PROFIBUS • PROFINET

# *IO-Link Integration – Edition 2*

## *Guideline for PROFINET*

Version 1.0 – June 2017
Order No.: 2.832

**File name: IO-Link-Integration-for-PROFINET_Ed2_2832_V10_Jun17**

# IO-Link Integration – Edition 2

## Guideline for PROFINET

This Version 1.0 of Edition 2 of IO-Link Integration for PROFINET has been prepared by the PI Project Group 5 in C4.

The mandatory clause "Management summary – scope of this document" for PNO documents is covered by the clauses "INTRODUCTION" and "Scope". Known patents are listed in clause "INTRODUCTION".

Any comments, proposals, requests on this document are appreciated. Please use the database www.io-link-projects.com for your entries and provide *name* and *email address*.

Login: **IO-Link-Integration-Ed2**
Password: **Report**

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

**NOTICE:**

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

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

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

PROFIBUS® and PROFINET® logos are registered trade marks. The use is restricted for members of PROFIBUS & PROFINET International. More detailed terms for the use can be found on the web page www.profibus.com/download. Please select button "Presentations & Logos".

In this guideline the following key words 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. Designer 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 (0) 721 / 96 58 590**
**Fax: +49 (0) 721 / 96 58 589**
**E-mail: info@profibus.com**
**http://www.profibus.com**

# CONTENTS

# 1 Management summary – Scope of this document

## 1.1 Objectives

It is the purpose of this document to map the IO-Link technology according to IEC 61131-9 into PROFINET in such a manner that the user benefit is maximized through standardization.

From a PROFINET point of view this can be achieved by pursuing the following objectives:

- IO-Link mechanisms should be transformed into PROFINET mechanisms to hide the differences as much as possible ("virtual PROFINET devices");
- "Look & Feel" of the deviations should be harmonized across manufacturer borders through standardization;
- It should be possible to use one of the IO-Link Tools (PDCTs) for several IO-Link Masters from different manufacturers/brands through standardized interfaces to the Masters.

## 1.2 PROFIBUS

It is not primary objective of this document to update the mapping into PROFIBUS. The mapping specified in [5] is still valid, whereas the mapping into PROFINET is optimized and replaced by methods specified in this document.

The results will be diverging behaviors of IO-Link systems in PROFIBUS and PROFINET environments. This should be acceptable since existing PROFIBUS equipment is not forced to change and PROFINET equipment can be optimized.

## 1.3 IO-Link engineering

IO-Link Devices are described via a language (IODD) independent from fieldbuses (see [2]). The corresponding description files are installed in PDCTs (Port and Device Configuration Tool), where the user can read Device information and assign parameter values.

For PROFINET environments the upper level Engineering Systems are responsible to handle similar requirements for the fieldbus devices via GSD files.

It is not an objective for this document to provide means for the integration of IO-Link Device descriptions (IODD) into GSD files with the help of "GSD Composers" but rather to specify the integration of the PDCT into the Engineering System via standard interfaces.

Thus, the task for this integration is two-fold:

- Integration of the PDCTs into Engineering Systems via existing standards such as FDT or TCI;
- A new standardized "PDCT interface" to IO-Link Masters for the exchange of IO-Link data objects or status and diagnosis information

## 1.4 Mapping of optional IO-Link functionality

The IO-Link functionality is specified in the IO-Link standard [1]. The mappings of all mandatory functions within this standard are specified within this document.

Table 1 shows optional IO-Link functionality and how it's covered in this document.

**Table 1 – Optional IO-Link functionality**

| Function | Reference in [1] | Remark |
|---|---|---|
| Inspection level: "Identical" | 9.2.3.4 | Comparison of SerialNumbers can be performed via user application. This functionality can be modelled on the basis of clause 9.3. |
| Virtual Port mode "DI-withSDCI" | 11.8.5 | The virtual Port mode can be performed via user application. |
| Anonymous parameter | 11.8.4 | Not required from a PROFINET point of view |
| MessageSync | 11.2.2.2 | Synchronization between Master ports is not required |

# 2 List of affected patents

There are no affected patents known by the members of the IO-Link integration working group. The list is empty. PI does not guarantee the completeness of this list.

# 3 Related documents and references

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

IEC 61131-9, *Programmable controllers – Part 9: Single-drop digital communication interface for small sensors and actuators (SDCI)*

IEC 61158-6-10:2015, *Industrial communication networks – Fieldbus specifications – Part 6-10: Application layer protocol specification – Type 10 elements,* or
PI specification, *PROFINET application layer protocol for decentralized periphery and distributed automation*, Version 2.3, Ed 2, MU4 from 2017, Order No. 2.722

PI Guideline, *Fieldbus integration in PROFINET IO*, Version 2.0, May 2011, Order No. 7.012

PI Specification, *IO-Link Integration, Part 1*, Version 1.0, December 2007, Order No. 2.812

PI Profile Guidelines, Part 1, *Identification & Maintenance Functions*, Version 2.0, January 2014, PROFIBUS & PROFINET International, Order No. 3.502

# 4 Terms, definitions, symbols, abbreviated terms and conventions

## 4.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

**4.1.1**
**Device**
single passive peer to a Master such as a sensor or actuator

NOTE 1 to entry: Uppercase "Device" is used for SDCI (IO-Link) equipment, while lowercase "device" is used in a generic manner.

**4.1.2**
**General Station Description**
GSD
description file of a particular PROFIBUS or PROFINET device

**4.1.3**
**General Station Description Markup Language**
GSDML
XML based language with elements and attributes enabling the description of PROFINET devices

**4.1.4**
**Linking Module**
part of a PROFINET device (IO device) representing the IO-Link functionality

**4.1.5**
**Master**
active peer connected through ports to one up to n Devices and which provides an interface to the gateway to the upper level communication systems or PLCs

NOTE 1 to entry: Uppercase "Master" is used for SDCI (IO-Link) equipment, while lowercase "master" is used in a generic manner.

NOTE 2 to entry: Part of a Linking Module driving the IO-Link Devices

**4.1.6**
**port**
communication medium interface of the IO-Link Master to one Device (sensor/actuator)

**4.1.7**
**PROFIBUS GSD**
PB-GSD
device description of a PROFIBUS master or slave

Note 1 to entry: Use this term only to distinguish between PB-GSD and PN-GSD, otherwise use the term GSD

**4.1.8**
**PROFINET GSD**
PN-GSD
device description of a PROFINET device

Note 1 to entry: Use this term only to distinguish between PB-GSD and PN-GSD, otherwise use the term GSD

## 4.2 Symbols and abbreviated terms

| | | |
|---|---|---|
| AI | analog input | |
| AL | application layer | IEC 61131-9 |
| AO | analog output | |
| API | application process identifier | IEC 61158-6 |
| AR | application relationship | IEC 61158-6 |
| ASE | application service element | |
| ASIC | application specific integrated circuit | |
| ATC | automatic tool changer | |
| CAP | client access point | |
| CC | CommandCode | |
| CPC | consolidated port configuration | |
| CRC | cyclic redundancy check | |
| DAP | device access point | |
| DCP | discovery and basic configuration protocol | |
| DI | digital input | IEC 61131-9 |
| DO | digital output | |
| ES | engineering system | |
| FAL | fieldbus application layer | |
| FB | function block | |
| FDI | field device integration | |
| FDT | field device tool | |
| GSD | general station description | |
| GWA | gateway application layer | |
| ID | identification | |
| IEC | International Electrotechnical Commission (www.iec.ch) | |
| IM or I&M | identification and maintenance | [6] |
| IO or I/O | input / output | |
| IO controller | PROFINET bus controller | |
| IOCS | IO consumer status | |
| IO-Link | point to point communication (single drop) | IEC 61131-9 |
| IO device | PROFINET device | |
| IOD | IO-Link Device | |

| | | |
|---|---|---|
| IODD | IO device description | |
| IOM | IO-Link Master | |
| IOPS | IO provider status | |
| IOxS | IO consumer status or IO provider status | |
| ISDU | indexed service data unit | |
| ISO | International Organization for Standardization (www.iso.ch) | |
| LED | light emitting diode | |
| LSB | least significant bit (byte) | |
| MSB | most significant bit (byte) | |
| OD | on-request data | |
| PC | personal computer | |
| PD | process data | |
| PDCT | port and Device configuration tool | IEC 61131-9 |
| PDU | protocol data unit | |
| PFID | profile identification | |
| PI | PROFIBUS & PROFINET International | |
| PLC | programmable logic controller | |
| PNO | PROFIBUS Nutzerorganisation e.V. (www.profibus.com) | |
| PQ | port qualifier | |
| PQI | port qualifier information | |
| RPC | remote procedure call | |
| SDCI | single-drop digital communication interface | IEC 61131-9 |
| SMID | Submodule identification | |
| TCI | tool calling interface | |
| USI | user structure identifier | |
| UUID | universally unique identifier | RFC 4122 |
| XML | extensible markup language | REC-xml-20081126 |

### 4.3 Conventions

The encoding of values shall be big endian if not otherwise stated in this document.

# 5 Introduction

## 5.1 IO-Link

The system technology (IO-Link) for low-cost sensors and actuators is standardized within IEC 61131-9. It provides a low-cost, digital communication for these devices to exchange process data, diagnosis information and parameters with a controller (PC or PLC) while maintaining backward compatibility with the current DI/DO signals as defined in IEC 61131-2.

Any IO-Link compliant Device can be connected to any available interface port of an IO-Link Master. IO-Link compliant Devices perform physical to digital conversion in the Device, and then communicate the result directly in a standard 24 V I/O digital format, thus removing the need for different DI, DO, AI, AO modules and a variety of cables.

Topology is point-to-point from each Device to the Master using 3 wires over distances up to 20 m. The IO-Link physical interface is backward compatible with signaling specified in IEC 61131-2. Transmission rates of 4,8 kbit/s, 38,4 kbit/s and 230,4 kbit/s are supported.

Tools allow the association of Devices with their corresponding electronic I/O device descriptions (IODD) and their subsequent configuration/parameterization to match the application requirements. For details see www.io-link.com or [1].

## 5.2 PROFINET IO-Link system topology

Figure 1 demonstrates the PROFINET IO-Link system topology.

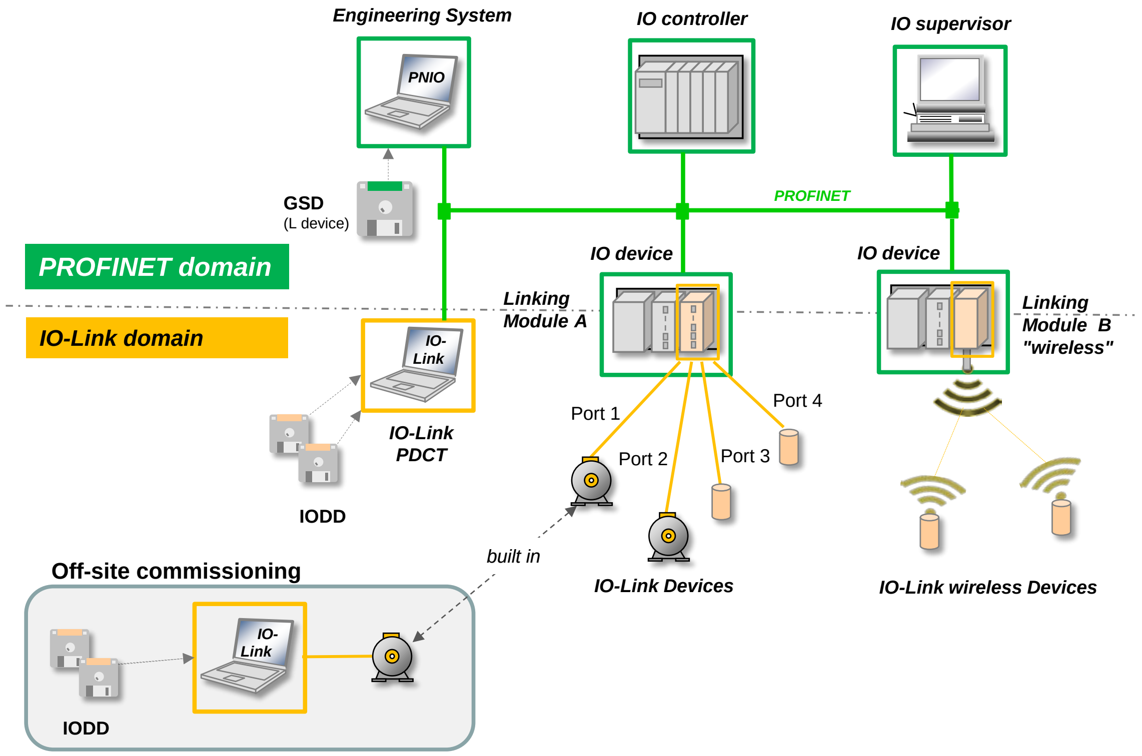



**Figure 1 – PROFINET IO-Link system topology**

The upper part of Figure 1 represents the components of a PROFINET network and is called the PROFINET domain.

An *IO controller* is the center part. It holds the information of all associated network participants and is responsible for parameterization of the IO devices at start-up as well as for cyclic and acyclic communication.

The *IO supervisor* has direct access to the IO devices. This can be done by means of an implicit AR which is only able to read record data. A special "device access AR" offers the possibility to read and write record data. And with the help of a so called "supervisor takeover" a supervisor can get control over an IO device including cyclic IO data. It is important for fieldbus integration, that an IO supervisor cannot have access to a project of the engineering system. Thus, all required information should be offered by the IO devices themselves.

Main task of the *Engineering System* is configuration of the IO controller together with the associated IO devices. This is performed by importing GSD files of the IO devices, by commissioning, and by setting up various device and PROFINET communication parameters. The entire configuration data record can be downloaded into an IO controller.

A *Linking Module* is part of a PROFINET IO device and handles the IO-Link specific information through proxies. One IOLD proxy represents one IO-Link Device. Each Linking Module holds one IO-Link Master instance and thus builds the "gateway" to the IO-Link domain. This document mainly specifies the mapping methods (behavior) of the Linking Modules.

The lower part of Figure 1 represents the components of IO-Link systems. Each IO-Link Master holds 1 to *n* ports where one IO-Link Device per port can be connected and operated.

The configuration of the IO-Link system including the parameterization of the IO-Link Devices can be performed with the help of an IO-Link port and Device configuration tool (PDCT). It uses IODD description files per IO-Link Device for this purpose.

## 5.3 Structure of Linking Modules

Figure 2 shows the structure of Linking Modules with proxies for the Master and for each Device.

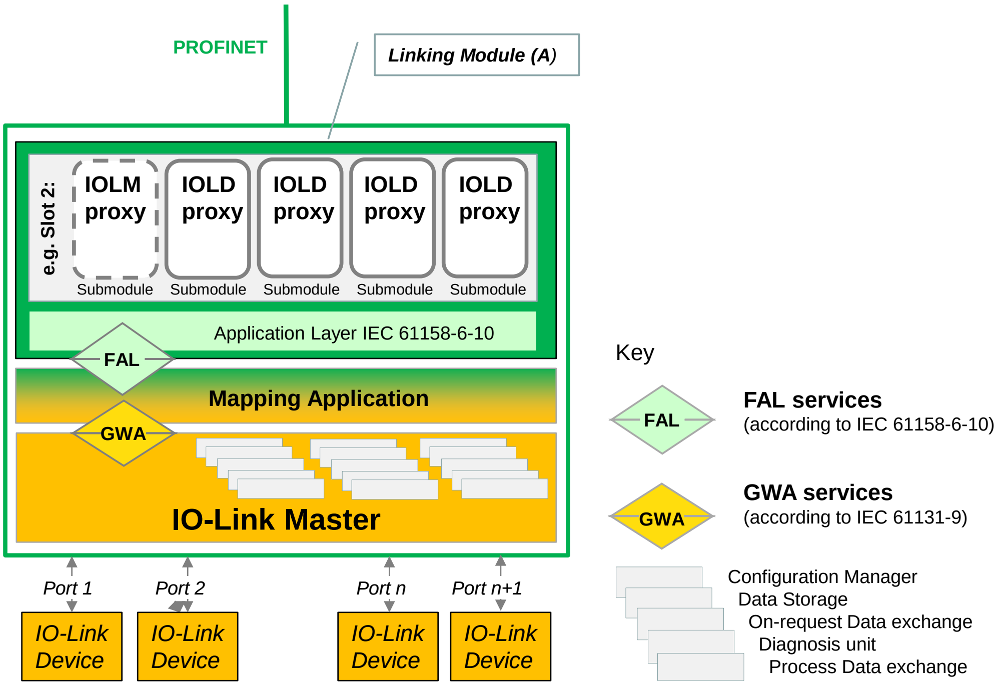



**Figure 2 – Structure of Linking Module**

The standardized mapping structure is implemented in a block called "Mapping Application". The access to the fieldbus or to the IO modules can be carried out by a standard software packet or by an ASIC with appropriate functionality.

The mapping architecture depends on the one hand on PROFINET IO (IEC 61158) fieldbus application layer service definitions (FAL) and on the other hand on IO-Link (IEC 61131-9) gateway application layer service definitions (GWA). The Mapping Application of the Linking Module model to the PROFINET module/submodule model is specified in clause 7.

From a PROFINET point of view it is essential to integrate more than one "Linking Module" in one PROFINET device, especially in case of modular remote IO systems, to achieve a flexible and efficient IO data structure. Figure 3 shows an example of Linking Modules within a remote IO system.

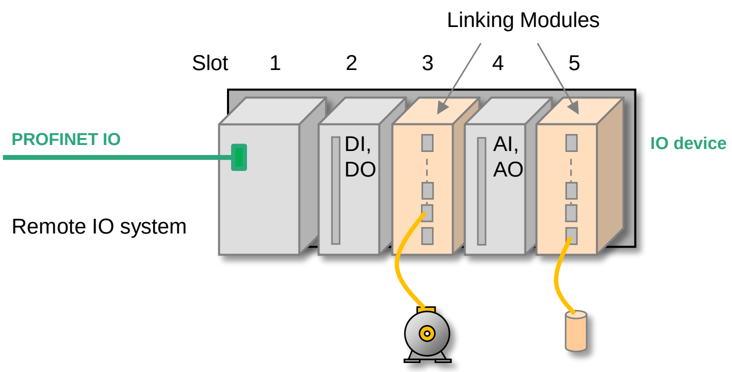



**Figure 3 – Example of Linking Modules within a remote IO**

## 5.4 Service interface

The Mapping Application as shown in Figure 2 handles the IO-Link functionality using two standardized interfaces:

- FAL services (fieldbus application layer) to control the PROFINET IO communication;
- GWA (gateway application services) to control the IO-Link Master and its ports.

### 5.4.1 FAL services

IEC 61158-6-10 specifies the standardized PROFINET fieldbus application layer and the services to get access to the functionality. Table 2 lists the available FAL services. See clause 7.7 in IEC 61158-6-10 for details.

**Table 2 – PROFINET FAL services**

| FAL service | req | ind | rsp | cnf | Definition |
|---|---|---|---|---|---|
| Alarm | x | – | x | – | Alarm handling Interface (e.g. Process alarm) |
| Application Ready | x | – | x | – | IO device signals if it is ready for operate |
| Connect | – | x | x | – | Shows a connect request |
| Prm End | – | x | x | – | Shows the End of module/ submodule parametrization |
| Prm Begin | – | x | x | – | Shows the Start of module/ submodule parametrization |
| Read | – | x | x | – | Signals a record read functionality |
| Write | – | x | x | – | Signals a record write functionality |
| Local Add Diagnosis Entry | x | – | – | x | Add diagnosis information (e.g. ExtChannel diagnosis) |
| Local Remove Diagnosis Entry | x | – | – | x | Remove diagnosis information (e.g. ExtChannel diagnosis) |
| Add Submodule | x | – | – | x | Add new submodule |
| Remove Submodule | x | – | – | x | Remove submodule |
| Local Set Input | x | – | – | x | Set Input data of the Submodule |
| Local Get Output | x | – | – | x | Get Output data and IOPS of the submodule |
| Local New Output | – | x | – | – | Signals if new Output data are available |

### 5.4.2 GWA services

Table 3 shows the relevant gateway applications of IO-Link. See clause 11.1.2 in [1] for details.

**Table 3 – Gateway application services of IO-Link**

| Service | Description |
|---|---|
| OperatingMode | This variable activates the port and provides the configuration parameters |
| ReadyToOperate | This variable indicates correct configuration of the port |
| StartOperate | This variable allows for explicit change of all ports to the OPERATE mode |
| Operating | This variable indicates all ports are in cyclic Process Data exchange mode |
| Fault | This variable indicates abandoned COMx |
| AL_Read | The AL_Read service is used to read On-request Data from a Device |
| AL_Write | The AL_Write service is used to write On-request Data to a Device |
| AL_Event | Indicates IO-Link Events |
| AL_Get_Input | Read Input data of a port (Device) |
| AL_NewInput | New Input data are available |
| AL_Set_Output | Set Output data of a port |
| AL_Control | Indicates the validity of input data via (indication)<br>• VALID (input Process Data valid) |

| Service | Description |
|---|---|
| | • INVALID (input Process Data invalid)<br>Indicates the validity of output signals (request)<br>• PDOUTVALID (output Process Data valid)<br>• PDOUTINVALID (output Process Data invalid) |

# 6 Slot model of the "Linking Module"

## 6.1 Concept

The "Linking Module" concept encapsulates the entire IO-Link data objects and functions of an IO-Link Master system into one PROFINET slot.

As a consequence, this means for the top level concept:

- The entire IO-Link Master system shall be mapped into one slot k with k = 1 to 32767.
- The IO-Link Master itself shall be mapped into an appropriate IOLM proxy submodule in subslot x of slot k.
- The IO-Link Devices shall be mapped each into one IOLD proxy submodule in subsequent subslots x+1, x+2 to x+n.

Figure 4 demonstrates the mapping of IO-Link system data objects and functions into PROFINET slots and subslots.

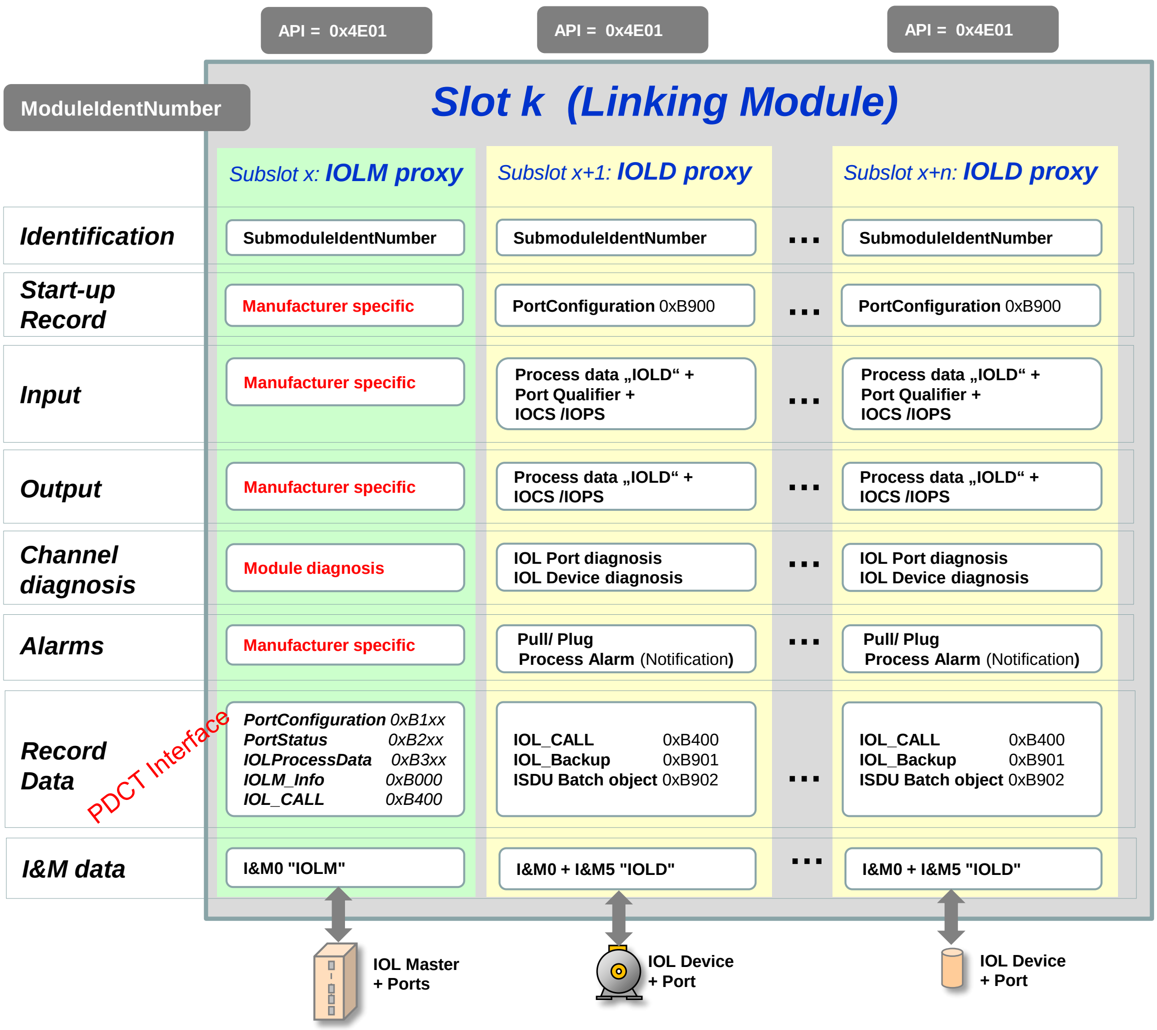



**Figure 4 – Mapping of IO-Link data objects and functions into slots and subslots**

The following rules define the detailed concept of "Linking Modules":

- Each submodule shall be assigned to API "0x4E01", which means IO-Link.
- It is manufacturer's/vendor's responsibility to assign subslots. However, it is recommended to start with subslot 1 for the IOLM proxy followed by *n* subslots representing the IO-Link ports
- The IOLM proxy represents the Master functionality and the access point of the PDCT interface
- The IOLD submodule represents first of all the proxy for the Device. Then, the functionality of the IO-Link Port (e.g. Port configuration) shall be mapped to the IOLD proxy submodule
- Items depicted in black colored text are mandatory. The functionality shall be implemented as specified in this document.
- Items depicted in red colored text are not mandatory. They are not part of this document. Manufacturers/vendors are free to design and implement according to their requirements.

Clause 7 provides more details of the individual PROFINET aspects such as identification, input/output, channel diagnosis, alarms, start-up, and I&M.

The following two clauses provide an idea of the roles and responsibilities of the proxies.

## 6.2 IOLM proxy

An IOLM proxy submodule is mandatory. It provides an interface to the user for:

- I&M data
- Master diagnosis (port unspecific)
- Parametrization of system behavior (e.g. enable diagnosis for the Linking Module)
- IO-Link Tools (PDCT access point)
- Manufacturer specific functions
- Additional port functions such as Pin2 behavior at port class A or power OFF/ON at Pin2/5 at port class B are described in Annex A.

## 6.3 IOLD proxy

### 6.3.1 General

The IOLD proxy submodule provides an interface to a particular port and to its connected Device. A number of data objects and functions are standardized in this document as shown in Figure 4 (black colored text):

- Process data mapping (Input and Output data) and IOPS/IOCS handling (see 7.5)
- Port and Device diagnosis mapping (Channel diagnosis, see 7.6)
- Pull/ Plug behavior of Devices
- Process alarm mapping according to IO-Link notification events (see 7.7)
- Access to Device objects (ISDU) via PLC function blocks (IOL_CALL, see 10)
- I&M0 data (see 7.8.3)
- Backup & Restore (see 9.1)

### 6.3.2 Exceptional use cases

In practice, exceptional use cases can interrupt the normal cyclic operation of Devices connected to a Master. These use cases are:

- Device is *not available* (no IO-Link communication possible), and
- Device is *available* (communication established). However, it got stuck in PREOPERATE state due to misfit of parameters or incorrect configuration.

In cases, where the Device is *not available* due to missing communication, the causes may be:

- Device not connected, or misconnected, or incorrect wiring
- Device has no power (supply voltage error)

As a consequence, the Linking Module will remove the IOLD proxy submodule resulting in:

- No I&M0 data available
- No access to Device possible

In cases, where the Device is communicating, but got stuck in PREOPERATE state, the causes may be:

- IO-Link validation fault (incorrect Device)
- Several port problems according to EventCode of "Master Local"

These cases can result in:

- IOPS remains "good"
- I&M0 data are still available (identification possible)
- Appropriate diagnosis event (e.g. incorrect Device)
- Access to Device possible
- Port Qualifier information will show the flags PQ = "Bad" and DevErr = "Error"
- Master sends SystemCommand "PDoutinvalid"

# 7 Mapping Application

## 7.1 General

The concept of the "Linking Module" has already been introduced in clause 6 and the associated Figure 4 demonstrates the mapping of IO-Link system data objects and functions into PROFINET slots and subslots as well as general aspects related to the Linking Module.

This clause provides details of the API, the slot/subslots and the individual PROFINET aspects such as Identification, Start-up, Input/Output, Channel diagnosis, Alarms, and I&M.

## 7.2 API (application process identifier instance)

In order to avoid competing accesses of user profiles, PROFINET uses the API (Application Process Identifier Instance) as an additional addressing level.

Therefore, all proxy submodules for IO-Link are associated with the API "0x4E01"(19969).

## 7.3 Identification (PROFINET)

PROFINET provides a comprehensive identification concept covering all IO-Link identification aspects. The following rules describe the usage of PROFINET identification means with respect to a Linking Module:

- *VendorID* shall be used to identify the manufacturer/vendor of the PROFINET IO device including the Linking Module (PROFINET IO device manufacturer specific)
- *DeviceID* shall be used to identify the type of the PROFINET device including the Linking Module (PROFINET IO device manufacturer specific)
- *ModuleIdentNumber* shall be used to identify the type of the Linking Module (PROFINET IO device manufacturer specific)

- *SubmoduleIdentNumber* shall be used to identify the type of the IOLM proxy or IOLD proxy with identical characteristics of IO data, diagnosis and start-up parameters within the range of a ModuleIdentNumber.

The SubmoduleIdentNumber of an IOLM proxy will be determined by the Linking Module manufacturer/vendor. Table 4 shows the coding for generic and DI/DO Devices. Coding "0" in all Octets 0 to 3 at once is not permitted.

**Table 4 – Coding of SubmoduleIdentNumber of IOLD_proxy (generic)**

| SubmoduleIdentNumber | Coding | IOLD proxy submodule |
|---|---|---|
| Octet 2,3 | 0x0000 | Device libraries in Tools shall display separate folders:<br>❖ IO-Link generic Devices<br>❖ Digital Input (DI), Digital Output (DO) |
| Octet 1 | 0x00 to 0x20 | Output length of IOLD proxy submodule (0 to 32 octets) |
| | 0x81 | Digital Output (DO) |
| | All other | Reserved |
| Octet 0 | 0x00 to 0x21 | Input length of IOLD proxy submodule (0 to 33 octets) |
| | 0x81 | Digital Input (DI) |
| | All other | Reserved |

Table 5 shows the coding for profile Devices.

**Table 5 – Coding of SubmoduleIdentNumber of IOLD_proxy (profiles)**

| SubmoduleIdentNumber | Coding | IOLD proxy submodule |
|---|---|---|
| Octet 2,3<br>(ProfileIdentifier PFID; see Annex B.2.5 in [1]) | 0x0001 to 0x3FFF | Device libraries in Tools shall display in a separate folder:<br>❖ IO-Link profile Devices |
| Octet 1 | 0x00 to 0x20 | Output length of IOLD proxy submodule (0 to 32 octets) |
| | All other | Reserved |
| Octet 0 | 0x00 to 0x21 | Input length of IOLD proxy submodule (0 to 33 octets) |
| | All other | Reserved |

Table 6 shows the coding for vendor specific Devices.

**Table 6 – Coding of SubmoduleIdentNumber of IOLD_proxy (vendor)**

| SubmoduleIdentNumber | Coding | IOLD proxy submodule |
|---|---|---|
| Octet 2,3 | 0x4000 to 0x4FFF | Vendor specific usage of SubmoduleIdentNumber |
| Octet 1 | 0x00 to 0x20 | Output length of IOLD proxy submodule (0 to 32 octets) |
| | All other | Reserved |
| Octet 0 | 0x00 to 0x21 | Input length of IOLD proxy submodule (0 to 33 octets) |
| | All other | Reserved |

Table 7 shows the coding for functional safety Devices.

**Table 7 – Coding of SubmoduleIdentNumber of IOLD_proxy (PROFIsafe)**

| SubmoduleIdentNumber | Coding | IOLD proxy submodule |
|---|---|---|
| Octet 2,3 | 0x5000 to | Vendor specific usage of SubmoduleIdentNumber |

| SubmoduleIdentNumber | Coding | IOLD proxy submodule |
|---|---|---|
| | 0x5FFF | |
| Octet 1 | 0x00 to 0x20 | Output length of IOLD proxy submodule (0 to 32 octets) |
| | All other | Reserved |
| Octet 0 | 0x00 to 0x21 | Input length of IOLD proxy submodule (0 to 33 octets) |
| | All other | Reserved |

Table 8 demonstrates examples of SubmoduleIdentNumbers.

**Table 8 – Examples of SubmoduleIdentNumbers**

| SubmoduleIdentNumber | Description |
|---|---|
| 0x0000 2021 | SubmoduleIdentNumber of "IO-Link 32I / 32O + PQI" (Supports 33 octets Input and 32 octets Output) NOTE |
| 0x0000 1011 | SubmoduleIdentNumber of "IO-Link 16I / 16O + PQI" (Supports 17 octets Input and 16 octets Output) NOTE |
| 0x0000 8100 | SubmoduleIdentNumber of "Digital Output" |
| 0x0000 0081 | SubmoduleIdentNumber of "Digital Input" |
| 0x0002 0800 | SubmoduleIdentNumber of Fixed Switching Sensor (FSS) profile Device |
| NOTE The difference in length stems from the additional PQI octet | |

## 7.4 Startup record

### 7.4.1 Overview

In PROFINET it is possible to define startup records for submodules (IOLD proxy and IOLM proxy) within the GSD file of the IO device. The user can assign actual values to those records during the engineering phase of the IO device.

The IO controller transmits these startup records to the IO device in the period between the "connect response" and the "end of parameter" requests. The following rules apply for startup records:

- Startup parameters of an IO device shall be part of the corresponding submodule within the GSD file of the IO device (see template GSD)
- Possible values and data types shall comply with the GSD standard

### 7.4.2 Port configuration record (Index 0xB900)

This record describes the port configuration from an IO-Link point of view. Each IOLD proxy shall support this record, which is volatile, readable, and writeable. It contains the expected port configuration for subslot Index 0xB900. Table 9 shows the specified structure.

**Table 9 – Port configuration record**

| Offset | Parameter name | Definition | Data type |
|---|---|---|---|
| 0 | BlockVersionHigh | Versioning of record; first version: 0x01 | Unsigned8 |
| 1 | BlockVersionLow | Versioning of record; first version: 0x00 | Unsigned8 |
| 2 | Reserved | – | Unsigned16 |
| 4 | PortConfigControl | Bit 0: Enable Port Diagnosis<br>Bit 1: Enable Process Alarm (Device notification)<br>Bit 2,3,4: Port Mode:<br>0: IOL-Autoconfig<br>1: IOL-Manual<br>2: IOL-Tool based | Unsigned8 |

| Offset | Parameter name | Definition | Data type |
|---|---|---|---|
| | | 3: Digital Input (Pin 4)<br>4: Digital Output (Pin 4)<br>5 to 7: Reserved<br>Bit 5: reserved<br>Bit 6: Enable Input fraction<br>Bit 7: Enable Pull/Plug | |
| 5 | Validation & Backup | 0: no Device check<br>1: type compatible Device (V1.0)<br>2: type compatible Device (V1.1)<br>3: type compatible Device (V1.1) with Backup + Restore<br>4: type compatible Device (V1.1) with Restore<br>5 to 255: reserved | Unsigned8 |
| 6 | VendorID | Expected IO-Link Device VendorID | Unsigned16 |
| 8 | DeviceID | Expected IO-Link DeviceID | Unsigned32 |
| 12 | PortCycleTime | 0: as fast as possible<br>16: 1,6 ms<br>32: 3,2 ms<br>48: 4,8 ms<br>68: 8,0 ms<br>100: 20,8 ms<br>133: 40,0 ms<br>158: 80,0 ms<br>183: 120,0 ms<br>Coding is derived from [1]. User can provide a selection of codings (see example) | Unsigned8 |
| 13 | Reserved | – | Unsigned8 |
| 14 | Reserved | – | Unsigned8 |
| NOTE1 Data types comply with IEC 61158-6-10<br>NOTE2 InputLength: Input data length of Submodule | | | |

User can select only a part of the Device's input data for the mapping to PROFINET via the parameter "Enable Input fraction". The mapping procedure ensures inclusion of PQI in any case. As a consequence, only "0" up to (InputLength – 1) octets of the Input data of the Device can be mapped.

## 7.5 Process Data (IO data)

### 7.5.1 General rules for IOLD proxy mapping

The input and/or output data of the Device (Process Data – PD) at a particular port are inserted in the input and/or output data of the dedicated IOLD proxy submodule. In principle, the IO-Link input/output data are cyclically "copied" into the PROFINET data "container" of the submodule whenever the port is running. The following general rules apply:

- The input PD of the Device starting with offset "0" will be mapped into the input data of the IOLD proxy starting with offset "0"
- The output data of IOLD proxy will be mapped into the output PD of Device starting with offset "0"
- The input data length of the IO-Link proxy submodule shall be greater or equal than the input Process Data length +1 of the Device.
- The output data length of the IO-Link proxy submodule shall be greater or equal than the output Process Data length of the Device.
- PQI will always be mapped into the highest octet of the IOLD proxy input data except in cases, where the data length is not sufficient. The Port Qualifier Information – PQI contains important status information on port and Device status.

- If "Enable Input fraction = 0" then the input data length of the IO-Link proxy submodule shall be greater or equal than the input Process Data length +1 of the Device. Otherwise the diagnosis "process data mismatch" will be created.
- If "Enable Input window = 1" then the Process data of the Device will be mapped to the IOLD proxy submodule up to the PQI information. That means the Input data of the Device could be partly mapped without errors. Only "0" up to (InputLength – 1) octets of the Input data of the Device will be mapped in the IOLD proxy submodule.

Figure 5 demonstrates the principles of the IO data mapping.

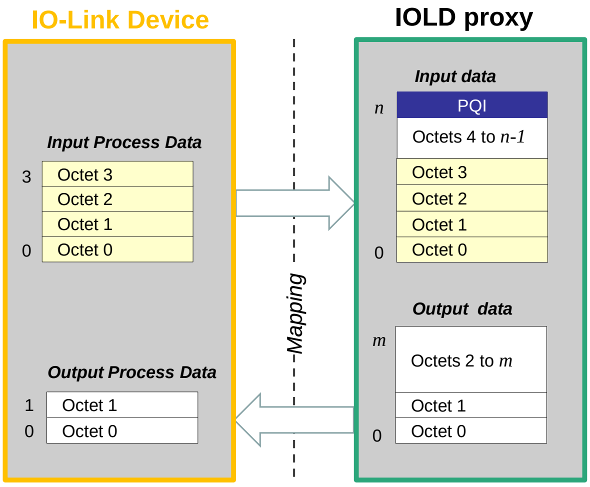



**Figure 5 – Principle of IO data mapping**

### 7.5.2 General rules for IOLM proxy mapping

The input and/or output data for additional port class A or B functions (see Annex A) are inserted in the input and/or output data of the dedicated IOLM proxy submodule.

The manufacturer/vendor of a Linking Module provides an IOLM proxy submodule adequate to the available functions of port class A or B.

### 7.5.3 IO data access services

Both PROFINET and IO-Link offer services for the access of IO data and control information.

*IO-Link Application Layer* (see Table 59 in [1] and GWA services in 5.4.2):

- "AL_GetInput" service shall be used to read the input Process Data of the Device
- "AL_SetOutput" service shall be used to write output Process Data to the Device
- "AL_Control" service shows the validity of input Process Data (PDValid, PDInvalid) and is used to propagate validity information for output Process Data (PDOUTVALID, PDOUTINVALID) to the Device

*Fieldbus Application Layer* (see FAL services in 5.4.1):

- "Local Set Input" service shall be used to write the input data of a slot/subslot. IOPS information can be added showing the validity of input data
- "Local Get Output" service shall be used to read output data from slot/sublot. In addition, IOPS will be readable via "Local Get Ouput" showing the validity of output data

### 7.5.4 IO data mapping (IOLD proxy)

The mapping of I/O data shall only be done in the regular case of a running port (see 7.4).

In other cases, the input data shall be marked as invalid. Invalid data shall be mapped to PQI (Port Qualifier Information, i.e. PQ shall be set to "bad"). The Device output Process Data shall be marked as PDOUTINVALID.

In a first step, the Process Data of the Device shall be mapped to the IOLD proxy and vice versa as shown in Figure 5 using the specified services. This mapping procedure shall be repeated cyclically to achieve a proper update rate (manufacturer specific).

In a second step, the I/O data validity handling shall be performed showing the application whether the I/O data are valid and can be processed.

The second step comprises in detail:

- "PD(IN)VALID" in the AL_Control service shall be mapped to PQI, i.e. PQ flag shall be set to "0"("bad") using the "Local Set Input" service.
  NOTE No IOPS handling takes place
- IOPS of IOLD proxy output data (detected via "Local Get Output" service) shall be mapped to "PDOUT(IN)VALID" in the AL_Control service indicating to the Device that output data are valid/invalid.

Figure 6 demonstrates the entire input mapping activity including the PQI handling. The input Process Data of the Devices are cyclically copied into the input data of the IOLD proxy. The PQ flag is set depending on PDInvalid or any detected port fault. In addition, all PQI flags are updated.

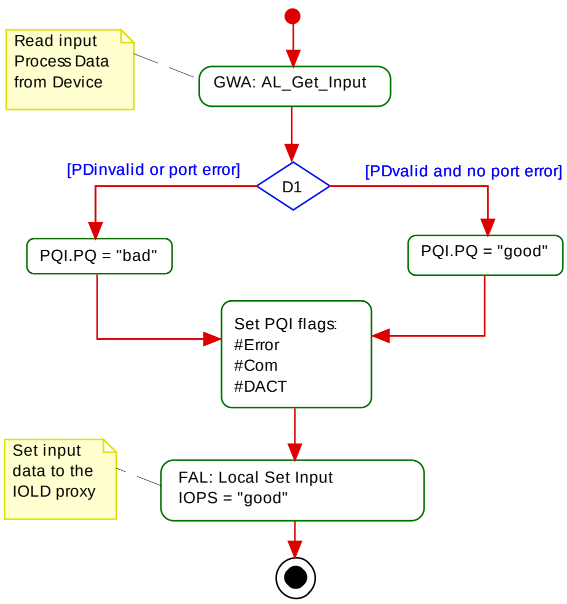



**Figure 6 – Input PQI handling**

Figure 7 demonstrates the entire output mapping activity including the IOPS handling. The output data of the IOLD proxy are cyclically copied into the output Process Data of the Device. Any change of the IOPS information causes an AL_Control service with the appropriate PDINVALID/PDVALID values.

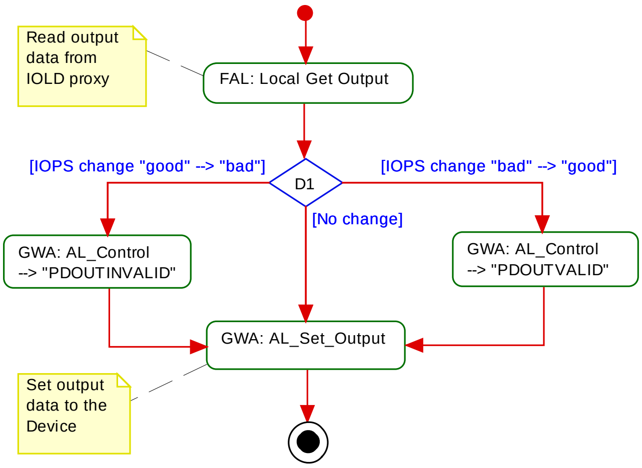



**Figure 7 – Output IOPS handling**

#### 7.5.4.1 Port qualifier information (PQI)

The "Port qualifier information" provides status information showing the status of the IO-Link port or the status of the Device respectively.

Figure 8 shows the layout of the PQI flag bits.

| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|
| Device Process Data validity | Port/Device error indication | Device communication | Port activation | Substitute Device detection | New parameter | Reserved ("0") | Reserved ("0") |
| PQ | DevErr | DevCom | PortActive | SubstDev | NewPar | – | – |

**Figure 8 – Port qualifier information (PQI)**

Table 10 shows the definitions of flag bits in Figure 8.

**Table 10 – Definition of flag bits in Figure 8**

| Flag | M/O | Value | Definition |
|---|---|---|---|
| NewPar | M | 0 | No update of Device parameter detected |
| | | 1 | Update of Device parameter detected: Master performed a Data Storage upload and a new IOLD Backup object (0xB904) is available (see 9.1) |
| SubstDev | M | 0 | No substitute Device detected (identical SerialNumber) |
| | | 1 | Substitute device detected (different SerialNumber) |
| PortActive | M | 0 | Port de-activated via port function |
| | | 1 | Port activated (default value, for details see 9.2.2) |
| DevCom | M | 0 | No Device available |
| | | 1 | Device detected and is in PREOPERATE or OPERATE state |
| DevErr | M | 0 | No error/warning occurred |
| | | 1 | Error/warning assigned to Device or Port occurred |
| PQ | M | 0 | Invalid IO Process Data from Device |
| | | 1 | Valid IO Process Data from Device |

#### 7.5.4.2 IO Update Rate

The IO update is performed in a cyclic manner. The rate is called "IO Update Rate" and is determined by the PortCycleTime, within which the IO data of the port (Device) are read or written.

It is highly recommended to perform the IO data update of the proxy submodule (Local Set Input) right after the read of the input data (AL_Get_Input) within the same port cycle. The PQI information shall be updated as well.

It is also highly recommended to write the output data of the IOLD proxy (Local Get Output) to the Device within one port cycle (AL_Set_Output).

## 7.6 Diagnosis

### 7.6.1 "Tunneling" concept

Devices and Master can create "Events" of type error, warning, or notification. Diagnosis means here the mapping of each relevant IO-Link Event to the PROFINET diagnosis mechanism. This document shows rules for this mapping based on the "Linking module model". The PROFINET diagnosis model is specified in [7].

Each client (PLC, etc) can get access to the entire IO-Link diagnosis information through the "Standard" PROFINET mechanism without IO-Link specific extensions. From a user's point of view it looks like "tunneling" of the IO-Link diagnosis information.

### 7.6.2 Base model

#### 7.6.2.1 Concerned "parties"

Figure 9 illustrates the base model. The upper part is defined in the PROFINET standard. The lower part shows the Linking Module with the task to achieve an integrated diagnosis view.

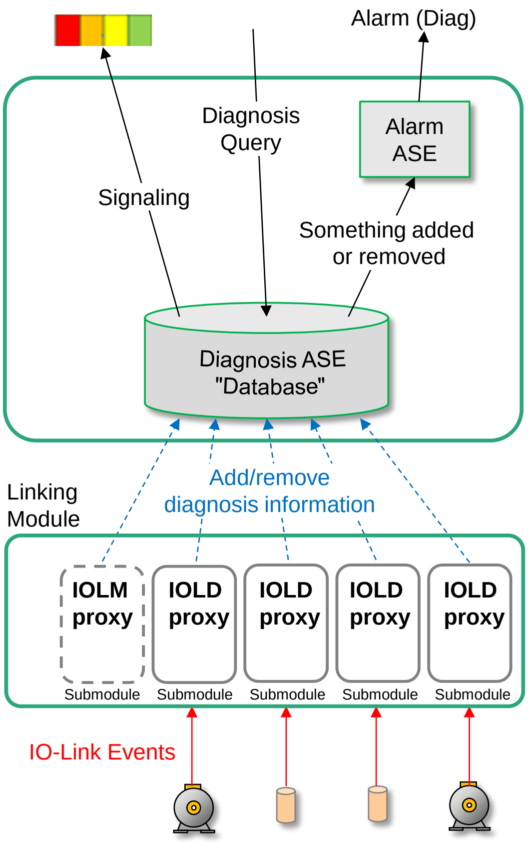



**Figure 9 – Base diagnosis model**

Center part is the Diagnosis ASE "Database" holding the current diagnosis information of each present submodule (IOLD proxy). The Diagnosis ASE uses the Alarm ASE to alert the IO controller and to inform it about pending incidents. It provides signaling of diagnosis levels for example for condition monitoring or predictive maintenance. Any external client can get access to this "Database" with filtering capabilities for the queries.

The Linking Module adds or removes diagnosis information upon IO-Link Event detection. Each IO-Link Event causes an adding or removing action. Details and concrete mapping is subject of the following diagnosis clauses.

### 7.6.2.2 Diagnosis ASE

#### 7.6.2.2.1 Diagnosis source

The Diagnosis ASE of an IO device contains its diagnosis information, which is arranged by the source of a diagnosis. From a Linking Module point of view one *diagnosis source (= key attribute)* per port is permitted, clearly represented by

- API
- Slot (the slot number of the diagnosis source/Linking module )
- Subslot (the subslot number of the IOLD proxy submodule)
- ChannelNumber (the channel number of the diagnosis source)
  0x8000: the entire submodule is the source
  0 to 0x7FFF: the source is a channel as specified by the manufacturer
- Direction (IN, OUT, IN/OUT or manufacturer specific)

The diagnosis source is the information where the diagnosis is located in the IO device. Each diagnosis source exists only once in the Diagnosis ASE if a diagnosis exists for this particular source. In other words, the tuple (API, Slot, Subslot, ChannelNumber, Direction, and Accumulative) is the key attribute.

#### 7.6.2.2.2 Diagnosis information

A diagnosis source holds one to many diagnosis informations. The number of diagnosis informations is defined by the number of appearing IO-Link Events per Device.

In case of IO-Link a standard format is used (see [1]):

- ChannelErrorType (expresses the type of diagnosis)
- ExtChannelErrorType (expresses the subtype of diagnosis)
- Priority (expresses the urgency of a maintenance demand)
- ExtChannelAddValue (additional information to the diagnosis acquired at the moment the incident appeared)

Each diagnosis information in standard format exists only once per key attribute in the Diagnosis ASE. If priority changes, the diagnosis information is overwritten.

### 7.6.2.3 Adding and removing diagnosis

The Mapping Application in the Linking Module adds or removes diagnosis information upon IO-Link Event detection. Thus, an appearing IO-Link Event causes diagnosis information to be added and a disappearing IO-Link Event causes diagnosis information to be removed.

Diagnosis information is added to the Diagnosis ASE in case it is new diagnosis information of this source (IO-Link Event). Whether the diagnosis information is new or not is derived from the diagnosis information’s key attribute.

The diagnosis information is updated in the Diagnosis ASE any time the existing diagnosis information of a known source changed its Priority.

### 7.6.3 IO-Link Events

Each IO-Link Event entry consists of an EventQualifier and an EventCode.

The structure of an EventQualifier is shown in Figure 10. It specifies "MODE", "TYPE", "SOURCE", and "INSTANCE" of an Event (see A.6.4 in [1]).

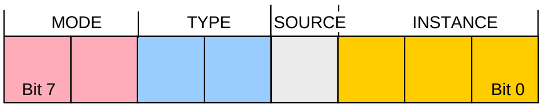



**Figure 10 – Structure of the EventQualifier**

Table 11 shows the possible values for EventQualifiers.

**Table 11 – Possible values for EventQualifier**

| EventQualifier | Possible values |
|---|---|
| MODE | 1 = Event single shot<br>2 = Event disappears<br>3 = Event appears |
| TYPE | 1 = Notification<br>2 = Warning<br>3 = Error |
| SOURCE | 0 = Device (remote)<br>1 = Master (local) |
| INSTANCE | 0 = Unknown<br>4 = Application |

The EventCode entry contains the identifier of an actual Event. Permissible values for EventCodes are listed in Annex D of [1]. They shall be assigned according to the following rules for "V1.1" Devices:

- Events of TYPE "Error" shall use the MODEs "Event appears" / "Event disappears"
- Events of TYPE "Warning" shall use the MODEs "Event appears" / "Event disappears"
- Events of TYPE "Notification" shall use the MODE "Event single shot"
- Each vendor specific EventCode shall be uniquely assigned to one of the TYPEs
- Each Event shall use static MODE, TYPE, and INSTANCE attributes

### 7.6.4 Diagnosis mapping

#### 7.6.4.1 General

Diagnosis mapping uses basic rules how to map IO-Link Events to the appropriate PROFINET diagnosis. Figure 11 shows the actions taking place when an error appears and disappeares.

IO-Link Event information is indicated by an AL_Event service (.ind) of the assigned port x and is mapped to PROFINET diagnosis using FAL services ("local add diagnosis entry" or "lo-cal remove diagnosis entry").

The Event dispatcher is responsible to map the AL_Event in a correct manner and to ensure flow control while directing the confirmation (.cnf) of the diagnosis entry service to the appro-priate AL_Event response (.rsp).

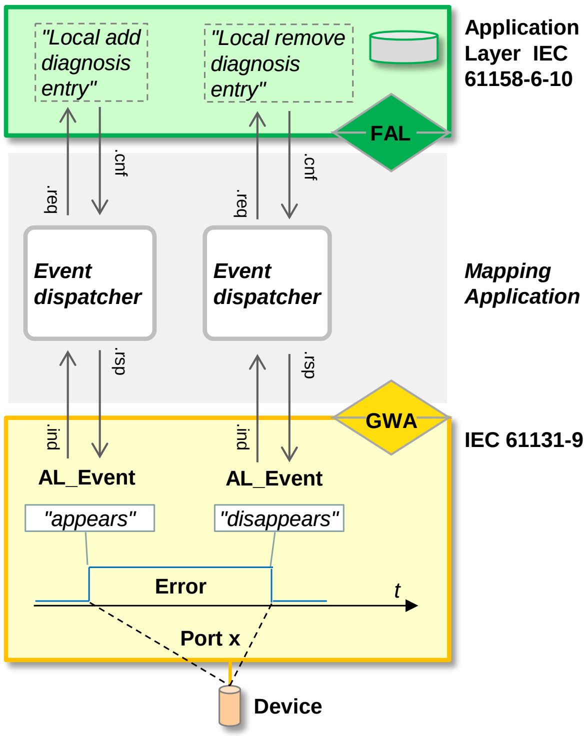



**Figure 11 – Diagnosis mapping**

### 7.6.4.2 Event dispatcher

Figure 12 shows the tasks of the Event dispatcher during the mapping process. It is responsible to map the AL_Event representing port, EventQualifier and EventCode to the appropriate diagnosis source.

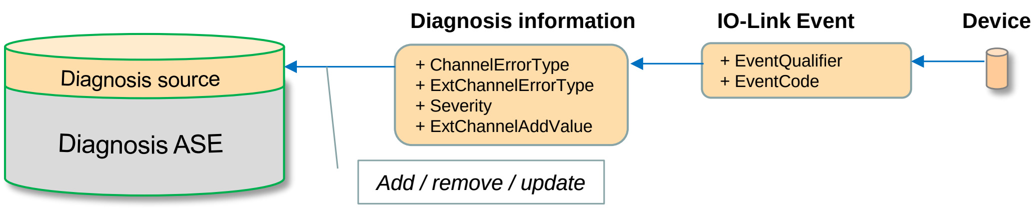



**Figure 12 – Event dispatcher tasks**

Detailed responsibilities:

- AL_Event assigned to Port number shall be assigned to correct diagnosis source
- EventCode mapping (7.6.4.3)
- Priority mapping (7.6.4.4)
- Disappearing events (7.6.4.5)

Diagnosis source:

For the "Linking module", each source will be defined by

- API: Fixed 0x4E01
- Slot: Linking Module
- Subslot: IO-Link Device/Master proxy (IOLD proxy/IOLM proxy)
- Direction: "0" (manufacturer/vendor specific)
- ChannelNumber: Vendor specific (e.g. 0x8000)

In a previous edition of this document it was recommended to use ChannelNumber 0x8000 (entire submodule) for an IOLD proxy. This Edition permits to use specific channel numbers, for example Port Number = channel number.

### 7.6.4.3 EventCode mapping

#### 7.6.4.3.1 Fixed EventCode assignments

Each IO-Link Event represented by EventQualifier and EventCode shall be mapped in the defined manner to diagnosis information represented by ChannelErrorType, ExtChannelErrorType, and ExtChannelAddValue as shown in Table 12.

**Table 12 – Fixed EventCode assignments**

| EventQualifier | Event-Code | ChannelError Type | ExtChannel ErrorType | ExtChannel-AddValue | Comment |
|---|---|---|---|---|---|
| INSTANCE: Application or unknown<br>SOURCE: Device | 0x0000 to 0x7FFF | 0x9500 | 0x0000 to 0x7FFF | Set to zero 0x0000 0000 | IOLD Events:<br>Direct mapping of EventCode to ExtChannelErrorType<br>Example: EventCode 0x6321 mapped to ExtChannelError-Type 0x6321 |
| INSTANCE: Application or unknown<br>SOURCE: Device | 0x8000 to 0xFFFF | 0x9501 | 0x0000 to 0x7FFF | Set to zero 0x0000 0000 | IOLD Events:<br>Mapping of EventCode to ExtChannelErrorType. Set MSB (EventCode) to “0”.<br>Example: EventCode 0x8005 mapped to ExtChannelError-Type 0x0005 |
| INSTANCE: Application or unknown<br>SOURCE: Master (local) | 0x0000 to 0x7FFF | 0x9502 | 0x0000 to 0x7FFF | Set to zero 0x0000 0000 | IOL port Events:<br>Direct mapping of local EventCode to ExtChannel-ErrorType |

#### 7.6.4.3.2 EventCode "Device"

EventCode identifiers and their definitions are defined in Annex D1 of [1]. The EventCodes are created by the technology specific Device application (instance = APP, SOURCE= Device).

#### 7.6.4.3.3 EventCode "Master local"

EventCode generated by the Master, or port respectively, is not standardized in [1]. In principle the coding is vendor specific with the exception of a few standardized Master (local) EventCodes.

Table 13 lists the EventQualifiers and EventCodes of Master or port incidents. This table expands the definitions of Annex D in [1].

**Table 13 – Master EventCode assignments**

| EventQualifier | EventCode | Description |
|---|---|---|
| INSTANCE: Application, unknown<br>SOURCE: Master (local) | 0x0000 to 0x17FF | Vendor specific |
| | 0x1800 | reserved |
| | 0x1801 | Startup parametrization error – check parameter |
| | 0x1802 | Incorrect Device – Inspection Level mismatch |
| | 0x1803 | Process Data mismatch – check submodule configuration |
| | 0x1804 | Short circuit at C/Q – check wire connection |
| | 0x1805 | IO-Link PHY overtemperature |
| | 0x1806 | Short circuit at L+ – check wire connection |

| EventQualifier | EventCode | Description |
|---|---|---|
| | 0x1807 | Undervoltage at L+ – check power supply (e.g. L1+) |
| | 0x1808 | Device Event overflow |
| | 0x1809 | Backup inconsistency – memory out of range (2048 octets) |
| | 0x180A | Backup inconsistency – Data storage index not available |
| | 0x180B | Backup inconsistency – Data storage unspecific error |
| | 0x180C | Backup inconsistency – upload fault |
| | 0x180D | Parameter inconsistency – download fault |
| | 0x180E | P24 (Class B) missing or undervoltage |
| | 0x180F | Short circuit at P24 (Class B) – check wire connection (e.g. L2+) |
| | 0x1810 to 0x5FFF | Vendor specific |
| | 0x6000 | Invalid cycle time |
| | 0x6001 | Revision fault – incompatible protocol version |
| | 0x6002 | Parameter inconsistency? – ISDU batch failed |
| | 0x6003 to 0x7F20 | Reserved |
| Table D.2 in [1] | 0x7F21 | Reserved |
| | 0x7F22 | Device not available – communication lost |
| | 0x7F23 | Invalid backup – Data Storage identification mismatch |
| | 0x7F24 | Invalid backup – Data Storage buffer overflow |
| | 0x7F25 | Invalid backup – Data Storage parameter access denied |
| | 0x7F26 to 0x7F30 | Reserved |
| | 0x7F31 | Event lost – incorrect Event signalling |
| | 0x7F32 to 0x7FFF | Reserved |

#### 7.6.4.4 PROFINET priority mapping

Priority expresses how urgent maintenance is demanded to cure a specific diagnosis:

- "Fault" requires immediate action; the channel is no longer working
- "Maintenance demanded" requires maintenance as soon as possible
- "Maintenance required" requires maintenance at the next opportunity
- "Good" means normal operation; this is indicated at absence of any of the above entries

From the Linking Module point of view, two priority levels are supported:

- TYPE "Error" of the EventQualifier shall be mapped to priority "Fault"
- TYPE "Warning" of EventQualifier shall be mapped to priority "Maintenance demanded"

#### 7.6.4.5 Appearing and disappearing Events

Each appearing Event (TYPE error or warning) shall result in "add diagnosis information" (Local add diagnosis entry).

Each disappearing Event (TYPE error or warning) shall result in "remove diagnosis information" (Local remove diagnosis entry).

### 7.6.5 Restrictions and characteristics

Events of TYPE = Error or warning shall be mapped to PROFINET diagnosis only in case of MODE "Event appears" and "Event disappears".

Events of MODE "Single shot" shall not be mapped to diagnosis. This Event shall rather be mapped to Process Alarm (see 7.7).

### 7.6.6 GSD and diagnosis

It should be noted that only particular ranges of ErrorCodes are assigned to defined incidents. Two ranges are defined "Vendor specific": EventCodes from 0x1800 to 0x18FF and 0x8CA0 to 0x8DFF. In general, EventCodes are defined in Annex D of [1].

From an IO-Link point of view it is recommended to use primarily these standardized EventCodes. The mapping solution shown herein supports the mapping of the standardized diagnosis information using the generic description. The template GSD supports all standardized errors and texts (see 11.3.2).

It is not possible to convey "Vendor specific" diagnosis.

Table 14 contains a list of diagnosis descriptions to illustrate the scope and types of error informations. The list does not claim to be complete.

**Table 14 – Predefined PROFINET Channel Errors**

| Error Type | Extended Error Type | API | Error text |
|---|---|---|---|
| 16 | – | | Parameter assignment error |
| 17 | – | | Missing 1L+ or 2L+ |
| 18 | – | | Fuse defective |
| 19 | – | | Slot not addressable |
| 20 | – | | Ground error |
| 21 | – | | Reference channel error |
| 22 | – | | Process interrupt lost |
| 23 | – | | Warning |
| 24 | – | | Shutoff |
| 25 | – | | Fail-safe shutoff |
| 26 | – | | External error |
| 27 | – | | Ambiguous error |
| 29 | – | | Actuator/sensor fault 1 |
| 31 | – | | Channel temporarily unavailable |
| 38144 | – | 19969 | IO-Link Device diagnosis – Page 1 |
| 38144 | 4096 | | General malfunction – Unknown error |
| 38144 | 16384 | | Temperature fault – Overload |
| 38144 | 16912 | | Device temperature over-run – Clear source of heat |
| 38144 | 16928 | | Device temperature under-run – Insulate Device |
| 38144 | 20480 | | Device hardware fault – Device exchange |
| 38144 | 20496 | | Component malfunction – Repair to exchange |
| 38144 | 20497 | | Non volatile memory loss – Check batteries |
| 38144 | 20498 | | Batteries low – Exchange batteries |
| 38144 | 20736 | | Generalpower supply fault – Check availability |
| 38144 | 20737 | | Fuse blown/open – Exchange fuse |
| 38144 | 20752 | | Primary supply voltage over-run – Check tolerance |
| 38144 | 20753 | | Primary supply voltage under-run – Check tolerance |
| 38144 | 20754 | | Extra supply voltage fault (port class B) – Check tolerance |
| 38144 | 24576 | | Device software fault – Check firmware revision |

| Error Type | Extended Error Type | API | Error text |
|---|---|---|---|
| 38144 | 25371 | | Parameter error – Check data sheet and values |
| 38144 | 25372 | | Parameter missing – Check data sheet |
| 38144 | 25424 | | Parameter changed – Check configuration |

## 7.7 Alarms (Process Alarm)

Normally, IO-Link Events (TYPE "Warning" and "Error") are mapped to diagnosis as shown in 7.6.3. A special case is TYPE "Notification" due to its "Single shot" character without acknowledgment.

Events of TYPE "Notification" are not subject of diagnosis aspects of Devices. Therefore, this Event shall be mapped to PROFINET Alarm type "Process alarm".

Events of TYPE "Error" or "Warning" and MODE "Event single shot" shall always be mapped to "Process alarm".

The Event dispatcher is responsible to bypass these Events to Process alarms using the FAL service "Alarm" (AlarmType, API, Slot number, Subslot number, USI, ChannelNumber, PRAL_ChannelProperties, PRAL_Reason, PRAL_ExtReason, PRAL_ReasonAddValue):

- USI = 0x8320 (ProcessAlarmReason).
- ChannelNumber = 0x8000 (entire Submodule)
- PRAL_ChannelProperties.Accumulative = 0x00 (Single)
- PRAL_ChannelProperties.Direction = 0x00 (manufacturer/vendor specific)

Coding of PRAL_Reason, PRAL_ExtReason, and PRALReasonAddValue is shown in Table 15.

**Table 15 – Coding of PRALxxx**

| EventQualifier | Event-Code | PRAL_Reason | PRAL_ExtReason | PRALReasonAdd Value | Comment |
|---|---|---|---|---|---|
| TYPE : don´t care<br>MODE : Single shot<br>INSTANCE: Application or unkown<br>SOURCE: Device | 0x0000 to 0x7FFF | 0x9500 | 0x0000 to 0x7FFF | Set to zero 0x0000 0000 | IOLD Events: Direct mapping of EventCode to PRAL_ExtReason<br>Example: EventCode 0x6321 mapped to PRAL_ExtReason 0x6321 |
| TYPE : don´t care<br>MODE : Single shoot<br>INSTANCE: Application or unkown<br>SOURCE: Device | 0x8000 to 0xFFFF | 0x9501 | 0x0000 to 0x7FFF | Set to zero 0x0000 0000 | IOLD Events: Mapping of EventCode to PRAL_ExtReason Set MSB (EventCode) to “0”.<br>Example: EventCode 0x8005 mapped to PRAL_ExtReason 0x0005 |

## 7.8 I&M data

### 7.8.1 Overview

Identification & Maintenance (I&M) is an integral part of each PROFINET device implementation. It provides standardized information about a device and its parts (e.g. IO-Link Device proxies). I&M data are accessible through PROFINET Record Objects and are always bound to a submodule associated to the item to be described. An item means here the PROFINET device itself or a part of this device, for example a pluggable module of a modular device. Submodules can provide their own I&M data or share I&M data of other submodules.

I&M data can be addressed via Record read/ write service:

- I&M0 read via index 0xAFF0 of the IOLM proxy and IOLD proxy subslots (see 7.8.2 and 7.8.3)
- I&M1 to I&M4 read/write via 0xAFF1 to 0xAFF4 if supported (manufacturer specific)

In case of IO-Link represented by a "Linking Module", I&M data shall be supported as specified in 7.8.2 and 7.8.3.

### 7.8.2 I&M data of IOLM proxy

The IOLM proxy shall support I&M0 ("name plate" or "type plate") containing the base information of the IO-Link Master defined by the Linking Module manufacturer. That means I&M0 is mandatory.

Support of I&M1 to I&M*n* is manufacturer specific (optional) and is not specified in this document.

### 7.8.3 I&M data of IOLD proxy

#### 7.8.3.1 Overview

IOLD proxies represent the IO-Link Devices. Therefore, I&M0 data ("name plate" or "type plate") are necessary to identify the Device from a PROFINET point of view. Users are expecting at least similar quality of information when reading PROFINET device "type plates" and IOLD proxy or Device "type plates" respectively. Table 16 and Table 17 show the best compromise found for the mapping of both "worlds".

**Table 16 – I&M mapping for IOLD proxy submodules**

| I&M0 data | Octets | Data type | Mapping rules |
|---|---|---|---|
| VendorID | 2 | Unsigned16 | IO-Link Direct parameter page 1: VendorID. Direct mapping, for example "0x136". Exceptions: 1 → 93; 26 → 257; 87 → 467. |
| OrderID | 20 | Visible String | "Product Name" or "DeviceID". For details see [1] |
| IM_Serial_Number | 16 | Visible String | Insert SerialNumber of Device (IO-Link Index 21). If it is not available set to "Not accessible" |
| IM_Hardware_Revison | 2 | Unsigned8 | Set to 0x0000 (Default value) |
| IM_Software_Revison | 4 | Char,3 x Unsigned8 | Set to V0.0.0 (official release but not detectable) |
| IM_RevisionCounter | 2 | Unsigned16 | Set to "0" (0x0000) |
| IM_Profile_ID | 2 | Unsigned16 | IO-Link (API = 0x4E01) |
| IM_Profile_Specific_Type | 2 | Unsigned16 | Set to "0" (0x0000) |
| IM_Version | 2 | 2 x Unsigned8 | Octet 1 (MSB): set to 0x01<br>Octet 2 (LSB): set to 0x00 |
| IM_Supported | 2 | Unsigned16<br>(Bit Array) | Profile specific I&M: 0x0001<br>i.e. I&M1 to I&M15 not supported |

**Table 17 – I&M mapping for DI/DO proxy submodules**

| I&M0 data | Octets | Data type | Mapping rules |
|---|---|---|---|
| VendorID | 2 | Unsigned16 | Vendor ID of the IO-Link Master Manufacturer |
| OrderID | 20 | Visible String | "Digital Input" / "Digital Output" |
| IM_Serial_Number | 16 | Visible String | "Not accessible" |

| I&M0 data | Octets | Data type | Mapping rules |
|---|---|---|---|
| IM_Hardware_Revison | 2 | Unsigned8 | Set to 0x0000 (default value) |
| IM_Software_Revison | 4 | Char,3 x Unsigned8 | Set to V0.0.0 (official release but not detectable) |
| IM_RevisionCounter | 2 | Unsigned16 | Set to "0" (0x0000) |
| IM_Profile_ID | 2 | Unsigned16 | IO-Link (API = 0x4E01) |
| IM_Profile_Specific_Type | 2 | Unsigned16 | Set to "0" (0x0000) |
| IM_Version | 2 | 2 x Unsigned8 | Octet 1 (MSB): set to 0x01<br>Octet 2 (LSB): set to 0x00 |
| IM_Supported | 2 | Unsigned16<br>(Bit Array) | Profile specific I&M: 0x0001<br>i.e. I&M1 to I&M15 not supported |

### 7.8.3.2 OrderID mapping

If a Device supports Index 18 "Product Name" (0x0012) and this product name is shorter than 20 characters it shall be inserted in the "OrderID field". If the "Product Name" is shorter than or equal 20 characters only the first 20 characters shall be mapped.

In case, where the Device does not support "Product Name" due to its optional nature, the DeviceID shall be mapped to "PROFINET OrderID" instead as shown in the section below.

The IO-Link DeviceID is available via Direct parameter page 1: DeviceID. That means three octets shall be casted to a Visible String (range 0…..16777215).

Representation of OrderID:

- Starting with String "DeviceID: " xxxxxxxx (9 characters)
- Followed by the DeviceID string: 1-8 characters
- Padding with blanks
- Example: Char 1 to Char20 ("DeviceID: 291")

### 7.8.3.3 I&M5

So far only I&M0 mapping has been specified and it turns out that the complete information of Devices could not be mapped yet. Therefore, I&M5 is used to complement the identification and to achieve a consistent view of a Device. Table 18 shows the mapping.

**Table 18 – Mapping of Device information beyond I&M0**

| I&M5 data | Octets | Data type | Mapping rules |
|---|---|---|---|
| IM_UniqueIdentifier | 16 | UUID | Reference according PROFINET specification [3] |
| AM_Location | 64 | 64 octets | Fill with 0x00 |
| IM_Annotation | 64 | String (UTF8) | “IO-Link Devices” |
| IM_OrderID | 20 | Visible String | "Product Name" or "DeviceID". For details see7.8.3.2 |
| AM_SoftwareRevision | 64 | String (UTF8) | Insert SoftwareRevison of Device (IO-Link Index).<br>If it is not available set to "Not accessible" |
| AM_HardwareRevision | 64 | String (UTF8) | Insert HardwareRevison of Device (IO-Link Index).<br>If it is not available set to "Not accessible" |
| IM_Serial_Number | 16 | Visible string | Insert SerialNumber of Device (IO-Link Index 21).<br>If it is not available set to "Not accessible" |
| IM_Software_Revision | 4 | Char,3 x Unsigned8 | Set to V0.0.0 (official release but not detectable) |
| AM_DeviceIdentification | 8 | Unsigned16<br>Unsigned16<br>Unsigned16 | DeviceSubID : 0x0000<br>DeviceID<br>VendorID |

| I&M5 data | Octets | Data type | Mapping rules |
|---|---|---|---|
| | | Unsigned16 | Organization: 0x0001 (IO-Link) |
| AM_TypeIdentification | 2 | Unsigned 16 | 0x4E01 (API of IO-Link) |
| IM_Hardware_Revision | 2 | Unsigned 8 | Set to 0x0000 (default value) |

### 7.8.4 I&M filter data

I&M data of the IOLD proxy shall be inserted in I&M filter data. I&M0 filter data indicate all submodules holding discrete identification and maintenance data.

The I&M0 Filter Data object contains the information which submodules are associated with their own I&M data, which submodules represent their superordinate module and which submodule represents the whole Device. This object is global and a read-only object and it is not associated with any Submodule.

## 7.9 Record data

### 7.9.1 ISDU batch object

ISDU batch object provides means to write one to many ISDUs in one "block object", for example to convey several Device parameters. This object can be used to transfer a number of ISDU write requests to the IO-Link proxy using Index "0xB902".

The IOLD proxy is responsible to execute this request in the sequence as posted in the ISDU batch object (see Table 19). This data object is optional.

**Table 19 – Structure of an ISDU batch object**

| Part | Parameter name | Definition | Data type |
|---|---|---|---|
| Header | BlockVersionHigh | Versioning of record | Unsigned8 |
| | BlockVersionLow | Versioning of record | Unsigned8 |
| | Reserved | – | Unsigned16 |
| ISDU Object 1 | ISDU_Index | ISDU Index (0 to 65535) | Unsigned16 |
| | ISDU_Subindex | ISDU Subindex (0 to 255) | Unsigned8 |
| | ISDU_Length | Length of subsequent record | Unsigned8 |
| | ISDU_Data | Record of Length "ISDU-Length" | Record |
| ISDU Object 2 | ISDU_Index | ISDU Index (0 to 65535) | Unsigned16 |
| | ISDU_Subindex | ISDU Subindex (0 to 255) | Unsigned8 |
| | ISDU_Length | Length of subsequent record | Unsigned8 |
| | ISDU_Data | Record of Length "ISDU-Length" | Record |
| … | | | |
| ISDU Object *n* | ISDU_Index | ISDU Index (0 to 65535) | Unsigned16 |
| | ISDU_Subindex | ISDU Subindex (0 to 255) | Unsigned8 |
| | ISDU_Length | Length of subsequent record | Unsigned8 |
| | ISDU_Data | Record of Length "ISDU-Length" | Record |

The following rules apply:

- The ISDU batch object shall be rejected if the Device at the port is not running.
- The ISDU batch object shall be accepted if a Device at the port is running.
- In case of an error during the batch process (ISDU write to the Device) a diagnosis error shall be generated:
  ErrorType: 0x0010 (16) "Parameter Error"

ExtChannelErrorType: 0x8001 "Parameter Fault detail"
ExtChannelAddValue:
Bit 0 to 16: (Index) Index of ISDU (0 to 32767)
Bit 17 to 31: (Offset) Subindex of ISDU (0 to 255)

- A disappearing Event shall be generated after a successful transfer only if a pending diagnosis "ISDU batch failed" is available
- An ongoing ISDU batch process shall be indicated in "Port Status" record as "ISDU batch pending".

The benefits of using ISDU batch objects are:

- The programmer can transfer the entire set of Device parameters without complex handling of individual transfers
- In case of startup parameterization via PROFINET, this ISDU batch object record can be generated by PROFINET engineering and appended to the PROFINET Startup parameter
- At each PROFINET Startup, this Device parameter set will automatically be written to the Devices only if this record is included in the GSD file.
- Depending on a particular application the user can either activate block parameterization or Data Storage. In this case the first ISDU batch object shall be a SystemCommand "PrmBegin" (=ParamDownloadStart) and the last ISDU batch object a "PrmEnd" (=ParamDownloadEnd) or "ParamDownloadStore" respectively.

### 7.9.2 IOL backup object

#### 7.9.2.1 General

The user can have access to the Data Storage object in the Master using records and Index "0xB901" (IOLD backup object). Structure of an IOLD backup object is shown in Table 20.

**Table 20 – Structure of an IOL backup object**

| Part | Parameter name | Definition | Data type |
|---|---|---|---|
| Header | BlockVersionHigh | Versioning of record; first version: 0x01 | Unsigned8 |
| | BlockVersionLow | Versioning of record; first version: 0x00 | Unsigned8 |
| | Reserved | – | Unsigned16 |
| Body | Data Storage object | See 7.9.2.2 | Record |

#### 7.9.2.2 Data Storage object

IO-Link specifies an object which includes all actual Device parameters. The structure of this Data Storage object is defined in Annex F of [1]. The Master shall store the Data Storage object of each Device locally in non-volatile memory. The content of the Data Storage object is manufacturer/vendor defined.

## 8 Dynamic characteristics

### 8.1 Consolidated port configuration (CPC database)

### 8.1.1 Concept

The consolidated port configuration data comprise all relevant parameters for the start-up of an IO-Link port. They are stored in a "CPC database" in non-volatile memory.

The user can decide which one of the following sources provides input for the CPC database (see 11.2 for "port configuration modes"):

- Port configuration using the PROFINET engineering system and an appropriate GSD file. In this case, the record of parameter instances is called "PN_PC" (PROFINET port configuration) and stored in Index 0xB900 (see 7.4.2 for coding).

- Port configuration using the IO-Link PDCT tool and an appropriate IODD file. In this case, the record of parameter instances is called "IOL_PC" (IO-Link port configuration) and stored in Index 0xB10x (see 11.4.3 for coding).

An "IOL_PC" port configuration is only possible if the parameter "Port Configuration Mode" within the GSD file is preset to "Tool based configuration – PDCT".

### 8.1.2 Port start-up trigger

Device/port start-up, at which the configuration data from the CPC database are used, is caused in the following cases:

a) After power up of the Linking Module and/or Device

b) After any change within the CPC database

### 8.1.3 Extended port start-up

The port start-up and the Device start-up respectively are both specified in [1]. In case of PROFINET integration, an "Extended Start-up" after this regular start-up will take place. It is characterized by a Read of the IO-Link identification data objects (see Figure 13 for a Device according "V1.1").

Basically, the port/Device start-up comprises the following phases:

- Wake-up and adjustment of transmission rate (establish communication)
- Verification (adjustment of IO-Link revision and comparison of configured versus real Device according to the inspection level)
- Data Storage handling with download and upload of parameter instance values
- Extended start-up with
  - Read SerialNumber (Index 0x0015)
  - Read ProductName (Index 0x0012)
  - Read ProductID (Index 0x0013)

After a successful "Extended Start-up" the GWA service "ReadyToOperate" shall be indicated.

A fault during start-up or "Extended Start-up" shall trigger the indication of GWA "ComFault". Additionally, it shall be accompanied by the cause for the fault, e.g. "no Device", "incorrect Device", etc.

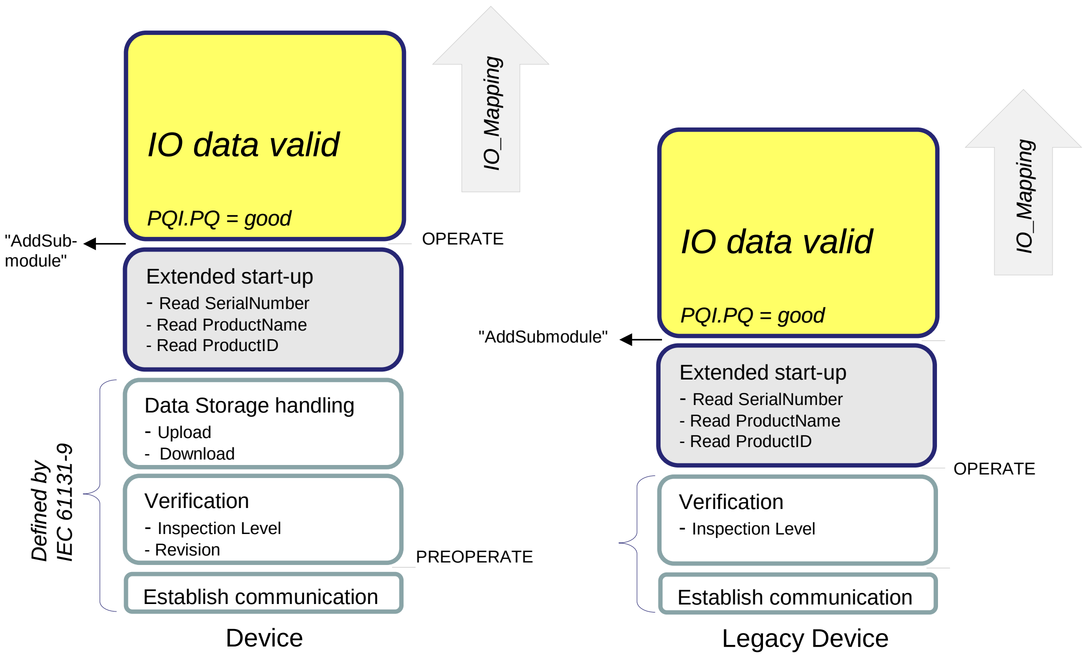



**Figure 13 – Extended port start-up**

### 8.1.4 Buffered port configuration model

#### 8.1.4.1 Principle

Figure 14 shows the data paths from the two sources to the CPC database.

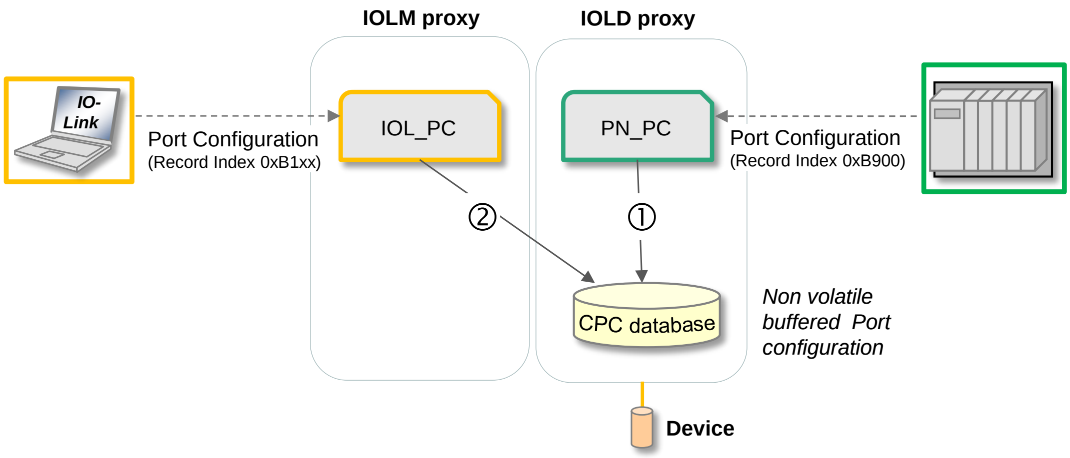



**Figure 14 – Data paths to a consolidated port configuration (CPC)**

Generally, the PROFINET port configuration (PN_PC) is transferred as start-up record 0xB900 to the IOLD proxy during PROFINET start-up.

An IO-Link Tool (PDCT) can transfer a port configuration (IOL_PC / 0xB1xx) to the IOLM proxy ("Download Port Configuration"). "xx" within the Index represents a particular port number (up to 255 ports).

The CPC database is built up from both activities ① and ② and contains finally a non volatile buffered port configuration as the basis for the IO-Link start-up.

A read of IOL_PC or PN_PC shall show the consolidated port configuration that is currently active.

#### 8.1.4.2 PROFINET port configuration (PN_PC)

The transfer of PN_PC is controlled by the parameter "PortConfigControl.PortConfigurationMode (see 7.4.2). The following rules apply:

- PortConfigurationMode "tool based configuration" within the record PN_PC results in no further activities since the tool (PDCT) is responsible to fill the CPC database.
- PortConfigurationMode "all other codings" means
  - matching data in CPC database and PN_PC lead to the mapping of payload Process Data and completion of the start-up ("Fast start-up")
  - no matching data in CPC database and PN_PC will cause the CPC database to accept the PN_PC configuration and will lead to a re-configuration

#### 8.1.4.3 Tool based port configuration (IOL_PC)

A download of the port configuration IOL_PC via record 0xB10x the following examinations take place:

- If the value of parameter PortConfigurationMode does not correspond to "tool based configuration", the record will be (-) acknowledged with "port configuration blocked".
- If the value of parameter PortConfigurationMode corresponds to "tool based configuration" and IOL_PC matches the content of the CPC database, the record will be (+) acknowledged.
- If the value of parameter PortConfigurationMode corresponds to "tool based configuration" and IOL_PC does not match the content of the CPC database, the following applies:

- CPC database incorporates the IOL_PC configuration
- port starts re-configuration due to new data
- record will be (+) acknowledged.

## 8.2 Power-on behavior

After power-on of the PN device/Linking Module the current IO-Link configuration of the Master shall be acquired and transferred to the PROFINET system via "RealIdentification".

For this, the following base mechanisms shall be used (see 5.4.1):

- AddSubmodule (API 0x4E01, Slotnumber $k$, SubslotNumber $x$, SubmoduleIdentNumber
- RemoveSubmodule (API 0x4E01, Slotnumber $k$, SubslotNumber $x$)

NOTE Clause 6 provides the mapping of port to Submodule

The ports are started promptly after power-on using the stored PortConfiguration within the CPC database ("OperatingMode").

NOTE Clause 8.1.4 explains the creation of the CPC database.

Parallel to this action, the IOLM proxy Submodule shall be registered via AddSubmodule (API 0x4E01, Slotnumber $k$, SubslotNumber $x$, SubmoduleIdentNumber = vendor specific).

Table 21 shows the conditions and actions for the power-on behavior.

**Table 21 – Condition/action table for power-on behavior**

| Condition | Action |
|---|---|
| GWA: ReadyToOperate<br>"Device is ready" | AddSubmodule (API, SlotNumber $k$, SubslotNumber $x$, SubmoduleIdentNumber = 0x0000 0021) |
| GWA: ComFault (no IOLD)<br>"No Device detectable/no communication" | Remove Submodule (API, SlotNumber $k$, SubslotNumber $x$) |
| GWA: ComFault (incorrect IOLD)<br>"Unexpected Device" | AddSubmodule (API, SlotNumber $k$, SubslotNumber $x$, SubmoduleIdentNumber = 0x0000 0021) |
| GWA: ComFault (other problems)<br>"Fault at start-up; communication possible" | AddSubmodule (API, SlotNumber $k$, SubslotNumber $x$, SubmoduleIdentNumber = 0x0000 0021) |

Figure 15 shows the message sequences of the "Linking Module" at power-on.

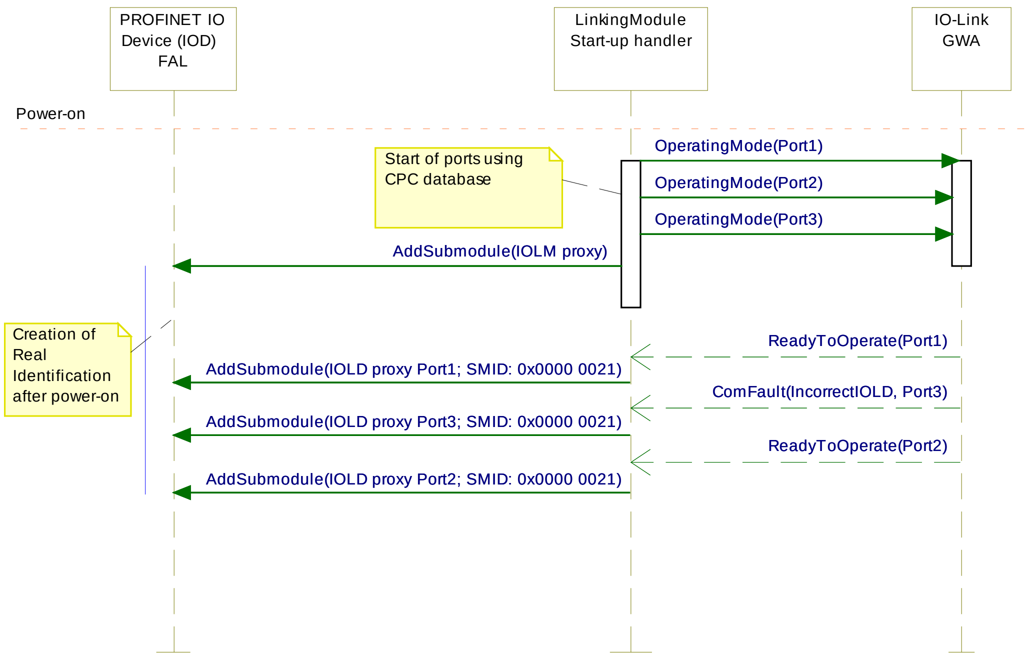



**Figure 15 – Linking Module at power-on**

## 8.3 Establish Application Relationship (AR)

### 8.3.1 General

At least one Application Relationship between the PROFINET IO Controller (PN-C) and PROFINET IO Device (PN-D) will be established to arrange for the data exchange between both. Each change of a Module/Submodule configuration will be handled via a new Application Relationship.

This clause provides a description on how an Application Relationship is established within the context of a Linking Module/Master in IO Controller engineering. Figure 16 shows the actions and reactions by means of FAL services and GWA services. The particular phases of the establish Application Relationship are associated with corresponding actions on the IO-Link side (GWA). Three main actions on the IO-Link side are described in the following.

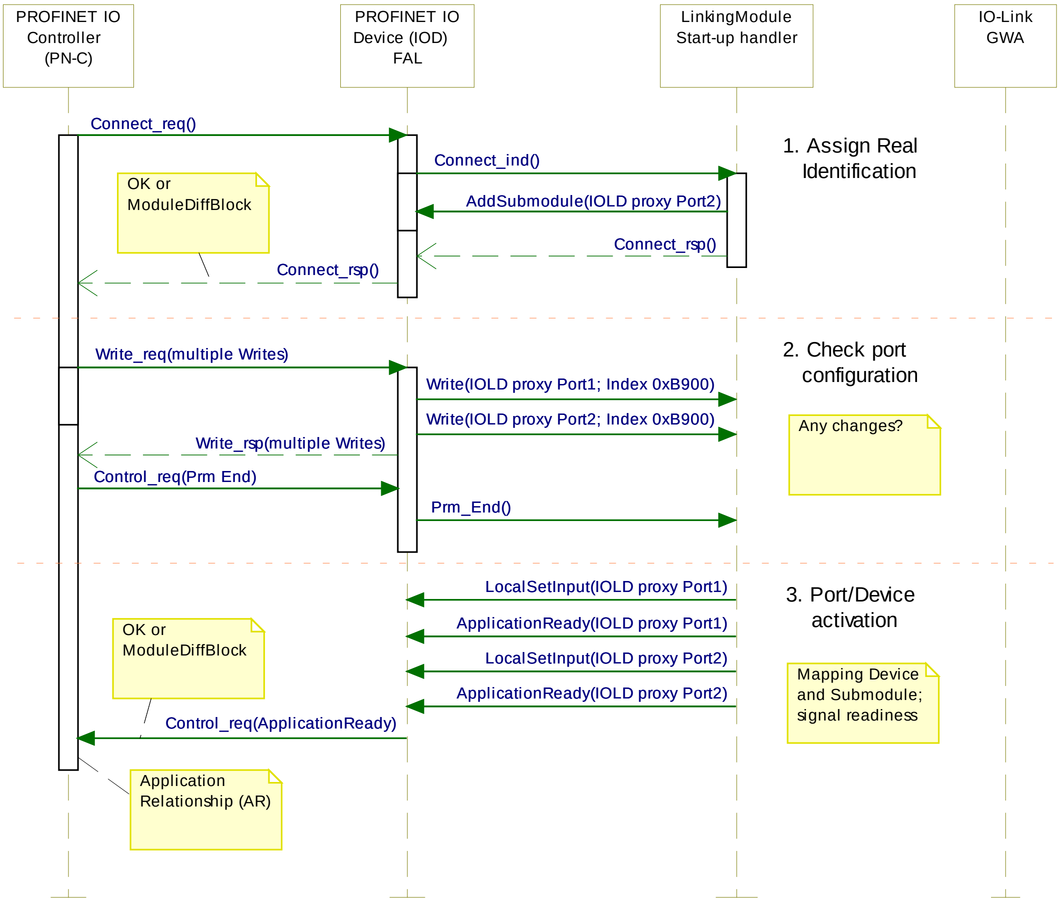



**Figure 16 – Establish Application Relation (AR)**

*Assign Real Identification*

After a "Connect.ind" of the PROFINET start-up, the mapping application within the Linking Module assigns the RealIdentification. For details see 8.3.2.

*Check port configuration*

The start-up phase is initiated by a "Write.req" (multiple Writes) of the PROFINET IO Controller (PN-C). As a consequence, the PortConfiguration records (0xB900) will be transferred to the "Mapping Application" and checked. For details see 8.3.3.

*Port/Device activation*

A "Prm_end" initiates that the Input Data (IOPS) will be updated and the end of the start-up will be signalled via an "ApplicationReady". For details see 8.3.4.

### 8.3.2 Phase: Assign Real Identification

Purpose of this action is to compare the Expected Identification (ExpectedSubmoduleBlock within the "Connect" service) with the current IO-Link port/Device status and to initiate adequate actions. Table 22 shows the actions within the Real Identification phase and Table 23 the used special internal SubmoduleIDs.

The beginning of this phase is characterized by a "Connect.ind" with the "ExpectedSubmoduleBlock" and the end by a "Connect.rsp" with the "ModuleDiffBlock".

**Table 22 – Real Identification actions**

| Conditions/cases/guards | Actions |
|---|---|
| If the ModuleIdentNumber (Expected Module) does not match the expected ModuleID | Modulestate: incorrect module |
| If the ModuleIdentNumber (Expected Module) matches the expected ModuleID or is compatible | Modulestate: OK or substitute |
| If an unknown Submodule (SMID unknown) is required in "ExpectedSubmoduleBlock" | AddSubmodule (API, SlotNumber *k*, Sub-slotNumber *x*, SubmoduleIdentNumber = 0xFFFF 0000*)*<br>SubmoduleState: incorrect Submodule |
| If an IOLD proxy Submodule (SMID="IOLD proxy x") is required in "ExpectedSubmoduleBlock" and no Submodule is registered | AddSubmodule (API, SlotNumber *k*, Sub-slotNumber *X*, SubmoduleIdentNumber = 0x0000 0021*) → temporary port proxy*<br>SubmoduleState: substitute |
| If an IOLD proxy submodule is required (SMID="IOLD proxy x") in "ExpectedSubmoduleBlock" and an IOLD proxy submodule is already registered. Compare "expected" and "registered" SMID.<br>If Identical:<br>If not identical: | <br><br><br>Submodule State: OK<br>SubmoduleState: Substitute |
| If a DI Submodule (SMID 0x0000 0081) is required in "Expected SubmoduleBlock" | AddSubmodule (API, SlotNumber *k*, Sub-slotNumber *x*, SubmoduleIdentNumber=0x0000 0081*)*<br>SubmoduleState: OK<br>Start DI Mode |
| If a DO submodule (SMID 0x0000 8100) is required in "ExpectedSubmoduleBlock" | AddSubmodule (API, SlotNumber *k*, Sub-slotNumber *x*, SubmoduleIdentNumber=0x0000 8100*)*<br>SubmoduleState: OK<br>Start DO Mode |

**Table 23 – Special internal SubmoduleIDs**

| SubmoduleID (SMID) | Definition |
|---|---|
| 0xFFFF 0000 | Incorrect Submodule |
| 0xFFFD 0000 | Temporary port proxy |

### 8.3.3 Phase: Check port configuration

In this phase the Port Configuration records (Index 0xB900) are evaluated per port. See 7.4.2 for details. Purpose of this action is to restart a port/Device in case of configuration changes within the CPC-base.

The beginning of this phase is characterized by a "Write.req" and the end by "Prm_end".

Table 24 shows the actions within the check port configuration phase.

**Table 24 – Check port configuration actions**

| **Conditions/cases/guards** | **Actions** |
|---|---|
| If CPC_base has been changed due to a new port configuration | RemoveSubmodule (API, SlotNumber *k*, SubslotNumber *x*)<br>OperatingMode (new Port Configuration) |

### 8.3.4 Phase: Port/Device activation

In this phase all connected IO-Link Devices shall be activated from a PROFINET point of view and the PROFINET start-up shall be closed.

Generally, a "Control.req/ApplicationReady" will be sent to the PROFINET IO Controller (PN-C) as soon as an "ApplicationReady" with the corresponding status for all Submodules of the ExpectedSubmoduleBlock has been sent. The PROFINET device start-up is closed thereafter.

The "Control.req" can carry a ModuleDiffBlock to inform the PN-C in case of deviations to the Expected Identification.

Hint: This behavior is port respectively Submodule specific and will be performed for each Submodule being part of the "expected identification (ExpectedSubmoduleBlock)".

The beginning of this phase is characterized by a "Prm_End.ind" and the end by an "ApplicationReady". Table 25 shows the port/Device activation actions.

**Table 25 – Port/Device activation actions**

| Conditions/cases/guards | Actions |
|---|---|
| If an IOLD proxy has been registered (Real) and the SMID is 0xFFFF 0000 (incorrect submodule) | Application Ready (incorrect Submodule)<br>SubmoduleState = incorrect Submodule |
| If an IOLD proxy has been registered (Real) and the SMID is within the range of 0x0000 0001 to 0x0000 FFFF (generic Submodules) and IO data length OK (see NOTE 1) | Update port qualifier<br>Local Set Input (Input data, IOxS = good)<br>Application Ready (substitute) |
| If an IOLD proxy has been registered (Real) and the SMID is within the range of 0x0000 0001 to 0x0000 FFFF (generic Submodules) and IO data length not OK (see NOTE 1) | Update port qualifier<br>Local Set Input (Input data, IOxS = bad)<br>Application Ready (incorrect Submodule) |
| If an IOLD proxy has been registered (Real) and the SMID is 0xFFFD 0000 (temporay port Submodule) | Timeout detection (after 1 s timeout)<br>RemoveSubmodule (see NOTE 2)<br>Application Ready (no Submodule) |
| If a DI submodule has been registered (Real) and the SMID is 0x0000 0081 (DI proxy submodule) | Detect Input<br>Local Set Input (Input data, IOxS = good)<br>Application Ready (OK) |
| If a DO submodule has been registered (Real) and the SMID is 0x0000 8100 (DO proxy Submodule) | Local Set Input (Input data, IOxS = good)<br>Application Ready (OK) |
| NOTE 1 IO data length OK: The IO data of the Device can be mapped into the IOLD proxy Submodule;<br>IO data length not OK: The IO data are too large to be mapped into the IOLD proxy Submodule.<br>NOTE 2 Until timeout "ReadyToOperate" or "ComFault" can occur. After timeout an "ApplicationReady (no Submodule)" shall be sent. Hint: A delayed start-up of a Device can cause a "Plug". | |

## 8.4 Pull/plug behavior

### 8.4.1 Overview

The availability or the failure of a Device respectively will be mapped to PROFINET services (FAL) as shown in Table 26. This mapping is independent from the phases such as power-on or start-up, i.e. the actions can occur also during establishing an Application Relationship and thus during a Device start-up.

The parameter "PortConfigControl.Enable Pull/Plug" = 0 cares for omittance of Pull/Plug Events (AddSubmodule/RemoveSubmodule), see Table 9.

**Table 26 – Availability/failure mapping to FAL services**

| Availability/failure | FAL services |
|---|---|
| GWA: ReadyToOperate | AddSubmodule (API, SlotNumber *k*, SubslotNumber *x*, SubmoduleIdentNumber=0x0020 0021*)* |
| GWA: ComFault (no Device) | RemoveSubmodule (API, SlotNumber *k*, SubslotNumber *x)* |
| GWA: ComFault (incorrect Device) | AddSubmodule (API, SlotNumber *k*, SubslotNumber *x*, SubmoduleIdentNumber = 0x0020 0021*)* |
| GWA: ComFault (other problems) | AddSubmodule (API, SlotNumber *k*, SubslotNumber *x*, SubmoduleIdentNumber = 0x0020 0021*)* |

### 8.4.2 Pull sequence

The failure of a Device will be reported via "ComFault" (no Device) and will lead to a "RemoveSubmodule" as shown in Table 26.

The actions in Table 27 shall be performed upon a pull when the Application Relationship is established and the IOLD proxy Submodule is activated (SubmoduleState: OK or substitute).

**Table 27 – Pull actions**

| Condition/case/guard | Action |
|---|---|
| GWA: ComFault (no Device) | Local Set Input (Input data, IOxS = bad)<br>Remove Submodule (API, SlotNumber *k*, SubslotNumber *x)* |

Figure 17 shows how a PROFINET IO Controller (PN-C) is informed about a pulled Submodule.

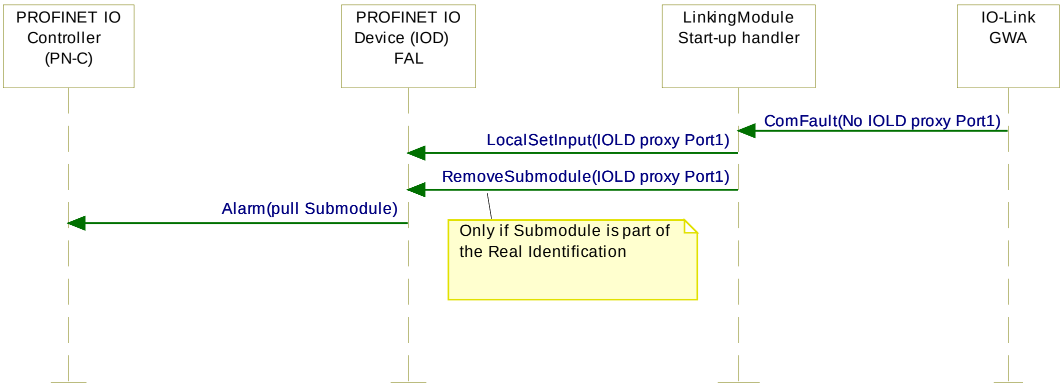



**Figure 17 – Pull sequence**

### 8.4.3 Plug sequence

The return of a Device can be signalled in several ways:

- "ReadyToOperate" (Device is ready)
- "ComFault" (incorrect Device). Inspection Level check resulted in the detection of an incorrect or incompatible Device, which shall be reported through an Appearing Event.
- "ComFault" (other problems). Device started but is not ready yet due to a fault, which shall be reported through an Appearing Event.

Table 26 shows the availability/failure mapping to FAL services.

The actions in Table 28 shall be performed upon a plug whenever the Application Relationship is established and a corresponding IOLD proxy Submodule has been identified.

**Table 28 – Plug actions**

| Conditions/cases/guards | Action |
|---|---|
| If an IOLD proxy has been registered (Real) and the SMID is within the range 0x0000 0001 to 0x0000 FFFF (generic Submodules) and IO data length OK (see NOTE) | Update Port qualifier<br>Local Set Input (Input data, IOxS = good)<br>Application Ready (substitute) |
| If an IOLD proxy has been registered (Real) and the SMID is within the range 0x0000 0001 to 0x0000 FFFF (generic Submodules) and IO data length not OK | Update Port qualifier<br>Local Set Input (Input data, IOxS = bad)<br>Application Ready (incorrect Submodule) |
| NOTE IO data length OK: The IO data of the Device can be mapped into the IOLD proxy Submodule;<br>IO data length not OK: The IO data are too large to be mapped into the IOLD proxy Submodule. | |

Figure 18 demonstrates how the PROFINET IO Controller (PN-C) is informed about a plugged Submodule (return).

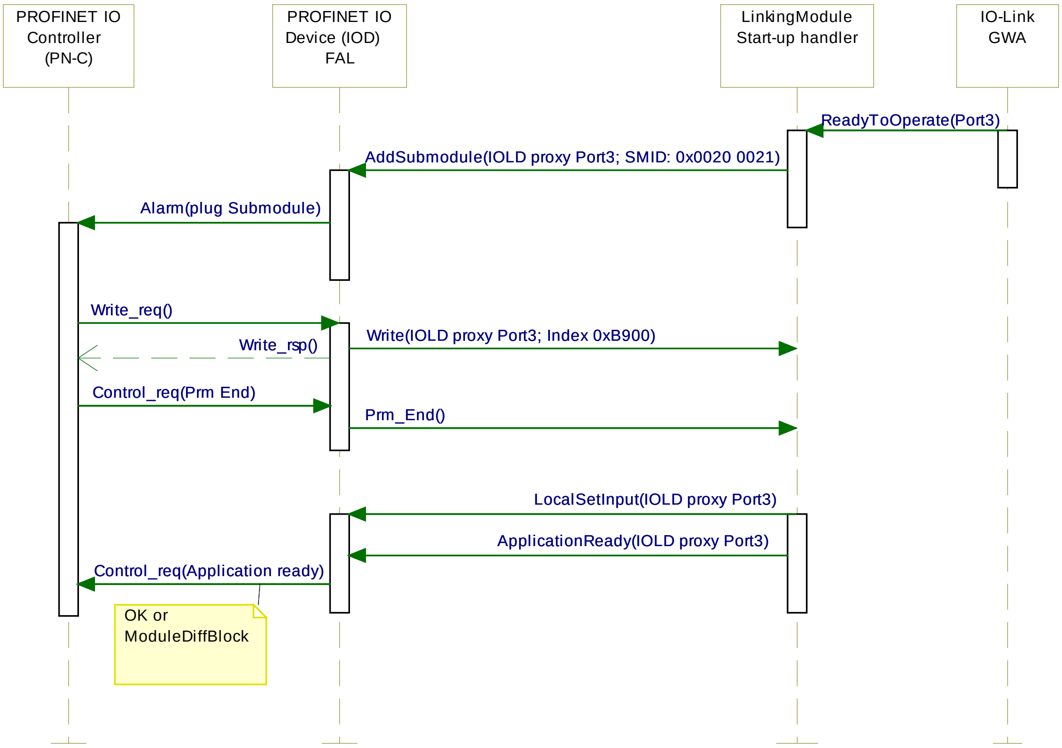



**Figure 18 – Plug sequence**

## 8.5 Impact of configuration changes

With respect to configuration changes, it has been a design objective for the system behavior during and after PROFINET start-up to prevent impacts on performance as much as possible.

Thus, it has been assumed that no Device start-up takes place if configuration and parameterization remains unchanged during start-up.

Configuration and parameterization data are always stored according to [1]. After start-up, new configuration/ parametrization data are checked against stored configuration/parameterization data. The following rules apply:

- If a new Device (Submodule) has been added, all other Devices/Submodules shall remain unchanged and the new Device shall be activated (Device start-up).
- If a Submodule (parameterization) has been changed, only the affected Device shall be restarted based on the new configuration/parameterization.

# 9 Extended Data Storage and application support (Tool-Changer)

## 9.1 Backup & Restore

### 9.1.1 Backup alert

The Data Storage mechanism between IO-Link Masters and Devices is specified in [1]. For integration into PROFINET, it is mandatory for Masters to provide an "IOLD Backup" object comprising all active Device parameters (if Data Storage is enabled).

The following rules apply (see Figure 19):

- Save the Device parameters in the system context using a Read (Index) method
- Restore Device parameters, for example after Device replacement

Each change of Device parameters leading to an update of the "IOLD Backup" object within the Linking Module shall be indicated by setting the flag "NewPar" in the Port Qualifier Information (PQI). A Read of the "IOLD Backup" object (Index 0xB904, see Figure 19) shall cause a reset of this flag.

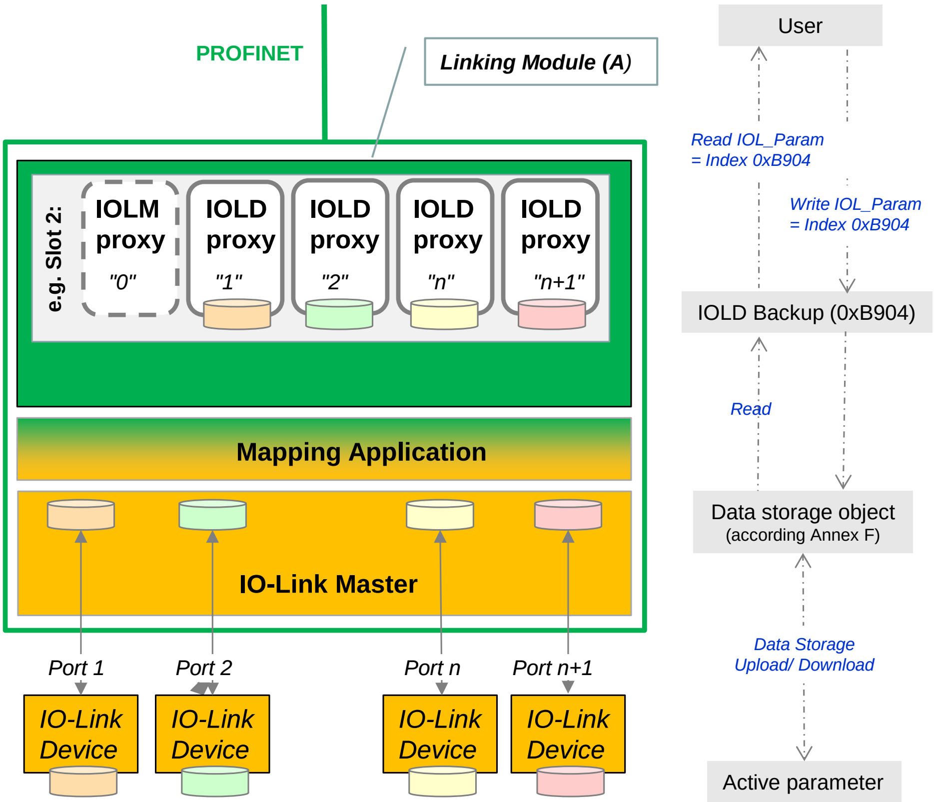



**Figure 19 – Extended Data Storage**

A Write of an "IOLD Backup" object to the appropriate port shall lead to a restart of the Device with the new parameters.

Figure 19 provides an overview of the different parameter storages in Devices, Masters, and Linking Modules.

### 9.1.2 Save IO-Link parameter

From an application point of view it is possible to save Device parameters by reading the "IOLD Backup" object from the corresponding IOLD proxy.

If an "IOLD Backup" object is invalid, the Read record "IOLD Backup" will be rejected (ErrorCode 0xB6 "Access denied"). Possible reasons are

- the connected Legacy Device ("V1.0") that does not support Data Storage

- Data Storage object is invalid or cleared

### 9.1.3 Restore IO-Link parameter

Whenever necessary, for example after plugging in a Device, the user can restore stored parameters using a Write "IOLD Backup".

A Write "IOLD Backup" shall be rejected (ErrorCode 0xB6 "Access denied") in case of:

- Backup Level "Off"
- Port configured for a legacy Device ("V1.0" = no support of Data Storage)
- Backup Record inconsistent or incorrect, for example a Backup object relying on a particular VendorID/DeviceID. Transfer of a Backup object to a Port with different configuration shall be denied.

A Write "IOLD Backup" shall be accepted if port is configured in "V1.1" mode and Backup Level in either "Backup and Restore" or "Restore".

The following two use cases shall be considered:

*Device is running during restore of IO-Link parameters:*

After a transfer of the Device parameters via Write "IOLD Backup", the Master shall adopt the content and shall on his part start writing to the Device according to the IO-Link Index list. Restore shall cause a restart of the Port.

*Device is not running during restore of IO-Link parameters:*

After a transfer of the Device parameters via Write "IOLD Backup", the Master shall adopt the content in the Data Storage object.

### 9.1.4 Data Storage on PROFINET level

A Data Storage upload causes an update of the "IOLD Backup" object and will set the "NewPar" flag in the Port Qualifier Information (PQI) as already shown in 9.1.1. This flag shall be reset by either a Read or Write of the "IOLD Backup" object.

A PLC user program or a system function block (FB) can store updated Device parameters upon the detection of a rising edge of the "NewPar" flag (see [8]). Thus, the Data Storage content is propagated to the PROFINET level and the Device parameters are saved in case of a Master failure and replacement.

## 9.2 Automatic Tool Changer application

### 9.2.1 Requirements

Figure 20 shows an example of an Automatic Tool Changer application.

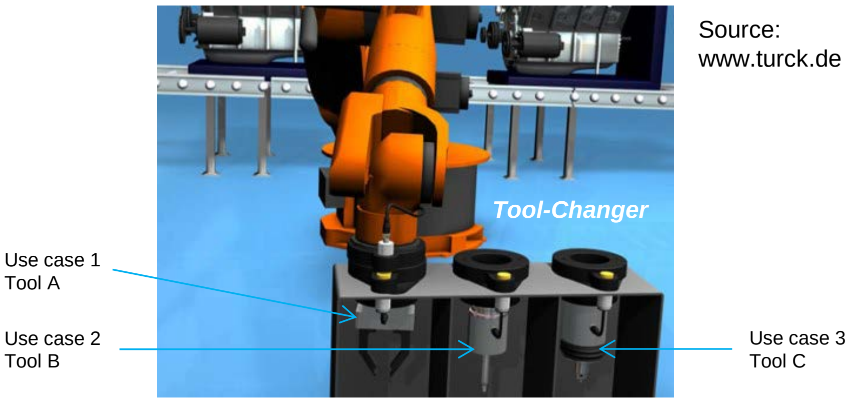



**Figure 20 – Example of an Automatic Tool Changer (ATC)**

Depending on the state of a production process, a machine undocks a particular tool, for example a gripper, in a magazine and docks another one. This docking and undocking comprises mechanical joint and electrical connections for power supply as well as for communication.

This document specifies support for ATC applications in two aspects:

- "deactivation" of a port prior to undocking a tool (state "Port deactivated")
- "activation" of a port after docking another tool (state "Port activated")

Three use cases shall be supported as defined in Table 29.

**Table 29 – ATC use cases**

| Use case | Description |
|---|---|
| 1 | A particular Device will be replaced by a Device of the same type with identical parameters |
| 2 | A particular Device will be replaced by a Device of the same type with different parameters |
| 3 | A particular Device will be replaced by a Device of a different type |

### 9.2.2 Activate/Deactivate Port in general

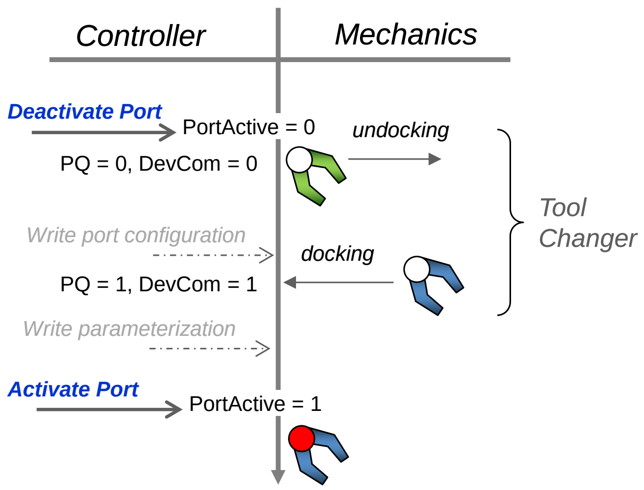
Basic concept of the user function "Deactivate Port" is to suppress the entire fault indications to the system/user since it concerns an intended action. In essence, after the deactivation all pending diagnosis messages of the related port and Device are deleted.

The user reaches state "Port activated" via function "Activate Port" (see 10.7.2, Table 43) or state "Port deactivated" via function "Deactivate Port" respectively (see 10.7.2, Table 43). The current port status is always visible to the user via the status of "Port Qualifier Information – PQI" (see 7.5.4.1).

*Automatic activation*

As a consequence of the following actions, state "Port deactivated" (indicated by flag PortActive = 1) will change automatically to state "Port activated":

- power-on of the PN-device or Master;
- a configuration change of the IOLD proxy;
- Port Mode Digital Input or Digital Output.




**Key** Port Qualifier Information: PQ, DevCom, PortActive

**Figure 21 – Port Activate/Deactivate**

Figure 21 provides an overview of the mechanisms and serves as visualization of the following actions:

- A successful deactivation (state "Port deactivated") leads to an indication of this state via PortActive = 0;
- Undocking of the Tool/Device leads to Port Qualifier bits PQ = 0 and DevCom = 0;
- Docking of a "new" Tool/Device leads to Port Qualifier bits PQ = 1 and DevCom = 1;
- A successful activation leads to an indication of this state via PortActive = 1.

Table 30 specifies the backup behavior for the three use cases of Table 29.

**Table 30 – Port activation/deactivation and backup behavior**

| Use case | Inspection Level | Description |
|---|---|---|
| 1 | 0: no Device check<br>1: type compatible Device (V1.0)<br>2: type compatible Device (V1.1)<br>3: type compatible Device (V1.1) with Backup + Restore<br>4: type compatible Device (V1.1) with Restore | All Inspection Levels are permitted in use case 1.<br>Recommended: type compatible Device V1.1 with Backup + Restore |
| 2 | 0: no Device check<br>1: type compatible Device (V1.0)<br>2: type compatible Device (V1.1) | Backup and Restore not reasonable in use case 2<br>Recommended: type compatible Device (V1.0 or V1.1) |
| 3 | 0: no Device check<br>1: type compatible Device<br>2: type compatible Device (V1.1) | Backup and Restore not reasonable in use case 3<br>Recommended: type compatible Device (V1.0 or V1.1) |

The following rules apply:

- The port configuration can be adapted while in state "Deactivate" (use case 3).
- In addition, the parameterization of the Device can be adapted after active communication (DevCom =1) through control program (use case 2).
- Especially in use case 2 and 3 it is recommended to deactivate the Backup & Restore function for better transparency and start-up performance.

### 9.2.3 "Deactivate Port" function

A "Deactivate Port" command shall switch the port in a particular mode. It is assumed for this mode that the user application continues processing Process Data and Port Qualifier information. However, it can be removed without causing specific Events. System diagnosis will not report any fault.

The port function "Deactivate Port" can be performed using the IOL_CALL function block (see 10.7). Access point is the IOLM proxy. The port function shall be rejected in case of an IOLD proxy such as Digital Input or Digital Output or Port Mode "Digital Input/Digital Output".

Reception of a "Deactivate Port" command via an IOL_CALL request shall cause the following reactions:

*In case of Device not available*

- Port function shall be acknowledged, Port Status shall be set to "Port deactivated", and IOL_CALL_rsp(+) shall be returned.

*In case of running IO-Link communication and available Device (see Figure 22)*

- Pending diagnosis Events shall be omitted via "disappearing" diagnosis Events (Local Remove Diagnosis Entry).
- The flag "PortActive" in Port Qualifier Information shall be set to "0" (Local Set Input) and the Status "Port deactivated" is stored.

An IOL_CALL_rsp(+) shall be sent if all actions are successful (Device is deactivated). For the user, the port indicates no faults (green LED).

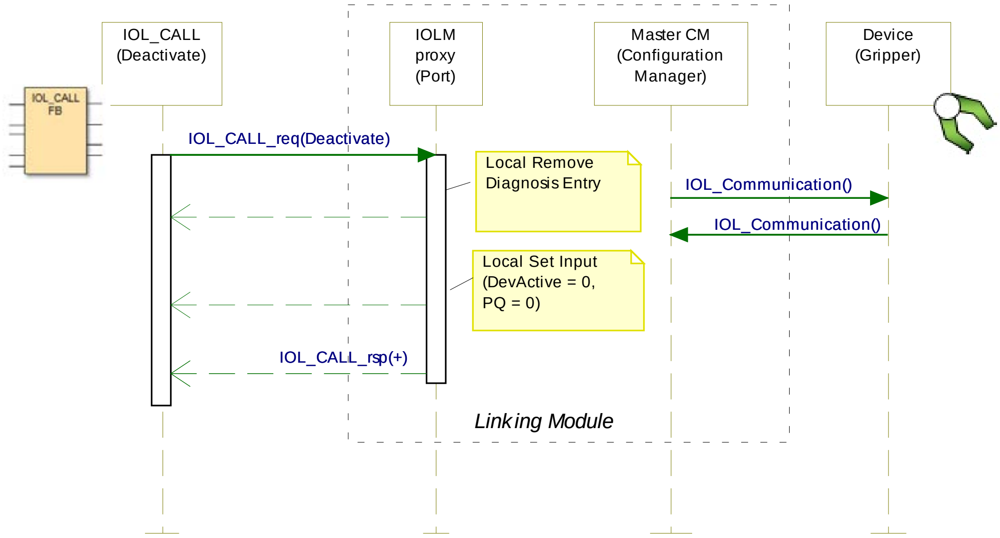



**Figure 22 – Sequence chart for "Deactivate Port" actions**

The state "Port Deactivated" is characterized by:

- Communication shall continue if Device is available
- Communication interrupt shall prevent from PROFINET Pull/Plug Events
- Process Data inputs shall be cyclically updated such that the user can process these Device data. In case of no communication with the Device, the input data shall be set to "0" (PQ = 0 = invalid, DevCom = 0 = no communication).
- The user can write Process Data to the Device if necessary (e.g. substitute value). In this case a preset is possible, taking place immediately after docking and activation.
- Read/Write via IOL_CALL function block is possible.
- The Port Qualifier Information shall be updated cyclically to show the correct port status.
- Diagnosis Events from port or Device respectively shall be recognized but not propagated to PROFINET (local storage of pending diagnosis Events).
- PDoutputInvalid shall be indicated when the Device is communicating.

### 9.2.4 "Activate Port" function

An "Activate Port" command shall re-activate the diagnosis mechanism. That means, the current diagnosis state shall be restored, or if no Device is available, a pull Event for the IOLD proxy shall be generated.

The port function "Activate Port" can be performed using the IOL_CALL function block (see 10.7). Access point is the IOLM proxy. The Port function shall be rejected in case of an IOLD proxy such as Digital Input or Digital Output ("Port function not supportd").

Port Status "Port activated" shall be indicated in case of:

- Reception of an "Activate Port" command. After activation IOL_CALL_rsp(+) shall be returned.
- Automatic activation (see 9.2.2)

Possible transitions from "Port deactivated" to "Port activated":

- Whenever a Device has been detected and no IOLDproxy is in place, this Device shall be registered via Plug (IOLD proxy) using "AddSubmodule" (IOLDproxy/SMID: 0x00000021). Upcoming diagnosis shall be sent (appearing Events while Device had been deactivated – see Local Add Diagnosis Entry).

- Whenever a Device has been detected and an IOLDproxy is in place, an upcoming diagnosis shall be sent (appearing Events while Device had been deactivated –see Local Add Diagnosis Entry).
- Whenever no Device has been detected and an IOLDproxy is in place, registration shall be canceled via "RemoveSubmodule".
- Whenever no Device has been detected and no IOLDproxy is in place, no action will take place.

IO-Link communication for "Activate Port" actions is shown in Figure 23.

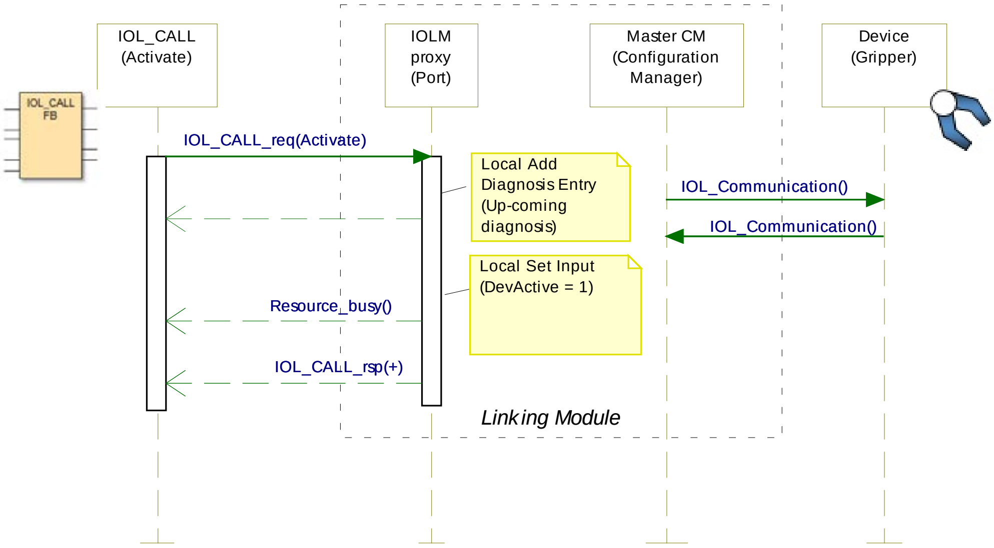



**Figure 23 – Sequence chart for "Activate Port" actions**

- The flag "PortActive" in Port Qualifier Information shall be set to "1" (Local Set Input)
- IOL_CALL response shall be positive (+)

## 9.3 Detection of Device exchange

The method specified herein detects with the help of a Device's serial number whether a Device has been replaced. Thus, it can only be used if SerialNumber is supported (see [1]). If this is not the case, the flag "SubDev" shall be set to "0".

At each start-up of a Device the SerialNumber (Index 0x0015) shall be read and stored if available.

The following actions apply:

- Perform Read of SerialNumber after Device start-up via Index 0x0015 after Data Storage handling prior to switching to OPERATE
- Comparison of acquired and stored SerialNumber
- In case of mismatch, flag "SubDev" shall be set and made available within the Port Status. No action in case of match.
- Subsequently, the SerialNumber shall be stored in non-volatile memory.

Flag "SubDev" shall be reset at each Read or Write of the "IOL Backup" object.

Consequence: The backup parameters can be read every time the flag "SubDev" has been set.

# 10 IOL_CALL method

## 10.1 Overview

The method "IOL_CALL" allows for accessing On-request Data (OD) across PROFINET into a Master and from there via ISDU to a Device. Thus, it is possible for example to parameterize a Device with the help of an AL_Write or to acquire a SerialNumber with the help of an AL_Read (see [1]).

In order to address the required IO-Link data, "IOL_CALL" uses PROFINET access points ("record index") and the Master port number in addition to the IO-Link Indices and Subindices as well as the transfer direction.

Additionally, "IOL_CALL" provides the possibility to initialize "port functions" via commands to the port such as "Deactivate/Activate Port" (see 10.7).

"IOL_CALL" has been specified already in [5] and the technology has been adopted here without major changes such that existing implementations can be reused.

## 10.2 Client interface of IOL_CALL

Clients for the method "IOL_CALL" can be located in Master tools, PC programs or in PLCs as function blocks (FB). The client interface consisting of the call and the arguments is shown in Figure 24.

**IOL_CALL (ID, CAP, Port, RD/WR, IOL_Index, IOL_Subindex, IOL_Data)**

**Figure 24 – IOL_CALL interface**

The arguments are specified in Table 31. Formally the method IOL_CALL uses a sequence of PROFINET Write record and Read record services to an Index specified via "CAP". The method addresses a proxy Submodule (IOLDproxy or IOLMproxy) characterized by IP address, AREP (reference to AR), API, Slot, and Subslot. In Table 31 the abbreviation "ID" is used for this "address".

**Table 31 – IOL_CALL arguments**

| Parameter | Definition |
|---|---|
| ID | Addresses the IOLD proxy (Submodule) and thus the Device or the IOLM proxy |
| CAP | The Client Access Point represents the PROFINET record index providing the "tunnel" to the IO-Link system. The value of this index is 0xB400. |
| Port | Port address the function shall be performed at. Supported range: 0 to 255 |
| RD/WR | Indicates whether the On-request Data (OD) shall be read (RD) or written (WR) |
| IOL_Index | Index of the On-request Data (see [1]) or Port Function Indicator respectively |
| IOL_Subindex | Subindex of the On-request Data (see [1]) or Port Function Command Code |
| IOL_Data | On-request Data to be written to, or to be read from the Device |

## 10.3 IOL_CALL Server entity

Figure 25 shows the mandatory IOL_CALL client/server structure/configuration. The Linking Module provides the server entities to perform the communication with the Device or the port functions. Several server entities are possible, addressable via the corresponding proxy submodules.

Server entities consist of a combination of an IOLD proxy and a Client Access Point (CAP, see Table 31). That means, it is possible for a manufacturer/vendor to implement several server entities per port for the sake of compatibility or the flexibility of several clients per port.

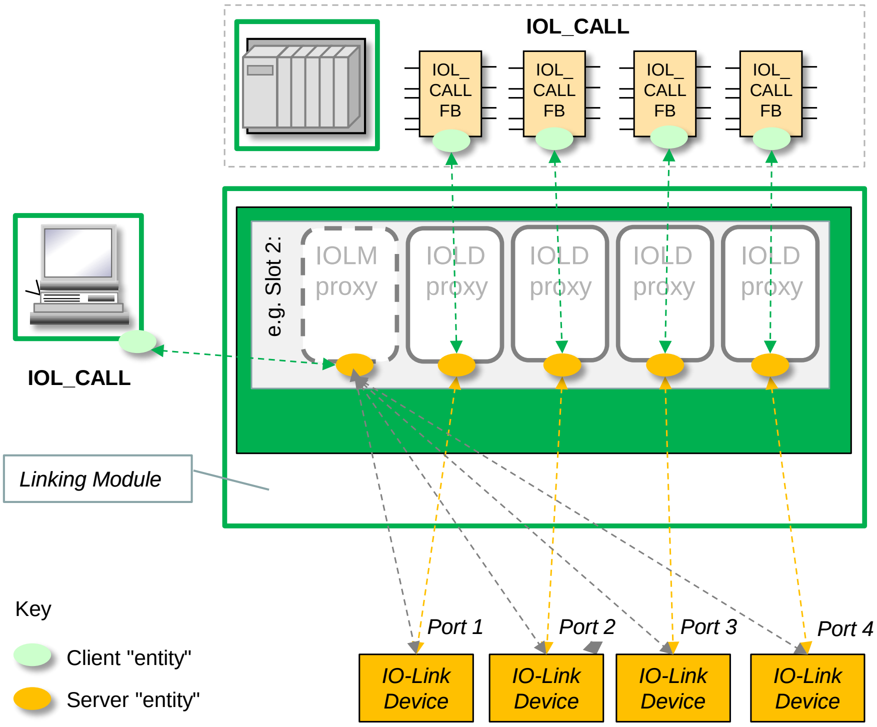



**Figure 25 – IOL_CALL client/server structure**

The following rules apply:

- Every IOLD proxy shall support at least one server entity with CAP = 0xB400 to manage tasks for the associated port. From a server's point of view the port number does not care in this case due to the 1:1 relationship of IOLD proxy and Device. IOL_CALL instances can be operated without mutual reaction.
- The IOLM proxy supports an additional server entity, which is reserved exclusively for the access of engineering tools such as PDCTs (see [1]). In this case the port number is relevant to address a particular Device.

In case an AL_Read or AL_Write of a port is pending (busy) or a port function is going to be performed, an IOL_CALL request shall be responded with an IOL_ERROR_PDU "*Resource temporarily not available*" (see 10.6.4).

## 10.4 Write On-request Data via IOL_CALL

The detailed sequence to write On-request Data (OD) via IOL_CALL into a Device is specified in this clause. In order to differentiate common terms between PROFINET and IO-Link, the prefixes PN_ and IOL_ are used here, for example PN_Index and IOL_Index.

An IOL_CALL uses the access point of a Device (IOLD proxy) or the Master (IOLM proxy). Access shall be performed via Write/Read services as shown in Table 32. The table specifies the mapping of the service arguments.

Attention should be payed to the fact that ID represents the address of the proxy submodule (e.g. Slot, Subslot) and CAP is represented by the record index. All other variables (WR, Port, IOL_Index, IOL_Subindex are part of the CALL-Header which is mapped to the "data" of the Record (see Figure 28).

**Table 32 – Mapping of IOL_CALL Write arguments to Read/Write services**

| IOL_CALL | Write_req | Read_req | Read_rsp |
|---|---|---|---|
| ID (address proxy) | ID (address proxy) | ID (address proxy) | |
| CAP | PN_Index = 0xB400 | PN_Index = 0xB400 | |
| WR | Data (CALL Header.Control) | | Data (CALL Header.Status) |

| IOL_CALL | Write_req | Read_req | Read_rsp |
|---|---|---|---|
| | See 10.6.2 | | See 10.6.2 |
| Port | Data (CALL Header.Port) | | Data (CALL Header.Port) |
| IOL_Index | Data (CALL Header.IOL_Index) | | Data (CALL Header.IOL_Index) |
| IOL_Subindex | Data (CALL Header.IOL_Subindex) | | Data (CALL Header.IOL_Subindex) |
| IOL_Data | Data (IOL_Data) | | Data (optinal IOL_Error PDU) |
| NOTE Table shows the invocation arguments from a client point of view. | | | |

Figure 26 shows the sequence of services in a diagram of the PROFINET and the IO-Link side and how the data transfer is "tunnelled". "Read_rsp" can result in a correct (OK) or an Error response (IOL_Error_PDU).

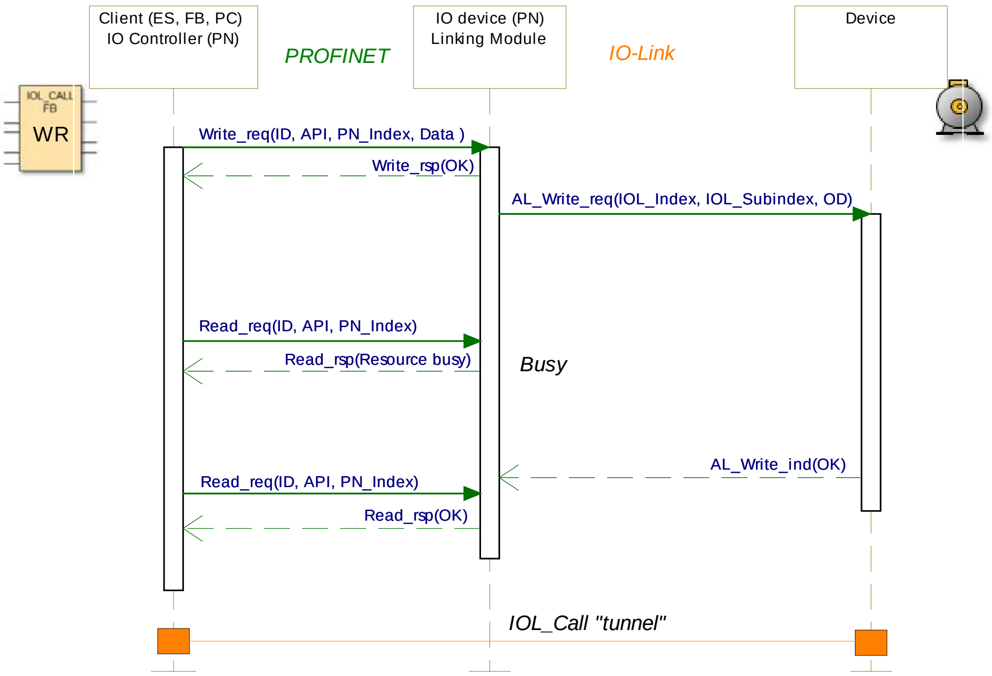



**Figure 26 – IOL_CALL (Write) sequence**

The client opens the sequence with a Write request. The request data are "tunnelled" within the CALL header (see 10.6.2) and lead to an AL_Write request service on the IO-Link side.

The client checks the result via a Read request service in polling manner. A rsp(-) response "*Resource busy – 0x80C2*" indicates that the job is not accomplished yet.

The Read request shall be repeated until a rsp(+) response occurs indicating that the job is done. The CALL header provides then the details.

## 10.5 Read On-request Data via IOL_CALL

The detailed sequence to read On-request Data (OD) via IOL_CALL from a Device is specified in this clause. With respect to terms, the rules in 10.4 apply.

An IOL_CALL uses the access point of a Device (IOLD proxy) or the Master (IOLM proxy). Access shall be performed via Write/Read services as shown in Table 33. The table specifies the mapping of the service arguments.

Attention should be payed to the fact that ID represents the address of the proxy submodule (e.g. Slot, Subslot) and CAP is represented by the record index. All other variables (WR, Port, IOL_Index, IOL_Subindex) are part of the CALL-Header which is mapped to the "data" of the Record (see Figure 28).

**Table 33 – Mapping of IOL_CALL Read arguments to Read/Write services**

| IOL_CALL | Write_req | Read_req | Read_rsp |
|---|---|---|---|
| ID (address proxy) | ID (address proxy) | ID (address proxy) | |
| CAP | PN_Index = 0xB400 | PN_Index = 0xB400 | |
| RD | Data (CALL Header.Control) See 10.6.2 | | Data (CALL Header.Status) See 10.6.2 |
| Port | Data (CALL Header.Port) | | Data (CALL Header.Port) |
| IOL_Index | Data (CALL Header.IOL_Index) | | Data (CALL Header.IOL_Index) |
| IOL_Subindex | Data (CALL Header.IOL_Subindex) | | Data (CALL Header.IOL_Subindex) |
| IOL_Data | – | | Data (IOL_Data or IOL_Error PDU) |
| NOTE Table shows the invocation arguments from a client point of view. | | | |

Figure 27 shows the sequence of services in a diagram of the PROFINET and the IO-Link side and how the data transfer is "tunnelled". "Read_rsp" can result in requested data (IOL_Data) or an Error response (IOL_Error_PDU).

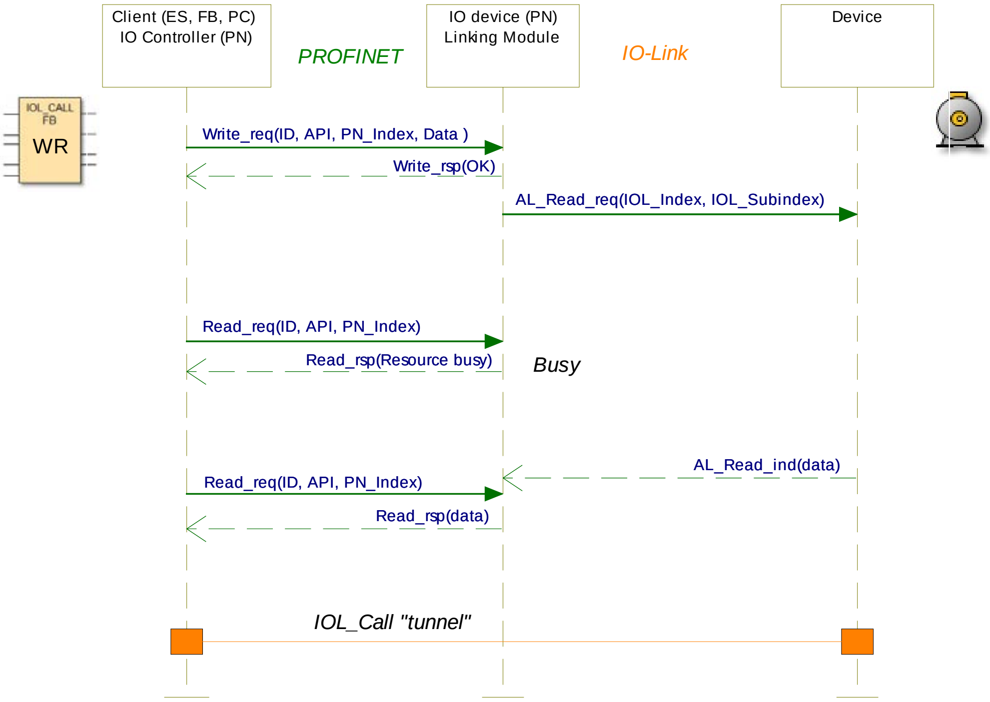



**Figure 27 – IOL_CALL (Read) sequence**

The client opens the sequence with a Write request. The request data are "tunnelled" within the CALL header (see 10.6.2) and lead to an AL_Read request service on the IO-Link side.

The client checks the result via a Read request service in polling manner. A rsp(-) response "*Resource busy – 0x80C2*" indicates that the job is not accomplished yet.

The Read request shall be repeated until a rsp(+) response occurs indicating that the job is done. The CALL header provides then the details. A correct result shall be acknowledged with Status = 0. Errors shall be acknowledged with Status = *IOL_ERROR_PDU*.

## 10.6 IOL_CALL protocol

### 10.6.1 General

The complexity of IOL_CALL requires a protocol defined by Client state machines and Server state machines as well as control and status information for the coordination of both. The server state machine is located within the Linking Module.

### 10.6.2 CALL Header coding

The IOL_CALL Write and Read services contain each a so-called CALL Header to allow for managing the protocol. The CALL Header consists of 8 octets and is part of the Data (see Table 33 and Table 42).

Table 34 shows the CALL Header coding.

**Table 34 – CALL Header coding**

| Parameter | Coding | Definition | Data type |
|---|---|---|---|
| Function | 0x08 | Indicates Call Header | Unsigned8 |
| Port | 0 to 255 | Port number | Unsigned8 |
| FI_Index | 0xFE4A | Fixed (compatibility) | Unsigned16 |
| Control/Status | See Table 35, Table 36 | Control or Status octet | Unsigned8 |
| IOL_Index | 0 to 32767;<br>65535 | IOLD data index;<br>Port function | Unsigned16 |
| IOL_SubIndex | 0 to 255 | IOLD data Subindex or CommandCode | Unsigned8 |

IOL_CALL Write carries a parameter "Control" within the CALL Header to manage for example the direction Write/Read or an abort of the IOL_CALL.

The response carries a parameter "Status" within the CALL Header to manage for example error cases.

Table 35 shows the coding of the Control parameter.

**Table 35 – Coding of Control parameter**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Definition of Control octets |
|---|---|---|---|---|---|---|---|---|
| 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 On-request Data or Port function |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Read On-request Data |
| Other codings | | | | | | | | Reserved |

Table 36 shows the coding of the Status parameter.

**Table 36 – Coding of Status parameter**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Definition of Status octets |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Done / Transfer terminated |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | IDLE Sequence |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | IOL_Error PDU |
| Other codings | | | | | | | | Reserved |

### 10.6.3 IOL_CALL data mapping

The mapping of IOL_CALL Write and IOL_CALL Read into AL_Write and AL_Read is shown in Figure 28. Data of an IOL_CALL Write are transferred via AL-Write, data of an IOL_CALL Read via AL_Read.

As soon as an error has been detected, its coding shall be transferred in an IOL_Error_PDU. The occurrence shall be indicated as Status "0x80" within the CALL Header.

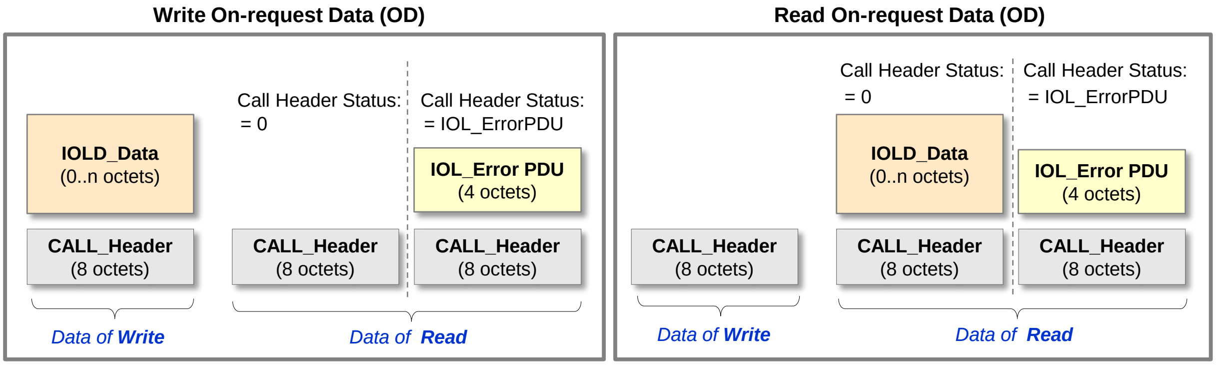



**Figure 28 – IOL_CALL data mapping**

### 10.6.4 Coding of the IOL_Error_PDU

Table 37 shows the coding of the IOL_Error_PDU.

Derived ErrorTypes (ErrorCode, Additional Code) are generated in the Master AL and are caused by internal incidents or those received from the Device. Table C.2 in [1] lists the specified Derived ErrorTypes.

**Table 37 – Coding of the IOL_Error_PDU**

| Offset | Parameter | Content | Data type |
|---|---|---|---|
| 0 | Port Error | Error Codes detected by the Linking Module or Client (see Table 38) | Unsigned16 |
| 2 | Error Code | IO-Link Error codes according AL_Read/ AL_Write services; see [1]. | Unsigned8 |
| 3 | Additional Code | IO-Link Error codes according AL_Read/ AL_Write services; see [1]. | Unsigned8 |

Table 38 shows the port error coding. Port errors shall be generated by the Linking Module.

**Table 38 – Port error coding**

| Port Error Code | Definition | Coding | Originator |
|---|---|---|---|
| No error | No error detected | 0x0000 | Server |
| Reserved | – | 0x0001 to 0x06FFF | – |
| IOL_CALL conflict | Inconsistent Header information | 0x7000 | Server and/ or Client |
| Incorrect IOL_CALL | Inconsistent Header information (send-/response) | 0x7001 | Server and/ or Client |
| Port blocked | Port temporary not available | 0x7002 | Server |
| Reserved | – | 0x7003 to 0x7FFF | – |
| Timeout | No correct termination of IOL_CALL (Resource Busy detection) | 0x8000 | Client |
| Invalid port number | Invalid port Number or port not | 0x8001 | Client and/ or |

| Port Error Code | Definition | Coding | Originator |
|---|---|---|---|
| | supported | | Server |
| Invalid IOL_Index | Invalid Index | 0x8002 | Client |
| Invalid IOL_Subindex | Invalid Subindex | 0x8003 | Client |
| No Device | No device | 0x8004 | Client |
| Reserved | – | 0x8005 to 0x8051 | – |
| RDREC Fault | Fault during Read record invocation | 0x8052 | Client |
| WRREC Fault | Fault during Write record invocation | 0x8053 | Client |
| Unexpected Error | Unspecific Error detected | 0x8054 | Client |
| Port Function error | Port function failed | 0x8055 | Server |
| Port Function not available | Port function is not available (in this state) | 0x8056 | Server |
| Port Function not supported | Port function (for this port) not supported | 0x8057 | Server |
| Manu | Manufacturer specific | 0x8058 to 0xFFFF | Server |

### 10.6.5 Timeout behavior

Each client entity maintains a watchdog timer to monitor the IOL_CALL sequence. The time duration is manufacturer specific but always > 10 s. It is not necessary for a server entity to maintain a watchdog timer.

The server shall be designed such that a task, for the sake of robustness, can always be restarted by another task PDU even if the current task has not been completed yet.

### 10.6.6 Sequence error detection

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 can be caused by several client entities trying to communicate with the same server entity through one client access point ("cap").

- Two or more clients are trying to communicate with a server entity through one client access point ("cap") at one point in time. The server will start processing one of them (Write) and usually will wait on a Read but receives the Write of a second client. No matter how this (very unlikely) conflict is resolved, the results can be unsatisfactory. The server entity accepts only to process the newest ("last") IOL_CALL request. No error message occurs.
- Read without a client's Write: The server entity responds with the Read service rsp(-) Error coding "*State conflict – 0x80B5*".
- Two or more correct Read services after a Write service (task PDU): The server entity responds with the Read service rsp(-) Error coding "*State conflict – 0x80B5*".

### 10.6.7 Client state machine

The finite state machine of a client as shown exemplary for a PLC function block in Figure 55 is characterized by four states and their responsibilities. It is generic enough to be used in other client applications such as a PDCT tool. A new IOL-CALL task can be started in state 1 (Idle) via variable REQ = 1. The tasks can be aborted via a client timeout within states 2 or 3 for example if a server fails.

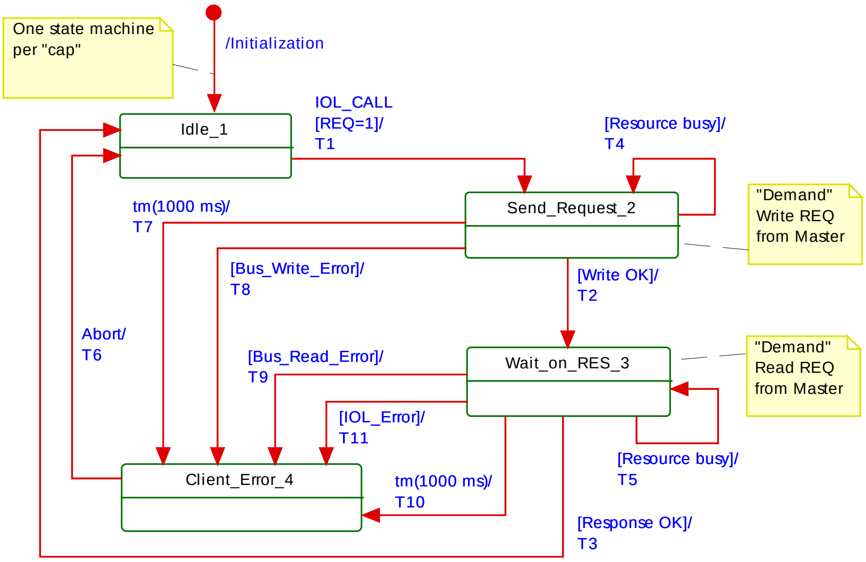



**Figure 29 – State diagram of an IOL_CALL client**

Table 39 lists the definition of terms of Figure 29.

**Table 39 – Definition of terms of Figure 29**

| Term | Definition |
|---|---|
| [REQ =1] | Guard: User program launches the IOL_CALL function |
| [Write OK] | Guard: Write service has been carried out correctly |
| [Bus_Write_Error] | Guard: Write service failed (for error codes see Table 38) |
| [Bus_Read_Error] | Guard: Read service failed (for error codes see Table 38) |
| [IOL_Error] | Guard: IOL services failed (for error codes see Table 38) |
| [Response OK] | Guard: Services have been carried out correctly; no RES PDU with error codes |
| tm(1000 ms) | IOL_CALL timeout after 1s |

States and transitions are defined in detail in Table 40.

**Table 40 – State transition table of a client**

| STATE NAME | STATE DESCRIPTION |
|---|---|
| Idle_1 | Initialization already completed after startup. Idle state, no activities. Reset and de-activate the client watchdog timer. Waiting on IOL_CALL request [REQ =1]. |
| Send_Request_2 | IOL_CALL request was sent via Write PDU to the server through a "cap". At the same time the watchdog timer shall be started. Server performs task (IOLD proxy or port function). Waiting on rsp(+) PDU (OK) or rsp(-) PDU (error). |
| Wait_on_RES_3 | Read PDU to acquire the server response through a "cap". Read PDUs to be repeated, when "Resource busy" (0x80C2) is returned (Table 38). A consistency check is per-formed between both the Headers of the IOL_CALL_REQ PDU and the IOL_CALL_RES PDU. Waiting on on rsp(+) PDU (OK) or rsp(-) PDU (error). |
| Client_Error_4 | Indicates a "faulty“abort of the protocol and stops the client watchdog timer. Indicates error. Sets 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:<br>DONE_VALID =0, BUSY =1, ERROR =0, STATUS =-1, IOL_STATUS =-1, LEN =0. |

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| T3 | 3 | 1 | Received Read-RES PDU:<br>Stop timer; DONE_VALID =1, BUSY =0, ERROR =0, STATUS =0, IOL_STATUS =-1, LEN =record length.<br>Invocation |
| T4 | 4 | 4 | Received Write-NRS PDU with "Resource busy" (=0xC2 from Table 38):<br>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 38):<br>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.<br>Invocation |
| T7 | 2 | 4 | Did not receive Write-RES PDU or Write-NRS PDU in time:<br>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:<br>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:<br>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:<br>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:<br>Stop timer; DONE_VALID =0, BUSY =0, ERROR =1, STATUS =0, IOL_STATUS =error code, LEN =0. |

| INTERNAL ITEMS | TYPE | DEFINITION |
|---|---|---|
| Timer | Variable | Client activities are monitored by a watchdog timer |
| Error_Code | Variable | In case of errors or failures an ErrorCode is returned. Possible errors: Table 38 |
| Resource busy | Guard | If IOLM proxy cannot perform task: Table 38 |

### 10.6.8 Server state machine

The finite state machine of the server is presented in Figure 30. The server maintains a watchdog timer for the services. The finite state machine can always be restarted by an incoming new IOL_CALL from the same or another client. All current processes of the associated port will then be aborted.

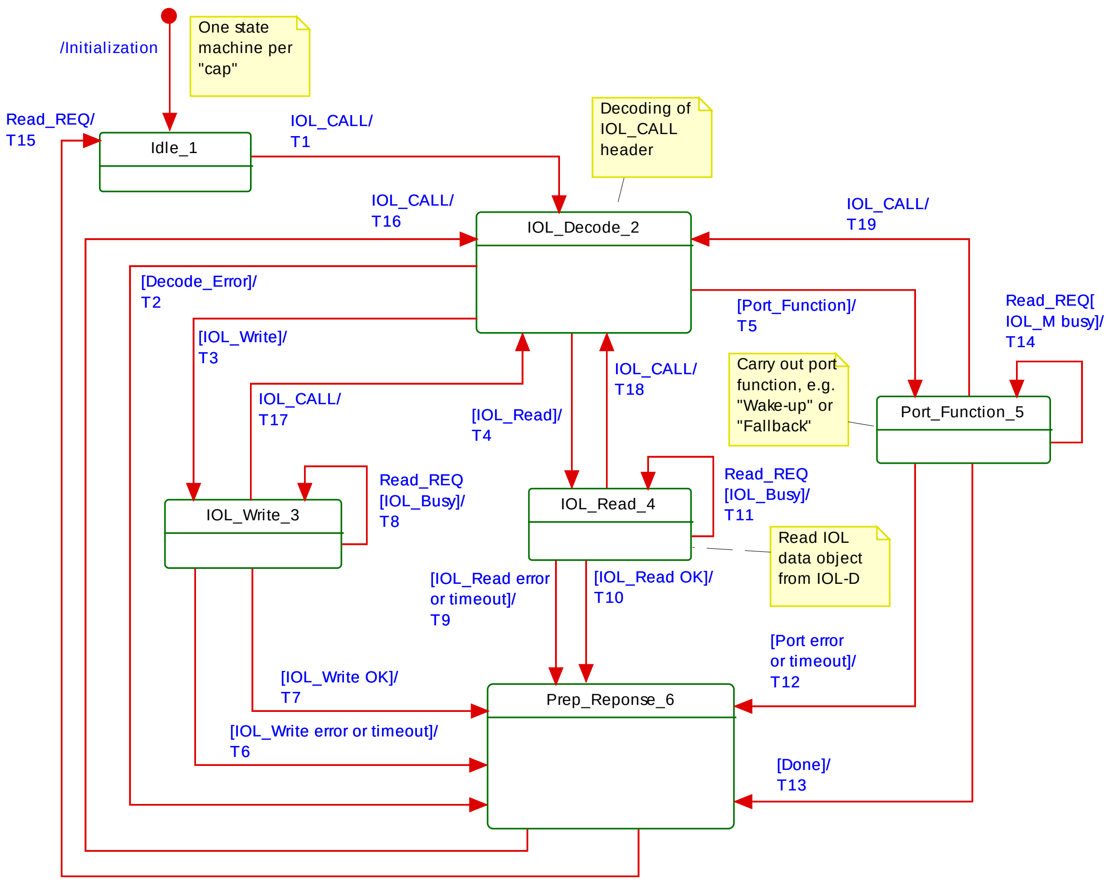



**Figure 30 – State diagram of an IOL_CALL server**

The terms used in Figure 30 are defined within the "Internal Items" section of Table 41.

**Table 41 – State transition table of a server**

| STATE NAME | STATE DESCRIPTION |
|---|---|
| Idle_1 | Idle state, no activities |
| IOL_Decode_2 | IOL_CALL request received either from the state 1 or any other. Decoding and checking for correctness of the IOL-CALL Header. |
| IOL_Write_3 | Write service to the IOLD proxy to be performed (address: IOL_Index, IOL_Subindex) as activity according to the flowchart in Figure 31.<br>Unexpected new IOL_CALL will cause the transition to the state 2. All other processes are aborted. |
| OL_Read_4 | Read service to the IOLD proxy to be performed (address: IOL_Index, IOL_Subindex) as activity similar to the flowchart in Figure 31.<br>Unexpected new IOL_CALL will cause the transition to the state 2. All other processes are aborted. |
| Port_Function_5 | 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 31. It is assumed that time monitoring is provided by the generic IOLM proxy layer.<br>Unexpected new IOL_CALL will cause the transition to the state 2. All other processes are aborted. |
| Prep_Response_6 | Prepare either rsp(-) PDU in case of error (incorporate error codes) or rsp(+) PDU with or without data. Waiting on next Read PDU. |

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| T1 | 1 | 2 | - |
| T2 | 2 | 6 | Return Write PDU and prepare error code for rsp(-) PDU "Incorrect IOL_CALL" |

| TRAN-SITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| 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 | | IOLD proxy did not accomplish a task from a particular port yet |
| IOL_CALL | | Write or read request for an IOLD proxy or an IOLM port |
| IOL-M_Busy | | IOLM proxy did not accomplish a task yet |
| IOL_Read | | Task to read IOL data from an IOLD proxy |
| IOL_Read Error | | Error occurred while reading IOL data from an IOLD proxy |
| IOL_Read OK | | Read task accomplished successfully |
| IOL_Write | | Task to write IOL data to an IOLD proxy |
| IOL_Write Error | | Error occurred while writing IOL data to an IOLD proxy |
| IOL_Write OK | | Write task accomplished successfully |
| Port_Error | | Error occurred while performing port functions |
| Port_Function | | Task to perform 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 IOLM proxy layer defined in [1]. 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 Device connected to the port. |

Principle of activity in "IOL_WRITE" state 3 is shown in Figure 31. T3, T6, T7, T8, and T17 refer to the transitions in Figure 30.

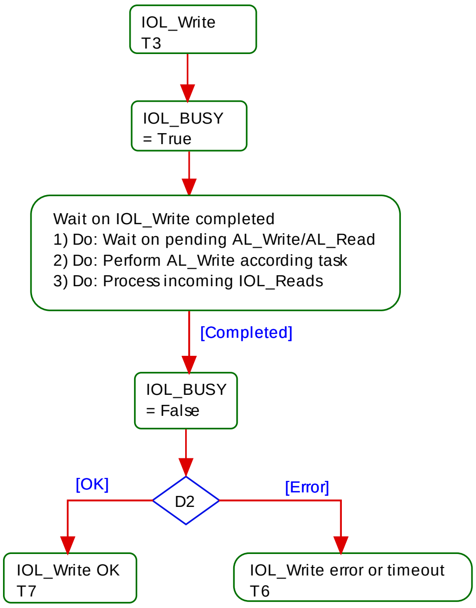



**Figure 31 – Activity in state 3 "IOL_Write"**

## 10.7 Port function via IOL_CALL

### 10.7.1 IOL_CALL arguments

With the help of IOL_CALL additional port functions can be invoked such as Wakeup or Fallback. Corresponding CommandCodes (CC) are specified in 10.7.2.

In this case, the IOL_CALL uses the access point of the Master (IOLM proxy). Access shall be performed via Write/Read services as shown in Table 42. The table specifies the mapping of the service arguments.

**Table 42 – Mapping of IOL_CALL port arguments to Read/Write services**

| IOL_CALL | Write_req | Read_req | Read_rsp |
|---|---|---|---|
| ID (IOLM proxy) | ID (IOLM proxy) | ID (IOLM proxy) | |
| CAP | PN_Index = 0xB400 | PN_Index = 0xB400 | |
| WR | Data (CALL Header.Control)<br>See 10.6.2 | | Data (CALL Header.Status)<br>See 10.6.2 |
| Port | Data (CALL Header.Port) | | Data (CALL Header.Port) |
| IOL_Index | Data (CALL Header.IOL_Index = 0xFFFF) | | Data (CALL Header.IOL_Index = 0xFFFF) |
| IOL_Subindex | Data (CALL Header.IOL_Subindex = CC) | | Data (CALL Header.IOL_Subindex = CC) |
| IOL_Data | – | | C: Data (optional IOL_Error PDU) |

Figure 32 shows the sequence of services for a port function.

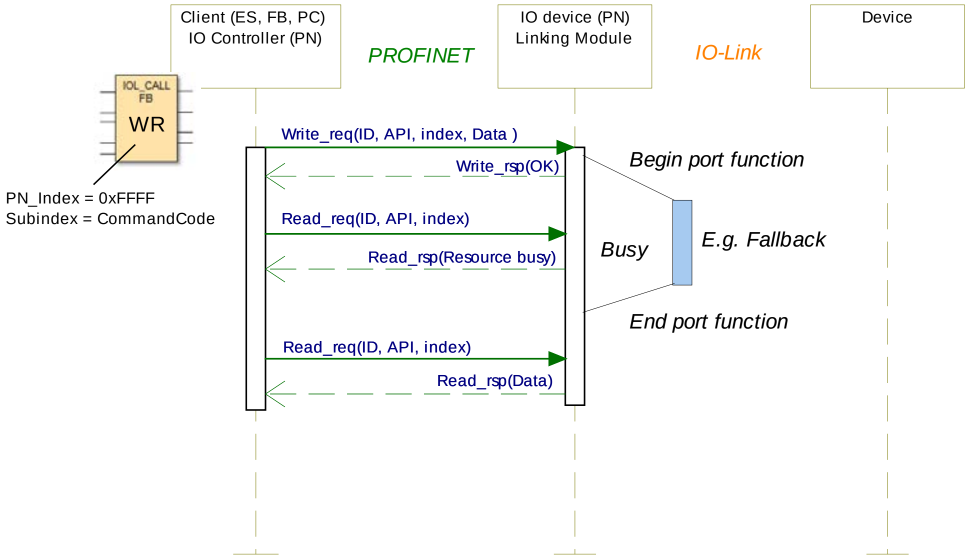



**Figure 32 – IOL_CALL (port function) sequence**

The client opens the sequence with a Write request. The IOL_Index "0xFFFF" within the argument indicates a port function. The IOL_Subindex indicates the CommandCode (CC).

The client checks the result via a Read request service in polling manner. A rsp(-) response "*Resource busy*" indicates that the job is not accomplished yet.

The Read request shall be repeated until a rsp(+) response occurs indicating that the job is done. The CALL header provides then the details. A correct result shall be acknowledged with Status = 0. Errors shall be acknowledged with Status = *IOL_ERROR_PDU.*

### 10.7.2 CommandCodes

Table 43 lists the specified CommandCodes (CC) for port functions.

**Table 43 – CommandCodes for port functions**

| CommandCode | Coding (decimal) | Definition |
|---|---|---|
| Reserved | 1 | – |
| Reserved | 2 | – |
| Deactivate Port | 3 | Automatic Tool Changer applications (see 9.2.3) |
| Activate Port | 4 | Automatic Tool Changer applications (see 9.2.4) |
| Reserved | 5 to 63 | – |
| Manufacturer specifc | 64 to 255 | – |

## 11 Engineering aspects

### 11.1 User view

This document provides two principle methods for commissioning of an IO-Link Master system on PROFINET depending on user's paradigm: a one-tier (solely PROFINET centric) or a two-tier (PROFINET and IO-Link) procedure model.

*One-tier commissioning (GSD based)*

The entire PROFINET system together with the subsidiary IO-Link system are configured and parameterized via the major engineering system using extended GSD files. The IO-Link system specifics are somewhat hidden and it is more or less treated like a remote IO terminal. No separate IO-Link tool is required. This procedure model is simple and coherent. However, it is limited and does not support the power and flexibility of IO-Link.

*Two-tier commissioning (GSD and PDCT based)*

Maximum power and flexibility can be achieved for configuration, parameterization, and commissioning when using a dedicated PDCT for the IO-Link system together with the Device IODDs after the PROFINET configuration.

Precondition for this procedure model is a port preset to "Tool based (PDCT) configuration". This allows for setting up the port and parameterization of the Device with the help of its IODD.

## 11.2 Port configuration modes

The user can choose between three top level port modes during PROFINET commissioning as shown in Table 44. This assignment takes place when parameterizing the Submodule (IOLD proxy).

**Table 44 – Top level port modes**

| Port mode | Definition | Use case |
|---|---|---|
| Autoconfig | "Plug & play": No explicit port configuration. Basic assignments such as Inspection level, port cycle time, VendorID and DeviceID are not required. | The user connects an offsite preconfigured and preparameterized Device to the port. Start-up of the Device without any additional actions. No verification possible. |
| GSD-based | One-tier commissioning: Explicit port configuration possible for Inspection level, port cycle time, VendorID and DeviceID. These parameters are GSD-based and displayed within the PROFINET engineering system for value entry. | The user enters values for port parameters prior to connecting an offsite preconfigured and preparameterized Device to the port. Verification takes place at start-up: Connected Device corresponds to projected Device? |
| Tool-based | Two-tier commissioning: No explicit port configuration within PROFINET engineering system. Port configuration and Device parameterization can be performed with the help of a Port and Device Configuration Tool (PDCT), sometimes called "Master-Tool", communicating with Master and Device. | After PROFINET engineering of the Submodule, the user launches the PDCT, then configures the port and parameterizes the connected Device online. Use of offsite preconfigured and preparameterized Device is possible. |

Table 44 provides an overview of supported features at chosen port configurations.

**Table 45 – Supported features at chosen port configurations**

| Feature | Autoconfig | GSD-based | Tool-based |
|---|---|---|---|
| Access on Process Data (PD) | ✓ | ✓ | ✓ |
| Diagnosis of port & Device | ✓ | ✓ | ✓ |
| I&M data (IM0) access | ✓ | ✓ | ✓ |
| Device check (consolidated/real) | No | ✓ | ✓ |
| Backup & Restore | No | ✓ | ✓ |
| Device parameterization (IOL_CALL) | ✓ | ✓ | ✓ |
| Device parameterization (Tool) | No | No | ✓ |
| Commissioning (online) | No | No | ✓ |

## 11.3 GSD/GSDML

### 11.3.1 Overview

GSD is the device integration standard for PROFINET. It is mandatory for engineering systems to support the usage of GSD files for PROFINET devices based on the GSDML language. Therefore, PROFINET devices with integrated IO-Link Master systems shall provide a description of the corresponding "Linking Module" within the GSD file.

In the following, the necessary means to develop this part of a GSD file are specified. Due to customer requirements it is highly recommended to care for a uniform "look and feel" besides specified common features during the design of the GSD.

### 11.3.2 GSD template

PI provides a "GSD template for IO-Link" on its website for download [9]. It contains the relevant elements for the integration of "Linking Modules". It is highly recommended for manufacturers/vendors of IO-Link Submodules (IOLD proxies) to adopt these XML based code models concerning the "Linking Module" approach into their GSD files.

### 11.3.3 IOLD_proxy submodule

The description of a "Linking Module" comprises a dedicated number of Submodule descriptions according to the proxy concept. Such an IOLD proxy type is the representative of a particular port and its connected Device.

Figure 33 shows exemplary the Device Catalog within the GUI of an engineering system. It contains besides the general part for "IOL Devices" three categories of IOLD proxies that are described in detail in the following: Digital Input/Output, IOL generic Devices, and IOL Profile Devices. The nomenclature is standardized and shown in the template.

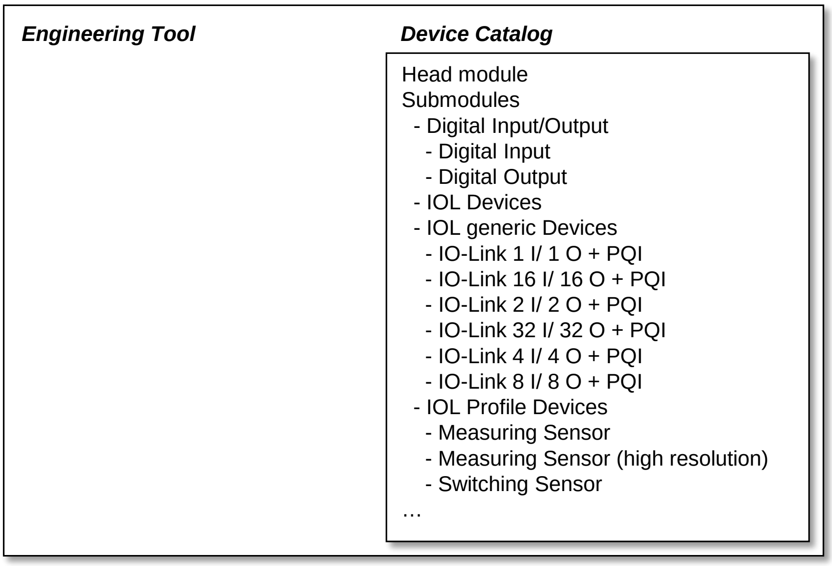



**Figure 33 – Submodule descriptions**

*Digital Input/Output*

Description of the Submodule types Digital Input and Digital Output. The user can configure a port into SIO modes "DI" or "DO".

*IOL generic Devices*

This category provides descriptions of a number of Submodules, which currently differ only in the number of Inputs and Outputs for Process Data (PD). The user can choose a Submodule according to the I/O requirements of the intended Device. The data type for the IO data is neutralized as *Octet String*.

The nomenclature for the proxy Submodule is: IO-Link $n$ I/$m$ O + PQI. "$n$" stands for the number of Input octets and "$m$" for the number of Output octets.

The following XML code shows an excerpt of the GSD coding of a 4I/4O Submodule.

```xml
<!-- submodule IO-Link 4 I/4 O + PQI -->
<SubmoduleItem API="19969" ID="IDS_1 Port x IO-Link 4I/4O" SubmoduleIdentNumber="0x00000405" Re-
quiredSchemaVersion="V2.31">
 <IOData IOPS_Length="1" IOCS_Length="1">
  <Input Consistency="All items consistency">
   <DataItem DataType="OctetString" TextId="Input_Data_4" Length="4" UseAsBits="false" />
   <DataItem DataType="Unsigned8" UseAsBits="true" TextId="PQI">
      <BitDataItem BitOffset="2" TextId="NP" />
      <BitDataItem BitOffset="3" TextId="SV" />
      <BitDataItem BitOffset="4" TextId="DA" />
      <BitDataItem BitOffset="5" TextId="DC" />
      <BitDataItem BitOffset="6" TextId="DE" />
      <BitDataItem BitOffset="7" TextId="PQ" />
    </DataItem>
  </Input>
  <Output Consistency="All items consistency">
   <DataItem DataType="OctetString" TextId="Output_Data_4" Length="4" UseAsBits="false" />
  </Output>
 </IOData>
```

*IOL Profile Devices*

This category provides descriptions of Submodules representing IO-Link Profile Devices such as the Smart Sensor Profile. The descriptions comprise profile specific IO interfaces (data types) and Device parameters (start-up parameters). One of the purposes of Profile Devices is prevention from usage of a PDCT.

### 11.3.4 Port configuration (port parameters)

The GSD Submodule description contains a standardized part for the port configuration parameters within the IOLD proxy module. The PN-Index is 0xB900. The layout corresponds to the PortConfiguration record in 11.4.3. Figure 34 shows an example of a port configuration dialog.

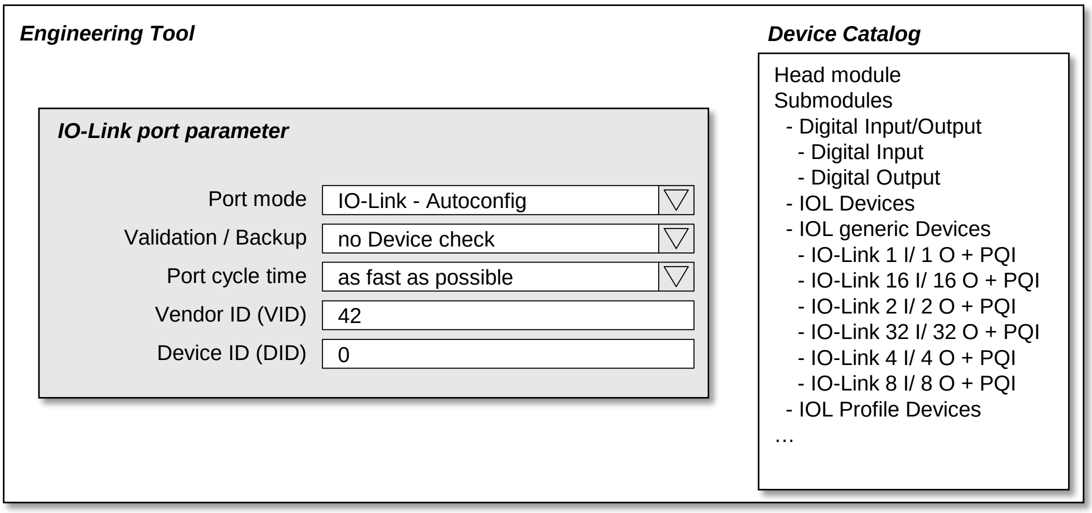



**Figure 34 – Port configuration dialog example**

The following XML code shows an excerpt of the GSD coding of a port configuration.

```xml
<!-- Profile Index=0xB900 (47360)-->
 <ParameterRecordDataItem Index="47360" Length="15" TransferSequence="0">
   <Name TextId="Port parameters" />
   <Const ByteOffset="0" Data="0x00,0X00,0x09,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x05,
0x04" />
   <Ref ValueItemTarget="Diagnostic release" DataType="Bit" ByteOffset="4" BitOffset="0" DefaultValue="0"
Changeable="true" Visible="true" TextId="Diagnostic release port" />
```

```
    <Ref ValueItemTarget="Diagnostic release" DataType="Bit" ByteOffset="4" BitOffset="1" DefaultValue="0" Changeable="true" Visible="true" TextId="Notification" />
    <Ref ValueItemTarget="Startup" DataType="BitArea" ByteOffset="4" BitOffset="2" BitLength="2" DefaultValue="0" AllowedValues="0 1 2" TextId="StartUp_Mode" Changeable="true" />
    <Ref ValueItemTarget="I_Level" DataType="BitArea" ByteOffset="5" BitOffset="0" BitLength="3" DefaultValue="0" AllowedValues="0 1 2 3" TextId="InspectionLevel" Changeable="true" />
    <Ref ValueItemTarget="M_Cycle" DataType="BitArea" ByteOffset="12" BitOffset="0" BitLength="8" DefaultValue="0" AllowedValues="0 5 10 20 40 68 88 128 148 188" TextId="CycleTime" />
    <Ref DataType="Unsigned16" ByteOffset="6" BitOffset="0" DefaultValue="42" AllowedValues="0..65535" TextId="VendorID" Changeable="true" Visible="true" />
    <Ref DataType="Unsigned32" ByteOffset="8" BitOffset="0" DefaultValue="0" AllowedValues="0..4294967295" TextId="DeviceID" Changeable="true" Visible="true" />
  </ParameterRecordDataItem>
```

## 11.4 Port and Device configuration tool (PDCT)

### 11.4.1 General

Figure 35 shows the center role of the PDCT tool with respect to the IO-Link system. It is used to configure ports, parameterize Devices, display diagnosis information, and provide identification and maintenance information. In case a manufacturer/vendor of an IO-Link Gateway provides a generic Port and Device configuration Tool, the interfaces shown in Figure 35 shall be supported.

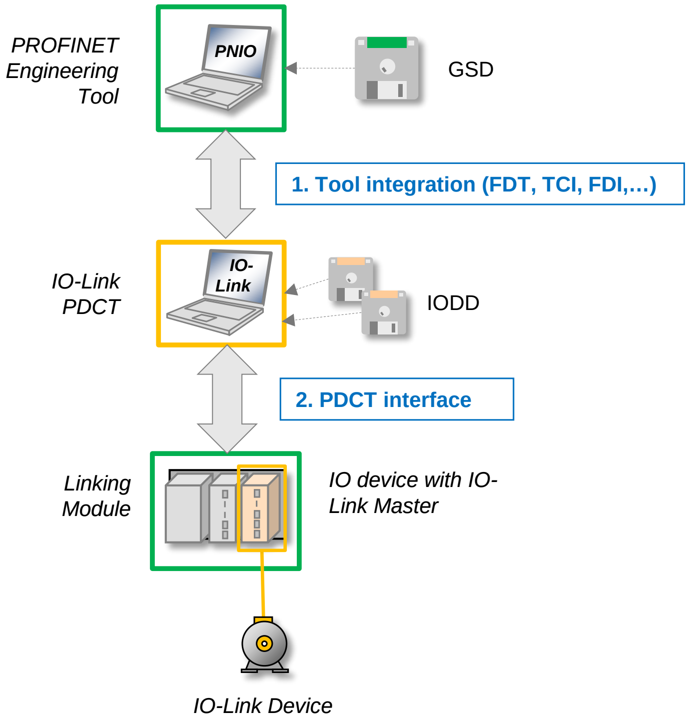



**Figure 35 – Overview of PDCT integration**

PDCT allows for full access to the IO-Link Master, its ports and the connected Devices.

Two points of concern exist with respect to tool integration. One is the integration of PDCT in PROFINET engineering systems as described in [10]. The second is a standardized PDCT interface as described in 11.6, which allows interactions between PDCT and IO-Link Master systems of different brands.

Concept here is the use of the IOLM proxy access point for all functions. The corresponding Submodule (IOLM proxy) shall always be implemented and available even if all Devices are unplugged.

### 11.4.2 Port configuration

PDCT provides the following services:

- Assignment of a Device from the IODD catalog/library to a Master port
- Assignment of port parameters such as port modes, Inspection Level, cycle times, etc.

The port configuration can be downloaded to the Linking Module (Master). It shall be stored per port in non-volatile memory.

### 11.4.3 PortConfiguration record and download

The port configuration information shall be handled in a record. Every IOLM proxy shall support these records per port. The records are readable and writeable.

The addressing of these records is maintained using the formula (1).

$$Index(PortConfiguration) = StartIndex(PortConfiguration) + PortNumber \qquad (1)$$

Example: Index for PortNumber 5: 0xB100 + 5 = 0xB105

Table 46 specifies the coding of the PortConfiguration record.

**Table 46 – Coding of PortConfiguration record**

| Offset | Parameter name | Definition | Data type |
|---|---|---|---|
| 0 | BlockVersionHigh | Versioning of record; first version: 0x01 | Unsigned8 |
| 1 | BlockVersionLow | Versioning of record; first version: 0x00 | Unsigned8 |
| 2 | Reserved | – | Unsigned16 |
| 4 | PortConfigControl | Bit 0: Enable Port Diagnosis<br>Bit 1: Enable Process Alarm (Device notification)<br>Bit 2,3,4: Port Mode:<br>0: IOL-Autoconfig<br>1: IOL-Manual<br>2: IOL-Tool based<br>3: Digital Input (Pin 4)<br>4: Digital Output (Pin 4)<br>5 to 7: Reserved<br>Bit 5: reserved<br>Bit 6: Enable Input fraction<br>Bit 7: Enable Pull/Plug | Unsigned8 |
| 5 | Validation & Backup | 0: no Device check<br>1: type compatible Device (V1.0)<br>2: type compatible Device (V1.1)<br>3: type compatible Device (V1.1) with Backup + Restore<br>4: type compatible Device (V1.1) with Restore<br>5 to 255: reserved | Unsigned8 |
| 6 | VendorID | Expected IO-Link Device VendorID | Unsigned16 |
| 8 | DeviceID | Expected IO-Link DeviceID | Unsigned32 |
| 12 | PortCycleTime | 0: as fast as possible<br>16: 1,6 ms<br>32: 3,2 ms<br>48: 4,8 ms<br>68: 8,0 ms<br>100: 20,8 ms<br>133: 40,0 ms<br>158: 80,0 ms<br>183: 120,0 ms<br>Coding is derived from [1]. User can provide a selection of codings (see example) | Unsigned8 |
| 13 | Input length | Input Data 0 to 31 of Device (IODD) | Unsigned8 |
| 14 | Output length | Output Data 0 to 31 of Device (IODD) | Unsigned8 |
| NOTE1 Data types comply with IEC 61158-6-10<br>NOTE2 Input / Output Length: including PQI | | | |

Figure 36 demonstrates the download of the PortConfiguration record to Index 0xB925 within the IOLM proxy. IOLM proxy initiates a restart of port 5. The successful startup or an error can be acquired via a Read of the PortStatus record from Index 0xB205.

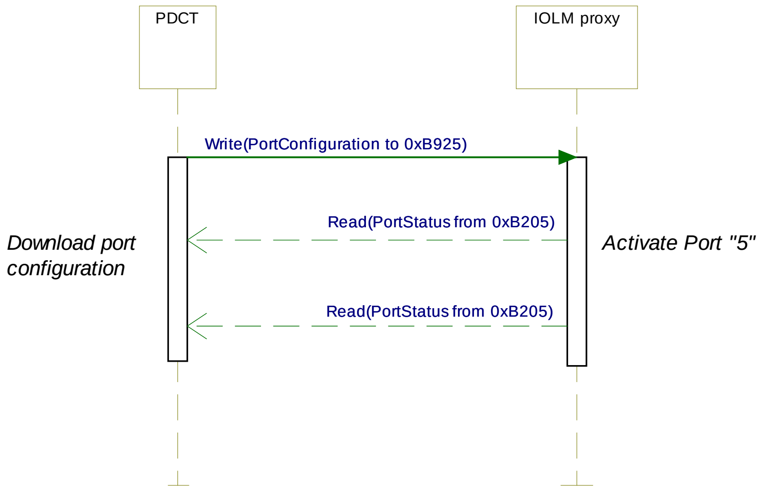



**Figure 36 – Download of port configuration**

### 11.4.4 Device parameterization

The PDCT allows for Device parameterization based on IODD Device descriptions. The Device parameters are downloaded to the Device with the help of IOL_CALL functions and subsequent ISDU operations. Parameter instance values are stored on the Device in a non-volatile memory. The IOL_CALL uses the IOLM proxy as access point as shown in Figure 37. Thereby, Index 0xB980 serves as CAP. The port number is carried within the Header of the IOL_CALL. See 7.9.1 for batch objects.

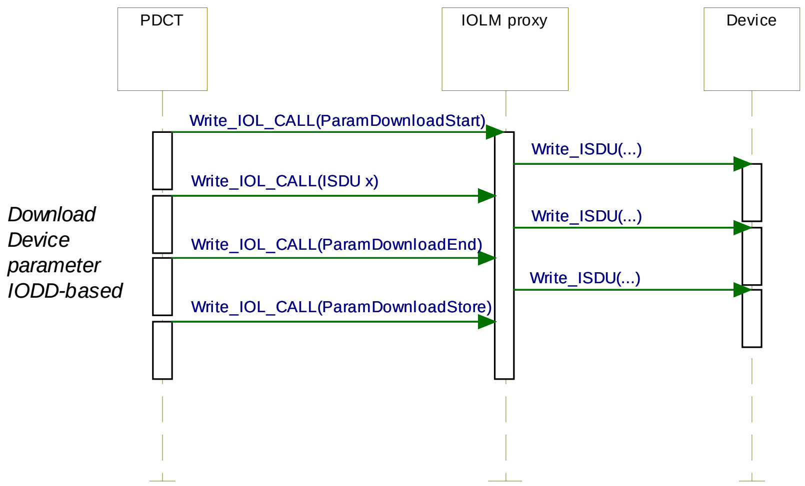



**Figure 37 – Download of Device parameters**

### 11.4.5 Port Status and Port diagnosis online

PDCT allows for comfortable commissioning and maintenance of Devices. One is the possibility to read out the current PortStatus record of each single port. Every IOLM proxy shall support these records.

The addressing of these records is maintained using the formula (2).

$$Index(PortStatus) = StartIndex(PortStatus) + PortNumber \quad (2)$$

Example: Index for PortNumber 2: 0xB200 + 2 = 0xB202

These records show the updated status of the ports. In addition, pending IO-Link diagnosis (Events) can be monitored. The records are only readable. Table 47 shows the coding.

**Table 47 – Coding of PortStatus record**

| Offset | Parameter name | Definition | Data type |
|---|---|---|---|
| 0 | BlockVersionHigh | Versioning of record | Unsigned8 |
| 1 | BlockVersionLow | Versioning of record | Unsigned8 |
| 2 | PortNumber | Value range 0 to 255 | Unsigned8 |
| 3 | PortStatusInfo | 0: Port Running (IO-Link)<br>1: DI<br>2: DO<br>3: Deactivated<br>4: No Device (Communication fault)<br>5: Incorrect Device (Inspection Level)<br>6: Fault<br>7 to 100: Reserved<br>101 to 254: Vendor specific<br>255: Info temporarily not available (try again) | Unsigned8 |
| 4 | PQI (Port Qualifier) | See Table 10 (Definition of flag bits)<br>Coding 255: no Port Qualifier available | Unsigned8 |
| 5 | PortStatusFlags | Bit 0: PDinvalid<br>Bit 1: PDoutValid<br>Bit 2 to 7: Reserved | Unsigned16 |
| 6 | FieldbusStatus | Bit 0: IOPS state (Input)<br>Bit 1: IOPS state (Output)<br>Bit 2: Port configuration with PDCT<br>Bit 3 to 6: Reserved<br>Bit 7: ISDUBatch Pending | Unsigned8 |
| 7 | RevisionID | 0x10: Version V1.0<br>0x11: Version V1.1 | Unsigned8 |
| 8 | Transmission rate | 0: No communication<br>1: COM1<br>2: COM2<br>3: COM3<br>4 to 255: Reserved | Unsigned8 |
| 9 | MasterCycleTime | MasterCycle Time | Unsigned8 |
| 10 | VendorID | Real IO-Link VendorID | Unsigned16 |
| 12 | DeviceID | Real IO-Link DeviceID | Unsigned32 |
| 16 | EventEntries | Shows the number of EventEntires (1..n) | Unsigned8 |
| 17 to 19 | EventEntry1 | Consists of IOL "EventQualifier" and "EventCode" | Struct Unsigned8/16 |
| 20 to 22 | EventEntry2 | Consists of IOL "EventQualifier" and "EventCode" | Struct Unsigned8/16 |
| n | EventEntry3 | Consists of IOL "EventQualifier" and "EventCode" | Struct Unsigned8/16 |
| n+1 | ProfileEntries | Number of Entries (1..n) in ProfileCharacteristics | Unsigned8 |
| n+2 | ProfileEntry1 | First ProfileID | Unsigned16 |
| n+3 | ProfileEntry2 | SecondProfileID | Unsigned16 |

Parameter "EventEntries" indicates the supported number of Event entries. The EventEntries contain codes of appearing Events and thus show the current status. Coding follows "DetailedDeviceStatus" in [1]. A code 0x000000 indicates no pending Events.

### 11.4.6 ProcessData monitoring online

This record is only readable and contains the transferred IO-Link Process Data (PD). The record is a direct copy of the PROFINET Input/Output data buffer. The PQI information is also part of the IOL Process Data record. The record can be acquired for a particular port from the IOLM proxy as shown in Figure 38. Table 48 shows the coding of the Process Data record.

The addressing of these records is maintained using the formula **(3)**.

$$Index(IOLprocessData) = StartIndex(IOLprocessData) + PortNumber \qquad (3)$$

Example: Index for PortNumber 8: 0xB300 + 8 = 0xB308

**Table 48 – Coding of Process Data record (Read)**

| Offset | Parameter name | Definition | Data type |
|---|---|---|---|
| 0 | BlockVersionHigh | Versioning of record | Unsigned8 |
| 1 | BlockVersionLow | Versioning of record | Unsigned8 |
| 2 | Reserved | – | Unsigned16 |
| 4 | InputDataLength | Length of Input data (LI= 0 to 33) | Unsigned8 |
| 5 | Input data | Input data of submodule | Octet String |
| 5+(LI+1) | OutputDataLength | Length of Output data (LO= 0 to 32) | Unsigned8 |
| 5+(LI+1)+1 | Output data | Read back Output data of submodule | Octet String |

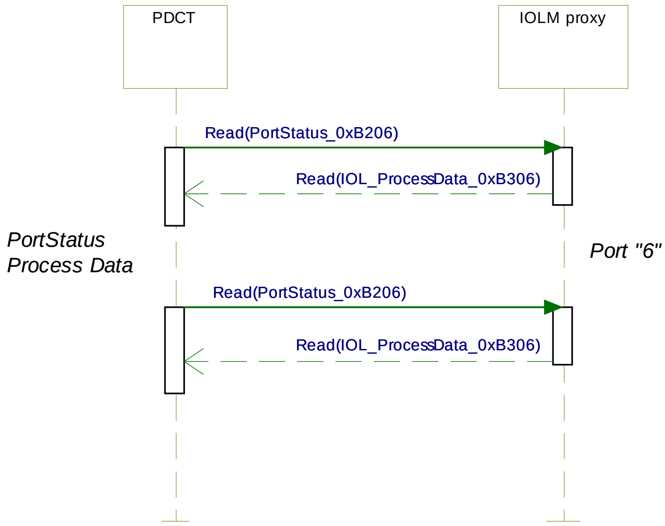



**Figure 38 – Cyclically reading PortStatus and ProcessData**

### 11.4.7 Substitute Value

The corresponding Process Data record can also be written (Substitute Value). The data of this record become only effective as long as no AR is established or IOPS (Output) has been detected "bad" (substitute value behavior). Thus, control of the Device via a tool is possible.

The addressing of these records is maintained using formula (3).

Table 49 shows the coding of the Process Data record.

**Table 49 – Coding of Process Data record (Write)**

| Offset | Parameter name | Definition | Data type |
|---|---|---|---|
| 0 | BlockVersionHigh | Versioning of record | Unsigned8 |
| 1 | BlockVersionLow | Versioning of record | Unsigned8 |
| 2+(LI+1) | OutputDataLength | Length of Input data (LO= 0 to 32) | Unsigned8 |
| 2+(LI+1)+1 | Output data | Output data of submodule | Octet String |

## 11.5 PDCT integration

Several standards for tool integration exist on the market such as FDI (IEC 62769), FDT/DTM (IEC 62453), and TCI [10]. They all are currently in use for the integration of PDCT into PROFINET engineering systems and for accessing the communication networks to Devices.

It is up to the owner of the engineering systems to choose any of these standards or to implement a proprietary method for the integration of PDCTs.

## 11.6 PDCT interface

Even though it is acceptable that several standards for integration of PDCTs are in use, it is a requirement that a particular PDCT shall be able to interact in a network with several Submodules with IO-Link Masters of different brand on board.

In order to achieve a standardized access, any "Linking Module" shall provide standardized mandatory data objects for the purpose of controlling the Master system. The IOLM proxy is the access point for a PDCT. Table 50 shows the data objects for the PDCT interface.

**Table 50 – Data objects for the PDCT interface**

| Data Object | Index | Read/Write | Description |
|---|---|---|---|
| IOLM_Info | 0xB000 | R | Master information e.g. cap of IOL-Call, PortCount,etc. |
| PortConfiguration | 0xB100 to 0xB1FF | R/W | Consolidated Port Configuration (CPC) |
| PortStatus | 0xB200 to 0xB2FF | R | Shows the status of the Port and the pending diagnosis information |
| IOLProcessData | 0xB300 to 0xB3FF | R | Shows the process data input and output for monitoring purposes |
| IOL_CALL | 0xB400 | R/W | Parameterization of Devices via IOL_CALL |

### 11.6.1 Master information

Any application within the system such as asset management shall be able to get information about the deployed Master. Therefore, the IOLM proxy shall support an IOLM_Info object within a record on Index 0xB000. This record is read only. Table 51 shows the coding of the IOLM_Info data object.

**Table 51 – Coding of IOLM_Info data object**

| Part | Parameter name | Definition | Data type |
|---|---|---|---|
| Header | BlockVersionHigh | Versioning of record | Unsigned8 |
| | BlockVersionLow | Versioning of record | Unsigned8 |
| | Reserved | – | Unsigned16 |
| | PortCount | Number of supported ports | Unsigned8 |
| | CAP | Supported client access point | Unsigned16 |

### 11.6.2 PortConfiguration behavior

There are two ways for a Master to perform port configuration. Start with GSD-based parameters (record 0xB900) or with Tool-based parameters from non-volatile memory (record 0xB10x). Upon power-on, the port is always started with Tool-based parameters, which causes the following possibilities:

- PROFINET start-up is preset to "Tool-based": No further actions.
- PROFINET start-up is preset to "Autoconfig": Restart in mode "Autoconfig".
- PROFINET start-up is preset to "GSD-based": Restart with parameters from 0xB900.

## 12 Overview of mandatory and optional features

Table 52 and Table 53 list mandatory and optional features. Support of the tool interface is optional for IO-Link Gateway manufacturers. However, if available, all conditional (C) objects shall be supported.

**Table 52 – Mandatory and optional features, part 1**

| Feature | | Remark | Reference specification |
|---|---|---|---|
| **GSD supported** | | | ***See also template GSD*** |
| *PortMode: IO-Link - Autoconfig* | M | Support of Plug and Play | see chapter 7.4.2 Port configuration Record |
| *PortMode: IO-Link - Manual* | M | Support of GSD based Port Configuration | see chapter 7.4.2 Port configuration Record |
| *PortMode: Digital Input* | M | Support of Digital Input Functionality | see chapter 7.4.2 Port configuration Record |
| *PortMode: Digital Output* | M | Support of Digital Output Functionality | see chapter 7.4.2 Port configuration Record |
| *Support of Validation & Backup and Port Cycle Time* | M | Support of GSD based Port Configuration | see chapter 7.4.2 Port configuration Record |
| *Enable Input fraction* | O | | see chapter 7.4.2 Port configuration Record |
| **PDCT support (Tool support)** | O | | See 11.4 Port and Device configuration Tool (PDCT) |
| *PortMode: IO-Link - Tool based* | C | Support of Plug and Play | |
| *Support PortConfiguration (record 0xB1xx)* | C | Mapped to IOLM proxy | |
| *Support PortStatus (record 0xB2xx)* | C | Mapped to IOLM proxy | |
| *Support IOLProcessData (record 0xB3xx)* | C | Mapped to IOLM proxy | |
| *Support IOLM_Info(record 0xB0xx)* | C | Mapped to IOLM proxy | |
| *Support IOL_CALL(0xB400)* | C | Mapped to IOLM proxy. PDCT access to IO-Link Device objects via IO-Link Call | |
| **Linking Module model** | | | See chapter 6 Slot model of the "Linking Module" |
| *Support IOLD proxy* | M | IO-Link Master proxy | |
| *Support IOLM proxy* | M | Port and Device proxy | |
| **IO-Link Startup behavior** | | | See chapter 8.1.4 Buffered port configuration model |
| *Buffered Port Cofiguration model* | M | Persistent storage of Port Configuration to speed up Startup. The port will be started after power on with the last valid Port configuration. | |

**Table 53 – Mandatory and optional features, part 2**

| Functionality | | Remark | Reference specification |
|---|---|---|---|
| **I&M support** | | | See chapter 7.8 I&M data |
| *I&M0 IOLM proxy* | M | via IOLM proxy | See chapter 7.8.2 I&M data of IOLM proxy |
| *I&M0 IOLD proxy* | M | via IOLD proxy | See chapter 7.8.2 I&M data of IOLD proxy |
| *I&M5 IOLD proxy* | O | via IOLD proxy | See chapter 7.8.3.3 I&M5 |
| **IO data mapping** | | | See chapter 7.5 Process Data (IO data) |
| *support PQI.PQ* | M | Programing interface for Port and Device to show the data validity and Port Status information | see chapter 7.5.4.1 Port Qualifier information (PQI) |
| *support PQI.DevCom* | M | | |
| *support.SubstDev* | M | | |
| *support PQI.NewPar* | M | | |
| *support PQI.DevErr* | M | | |
| *IO data mapping* (0..32 octets) | M | | see chapter 7.5 |
| **Diagnosis / Event mapping** | | | |
| Pull/ Plug support | M | via IOLD proxy, Pull / Plug according availability of IOL Device | See chapter 8.4 Pull/ Plug behavior |
| *Device diagnosis mapping* | M | via IOLD proxy | See chapter 7.6 Diagnosis |
| *Port diagnosis mapping* | M | via IOLD proxy | See chapter 7.6 Diagnosis |
| Process Alarm mapping (Notification) | M | via IOLD proxy | See chapter 7.7 Alarms (Process alarms) |
| **Backup / Restore** | | | See chapter 9 Extended Data storage and application support |
| *Support Backup (Read 0xB904)* | M | via IOLD proxy | see 9.1 Backup & Restore" |
| *Support Restore (Write 0xB904)* | M | via IOLD proxy | see 9.1 Backup & Restore" |
| **Device / Parametrization** | | | See chapter 10 IOL_CALL method |
| *Support IOL_CALL* (0xB400) | M | Support IO-Link Call via IOLD proxy | See 10.7 IOL_CALL protocoll |
| *Support ISDU batch object* (0xB903) | O | Support ISDU batchl via IOLD proxy | See 10.8.2 ISDU batch object |
| **Port Functionality** | | | See chapter 9 Extended Data storage and application support |
| *Support Port Deactivation* | O | Port deactivation (Tool changer use case) | see 9.2.3 "Deactivate Port" function |
| *Support Port Activation* | O | Port activation (Tool changer use case) | see 9.2.4 "Activate Port" function |
| *Device Exchange detction* | M | Detection of a new device (serial number based) | see 9.3 Detection of Device exchange |

# Annex A
(informative)

# Extended Port functions

## A.1 General

The IO-Link specification [1] defines two Port types, class A and class B. Pin2 of Port class A can be configured as DI or DO. This integration document even expands this definition by AI or AO. The user can switch the extra power OFF/ON at Pin2/5 in case of Port class B.

The IO-Link specification [1] defines the following Port constellations:

- Port class A with nothing connected at Pin2 (IOLM proxy without IO data)
- Port class B without switchable extra power (IOLM proxy without IO data)

In practice, extended Port functions are available that are beyond the IO-Link specification and thus in principle not normative.

Those extended Port functions are:

- Port class A with functions at Pin2 (manufacturer specific DI, DO, AI, or AO)
- Port class B with switchable extra power at Pin2

## A.2 Mapping of Extended Port functions

It is highly recommended to map the extra Input/Output data into the IOLMproxy:

- In case of port class B with switchable extra power a switchbit per port is defined within the output area – port1 is mapped to bit 0 of octet 0
- In case of port class A with DO on Pin2 a DO bit per port is defined within the output area – port1 is mapped to bit 0 of octet 0
- In case of port class A with DI on Pin2 a DI bit per port is defined within the input area – port1 is mapped to bit 0 of octet 0
- In case of port class A with AI on Pin2 an INT16 variable per port is defined within the input area – port1 is mapped to octet 0 and 1

The update rate is manufacturer/vendor specific.

# Annex B
(normative)

# Test

## B.1 Overview

This part of the document will contain the test cases once the community reviews have been performed.

## B.2 Use cases as basis

The lists in Figure B.1 and Figure B.2 provide an overview of the relevant use cases for both "GSD-based port configuration" and "Tool-based configuration". The behavior specified in these lists are the basis for conformity testing.

Figure B.1 shows the use cases for "GSD-based port configuration".

| **GSD based Port Configuration** | | | **User view (offline GSD based)** | | **IO-Link Device** | Linking Module behavior | | | | User view (online) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Use case** | **Description** | **Remedy** | **IOLD proxy** | **Port Config (record 0xB900)** | **Port/ IO-Link device** | **ModuleDiff Info** | **Real Identification** | **Expected Identification** | **IOxS** | **Diagnosis** | **Port Qualifier (PQ)** | **I&M0** | **I&M5** |
| Use case 1a: | *no IOLD available* | *Check IO-Link Device connect* | IO-Link x I/ y O + PQI | don´t care | **not available** | **no submodule** | **no submodule** | SMID = 0x0000 xxxx | **bad** | no diagnosis | not accessible ("0") | **no submodule** | **no submodule** |
| Use case 2.1a: | **O.k** | **Everything correct** | IO-Link x I/ y O + PQI | VID, DID : Type a<br>Inspection Level: Type compatible | **VID, DID: Type a** | substitute | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | good | no diagnosis | PQ= 1; DevCom=1, DevErr=0 | Reduced Info for IOLD (typ a) | Full information from the IOLD itself<br>Type a |
| Use case 2.2a: | **O.k** | **Everything correct** | IO-Link 32 I/ 32 O + PQI | VID, DID : Type a<br>Inspection Level: Type compatible | **VID, DID: Type a** | o.k | SMID = 0x0000 2021 | SMID = 0x0000 2021<br>== Real | good | no diagnosis | PQ= 1; DevCom=1, DevErr=0 | Reduced Info for IOLD (typ a) | Full information from the IOLD itself<br>Type a |
| Use case 3a: | Wrong IOLD device according Inspection Level | *Check if the real device is correct or the Port Parametrization (GSD) must be changed* | IO-Link x I/ y O + PQI | VID, DID : Type a<br>Inspection Level: Type compatible | **VID, DID: Type b** | substitute | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | good | **Port diagnosis - Inspection Level fault** | PQ= 0; DecCom=1, DevErr=1 | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 4a: | Inspection Level: no check | **Everything correct** | IO-Link x I/ y O + PQI | VID, DID : xxx<br>Inspection Level: no check | **VID, DID: Type b** | substitute | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | good | no diagnosis | PQ= 1; DevCom=1, DevErr=0 | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 5a: | No proxy available | | not available | not available | **IOLD (type c)** | not available | SMID = 0x0000 2021 | --- | | | | Reduced Info for IOLD (Type c) | Full information from the IOLD itself<br>Type c |
| Use case 6a: | Process data length mismatch | *Check PROFINET configuration. Maybe a wrong submodule was plugged* | IO-Link x I/ y O + PQI | don´t care | **IOLD (type b)** | **wrong submodule** | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | **bad** | no diagnosis | not accessible ("0") | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 7a: | invalid submodule | *Check PROFINET configuration. Maybe a wrong submodule was plugged* | xyz (invalid) | don´t care | **IOLD (type b)** | **wrong submodule** | SMID = 0xFFFF 0000 | SMID = xxyz ????<br>'!= Real | **bad** | no diagnosis | not accessible ("0") | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 8a: | Special error in Startup phase | Check the IO-Link device or change the device, because it is faulty | IO-Link x I/ y O + PQI | don´t care | **Problem during startup** | substitute | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | good | **Port diagnosis - "Error"** | not accessible ("0") | **no submodule** | **no submodule** |
| Use case 9a: | Set to DI | | Digital Input | don´t care | **set to DI Mode** | o.k | SMID = 0x0000 0081 | SMID = 0x0000 0081<br>== Real | good | no diagnosis | not available | Digital Input | --- |
| Use case 10a: | Set to DQ | | Digital Output | don´t care | **set to DQ Mode** | o.k | SMID = 0x0000 8100 | SMID = 0x0000 8100<br>== Real | good | no diagnosis | not available | Digital Output | --- |

**Figure B.1 – Use cases for "GSD-based port configuration"**

Figure B.2 shows the use cases for "Tool-based port configuration".

**Tool based Port Configuration**

| | | | User view GSD | IO-Link Tool | IO-Link Device | Linking Module behavior | | | | User view (online) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Use case | Description | Remedy | IOLD proxy | Port Config (record 0xB10x) | Port/ IO-Link device | ModuleDiff Info | Real Identification | Expected Identification | IOxS | Diagnosis | Port Qualifier (PQ) | I&M0 | I&M5 |
| Use case 1b: | *no IOLD available* | *Check IO-Link Device connecti* | IO-Link x I/ y O + PQI | don´t care | **not available** | **no submodule** | **no submodule** | \`? | **bad** | no diagnosis | not accessible ("0") | **no submodule** | **no submodule** |
| Use case 2b: | **O.k** | **Everything correct** | IO-Link x I/ y O + PQI | VID, DID : Type a<br>Inspection Level: Type compatible | **VID, DID: Type a** | substitute | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | good | no diagnosis | PQ= 1; DevCom=1, DevErr=0 | Reduced Info for IOLD (typ a) | Full information from the IOLD itself<br>Type a |
| Use case 3b: | Wrong IOLD device according Inspection Level | *Check if the real device is correct or the Port Parametrization (Tool based) must be changed* | IO-Link x I/ y O + PQI | VID, DID : Type a<br>Inspection Level: Type compatible | **VID, DID: Type b** | substitute | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | good | **Port diagnosis - Inspection Level fault** | PQ= 0; DecCom=1, DevErr=1 | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 4b: | Inspection Level: no check | **Everything correct** | IO-Link x I/ y O + PQI | VID, DID : xxx<br>Inspection Level: no check | **VID, DID: Type b** | substitute | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | good | no diagnosis | PQ= 1; DevCom=1, DevErr=0 | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 5a: | No proxy available | | not available | not available | **IOLD (type c)** | not available | SMID = 0xFFFD 0000 | --- | | | | Reduced Info for IOLD (Type c) | Full information from the IOLD itself<br>Type c |
| Use case 6b: | Process data length mismatch | *Check PROFINET configuration. Maybe a wrong submodule was plugged* | IO-Link x I/ y O + PQI | don´t care | **IOLD (type b)** | **wrong submodule** | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | **bad** | no diagnosis | not accessible ("0") | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 7b: | invalid submodule | *Check PROFINET configuration. Maybe a wrong submodule was plugged* | xyz (invalid) | don´t care | **IOLD (type b)** | **wrong submodule** | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | **bad** | no diagnosis | not accessible ("0") | Reduced Info for IOLD (Type b) | Full information from the IOLD itself<br>Type b |
| Use case 8b: | Special error in Startup phase | Check the IO-Link device or change the device, because it is faulty | IO-Link x I/ y O + PQI | don´t care | **Problem during startup** | **substitute** | SMID = 0x0000 2021 | SMID = 0x0000 xxxx<br>'!= Real | **bad / good ?** | **Port diagnosis - "Error"** | not accessible ("0") | **no submodule** | **no submodule** |
| Use case 9b: | Set to DI | | **not possible** | | | | | | | | | | |
| Use case 10b: | Set to DQ | | **not possible** | | | | | | | | | | |

**Figure B.2 – Use cases for "Tool-based port configuration"**

# Bibliography


[1] IO-Link Community, *IO-Link Interface and System*, V1.1.2, July 2013, Order No. 10.002, or
IEC 61131-9, *Programmable controllers – Part 9: Single-drop digital communication interface for small sensors and actuators (SDCI)*

[2] IO-Link Community, *IO Device Description (IODD)*, V1.1, July 2011, Order No. 10.012

[3] PI specification, *PROFINET application layer protocol for decentralized periphery and distributed automation*, Version 2.3, Ed2, MU1, March 2014, Order No. 2.722, or
IEC 61158-6-10, *Industrial communication networks – Fieldbus specifications – Part 6-10: Application layer protocol specification – Type 10 elements*

[4] PI Guideline, *Fieldbus integration in PROFINET IO*, Version 2.0, May 2011, Order No. 7.012

[5] PI Specification, *IO-Link Integration, Part 1*, Version 1.0, December 2007, Order No. 2.812

[6] PI Profile Guidelines, Part 1, *Identification & Maintenance Functions,* Version 2.0, January 2014, PROFIBUS & PROFINET International, Order No. 3.502

[7] PI Profile Guideline, *PROFINET Diagnosis*, Version 1.2, June 2016, Order No. 7.142

[8] PI Profile Guidelines, Part 4, *iPar-Server*, Version 1.0.1, July 2011, Order No. 3.532

[9] PI library, *GSD template for IO-Link*, www.profibus.com

[10] PI Specification, Tool Calling Interface, Version 1.1, October 2008, Order No. 2.602


___________

PROFIBUS Nutzerorganisation e. V. (PNO)
PROFIBUS & PROFINET International (PI)
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
www.profibus.com • www.profinet.com