IO-Link
Common Profile Draft

Specification

Version 1.0.91
December 2020

Order No: 10.072

File name: **IOL_Common-Profile_D1091_Dec2020**

This draft covers the change requests ID 26 to 48 to the Common Profile.

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

Important notes:

NOTE 1 The IO-Link Community Rules shall be observed prior to the development and marketing of IO-Link products. The document can be downloaded from the www.io-link.com portal.

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

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

Disclaimer:

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

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

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

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

IO-Link ® is registered trade mark. The use is restricted for members of the IO-Link Community. More detailed terms for the use can be found in the IO-Link Community Rules on www.io-link.com.

Conventions:

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

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

Publisher:
IO-Link Community
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 721 / 96 58 590
Fax: +49 721 / 96 58 589
E-mail: info@io-link.com
Web site: www.io-link.com

CONTENTS

0 Introduction

0.1 General

The Single-drop Digital Communication Interface (SDCI) and system technology (IO-Link™[1])) for sensors and actuators is standardized within IEC 61131-9 [2]. The technology is an answer to the need of these digital/analog sensors and actuators to exchange process data, diagnosis information and parameters with a controller (PC or PLC) using a digital communication technology while maintaining backward compatibility with the current DI/DO signals as defined in IEC 61131-2.

Tools allow the association of Devices with their corresponding electronic I/O device descriptions (IODD) and their subsequent configuration to match the application requirements [3].

This document describes the common parts of sensor and actuator models to be used in all Device profiles as well as the base profile "Identification & Diagnosis" which is mandatory for all profiled Devices.

This document follows the IEC 62390 [4] to a certain extent.

Terms of general use are defined in IEC 61131-1 or in [5]. Specific SDCI terms are defined in this part.

0.2 Patent declaration

There are no known patents related to the content of this document.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The IO-Link Community shall not be held responsible for identifying any or all such patent rights.

[1] IO-Link™ is a trade name of the "IO-Link Community". This information is given for the convenience of users of this specification. Compliance to this specification does not require use of the registered logos for IO-Link™. Use of the registered logos for IO-Link™ requires permission of the "IO-Link Community".

Common Profile Draft — Related to IEC 61131-9

1 Scope

The single-drop digital communication interface (SDCI) technology described in part 9 of the IEC 61131 series focuses on simple sensors and actuators in factory automation, which are nowadays using small and cost-effective microcontrollers. With the help of the SDCI technology, the existing limitations of traditional signal connection technologies such as switching 0/24 V, analog 0 to 10 V, etc. can be turned into a smooth migration. Classic sensors and actuators are usually connected to a fieldbus system via input/output modules in so-called remote I/O peripherals. The (SDCI) Master function enables these peripherals to map SDCI Devices onto a fieldbus system or build up direct gateways. Thus, parameter data can be transferred from the PLC level down to the sensor/actuator level and diagnosis data transferred back in turn by means of the SDCI communication. This is a contribution to consistent parameter storage and maintenance support within a distributed automation system. SDCI is compatible to classic signal switching technology according to part 2 of the IEC 61131 series.

The Profile Guideline part of this document defines the common characteristics (models) of Device profiles. These models comprise process data structures, identification objects, best practice handling of quantity measurements with or without associated units, and diagnosis objects. To achieve a high grade of conformity to the profiles, this document contains statements on conformity testing for all Device profiles including IODD checker rules.

The Profile part of this document contains the general Identification & Diagnosis profile which is mandatory for all other profiles.

2 Normative 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 61131-3*, Programmable controllers – Part 3: Programming languages*

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

3 Terms, definitions, symbols, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions in addition to those given in IEC 61131-1, IEC 61131-2, and IEC 61131-9 apply.

3.2 Profile Guideline : Additional terms and definitions

3.2.1
BinaryDataChannel
BDC

binary information as switching or control signal

3.2.2
Function block
software functional element comprising an individual, named copy of a data structure and associated operations specified by a corresponding function block type

[SOURCE: IEC 62390, 3.1.13]

3.2.3
FunctionClass
particular function within a Device profile

Note 1 to entry: A profile Device can use one or several FunctionClasses once or several times.

3.2.4
not applicable
n/a
this entry cannot be applied within this context

3.2.5
ProfileIdentifier
unique identifier for Device Profile, CommonApplicationProfile, or FunctionClass

3.3 Symbols and abbreviated terms

PD	Process Data
PLC	Programmable logic controller
SDCI	Single-drop digital communication interface

3.4 Conventions

3.4.1 Behavioral descriptions

For the behavioral descriptions, the notations of UML2 [12] are used, mainly state diagrams. The layout of the associated state-transition tables is following IEC 62390 [4].

The state diagrams shown in this document are entirely abstract descriptions. They do not represent a complete specification for implementation.

4 Objectives for Device profiles

4.1 Purpose of Device profiles

In factory automation, sensors nowadays are using a broad spectrum of transducers based on many different physical or chemical effects. They are converting one or more physical or chemical quantities (for example position, pressure, temperature, substance, etc.) and propagate them in an appropriate form to data processing units such as for example PLCs.

Also actuators like lamps, locks, valves, motors, and so on are not only actuators. The internal states are going to be important for the customers. Even the acknowledged control and configuration is of increasing importance and not covered by simple digital output signals.

It is the purpose of SDCI to overcome the limitations of the classic Device interfaces DI, DO, AI, and AO via a point-to-point digital communication that allows transmitting digitally not only binary and/or analog information but additional information also. Very often, the changes to the core sensor or actuator application ("sensor/actuator technology") are very little during the migration to SDCI. However, the user realizes a dramatic increase in comfort and flexibility through the identification, parameterization, and diagnosis features.

As a consequence of the digitization, the Devices can also provide many more technology features and data structures to the user for processing within for example a PLC user program than before with the classic interfaces.

Device profiles define terminologies, features, behaviours, commands, responses, corresponding data structures, and other things common to particular Device families and thus prevent the user from a confusing variety.

4.2 Interoperability

The major parts of the Device profiles deal with process data structures and behavior as well as parameter data structures and dynamic parameterization at runtime. These features streamline the functions of comparable Devices though requiring more sophisticated and powerful PLC user programs. Thus, interoperability between existing user programs and Devices of a corresponding family is the goal of profile Device design and testing (see [4]). Figure 1 shows compatibility levels based on IEC 62390.

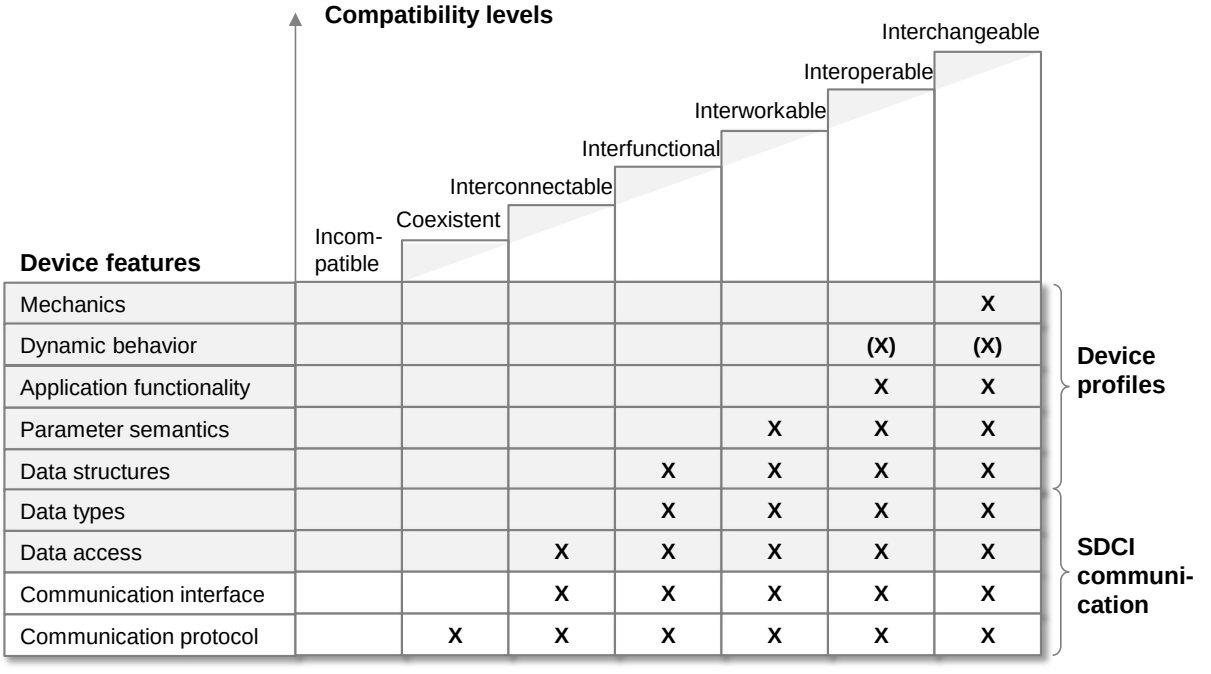


Figure 1 – Compatibility levels based on IEC 62390

The different compatibility levels are described in Table 1 and the different Device features are described in Table 2, based on [4].

118 **Table 1 – Explanation of compatibility levels**

Compatibility level	Definition
Incompatible	Two or more devices are incompatible if they cannot exist together in the same distributed system
Coexistent	Two or more devices coexist on the same communications network if they can operate independently of one another in a physical communication network or can operate together using some or all of the same communication protocols, without interfering with the application of other devices on the network
Interconnectable	Two or more devices are interconnectable if they are using the same communication protocols, communication interface and data access
Interfunctional	Two or more devices are interfunctional if they can exchange data for specific purposes without manual configuration, the parameter semantics are defined and the devices provide the necessary identifier
Interworkable	Two or more devices are interworkable if they can transfer parameters between them, i.e. in addition to the communication protocol, communication interface, and data access, the parameter data types are the same
Interoperable	Two or more devices are interoperable if they can work together to perform a specific role in one or more distributed applications. The parameters and their application related functionality fit together both syntactically and semantically. Interoperability is achieved when the devices support complementary sets of parameters and functions belonging to the same profile
Interchangeable	Unlike the other compatibility levels (which refer to two or more devices working in the same system) interchangeability refers to the replacement of one device with another. Devices are interchangeable for a given role in a distributed application if the new device has the functionality to meet the application requirements

119

120 **Table 2 – Explanation of Device features**

Device features	Definition
Mechanics	This feature is defined by the mechanical outline, process connector, and/or electrical connection
Dynamic behavior	This feature is defined by time constraints that influence the parameter update or the general device behavior. For example, the update rate of a process value can influence block algorithms.
Application functionality	This feature is defined by specifying the dependencies and consistency rules within the functional element. This is done in the parameter description characteristics or in the device behavior section
Parameter semantics	This feature is defined by the parameter characteristics: parameter name, parameter description, parameter range, substitute value of the parameter, default value, persistence of the parameter after power loss and deployment
Data structures	This feature is defined by the combination of data types, such as records of simple data types
Data types	This feature is defined by characteristics such as “data type”, see Note
Data access	This feature is defined by characteristics such “parameter timing” and “access direction”, see Note
Communication interface	This feature is defined by the protocols of layer 5 to 7 of the OSI reference model including the services and the servie parameters. Additional mapping mechanisms can be necessary. The dynamic performance of the communication system is part of this feature
Communication protocol	This feature is defined by all protocols of layer 1 to 4 of the OSI reference model, i.e. from the physical medium access to the transport layer protocol

Device features	Definition
Note: “parameter timing”: in the context of SDCI this refers to the used data channels like acyclic or process data “access direction”: specification whether the parameter can be read and/or written “data type”: IEC 61131-3 data types are preferred	

4.3 Standard structure

This standard provides two separate contents, the first covers the overall aspects on which the SDCI profiles are based on. The second part contains the specification of the profile Identification and Diagnosis which shall be used whenever another SDCI profile is used.

Clause 5 shows the environment in which the profiles are embedded and clause 6 specifies general base rules for all SDCI related profiles. Clause 7 shows the utilization of the available Device profiles during the design phase of an SDCI Device. A guideline how process data structures are build is defined in Annex H. Annex I contains an explanation on how the Devices are integrated into an abstract automation system.

Clause 8 specifies the base profile Identification and Diagnosis, with further definitions placed in Annex A (functionality), Annex B (parameters), Annex C (PLC function block), and Annex D (IODD rules).

Annex E specifies the base technologies which extend the test specification in [6] to ensure the specified profile characteristics. Specific test definitions for the Identification and Diagnosis profile are defined in Annex F.

5 Device profiles related to IEC 61131-9

5.1 SDCI technology specified in IEC 61131-9

Figure 2 shows the domain of the SDCI technology within the automation hierarchy.

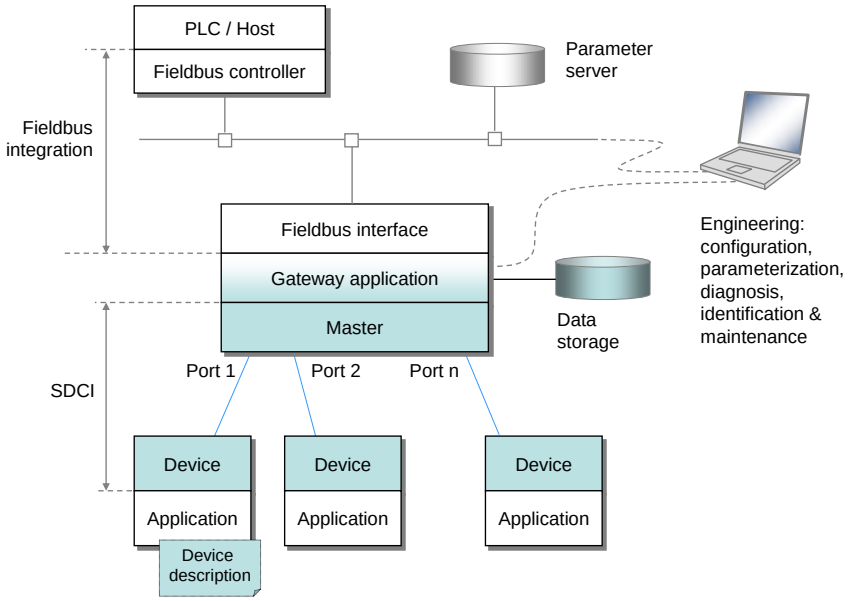


Figure 2 – Domain of the SDCI technology within the automation hierarchy

The SDCI technology defines a point-to-point digital communication interface for connecting "digital" or "analog" type sensors and actuators to a Master unit, which can be combined with gateway capabilities to become a fieldbus remote I/O node. The technology is specified in [1] and [3].

5.2 Profile classification

Figure 3 shows an overview of the SDCI technologies and profiles.

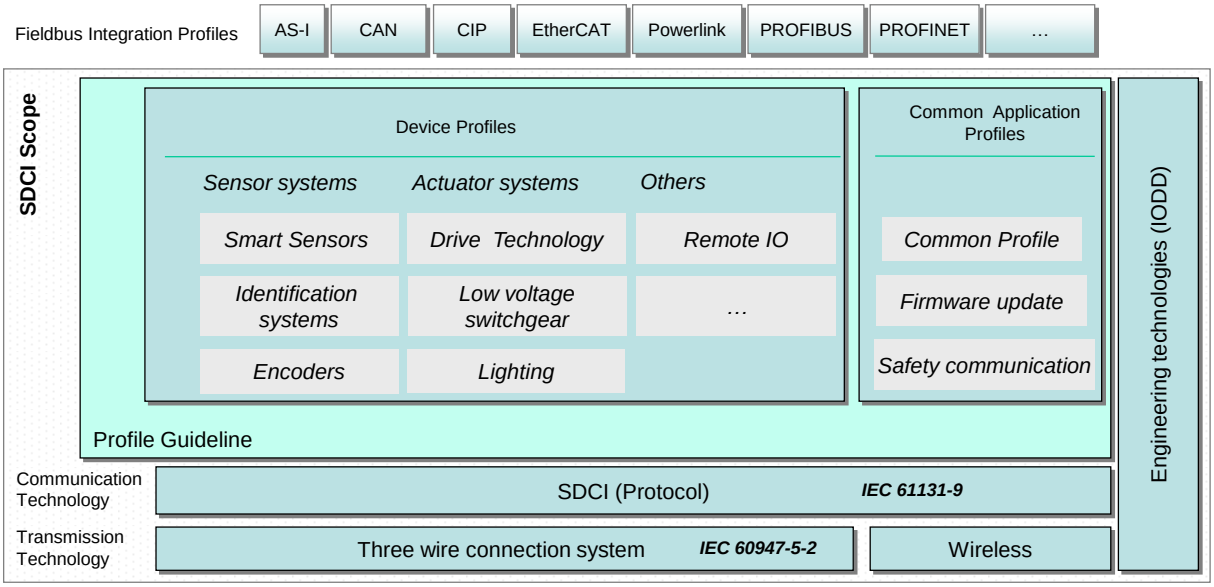


Figure 3 – Overview of SDCI technologies and profiles

The "Device Profiles" represent specifications of common functionality of particular Device type families/classes such as

- smart sensors,
- smart actuators,
- lighting
- etc.

These profiles primarily focus on the structure and behavior of the Device technology and secondarily on the data structure mapping on SDCI. Thus, the user experiences a well-known Device behavior to a certain extent even when he uses different Devices or switches from one brand to another.

The "Common Application Profiles" represent specifications that several Device type families/ classes such as

- Common Profile
- Firmware update
- Safety communication
- etc.

The "Fieldbus Integration Profiles" specify the adaptation of the SDCI technology to particular fieldbuses. These specifications are outside the responsibility of the organization listed in Annex G. However, this organization is interested in harmonizing the "views" of SDCI users through the different fieldbuses.

5.3 Concept of profiles

The approach for profiling SDCI Devices follow the guideline to enforce as many equal functionality or behavior in all SDCI Devices. It is a stacked approach by defining a common subset which are highly recommended for all Devices like defined in the basic requirements. Furthermore the application specific profiles such as smart sensors or specific actuators define application specific requirements and solutions.

5.3.1 Basic requirements for profile Devices

As [1] defines only a few parameter as mandatory, the profile Devices shall support more parameters to allow standardized handling of these devices.

The following base features shall be supported by all profile devices :

- IO-Link Version 1.1
- ISDU support
- DataStorage for all parameters which a spare part needs for full functionality
- Support of block parameter handling
- Identification and Diagnosis

5.3.2 Distinct profiles

The distinct profiles consist of a defined combination of one or multiple FunctionClasses identified by ProfileIdentifier. The mandatory FunctionClasses are defined in the related specifications and may contain FunctionClasses from different specifications. The Device functionality may be extended by manufacturer specific parameters or additional FunctionClasses. The user can always and only rely on the functions defined by the ProfileIdentifer of the Device, this provides a higher level of interchangeablility even between different manufacturers.

To sharpen the distinct profiles and reduce the amount of similar profiles with minor differences, some FunctionClasses can be accompanied to profiles when they are supplements to the profile functionality.

5.4 Profile characteristics

The profiles are based on the definition of FunctionClasses. These FunctionClasses are combined to ProfileIDs such as

- DeviceProfileIDs for particular classes of Devices, or
- CommonApplicationProfileIDs for generic use in all Devices.

The supported functionality of a Device shall be listed within an array of ProfileIdentifier. It is also possible for a Device to support several DeviceProfiles, CommonApplicationProfiles as well as FunctionClasses as extensions for specific ProfileIDs(see 5.5).

The DeviceProfiles or CommonApplicationProfiles define the mandatory FunctionClasses for the specific ProfileIdentifier.

FunctionClasses defined in particular Device profiles can be inherited to other Device profiles.

The CommonApplicationProfile Identification and Diagnosis (I&D) is mandatory for Devices which provide at least one other profile and highly recommended for all other Devices.

An overview of the defined ProfileIdentifier is available on www.io-link.com.

The parameter object "ProfileCharacteristic" provides a list of the supported profiles. Each supported ProfileID and FunctionClass shall be indicated and coded as specified in Table 3.

To avoid multiple occurences of a ProfileIdentifier in the list, all FunctionClasses which are included in the referenced ProfileIDs shall be omitted.

Table 3 – Coding of ProfileIdentifiers (PID)

Parameter object name	Data type	Value range	Profile type
ProfileIdentifier (PID)	UIntegerT16	0x0000	No profile supported
		0x0001 to 0x3FFF	DeviceProfileID
		0x4000 to 0x7FFF	CommonApplicationProfileID
		0x8000 to 0xBFFF	FunctionClassID
		0xC000 to 0xFFFF	Reserved

The following rules apply:

1) Whenever 1 to n Device profiles are supported, they shall be indicated via 1 to n DeviceProfileID entries
2) Whenever 1 to n common application profiles are supported, they shall be indicated via 1 to n CommonApplicationProfileIDs
3) Additionally supported FunctionClasses which are not covered by DeviceProfileIDs or CommonApplicationProfileIDs shall be indicated by 1 to n FunctionClassIDs
4) The IDs shall be listed in ascending order

Figure 4 – Indication rules for ProfileIdentifiers

The different profile identifiers shall be ordered within the array of the parameter object "ProfileCharacteristic" in a sequence shown in Table 4.

Table 4 shows the example content of the "ProfileCharacteristic" of an adjustable switching sensor.

Table 4 – Example of the profile identification of a distinct switching sensor

Index	ProfileIdentifier type	Referenced Profile ID
0x000D	DeviceProfileID	0x0005: SSP 2.2
		0x4000: I&D

Table 5 shows the example content of the "ProfileCharacteristic" of a Smart Sensor with additional FunctionalClasses.

Table 5 – Example of the profile identification of an extended Profile

Index	ProfileIdentifier type	Referenced ProfileID
0x000D	DeviceProfileID	0x0005: SSP 2.2
		0x4000: I&D
	FunctionClassID	0x8100: Visual Localization

For further details, see B.4.

5.5 Concept of FunctionClasses

So far only a so-called function-driven Device model instead of for example an architectural model is defined. That means it only defines independent and consistent functions (FunctionClasses) that are available via the communication channels. This allows the community and the Device manufacturers/vendors to create a variety of subsets from basic switching sensors/actuators using only the FunctionClasses like the Switching Signal Channel (SSC) up to complex sensors/actuators with several measurement values using the FunctionClasses like the ProcessDataVariables (PDV).

Figure 5 shows a structure of the function-driven Device model and its FunctionClasses.

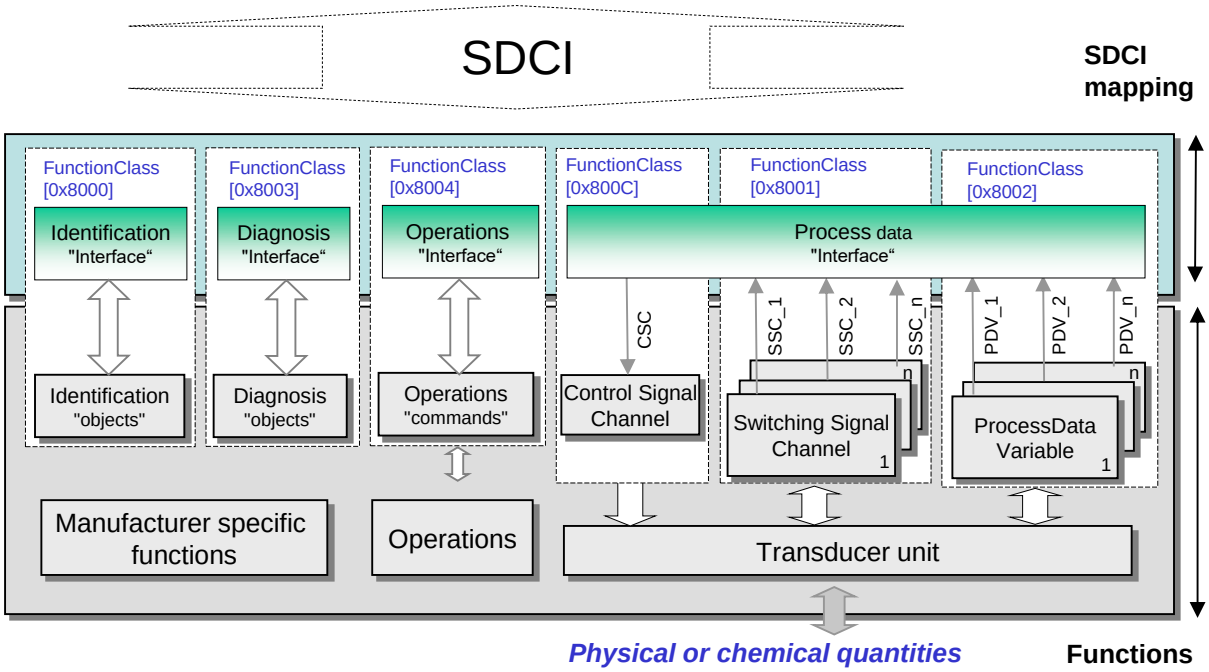


Figure 5 – Overview of typical FunctionClasses

A profile Device shall only support the indicated FunctionClasses. Normally, the measurement technology (transducer) is manufacturer/vendor specific and not part of profiles.

Each and every FunctionClass consists of a communication dependent function and an associated mapping on the SDCI communication. FunctionClasses are represented and referenced by ProfileIdentifiers, for example FunctionClassID = 0x8000, as shown in Figure 5.

The FunctionClasses DeviceIdentification, DeviceDiagnosis, ProcessDataVariable, and ExtendedIdentification are highly recommended for all profile Devices. They are combined to the DeviceProfile [0x4000] in this document. They may be defined as mandatory by definition of any CommonApplicationProfile or DeviceProfile.

A Switching Signal Channel (e.g. FunctionClass [0x8001]) uses the measurement values out of the transducer unit and creates switching information (SSCn), whenever certain thresholds are passed. These thresholds are defined via parameters.

In case of a combined sensor/actuator Device, the FunctionClass [0x800C] is used for switching the transducer ON or OFF.

The ProcessDataVariables (FunctionClass [0x8002]) uses in case of a sensor the measurement values out of the transducer unit and creates data structures (PDV_n) representing the physical or chemical quantity, for example pressure or temperature. In case of an actuator, the values are used to control the process. These data structures within the ProcessDataVariables are standardized to a maximum extent as shown in A.3.

The operation commands like FunctionClasses [0x8007] to [0x8009] allow the user for example to remotely adjust a Device in the automation process via the user program in a controller (PLC).

The mapping of SSCs, CSCs and PDVs into SDCI communication messages is specified in the corresponding FunctionClass definitions or at least in A.3. These data structures are designed for simplicity and highest efficiency.

The shown combination is just an example and may be expanded by additional FunctionClasses or reduced by omitting some of the FunctionClasses.

Figure 6 shows a possible object model of a combined sensor/actuator Device profile.

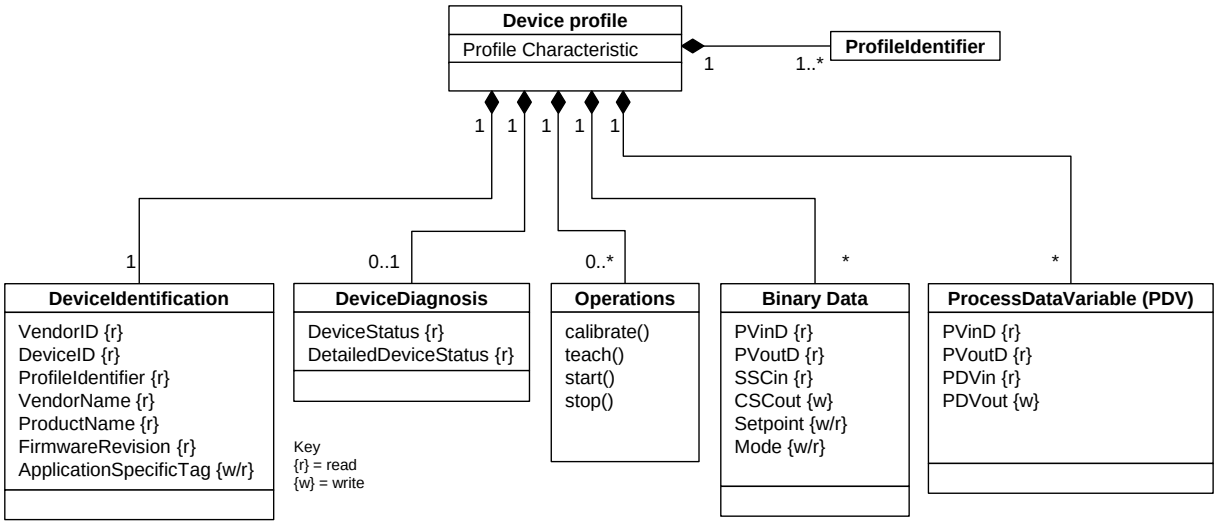


Figure 6 – Typical object model for Device profiles

The parameter "ProfileCharacteristic" contains at least one ProfileIdentifier or an array of ProfileIdentifier. Besides the objects for identification and diagnosis, it contains the objects ProcessDataVariable (PDV) and Binary Data Channel (SSC and CSC). These objects show the associated attributes, whereas the object "Operations" shows only its defined methods (commands).

The objects SSC, CSC and PDV can be used once or more times depending on the complexity of the sensor/actuator.

The parameter set of a FunctionClass can be classified into two groups:

- Operating parameters, which can be modified during production, and
- Configuration parameters (static data), which are only set/modified during commissioning.

5.6 User benefits

As already mentioned in 5.2 the user recognizes from the application point of view a "generic" Device through the communication interface even though he switches from one brand to another. The customer experiences the following advantages of profile Devices at different points in time:

- At commissioning time the process data can be easily configured due to reduced sets of process data structures. In future this could be extended by explicit support of profile Devices by the system provider.
- At programming time the process data and common parameters can be used with expected behavior and without checking the IODD of the specific Device just based on the profile defined behavior. This will be supported by specific Proxy Function Blocks for the defined Profiles.
- At runtime the Devices represent their process data in an equal manner and can be replaced by Devices with the same ProfileIdentifier and the same physical measurement or actuator behavior. For the replacement only the configured Device Identification has to be updated.

However, due to the objectives for the individual Device profiles, the interoperability levels can be different and the compatibility between the profile Devices can be partly limited. For example the measurement range of a sensor or actuator strength may be different and not suitable for the specific application. It is the responsibility of system maintenance to check this prior to a replacement of the Device.

A user program ("client") for example in a PLC can access the objects via corresponding functions or methods respectively. Table 6 shows an example.

303 **Table 6 – Tag notation for BDC and PDV access of a PLC client**

Read/Write access	Description
Read Sensor1.AppSpecTag	Readout of the parameter "ApplicationSpecificTag" of the Device
Read Sensor1.DeviceStatus	Readout of the parameter "DeviceStatus"
Write Sensor1.switch point1.SetPointValueSP1	Write parameter "SetPointValueSP1"
Write Sensor1.TeachTP1	Start teach procedure for TP1

304

305 # 6 Rules and constraints for developing Device profiles

306 Within this clause the general rules and constraints for the design of future profile specification
307 are defined.

308 ## 6.1 How to create a Device profile

309 When defining new Profiles or FunctionClasses, the following rules shall be observed

310 • The DeviceProfileID Identification and Diagnosis (I&D) is mandatory for all profile Devices
311 and shall not be redefined by any other profile.

312 • Each profile defines the mandatory FunctionClasses within the specific ProfileIdentifier

313 • There shall not be two or more FunctionClasses with an identical function (no duplicates)

314 • There will be no versioning of profiles (identifiers). Changes to a profile result in a new
315 ProfileIdentifier

316 • The profiles shall specify as mandatory all parameters and contents required to guarantee
317 the compatibility of profile Devices supporting a ProfileIdentifier

318 • The defined ProfileIdentifier and parameters shall be published in the SDCI Profile overview
319 as defined in 5.4

320 • Restrictions on combinations of ProfileIdentifier shall be described in the DeviceProfile or
321 FunctionClass definition (responsibility of the profile working group)

322 • FunctionClass specific parameters shall be defined with all attributes such as
323 – Parameter name as defined in the IODD
324 – Index
325 – Subindex access availability
326 – Subindex structure
327 – Subindex offset
328 – Parameter type or length
329 – Allowed parameter content
330 – Default value
331 – Access rights
332 – Mandatory, optional, or conditional availibility

333 • FunctionClass specific process data structures shall be defined with all attributes such as
334 – Subindex structure
335 – Subindex offset
336 – Allowed process data content
337 – Validity rules

338 • Profile parameters shall use the index range reserved for the profile considering possibly
339 combinable profiles

340 • Profiles shall define their common EventCodes within range reserved for the profile consid-
341 ering possibly combinable Profiles

342 • Profiles shall define their specific parameter naming and menu entries within an IODD

343 • Profiles shall define their specific test cases according [6] to extend the existing Device test
344 rules

345 • Profiles shall define their specific IODD checker extensions according E.1E.1.3 if necessary

6.2 Basic parameter rules

The parameters defined in a Device profile shall be accessable as defined in the corresponding ProfileIdentifier definition. In general, the rules of [1] apply. In detail the following rules shall be observed

- Any parameter shall follow the accessibility rule defined in the profile
- Optional or conditional Subindices shall always be readable and return the defined default value
- Optional or conditional Subindices with write access shall always accept a writing of the defined default value

Table 7 shows the profile related Indices defined in [1]. It is the responsibility of the community to avoid interferences and duplicates of parameters over different profile specifications.

Table 7 – Excerpt of the SDCI Indices related to profiles

Index (dec)	Object name	Access	Length	Data type	M/O/C	Smart Sensor profile definitions
…						
0x0031 to 0x003F (49 to 63)	Reserved for profiles					To be defined in profiles
…						
0x4000 to 0x4FFF (16384 to 20479)	Profile specific Index					To be defined in profiles
Key M = mandatory; O = optional; C = conditional						

6.3 Basic Data structure rules

For the layout of Process Data or parameter structures the following rules shall be observed

- Combinations of boolean data are right-aligned
- Preferred data types are (in prioritized order)
    - IntegerT with 8, 16, and 32 bit length
    - UIntegerT with 8, 16, and 32 bit length
    - Float32T
    - Any other data type defined in Annex E in [1]
- Any data structure definition shall observe the alignment rules in [3]

6.4 Profile EventCodes constraints

Annex D in [1] and [2] reserves the EventCode range from 0xB000 to 0xBFFF for profiles.

If any profile defines EventCodes, also the trigger and behaviour accompanied to this event shall be defined.

6.5 Profile IODD constraints

The following naming conventions shall be considered for all additional profile specifications :

– The displayed name may be human-readable and can be abbreviated to shorten the name
– Commands should be named in imperative form
– Menu groups should order functionalities in a menu with a comprehensive title of the menu
– SingleValues shall be human-readable and are abbreviated to shorten the name
– The predefined name shall always be used in any Device specific IODD

– A vendor/manufacturer specific extension can be added to the predefined name in order to enable vendor specific explanations even in different languages, these shall be separated by " – "

– For each menu entry the corresponding IODD menu section like IdentificationMenu, ParameterMenu, ObservationMenu, or DiagnosisMenu shall be defined.

– The availability in the different user roles shall be defined for all menu entries

7 Rules and constraints for developing IO-Link profile Devices

Within this clause the general rules and constraints for the design of IO-Link Devices with support of profiles are defined.

7.1 Constraints for developing IO-Link Devices

When designing a new IO-Link Device the designer should consider the existing Device Profiles available on www.io-link.com. To offer a potential user a maximum of benefits as defined in 5.6 a maximum of CommonApplicationProfiles, DeviceProfiles or at least FunctionClasses should be supported by the new Device.

If the Device does not fit exactly into the restricted Profiles, the designer has to consider between an easy usage with DeviceProfiles and on the other hand special capabilities which are not commonly used. In simple words: does the special characteristic or behavior justify a device without standardized or profiled functionalities.

7.2 How to select the appropriate Profile functions

At first the CommonApplicationProfiles should be considered, they define base functionalities corresponding to the IO-Link system itself.

It is highly recommended to support the Identification & Diagnosis profile according 8.2 to harmonize the core functionality of all IO-Link Devices.

As a next step the specific DeviceProfiles should be considered, they define specific functionalities for the specific types of Device like Sensor or Actuator. As a last step the remaining functionalities should be checked if they fit into one of the FunctionClasses. The support of the FunctionClasses only is just an exit strategy, the best way is to use the CommonApplicationProfiles or DeviceProfiles as a whole.

7.3 Identification of supported Profiles

It is highly recommended to provide the supported Device Profiles by mentioning them with their associated Profile Characteristic Name or FunctionClass Name in the technical documentation and in marketing brochures. This enables the customer to identify these profiled Devices amongst others and allows to identify the standardized features of a particular Device.

8 Identification and Diagnosis (I&D)

8.1 Overview

It is very important to provide all necessary identification and diagnosis information in a unique manner and with the same contents to interpret.

As [1] specifies the required objects as optional, this CommonApplicationProfile specifies these parameters as mandatory for the profile Devices

8.2 Identification and Diagnosis Profile

Table 8 provides an overview of the FunctionClasses for Identification and Diagnosis. Since there are no options only the ProfileIdentifier shall be listed in the ProfileCharacteristic Index, see 5.4.

Table 8 – Identification and Diagnosis Device profile

ProfileID	Profile characteristic name	Function Classes		
0x4000	Identification and Diagnosis	0x8000	Device Identification	See A.2
		0x8003	Device Diagnosis	See A.4
		0x8002	Process Data Mapping	See A.3
		0x8100	Extended Identification	See A.5

8.3 Extension of Identification and Diagnosis

The DeviceProfile Identification and Diagnosis can be accompanied by the FunctionClass VisualLocalization. The possible extension is defined in Table 9.

Table 9 – Extension for I&D

ProfileType	Possible extension	
I&D	Visual localization (0x8101)	See A.6

8.4 Proxy Function Block for Identification and Diagnosis

To ease the integration in Run-Time systems like PLCs, an appropriate Function Block is specified in C.1. The Function Block reads or writes identification or diagnosis data from the Device and shows the status of the Function Block. The information is provided in a way an operator can use directly in any PLC program for further handling. All specific action is taken without any required specific knowledge of the operator.

Annex A
(normative)

FunctionClasses

A.1 Overview

Table A.1 provides an overview of the defined FunctionClasses within this document.

Table A.1 – Overview of FunctionClasses

FunctionClass	Name	Reference / Clause
[0x8000]	Device Identification	A.2
[0x8002]	Process Data Mapping (PDV)	A.3
[0x8003]	Device Diagnosis	A.4
[0x8100]	Extended Identification	A.5
[0x8101]	Visual Localization	A.6

A.2 Device identification objects [0x8000]

The FunctionClass 0x8000 defines some optional parameters as mandatory for profile Devices. These are

- ProductID
- FirmwareRevision
- ApplicationSpecificTag

The ProductID and the FirmwareRevision are unchanged to the definition in clause B.2.11 and B.2.15 of [1]. The parameter ApplicationSpecificTag defined in clause B.2.16 in [1], is defined in this FunctionClass with the maximum size of 32 octets to get a maximum reusability over all profile Devices.

A.3 Process Data mapping (PDV) [0x8002]

A.3.1 Overview

Depending on the particular profile type, a Device arranges binary information and/or more complex data structures for the cyclic transmission to and/or from the Master via SDCI in a so-called "PDinput data stream" and/or "PDoutput data stream".

A.3.2 Process data description

The profile Device provides an input Process Data description (PDInputDescriptor) indicating the composition (mapping) in the "PDinput data stream" and/or a similar output Process Data description (PDOutputDescriptor). The coding of the corresponding parameters is defined in B.5.

The content of the process variable descriptors PVinD or PVoutD shall be available via the corresponding Index.

Each part of the process data stream is described unambiguously via its coding in PVinD and/or PVoutD. Subsequent Boolean variables can be described within one descriptor. The following information shall be provided within a PVinD or PVoutD respectively:

- the data type (DataType) of the particular process variable. "Set of BoolT" describes combined independent Boolean values

- the length of the data type (TypeLength) in bit, for example 6 for UIntegerT6
- the bit offset (Bit offset) as the beginning of the variable in the data stream

The user program within a controller (e.g. PLC) can thus read this information.

A.4 Diagnosis [0x8003]

The FunctionClass 0x8003 defines some optional parameters as mandatory for profile Devices. These are

- DeviceStatus
- DetailedDeviceStatus

Both parameters are unchanged to the definition in clause B.2.20 and B.2.21 in [1].

As already described in [1], the Events, the DetailedDeviceStatus and the DeviceStatus are interconnected. Whenever an Event appears, triggered by the device application, the DetailedDeviceStatus contains this Event as long as it disappears, see B.2.21 in [1]. The resulting DeviceStatus of each predefined Event is defined in Table D.1 in [1], the highest DeviceStatus value of all current active Events determines the content of the DeviceStatus.

A.5 Extended Identification [0x8100]

The FunctionClass 0x8100 defines extended identification which can be used e.g. for localization in a plant, machine, etc in any readable location format. Another parameter can contain a detailed description of the specific Device like “Hot water valve”, etc. Both parameter provide only a sequence of characters without any interpretation within the Device itself.

The parameter

- FunctionTag
- LocationTag

defined in B.6 provide the necessary non-volatile memory space.

Additionally the standardized parameter SerialNumber and HardwareRevision are set to mandatory.

A.6 Locator [0x8101]

The FunctionClass 0x8101 defines the visual localization via an optical indicator within the Device which can be used during setup of the installation to localize a specific Device among other Devices.

Figure A.7 shows the state machine of the optical indication of a Device.

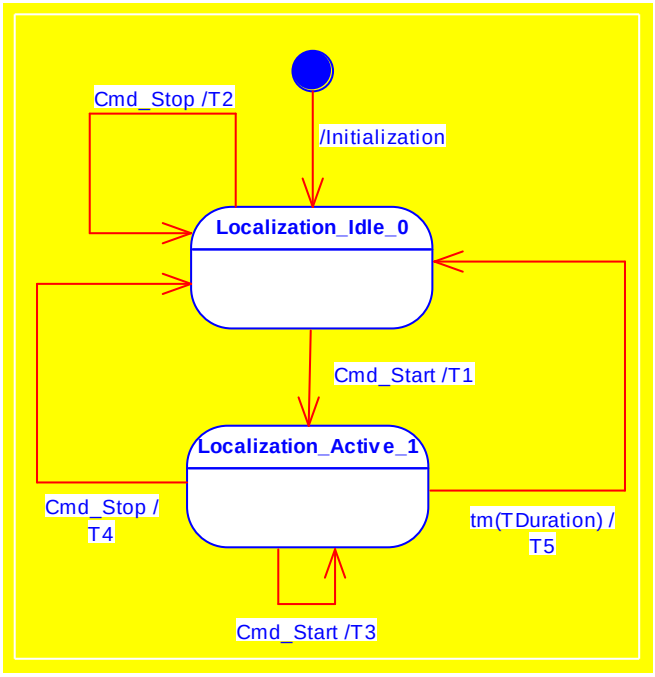


Figure A.7 – State machine for optical indication

Table A.2 shows the state transition tables for the optical indication.

Table A.2 – State transition tables for optical indication

STATE NAME	STATE DESCRIPTION
Localization_Idle_0	In this state the Device is waiting for a SystemCommand LocatorStart and performs no specific optical indication
Localization_Active_1	In this state the Device performs the specific optical indication according to Figure A.8 to allow easy identification of this Device until the LocatorStop command is received or the timeout elapses

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
Initialization	–	0	-
T1	0	1	Start optical indication, start timer
T2	0	0	-
T3	1	1	Restart timer
T4	1	0	Stop optical indication, stop timer
T5	1	0	Stop optical indication

INTERNAL ITEMS	TYPE	DEFINITION
Cmd_Start	Service	Reception of ISDU with SystemCommand containing LocatorStart
Cmd_Stop	Service	Reception of ISDU with SystemCommand containing LocatorStop
TDuration	Time	See Table A.3
timer	Variable	Timeout timer

The indication of the Device localization follows the timing shown in Figure A.8.

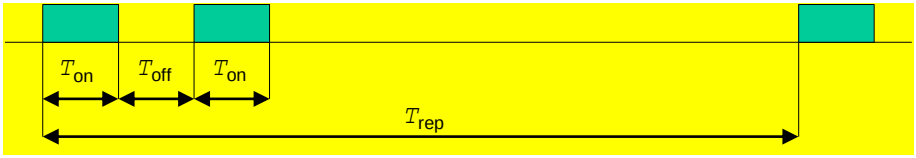


Figure A.8 – Device optical indicator timing for localization

Table A.3 defines the timing for the optical indication of Devices.

Table A.3 – Timing for the optical indication

Timing	Minimum	Typical	Maximum	Unit
T_{rep}	900	1000	1100	ms
T_{on}	90	100	110	ms
T_{off}	90	100	110	ms
$T_{Duration}$	9	10	11	min

The sequence of repeated double flashing is started after reception of the SystemCommand "LocatorStart", see Table B.2, and lasts for the time $T_{Duration}$. After this time or by reception of a SystemCommand "LocatorStop" the sequence will stop. The timeout can be retriggered to $T_{Duration}$ by reception of another SystemCommand "LocatorStart".

The sequence shows a double flashing to avoid interference with other indications like output indicators.

It is on behalf of the Device designer to select the means for the optical indication like LEDs or graphical displays. The intended effect should be as outstanding as possible to enable the user to identify the selected Device as easy as possible. The standardized recognition of the IO-Link Device visual localization is based on the double flashing, and not on any specific color or symbol on any display.

The implementation of Visual Localization is recommended for all Device which have controllable optical indications like LEDs or means like displays of any kind.

Annex B
(normative)

Profile relevant Device parameters

B.1 Overview

The manufacturer can provide Subindex access to objects with RecordItems, the Common Profile specification does not define this behaviour. Any overall usable software shall always use the Subindex 0 access instead as this access is granted by any Device.

The persistence or volatility of the objects is stated for each object.

The Device reset option rules defined in clause 10.7.1 in [1] shall be considered and reset all Device parameters to their default value.

The profile relevant Device parameters are specified in [1]. An overview is shown in Table B.1.

Table B.1 – General profile relevant Device parameters

Index (dec)	Object name	Access	Length	Data type	M/C	Remark
…						
0x0002 (2)	SystemCommand	W	1 octet	UIntegerT	M	See B.2
0x000D (13)	ProfileCharacteristic	R	variable	ArrayT of UIntegerT16	M	See B.4 See clause B.2.5 in [1]
0x000E (14)	PDInputDescriptor	R	variable	ArrayT of OctetStringT3	C	Conditional on availability of PDIn See B.5 and clause B.2.6 in [1]
0x000F (15)	PDOutputDescriptor	R	variable	ArrayT of OctetStringT3	C	Conditional on availability of PDOut See B.5 and clause B.2.7 in [1]
…						
0x0010 (16)	VendorName	R	max. 64 octets	StringT	M	See clause B.2.8 in [1]
…						
0x0012 (18)	ProductName	R	max. 64 octets	StringT	M	See clause B.2.10 in [1]
0x0013 (19)	ProductID	R	max. 64 octets	StringT	M	See clause B.2.11 in [1]
…						
0x0015 (21)	SerialNumber	R	max. 16 octets	StringT	M	See clause B.2.12 in [1]
0x0016 (22)	HardwareRevision	R	max. 64 octets	StringT	M	See clause B.2.14 in [1]
0x0017 (23)	FirmwareRevision	R	max. 64 octets	StringT	M	See clause B.2.15 in [1]
0x0018 (24)	ApplicationSpecific-Tag	R/W	32	StringT	M	See B.2 See clause B.2.16 in [1]
0x0019 (25)	FunctionTag	R/W	32	StringT	M	See B.6
0x001A (26)	LocationTag	R/W	32	StringT	M	See B.6
…						
0x0024 (36)	DeviceStatus	R	1 octet	UIntegerT	M	See B.7 See clause B.2.20 in [1], default value is "0".

Index (dec)	Object name	Access	Length	Data type	M/C	Remark
0x0025 (37)	DetailedDeviceStatus	R	variable	ArrayT of OctetStringT3	M	See B.7 See clause B.2.21 in [1], default values are "0". Contains a minimum of one Event entry.
Keys M = mandatory; C = conditional						

B.2 System Commands

This clause describes the SystemCommands which are used by the CommonProfile. The availability of the commands specified in Table B.2 is depending on the support of the corresponding FunctionClass.

Table B.2 – Conditional "SystemCommand"

Command (hex)	Command (dec)	Command name	Definition
0x7E	126	Locator Start	Applicable for Visual Localization, see A.6
0x7F	127	Locator Stop	

The reaction to the SystemCommands "FlashStart" and FlashStop" shall meet the requirement in clause 10.3.7 in [1] with an immediate return after checking the availability of the command.

B.3 Identification parameters

As identification parameters ProductID, FirmwareRevision, and ApplicationSpecificTag are defined as mandatory, the structure and coding is defined in clauses B.2.11, B.2.15 and B.2.16 in [1].

As a difference the parameter ApplicationSpecificTag is defined with the maximum size of 32 octets as defined in Table B.3. The object shall be stored persistent, follows the Device reset option rules defined in clause 10.7.1 in [1]. and handled by the DataStorage mechanism.

Table B.3 – Definitions for identification data objects

Index (dec)	Subindex	Offset	Access	Object name	Length (octets)	Data Type
0x0018 (24)	n/a	n/a	R/W	ApplicationSpecificTag	32	StringT
Keys R = read W = write						

B.4 ProfileCharacteristics parameter

This clause describes the parameter which contains the ProfileIdentifier of the supported Device profiles and FunctionClasses.

Table B.4 defines the structure of the parameter ProfileCharacteristics.

Table B.4 – Parameter “ProfileCharacteristics"

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x000D (13)	1	(n-1) * 2	R	ProfileIdentifier 1	16 bit	UIntegerT16
	…					
	n	0	R	ProfileIdentifier n		
Key n = number of supported ProfileIdentifier						

See 5.4 for further content rules regarding the ProfileIdentifier.

B.5 Process data structure descriptors

This clause describes the parameters which contain the structure information of the process data input and output. Each part of the process data is described with an PVinD or PVoutD. The generic rules for defining the structures are described in A.3, specific process data structure definitions for ProfileIDs are defined in the corresponding profile specification like [7].

B.5.1 Coding of PVinD and PVoutD

Table B.5 shows the coding of each process variable to be placed in the descriptors using PVinD or PVoutD.

Table B.5 – Coding of PVinD or PVoutD

Location	Item	Coding
Octet 1	DataType	0: OctetStringT 1: Set of BoolT 2: UIntegerT 3: IntegerT 4: Float32T 5: StringT 6: TimeT 7: TimeSpanT 8 to 127: reserved: 128 to 255 reserved for profiles
setOctet 2	TypeLength	0: 256 Bit 1 to 255: 1 to 255 Bit
Octet 3	Bit offset	0 to 255 Bit

NOTE The abstract notation of for example a PVinD is: DataType.TypeLength.Bit_offset
Set of BoolT describes a combination of one or more BooleanT without gaps

Any profile may define their own complex DataTypes if necessary. Reuse of same DataType definitions is mandatory.

B.5.2 PDInputDescriptor

Profile Devices with process input data shall use the standard Device parameter "PDInputDescriptor" in Index 0x000E to provide the description information according to Table B.5.

Table B.6 defines the structure of the PDInputDescriptor regarding the offset and Subindex layout.

Table B.6 – Structure of "PDInputDescriptor"

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
	1	(n-1) * 3	R	PVinD 1	24 bit	OctetStringT3

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x000E (14)	…					
	n	0	R	PVinD n	24 bit	OctetStringT3
Key	n = number of provided descriptors					

590

591 ### B.5.3 PDOutputDescriptor

592 Profile Devices with process data output shall use the standard Device parameter "PDOut-
593 putDescriptor" in Index 0x000F to provide the description information according to Table B.5.

594 Table B.7 defines the structure of the PDOutputDescriptor regarding the offset and Subindex
595 layout.

596 **Table B.7 – Structure of "PDOutputDescriptor"**

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x000F (15)	1	(n-1) * 3	R	PVoutD 1	24 bit	OctetStringT3
	…					
	n	0	R	PVoutD n	24 bit	OctetStringT3
Key	n = number of provided descriptors					

597

598 ## B.6 Extended Identification parameters

599 This clause defines the extended identification parameters which can be used for overall local-
600 ization and identification of any Device.

601 The content is not predefined, the customer can provide any visible string conform to his own
602 naming rules. The R/W parameters “FunctionTag” and “LocationTag” shall be stored persistent,
603 follows the Device reset option rules defined in clause 10.7.1 in [1], and handled by the DataS-
604 torage mechanism. As default it is recommended to fill the parameter “FunctionTag” and “Lo-
605 cationTag” with “***”.

606 Table B.8 defines the structure of the parameters.

607 **Table B.8 – Parameter Extended Identification**

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x0015 (21)	n/a	n/a	R	SerialNumber	Max 16 octets	StringT
0x0016 (22)	n/a	n/a	R	HardwareRevision	Max 64 octets	StringT
0x0019 (25)	n/a	n/a	R/W	FunctionTag	32 octets	StringT32
0x001A (26)	n/a	n/a	R/W	LocationTag	32 octets	StringT32
Key	n/a = not applicable					

608

609 ## B.7 Diagnosis parameters

610 The structure and coding is defined in clauses B.2.20 and B.2.21 in [1].

611 Table B.9 defines the structure of the DetailedDeviceStatus regarding the offset and Subindex
612 layout.

613 **Table B.9 – Structure of "DetailedDeviceStatus"**

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x0025 (37)	1	(n-1) * 3	R	Event 1	24 bit	OctetStringT3
	…					
	n	0	R	Event n	24 bit	OctetStringT3
Key	n = number of provided Event entries					

614

Annex C
(normative)

Function block definitions

C.1 Overview

This annex contains the proxy Function Blocks supporting the CommonApplicationProfileID.

The specification is based on IEC 61131-3 definitions.

As there are still some differences between the existing systems regarding the PLC system or fieldbus, the system dependent features are marked and have to be defined for each system separately.

The proxy Function Block is asynchronous, which means that the Function Block is triggered and after accomplishing the functionality the results are available.

C.2 Proxy function block (FB) for identification and diagnosis

The layout of the proxy function block for the CommonApplicationProfile Identification and Diagnosis (0x4000) which supports the FunctionClasses DeviceIdentification (0x8000), DeviceDiagnosis (0x8003), and ExtendedIdentification (0x8100) is shown in Figure C.1.

The input and output data types of the proxy function block correspond to those of IEC 61131-3 (PLC programming languages).

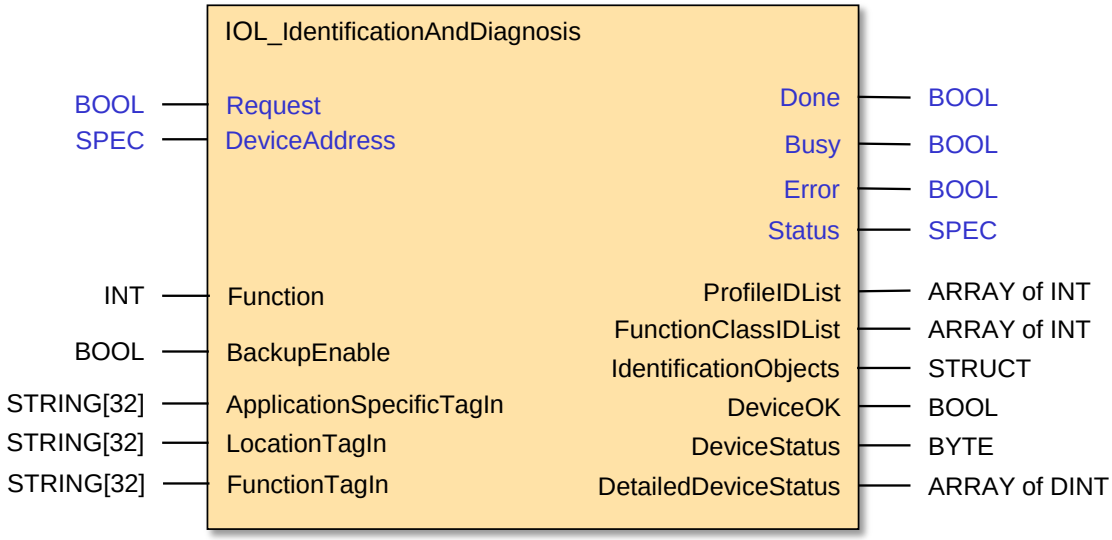


Figure C.1 – Proxy FB for Device Identification and Diagnosis

Table C.1 defines the variables of this proxy FB.

Table C.1 – Variables of "IOL_IdentificationAndDiagnosis" FB

Variable	PLC Type	Description
Inputs		
Request [a]	BOOL	A trigger causes the function selected with variable Function to be executed
DeviceAddress [a]	SPEC [b]	This variable depends on the individual fieldbus address mechanism of an SDCI Device at an SDCI Master port (see SDCI integration specification of a particular fieldbus)

Variable	PLC Type	Description
Function	INT	This variable selects the functionality to be triggered by a Request 0 = no_func A Request is neglected, no function is executed 1 = rd_all A Request starts the read back of current identification and diagnostic parameter values from the Device. 2 = rd_diag A Request starts the read back of current diagnostic parameter values by reading DeviceStatus and DetailedDeviceStatus from the Device. 3 = wr_ident A Request causes a previously applied value for ApplicationSpecificTagIn, LocationTagIn, and FunctionTagIn to be written to the Device
BackupEnable	BOOL	This variable configures the behavior of the FB in case of the requested function wr_ident. "true" = enabled The backup mechanism is triggered by the FB by issuing the SystemCommand ParamDownloadStore after wr_ident. "false" = disabled The backup mechanism is not triggered by the FB
ApplicationSpecificTagIn	STRING[32]	See Device parameter in clause B.2.16 in [1]
LocationTagIn	STRING[32]	See B.6
FunctionTagIn	STRING[32]	See B.6
Outputs		
Done [a]	BOOL	The signal is set, if the FB has completed a requested operation.
Busy [a]	BOOL	The signal is set, if the FB is executing a requested operation
Error [a]	BOOL	The signal is set, if an error occurred during execution of a requested operation.
Status [a]	SPEC [b]	The value represents the current status of the FB operation and executed functions. The content is system specific and contains the status information
ProfileIDList	ARRAY of INT	List of ProfileIDs supported by the Device
FunctionClassIDList	ARRAY of INT	List of FunctionClassIDs supported by the Device
IdentificationObjects	STRUCT	Structured list of identification objects, see Table C.2 for further details
DeviceOK	BOOLEAN	The signal is set when no further diagnosis info is available, it is false when further information is available at DeviceStatus and DetailedDeviceStatus
DeviceStatus	BYTE	See Device parameter in clause B.2.20 in [1]
DetailedDeviceStatus	ARRAY of DWORD	This parameter contains the type casted values from the Device parameter defined in clause B.2.21 in [1]
Key a: This variable name may be adapted to the PLC specific naming guide lines b: SPEC represents the applicable data type for this specific parameter, this may vary over different PLC systems		

637

638 The lists ProfileIDList, FunctionClassIDList, and DetailedDeviceStatus are set to 0 by default
639 and are overwritten by data read from the Device.

640 The structured information in the variable IdentificationObjects is specified in Table C.2.

641 The default value is provided when the corresponding parameter is not already read from the
642 Device or not available in the Device.

643 **Table C.2 – Elements of the IdentificationObjects**

Name	PLC Type	Default	Remark
VendorID	WORD	00 00	See clause B.1.8 in [1]
DeviceID	DWORD	00 00 00 00	See clause B.1.9 in [1]
VendorName	STRING[64]	“na”	See clause B.2.8 in [1]
VendorText	STRING[64]	“na”	See clause B.2.9 in [1]
ProductName	STRING[64]	“na”	See clause B.2.10 in [1]
ProductID	STRING[64]	“na”	See clause B.2.11 in [1]
ProductText	STRING[64]	“na”	See clause B.2.12 in [1]
SerialNumber	STRING[16]	“na”	See clause B.2.13 in [1]
HardwareRevision	STRING[64]	“na”	See clause B.2.14 in [1]
FirmwareRevision	STRING[64]	“na”	See clause B.2.15 in [1]
ApplicationSpecificTag	STRING[32]	“na”	See clause B.2.16 in [1]
LocationTag	STRING[32]	“na”	See B.6
FunctionTag	STRING[32]	“na”	See B.6

644

645 C.3 Implementation rules for "IOL_IdentificationAndDiagnosis" FB

646 The access of acyclic parameters in IO-Link Devices requires the usage of BlockParametriza-
647 tion according to 10.3.5 in [1] by following these steps

648 – Hidden SystemCommand "ParamUploadStart" or "ParamDownloadStart" depending on di-
649 rection

650 – Perform acyclic parameter access

651 – SystemCommand "ParamUploadEnd" or "ParamDownloadEnd" depending on direction

652 – SystemCommand "ParamDownloadStore" if parameters were written and BackupEnable =
653 "true"

Annex D
(normative)

IODD definition and rules

D.1 Overview

The objective of the Common Profile specification is to ease the integration of Devices and to provide additional information in a uniformed manner. The integration is part of the specialised profile specifications, the uniformed information about profile support is part of this clause. As the parameter and the behavior is specified the look and feel of the Devices should also be harmonized, otherwise the appearance of the same profile is different between different manufacturer.

To achieve a common look and feel, the IODD content has to be defined as well. This clause includes the rules for the naming conventions and menu layout.

As an example end the Identification & Diagnosis profile IODD rules are specified at the end of this annex.

D.2 Constraints and rules

The following naming conventions shall be considered for all additional profile specifications :

– The predefined name shall always be used in any Device specific IODD
– A vendor/manufacturer specific extension can be added to the predefined name in order to enable vendor specific explanations even in different languages, these shall be separated by " – "
– The menu entries shall be located in the specified menu section
– The menu entries shall not be altered in layout and structure
– All StringT variables shall be encoded in UTF-8

D.3 Name definitions

D.3.1 Profile type characteristic names

The profile characteristic name defined in 8.1 and in separated profile specifications shall be used whenever any profile functionality is referenced in the IODD.

D.4 IODD Menu definitions

D.4.1 Overview

Examples for layouts of Port and Device configuration tools are shown in clause 11.7 in [1].

Within these examples the IODD defines the parameter layout of the connected device. In this clause the object and parameter layout of the ProfileIdentifier is specified.

To harmonize the layout, the parameter shall be referenced in the menu. If RecordItems are available, these shall be referenced in the menu. The shown variable figures and the SingleValues are examples.

It is mandatory to provide all definied parameters in the defined order, it is on behalf of the manufacturer to group them by menu groups or extend by further parameters.

D.4.2 Explanation of used object layout

Figure D.1 shows the basic layout objects to describe the look of the profile parameters in any IODD based tooling.

The content description is placed at the corresponding positions.

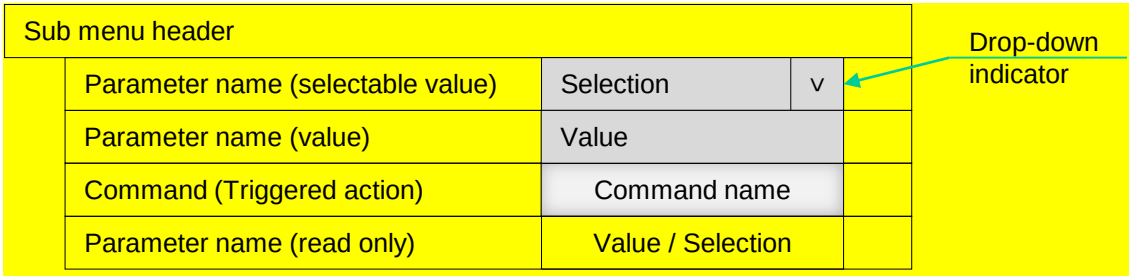


Figure D.1 – IODD object layout description

D.4.3 Menu structure of the Device Diagnosis parameter

In Figure D.2 the menu structure Device Diagnosis according to 8.2 and A.4 is specified, it shall be located in the Diagnosis section of the menu.

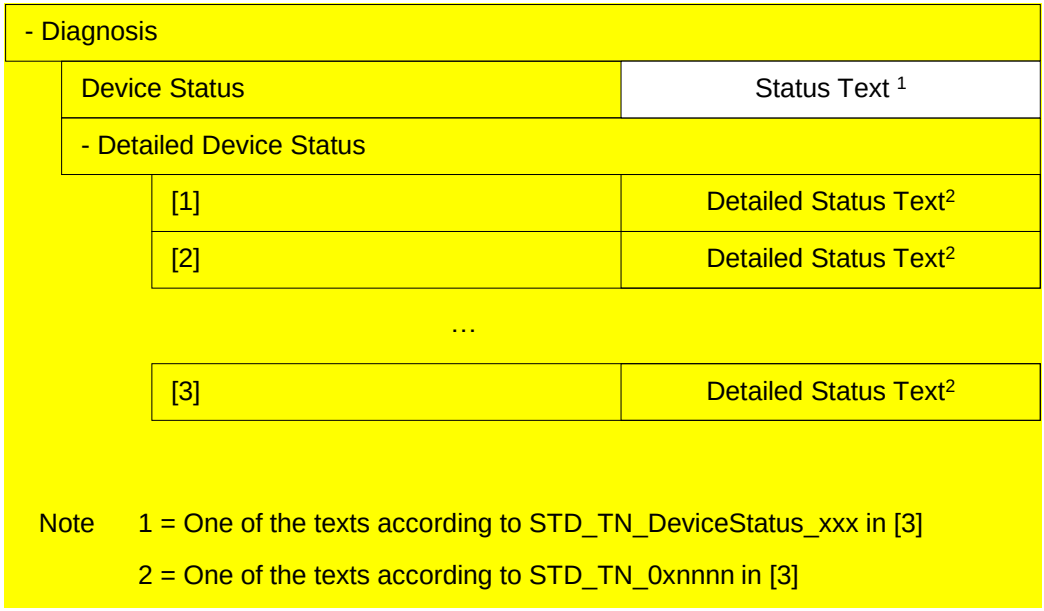


Figure D.2 – Menu Profile Diagnosis

The texts are already defined in [3] as standard parameter texts.

D.4.4 Menu structure of the Device Identification parameters

In Figure D.3 the menu structure Device Identification according to 8.2, A.2, and A.5 is specified, it shall be located in the Identification section of the menu.

- Identification	
Vendor Name	Text [1]
Product Name	Text [1]
Product Text	Text [1]
Product ID	Text [1]
Serial Number	Text [1]
Hardware Version	Text [1]
Firmware Version	Text [1]
Application Specific Tag	*** [2]
Function Tag	*** [2]
Location Tag	*** [2]

Note 1 = Texts read out from Device or default from IODD

2 = Default entry or changed name by customer

Figure D.3 – Menu Profile Identification

D.4.5 Menu structure of the Localization functionality

In Figure D.4 the menu structure Device localization according to A.6 is specified, it shall be located in the identification section of the menu.

- Identification	
…	
Standard Command	Locator Start
Standard Command	Locator Stop
…	

Figure D.4 – Menu Profile Localization

Annex E
(normative)
Profile testing and conformity

E.1 General

E.1.1 Overview

It is the responsibility of the vendor/manufacturer of a profile Device to perform a conformity testing according to the test specification [6] and to provide a document similar to the manufacturer declaration defined in [1] or based on the template downloadable from the IO-Link website (www.io-link.com).

E.1.2 Issues for additional testing of profile Devices

It is recommended to perform a conformity test based on the test specified in F.1 and corresponding test specifications of the implemented profiles.

E.1.3 General business rule extensions for the IODD Checker

To achieve consistency and conformity of the profiled Devices to the claimed profiles, the business rules of the checker are extended covering the profile requirements.

The rule extensions are generic to suit the profile requirements and based on IODD snippets which are provided together with the corresponding profile specifications.

Each profile provides xml based files containing IODD related snippets, which can be copied and adapted to create well formed Device IODDs. These xml files contain xml elements following the rules of [3] which are extended by the following attributes. These specific extensions must be removed when copying the parts into a specific Device IODD.

E.2 Rules for IODD snippet files

This clause defines the layout und and content rules which apply to the IODD snippet files which support the design and test of profiled Devices.

The base rules specify the layout and strategy when an IODD snippet file is generated.

The extensions by attributes are specified in the later clauses, together with the applicable checking rules.

E.2.1 Base rules

The following rules apply for a profile describing IODD snippet file:

- The xml-file shall be formatted with "Pretty-Print" to provide a common layout
- The attribute "excludedFromDataStorage" shall be predefined whenever applicable
- 'ids' shall be unique in the scope of the specific profile predefinition and use unique prefixes throughout a profile specification
- Any elements like datatypes or texts which are referenced by another element do not provide profileConstraints, the reference is used to derive the data from snippet and IODD, which has to match according the check rules
- White spaces can improve the readability, but will be ignored by the parser

E.2.2 Attribute profileConstraints

The attribute profileConstraints specifies for which ProfileID the element is applicable and controls the scope of check.

The following syntax and behavioral rules apply:

- Syntax: profileConstraints="<profileID1>, <profileID2>, <profileID3> AND <profileID4>, ..."

- An omitted attribute profileConstraints defines a positive matching
- The attribute may contain several ProfileIDs
- The list of ProfileIDs or expressions are separated by comma
- Dependencies between two ProfileIDs are indicated by the logical expression 'AND'
- The attribute profileConstraints applies to the entire element including sub-elements
- The attribute profileConstraints can be used in a sub-element in order to exclude this sub-element from checking. However, only a subset of the ProfileIDs of the next higher hierarchical element is allowed

E.2.3 Attribute checkAttributes

The attribute checkAttributes defines the checking rules for the attributes of this element, the possible values are specified in Table E.1.

Table E.1 – Rules of checkAttributes

Value name	Rule description
exact	All predefined attributes shall exist as predefined, additional attributes are prohibited
atLeast	All predefined attributes shall exist as predefined, additional attributes are allowed
option <attribute>	As "atLeast", but only specific attributes are allowed. Several attributes can be referenced by multiple instantiations in the form "option <attribute1>, option <attribute2>, ..."
startsWith	This rule enforces a predefined beginning of an attribute's value.
contains	This rule enforces the coverage of predefined values within an attribute's value.

The following additional syntax and behavioral rules apply:

- Syntax: checkAttributes="<rule1>, <rule2>, ..."
- The list of checking rules are separated by comma
- The rule "exact" is predefined when the attribute is omitted
- The rules "exact", "atLeast", and "option" cannot be combined
- The check does not only cover the attribute's presence, but also performs a check of the content. A predefined value of "#tbd#" indicates a wildcard for any allowed content

E.2.4 Attribute checkElement

The attribute checkElement is an optional attribute in each element to check the order or presence of subelements, the possible values are specified in Table E.2.

785 **Table E.2 – Rules of checkElement**

Value name	Rule description
exact	All predefined sub elements shall exist as predefined, additional sub elements are prohibited
atLeast	All predefined sub elements shall exist as predefined, additional sub elements are allowed
atLeastSequence	The predefined sub elements shall exist as predefined and within the defined order without gaps. This rule is only applicable for the elements 'ProcessDataRef' and 'Menu' within the section 'UserInterface'
maxOccurs <n>	This type of element is allowed with a maximum number of instances of n
minOccurs <n>	This type of element is required with a minimum number of instances of n

786

787 The following additional syntax and behavioral rules apply:

788 • Syntax: checkElement="<rule1>, <rule2>, …"

789 • The list of checking rules are separated by comma

790 • The rule "exact" is predefined when the attribute is omitted

791 • The rules "exact", "atLeast", and "atLeastSequence" cannot be combined

792

793 ### E.2.5 Attribute contextConstraint

794 The attribute contextConstraint is an optional attribute used within the section 'UserInterface'
795 only. It enforces that an element is referenced within the indicated menu group. The attribute
796 has no default value when omitted.

797 Syntax: contextConstraint="<menugroup>"

798 Permissible values for menugroup are:

799 • IdentificationMenu

800 • ParameterMenu

801 • ObservationMenu

802 • DiagnosisMenu

803

804 ## E.3 Test sequence

805 The following test sequence is performed by the IODD checker as an extended business rule
806 whenever the ProfileCharacteristics are not empty.

807 The sequence of test steps as extension of the IODD checker is specified in Table E.3.

Table E.3 – Test sequence

Test step	Description
1	Read profileCharacteristics from Test-IODD and decompose ProfileIDs
2	Iterate over all available profile snippets files and perform the following tests
2.1	Read profile snippets file
2.2	Perform the following tests on each matching element including the references in /IODevice/ProfileBody/DeviceFunction/Features, /IODevice/ProfileBody/DeviceFunction/VariableCollection, /IODevice/ProfileBody/DeviceFunction/ProcessDataCollection, /IODevice/ProfileBody/DeviceFunction/UserInterface/ProcessDataRefCollection, and /IODevice/ProfileBody/DeviceFunction/UserInterface/MenuCollection Check the presence of the corresponding element in the Test-IODD. In case of failure, create an error log entry and skip tests 2.3 to 2.6 on this element
2.3	Read checkAttribute Perform test on specific element regarding attribute rules. In case of failures, create an error log entry
2.4	Read checkElement Perform test on "minOccurs" and "maxOccurs" of this type of element against elements on the same level, In case of failures, create an error log entry Perform test on structure rules "exact", "atLeast", and "atLeastSequence" against sub-elements (on each level). In case of failures, create an error log entry
2.5	Read contextConstraints Check presence for menu, variable or recorditem reference within the indicated menu group. In case of failures, create an error log entry
2.6	For each reference as "Name textId", "Description textId", "DatatypeRef datatypeId", and "ProcessDataRef processDataId" Get matching elements from snippet and Test-IODD and perform check based on the rules for checkAttributes and checkElement (see 2.3 and 2.4). In case of failure, create an error log entry
3	Perfom action for each snippet file from 2.1 on

The following rules specify the check of the attribute's value constraints:

- any predefined value shall be provided as predefined
- exception: an entry of "#tbd#" is used as a wildcard for any allowed content, it's on behalf of the standard IODD Checker rules to perform any further contextual check
- exception: an entry of #tbd n-m# is used to restrict the allowed value to a range of numbers from n to m
- when checking the attribute excludedFromDataStorage, the presetting of "false" is assumed when the checked IODD does not contain the attribute

Annex F
(normative)

Testing Identification and Diagnosis

F.1 Test case extension for common profile requirements

F.1.1 Ordering of Profile characteristics

Table H.1 defines the test conditions for this test case.

Table H.1 – Ordering of Profile characteristics

TEST CASE ATTRIBUTES	IDENTIFICATION / REFERENCE
Identification (ID)	CMPR_TC_0001
Name	TCD_CMPR_ID_ASCENDID
Purpose (short)	Consistence and ascending order of supported ProfileIDs
Equipment under test (EUT)	Device, IODD; ProfileCharacteristics not empty
Test case version	1.0
Category / type	Parameter verification test; test to pass (positive testing)
Specification (clause)	Figure 4
Configuration / setup	Device-Tester-Unit
TEST CASE	**CONDITIONS / PERFORMANCE**
Purpose (detailed)	Consistency between Device and IODD of supported ProfileIDs and test of ascending order of the ProfileIDs
Precondition	Master and Device in Operate
Procedure	a) Read parameter ProfileCharacteristic and memorize b) Read from IODD /IODevice/ProfileBody/DeviceFunction/Features/@profileCharacteristic
Input parameter	-
Post condition	-
TEST CASE RESULTS	**CHECK / REACTION**
Evaluation	1. Check after step a) for positive result 2. Match ProfileIDs from Device against IODD content 3. Check ProfileIDs for ascending order
Test passed	Evaluation from 1) to 3) without failure
Test failed (examples)	Any failure in 1) to 3)
Results	Consistence and order of ProfileIDs < ok/nok >

F.1.2 Hiding FunctionClasses by ProfileIDs

Table H.2 defines the test conditions for this test case.

Table H.2 – Hiding FunctionClasses of I&D

TEST CASE ATTRIBUTES	IDENTIFICATION / REFERENCE
Identification (ID)	CMPR_TC_0002
Name	TCD_CMPR_ID_HIDDEN_I_D
Purpose (short)	Already incorporated FunctionClasses by higher ProfileID 0x4000 shall not be listed
Equipment under test (EUT)	Device, IODD supporting ProfileId 0x4000
Test case version	1.0
Category / type	Parameter verification test; test to pass
Specification (clause)	Figure 4
Configuration / setup	Device-Tester-Unit
TEST CASE	**CONDITIONS / PERFORMANCE**
Purpose (detailed)	All already by the CommonApplicationProfileID 0x4000 incorporated Function-Classes as 0x8000, 0x8002, 0x8003, and 0x8100 shall not be listed in the Profile-Characteric.
Precondition	Master and Device in Operate
Procedure	a) Read parameter ProfileCharacteristic
Input parameter	–
Post condition	–
TEST CASE RESULTS	**CHECK / REACTION**
Evaluation	a) Check for positive result b) Check for absence of entries of intrinsic FunctionClasses 0x8000, 0x8002, 0x8003, and 0x8100
Test passed	Evaluation from 1) to 2) without failure
Test failed (examples)	Any failure in 1) to 2)
Results	Intrinsic FunctionClasses of I&D hidden < ok/nok >

836 ### F.1.3 Minimum required profile support

837 Table H.3 defines the test conditions for this test case.

838 **Table H.3 – Minimum required profile support**

TEST CASE ATTRIBUTES	IDENTIFICATION / REFERENCE
Identification (ID)	CMPR_TC_0003
Name	TCD_CMPR_ID_LEASTPROFILE
Purpose (short)	Test if required ProfileID 0x4000 is supported
Equipment under test (EUT)	Device, IODD; ProfileCharacteristic not empty
Test case version	1.0
Category / type	Parameter verification test; test to pass
Specification (clause)	Table 8
Configuration / setup	Device-Tester-Unit

839

TEST CASE	CONDITIONS / PERFORMANCE
Purpose (detailed)	Check for availability of ProfileID 0x4000 (Identification and Diagnosis) when at least one other ProfileID is listed
Precondition	Master and Device in Operate
Procedure	a) Read parameter ProfileCharacteristic
Input parameter	–
Post condition	–

840

TEST CASE RESULTS	CHECK / REACTION
Evaluation	a) Check for positive result b) Check for presence of 0x4000
Test passed	Evaluation from 1) to 2) without failure
Test failed (examples)	Any failure in 1) to 2)
Results	Identification and Diagnosis supported < ok/nok >

841

842 ### F.1.4 PDInput-, PDOutputDescriptor parameter

843 Table H.4 defines the test conditions for this test case.

844 **Table H.4 – PDInput-, PDOutputDescriptor parameter**

TEST CASE ATTRIBUTES	IDENTIFICATION / REFERENCE
Identification (ID)	CMPR_TC_0004
Name	TCD_CMPR_ID_PDOUTDESCR
Purpose (short)	Test correct description of PDInput- and PDoutputDescriptor
Equipment under test (EUT)	Device supporting Identification and Diagnosis prfile
Test case version	1.0
Category / type	Parameter verification test; test to pass
Specification (clause)	A.3.1, B.5
Configuration / setup	Device-Tester-Unit

845

TEST CASE	CONDITIONS / PERFORMANCE
Purpose (detailed)	Test availability and correct structure of provided process data description. This test checks for general compliance to the rules in B.5. If any specific profile requires a dedicated process data layout, this is tested by a specific test of this profile.
Precondition	Master and Device in Operate
Procedure	a) Read parameter PDInput Descriptor b) Read PDOutputDescriptor c) Read from IODD /IODevice/ProfileBody/DeviceFunction/ProcessDataCollection
Input parameter	-
Post condition	-

846

TEST CASE RESULTS	CHECK / REACTION
Evaluation	1. If read request from a) was successful than check content against /IODevice/ProfileBody/DeviceFunction/ProcessDataCollection/ProcessData/ProcessDataIn 2. If read request from b) was successful than check content against /IODevice/ProfileBody/DeviceFunction/ProcessDataCollection/ProcessData/ProcessDataIn 3. If both read requests from a) and b) failed than check against empty /IODevice/ProfileBody/DeviceFunction/ProcessDataCollection/ProcessData
Test passed	Evaluation from 1) to 3) without failure
Test failed (examples)	Any failure in 1) to 3)
Results	a) PDIn description available < ok/nok > b) PDOut description available < ok/nok >

847

848 ### F.1.5 Device localization commands

849 Table H.5 defines the test conditions for this test case.

850 **Table H.5 – Device localization commands**

TEST CASE ATTRIBUTES	IDENTIFICATION / REFERENCE
Identification (ID)	CMPR_TC_0005
Name	TCD_CMPR_ID_ LOCATOR
Purpose (short)	Test if localization commands are performed adequate.
Equipment under test (EUT)	Device, IODD supporting Visual Localization (0x8101)
Test case version	1.0
Category / type	Parameter verification test; test to pass
Specification (clause)	A.6
Configuration / setup	Device-Tester-Unit

851

TEST CASE	CONDITIONS / PERFORMANCE
Purpose (detailed)	Test if the SystemCommands for Localization are responded with the correct return code. A Write response (+) shall be returned, in case of “Function temporarily not available” the test should be retried for 3 times. No check on unsupported command behaviour.
Precondition	Master and Device in Operate,
Procedure	a) Write request to SystemCommand with 0x7E "Locator Start" with 3 retries at "Function temporarily not available" b) Wait 5 sec c) Write request to SystemCommand with 0x7E "Locator Start" with 3 retries at "Function temporarily not available" d) Wait 5 sec e) Write request to SystemCommand with 0x7F "Locator Stop" with 3 retries at "Function temporarily not available" f) Wait 2 sec
Input parameter	-
Post condition	-

852

TEST CASE RESULTS	CHECK / REACTION
Evaluation	1. Check after step a) for positive response 2. Check after step c) for positive response 3. Check for visualization scheme on Device 4. Check after step e) for positive response
Test passed	Evaluation in 1) to 4) without failure
Test failed (examples)	Any failure in 1) to 4)
Results	a) Localization control < ok/nok > b) Localization visible < ok/nok >

853

Annex G
(informative)
Information on conformity testing of profile Devices

Information about testing profile Devices for conformity can be obtained from the following organization:

IO-Link Community
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 (0) 721 / 96 58 590
Fax: +49 (0) 721 / 96 58 589
E-mail: info@io-link.com
Web site: http://www.io-link.com

Annex H
(informative)
Process data structure guide line

H.1 Overview

Depending on the particular profile type, a Device arranges binary information (BDC) and/or ProcessDataVariables (PDV) for the cyclic transmission to and/or from the Master via SDCI in a so-called "PDinput data stream" and/or "PDoutput data stream".

Device profiles shall either define the data structures in a new FunctionClass or reference to this generic FunctionClass.

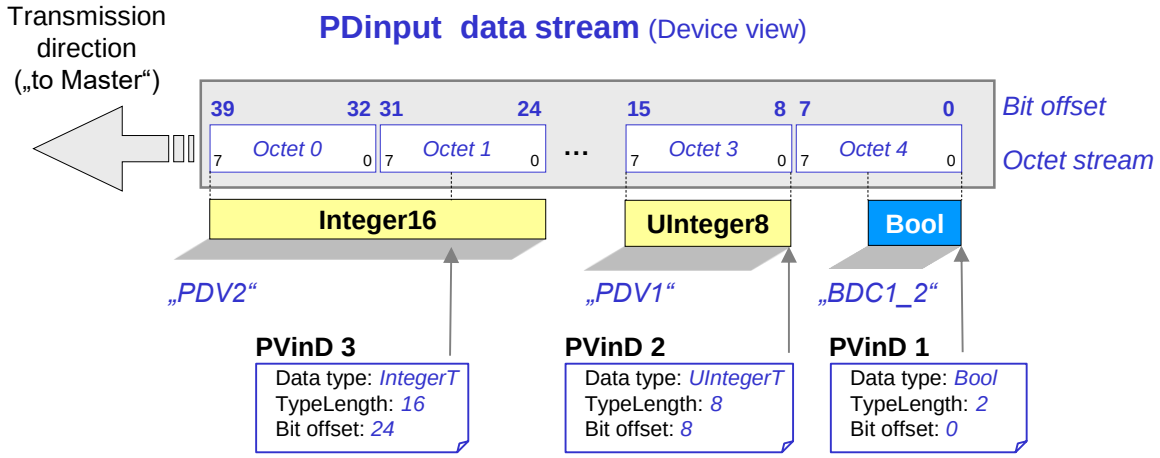


Figure F.1 – Example PDinput data stream

The "PDinput data stream" example shown in Figure F.1 comprises 5 octets (octet 0, 1, 2, 3, 4) to be transmitted to the Master. The profile technology (application) maps BDCs and PDVs into the data stream. The location of each of these data elements within the data stream is described in a process variable descriptor (PVinD, PVoutD). Basis for this description is the "Bit offset" reaching from the last transmitted bit to the first one as defined in Annex E "Data types" in [1].

NOTE From the user program perspective, usage of standard data types such as UInteger16, or Integer16 would be the preferred way of mapping. However, due to performance reasons "packed" data structures cannot be avoided.

H.2 Profile specific PD structures

In order to avoid a large variety of data structures and descriptors and as a consequence complexity, this profile specification specifies and recommends only a few variable descriptions.

H.2.1 General rules for Process Data mapping

It is highly recommended to observe the following rules in order to simplify the programming and to increase performance:

- PDVs of size > 15 bit should be represented in octet granular data types (16, 32, 64), preferably IntegerT
- For data < 16 bit the data type IntegerT should be used that is easily extendable to octet granular data types
- Preferred data lengths are 8, 12, 14, 16, 32, or 64 bit
- PDVs should carry dimensioned measurement values

H.2.2 One or more BDCs (recommended)

It is highly recommended for pure binary profile Devices without additional PDVs to use the data structure demonstrated in Figure F.2. The number of supported BDCs, four in Figure F.2, defines the size of the bit field. The BDCs are right-aligned in ascending order without gaps.

The PVinD in this case is: Set of BoolT.4.0

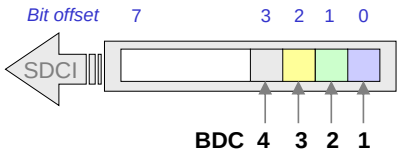


Figure F.2 – Recommended data structure for pure BDCs

H.2.3 One PDV

It is highly recommended for profile Devices with one PDV to use the data structure demonstrated in Figure F.3. The example shows, that a profile Device can cast an 8, 10, or 14 bit value into a UIntegerT16 data type, thereby using only part of the space. In this case the padding bits shall be "0". Variables of type Integer < 16 bit shall also be casted into variables of type IntegerT16. Type casting rules are specified in Annex E.2.3 and E.2.4 in [1].

The PVinD in this case is: UIntegerT.16.0

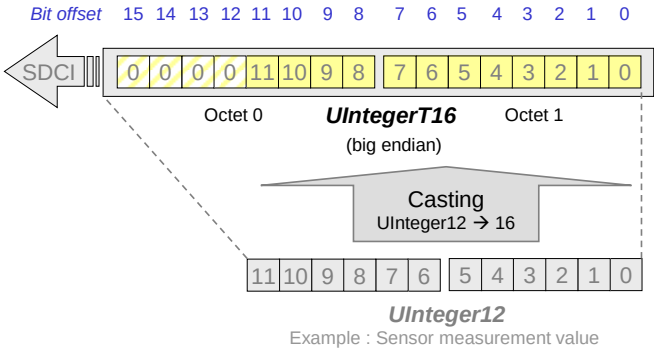


Figure F.3 – Recommended data structure for one PDV

H.2.4 PD lengths up to two octets

Exceptions exist for PD lengths up to two octets. Especially for bit offsets up to 16 other than octet aligned data types may be used ("packed format"). For PD with more than two octets the rules in H.2.7 apply.

H.2.5 One PDV and several BDCs

It is highly recommended for profile Devices with one PDV and one to two BDCs to use the data structure demonstrated in Figure F.4.

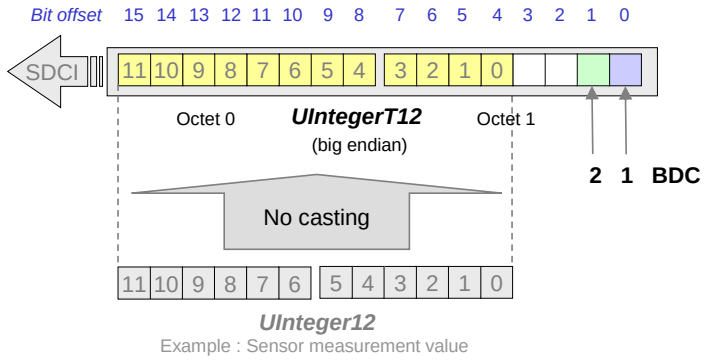


Figure F.4 – Recommended data structure for a PDV and up to two BDCs

The rules in 6.3 shall be observed.

H.2.6 One PDV, several BDCs, and auxiliary variables

It is highly recommended for Smart Sensors with one PDV, one to two BDCs, and auxiliary variables such as qualifiers to use the data structure demonstrated in Figure F.5.

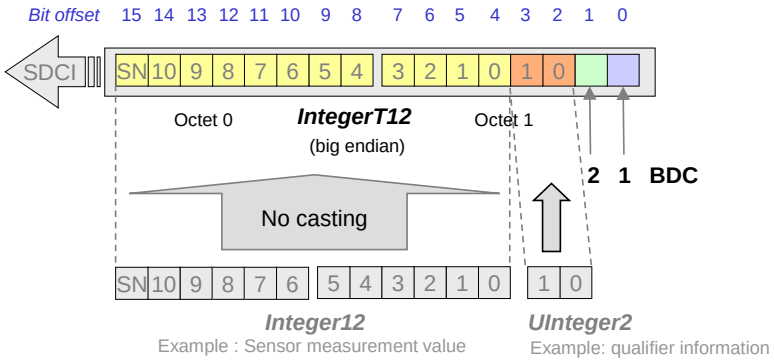


Figure F.5 – Recommended data structure for a PDV, BDCs, and auxiliary variables

The rules in 6.3 shall be observed.

H.2.7 PD lengths larger than two octets

It is highly recommended for profile Devices with 0 or more BDCs, 2 or more PDVs, and manufacturer/vendor specific process data (outside the scope of these profile definitions) to use the data structure demonstrated in the example in Figure F.6. The following rules shall be observed (mandatory):

- Within the first two octets the rules of H.2.4 apply. Especially the BDCs are always starting at bit offset 0.
- All variables starting at bit offset 16 shall be mapped octet aligned. Potential waste of bits is accepted. Variables shall be casted to SDCI data structures if necessary. See clauses Annex E2.3 and E2.4 in [1], for casting rules.

In addition, it is highly recommended to observe the following rules (recommended):

- Best practice for PDVs is the usage of UInteger16 or Integer16 respectively (easier data processing)
- IntegerT to be favored over UIntegerT
- Manufacturer/vendor specific process data can use their own rules. However, it is highly recommended to observe the rules within this profile

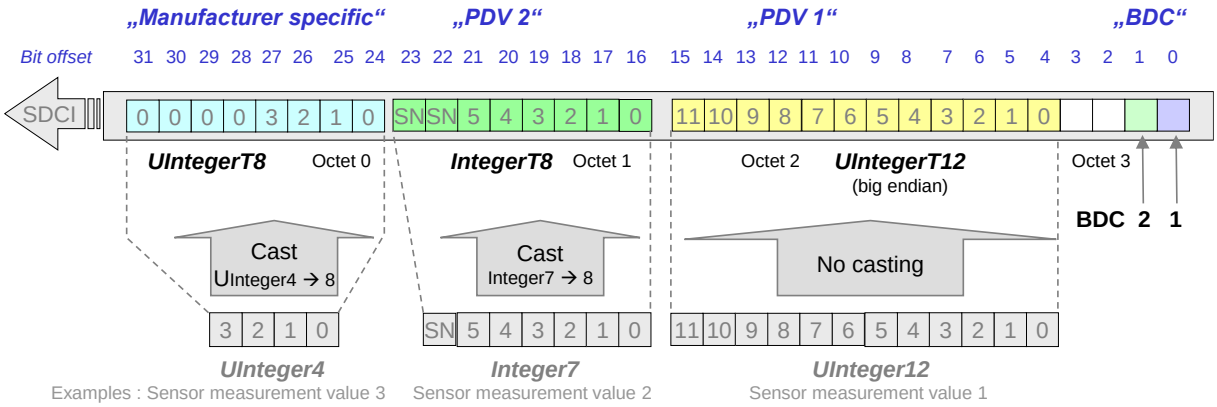


Figure F.6 – Recommended data structure for multiple PDVs and zero or more BDCs

The PVinDs in Figure F.6 are:

PVinD 1	Set of BoolT.2.0	(BDC2 and BDC1)
PVinD 2	UInteger.12.4	(PDV1)
PVinD 3	Integer.8.16	(PDV2)
PVinD 4	UInteger.8.24	(Manufacturer/vendor specific)

Annex I
(informative)
Device integration strategies into automation systems

I.1 Overview

I.1.1 Data and information providers and consumers

Ever since SDCI enables digital communication it is possible for the entire automation hierarchy to exchange data and information directly with nearly all kinds of sensors and actuators. While controllers such as PLCs or industrial PCs take over the task of automatically controlling machines and plants, operators are monitoring and maintaining the equipment and processes through human-machine-interfaces. More and more of these machines and plants are integrated in higher level enterprise data processing systems. For commissioning and trouble-shooting engineering tools are temporarily engaged. Figure E.1 shows the principle data flow between all of these systems including Masters and Devices. All of these can be provider and consumer of data and information (see 3.2.1 and 3.2.2).

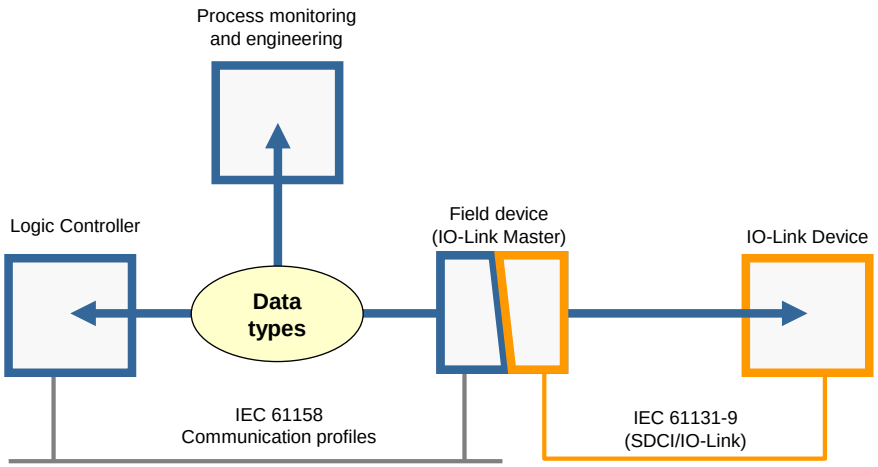


Figure E.1 – Data flow within automation systems

Unfortunately, there is no straightforward direct exchange of data and information between any of these systems since all of them are using their own data types and codings, even the communication systems (see Figure E.2).

Traditionally, the PLC has been a hub between the connected sensors and actuators and the upper level systems and acted as a "representative" for process data. However, with the advent of fieldbuses and now with SDCI direct access to identification and maintenance information is possible and a number of software tools are now eager to acquire data and information such as

- commissioning and diagnosis software,
- asset management,
- audit trailing,
- manufacturing execution systems,
- data server (OPC UA),
- process monitoring (SCADA),
- condition monitoring,
- WEB server.

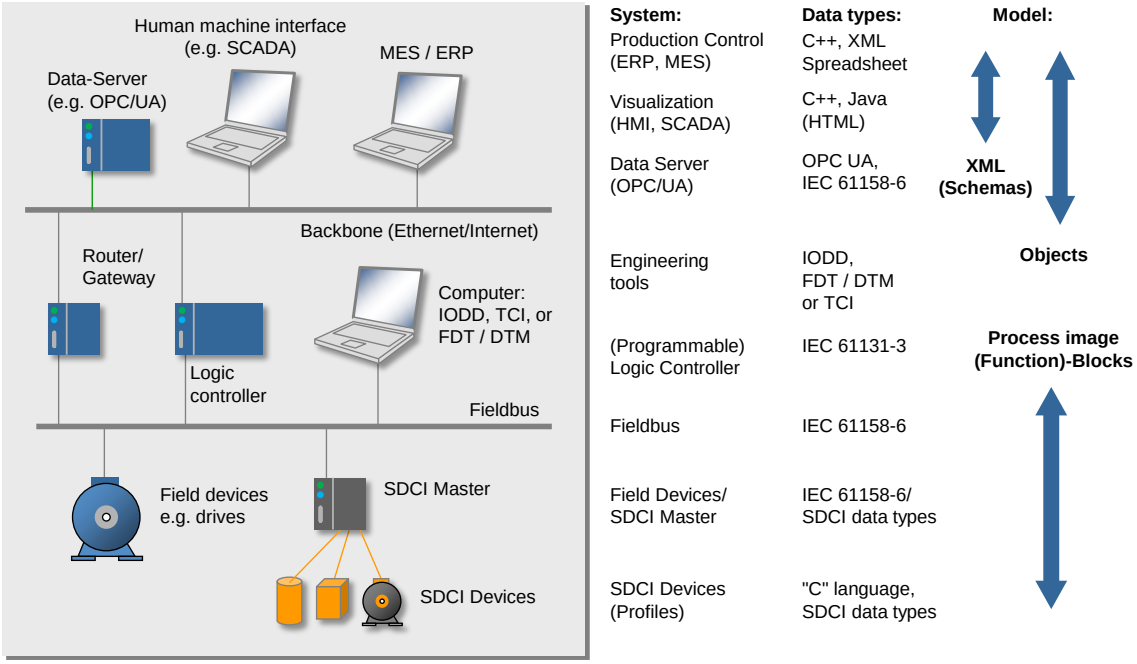


Figure E.2 – Data types within automation systems

I.1.2 Consistency of data and information

In most cases it is possible for a PLC programmer to adjust any SDCI data structures via masking, bit shifting, and/or type casting in PLCs to prepare cyclic Device data ("PD") and acyclic Device data ("OD") for the processing within PLCs. However, this causes increased engineering efforts and risk of erroneous program codes.

Complex data structures are even worse for independent software tools, where no access to IODD information is available.

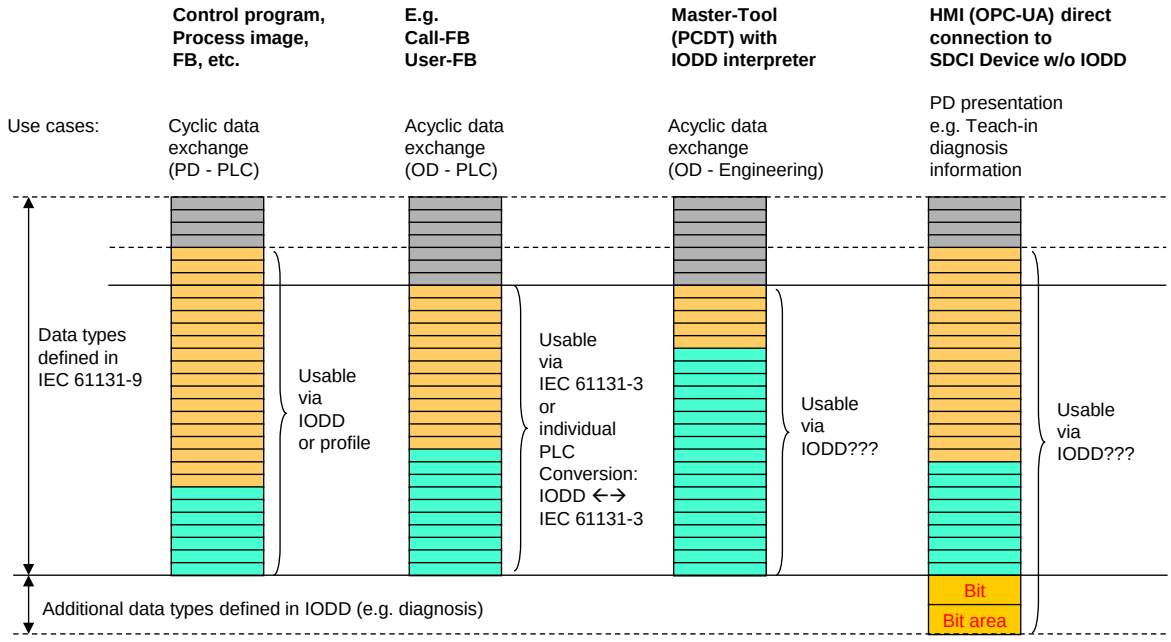


Figure E.3 – Use cases and data type recommendations

Direct access to Devices via SDCI may be impossible or at least associated with intricate adaptation steps. This document provides assistance via consistency tables and rules for the design of easy to handle data types and data structures.

Table E.1 shows the list of the most important PLC data types according to IEC 61131-3. These data types are the initial point for the reference tables. (to be defined).

Table E.1 – PLC data types (IEC 61131-3)

Data type	Definition	Length (bit)	Value range
BOOL	boolean	1	TRUE, FALSE or 1,0
BYTE	octet	8	B#16#00 to B#16#FF
WORD	word	16	W#16#0000 to B#16#FFFF
DWORD	double word	32	DW#16#00000000 to …
LWORD	long word	64	LW#16#0000000000000000 to …
USINT	unsigned short integer	8	0 … 255
UINT	unsigned integer	16	0 .. 65535
UDINT	unsigned double integer	32	-2147483648... 2147483647
ULINT	unsigned long integer	64	-9,2 Trio... 9,2 Trio
SINT	short integer	8	-128... 127
INT	integer	16	-32768... 32767
DINT	double integer	32	-2147483648... 2147483647
LINT	long integer	64	-9,2 Trio... 9,2 Trio
REAL	real	32	± 1,18 x10^{-38} to 3,40 x10^{38}
LREAL	long real	64	± 2,2... x 10^{-308} to 1,79... x 10^{308}
TIME	duration (ms)	32	
LTIME	duration (ns)	64	
DATE	date	16	
TIME_OF_DAY	time	32	
CHAR	character	8	
STRING	character sequence	> 2 x 8	
NOTE Data types marked in grey may not be available in older PLC systems.			

I.1.3 Profile specific data types

Any profile can define specific data types usable for the profile. It is preferred to define structures based on data types from Table E.1 to achieve an easy support of the parameters in the PLC systems.

I.1.4 Memory and transmission octet order

One critical aspect when considering the data type consistency is the memory and transmission octet order as demonstrated in 3.3.6 in [1].

I.2 Integration via process image

I.2.1 General

The target of the process values is the user program coded in the PLC. The data is similar to the data provided by any other fieldbus device. Simple binary data is provided as boolean data. The transport and placement is shown in Figure E.4.

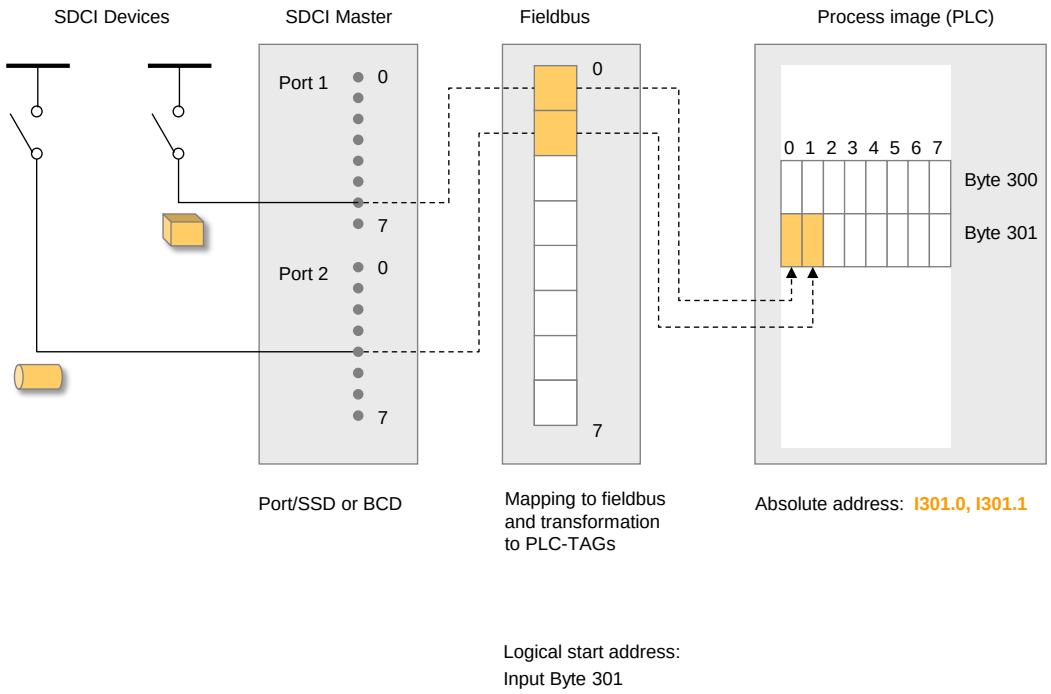


Figure E.4 – Transformation from SDCI to process image of boolean data

Measurement data is provided with data types greater one octet and the corresponding data types according Table E.1. The transport and placement is shown in Figure E.5.

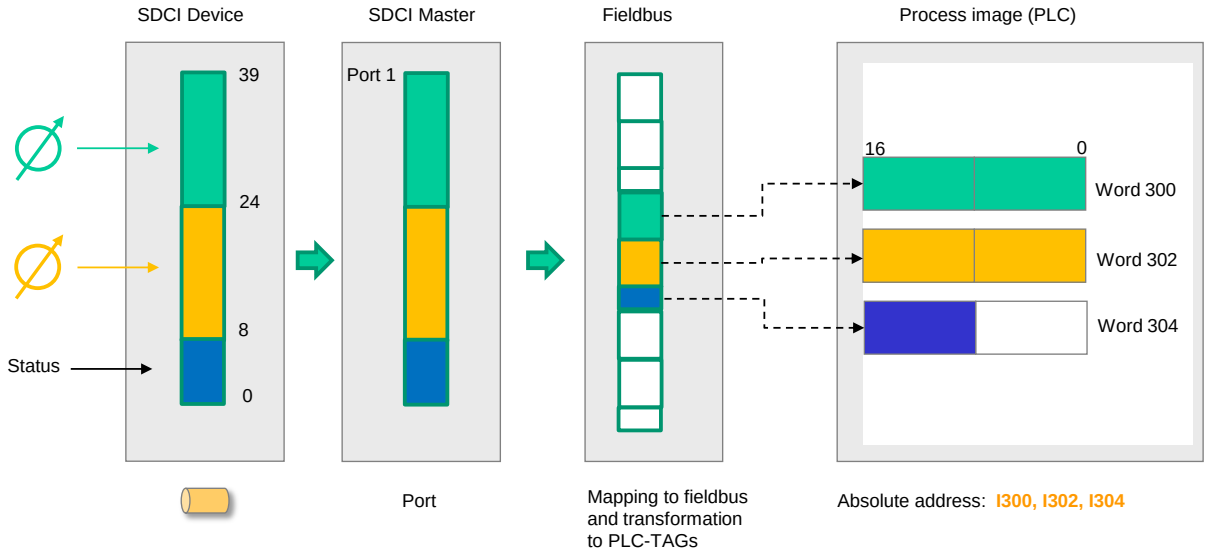


Figure E.5 – Transformation from SDCI to process image of complex data

I.2.2 Import of TAG lists based on the IODD

To reduce the engineering effort for the integration it is possible to generate and provide function blocks for individual Devices of Device families. Any vendor can provide the generator or the ready-to-use function block.

It is also possible to generate lists of tags to be used in the PLC generated on base of the IODD. This list of tags can be imported in the engineering system and allows to use the vendor defined parameter names and settings.

Figure E.6 shows the generation of predefined tags of process data variables.

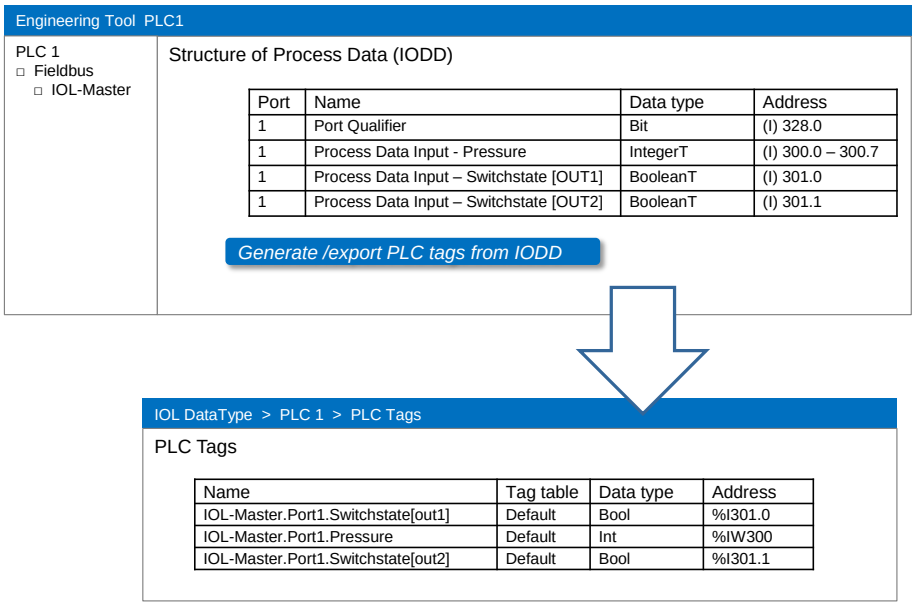


Figure E.6 – Integration via process image

I.2.3 Decomposing packed process data values

As defined in H.1 the process data mapping may contain packed data structures like 14 bit plus 2 bit in one 16 bit variable, like the data structure shown in Figure E.7.

Figure E.7 illustrates the relationship between a dimensioned PDV and its PLC variable for this packed structure.

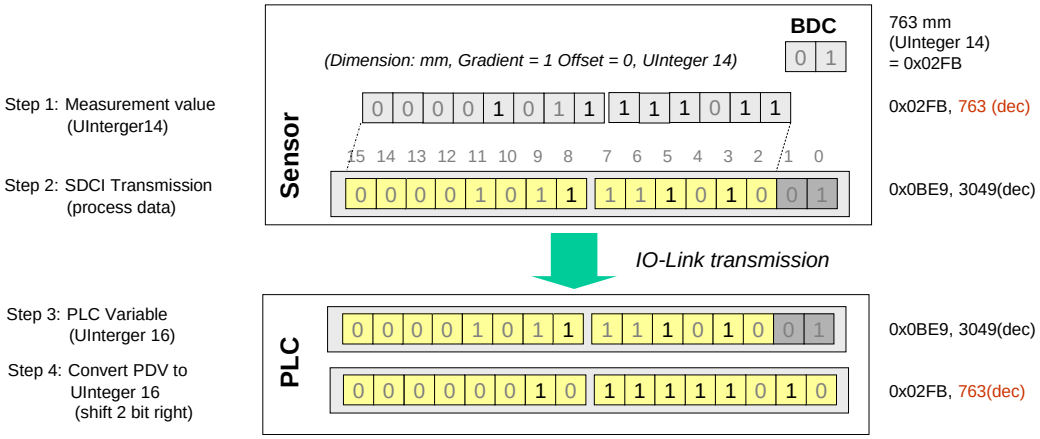


Figure E.7 – Relationship between a dimensioned PDV and its PLC variable

Figure E.8 demonstrates a typical PLC user program for a measurement value conversion. A PLC user program transforms the PDV via shift operations into a 16 bit UInteger variable.

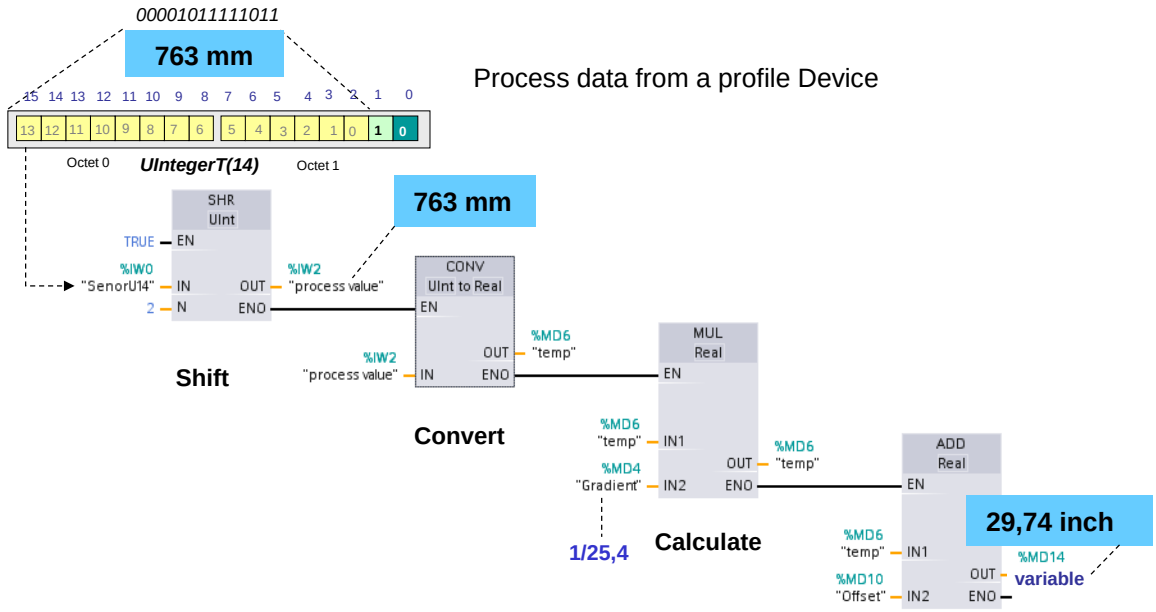


Figure E.8 – Example PLC program for a measurement value conversion

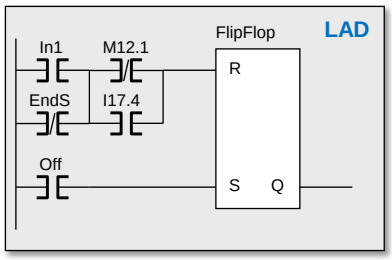
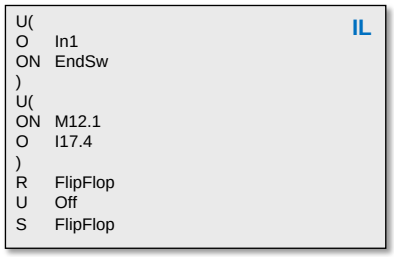
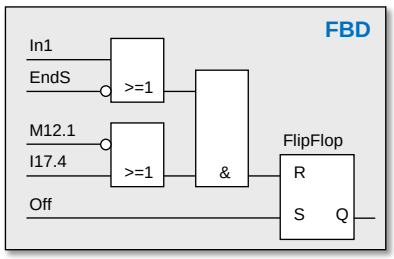
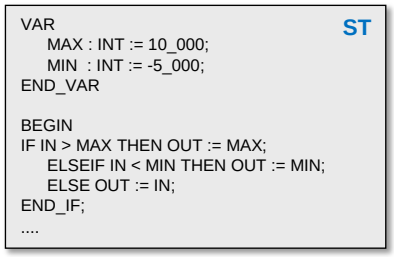
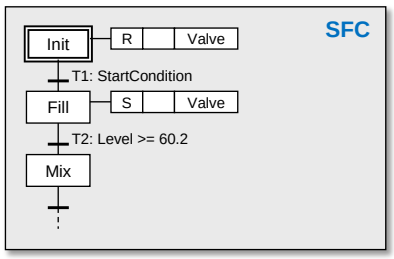
I.3 Integration via PROXY function block (IEC 61131-3)

I.3.1 IEC 61131-3 programming languages

In contrast to the classic Information Technology (IT) with its programming languages Assembler, C, C#, Java, etc. the automation technology developed its own set of programming languages such as IL (Instruction List), LD (Ladder Diagram), FBD (Function Block Diagram), ST (Structured Text, which is close to the PASCAL language), and SFC (Sequential Function Chart) each with its own advantages and historic background (see Table E.2). This development was triggered by the advent of programmable logic controllers (PLC) in the 70's, which started replacing the relay based logic controls. The user was enabled to "draw" his "relay circuits" on the screen (relay ladder logic) that were translated into machine code of microprocessors and processed there in a sequential and cyclic manner (Linear Code). At the beginning of a cycle the information of input "peripherals" (sensors, switches, etc.) were read into a "process image memory" and at the end of the same cycle the result information was written into output "peripherals" (relays, switches, valves, etc.). Up to now, this "polling" principle proved to be a very robust data processing procedure especially with thousands of input and output signals.

These programming languages are standardized in IEC 61131-3 [8]. The following table characterizes the idea of the basic features and graphical layouts of the languages. The process variables are either the physical address (e.g. I17.4) of an I/O-module or a unique symbol (e.g. EndS) of a signal in the field. See [9] and [10] for additional information on IEC 61131-3 programming languages.

1069 **Table E.2 – Characeristic of IEC 61131-3 programming languages**

Language representation	Characteristics
	Ladder Diagram (LD) German: "Kontaktplan" (KOP) Derived from the former Relay Ladder Logic diagrams on "blue prints". Allows immediate fault detection through online status of the signals within the graphical representation.
	Instruction List (IL) German: "Anweisungsliste" (AWL) Derived from Assembler Language. Most flexible for experienced programmers. Portability of programs from one PLC to another is difficult due to machine code dependency.
	Function Block Diagram (FBD) German: "Funktionsplan" (FUP) Derived from Circuit Diagrams of electronic boards. More comprehensible than LD with complex logic. However, root cause fault detection of machines needs more (mental) steps for the user.
	Structured Text (ST) German: "Strukturierter Text" (ST) Derived from the PASCAL programming language. Its purpose is to easily support mathematical type of tasks or complex data manipulation in automation. These program parts can be encapsulated within Function Blocks and hidden from maintenance people. Knowhow protection is possible. This language is best suited for portability of PLC programs.
	Sequential Function Chart (SFC) German: "Ablaufkettensprache" (AS) Goes back to Harel's State Charts for finite state machines in 1967. Programming of states and transitions can be done by the other languages for program logic (IL, LD, and FBD). This language is most efficient in respect to programming effort, debugging, risks at program changes, and root cause fault detection. Highly recommended for machine programs. Acceptance limited due to missing training at schools and universities.

1070

1071 ### I.3.2 Function Blocks and Function Calls according IEC 61131-3

1072 While the first PLCs were providing unstructured so called "Linear Code", the next generation
1073 offered technology to better structure the programs. Thus, common to all of the IEC 61131-3
1074 languages is the powerful element of the "Function Block" (FB). With the help of FBs it could
1075 be managed to satisfy the hardware paradigm of logic components with interfaces and encap-
1076 sulated functionality including memory effects and thus to support reusability (see Figure E.9).
1077 On the other hand, the associated type/instance concept follows the direction of object orienta-
1078 tion within the office automation (IT) in an ideal manner:

A piece of consistent program that shall be reusable can be declared a Type-FB and entered in a program library. The re-entrant code of this Type-FB can have behavior (state machine) and several input and output variables.

Hence, in a real application program this Type-FB can be implemented as an Instance-FB with different persistent data sets (FB Instancies). Local variables within those FBs can be used for (volatile) temporary functions.

The FB Instancies can be called anytime within the Main Program.

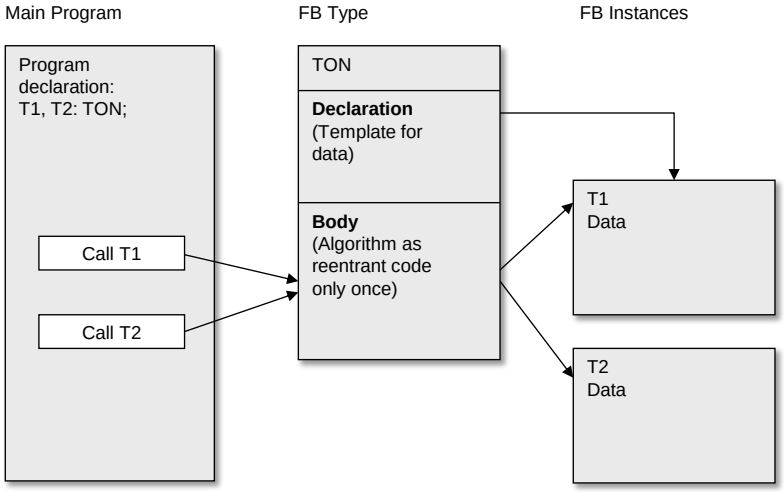


Figure E.9 – Function block and its instance data

Function Calls are following a similar concept of reusable code. However, there is only one return value, no behaviour and no persistent local data. The unit conversion of a variable is an example of a Function Call (FC).

See [9] and [10] for additional information on Function Block and Function Call concepts.

I.3.3 Concept of PROXY Function Blocks

PROXY Function Blocks are representatives of field devices on fieldbuses or lower level SDCI Devices. These function blocks are translating the transmitted data structures (octets) into easy user understandable input and output signals and data at the PROXY-FB interface level (see Figure E.10). Usually, the PROXY-FBs are working with a standard communication function block platform (fieldbus to SDCI) to contact their Devices (see details in Figure E.11).

For complex Devices such as drives, RFID reader/writer, weighing and dosage systems, the PROXY FBs can be become rather complex and the usage of sequential function chart programming (SFC) nearly impossible. In these cases a modular design of the PROXY-FBs using Function Calls is more appropriate. More details for such a design can be retrieved from [11].

Figure E.10 shows the integration of a proxy function block using the generic communication function block which is shown more detailed in Figure E.11

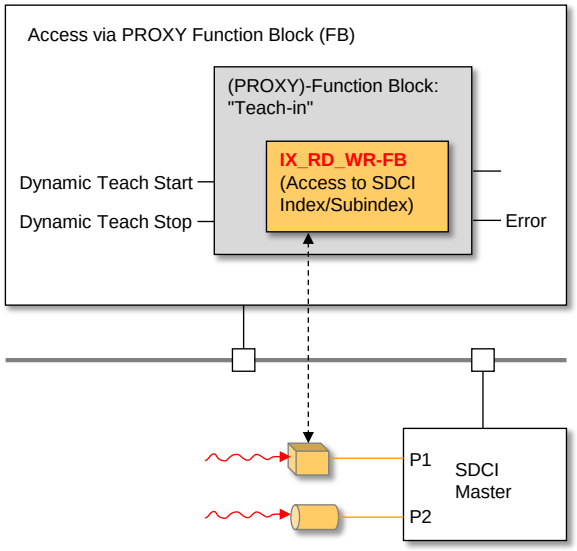


Figure E.10 – PROXY FB using standardized SDCI communication FB

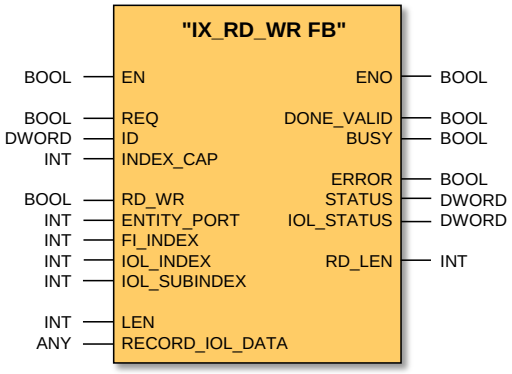


Figure E.11 – Standardized SDCI communication FB "IX_RD_WR"

The responsibility for the standardized communication FB is on behalf of the system manufacturer (PLC and/or engineering tool).

The responsibility for the PROXY-FB is on behalf of a profile group for profile Devices or a manufacturer/vendor for specific Devices.

I.3.4 Device profile activities

It is the responsibility of the Device profiles to specify a proxy function block for each CommonApplicationProfile or DeviceProfile if this supports the engineering of profile Devices.

The specification shall contain

- parameter specifications
- state machines for dynamic behavior
- optionally pseudo code close to ST (Structured Text programming language)

to provide the same functionality in most of the PLC systems.

I.4 Integration into self-configuring systems

It is possible to compose self-configuring systems based on the ProfileIdentifier and the corresponding parameters or behaviour which read out the Device abilities. Then these systems may generate appropriate FunctionBlocks or system components to provide standardized functionality based on the detected Devices.

Precondition for this feature is the availability of Devices conform to distinct profiles.

Bibliography

[1] IO-Link Community, *IO-Link Interface and System*, V1.1.3, June 2019, Order No. 10.002

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

[3] IO-Link Community, *IO Device Description (IODD)*, V1.1.3, December 2020, Order No. 10.012

[4] IEC/TR 62390:2005, *Common automation device profile guideline*

[5] IEC 60050 (all parts), *International Electrotechnical Vocabulary*

[6] IO-Link Community, *IO-Link Test Specification*, V1.1.3, ???? 2020, Order No. 10.032

[7] IO-Link Community, *IO-Link Smart Sensor Profile Ed.2*, V1.1, ???? 2020, Order No. 10.042

[8] IEC 61131-3:2013, *Programmable controllers - Part 3: Programming languages*

[9] Frank Petruzella, *Programmable Logic Controllers*, 4th edition, McGraw-Hill, 2010, ISBN-13: 978-0073510880

[10] Karl-Heinz John, Michael Tiegelkamp, *Programming Industrial Automation Systems – Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids*, 2nd edition, Springer, 2010, ISBN-13: 978-3642120145

[11] PLCopen: <www.plcopen.org>

[12] ISO/IEC 19505-2:2012, Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 2: Superstructure

_____

IO-Link Community
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 (0) 721 / 96 58 590
Fax: +49 (0) 721 / 96 58 589
e-mail: info@io-link.com
http://www.io-link.com/