 05 / 00 / xx xx xx xx xx xx xx xx xx xx	Read-Request IOL-MM / Slot nr. / Port nr. / Index High Index Low / Subindex / 10 unused bytes

Read response (Slave -> Master)

PB IN Data of VC1 (8 words [hex])	Data structure
83 / 00 / 0C / 01 01 00 02 00 00 03 01 00 04 FF 00 / xx	Read-Response IOL-MM / Status / Length / 12 Data bytes / 1 unused byte

Clear request (Master -> Slave)

PB OUT Data of VC1 (8 words [hex])	Data structure
00 / xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx	Clear request / 15 unused bytes

Read-Response (Slave -> Master)

PB IN Data of VC1 (8 words [hex])	Data structure
00 / xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx	Clear response / 15 unused bytes

The *Status* byte equals zero. This means that communication via the VC1 "device" was error-free. The data shows that there is IO-Link communication each on port 1 and 3. Port 4 is configured to Scan Mode but not able to detect an IOL-D. Port 2 is configured to SIO-Mode.

The communication data (VC1) can be reset to the initial state via "Clear".

A.1.3.2.8 Example 2: Writing (VC1 Length 4 words)

In this case an object of an IOL-D connected to an IO-Link master is written in fragmented form: Port = 4, Index 67, Subindex 3, Length 10, Data “Everything ok“)

Data = 45 76 65 72 79 74 68 69 6E 67 20 6F 6B

Write-Request (Master -> Slave)

PB OUT Data of VC1 (4 words [hex])	Data structure
14 / 00 / 04 / 00 43 / 03 / 0D / 45	Write request IOL-MM / Slot nr. / Port nr. / Index High Index Low / Subindex / 2 bytes object data

Write-Response (Slave -> Master)

PB IN Data of VC1 (4 words [hex])	Data structure
14 / xx xx xx xx xx xx xx	Response / 7 unused bytes

Write -Request (Master -> Slave)

PB OUT Data of VC1 (4 words [hex])	Data structure
21 / 76 65 72 79 74 68 69	first continue fragment / 7 Data bytes

Write -Response (Slave -> Master)

PB IN Data of VC1 (4 words [hex])	Data structure
21 / xx xx xx xx xx xx xx	Response / 7 unused bytes

Write-Request (Master -> Slave)

PB OUT Data of VC1 (4 words [hex])	Data structure
40 / 6E 67 20 6F 6B / xx xx	End fragment / 5 Data byte / 2 unused bytes

Write -Response (Slave -> Master)

PB IN Data of VC1 (4 words [hex])	Data structure
84 / 00 / xx xx xx xx xx xx	**response / Status** / 6 unused bytes

Clear Request (Master -> Slave)

PB OUT Data of VC1 (4 words [hex])	Data structure
00 / xx xx xx xx xx xx xx	Clear request / 7 unused bytes

Clear-Response (Slave -> Master)

PB IN Data of VC1 (4 words [hex])	Data structure
00 / xx xx xx xx xx xx xx	Clear response / 7 unused bytes

This write request will be executed in 3 fragments. The IOL-MM responds with the “response / Status” ”84 / 00“, the success of the write request.

The communication data (VC1) can be reset to the initial state via "Clear".

A.1.4 Flow charts

The flow charts refer to a VC1 length of 8 bytes in the PB process data. Figure 62 is showing the main flow chart of the VC1 parameter channel between the VC1-Driver program and an IO-Link master.

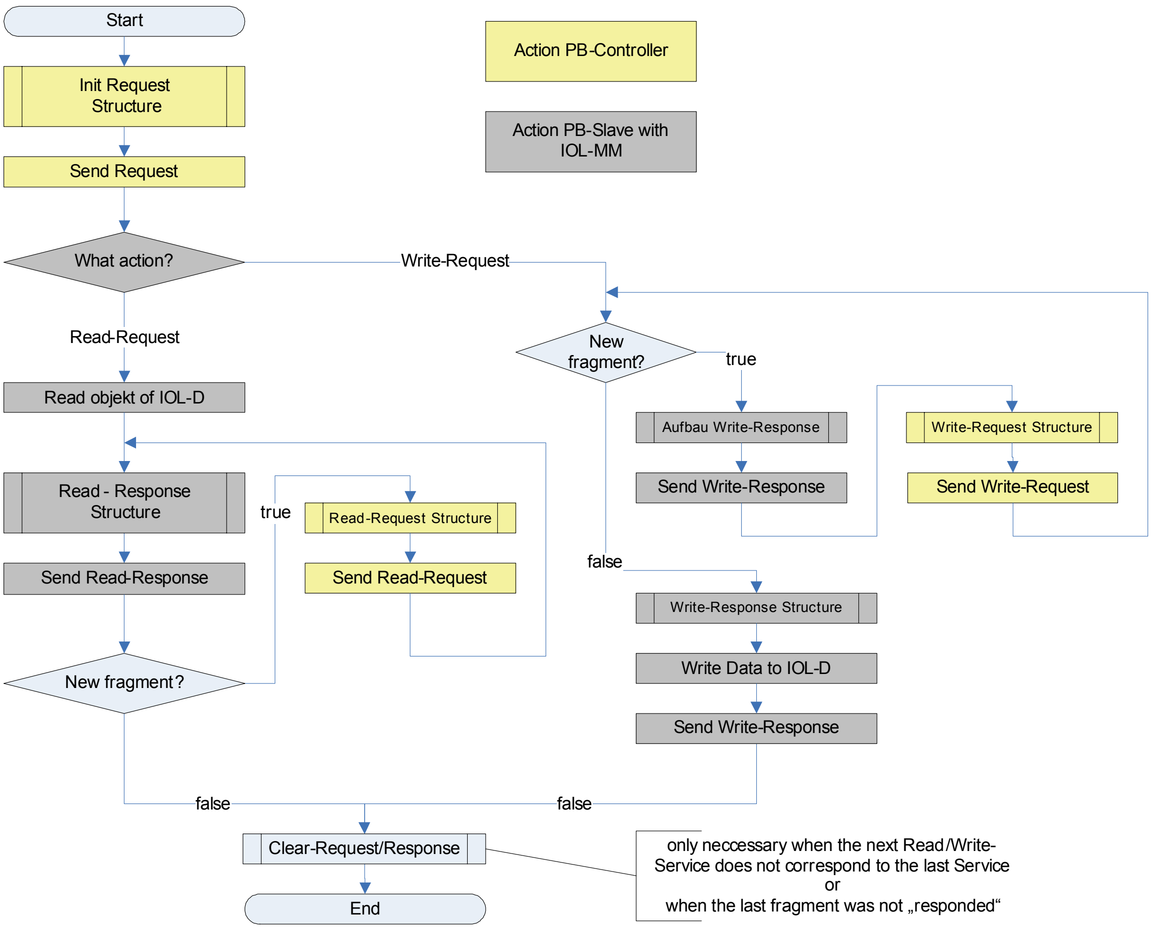


Figure 62 — Main flow of the VC1 parameter channel

A.1.4.1 Init Request Structure

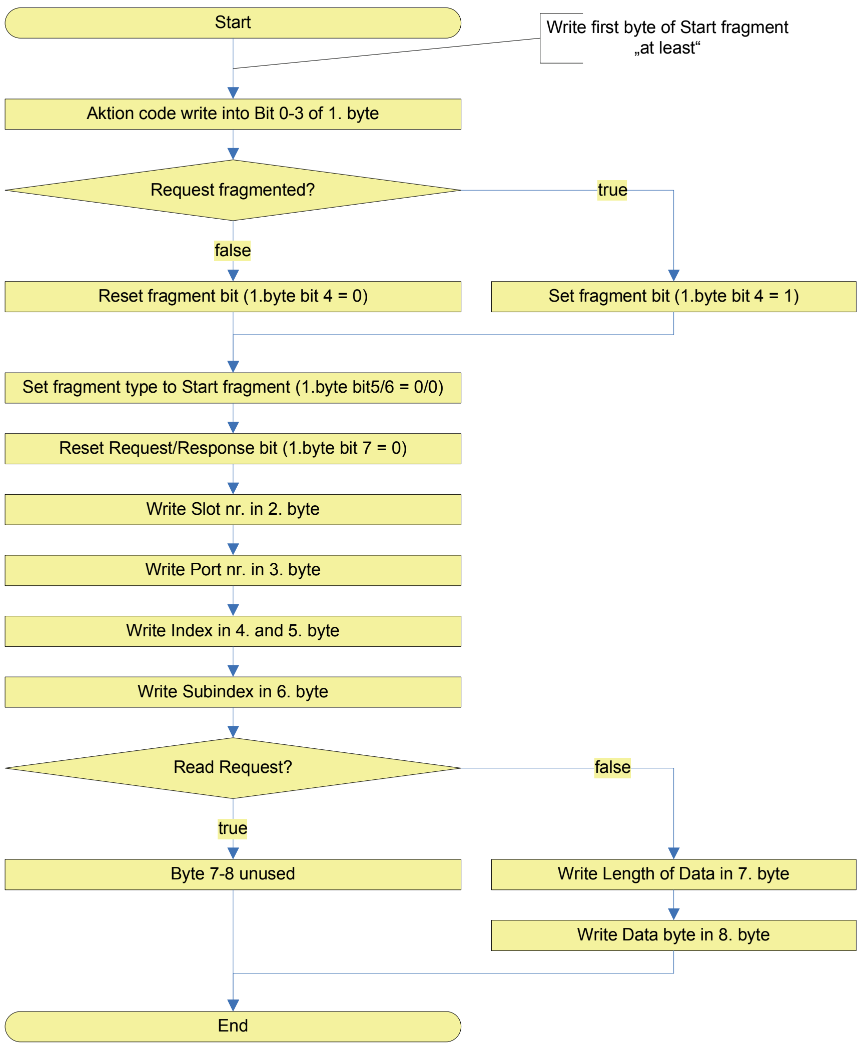

Start
Write first byte of Start fragment „at least“
Aktion code write into Bit 0-3 of 1. byte
Request fragmented?
true
false
Reset fragment bit (1.byte bit 4 = 0)
Set fragment bit (1.byte bit 4 = 1)
Set fragment type to Start fragment (1.byte bit5/6 = 0/0)
Reset Request/Response bit (1.byte bit 7 = 0)
Write Slot nr. in 2. byte
Write Port nr. in 3. byte
Write Index in 4. and 5. byte
Write Subindex in 6. byte
Read Request?
false
true
Byte 7-8 unused
Write Length of Data in 7. byte
Write Data byte in 8. byte
End

Figure 63 — Init request

A.1.4.2 Read-Response Structure

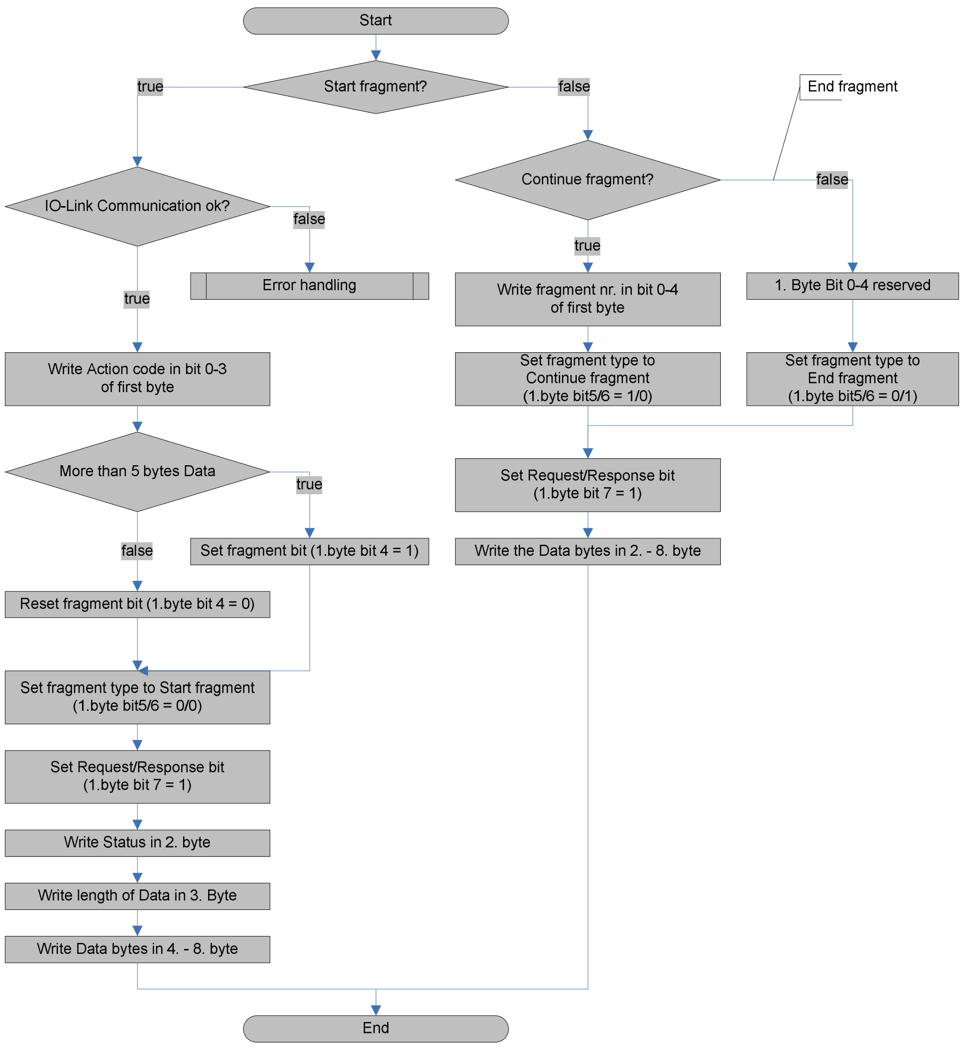

Start
Start fragment?
true
false
End fragment
IO-Link Communication ok?
false
Error handling
true
Write Action code in bit 0-3 of first byte
More than 5 bytes Data
true
Set fragment bit (1.byte bit 4 = 1)
false
Reset fragment bit (1.byte bit 4 = 0)
Set fragment type to Start fragment (1.byte bit5/6 = 0/0)
Set Request/Response bit (1.byte bit 7 = 1)
Write Status in 2. byte
Write length of Data in 3. Byte
Write Data bytes in 4. - 8. byte
Continue fragment?
true
Write fragment nr. in bit 0-4 of first byte
Set fragment type to Continue fragment (1.byte bit5/6 = 1/0)
false
1. Byte Bit 0-4 reserved
Set fragment type to End fragment (1.byte bit5/6 = 0/1)
Set Request/Response bit (1.byte bit 7 = 1)
Write the Data bytes in 2. - 8. byte
End

Figure 64 — Read response

A.1.4.3 Read-Request Structure

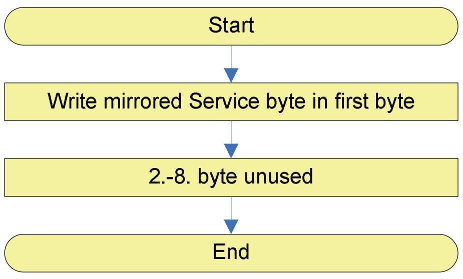


Figure 65 — Read request

A.1.4.4 Write-Response Structure

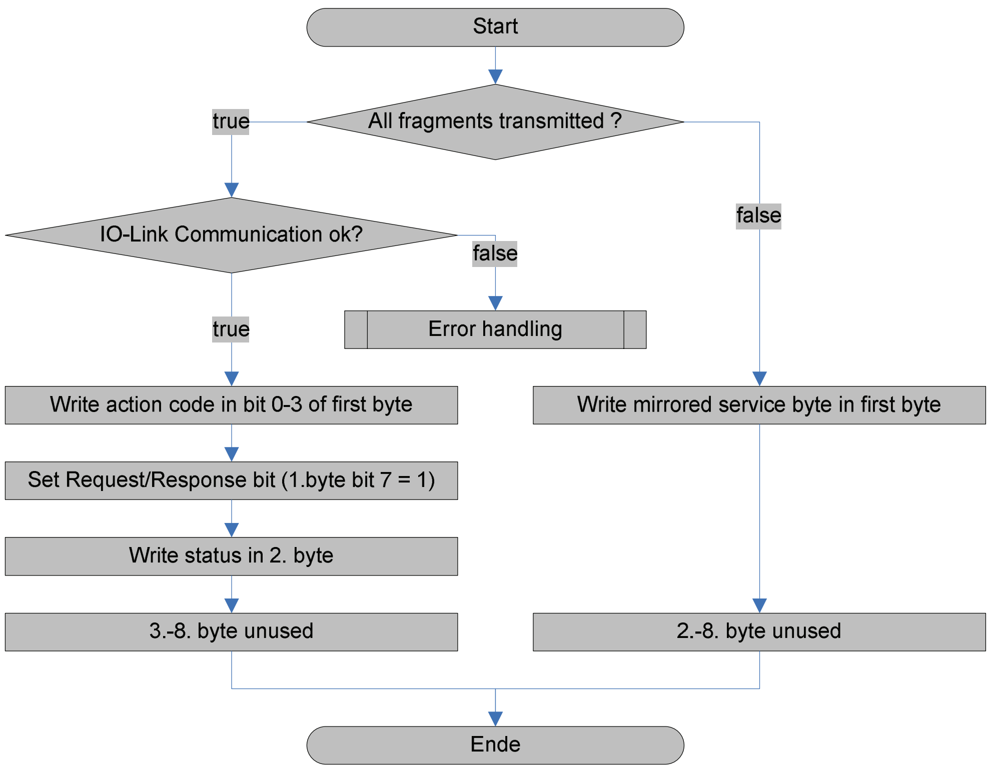

Start
All fragments transmitted ?
true
false
IO-Link Communication ok?
false
true
Error handling
Write action code in bit 0-3 of first byte
Set Request/Response bit (1.byte bit 7 = 1)
Write status in 2. byte
3.-8. byte unused
Write mirrored service byte in first byte
2.-8. byte unused
Ende

Figure 66 — Write response

A.1.4.5 Write(Fast) Response Structure

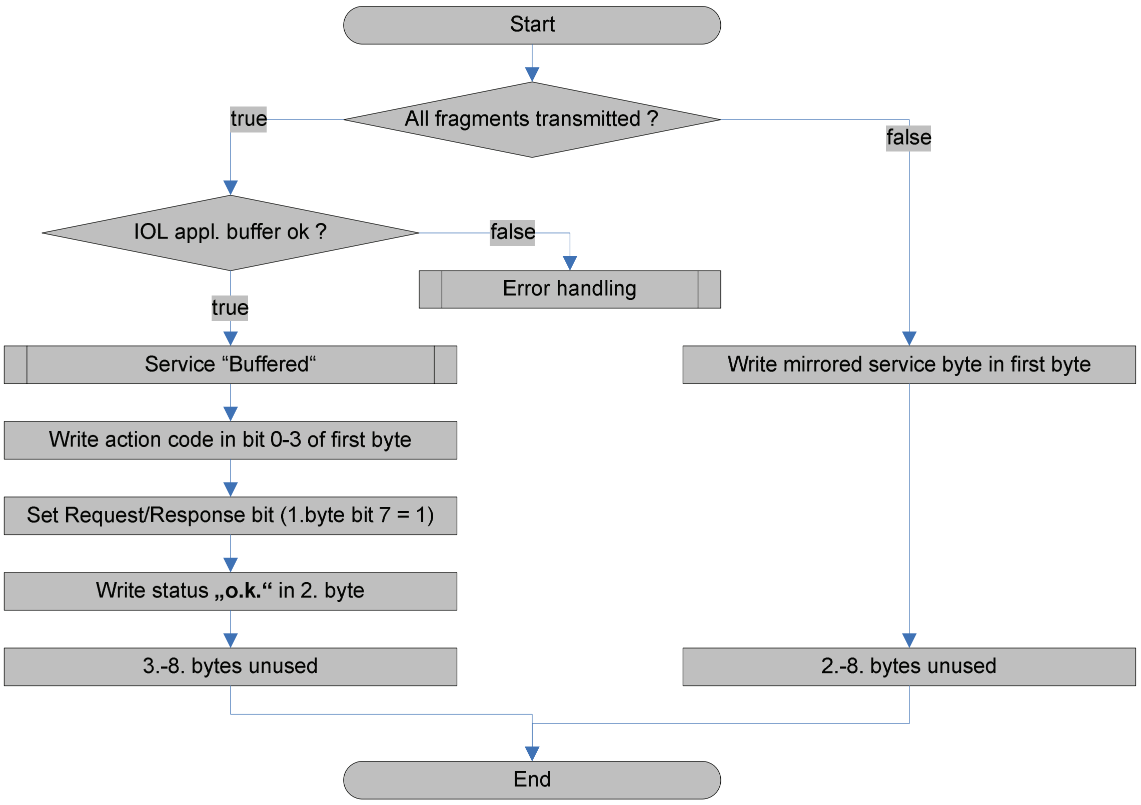

Start
All fragments transmitted ?
true
false
IOL appl. buffer ok ?
false
Error handling
true
Service “Buffered“
Write mirrored service byte in first byte
Write action code in bit 0-3 of first byte
Set Request/Response bit (1.byte bit 7 = 1)
Write status „o.k.“ in 2. byte
3.-8. bytes unused
2.-8. bytes unused
End

Figure 67 — Write(Fast) response

A.1.4.6 Write-Request Structure

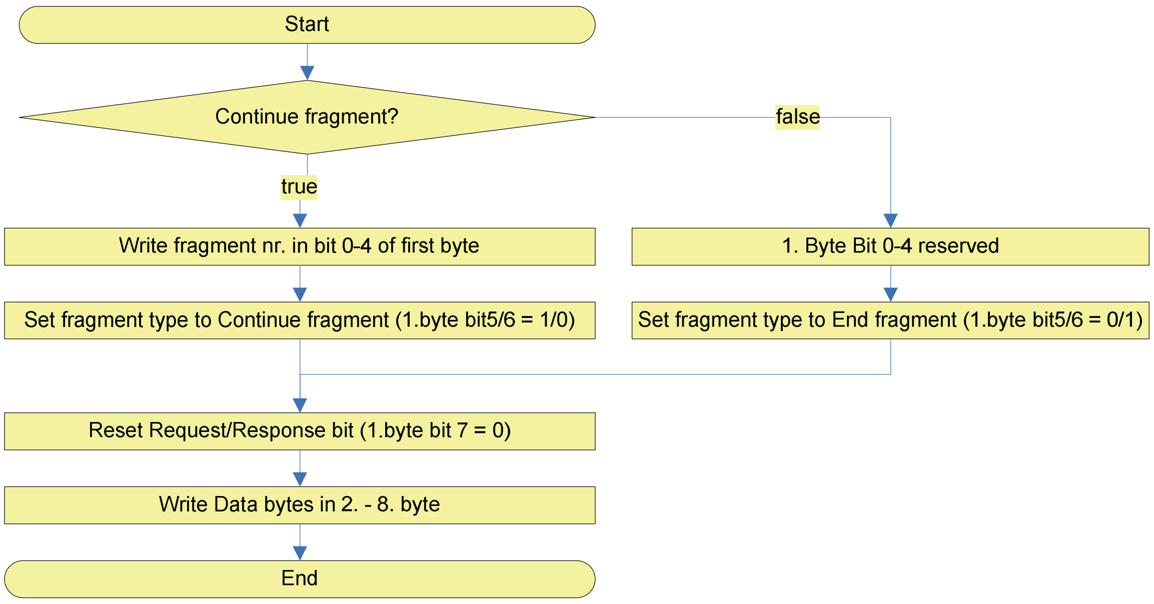

Start
Continue fragment?
false
true
Write fragment nr. in bit 0-4 of first byte
1. Byte Bit 0-4 reserved
Set fragment type to Continue fragment (1.byte bit5/6 = 1/0)
Set fragment type to End fragment (1.byte bit5/6 = 0/1)
Reset Request/Response bit (1.byte bit 7 = 0)
Write Data bytes in 2. - 8. byte
End

Figure 68 — Write request

A.1.4.7 Clear-Request / Clear-Response Structure

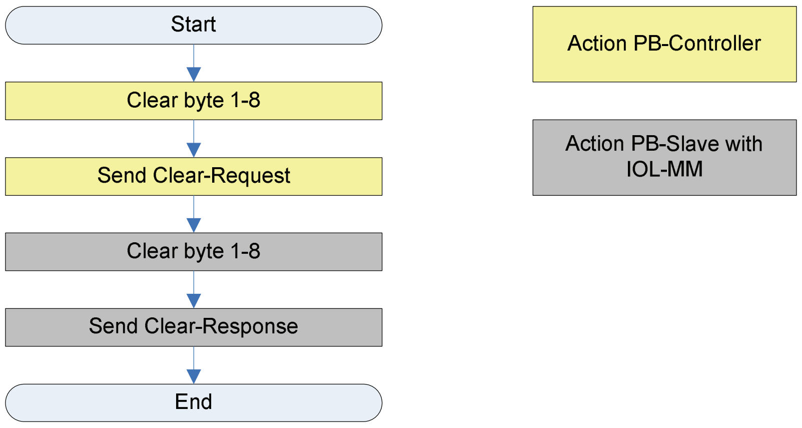


Figure 69 — Clear response

A.1.4.8 Error Handling Structure

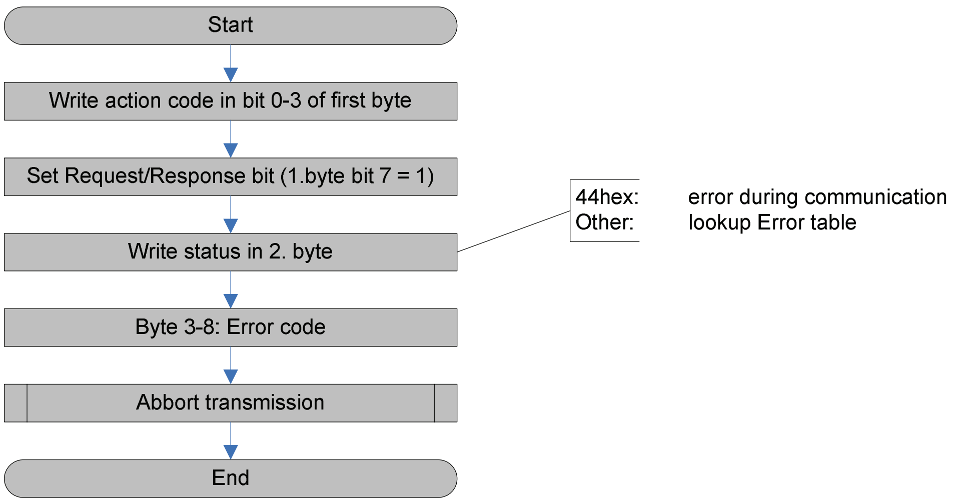

Start
Write action code in bit 0-3 of first byte
Set Request/Response bit (1.byte bit 7 = 1)
Write status in 2. byte
Byte 3-8: Error code
Abbort transmission
End
44hex: error during communication
Other: lookup Error table

Figure 70 — Error handling

A.1.4.9 Service "Buffered"

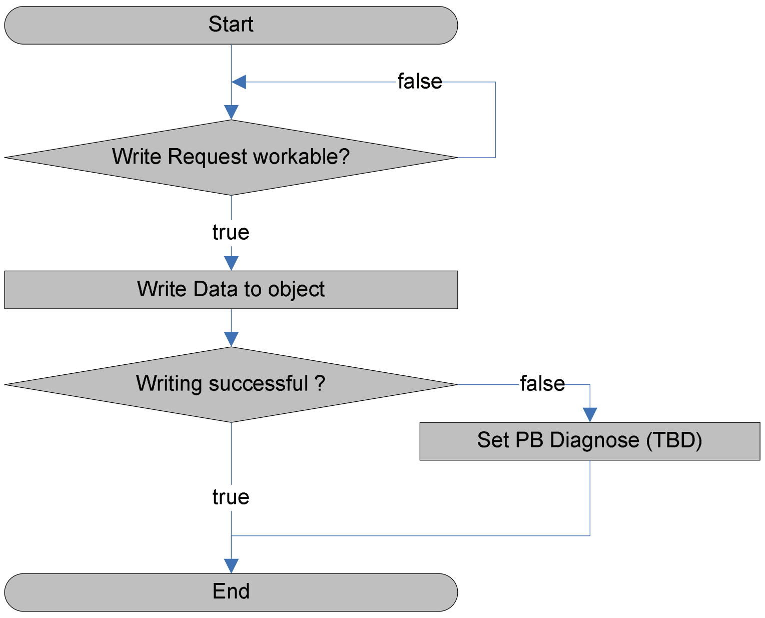


Figure 71 — Service "buffered"

A.1.4.10 Abort Structure

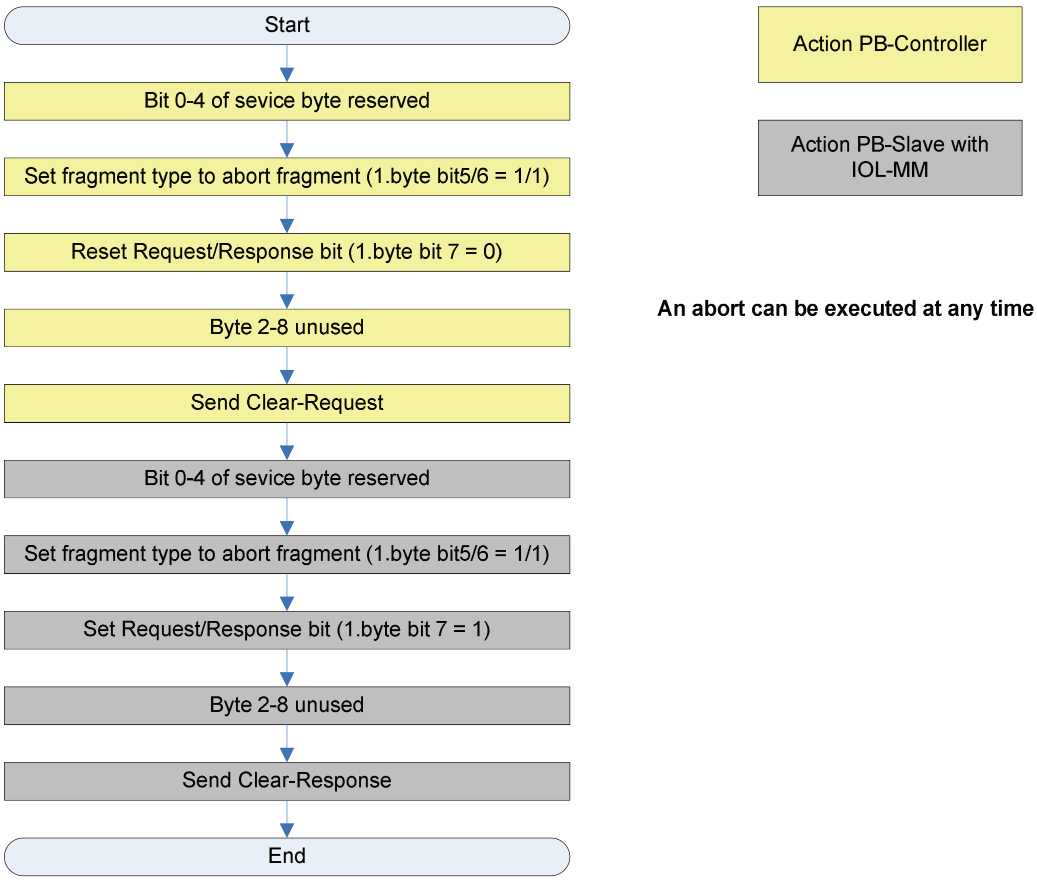

Start
Bit 0-4 of sevice byte reserved
Set fragment type to abort fragment (1.byte bit5/6 = 1/1)
Reset Request/Response bit (1.byte bit 7 = 0)
Byte 2-8 unused
Send Clear-Request
Bit 0-4 of sevice byte reserved
Set fragment type to abort fragment (1.byte bit5/6 = 1/1)
Set Request/Response bit (1.byte bit 7 = 1)
Byte 2-8 unused
Send Clear-Response
End
Action PB-Controller
Action PB-Slave with IOL-MM
An abort can be executed at any time !

Figure 72 — Abort

Annex B Main focus types of IO-Link masters

At this stage of IO-Link it is not possible to specify exactly the features of several levels of IO-Link masters (classes). However, three main focus types with some feature variations can be identified that are presented as informative annex in Table 55. It serves as a modest guide for designers and customers.

Table 55 — Main focus types of IO-Link masters

Feature	Type A	Type B	Type C	Remark
Integration concept (engineering), GSD based				
IO data	✓	✓	✓	IO structure description per (sub)module
Port config.	✓	(-)	(-)	"User parameter" for port configurations
IOL-D Para.	✓	(-)	(-)	10 Bytes anonymous IOL-D parameters
Integration concept (engineering), PCT based				
Port config.	(-)	✓	✓	Parameter for port configurations
IOL-D Para.	(-)	✓	✓	10 Bytes anonymous IOL-D parameters
IODD (XML)	(-)	✓	✓	IODD interpreter / checker within PCT
IOPD (tool)	(-)	✓	✓	IOL-D tools online via TCI/FDT
IOL-M function, startup port configuration check				
Level 1	✓	✓	✓	"No check"
Level 2	(-)	✓	✓	"Profile device" (Function_ID only)
Level 3	(-)	✓	✓	"Vendor specific" (Vendor_ID + Function_ID)
Level 4	(-)	✓	✓	"Type equivalent" (Vendor_ID + Device_ID)
Level 5	(-)	(-)	✓	"Identical device" (dito + serial number)
IOL-M function, IO data				
Input data	✓	✓	✓	Range 1…n Bytes, n = vendor specific limit
Output data	✓	✓	✓	Range 1…n Bytes, n = vendor specific limit
Addressing	Multi-…	Single/Multi-…	Single/Multi-…	Single or multi-(sub)-slot model
IOL-M function, events / diagnosis (runtime)				
Errors	C-R-D	C-R-D	ext. C-R-D, S.M.	extended C-R-D, S.M. =Status Message
Warnings	(-)	(-)	ext. C-R-D, S.M.	dito
Messages	(-)	(-)	ext. C-R-D, S.M.	dito
IOL-M function, user program controlled parameterization				
Proxy-FB	ParChannel-FB	IOL_CALL	IOL_CALL	5.2.10, "Parameter tunneling"
IOL-M function, identification & maintenance (I&M)				
Mapping	(-)	IM0	IM0 + IM16…	IM0 = IOL-M, IM16 = Port 1, etc.
IOL-M function, add-on functions and parts replacement				
Fallback	(-)	✓	✓	
IOL-D failed	limited	✓	✓	GSD (10 Bytes) / IOL-M parameter server
IOL-M failed	(-)	✓	✓	iPar-Server
Legend: ✓ supported (-) normally not supported				

This page intentionally blank.