IO-Link

IO-Link Common Profile

Specification

Version 1.0
July 2017

Order No: 10.072

IO-Link

File name: **IOL_Common-Profile_V10_Jul2017.docx**

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.

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 low-cost 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 low-cost, digital communication technology while maintaining backward compatibility with the current DI/DO signals as defined in IEC 61131-2.

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

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

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.

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 — 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.

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.

This document contains statements on conformity testing for all Device 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 and IEC 61131-2 apply.

3.1.1
ApplicationSpecificTag

read/write data object for the user application to identify the specific Device

[SOURCE: IEC 61131-9, B.2.16]

3.1.2
Device
single passive peer to a Master such as a sensor or actuator

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

[SOURCE: IEC 61131-9, 3.1.14]

3.1.3
DeviceID

unique Device identification allocated by a vendor

[SOURCE: IEC 61131-9, B.1.9]

3.1.4
Event
instance of a change of conditions in a Device

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

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

[SOURCE: IEC 61131-9, 3.1.17]

3.1.5
FirmwareRevision

vendor specific coding of the firmware revision of the Device

[SOURCE: IEC 61131-9, B.2.15]

3.1.6
HardwareRevision

vendor specific coding for the hardware revision of the Device

[SOURCE: IEC 61131-9, B.2.14]

3.1.7
Master
active peer connected through ports to one up to n Devices and which provides an interface to the gateway to the upper level communication systems or PLCs

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

[SOURCE: IEC 61131-9, 3.1.27]

3.1.8
port
communication medium interface of the Master to one Device

[SOURCE: IEC 61131-9, 3.1.31]

3.1.9
Process Data
PD

input or output values from or to a discrete or continuous automation process cyclically transferred with high priority and in a configured schedule automatically after start-up of a Master

[SOURCE: IEC 61131-9, 3.1.33]

3.1.10
ProductID

vendor specific product or type identification

[SOURCE: IEC 61131-9, B.2.11]

3.1.11
ProductName

product name to distinguish between variants

[SOURCE: IEC 61131-9, B.2.10]

3.1.12
ProductText

additional product information such as product category

[SOURCE: IEC 61131-9, B.2.12]

3.1.13
SerialNumber

unique vendor specific code for each individual Device

[SOURCE: IEC 61131-9, B.2.13]

3.1.14
switching signal

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

[SOURCE: IEC 61131-9, 3.1.38]

3.1.15
VendorID

unique vendor identification assigned by the IO-Link community

[SOURCE: IEC 61131-9, B.1.8]

3.1.16
VendorName

vendor name associated to the VendorID

[SOURCE: IEC 61131-9, B.2.8]

3.1.17
VendorText

additional information about the vendor

[SOURCE: IEC 61131-9, B.2.9]

3.2 Profile Guideline : Additional terms and definitions

3.2.1
BinaryDataChannel
BDC

binary information as switching or control signal

3.2.2
data (digital)
quantities, characters, or symbols on which operations are performed by a processor within an automation device, transmitted in the form of electrical signals, stored and recorded on magnetic, optical, or mechanical recording media, and displayed on human machine interfaces

EXAMPLE A number '32' and the characters '°C' represent digital data, whereas the combination '32 °C' represents information

Note 1 to entry: Digital data, information, and knowledge describe an ascending seamless concept correlated to syntax, semantics, and reasoning

3.2.3
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]

3.2.4
FunctionClass
particular function within a Device profile

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

3.2.5
information

any propagation of cause and effect within an automation system conveyed either as the content of a message or through direct or indirect observation

EXAMPLE Correlated data presented on a display

Note 1 to entry: Digital data, information, and knowledge describe an ascending seamless concept correlated to syntax, semantics, and reasoning

3.2.6
knowledge

understanding of real things such as causes and effects (processes) as well as abstract concepts

EXAMPLE Longer-term interpretation of information

Note 1 to entry: Knowledge can be acquired by a cognitive observer of information

Note 2 to entry: Digital data, information, and knowledge describe an ascending seamless concept correlated to syntax, semantics, and reasoning

3.2.7
ProcessDataVariable

PDV

Representation of process values

3.2.8
ProfileIdentifier

unique identifier for Device Profile, CommonApplicationProfile, or FunctionClass

3.2.9
Process Variable Input Descriptor

PVinD

descriptor for position and offset of variables within the Process (input) Data entity

3.2.10
Process Variable Output Descriptor

PVoutD

descriptor for position and offset of variables within the Process (output) Data entity

3.3 Symbols and abbreviated terms

DI	Digital input	
DO	Digital output	
I/O	Input / output	
OD	On-request Data	
PD	Process Data	
PLC	Programmable logic controller	
SDCI	Single-drop digital communication interface	
SIO	Standard Input Output (binary switching signal)	[IEC 61131-2]

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.

3.4.2 Memory and transmission octet order

Figure 1 demonstrates the order that shall be used when transferring WORD based data types from memory to transmission and vice versa (see also clause 7.3.3.2 in [1]).

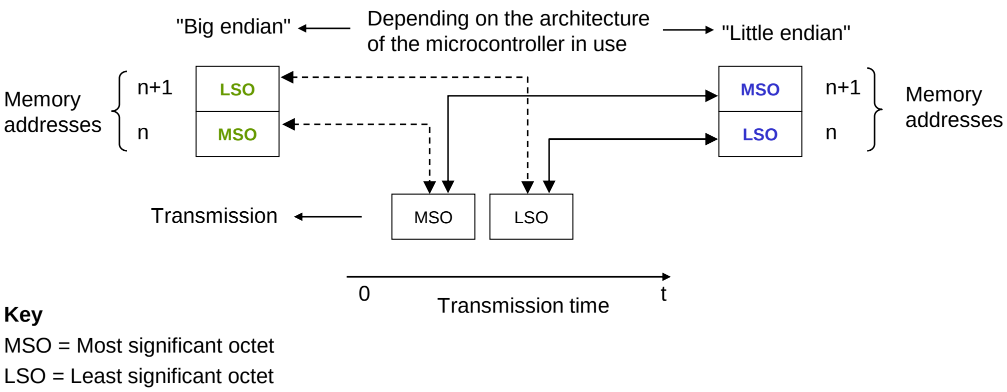


Figure 1 – Memory and transmission octet order

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.

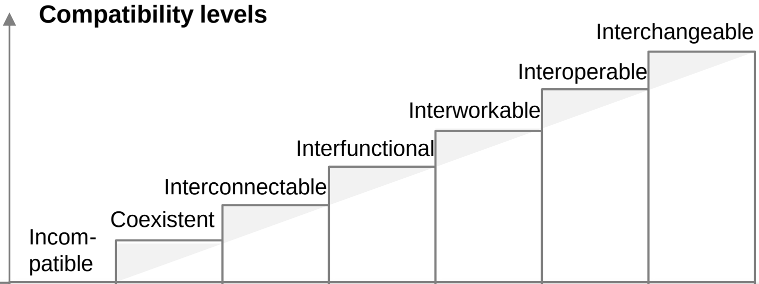
It is the purpose of SDCI to overcome the limitations of the classic Device interfaces DI, DO, AI, and AO via a point-to-point digital communication that allows transmitting digitally not only binary and/or analog information but additional information also. Very often, the changes to the core sensor 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 2 shows compatibility levels based on IEC 62390.



Device features	Incom-patible	Coexistent	Interconnectable	Interfunctional	Interworkable	Interoperable	Interchangeable	
Mechanics							X	Device profiles
Dynamic behavior						(X)	(X)	
Application functionality						X	X	
Parameter semantics					X	X	X	
Data structures				X	X	X	X	
Data types				X	X	X	X	SDCI communi-cation
Data access			X	X	X	X	X	
Communication interface			X	X	X	X	X	
Communication protocol		X	X	X	X	X	X	

Figure 2 – 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].

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

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 acayclic or process data “access direction”: specification whether the parameter can be read and/or written “data type”: IEC 61131-3 data types are preferred	

5 Device profiles related to IEC 61131-9

5.1 SDCI technology specified in IEC 61131-9

Figure 3 shows the domain of the SDCI technology within the automation hierarchy.

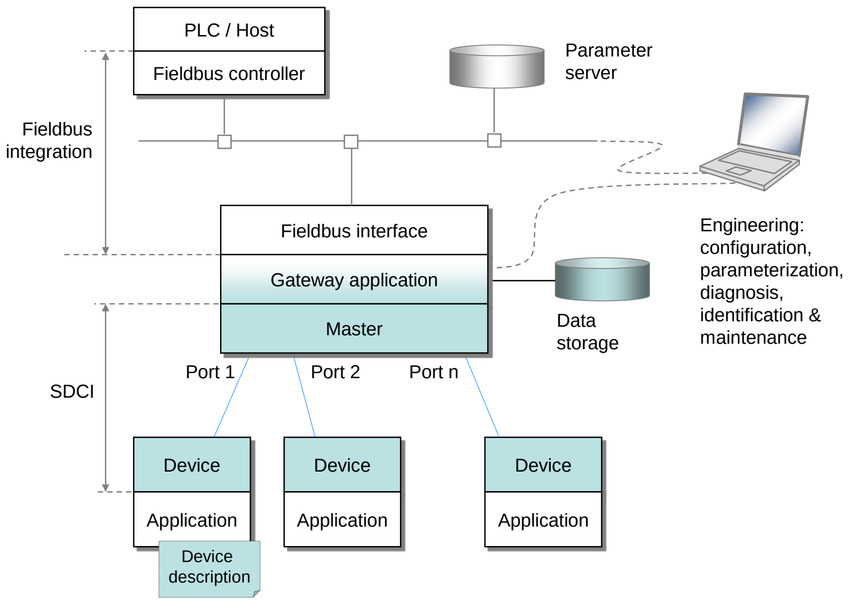


Figure 3 – Domain of the SDCI technology within the automation hierarchy

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 4 shows an overview of the SDCI technologies and profiles.

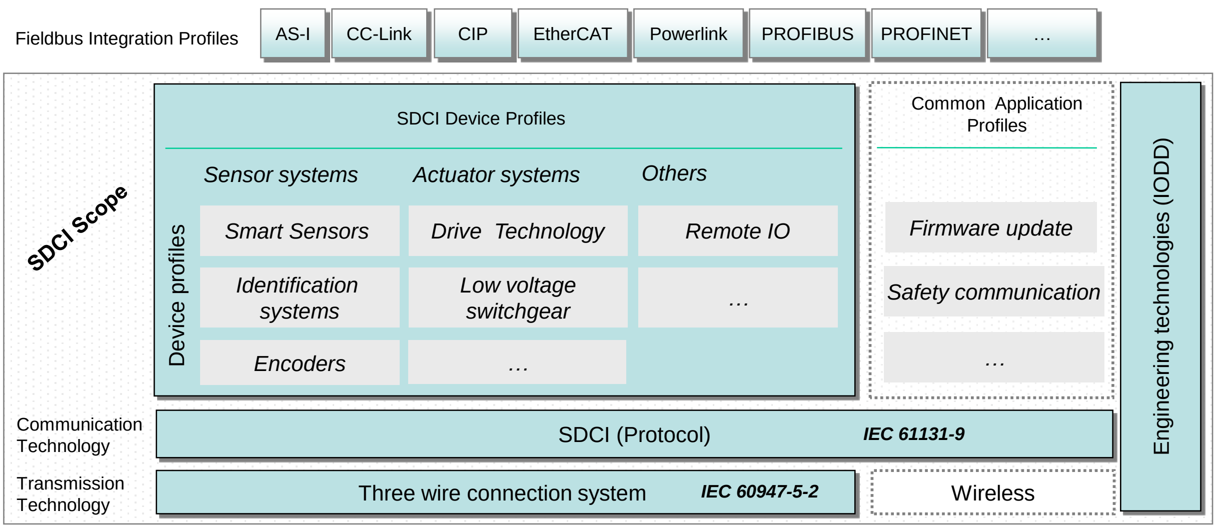


Figure 4 – Overview of SDCI technologies and profiles

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

- smart sensors,
- identification systems,
- drives,
- low voltage switch gears,
- encoders,
- 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 recognizes a "generic" Device to a certain extent even when he switches from one brand to another.

The "Common Application Profiles" represent specifications that several Device type families/ classes can use. Examples are firmware update, functional safety communication or energy management.

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 H. However, this organization is interested in harmonizing the "views" of SDCI users through the different fieldbuses.

5.3 Concept of profiles

There are two approaches for profiling SDCI Devices. The distinct profile is strictly defined with detailed process data layout and functionality. The generic profile uses different FunctionClasses to form a manufacturer specific Device behavior.

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 remanent parameters
- Support of block parameter handling

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.

It can be compared with a tool box with dedicated and identical tools, the minimum set of tools is defined, there may be more tools inside.

5.3.3 Generic profiles

The generic profiles consist of a manufacturer specific combination of different FunctionClasses which may have variable definitions like process data layouts. The user can only rely on the common subset of the referenced FunctionClasses, the interchangeability is reduced.

It can be compared with a tool box, the tools are nearly the same, but in every tool box is another set of tools.

5.4 Profile characteristics

The profiles are based on the definition of FunctionClasses. These FunctionClasses can be used as a standalone property or combined to ProfileIdentifier 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 additional FunctionClasses (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.

Some FunctionClasses are highly recommended or mandatory for DeviceProfiles or Common-ApplicationProfiles such as Device Identification or Device Diagnosis, see 6 or the specification of the distinct profiles.

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

The parameter object "Profile Characteristic" supports up to 32 ID entries. Each and every supported profile and FunctionClass shall be indicated and coded as specified in Table 3. To avoid multiple occurences of a Prrofileidentifier 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)	UInte-gerT16	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 CommonApplication-ProfileIDs shall be indicated by 1 to n FunctionClassIDs
4) The IDs shall be listed in ascending order: DeviceProfileIDs → CommonApplicationProfileIDs → FunctionClassIDs

Figure 5 – Indication rules for ProfileIdentifiers

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

Table 4 shows the example content of the "Profile Characteristic" 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	0x0004: SSP 2.2
		0x4000: I&D

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

Table 5 – Example of the profile identification of a Generic Sensor

Index	ProfileIdentifier type	Referenced ProfileID
0x000D	DeviceProfileID	0x0001: Smart Sensor Profile
	FunctionClassID	0x8001: Binary Data Channel
		0x8002: Process Data Variable
		0x8004: Teach-in

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

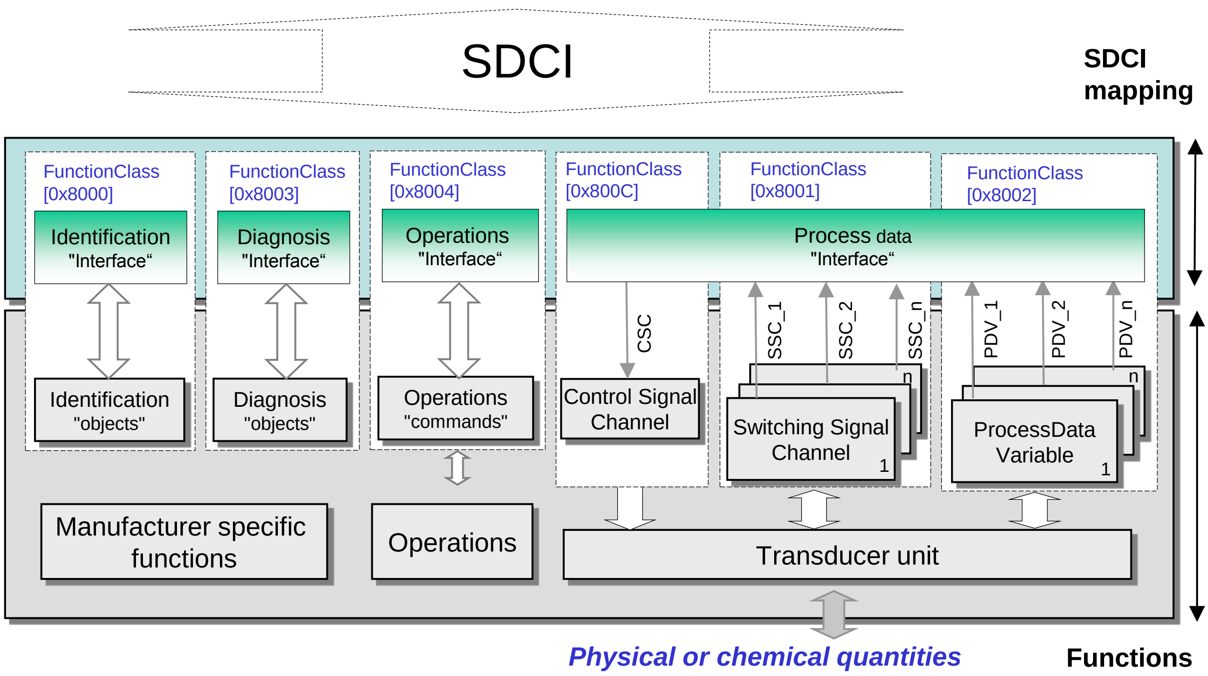


Figure 6 – Overview of typical FunctionClasses

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 6.

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 an actuator, 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 7 shows a possible object model of a combined sensor/actuator Device profile.

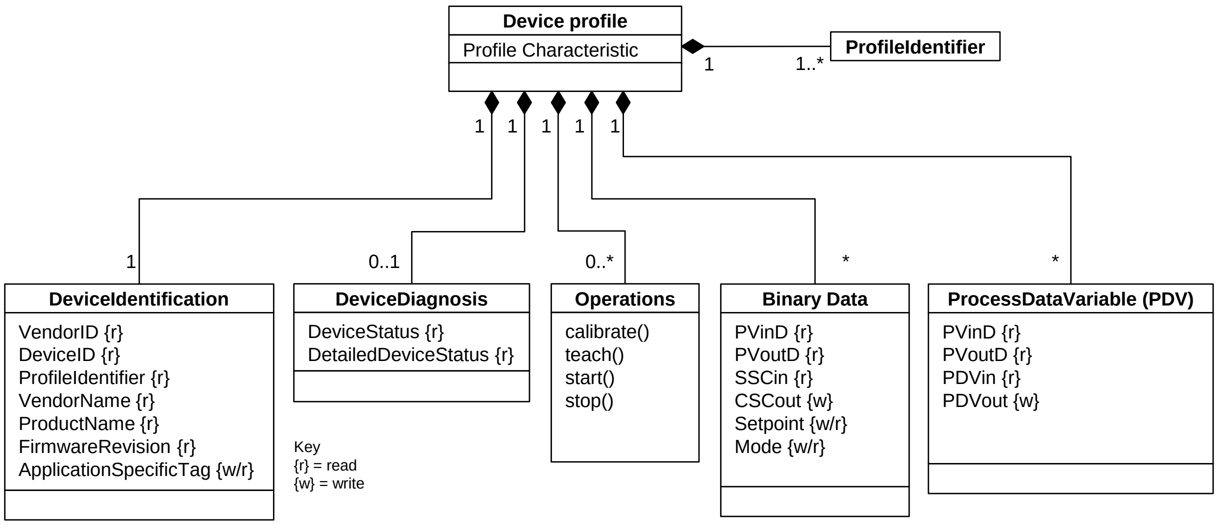


Figure 7 – Typical object model for Device profiles

The parameter "Profile Characteristic" 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 Masters 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.

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

6 Rules and constraints for developing Device profiles

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

6.1 How to create a Device profile

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

- Each profile defines the mandatory FunctionClasses within the specific ProfileIdentifier
- The FunctionClass Identification (0x8000] shall be mandatory for every ProfileIdentifier
- The FunctionClasses Diagnosis [0x8003] is highly recommended for every ProfileIdentifier
- There shall not be two or more FunctionClasses with an identical function (no duplicates)

- There will be no versioning of profiles (identifiers). Changes to a profile result in a new ProfileIdentifier
- The profiles shall specify as mandatory all parameters and contents required to guarantee the compatibility of profile Devices supporting a ProfileIdentifier
- The defined ProfileIdentifier and parameters shall be published in the SDCI Profile overview as defined in 5.4
- Restrictions on combinations of ProfileIdentifier shall be described in the DeviceProfile or FunctionClass definition (responsibility of the profile working group)
- FunctionClass specific parameters shall be defined with all attributes such as
  – Parameter name as defined in the IODD
  – Index
  – Subindex access availability
  – Subindex structure
  – Subindex offset
  – Parameter type or length
  – Allowed parameter content
  – Default value
  – Access rights
  – Mandatory, optional, or conditional availibility
- FunctionClass specific process data structures shall be defined with all attributes such as
  – Subindex structure
  – Subindex offset
  – Allowed process data content
  – Validity rules
- Profile parameters shall use the index range reserved for the profile considering possibly combinable profiles
- Profiles shall define their common EventCodes within range reserved for the profile considering possibly combinable Profiles

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 Process Data structure rules

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

- BDCs are right-aligned in ascending order, always at bit offset 0 (Figure A.4 or Figure A.5) PVinD in this case is: Set of BoolT.2.0
- PDV with e.g. UIntegerT12 is left-aligned mapped to bit offset 4 (Figure A.4 or Figure A.5) PVinD in this case is: UIntegerT.12.4
- Auxiliary variables (e.g. qualifier information) shall be right-aligned to the BDCs PVinD in this case is: UIntegerT.2.2 (Figure A.5)
- 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 Annex E.2.3 and E.2.4 in [1] for casting rules.

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

- 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

6.4 Profile EventCodes

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.

7 Identification and Diagnosis (I&D)

7.1 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

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	DeviceIdentification	See A.2
		0x8003	DeviceDiagnosis	See A.4
		0x8002	Pro-cessDataMapping	See A.3
		0x8100	ExtendedIdentifica-tion	See A.5

7.2 Proxy Function Block for Identification and Diagnosis

To ease the integration in Run-Time systems like PLCs, an appropriate FunctionCall is specified in C.1. The FunctionBlock 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 his 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]	DeviceIdentification	A.2
[0x8002]	ProcessDataVariable (PDV)	A.3
[0x8003]	DeviceDiagnosis	A.4
[0x8100]	ExtendedIdentification	A.5

A.2 Device identification objects [0x8000]

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

- Product ID
- Firmware Revision
- Application Specific Tag

The Product ID and the Firmware revision are unchanged to the definition in clause B.2.11 and B.2.15 of [1]. The parameter Application Specific Tag 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.

This standardized FunctionClass [0x8000] is mandatory for all Device profiles.

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

A.3.1 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.

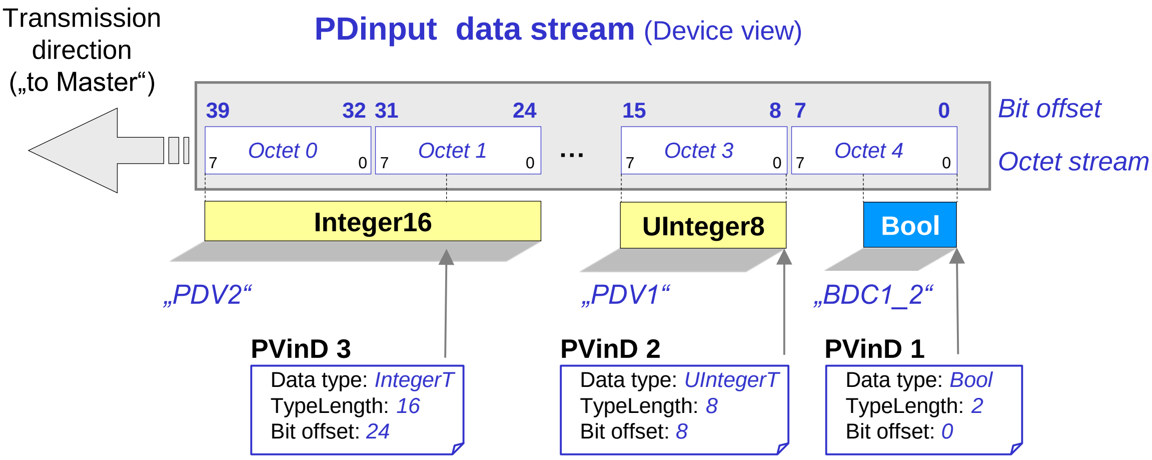


Figure A.1 – Example PDinput data stream

The "PDinput data stream" example shown in Figure A.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.

A.3.2 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.

A.3.3 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 as shown in Figure F.2 and Figure E.

A.3.4 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 A.2. The number of supported BDCs, four in Figure A.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

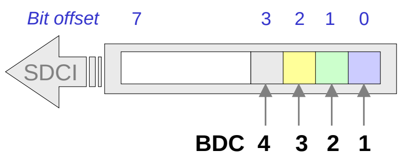


Figure A.2 – Recommended data structure for pure BDCs

A.3.5 One PDV

It is highly recommended for profile Devices with one PDV to use the data structure demonstrated in Figure A.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

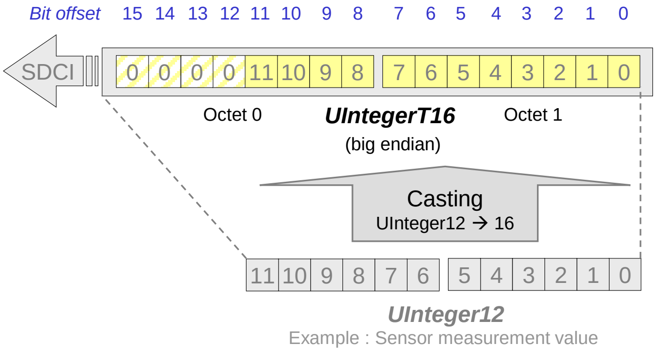


Figure A.3 – Recommended data structure for one PDV

A.3.6 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 A.3.9 apply.

A.3.7 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 A.4.

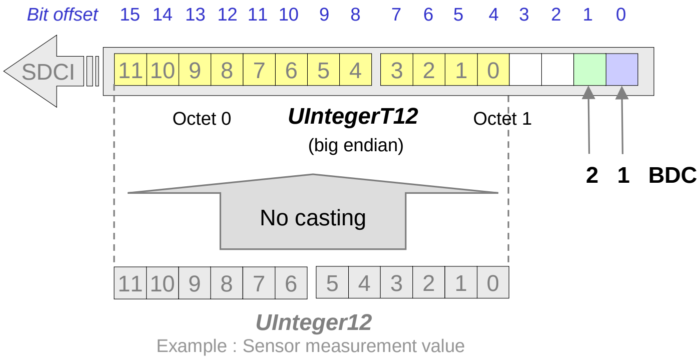


Figure A.4 – Recommended data structure for a PDV and up to two BDCs

The rules in 6.3 shall be observed.

A.3.8 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 A.5.

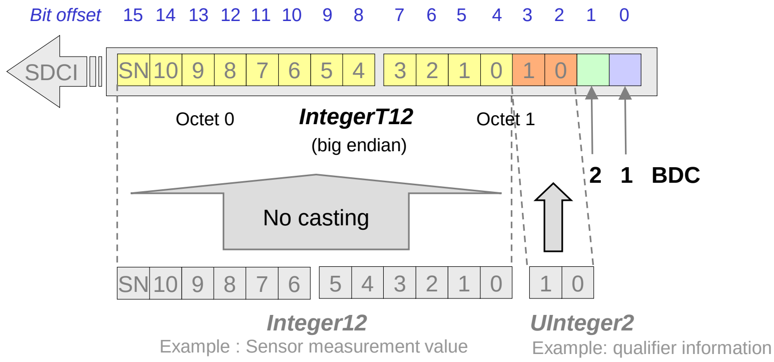


Figure A.5 – Recommended data structure for a PDV, BDCs, and auxiliary variables

The rules in 6.3 shall be observed.

A.3.9 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 A.6. The following rules shall be observed (mandatory):

- Within the first two octets the rules of A.3.6 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

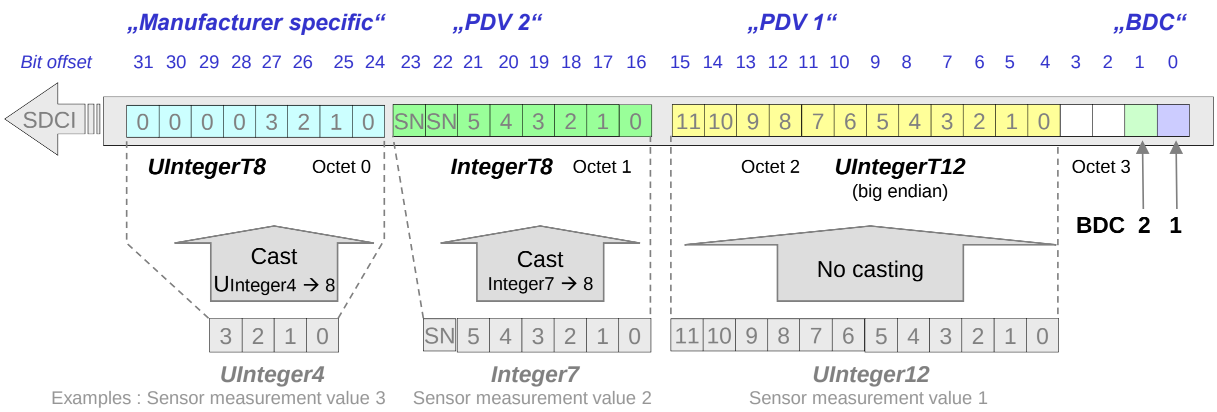


Figure A.6 – Recommended data structure for multiple PDVs and zero or more BDCs

The PVinDs in Figure A.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)

A.3.10 Process data description

Each and every profile Device provides an input Process Data description (PDInputDescriptor) indicating the composition (mapping) of the BinaryDataChannels (BDC) and ProcessDataVariables (PDV) in the "PDinput data stream" with the necessary number of octets and/or an adequate output Process Data description (PD Output Descriptor). The coding of the corresponding parameters is defined in B.5.

The content of the process variable descriptors PVinD or PVoutD shall be available

- in the user manual of the profile Device,
- within the profile Device in the corresponding Index.

Each and every PDV or BDC respectively is described unambiguously via its descriptor 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 here combined BinaryDataChannels (BDCs)
- the length of the data type (TypeLength) in bit, for example 6 for UInteger6
- the bit offset (Bit offset) as the beginning of the variable in the data stream
- any manufacturer/vendor specific data structures, which cannot be described via the standard BDC or PDV descriptors, are described via a process variable descriptor (e.g. additional output data)

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

- Device Status
- Detailed Device Status

Both parameters are unchanged to the definition in clause B.2.18 and B.2.19 in [1].

This standardized FunctionClass [0x8003] is highly recommended for all Device profiles.

A.5 Extended Identification [0x8100]

The FunctionClass 0x8100 defines extended identification which can be used e.g. for localisation 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

- Function Tag
- Location Tag

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

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 soft-ware 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 SystemCommand "Restore factory settings" (0x82) will 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/O/C	Remark
…						
0x0002 (2)	SystemCommand	W	1 octet	UIntegerT	M	See B.2
0x000D (13)	Profile Characteristic	R	variable	ArrayT of UIntegerT16	M	See B.4 See clause B.2.5 in [1]
0x000E (14)	PDInput Descriptor	R	variable	ArrayT of OctetStringT3	M	See B.5 See clause B.2.6 in [1]
0x000F (15)	PDOutput Descriptor	R	variable	ArrayT of OctetStringT3	M	See B.5 See clause B.2.7 in [1]
…						
0x0010 (16)	Vendor Name	R	max. 64 octets	StringT	M	See clause B.2.8 in [1]
…						
0x0012 (18)	Product Name	R	max. 64 octets	StringT	M	See clause B.2.10 in [1]
0x0013 (19)	Product ID	R	max. 64 octets	StringT	M	See clause B.2.11 in [1]
…						
0x0015 (21)	Serial Number	R	max. 16 octets	StringT	M	See clause B.2.12 in [1]
0x0016 (22)	Hardware Revi-sion	R	max. 64 octets	StringT	M	See clause B.2.14 in [1]
0x0017 (23)	Firmware Revision	R	max. 64 octets	StringT	M	See clause B.2.15 in [1]
0x0018 (24)	Application Specific Tag	R/W	32	StringT	M	See B.2 See clause B.2.16 in [1]
0x0019 (25)	Function Tag	R/W	32	StringT	M	See B.6
0x001A (26)	Location Tag	R/W	32	StringT	M	See B.6
…						
0x0024 (36)	Device Status	R	1 octet	UIntegerT	M	See B.7 See clause B.2.18 in [1], default value is "0".

Index (dec)	Object name	Access	Length	Data type	M/O/C	Remark
0x0025 (37)	Detailed Device Status	R	variable	ArrayT of OctetStringT3	M	See B.7 See clause B.2.19 in [1], default values are "0". Contains a minimum of one Event entry.
Keys M = mandatory; O = optional; C = conditional						

B.2 Mandatory SystemCommands

This clause describes the SystemCommands which are already defined in [1]. The Common Profile Specification defines the commands defined in Table B.2 as mandatory for all profile Devices.

Table B.2 – Parameter “SystemCommand”

Command (hex)	Command (dec)	Command name	M/O	Definition
0x01	1	ParamUploadStart	M	See clause B.2.2 in [1]
0x02	2	ParamUploadEnd	M	
0x03	3	ParamDownloadStart	M	
0x04	4	ParamDownloadEnd	M	
0x05	5	ParamDownloadStore	M	
0x06	6	ParamBreak	M	
0x82	130	Restore factory settings	M	
Key M = mandatory				

B.3 Identification parameters

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 Application Specific Tag is defined with the maximum size of 32 octets as defined in Table B.3. The parameter shall be saved remanent 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	Application Specific Tag	32	StringT
Keys R = read W = write n/a = not applicable						

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 (14)	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 rules regarding the ProfileIdentifier. There is no Subindex support required.

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

Bit	Item	Coding
Octet 1	DataType	0: OctetStringT 1: Set of BoolT 2: UIntegerT 3: IntegerT 4: Float32T 5 to 255: reserved
Octet 2	TypeLength	0 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

B.5.2 PD Input Descriptor

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

Table B.6 defines the structure of the PD Input Descriptor regarding the offset and Subindex layout. Subindex support is not required.

Table B.6 – Structure of "PD Input Descriptor"

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x000E (15)	1	(n-1) * 3	R	PVinD 1	24 bit	OctetStringT3
	…					
	n	0	R	PVinD n	24 bit	OctetStringT3
Key	n = number of provided descriptors					

B.5.3 PD Output Descriptor

Profile Devices with process data output shall use the standard Device parameter "PD Output Descriptor" in Index 0x000F to provide the description information according to Table B.5.

Table B.7 defines the structure of the PD Output Descriptor regarding the offset and Subindex layout. Subindex support is not required.

Table B.7 – Structure of "PD Output Descriptor"

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x000F (16)	1	(n-1) * 3	R	PVoutD 1	24 bit	OctetStringT3
	…					
	n	0	R	PVoutD n	24 bit	OctetStringT3
Key	n = number of provided descriptors					

B.6 Extended Identification parameters

This clause defines the extended identification parameters which can be used for overall localisation and identification of any Device.

The content is not predefined, the customer can write any visible string conform to his own naming rules. The R/W parameters “Function Tag” and “Location Tag” shall be saved remanent and handled by the DataStorage mechanism. As default it is recommended to fill the parameter “Function Tag” and “Location Tag” with “***”.

Table B.8 defines the structure of the parameters.

Table B.8 – Parameter Extended Identification

Index (dec)	Subindex (dec)	Offset	Access	Parameter Name	Length	Data type
0x0015 (21)	n/a	n/a	R	Serial Number	Max 16 octets	StringT
0x0016 (22)	n/a	n/a	R	Hardware Revision	Max 64 octets	StringT
0x0019 (25)	n/a	n/a	R/W	Function Tag	32 octets	StringT32
0x001A (26)	n/a	n/a	R/W	Location Tag	32 octets	StringT32
Key	n/a = not applicable					

Opposite to the definition in clause E.2.6 in [1], the content of the parameters Function Tag and Location Tag shall never be transmitted in the unmodified version with padding octets, it shall always be transmitted in the condensed form. This restriction allows to use this parameters even in systems with shorter string length than defined here.

Figure B.1 shows an example of this restriction.

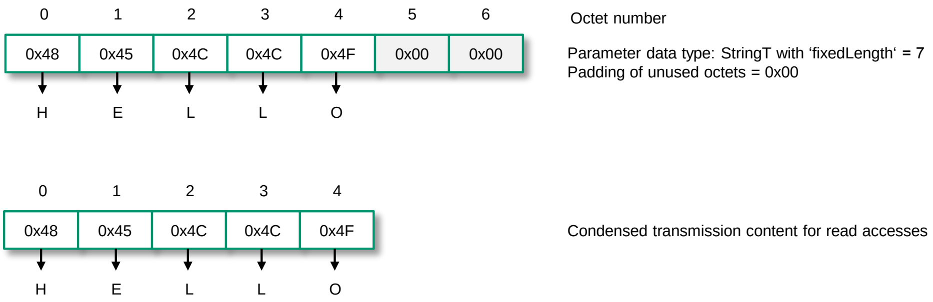


Figure B.1 – Condensed transmission of StringT variables

B.7 Diagnosis parameters

The structure and coding is defined in clauses B.2.18 and B.2.19 in [1].

Table B.9 defines the structure of the Detailed Device Status regarding the offset and Subindex layout. Subindex support is not required.

Table B.9 – Structure of "Detailed Device Status"

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					

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).

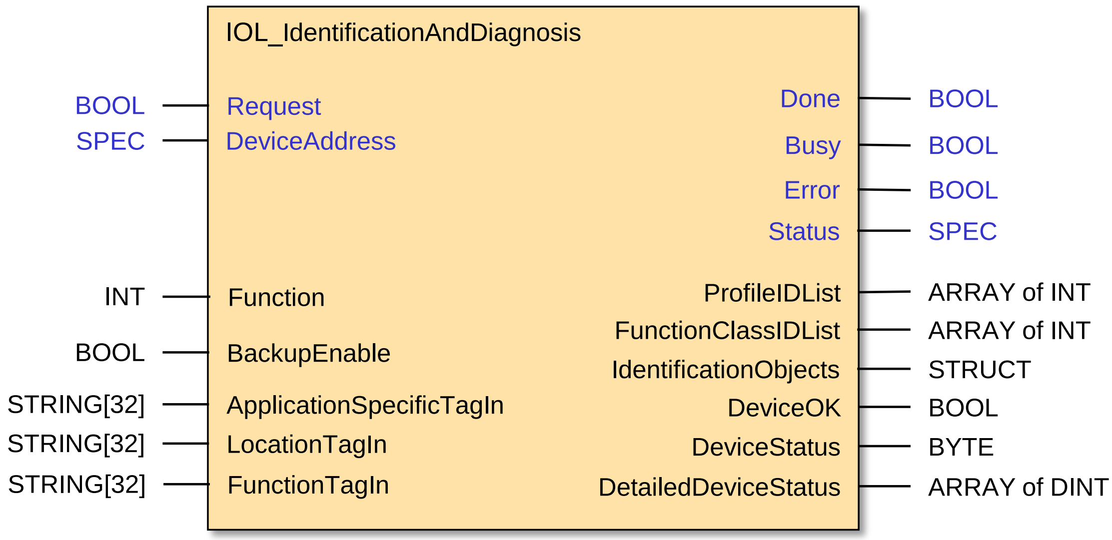


Figure C.1 – Proxy FB for Device Identification and Diagnosis

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. "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.2.18 in [1]
DetailedDeviceStatus	ARRAY of DWORD	This parameter contains the type casted values from the Device parameter defined in clause B.2.19 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		

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

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

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

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

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.

D.2 Constraints and rules

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

- Every object name shall start with an appropriate abbreviation of the FunctionClass
- The object name shall be human-readable and can be abbreviated to shorten the name
- Commands shall be named in imperative
- A menu group shall represent the FunctionClass without abbreviation
- 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 " – "
- The menu entries shall be located in the specified menu section
- The menu entries shall not be altered in layout and structure

D.3 Name definitions

D.3.1 Profile type characteristic names

The profile characteristic name defined in 7.1 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.

D.4.2 Menu structure of the ProfileIdentifier

In Figure D.1 the menu structure of a sample profiled Device is specified, it shall be located in the identification section of the menu.

- Profile Support	
- Profile Characteristic	
DeviceProfile 1	First DeviceProfile
…	
DeviceProfile d	Last DeviceProfile
ApplicationProfile 1	First ApplicationProfile
…	
ApplicationProfile a	Last ApplicationProfile
FunctionClass 1	First additional FunctionClass
…	
FunctionClass f	Last additional FunctionClass

Note d = amount of supported device profiles

a = amount of supported application profiles

f = amount of supported function classes

Figure D.1 – Data flow within automation systems

The ProfileIDs shall be sorted in ascending order, the different types shall be distinguished by their associated Profile type name like DeviceProfile, ApplicationProfile and FunctionClass.

Annex E
(informative)
Device integration strategies into automation systems

E.1 Overview

E.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, 3.2.3, and 3.2.6).

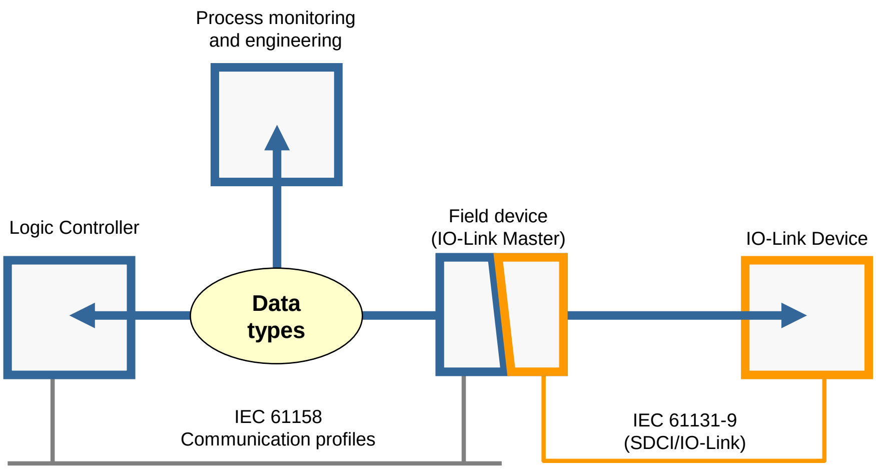


Figure E.1 – Data flow within automation systems

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.

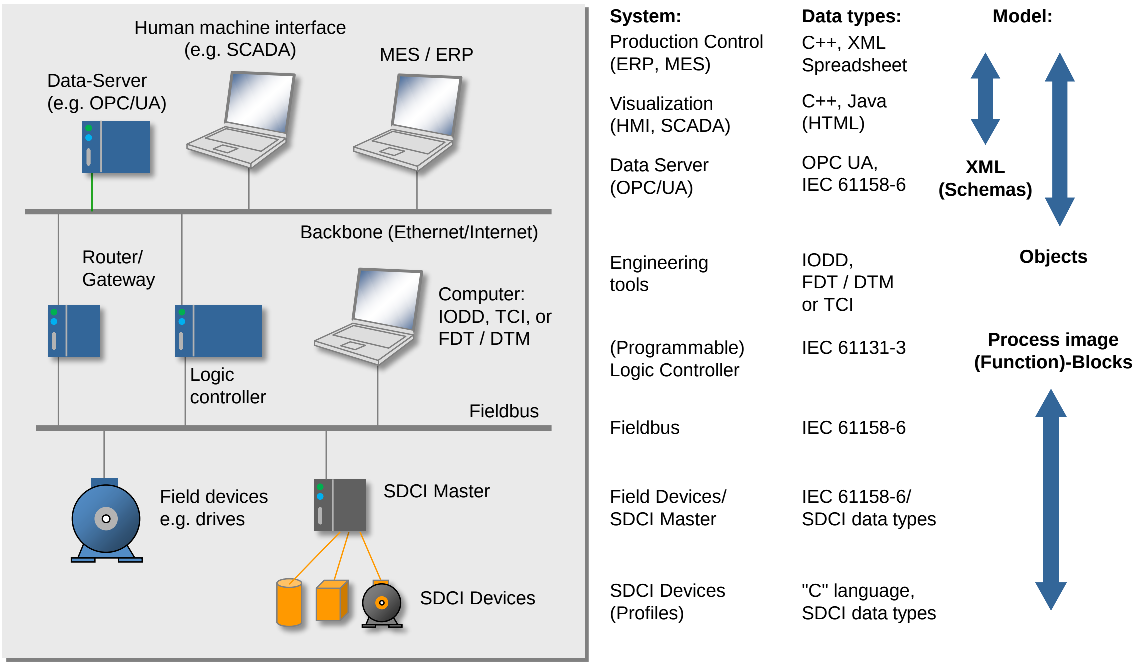


Figure E.2 – Data types within automation systems

E.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.

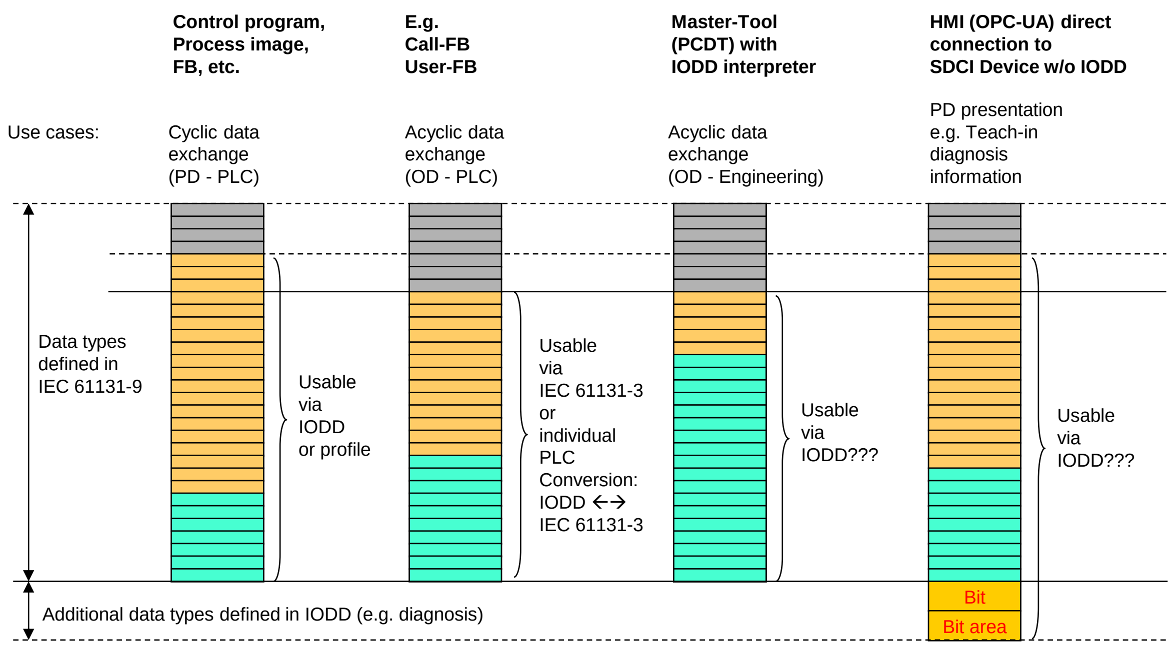


Figure E.3 – Use cases and data type recommendations

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.			

E.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.

E.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 Figure 1.

E.2 Integration via process image

E.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.

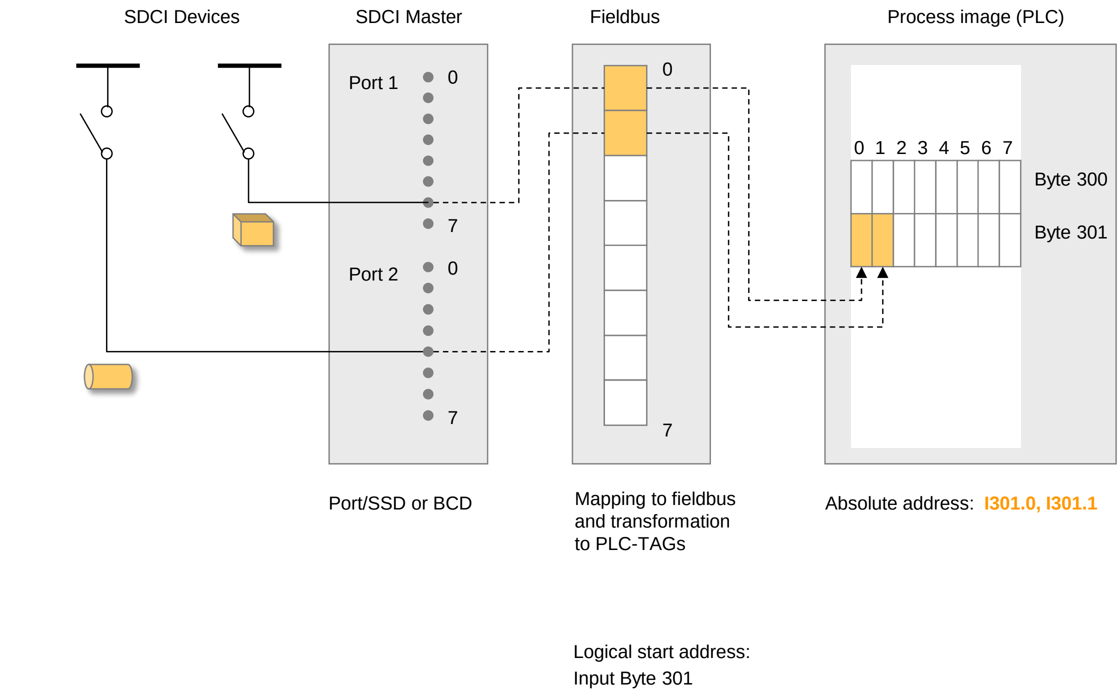


Figure E.4 – Transformation from SDCI to process image of boolean data

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.

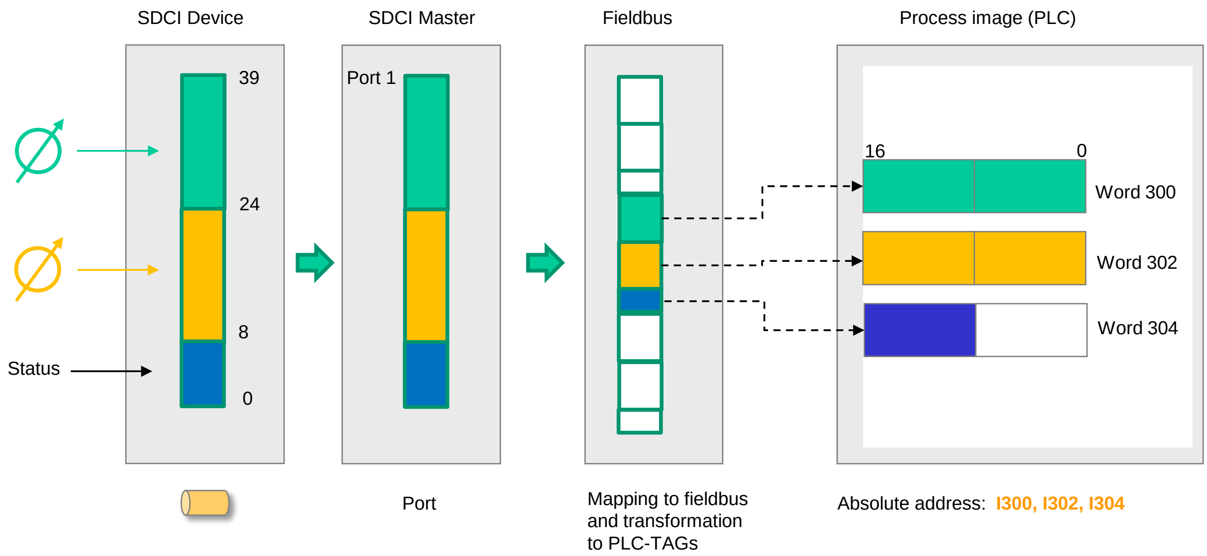


Figure E.5 – Transformation from SDCI to process image of complex data

E.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.

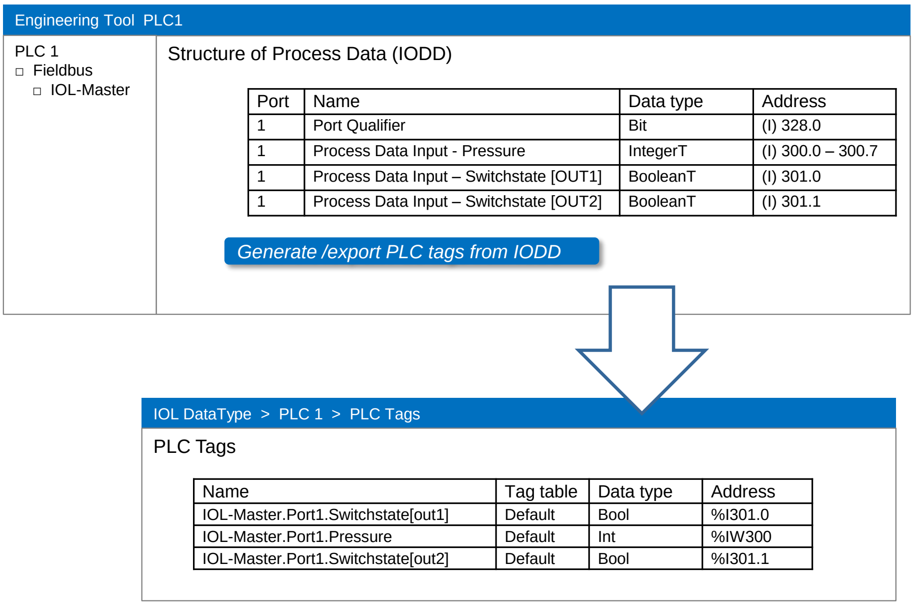


Figure E.6 – Integration via process image

E.2.3 Decomposing packed process data values

As defined in A.3.1 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.

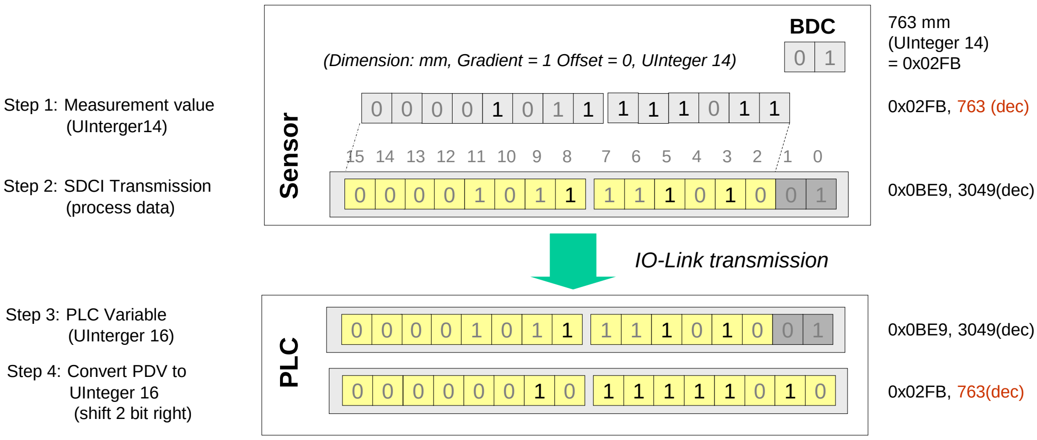


Figure E.7 – Relationship between a dimensioned PDV and its PLC variable

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.

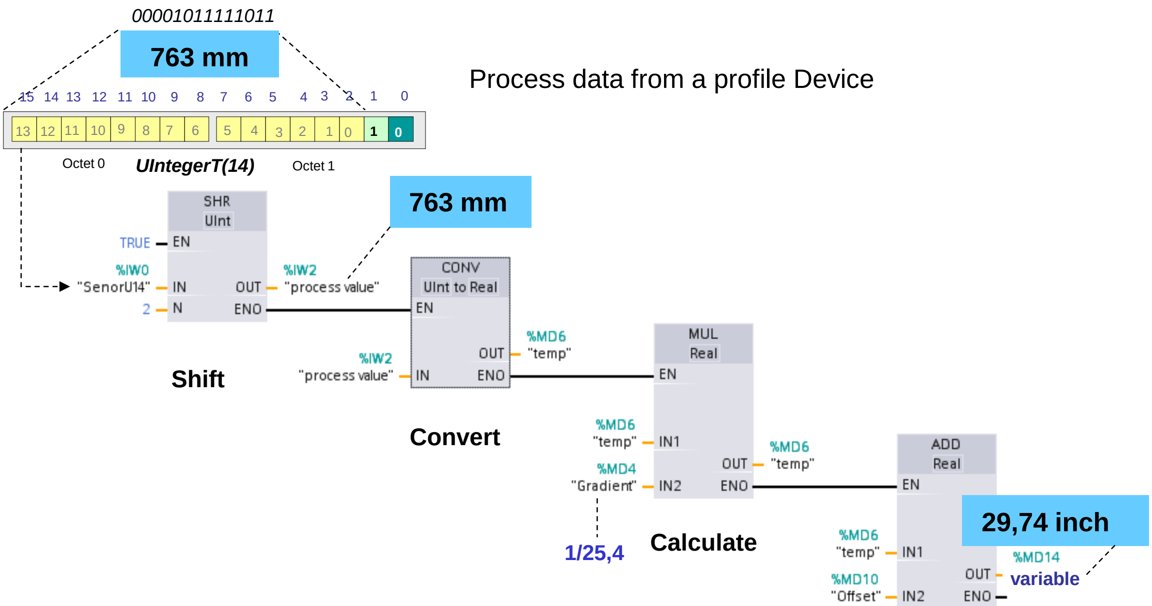


Figure E.8 – Example PLC program for a measurement value conversion

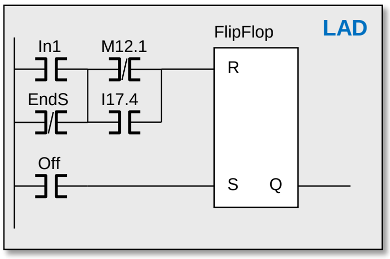
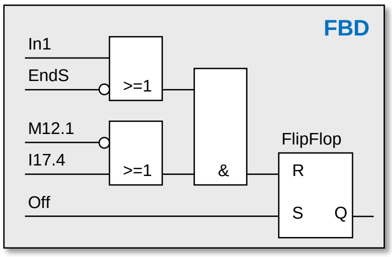
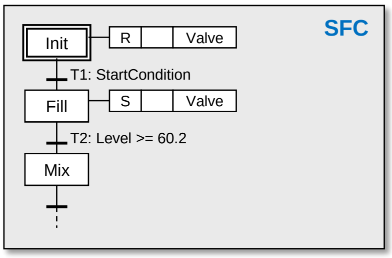
E.3 Integration via PROXY function block (IEC 61131-3)

E.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.

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.
IL U(O In1 ON EndSw) U(ON M12.1 O I17.4) R FlipFlop U Off S FlipFlop	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.
ST VAR MAX : INT := 10_000; MIN : INT := -5_000; END_VAR BEGIN IF IN > MAX THEN OUT := MAX; ELSEIF IN < MIN THEN OUT := MIN; ELSE OUT := IN; END_IF;	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.

E.3.2 Function Blocks and Function Calls according IEC 61131-3

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

a) 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.

b) 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.

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

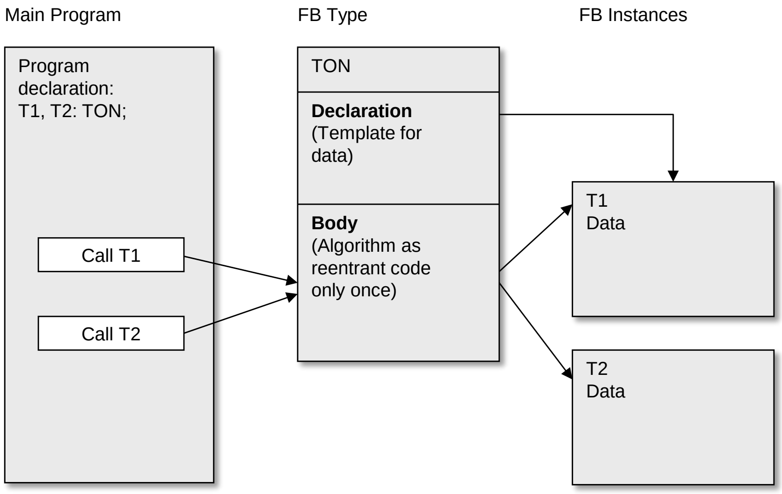


Figure E.9 – Function block and its instance data

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.

E.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

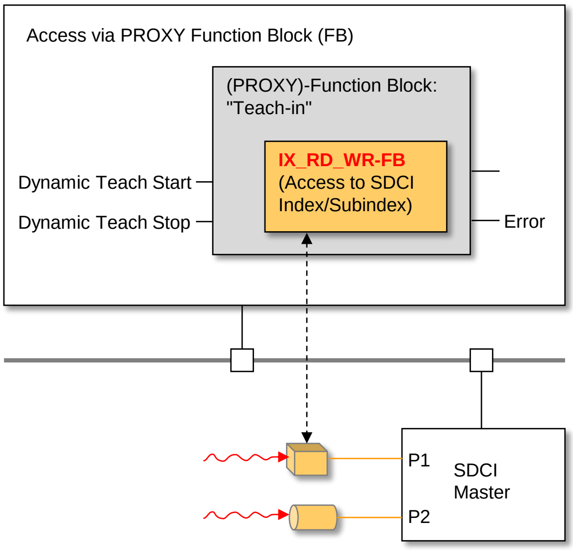


Figure E.10 – PROXY FB using standardized SDCI communication FB

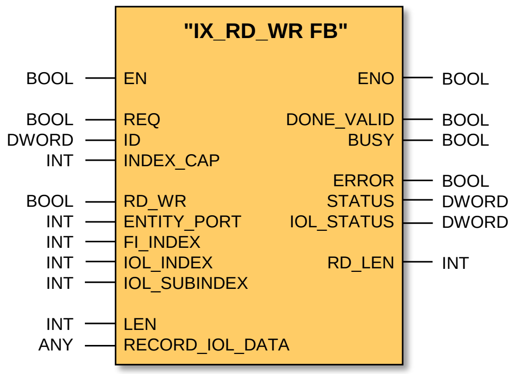


Figure E.11 – Standardized SDCI communication FB "IX_RD_WR"

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.

E.3.4 Device profile activities

It is the responsibility of the Device profiles to specify a proxy function block for each Common-ApplicationProfile 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.

E.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.

Annex F
(informative)
Scaling and dimensions

F.1 Gradient and offset

Normally, the ProcessDataVariable (PDV) of a profile Device carries a measurement value of a physical or chemical quantity within the data structures (PDV) defined by the manufacturer/vendor of the Device. See clause A.3.2 for details.

The transmitted value can be converted into a dimensioned value (°F, °C, inch, m, etc.) via a linear equation $y = m \cdot x + b$. "m" represents the slope and "b" the intercept with the y coordinate. Within these profile definitions, "Slope" is called "Gradient" and the value of the intercept is called "Offset". Figure F.1 illustrates the relationships.

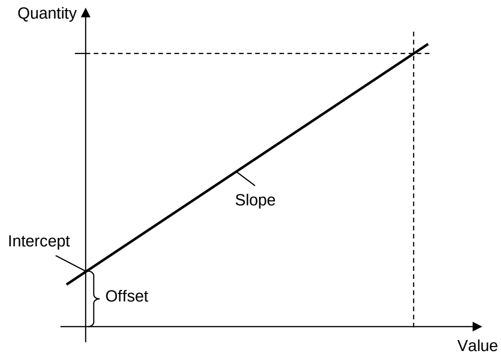


Figure F.1 – Value to quantity conversion via linear equation

The manufacturer/vendor is responsible for the provision of the "Gradient" and the "Offset" values for the conversion equation (1).

$$Variable = Gradient \times PDV + Offset \qquad (1)$$

F.2 Conversions

Usually the data type for Gradient and Offset is Float32T (Überarbeiten!). With the help of this information any computer software or PLC can calculate the dimensioned variable out of the transmitted PDV. Figure F.2 illustrates two conversion examples for pressure and temperature.

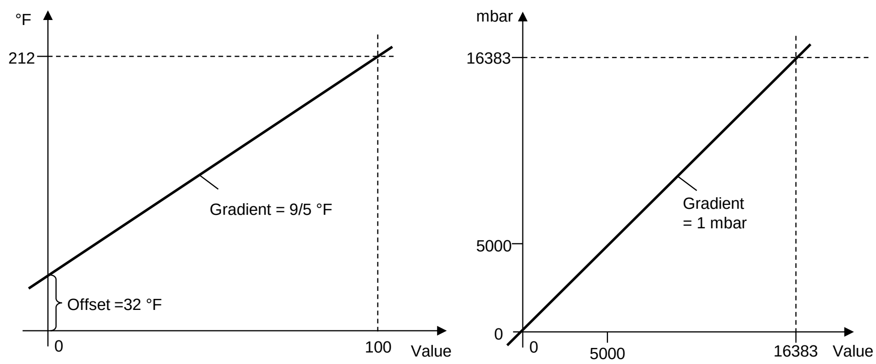


Figure F.2 – Conversion examples

Usually, the transmitted PDV value is based on a dimensioned measurement value as shown in the right example of Figure F.2 (pressure in mbar). In the left example a dimensioned temperature measurement value (°C) is converted in °F.

Annex G
(normative)
Profile testing and conformity

G.1 General

G.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).

G.1.2 Issues for additional testing/checking of profile Devices

- Identification complete and correct?
- Descriptors available and correct?
- All rules observed?
- Switching behavior conform to the specification?
- FunctionClasses available and correct?
  - Indices available and correct?
  - Read/write correct?
  - Data structures: Record? Value ranges?
  - Behavior of the FunctionClass conforms to the specification?
- Extract BDCs (switching functions) from user manual or IODD and check conformity with the specification
- Checklist: checkbox "relevant" and checkbox "verified"
- IODD: see [3]

Annex H
(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

Bibliography

[1] IO-Link Community, *IO-Link Interface and System*, V1.1.2, July 2013, 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, July 2011, 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*, V1.1.2, July 2014, Order No. 10.032

[7] IO-Link Community, *IO-Link Smart Sensor Profile Ed.2*, V1.0, March 2017, 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/

IO-Link