PUBLISHED DOCUMENT

PD IEC/TR
62390:2005

Common automation device — Profile guideline

ICS 25.040.40; 35.240.50

BSI
British Standards

National foreword

This Published Document reproduces verbatim IEC/TR 62390:2005.

The UK participation in its preparation was entrusted to Technical Committee GEL/65, Measurement and control, which has the responsibility to:

— aid enquirers to understand the text;

— present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;

— monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international publications referred to in this document may be found in the *BSI Catalogue* under the section entitled "International Standards Correspondence Index", or by using the "Search" facility of the *BSI Electronic Catalogue* or of British Standards Online.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the IEC/TR title page, pages 2 to 62, an inside back cover and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 30 March 2005

ISBN 0 580 45688 9

TECHNICAL REPORT

IEC TR 62390

First edition
2005-01

Common automation device – Profile guideline

Reference number
IEC/TR 62390:2005(E)

CONTENTS

INTRODUCTION

This guideline is a recommended outline for use by standardization product committees, fieldbus consortia and product manufacturers to develop and provide profiles for networked devices. Some aspects of this guideline may also be applicable to stand-alone devices. The present wide variation in the form of concepts and methods used for disclosing device information and behaviour to users of devices leads to longer evaluations required to understand how to use and apply networked industrial devices. This variation makes determining device interoperability, interchangeability, comparisons and common device behaviour more difficult. Therefore, it is the intention of this guideline to provide a common and more generic way to publish device information and behaviour. This is a contribution to reduce the total cost of the industrial control system.

Profiles define a common set of functionality for a class of devices in a given industrial domain, thus allowing system designers, system integrators and maintenance staff to handle profile-based devices without special tool configuration. They also allow consistent structuring and semantics of device functionality.

NOTE Other technologies are available to support the integration of devices into control systems, in particular to handle manufacturer-specific extensions in commissioning and engineering tools. Examples of such technologies are device description languages, which detail the internal structure of the device, or standardized software interfaces, where each device is represented by a dedicated software component.

Figure 1 shows the various possible profile documents and the typical writer of each document. The figure also illustrates the developing sequence for the developing of the profile documents. It is proposed that this profile guideline be the base for other working groups to develop profile standards and product class profiles. The root device profiles and the manufacturer device profiles can be developed from these profile standards. Finally, the manufacturer can create the specific device descriptions for his products. Any shortcut is possible between device profile documents.

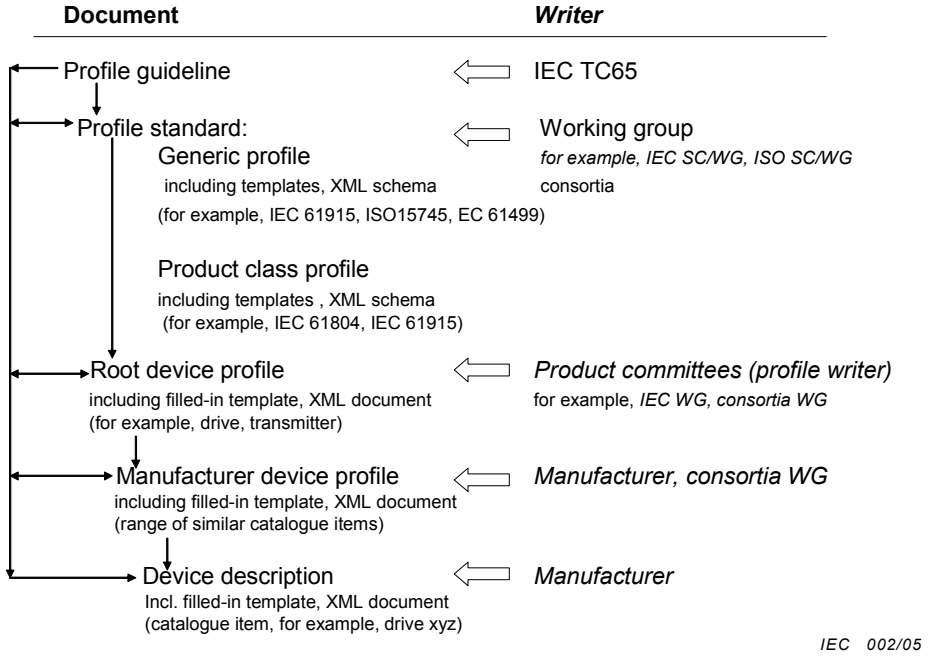


Figure 1 – Profile documents and their profile writer

This guideline provides the context, recommended minimum contents and construction rules for device profiles. Recommended generic device models, appropriate analysis and design diagrams using standards as UML (Unified Modeling Language) and methods to construct those models are provided.

This guideline provides recommendations for conveying the necessary device information to non-human users of the device profile such as software tools and application programs in an electronic file. These recommendations include the use of standards such as XML (eXtensible Markup Language).

COMMON AUTOMATION DEVICE – PROFILE GUIDELINE

1 Scope

This Technical Report provides guidance for the development of device profiles for industrial field devices and control devices, independent of their complexity.

NOTE 1 Examples of devices covered are limit switches and contactors for simple device networks, medium complex devices, such as transmitters and actuators for process control, and complex devices for fieldbuses, such as power drive systems.

NOTE 2 This guideline is also recommended to be used for devices such as programmable controllers, network components and HMI. If a device is user programmable, its features, as introduced in this guideline (for example, parameters and behaviour), cannot be completely described in the profile. However, profile writers may agree on general common functions like Start, Stop and Reset as well as identification and process inputs/outputs.

A device profile may cover various aspects such as physical, functional, communication, electrical and functional safety as well as application system aspects, irrespective of whether these aspects are accessible over the network. This guideline focuses on the functional aspects of the device (see 3.1.9).

NOTE 3 Different users of a device profile such as device manufacturers, system integrators and maintenance operators may only use specific aspects of the profile.

The guideline is written in a network independent way. Therefore, it is applicable for various fieldbuses, including those based on Ethernet. The guideline is intended to be used by IEC product standards committees and industrial communications networks consortia when they develop their device profile organizations and structures. It is not intended to provide an outline for a specific device profile. Further, this guideline presents device models to better guide and delineate a device profile's content. The profile guideline allows the use of a parameter list, function block model and/or object model to convey the structure and behaviour of the device in a unique manner. It is up to the profile writers to decide which of the models they apply.

To be useful to users a common method for conveying the device profile information is required. This guideline recommends the use of device profile templates. This guideline gives an example of a template, which is intended to be the basis of the structure and content of further templates which may be developed by the relevant profile groups.

This will allow users of these profiles to make comparisons, determine interoperability and interchangeability, and recognize common device behaviour.

The development of industrial application and process profiles, as covered by ISO 15745-1, is not within the scope of this guideline.

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 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

IEC/PAS 61499-1:2000, *Function blocks for industrial-process measurement and control systems – Part 1: Architecture*

IEC/PAS 61499-2:2001, Function blocks for industrial-process measurement and control systems – Part 2: Software tools requirements

IEC/PAS 61804 (all parts), *Function blocks (FB) for process control*

IEC/PAS 61804-2:2004, *Function blocks (FB) for process control – Part 2: Specification of FB concept and Electronic Device Description Language (EDDL)*

ISO 15745 (all parts), *Industrial automation systems and integration – Open systems application integration framework*

ISO 15745-1:2003, *Part 1: Generic reference description*

3 Definitions and abbreviations

3.1 Definitions

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

3.1.1
algorithm
completely determined finite sequence of instructions by which the values of the output variables can be calculated from the values of the input variables

[IEV 351-11-21].

3.1.2
application program
software *functional element* specific to the solution of a problem in industrial-process measurement and control

NOTE An application may be distributed among *resources*, and may communicate with other applications.

3.1.3
attribute
property or characteristic of an entity

3.1.4
class
description of a set of *objects* that share the same *attributes*, *operations*, methods, relationships, and semantics

[UML V1.5]

3.1.5
data
reinterpretable representation of *information* in a formalized manner suitable for communication, interpretation or processing

[ISO 2382, 01.01.02]

3.1.6
data type
set of values together with a set of permitted *operations*

[ISO 2382, 15.04.01]

3.1.7
device
field device
1. networked independent physical *entity* of an industrial automation system capable of performing specified functions in a particular context and delimited by its *interfaces*

[IEC 61499-1]

2. entity that performs control, actuating and/or sensing functions and interfaces to other such entities within an automation system

[ISO 15745-1]

3.1.8
device class
set of devices with a defined functional commonality in terms of their *parameters* or *functional elements*

3.1.9
device profile
representation of a device in terms of its parameters, parameter assemblies and behaviour according to a device model that describes the data and behaviour of the device as viewed through a network, independent from any network technology;

NOTE 1 This is a definition from IEC 61915 which is extended by the addition of the device functional structure.

NOTE 2 The mapping onto a given network technology is the task of the communication profile.

3.1.10
entity
particular thing, such as a person, place, *process*, object, concept, association, or *event*

[dpANS X3.172, 1989].

3.1.11
execution
process of carrying out a sequence of *operations* specified by an *algorithm*

3.1.12
functional element
entity of software or software combined with hardware, capable of accomplishing a specified function of a *device*

NOTE 1 A functional element has an interface, associations to other *functional elements* and functions.

NOTE 2 A functional element can be made out of function block(s), object(s) or parameter list(s).

3.1.13
function block
software functional element comprising an individual, named copy of a data structure and associated *operations* specified by a corresponding *function block type*

NOTE Adapted from IEC 61499.

3.1.14
input data
data transferred from an external source into a *device*, *resource* or *functional element*

3.1.15
instance
functional element comprising an individual, named copy of a data structure and associated *operations* specified by a corresponding *functional element type*

3.1.16
interface
shared boundary between two *functional units* defined by functional characteristics, signal characteristics, or other characteristics as appropriate

[IEV 351-11-18].

NOTE The interface typically includes the device parameters.

3.1.17
method
implementation of an operation, which specifies the algorithm or procedure associated with an operation

3.1.18
model
mathematical or physical representation of a system or a process, based with sufficient precision upon known laws, identification or specified suppositions

[IEV 351-11-20].

3.1.19
object
entity with a well-defined boundary and identity that encapsulates state and behaviour

[UML V1.5]

NOTE State is represented by attributes and relationships, behaviour is represented by operations, methods, and state machines. An object is an instance of a class.

3.1.20
operation
service that can be requested from an object to effect behaviour

[UML V1.5]

3.1.21
output data
data originating in a device, resource or functional element and transferred from them to external systems

3.1.22
parameter
data element that represents device information that can be read from or written to a device, for example, through the network or a local HMI

NOTE 1 Adapted from IEC 61915.

NOTE 2 A parameter is typically characterized by a parameter name, data type and access direction.

3.1.23
resource
– logical device
– module
– group of functional elements which has independent control of its operation, and which provides various services to *application programs*, including the scheduling and execution of algorithms

NOTE The RESOURCE defined in IEC 61131-3 is a programming language element corresponding to the *resource* defined above.

3.1.24
service
specific work performed by a *device* or *object*

3.1.25
type
hardware or software element which specifies the common *attributes* shared by all *instances* of the type

3.1.26
use case
class specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system

[UML V1.5]

3.1.27
variable
software entity that may take different values, one at a time

[ISO 2382]

NOTE The values of a variable as well as of a parameter are usually restricted to a certain data type.

Abbreviations AIP Application Interoperability Profile

DCS Distributed Control System

ERP Enterprise Resource Planning

FBD Function block Diagram

HMI Human Machine Interface

H/W Hardware

I/O Input/Output

MES Manufacturing Execution System

OMG Object Management Group

S/W Software

UML Unified Modeling Language

URL Universal Resource Locator

XML Extensible Markup Language

4 Guideline overview

The device profile guideline

– presents a short introduction to the entire scope of profiles;
– specifies the subset which is the focus of this guideline;
– introduces a general structural view to a device.

A sequence of six profile definition steps is proposed to the profile writer groups to develop the necessary information for a device profile. This is recorded in a profile template, which is introduced in a corresponding clause. The profile template is to be collected in an electronically readable form and in a printed human readable document.

This guideline is based on the three typical approaches used in the automation industry:

– the parameter list model;
– the function block model; and
– the object model.

The guideline recommends using one of these three models. As a minimum, the parameter list model should be applied to be in line with this guideline. Further models based on the parameter list model are possible provided that they may be mapped to one of the models.

Special annexes provide the model-based background of the profile development steps and the template.

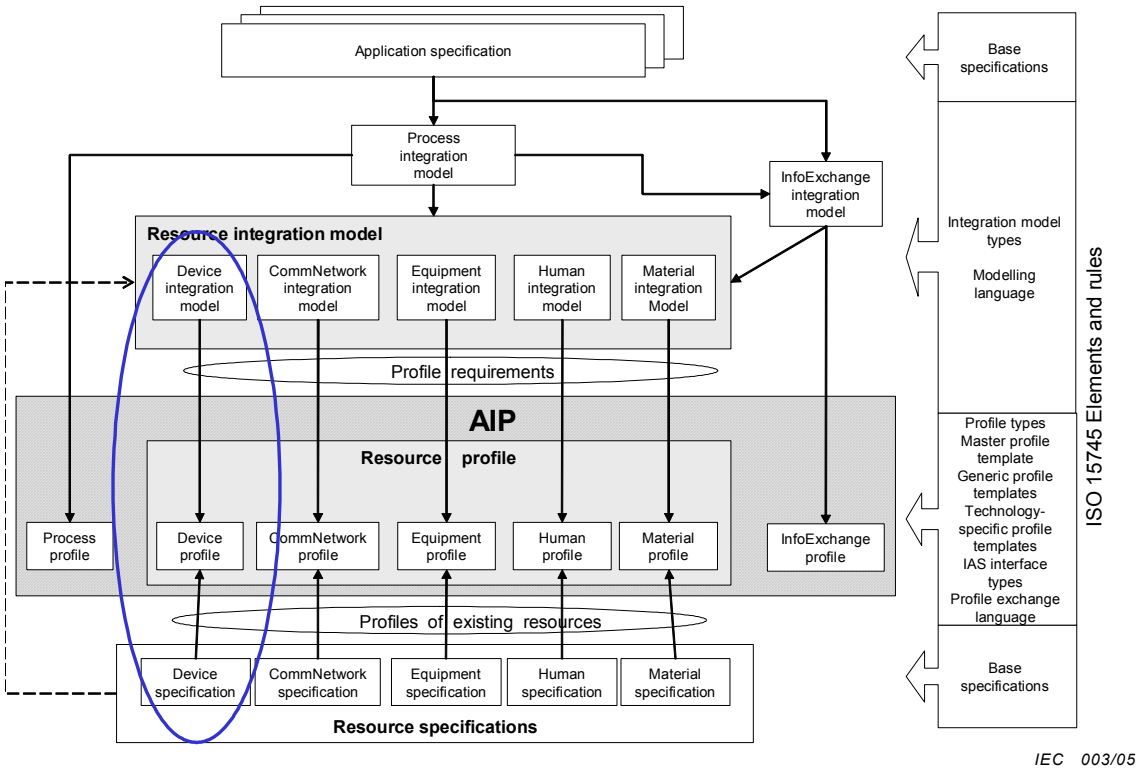
Several annexes provide additional information and material for the profile writer groups.

5 Automation model and device profiles

5.1 ISO 15745

There are several aspects to be considered during device profiling. Figure 2 shows where the device profile fits in relation to other profiles necessary to build an automation application. Figure 3 shows a typical hardware implementation of an automation application. Figure 4 shows an example of a functional device structure.

According to ISO 15745, a general application system includes the technological process which has a material and energy flow, equipment and devices which carry these flows, devices which carry out the information processing, the communication systems connecting these devices as well as the interaction of human beings with the devices and, at least, the process. The ISO 15745-1 model contains each component of this system (see Figure 2). The automation devices are a subset of this entire framework, which is within the scope of this profile guideline. Therefore, this profile guideline deals with the device model and device profile parts of ISO 15745-1 only (the scope is highlighted in Figure 2 using an ellipse). The communication network integration model, profile and specifications are outside the scope of this guideline.



Key

——> indicates order in which activities are performed

- – - > indicates information flow

Figure 2 – Profile development using ISO 15745-1

5.2 Typical automation configuration

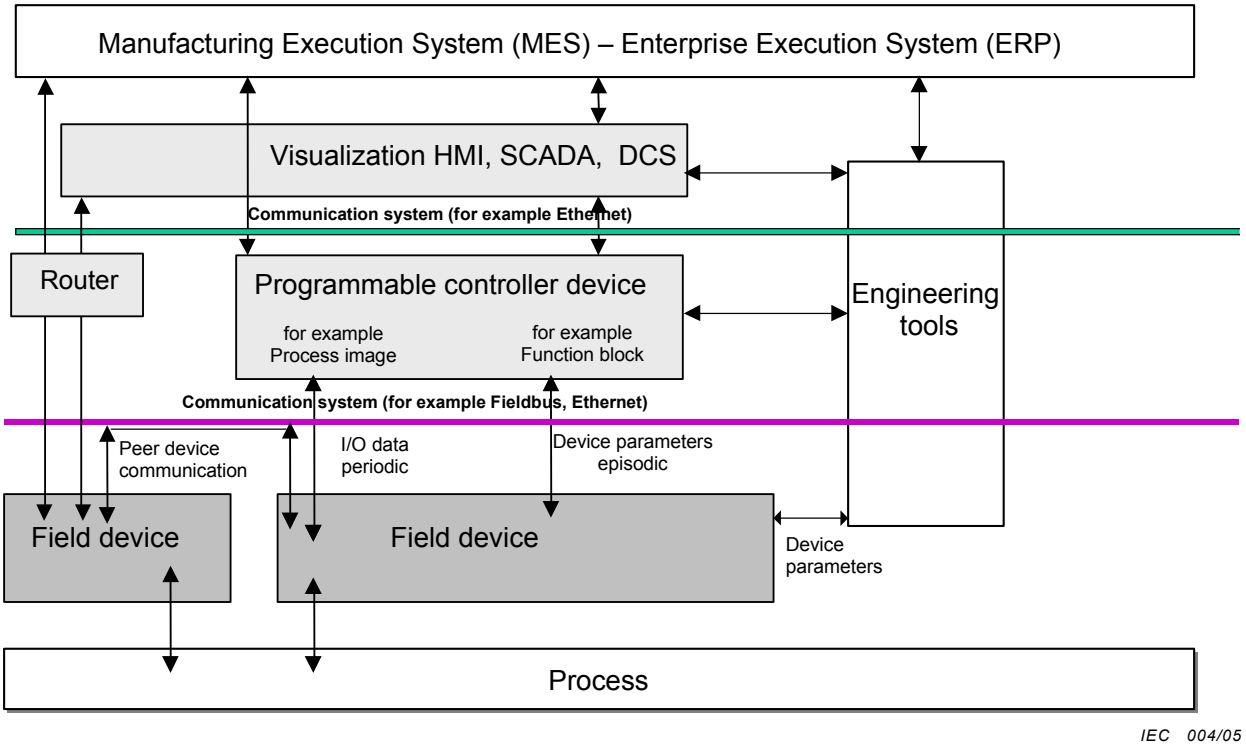
The field device is typically integrated as a component in an industrial automation system. The automation system performs the automation-related part of the entire application system. To define the complete profile of a specific field device class, it is necessary for the profile writers to agree on the interactions of the device with the other components of the automation system: functions and corresponding required interfaces (including the defined device parameters). The components of an industrial automation system may be arranged in multiple hierarchical levels connected by communication systems as illustrated in Figure 3.

The field devices are components in the application system connected via inputs and outputs to the process or the physical or logical subnetworks. This also includes programmable devices and routers or gateways.

A communication system (for example, a fieldbus) connects the field devices to the upper level controllers, which are typically programmable controllers or Distributed Control Systems (DCS) or even Manufacturing Execution Systems (MES). Since the engineering tools and the commissioning tools should have access to the field devices as well as to the controllers, these tools are also located at the controller level. The “intelligent” field devices may communicate direct with each other via the fieldbus or the controller (Programmable Controller).

In larger automation systems another higher level may exist, connected via a communication system like LAN or Ethernet. In these higher level visualization systems (HMI), DCS, central engineering tools and SCADA are located. Multiple clusters of field devices (with or without a controller, as described above) may be connected over the LAN with each other or to the higher level systems.

Manufacturing Execution System (MES), Enterprise Resource Planning (ERP) and other Information Technology (IT) based systems can have access to field devices indirectly via the LAN and the controllers or directly via routers.



NOTE Dark-grey boxes represent field devices which are in the main scope of profiling. Light-grey boxes are network-connected devices which may also be considered as devices within the scope of the guideline.

Figure 3 – Typical automation application system

5.3 Modular device structure

The device can be structured in a hierarchical way as shown in Figure 4. The main components of the structure can be containers, which are known as modules (for example, in the remote I/O domain), resources (for example, in IEC 61131-3 and IEC 61499) or logical devices (for example, within some fieldbuses), which can be further subdivided into *functional elements*. *Functional element* is a generic term for parameter list members, function blocks and objects. All *functional elements* have parameters and optional behaviour. Modules, resources and logical devices as well as the *functional elements* can be hierarchically structured.

In some cases, the hierarchy of a device, a resource (module/ logical device) and a *functional element* can be collapsed, for example, if a device has only one resource with a single *functional element*, it may only provide a parameter list.

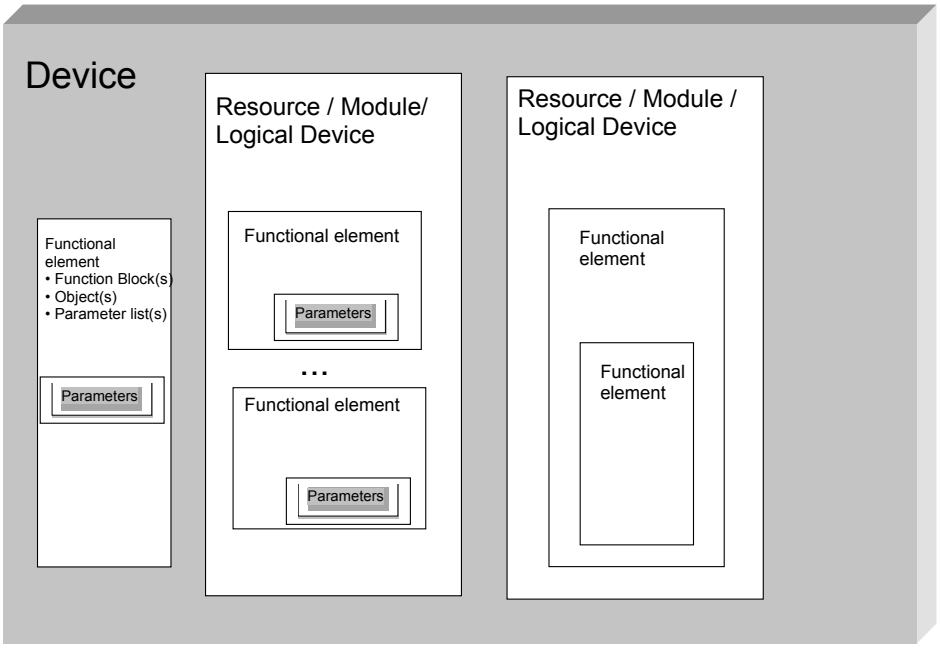


Figure 4 – Example of a modular view of the hardware and software structures of a device

The device structure shown above is formally defined in the UML class diagram in Figure 5 (see Annex F).

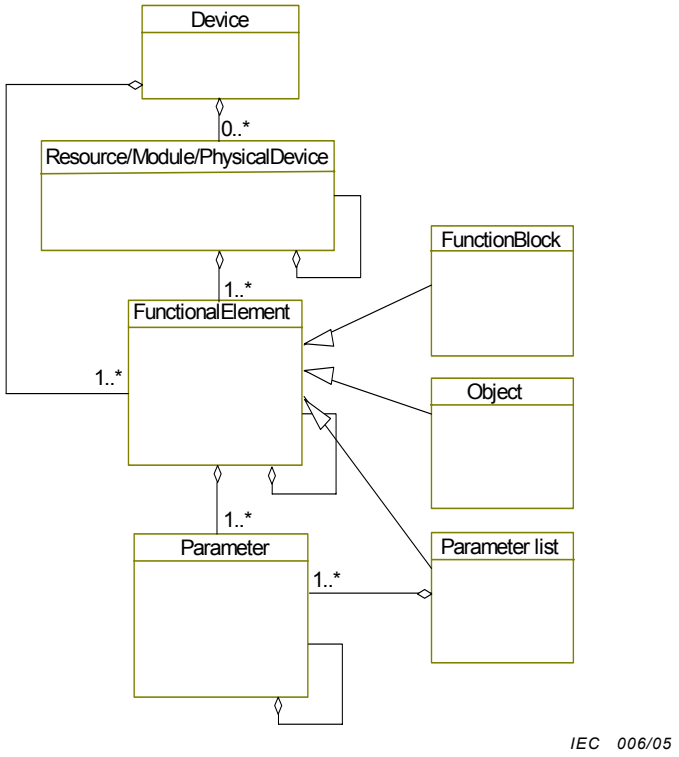


Figure 5 – Example of a device structure class diagram

5.4 Interface model

A device may also be modelled from an interface point of view if its internal structure is not relevant. The structure of the interface can be derived from the roles the device plays during its operation.

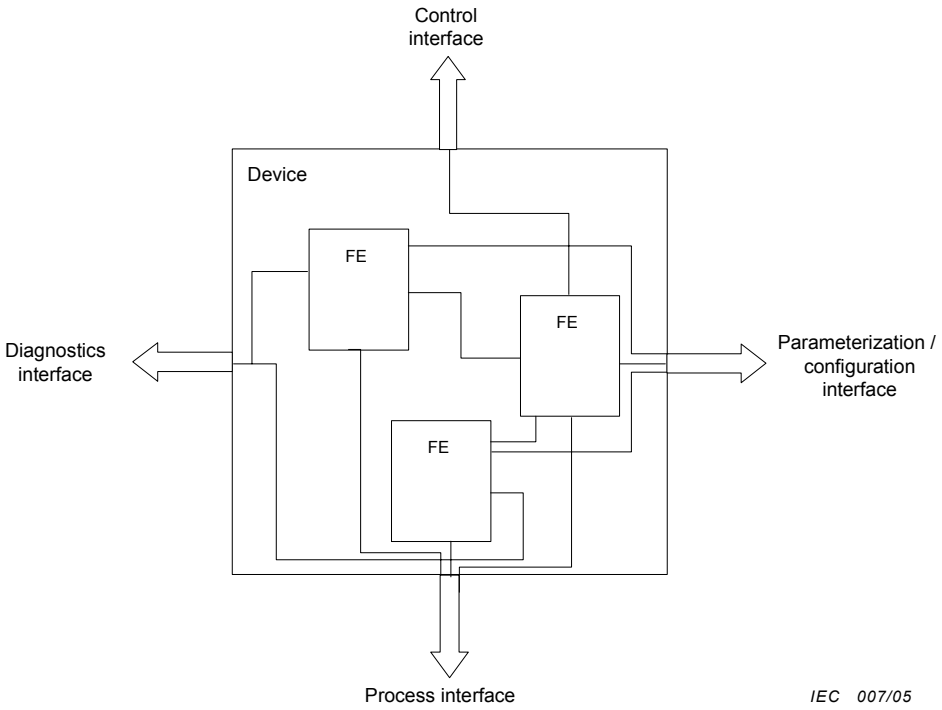
Figure 6 shows the following interfaces:

– process interface, which represents the attachment to the process;

– diagnostics interface, which represents all diagnostics information which is provided by a device;

– parameterization/configuration interface, which represents all structural and functional adjustments of the device during commissioning, operation parameterization and maintenance;

– control interface, which represents all data related to the control of other devices or higher control hierarchies.



NOTE Control, diagnostics and parameterization/configuration data is typically accessible via communication interfaces for network-connected devices.

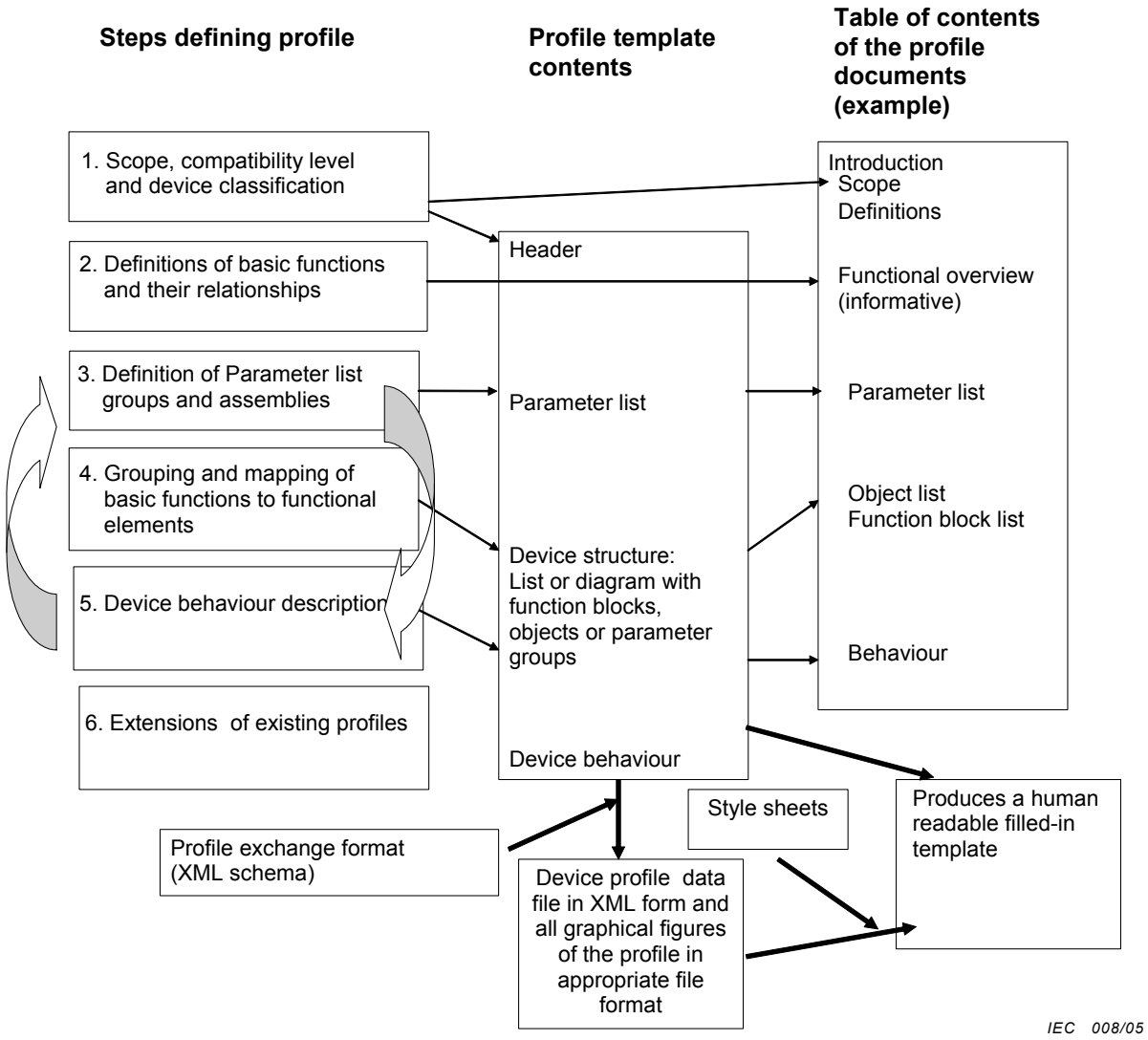
Figure 6 – General interface model of a device

6 Profile definition steps

6.1 Outline

The developing of field device profiles is a six-step process as illustrated in Figure 7. During this process all relevant information should be collected in the filled-in (completed) profile template. These steps are described in detail in 6.2 to 6.6. A profile document may provide additional explanations and background information.

The first step of each device profile development is the definition of the profile scope and its device classification. This means that the topic of interest has to be clarified by defining which device classes are the subjects of the profile. Additionally, it is very helpful to show the roles of the chosen device classes in the automation system. This helps to take decisions during the specification work.



NOTE Other machine-readable representations are also appropriate, for example, using the languages defined in IEC 61804-2, ISO 15745-2, ISO 15745-3, ISO 15745-4 or spread sheets.

Figure 7 – Profile definition steps

The second step determines the basic functions, i.e. the functionality of the device classes that are within the scope of the profile. A corresponding functional overview is dedicated to the users of the profile to understand the main functionality, which is accessible over the communication network but also to the profile writer to refer to the basics during the profile development process.

The third step defines the parameter list, which contains all parameters of the device that are accessible via the communication system. The parameter list is recorded in the parameter list section of the profile template. The parameter list defines the parameters application-specific characteristics, excluding communication-specific aspects. Profile groups may decide to stop the profile definition work at this level.

The fourth optional step groups parameters and related functions into so-called *functional elements*. The resulting *functional elements* are characterized by parameters and behaviour. This step transfers the functional overview developed in the second step into the visible device structure consisting of function blocks or objects, which is recorded in the device functional structure section of the profile template. Details are described in Clause 8.

The fifth optional step describes the behaviour of the device and/or *functional elements*. This is done separately and recorded in the device behaviour section of the profile template.

It will be helpful to repeat iteratively the execution of steps 3 to 5 to assure a complete analysis of the device.

The sixth optional step may be applied for the extension of an existing device profile to develop specific members of a device family. This may be done also directly during the execution of steps 1 to 5.

NOTE Extensions may be made either by adding parameters and *functional elements* to the base profile or by requiring support of optional items of the base profile.

The profile template is the human readable printed form of the device profile structure and, when filled in, contains the result of the profile development work. Additional profile documentation may be supplied to provide more detailed information, by extending the information captured in the filled-in template with explanations and additional text and figures.

A machine-readable representation of the device profile is also necessary for use by engineering tools. XML provides among others (for example, spread sheets) an appropriate technology (Figure 7). The use of XML is recommended.

If XML is used, the profile template structure should be represented as an XML schema. A device profile, i.e. the filled-in profile template is then represented by an XML file. Style sheets can be used to generate out of the device profile XML files different formats, which can then be used by browsers, text processors, or other tools. The details on how to use this technology are are given by the profile writers or their organizations.

6.2 First step: Scope, compatibility levels and device classification

6.2.1 Overview

The information flow within a system between devices goes from the information detection of the process (transmitters, input devices) through the information processing (control) to the information use at the process (actuators, drives, output devices). A typical automation configuration is given in Figure 8. The information flow is carried out by a signal flow along a chain of functions. Human Machine Interfaces (HMI) and information processing for maintenance and technical management accompany this chain. The device classification step chooses those devices that shall be under profile development. Additionally device identity information (such as device family information and manufacturer) are collected and defined in the first step.

6.2.2 Compatibility levels

6.2.2.1 Background

The following considerations should be carried out at the beginning of a profile development process. There are certain device profile groups, which deal with different device classes such as proximity switches, transmitters, or drives. Device products make use of the device profiles and add certain manufacturer-specific functionalities. The device manufacturer connects his device with several communication systems, which are also based on communication profiles provided by communication system organizations ((AX, AY, AZ), (BX, BY, BZ) or (CX, CY, CZ)). From the system point of view, there are products based on different device profiles using one certain communication system ((AX, BX, CX), (AY, BY, CY),

or (AZ, BZ, CZ)). When connecting devices to a system there are N^*M combinations of device profiles and communication systems, where N is the number of device profiles and M is the number of communication profiles. These numbers may be increased by manufacturer additions.

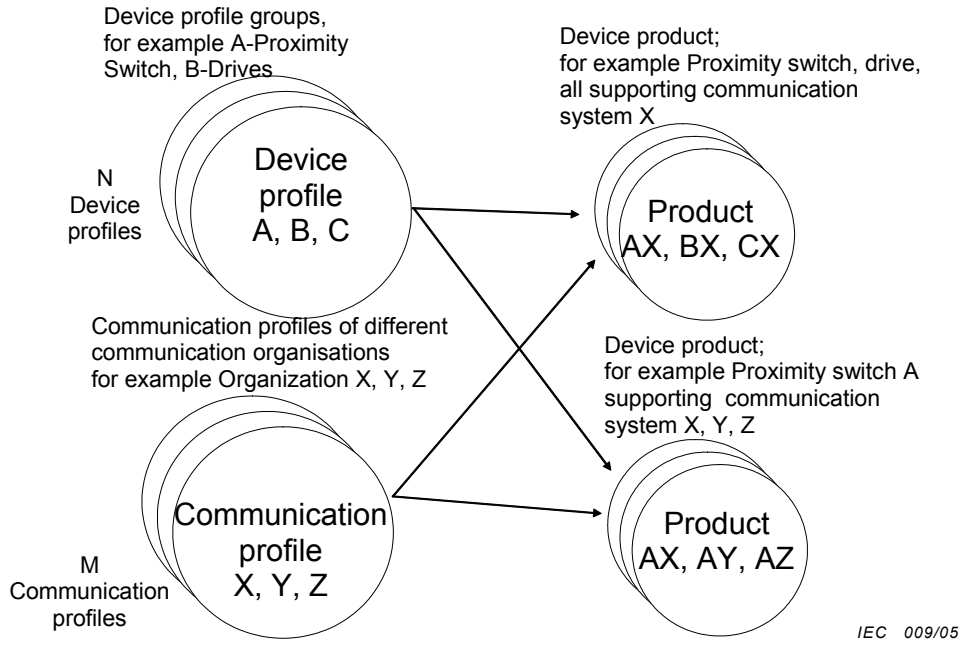
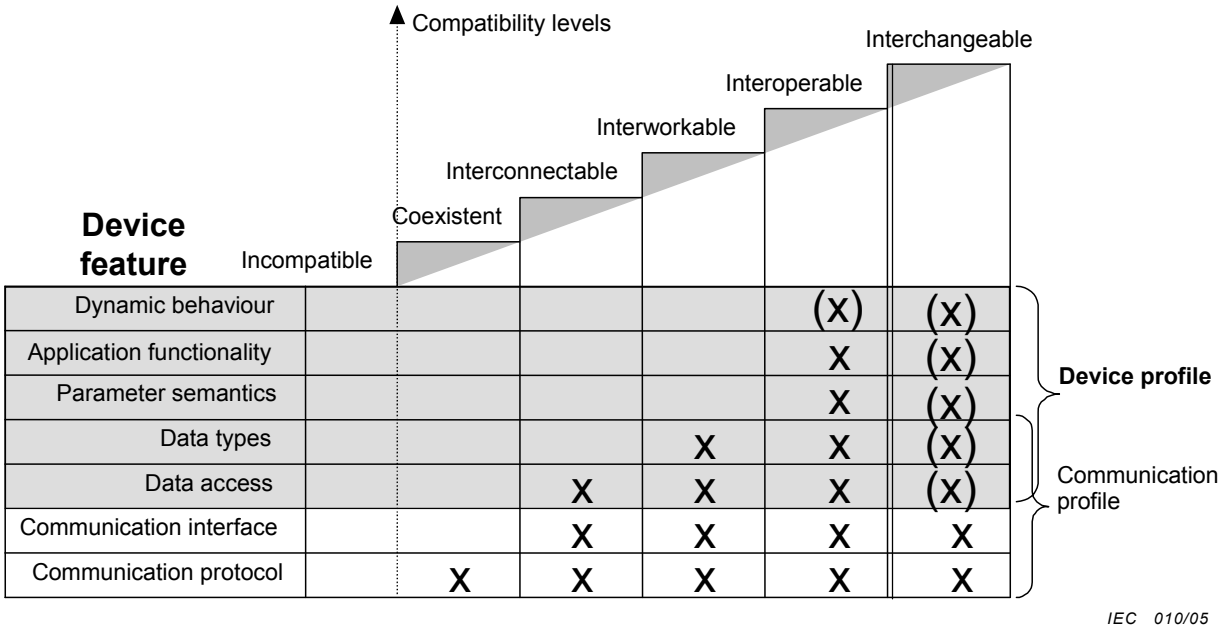


Figure 8 – Relations between profiles and products

There are certain degrees of compatibility and corresponding degrees of cooperation between profile-based devices (compatibility levels), as shown in Figure 9. The levels are dependent on well-defined communication and application device features.



NOTE 1 The definition of the compatibility level is intended for devices on the same communication platform.

NOTE 2 The communication profile is not within the scope of this guideline

Figure 9 – Levels of functional compatibility

The device features are defined as follows.

Table 1 – Device application and communication features

Device feature	Description
Communication profile	
Communication Protocol (lower layers)	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
Communication Interface (upper layers)	This feature is defined by the protocols of layer 5 to 7 of the OSI reference model including the services and the service parameters. Additional mapping mechanisms can be necessary. The dynamic performance of the communication system is part of this feature
Device profile and Communication profile	
Data access	This feature is defined by characteristics such as “parameter timing” and “access direction” (see Table B.1)
Data types	This feature is defined by characteristics such as “data type” (see Table B.1)
Device profile	
Parameter semantics	This feature is defined by the parameter characteristics: parameter name, parameter descriptions, parameter range, substitute value of the parameter, default value, persistence of the parameter after power loss and deployment
Application functionality	This feature is defined by specifying the dependencies and consistency rules within the *functional elements*. This is done in the parameter description characteristics or in the device behaviour section
Dynamic behaviour	This feature is defined by time constraints that influence the parameter update or the general device behaviour. For example, the update rate of a process value can influence block algorithms

The relation between device features and parameter characteristics defined in 6.4 is specified in Annex C.

6.2.2.2 Incompatibility

Two or more devices are incompatible if they cannot exist together in the same distributed system.

NOTE Incompatibility can result from differences in application functionality, parameter semantics, data types, communications interface, or even communications protocols used by the affected devices. Incompatible devices may even interfere with, or prevent, each other’s proper communication or functioning (possibly even destructively), if placed in the same physical communication network.

6.2.2.3 Coexistence

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 use of other devices on the network.

6.2.2.4 Interconnectability

Two or more devices are interconnectable if they use the same communication protocols, communication interface and data access.

6.2.2.5 Interworkability

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.

6.2.2.6 Interoperability

Two or more devices are interoperable if they can work together to perform a specific role in one or more distributed application programs. 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.

6.2.2.7 Interchangeability

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 system if the new device has the functionality to meet the application requirements.

NOTE Full interchangeability regarding the entire device performance is nearly impossible to achieve. However, actual device interchangeability is dependent on the application requirements for this device.

Different degrees of interchangeability may be applicable for various roles of a device, for example, control, diagnosis, parameterization/configuration. That means that one device can have different degrees of interchangeability regarding different interfaces to the system.

The profile writer may want to qualify these degrees of interchangeability between two devices (different versions or manufacturers) supporting a given device profile. This may be done using a table such as Table 2, the contents of which correspond to detailed profile specifications.

Table 2 – Interchangeability matrix for device exchange purpose

Device roles (interfaces)	Device features				
	Data access	Data types	Parameter semantic	Application functionality	Dynamic behaviour
Control	See NOTE	See NOTE	See NOTE	See NOTE	See NOTE
Diagnosis	See NOTE	See NOTE	See NOTE	See NOTE	See NOTE
Configuration	See NOTE	See NOTE	See NOTE	See NOTE	See NOTE
Parameterisation	See NOTE	See NOTE	See NOTE	See NOTE	See NOTE
Process Interface	See NOTE	See NOTE	See NOTE	See NOTE	See NOTE
...	See NOTE	See NOTE	See NOTE	See NOTE	See NOTE
NOTE For each role/interface of a device, the profile writer should specify for each device feature the exchangeability level using keywords such as “not applicable”, “not defined”, “defined”, “manufacturer specific”.					

6.2.3 Device classes

There are already various classification overviews for measurement and actuation devices. These activities standardize the structure of device manuals and, additionally, the semantics of technical terms, for example, environmental conditions, signal input and others. They point out the relation to already existing international standards and provide at least a taxonomy of measurement and actuation devices. The basic example of automation device classification is provided in Annex G. A device class is a set of devices with a defined functional commonality in terms of their *parameters* or *functional elements*.

This step shall result in an agreement on a common scope of the profile, a specific device class or device family, a commonly targeted level of compatibility of the discussed devices and the necessary information for the profile template and documentation. As shown in Figure 7, all relevant information of this step is recorded in the header section of the profile template as shown in 7.2.5.

6.3 Second step: Definition of device functions and their relations

The device is described in a top-down approach based on a black-box model, i.e. starting with the external interface (for example, process input/output) or connection (for example, sensor, valve, motor) and the main control input/output (for example, set point, measurement value). This first model is detailed by stepwise refining of the main signal flow between device functions. The degree of details depends on the device class. Different device subclasses may be introduced at a detailed level. Devices may have certain subclasses, which provide functions for different purposes. These subclasses should be shown.

The overview of device functions provides the functional structure of the chosen device class (for example, drives). It is possible that multiple functional diagrams are necessary to cover the device subclasses (for example, AC and DC drives) of the profile.

The list of device functions is not part of the profile template.

Functional diagrams (see Figure 10) are the main descriptions of this step that should be accompanied by textual descriptions. Additionally, it is recommended that use cases and scenarios be defined (for example, using UML as shown in Figure 11) for which the device profiles are defined.

The example in Figure 10 shows a non-standardized functional diagram of a power drive system. The purpose is to express the main device functions and their relation, which reflect the common view of the device profile group for a specific device class. For example, the PDSs (Power Drive Systems) have hundreds of functions which might be considered. It is not possible to consider all of them. Therefore, a significant choice has to be made.

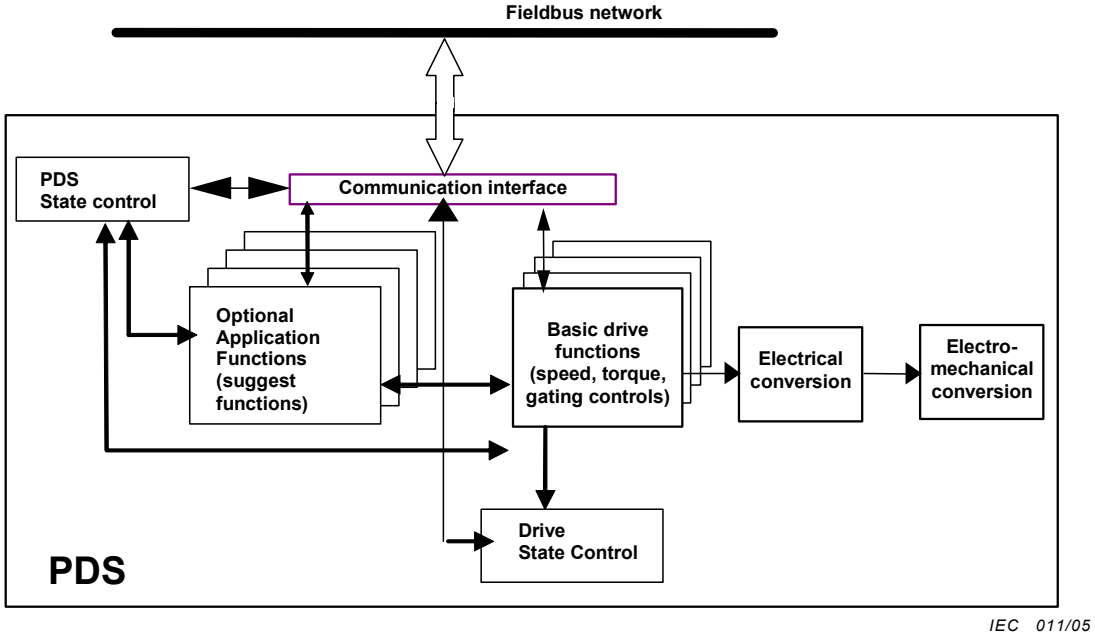


Figure 10 – Example functional diagram of a power drive system (PDS)

6.4 Third step: Parameter list definition

The parameter list defined in this step contains all accessible parameters of a device. The definition of the parameters can follow various procedures, for example,

– derivation of the parameters out of the device functions (see 6.3);

– considerations of life-cycle aspects (see Annex A);

– investigations of the device use cases (see Figure 11).

Examples of parameters are as follows:

– input variable of the data flow (for example, signal, set point);
– output variable of the data flow (for example, signals, status, device state);
– data used for the configuration (change of the functional structure) and for the adjustment of device functionality (for example, tuning);
– mode is a special case of configuration which selects the active functions (for example, manual, automatic, jog; position, speed, torque);
– status data which reports internal behaviour (for example, warning, error, fault, overload, limit alarm, accelerating);
– state of a device (for example, initialization, operation, stand-by, out-of-service; running, stopping, stopped);
– service interface for triggering an event to cause a transition of a state machine with and without parameters (for example, run command, stop command; calibration command);

NOTE 1 The use of parameters is one possible implementation of a service interface.

– object attribute as defined in the object model which may represent some of the above examples.

NOTE 2 Other examples are included in IEC 61915, Annex E.

NOTE 3 The transportation of parameters is technology dependent. It can be done, for example, using read, write and messages.

These parameters have characteristics (for example, name and data type) that allow the user of the device to properly provide, use and display a parameter value. A representative collection of possible parameter characteristics is provided in Annex B. Depending on the functional compatibility level (see 6.2.2) the profile writer intends to achieve, a certain subset of parameter characteristics shall be defined. The “support” characteristic is a major one which may be used to extend a base profile, for example, by requiring support of optional parameters of the base profile.

The parameter list is recorded in the parameter list section of the profile template (see Clause 7).

Complex devices may have large numbers of parameters. For this purpose, parameter groups may be defined; a parameter group is a logical set of parameters which may be associated with the same function and/or use case of a device. The definition of parameter groups is optional.

A parameter assembly is a set of parameters which is accessible in a single network read or write service. Not all networks support parameter assemblies. Therefore, the definition of parameter assemblies is optional.

To define the parameter list as part of the profile template, one possibility is to start with use cases. Use case definition should follow the definition of UML V1.5. A use case specifies a sequence of actions that a system can perform, interacting with so-called actors (see UML V1.5 and Figure 11). From the profile definition point of view, all phases of the device life cycle shall be considered. Actors can be human beings as well as software and hardware components. Typical actors here are controller, PC-based tools and operators, which interact with the devices in terms of operation, parameterization, configuration and maintenance. The analysis of the interactions between the controller and PC tools and the device leads to a list of relevant parameters. Additionally, the interactions need a defined device behaviour which is also part of the device profile.

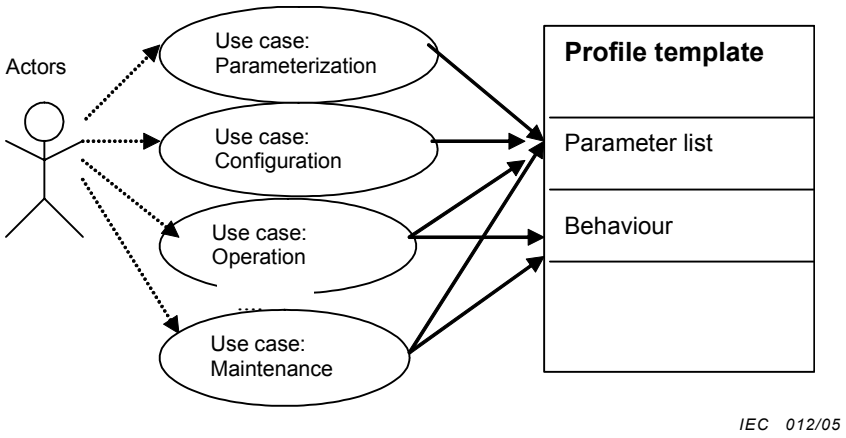


Figure 11 – UML use case examples

Examples of the possible role of actors and their actions are listed in Annex A.

Profile groups may decide to stop the profile definition work at this level.

6.5 Fourth step: Grouping of functions to functional elements

6.5.1 Description

The functional diagrams of a certain device (see step 2 of 6.3) give an overview of device functions and their relation to each other. The grouping of the functions to *functional elements* will be done in this profile development step.

In the industrial automation domain, there are different approaches on how to model devices. These approaches are described in the annexes as follows:

– parameter list model: see Annex H;

– function block model: see Annex I;

– object model: see Annex J.

The profile writer group shall agree on which model they will use for their profile development. To offer a common view of these three approaches the term *functional element* is introduced in this guideline. A *functional element* is either a parameter or a parameter group, a function block, or an object.

The defined functional elements are recorded in the device structure section of the profile template (see Clause 7).

6.5.2 Example of a flow transmitter using object model and function block model

Figure 12 and 13 show the structure of a flow transmitter which is modelled using either objects or function blocks.

NOTE The relationship between the two models is shown in Table 5.

The dotted ellipse of Figure 12 represents the application object class with an internal structure. Each subobject class is specified in detail in the corresponding profile. The flow transducer represents the hardware only and is outwith the scope of the corresponding profile. Parameter and assembly object classes are the interfaces of the application object class with the communication system. Message router and connection object class are communication-system specific classes.

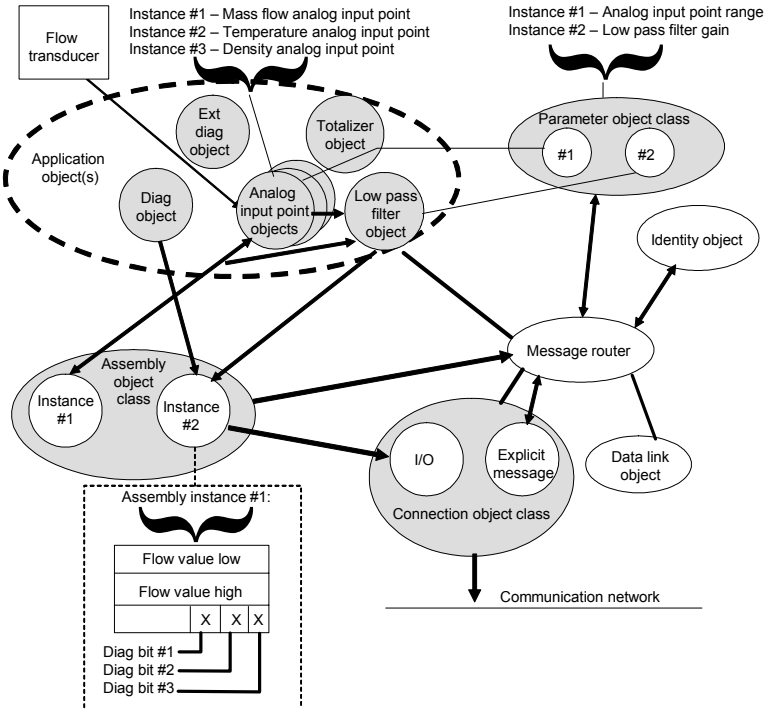


Figure 12 – Device functional structure of a flow transmitter based on the object model (example)

The flow transmitter example based on the function block model shows a detailed structure regarding the internal signal flow (Figure 13). This example models three process values (i.e. mass flow, density and temperature), each with a separate analog application signal processing. Additionally, a totalizer function block counts the mass flow. Communication services may access function block parameters direct or by using view function blocks.

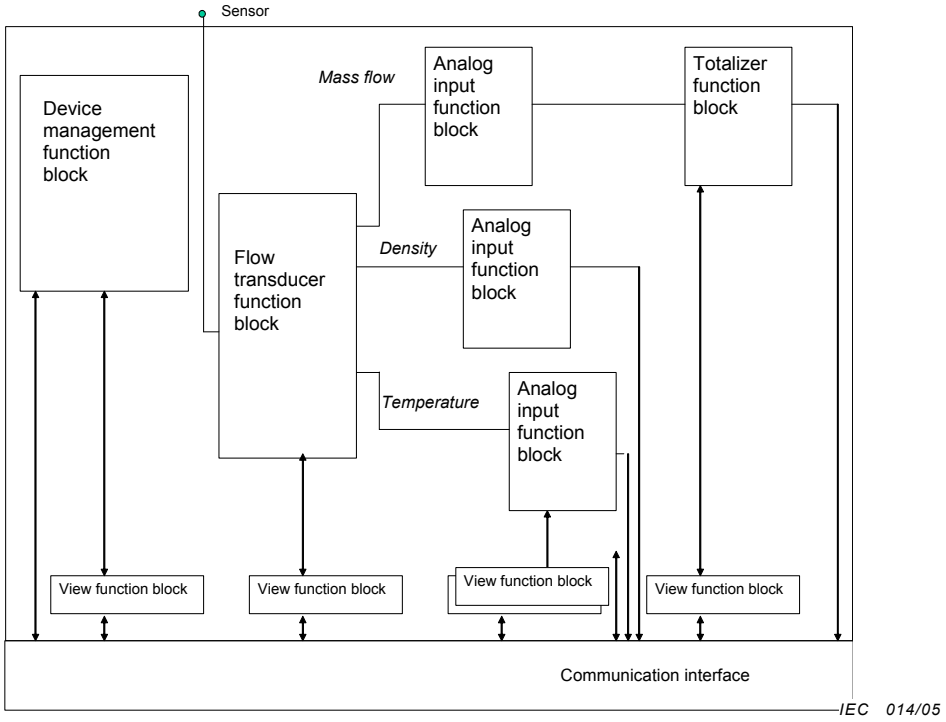


Figure 13 – Device functional structure of a flow transmitter based on the function block model (example)

6.6 Fifth step: Device behaviour description

Dependencies between parameters and resources cannot be easily expressed using parameter characteristics. Each *functional element* may have its own behaviour. Therefore, this step addresses the behaviour point of view.

Profile writers choose a relevant subset of the device behaviour. The behaviour of the device and/or function blocks and objects is composed of a set of algorithms and methods respectively. Behaviour is commonly described using the following.

– An algorithm in the mathematical sense (for example, normalize, scaling, filter, invert) which describes how to calculate output data from input data using parameters; one possible description of these algorithms are IEC 61131-3 functions.
– A sequence of algorithms inside one functional element including process and communication interactions (for example, alarm handling, calibration, start-up phase, time-dependent start of system self-test or sensor cleaning process).
– A state machine (example states are the operation modes of a device like running, ready, stop, as shown in Figure 14 and Table 3).
– A sequence diagram in case a functional element interacts with one or more other functional elements.

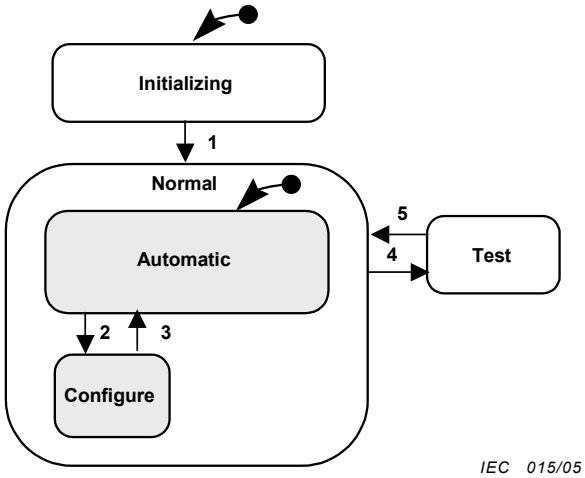


Figure 14 – Device behaviour as state chart diagram (example)

Table 3 – Device behaviour as state transition table (example)

STATE NAME	STATE DESCRIPTION
Initializing	Initial state of the device upon power-up. The device is not yet available for normal operation
Normal	The device is available for automatic operation
Automatic	"Presence" and "Alarm" parameter values are available to the network
Configure	When in this state, the device can be commanded by the "Operate mode" parameter to alter its operation accordingly (light or dark signal indicates presence) The device will not perform its normal sensing operations in this state – the "Presence" and "Alarm" parameter values should not be read by the network
Test	The device does not perform its normal sensing operations. The "Presence" and "Alarm" parameter values are set to one (1)

TRAN-SITION	SOURCE STATE	TARGET STATE	EVENT
01	Initializing	Normal	Device initialized and ready for normal operation
02	Automatic	Configure	“Device mode” parameter is commanded to change from zero (0) to one (1), or service “Set configure mode” is invoked
03	Configure	Automatic	“Device mode” parameter is commanded to change from one (1) to zero (0)), or service “Set automatic mode” is invoked
04	Normal	Test	“Test” parameter is commanded to change from zero (0) to one (1)), or service “Enter test mode” is invoked
05	Test	Normal	“Test” parameter is commanded to change from one (1) to zero (0)), or service “Exit test mode” is invoked

The result of this step is the behaviour description of the device and/or *functional elements* which are necessary to properly understand the device and interact with it. The results are recorded in the device behaviour section of the profile template.

NOTE If a device is user-programmable, its behaviour cannot be completely described in the profile. However, profile writers may agree on general common functions like Start, Stop and Reset.

The third to the fifth step will be carried out iteratively because they are tightly connected.

6.7 Sixth step (optional): Extensions of existing profiles

This optional step is necessary if a defined set of profiles or devices is derived out of a root or manufacturer device profile. The derived profile may extend these profiles by

– adding parameters, behaviour, *functional elements* items;

– requiring support of optional items of root or manufacturer device profiles.

Steps 1 to 5 may have to be reconsidered. This process leads to a set of related device profiles.

7 Profile templates

7.1 General

As explained in the introduction of this guideline and in Figure 1, profile writers provide a common representation of the networked industrial devices. This guideline recommends a profile template for documenting that representation. This profile template serves as a form that, when filled in by profile writers, becomes a human readable device profile. A filled-in profile template is the result of the profile development procedure described in Clause 6. The filled-in template may be exchanged using a profile exchange language such as XML (see 6.1).

7.2 Profile template structure

7.2.1 Overview

The profile template is organized in sections. The following sections are recommended.

– The header section contains the results of the first profile development step “scope, device classification”, together with revision information.

– The parameter list section contains the results of the third profile development step. There are three subsections: parameters with their characteristics, parameter groups and parameter assemblies.

– The device functional structure section contains the result of the second, fourth and fifth profile development steps. There are two subsections: the functional structure based on the functional elements (defined in the fourth step) and the device behaviour (defined in the fifth step).

These profile template sections are detailed in 7.2.2 to 7.2.4. It is the responsibility of the profile writer group to define the details of the profile template sections. An example of such a profile template is provided in 7.2.5, showing all the profile sections above.

7.2.2 Device profile header

The contents of the header are the result of the first profile development step.

The header section provides device profile identification. The header makes it clear to the reader of the profile that he has the correct device profile.

If a profile is defined for a class of devices (for example, a profile defined by an IEC product committee, called “root profile” in Figure 1), the contents of the header provides an unambiguous identification of this profile, including the identifier assigned to this profile by the profile writer organization (device profile ID), profile revision (device profile version) and profile release date (device profile release date). Fields for additional device information may be provided.

If a profile is defined for a specific device (for example, a profile defined by a manufacturer, called “manufacturer’s profile” in Figure 1), the contents of the header provide additionally an unambiguous identification of this specific device. This identification information includes for example the name of the device, the catalogue number, the manufacturer, and the version. Fields for additional device information may be provided.

Annex K provides references for future work on harmonization of common profile and device identification information.

7.2.3 Parameter list

7.2.3.1 Parameters

The contents of the parameter list are the result of the third profile development step.

The “parameters” section of the template provides a list of all the device parameters that are accessible in the device through the network, together with their relevant characteristics.

A separate named field should be provided for each specified characteristic.

An example of parameter characteristics is included in the example template of 7.2.5.

7.2.3.2 Complex data types

Complex data types need to be defined if parameters are of data type array or structure. Some characteristics may be defined at the data type or parameter level. Examples of parameters using complex data types are included in the example template of 7.2.5.

7.2.3.3 Parameter groups

The parameter group definitions are the result of the third profile development step.

7.2.3.4 Parameter assemblies

The parameter assembly definitions are the result of the third profile development step.

7.2.4 Device functional structure

7.2.4.1 Functional elements

The device functional structure definitions are the result of the second and fourth profile development step.

The template contains the description of the functional structure using a *functional element* list and an optional functional structure diagram showing the relationships between the *functional elements*.

Simple devices may only consist of a single *functional element*.

Complex devices may consist of a collection of *functional elements*.

7.2.4.2 Device behaviour

7.2.4.2.1 State machines

The device functional structure definitions are the result of the second and fifth profile development step.

It is recommended that the behaviour of the device or *functional element* be described. Good practice is to do this in terms of a state chart; an example is shown in Figure 14. In this case, the profile template in 7.2.5 contains a state machine section for specifying a state chart diagram and a state transition table. It is recommended that both diagram and table be specified because the diagram is suitable for a quick human overview and the table is suitable for a mapping to a machine-readable format. The descriptions of state machines and state chart diagrams should follow the UML specification.

7.2.4.2.2 Mathematical and procedural behaviour

The device behaviour definitions are the result of the fifth profile development step.

Behaviour of *functional elements* may be expressed using mathematical equations and procedural and consistency rules among parameters and conditions. IEC 61131 should preferably be used to describe this behaviour.

7.2.5 Template form

Table 4 shows an example of a representation format for the profile template.

NOTE This example is derived from the template specified in IEC 61915:2003.

Table 4 – Filled-in template of a device profile (example)

DEVICE PROFILE HEADER

NOTE Table K.1 provides common profile identification information.

Device profile ID	Device profile version	Device profile release date	Device profile description
Chd	V.53	2003-02-27	This is an example profile.

PARAMETER LIST [optional]

NOTE 1 A parameter may be assigned to multiple parameter groups.
NOTE 2 Annex B provides a list of possible parameter characteristics.
NOTE 3 Table K.2 provides common identification parameters stored in the device.

PARAMETERS

Parameter name	Data type	Access direction	Value range	Default value	Support	Parameter description
ChV	Bool	r/w	0 .. 1	1	Mandatory	This is an example parameter
OJ	Real	r	-100.0 .. 5400.0	0.0	Optional	
SPA	JB	r/w	0 ..1, -500 .. 100	0, 65535	Mandatory	
APB	FR	r	2 (-1.0 .. 2.0), -24.1957 .. 1806.1968	0.0, 0.0 ,1.5	Optional	

COMPLEX DATA TYPES

Data type name	Category	Number of elements or element names	Element data type	Additional information
JB	Struct	-	-	This is an example of a structured data type
-	-	Element_1	Boolean	NOTE Additional columns may be added to define characteristics at the data type level (for example, value range, default value)
-	-	Element_2	Integer	
FR	Array	3	Real	This is an example of an array data type

PARAMETER GROUPS

Group name	Number of elements	Group description
JPWG		This is an example group
Member names:		
HPO HW		

PARAMETER ASSEMBLIES

NOTE Parameter assemblies are defined for communication purposes only, for example, for cyclic data exchange. They are independent of the parameter groups of the parameter list.

Parameter assembly name	Access	Support	
BD	r/w	Mandatory	
Byte and Bit structure			

DEVICE FUNCTIONAL STRUCTURE [optional]

FUNCTIONAL ELEMENTS

FUNCTIONAL STRUCTURE DIAGRAM
Provide a function block diagram or object diagram (examples are shown in Figure 12 and 13).

FUNCTIONAL ELEMENT LIST

Functional element name	Support	Description
IL	Mandatory	This is an example functional element

DEVICE BEHAVIOUR [optional]
NOTE A behaviour description may be provided for the entire device and/or for functional elements.

STATECHART DIAGRAM
Provide state chart diagram here (an example is shown in Figure 14)

STATE TRANSITION TABLE (an example is shown in Table 3)

State name	State description

Transition	Source state	Target state	Event

MATHEMATICAL BEHAVIOUR, PROCEDURAL BEHAVIOUR or SEQUENCE DIAGRAM
(see 6.6)

8 Device models

8.1 Mapping of ISO device profile classes

This guideline is based on the device profile definition of ISO 15745-1. The corresponding device profile class diagram is shown in Figure 15.

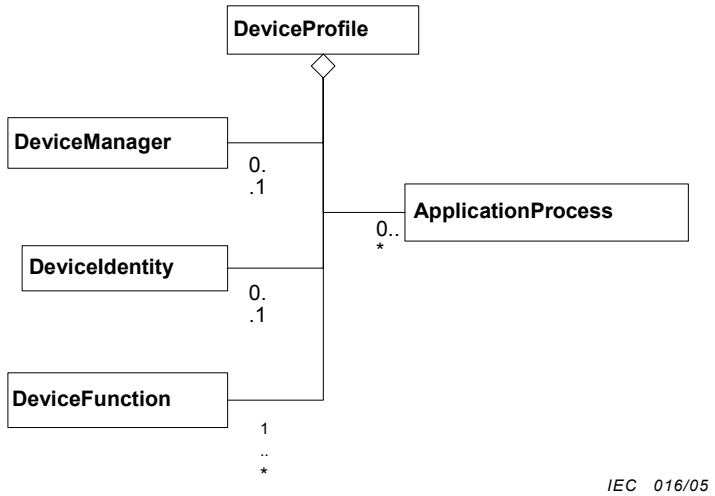


Figure 15 – ISO 15745-1 device profile class diagram

The following class definitions are given in ISO 15745-1:2003.

Device identity
: The device identity object contains attributes which uniquely identify the device. Examples of such attributes are the manufacturer's identification, part number, revision, location of storage of additional information, and indication of the number and type of additional objects within the device.

Device manager
: The device manager object represents the set of attributes (for example, revision of the device identity object) and services (for example, reset, configure/run mode, retrieval of device manager object attributes) used to configure and to monitor a device integrated into the application system.

Device function
: The device function object describes the intrinsic function of a device in terms of its technology (for example, mechanical limit switch, proximity sensor, ultrasonic sensor). The device function object differentiates the technology of the device from the application of the device. Examples of device function objects are analog current input in milliamps and discrete voltage output in volts.

Application process
: The application process object represents a set of attributes and services that correspond to the application requirements captured in the attributes and services of the associated process profile. The application process object, therefore, describes the behaviour of the device in terms of the application, independent of the device technology.

EXAMPLE An example of an application process object is a section of code within a device that detects, validates, and reports the presence (or absence) of a part, independent of the device technology being used. An infrared photoelectric sensor, a capacitive proximity switch, or a piezoelectric pressure device can meet the application requirement represented by the same application process object.

A simple device may contain one application process object. A complex device may contain one or more application process objects. In a distributed system, one application process object may span a number of devices.

The ISO 15745 device profile classes are seen from an abstract point of view. In principle, there are multiple mappings to it, as shown in Figure 16.

The parameter list model b) in Figure 16 represents the most simple mapping of the device model. It is described in Annex H. Simple devices may be fully described using parameter lists only. Parameters may be organized into parameter groups.

NOTE Parameter groups organize parameters of the parameter list for specific purposes, for example, identification, monitoring, configuration and HMI which can be mapped onto the relevant ISO 15745-1 device profile classes.

The parameter list model may be extended using either the function block model c) or the object model d) shown in Figure 16. The function block model is described in Annex I and the object model in Annex J.

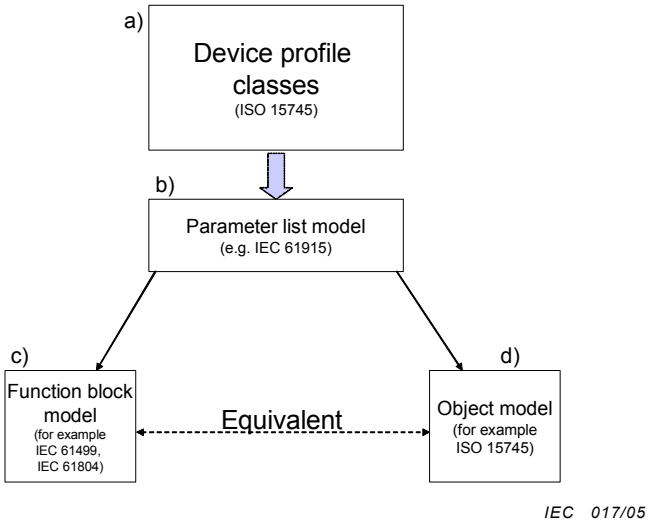


Figure 16 – Device profile models

8.2 Comparison of function block and object models

The function block model and the object models are equivalent and follow the object oriented concepts in terms of

– encapsulation of data and functions;
– instantiations of classes to objects (these are also called types and instances);
– class/object hierarchies.

The equivalence between the function block and the object models is shown in Table 5.

Table 5 – Equivalence between function block and object models

Device profile objects ISO 15745-1	Parameter list model elements	Function block model elements	Object model elements
Application process	Parameter group	Application function block View	Application specific object(s) Parameter object Assembly object
Device function		Transducer function block	Application specific object(s)
Device identity		Device management function block	Device identity and management
Device manager			
-	Parameter	Function block parameter	Attribute of a parameter object or an application specific object

Annex A
(informative)

Roles of the device in the life cycle

All phases of the device life cycle should be considered during the device profile development process, more specifically when defining the parameter list (see 6.4). Examples of such phases are listed below.

Commissioning, maintenance and servicing phases include the following tasks:

Parameterization
: Determines the hardware (for example, installed module) and software (for example, instantiated objects/blocks) structure of the device

Configuration
: Determines the initial/default and process-related values of the device parameters.

Working point adjustments/calibration
: Determines the relation between the real process (i.e. physical, biological, chemical) values and the measurement and actuation variables.

Structural information for design and engineering
: Provides information on the functional elements (for example, parameters, function blocks, objects) implemented in the device. This information may be accessed via a communication system by tools or other devices.

Commands and management functions
: Allow operators or the device itself to change device states and operation modes.

Firmware update

Device information
: Allows access to device identification and configuration information (for example, catalogue number, manufacturer's URL for additional information)

Diagnostics
: Provides device status and faults

Statistics/trends
: Provides device activity history

Operation phases include the following tasks:

Control application
: Provides measurement values, gets actuation set points and exchanges other information with other devices or a control program.

Visualization/HMI
: Provides measurement values, gets actuation set points and exchanges other information with operator for visualization and interactions.

Annex B
(informative)

Collection of parameter characteristics

Table B.1 provides a representative collection of parameter characteristics which may be used by the profile writer to specify the details of the parameter fields in the parameter list section of the profile template (see 6.4 and 7.2.3.1).

Table B.1 – Collection of parameter characteristics

Parameter characteristic	Conceptual data type	Description
Parameter ID	Implementation specific	Identifier of a particular parameter. The ID has to be unique in a device (see NOTE 1)
Parameter name	String	Name of a particular parameter (see NOTE 1)
Description	String	Textual description of parameter purpose (see NOTE 2)
Data type	Enumeration	IEC 61131-3 data types are preferred
Grouping level	Enumeration	*Simple* - basic data type *Array* - collection of data items of the same type *Structure* - collection of data items of different data types
Number of bytes	Numeric	Number of bytes of the parameter values
Parameter timing	Enumeration	There are two different time-related points of view: a) change of the parameter value in the context of the device and/or its process usage; and b) transmission of the parameter values via the communication system. a) The application system related change timing of parameters may have the following dynamic characteristic: • cyclic change (for example, periodic, polled, scanned for I/O parameters, control parameters used in a continuous manner) • acyclic change (for example, episodic, event driven, change of state/value for device management parameters used in a stochastic manner) b) The actual timing for transmission of the parameters values is technology-dependent (for example, a change of state parameter may be implemented using cyclic transmission). This mapping is the task of the communication profile
Access direction	Enumeration	Specification whether a parameter can be read and/or written: • read only • read/write • write only (for example, password parameter) The way to interact within a system is communication system dependent
Persistence	Enumeration	Specification whether or not the value is retained in device memory after the device's power was lost • volatile • non-volatile
Value range	Numeric	Range of supported values and/or list of supported values (see NOTE 2)
Coding	String	Enumeration of supported parameters values and their usage (see NOTE 2) EXAMPLE 0: OFF; 1: ON.

Default value	Corresponds to data type parameter characteristic	Defines the value that shall be contained in the parameter upon device delivery. If the profile does not specify a default value a manufacturer specific value may be implemented
Substitute value	Corresponds to data type parameter characteristic	Defines a specific value of the parameter that is provided to the application program in certain device operating states (for example, device fault)
Engineering unit	String or enumeration	Defines the engineering unit (if relevant) of the parameter. A list of engineering units is provided in Annex E
Offset	Numeric	The offset element specifies an offset which is added to an actual value to form a scaled value Engineering value = (parameter value + offset) * multiplier
Multiplier	Numeric	Scaling factor by which an actual value is multiplied to form a scaled value Engineering value = (parameter value + offset) * multiplier
Constraint	String	Constraints between parameters, for modelling certain dynamic device behaviour (for example, disable, enable, change another parameter depending on certain values of this parameter)
Support	Enumeration	Defines whether or not the parameter has to be implemented in the device: • optional: parameter implementation is possible but not required • mandatory: parameter implementation is required • conditional: parameter implementation is required if one or more other optional parameter(s) is (are) implemented. These parameters are specified using "Conditional support"
Conditional support	String	If Support = Conditional, the optional parameter(s) is (are) specified via this characteristic

NOTE 1 The usage of parameter ID and parameter name for parameter keying depends on the selected device model.

NOTE 2 Value range and coding contents may be combined within a single field or described in the description characteristic field.

NOTE 3 The data types used in this table are conceptual types, i.e. they describe the kind of physical signal, not the implementable data type. Mapping onto implementable data types (for example, IEC 61131-3) is required in actual profiles.

Annex C
(informative)

Compatibility level details

Table C.1 shows the relations between device features and parameter characteristics. The rows of the table correspond to the parameter characteristics listed in Annex B. The columns correspond to the device features (see 6.2.2).

Table C.1 – Relation between parameter characteristics and device features

Parameter characteristics	Device features						
	Communi-cation protocol	Communi-cation interface	Data access	Data types	Parameter semantics	Appli-cation func-tionality	Dynamic behaviour
Parameter ID			X				
Parameter name					X		
Description					X		
Data type				X			
Grouping level				X			
Number of bytes				X			
Access timing							X
Access direction			X				
Persistence						X	
Value range					X		
Coding					X		
Default value					X		
Substitute value					X		
Engineering unit						X	
Offset						X	
Multiplier						X	
Constraint						X	
Support						X	
Conditional support						X	

Annex D
(informative)

Data type

The data types of the device parameters need to be defined in the device profile. Table D.1 shows equivalence between data types used by various standards.

NOTE Only a subset of data types defined in these standards are listed in Table D.1.

It is recommended that device profile writers only use data types which are highlighted in grey.

Table D.1 – Data types

Bits	IEC 61131-3:2003 Subclause 2.3	IEC 61158-5:2003 Clause 5	OPC	"C"
1	BOOL	Boolean (see NOTE 1)	Boolean	-
8	BYTE	BitString8	Byte	Char
16	WORD	BitString16	unsignedShort	Unsigned short
32	DWORD	BitString32	unsignedInt	Unsigned long
64	LWORD	BitString64	unsignedLong	-
8	SINT	Integer8	Byte	Char
16	INT	Integer16	Short	Short
32	DINT	Integer32	Int	Long
64	LINT	Integer64	Long	-
8	USINT	Unsigned8	unsignedByte	~~Unsigned~~ char
16	UINT	Unsigned16	unsignedShort	Unsigned short
32	UDINT	Unsigned32	unsignedInt	Unsigned long
64	ULINT	Unsigned64	unsignedLong	-
32	REAL	Float32	Float	Float
64	LREAL	Float64	Double	Double
8 x n ASCII	STRING	VisibleString (ISO/IEC 10646)	Array of char	Array of char
8 x n	Array of BYTE	OctetString	-	Array of char
16	WSTRING [1]	UNICODE Char	UNICODE Char	Array of char
16 x n	WSTRING	UNICODE String	UNICODE String	Array of char
-	TIME (duration)	See NOTE 2	Duration	-
-	DATE	See NOTE 2	Date	-
-	TIME_OF_DAY	See NOTE 2	-	-
-	DATE_AND_TIME	See NOTE 2	DateTime	-
-	ARRAY	See NOTE 2	ArrayOfxx	[ ]
-	STRUCT	See NOTE 2	-	Struct
-	Derived data types	-	-	typedef

NOTE 1 IEC 61158 communication systems transmit a boolean via one octet. For performance reasons, profile writers may pack several device parameters of various data types into other data types.

NOTE 2 Implementation of this data type is technology dependent.

Annex E
(informative)

Engineering unit

Table E.1 lists example values for the engineering units parameter characteristic in Annex B.

NOTE 1 Engineering units are defined in the ISO/IEC 80000 series.

NOTE 2 Engineering units are also defined at http://www.physics.nist.gov/Pubs/SP811/cover.html.

Table E.1 – Engineering units (examples)

Quantity	Name	Symbol	Visible exchange format (see NOTE)
Length, displacement	meter	m	m
	millimeter	mm	mm
	kilometer	km	km
	micrometer	µm	um
Area	square meter	m²	m**2
	square millimeter	mm²	mm**2
	square kilometer	km²	km**2
Volume	cubic meter	m³	m**3
	liter	l	l
Time	second	s	s
	minute	min	min
	hour	h	h
	day	d	d
	millisecond	ms	ms
	microsecond	µs	us
Force	newton	N	N
	kilonewton	kN	kN
	meganewton	MN	MN
Pressure	pascal	Pa	Pa
	kilopascal	kPa	kPa
	millibar	mbar	mbar
	bar	bar	bar
Mass	kilogram	kg	kg
	gram	g	g
	milligram	mg	mg
	tonne	t	t
Energy	joule	J	J
	kilojoule	kJ	kJ
	megajoule	MJ	MJ
	watt hour	Wh	Wh
	kilowatt hour	kWh	kWh
	megawatt hour	MWh	MWh
Apparent power	volt ampere	VA	VA
	kilovolt ampere	kVA	kVA
	megavolt ampere	MVA	MVA
	millivolt ampere	mVA	mVA
Rotation speed	1 per second	s^{-1}	s**-1
	1 per minute	min^{-1}	min**-1
	1 per hour	h^{-1}	h**-1
Angle	radian	rad	rad
	second	“	“
	minute	‘	‘
Velocity	meter per second	m/s	m/s
	millimeter per second	mm/s	mm/s
	millimeter per minute	mm/min	mm/min
	meter per minute	m/min	m/min
	kilometer per hour	km/min	km/min
	millimeter per hour	mm/h	mm/h
	meter per hour	m/h	m/h
	kilometer per hour	km/h	km/h

Quantity	Name	Symbol	Visible exchange **format** (see NOTE)
Volume flow	cubic metre per second	m^3/s	m**3/s
	cubic metre per minute	m^3/min	m**3/min
	cubic metre per hour	m^3/h	m**3/h
	litre per second	l/s	l/s
	litre per minute	l/min	l/min
	litre per hour	l/h	l/h
Mass flow	kilogram per second	kg/s	kg/s
	gram per second	g/s	g/s
	tonne per second	t/s	t/s
	gram per minute	g/min	g/min
	kilogram per minute	kg/min	kg/min
	tonne per minute	t/min	t/min
	gram per hour	g/h	g/h
	kilogram per hour	kg/h	kg/h
	tonne per hour	t/h	t/h
Torque	newton meter	Nm	Nm
	kilonewton meter	kNm	kNm
	meganewton meter	MNm	MNm
Temperature	kelvin	K	K
	degree celsius	°C	C
	degree fahrenheit	°F	F
Temperature difference	kelvin	K	K
Entropy	joule per (kelvin times kilogram)	J/(K*kg)	J/(K*kg)
	kilojoule per (kelvin times kilogram)	kJ/(K*kg)	kJ/(K*kg)
	megajoule per (kelvin times kilogram)	MJ/(K*kg)	MJ/(K*kg)
Enthalpy	joule per kilogram	J/kg	J/kg
	kilojoule per kilogram	kJ/kg	kJ/kg
	megajoule per kilogram	MJ/kg	MJ/kg
Electrical voltage	volt	V	V
	kilovolt	kV	kV
	millivolt	mV	mV
	microvolt	μV	uV
Electrical current	ampere	A	A
	milliampere	mA	mA
	kiloampere	kA	kA
	microampere	μA	uA
Electrical resistance	ohm	Ω	O
	milliohm	mΩ	MO
	kiloohm	kΩ	KO
	megaohm	MΩ	MO
Relation	percentage	%	%
Relative humidity	percentage	%	%
Absolute humidity	gram per kilogram	g/kg	g/kg
Relative change	percentage	%	%
Frequency	hertz	Hz	Hz
	kilohertz	kHz	kHz
	megahertz	Mhz	Mhz
	gigahertz	GHz	GHz
Power	watt	W	W
	milliwatt	mW	mW
	kilowatt	kW	kW
	megawatt	MW	MW
Power (US)	horsepower	HP	HP
Acceleration	meter per second squared	m/s^2	m/s**2
Torque	meter per second cubed	m/s^3	m/s**3

NOTE The visible exchange format based on ISO/IEC 10646 (ASCII code) shown in this table is a recommendation for the encoding of the engineering unit if this information needs to be stored or exchanged via a communication network.

Annex F
(informative)

UML class diagram semantics

The class diagram is one of the UML specification methods. The UML elements, which are used in the class diagrams of this guideline (Figure 5 and Figure 11), are shown in Figure 1.

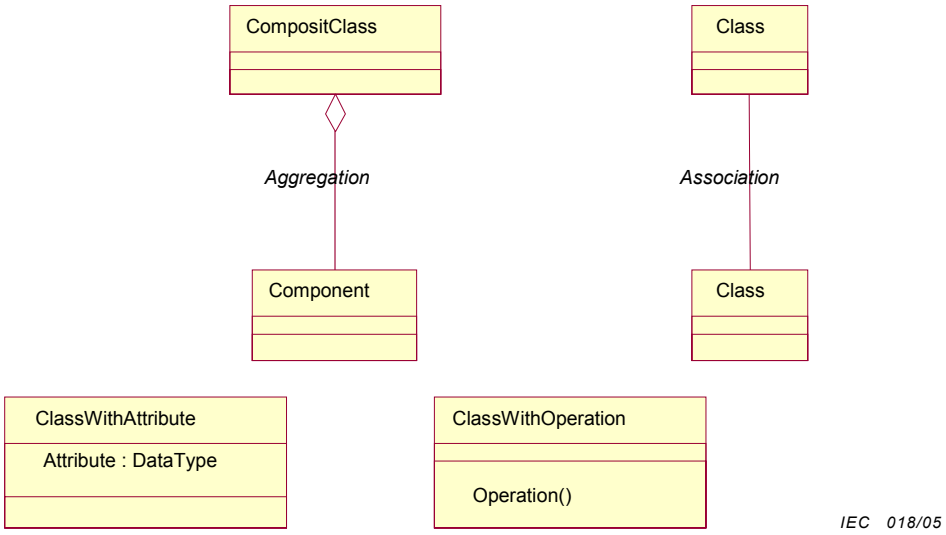


Figure F.1 – Description elements of UML class diagrams

The terms used in the figure are defined in OMG-UML V1.4 as follows.

Aggregation
: special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part.

Association
: semantic relationship between two or more classifiers that specifies connections among their instances.

Attribute
: feature within a classifier that describes a range of values that instances of the classifier may hold.

Class
: description of a set of *objects* that share the same *attributes*, *operations*, methods, relationships, and semantics.

Component
: modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. A component is typically specified by one or more classifiers (for example, implementation classes) that reside on it, and may be implemented by one or more artefacts (for example, binary, executable, or script files).

Operation
: service that can be requested from an object to effect behaviour. An operation has a signature, which may restrict the actual parameters that are possible.

Annex G
(informative)

Device classification examples

The following classification should be a support to locate these devices in the entire industrial automation market. The list is based on the experience and good practice of industrial application systems. Additionally, there are catalogues of devices available in the field of e-business systems. Table G.1 is proposed for use in the first profile development step (see 6.2.3).

NOTE Table G.1 is not intended to be complete; however, typical field devices of industrial automation are covered.

Table G.1 – Device classification (hierarchy) (examples)

Domain/group	Subdomain (family)	Principles (family members)
Power distribution		
	Switchboards	
	Circuit-breakers	
	Power monitoring	
	Distribution panel	
Motion control	Contactors	
	Protection starters	
	Soft starters	
	Drives	
	Axis control	
	Motor control centre	
	Motor monitoring	
	Positioners	
	Control valves	
Detection, measurement	E (electrical)	
(sensor discrete i/o or analog)	D (density)	
	F (flow)	
		Differential pressure
		Floating body
		Electromagnetic
		Ultrasonic
		Vortex counter
		Deplacement counter
		Turbine wheel counter
		Coriolis
		Thermal
	L (level)	
		Hydrostatic
		Deplacement
		Float
		Ultrasonic
		Microwave
		Laser/optical
		Radiometric
		Capacitance

	Q (quality)	
	P (pressure)	
		Pressure
		Differential pressure
	S (speed, rotary frequency)	
	R (radiation)	
	T (temperature)	
		Resistance, thermocouple
		Pyrometer
		Expansion
		Bimetallic strip
		Hot/cold conductor
	W (weight mass)	
	Distance, position, presence	
		Limit switches
		Inductive sensors
		Photoelectric sensors
		Capacitive sensors
		Ultrasonic sensors
		Pressure switches
Dialogue/operator interfaces		
	Push buttons	
	Joysticks	
	Keypads	
	Pilot lights	
	Stack lights	
	Displays	
	Combined buttons/lights	
	Operator stations	
Logic/universal I/O modules and controllers		
	General input	
	General output	
	Combined input/output	
	Relays	
	Timer	
	Scanners	
	Programmable controller	

Annex H
(informative)

Parameter list model

The parameter list model is a simple approach to define a device and consists only of device parameters. A parameter may be made of several subparameters that may not be accessible as a single parameter in the parameter list. These parameters are put together as a list where each list entry represents one parameter. The profile template may consist only of a header, a parameter list which can be optionally structured in parameter groups and parameter assembly. A graphic device model is not required.

NOTE A simple device may provide the ability to adjust the parameterization or configuration of a device by writing a single value to a specific parameter of the device. No special device behaviour process is provided or necessary.

Annex I
(informative)

Function block model

I.1 Background

The function block model can be useful for the user of the devices and also the device manufacturers as well as the device profile writers. They can apply the function block concept for the design of the device functionality or for designing the interaction between the application control program in the programmable controller and the devices.

Independent of the location of a function block (in a device or a programmable controller), its main purpose is the encapsulation of algorithms. The algorithms execute the processing of a set of defined input data and parameters and calculate the desired output data. A single function block or a combination (network) of function blocks builds the control application programs.

In the design process of a process control application program, the function blocks are often derived from the pipe and instrumentation diagram (P&ID) that is illustrated in Figure I.1, which is well-known in the process control industry. The information of the measurement and actuation points together with the designed control structure within the P&ID is transferred to the FB network. In the design process of factory automation application, function block networks are typically derived from the electrical design schemes (for example, ECAD).

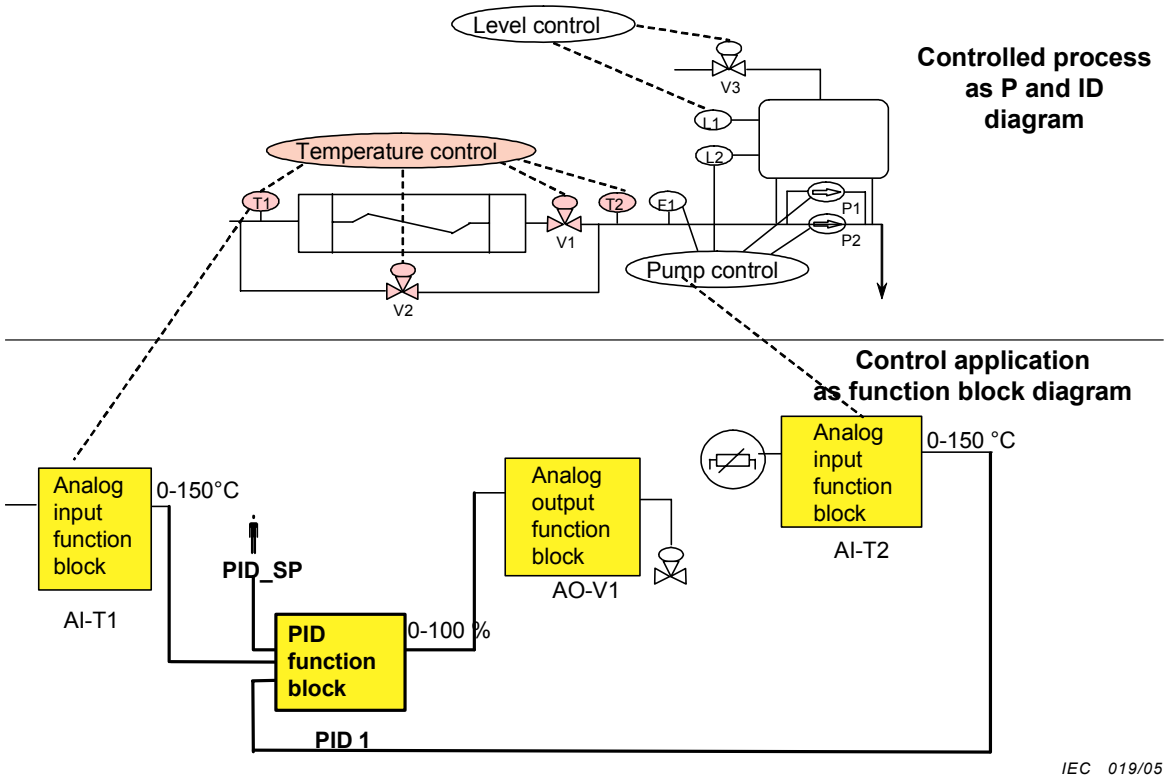


Figure I.1 – Function block diagram derived from the P&ID

Figure I.2 illustrates that the function blocks can reside in the field devices (FD), in the programmable controllers and also in the visualization tools.

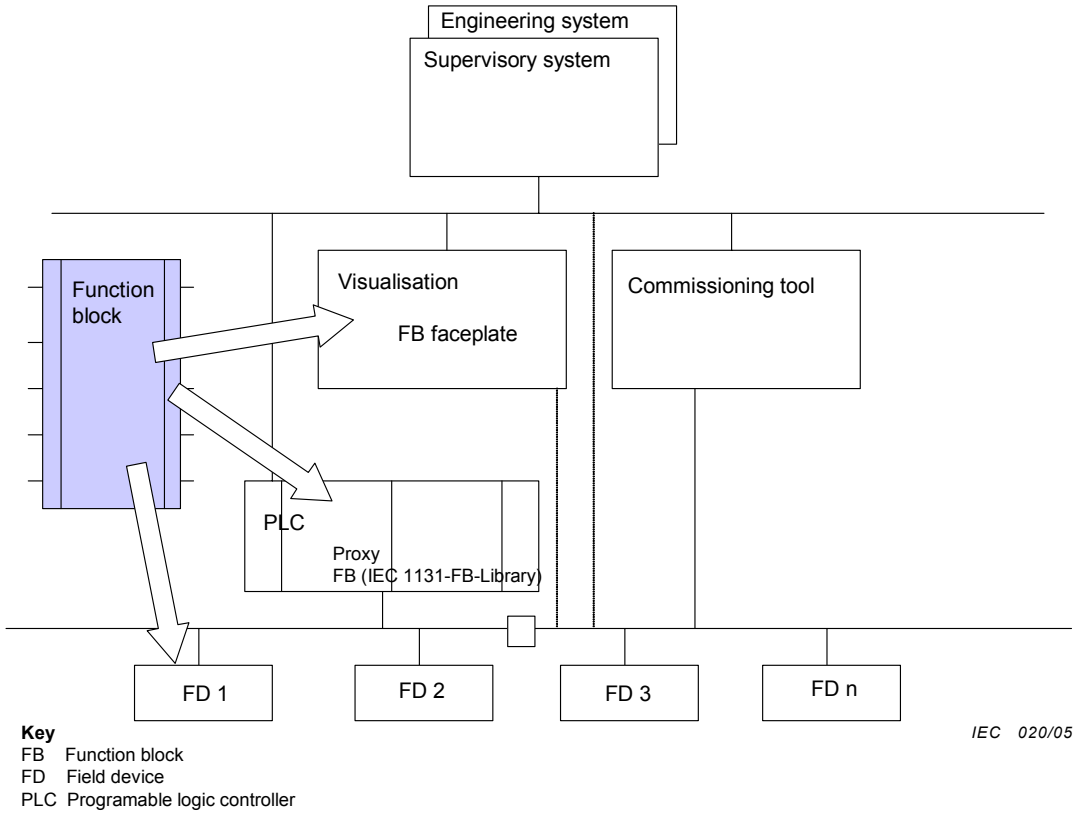


Figure I.2 – Function blocks implemented in different devices

I.2 Control system structure paradigms

There are three main paradigms dealing with control system structure using function blocks as shown in Figure I.3.

The classical system structure is the *central control* with a function block implementation typically written in an IEC 61131-3 language using central and/or remote I/O modules, for example, on a fieldbus. The typical modules are simple devices or devices with an interface like simple devices. To address the I/O data the application program uses the directly represented I/O variables, which are defined in IEC 61131-3 as, for example, I1.1 or Q2.3.

An increasing number of systems make use of the *decentral control* structure with intelligent field devices connected by a fieldbus. Here, the proxy function block concept applies specific function blocks in the “central” PLC application program, which represent the functionality of the devices and realize the implicit communication to the device over the fieldbus. In this case, the devices are addressed by the input and output variables of the proxy function blocks. The application program with the function blocks in the controller is programmed “centrally” scheduled according to IEC 61131-3 or IEC 61804.

A new paradigm is the *distributed control* structure with function blocks distributed over networks, which communicate directly as defined in IEC 61499. In this case, the application program is distributed in the devices and the controllers (if any). In contrast to the proxy concept of the decentral structure, here the function blocks need a system wide scheduling or an event-driven mechanism for their invocation.

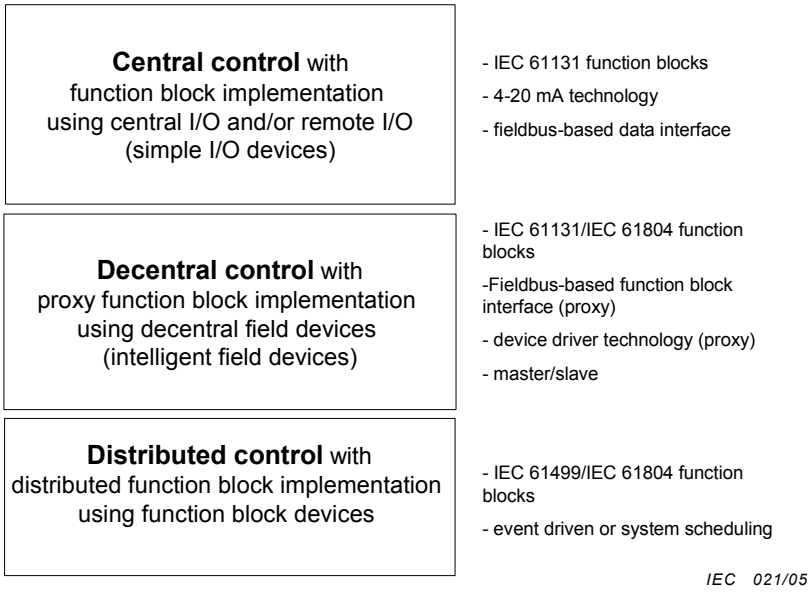


Figure I.3 – Function block application program in control system structure paradigms

I.3 Function block paradigms

The basic function block model, which is common for all IEC 6113-3 languages and also used in IEC 61499-1 and IEC 61804-2 supports the following two paradigms.

– It reflects the *component* paradigm of the reusable logic modules derived from the hardware design method with defined interface and encapsulated functionality including a persistent data memory.

– It also includes the *type/instance* paradigm of the object oriented IT programming method. That means a piece of a program (algorithms), which is supposed to be reusable is to be programmed as a function block type and stored in a library. A function block type consists of a declaration and a body as shown in Figure I.4. In the application program, this function block type can be invoked (called) as one or more function block instances with different parameter sets. In context with this device profile guideline, each instance can represent a device with local memory data.

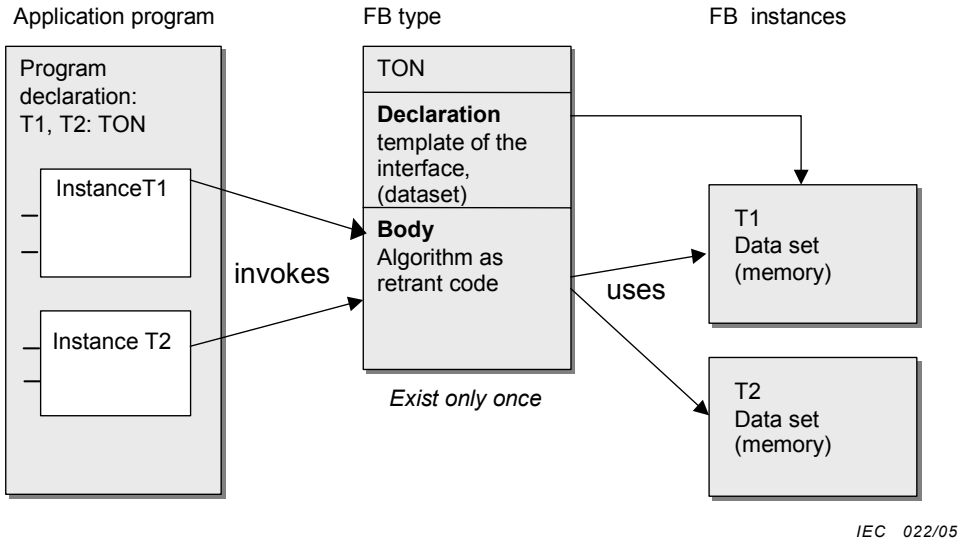


Figure I.4 – Function blocks of IEC 61131-3

In IEC 61131-3, mainly the principle of a central control with direct data access is considered even if the data is accessed over a fieldbus.

In smart devices, it is possible to describe the device functions in terms of FBs as shown in Figure I.5. This is the basic idea of IEC 61804. The main reason is that the application program design may integrate the decentralized field device functions in terms of FB in the entire application system.

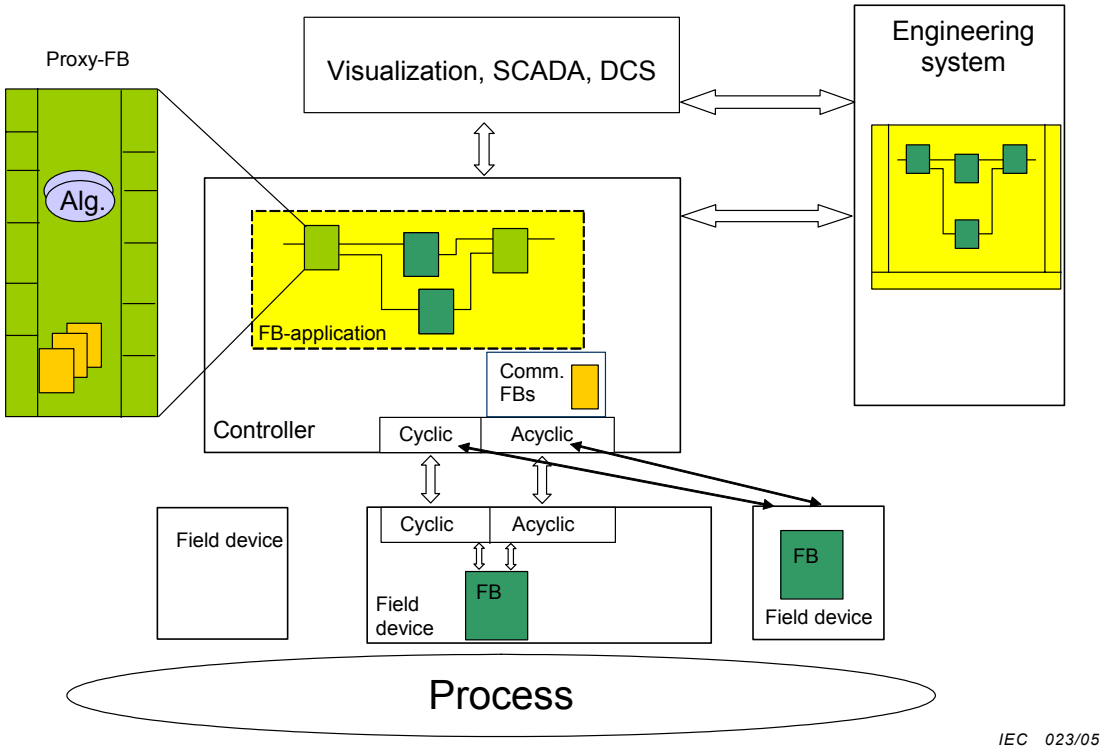


Figure I.5 –Function blocks in field devices and their integration in control programs

It is possible to design data flow between FBs in field devices without their integration in the controller program if additional scheduling and management functionality are implement in the devices.

I.4 Proxy function block

Figure I.6a) shows a motor control application program consisting of a function block and some combined basic blocks (AND, OR) that directly execute the process I/O. In the process I/O the motor terminals and some associated interlocking signal are mapped.

In Figure I.6b) the above-mentioned “combined basic blocks” are hidden in a predefined function block specific motor control that represents the device and hides the communication. For the user of this motor control, the function block is not visible whether there is central I/O or remote I/O connected. This usage of function block is called *proxy function blocks.*

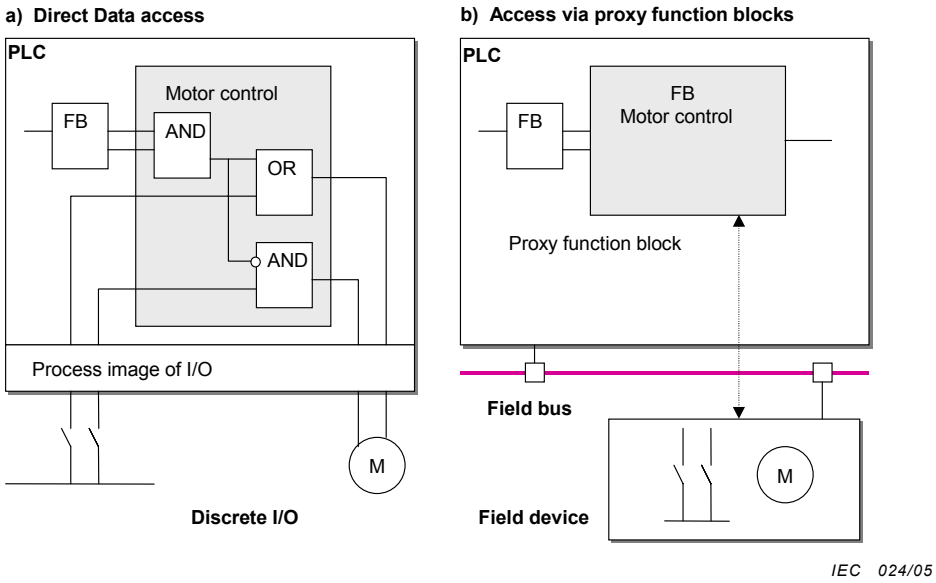


Figure I.6 – Concept of a central controller according to IEC 61131-3

There are various possibilities in a PLC program to access to the data in remote modules and devices.

A typical solution in the PLC is the "cyclic access" via the so-called *process image* to the remote inputs and outputs. In the application program of the PLC, these remote parameters are used like local I/O variables. The data exchange over the fieldbus is cyclically executed. The device parameters are transferred independently of the execution of the application program and mapped in the process image.

To achieve a standardized communication between the PLC and the device a set of *Communication Function Blocks* can be offered by the system manufactures. The reasons for this standardization are as follows.

– Standardized communication function blocks are the basis for portable PLC programs.

– Different process states usually need different sets of device parameters, which have to be transferred. Therefore, selective parameter access provides a better communication performance

Figure I.7 shows a PLC as a fieldbus master and a device as a fieldbus slave. They communicate by using generic communication function blocks, for example, read data record and write data record which can be used by the application programmer.

Figure I.7 also shows an instance of the *Proxy Function Block* representing the field device in the IEC 61131-3 application program in the PLC. This device-specific *Proxy FB* exhibits the functionality of the device by its input and output parameters. Inside the Proxy FB, basic *Communication Function Blocks* provide the reading and writing access to the field device data using the standard fieldbus protocols.

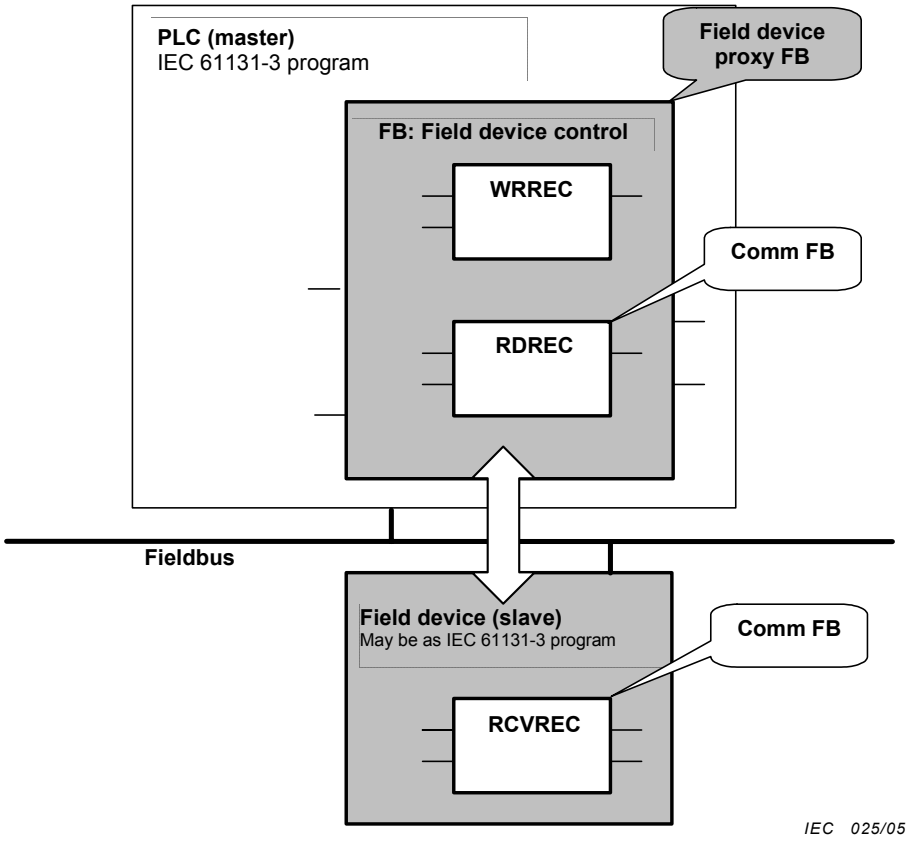


Figure I.7 – Proxy FB and communication FB

I.5 Distributed control structure according to IEC 61499

The distributed control system structure paradigm according to IEC 61499 is illustrated in Figure I.8.

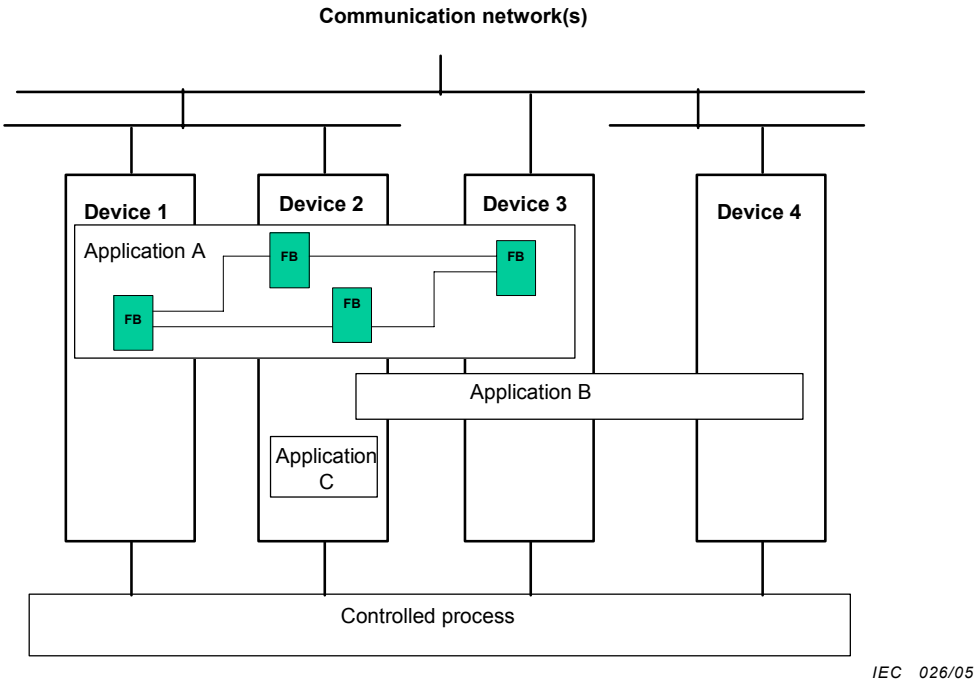


Figure I.8 – Function blocks application programs distributed in devices according to IEC 61499

According to IEC 61499, the entire functionality is located in field devices using distributed function blocks. Since, here, the scheduling of the function blocks by the common operation system, for example, of the PLC is missing, the invocation of the function blocks is made by the events coming from other function blocks. The data flow can be coupled with the event flow from function block to function block as shown in Figure I.9.

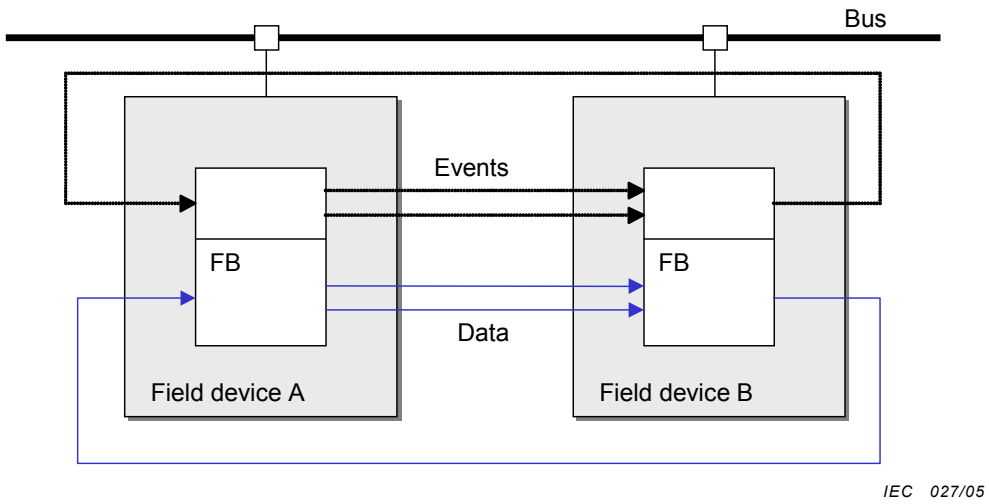
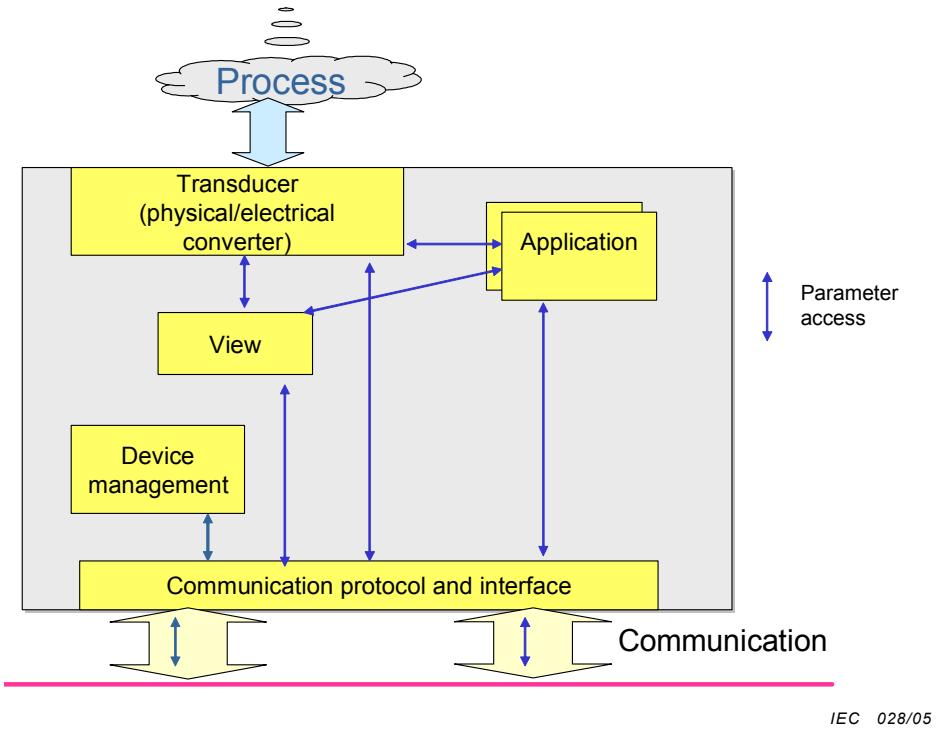


Figure I.9 – Application program distributed among a field device

I.6 Function block model in the process control devices

In the process control industry often the function block oriented device model is used as shown in Figure I.10. The figure shows the *functional elements* of a device modelled as function blocks (IEC 61804-2).



NOTE The communication profile part is not in the scope of this guideline.

Figure I.10 – Function block model in a field device

The following list gives explanations for the function block types of Figure I.10.

– Device Management function block

  Identification information (boiler plate), diagnosis, device controlling state machine

– Transducer function block

  Process attachment control

– Application function block

  Process control application system dependent signal processing, for example, manual/automatic operation modes, limit checking, scaling

– View

  Concatenation of function block parameters for communication optimized parameter access

Annex J
(informative)

Object model

J.1 Background

The use of object modelling within automation application systems has been triggered by increasing requirements from device manufacturers, system designers and users for

– easier integration of plant databases;

– improved interoperability;

– re-use of existing application system knowledge for design, operation and maintenance.

In an industrial environment where public and private networks are more and more often an integral part of the overall enterprise architecture, plant-floor devices may be required to interoperate with corporate information application programs, in addition to supporting their usual control interface. Besides, users are requiring that devices from different vendors interoperate in the same application system.

In parallel, control systems paradigms have gradually evolved.

– In the centralized control system structure, a single controller, connected to I/O modules or similar simple devices, handles the complete application program and data exchanges.
– In the decentralized control system structure, controllers may be additionally connected to intelligent devices with local processing capabilities but will still handle most data exchanges.
– In the distributed control system structure, both controllers and devices handle their part of the overall application and exchange data as needed, using peer-to-peer communication.

The object approach may be used for all three paradigms, directly in the case of distributed control, or via the definition of mapping/interface objects in the other cases.

Object modelling helps provide

– consistent characteristics for all devices;

– a well-defined externally visible behaviour;

– common interfaces and services;

– uniform description of exchanged information.

The definition of a library of re-usable objects and device profiles based on these objects reduces the design time and facilitates plug-and-play interoperability between complex devices from different manufacturers.

J.2 Object modelling paradigms

J.2.1 Object elements

An object provides an abstract representation of a particular component within a device.

Traditional application design, i.e. procedural programming, includes data structure, functions and processes: object modelling organizes related data and procedures into a single entity, using a specific terminology, and refers to it as an object (see Figure J.1).

NOTE The actual mapping of the abstract object model within a device is implementation dependent.

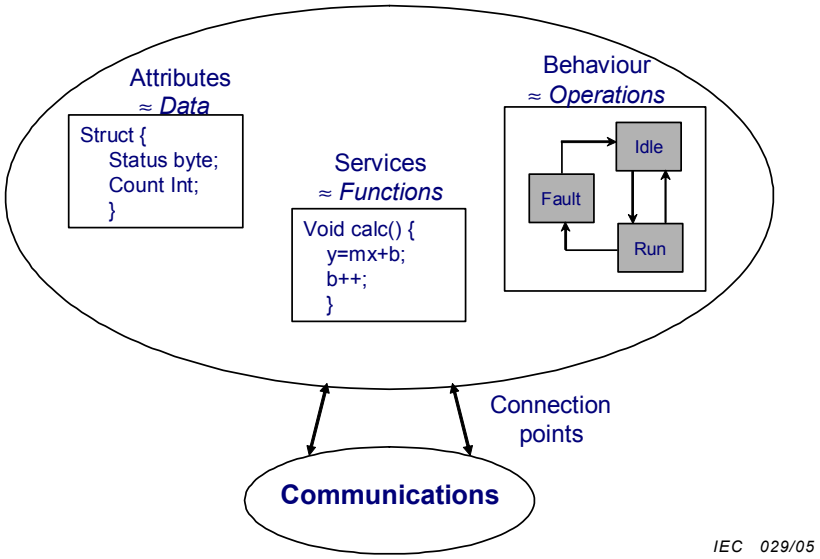


Figure J.1 – Object model elements versus procedural programming elements

An object is composed of the following elements:

– a set of related attributes (data);
– services (functions);
– defined behaviour (algorithms);
– supported connection points (for mapping onto communication).

Attributes are represented by values or variables and provide a description of externally visible characteristics or features of an object.

Attributes may be used to exchange information about the process signal flow, for example, to provide status information or to govern the operation of an object: the behaviour of an object may be affected by the value associated with an attribute.

A service is invoked to trigger an object to perform a task: it provides a function supported by the object. Object modelling supports both common (for example, get, set, reset) and object-specific services. Object-specific services are those that are defined for a particular object class to perform a required function not covered by a common service.

Services may be used to provide the following functions:

– access to various object attributes (get and set), for example, to provide input for process signal flow (set point) or to provide a parameter for scaling functions;
– trigger transition of the internal state machine governing object behaviour (for example, start, stop, resume);
– start specific operation/algorithm featured by the object;
– retrieve information about dynamic object structure or interface.

The behaviour of an object indicates how it responds to particular events. Actions result from different events of which the object is aware, such as receiving service requests, detecting internal faults, elapsing timers, or a change in an attribute value.

Connection points are object interfaces with the communication system.

J.2.2 Object classes

A class is a group of objects that defines a particular type of object and defines the characteristics shared by all the objects within a class.

Objects within a class are called object instances. An object instance is the actual representation of a particular object within a class. Each instance of a class has the same behaviour and the same set of attributes but has its own set of attribute values, which makes each instance in the class unique.

EXAMPLE The class "human" may be characterized by a common set of attributes (for example, gender, age, size, eye colour). Every person can be associated with an instance of this class, with a different set of values for the attributes, which allow individual persons to be distinguished.

Multiple object instances within a particular class can reside in a device. Object items can be uniquely identified within a device by using a class identifier, an instance identifier and an attribute identifier, as needed (see Figure J.2).

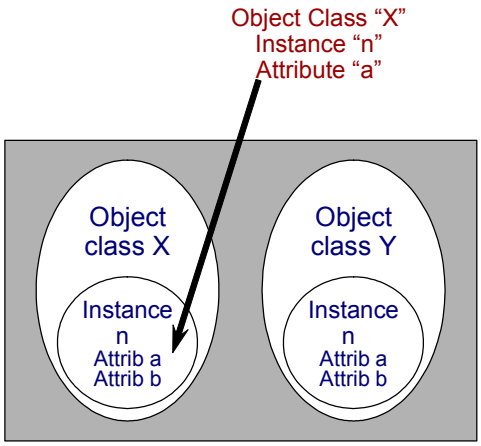


Figure J.2 –Object addressing

J.3 Device object model

J.3.1 Overview

A device may be modelled as a collection of objects. Figure J.3 shows the basic device object model, with the corresponding mapping onto the classes defined in ISO 15745-1.

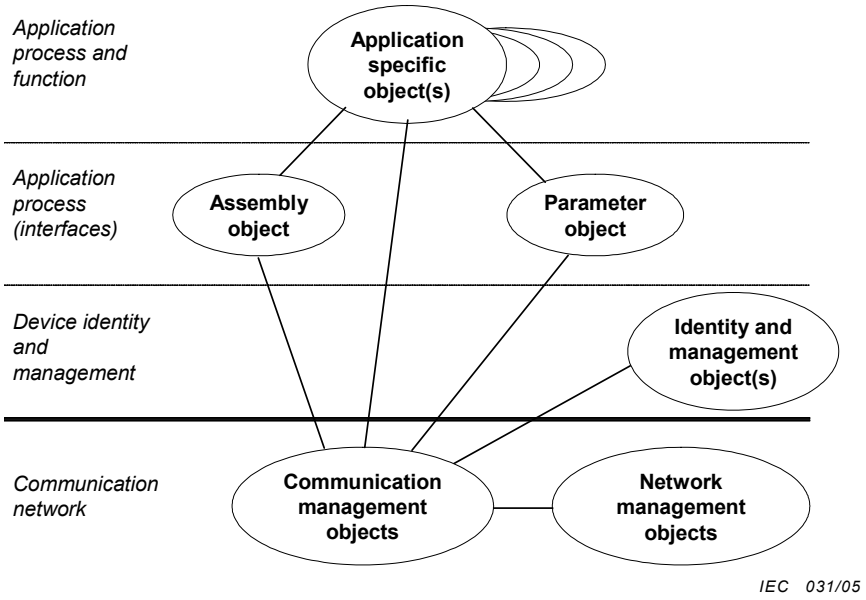


Figure J.3 – Device object model

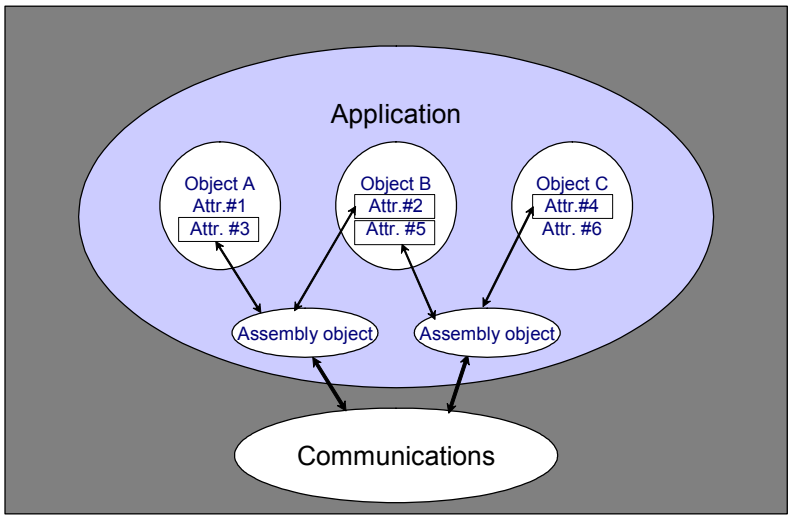
J.3.2 Object categories

The identity and management object(s) contain(s) the characteristics of the virtual device that are independent of the network being used and the process being controlled, such as manufacturer identification, part number, revision. It also provides device status information and supports services to control the device, i.e. device administration.

Application-specific objects are designed for a particular type of product or product range. They are mostly used to describe the behaviour of the device in terms of the application program (for example, signal processing). Some may also be used to describe the intrinsic function of a device in terms of its technology, if there is a need to exchange information related to this device technology (for example, specific calibration).

Assembly and parameter objects are general-purpose application objects. They are intended as interfaces between the application objects and the communication system.

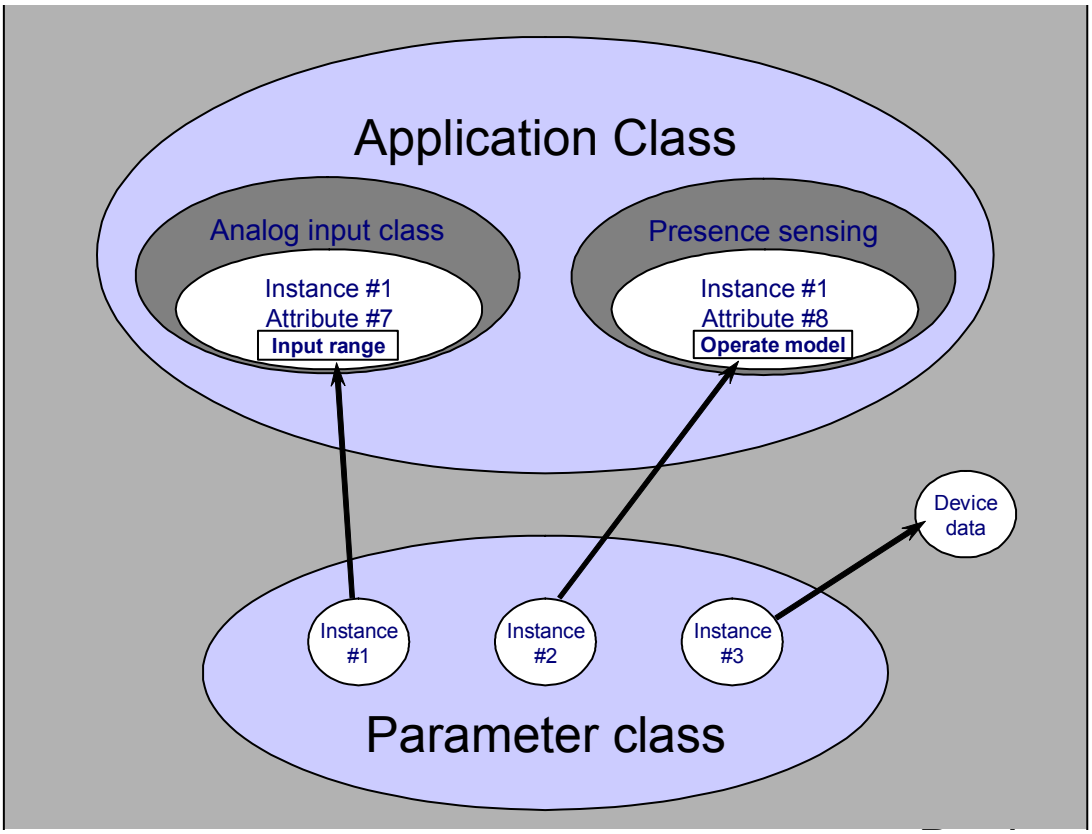
Assembly objects allow an optimized access to object attributes by gathering attributes from several objects into a single object: they are typically used for I/O exchanges (see Figure J.4).



IEC 032/05

Figure J.4 – Assembly object

Parameter objects are used to provide a common method to access device data (i.e. typically attributes of different application objects, but not necessarily): they contain parameter characteristics, i.e. information needed to interpret or display this data (see Figure J.5 and Annex B).



IEC 033/05

NOTE These two types of objects are typically used in centralized or decentralized control systems.

Figure J.5 – Parameter object

Communication/network management objects are used as interfaces between the device application objects and the network or for network management (for example, setting and monitoring of data link layers parameters). Figure J.6 shows an example of how communication management objects may be used for various types of data exchange (for example, I/O data or configuration and diagnostics).

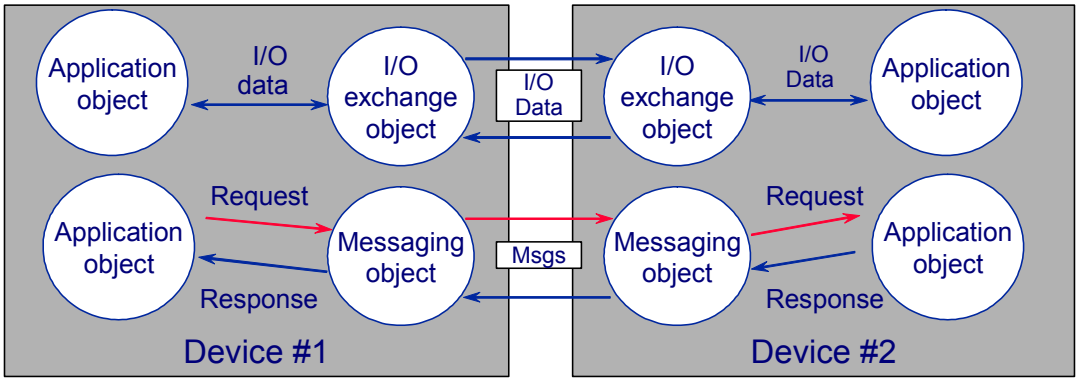


Figure J.6 – Communication management objects (example)

Annex K
(informative)

Common profile and device identification information

This annex is intended to be used as a reference for future work on harmonization of common profile and device identification information.

There are various locations for identification information:

– for profile header elements (ISO 15745-1, Table 1), see Table K.1;

– for parameters stored in the device, see Table K.2.

The following tables provide an overview of some existing and suggested implementations. Some elements may be available both within device profile description and in the device itself for retrieval through the network.

Table K.1 – Common profile header elements (ISO 15745-1, Table 1)

Attribute	Description
ProfileIdentification	Profile identification XML data type: string EXAMPLE: ABC-123-XX
ProfileRevision	Revision of the profile XML data type: string EXAMPLE: 2.34
ProfileName	Descriptive name of the profile XML data type: string EXAMPLE: AIP Material Handling
ProfileSource	Identification of the AIP developer XML data type: string EXAMPLE: Profiles'R'Us
ProfileClassID	Identification of the profile class. XML data type: ProfileClassID_DataType (based on "string") Valid profiles classes are: AIP Process InformationExchange Resource Device CommunicationNetwork Equipment Human Material EXAMPLE: AIP
ProfileDate	The release date of this revision of the profile in CCYY-MM-DD format This field is optional. XML data type: date EXAMPLE: 2002-10-25
AdditionalInformation	Location of diagrams/additional information for the profile This field is optional. XML data type: anyURI EXAMPLE: http://www.profilesrus.net
ISO15745Reference	The part of ISO 15745 (see ISO15745Part), together with its edition (see ISO15745Edition) and the profile technology (see ProfileTechnology) XML data type: ISO15745Reference_DataType Multiple references are allowed e.g. for a device with more than one communication interface
ISO15745Part	The part of ISO 15745 with which the profile complies XML data type: positiveInteger EXAMPLE: 1 (indicating ISO 15745-1)
ISO15745Edition	Edition of the referenced part of ISO 15745 XML data type: positiveInteger EXAMPLE: 1
ProfileTechnology	Name of the referenced technology within the previously specified part of ISO 15745 (see ISO15745Part field) XML data type: string The name associated with each technology is specified in the relevant part of ISO 15745 If no ISO 15745 technology is applicable, then the value “None” shall be used EXAMPLE: None

IASInterfaceType	The IAS interface type XML data type: IASInterfaceType_DataType (based on "string") This field is optional Valid IAS interface types are listed below and described in Annex B of ISO 15745-1 Any combination of the following is permitted: a) IAS interface types defined in ISO/IEC TR 14252: CSI Communication Services Interface HCI Human/Computer Interface ISI Information Services Interface API Application Program Interface b) IAS interface types defined in ISO 15745: CMI Configuration Management Interface ESI Engineering Support Interface FSI Facility Services Interface MTI Material Transport Interface SEI Safety And Environmental Interface USI Utility Services Interface c) User defined IAS interface types. EXAMPLE 1: ISI ESI EXAMPLE 2: CMI 37X6

Table K.2 – Common identification parameters stored in the device

Parameter name	Purpose	Examples in use	Comments
Manufacturer/Vendor identification	Source for more information about the device and where to order a new device or spare parts	Vendor ID Manufacturer_ID Manufacturer ID Vendor name	If devices are brand labelled, it could be be the vendor instead of the manufacturer, if he takes care of the maintenance
Product identification	Identify the device type or model	Product code/ Product name Order_ID Model number Product name/ Product ID	May be used for device ordering, device replacements, manual downloads, firmware updates
Serial number	Product tracking	Serial number Serial_ number	May be used for individual support, for example, in case of product recalls
Product revision	Information about product features	Major revision/ Minor revision Hardware_ revision/ Software_ revision Hardware revision/ Software revision	May be used to check hardware or software compatibility and to manage updates
Profile ID	Identifies the profile to which the device conforms	Device type Profile_ID Root device profile ID	Profile specification may be retrieved from a different source; therefore, additional information need not be stored in the device
Profile revision	Information about profile changes or additions	Root device profile version	
Application information	Provide a writeable placeholder where the system integrator can store application identification information	Tag_ Function/ Tag_Location User application name	Mainly used in process industry to store P&ID tags and location

NOTE 1 The examples in use are taken from IEC 61158, Types 2 and 3, IEC 61915 and modbus.org.

NOTE 2 It is recommended that device profile writers include at least the items which are highlighted in grey.

Bibliography

IEC 60050-351:1998, *International Electrotechnical Vocabulary (IEV) – Part 351: Automatic control*

IEC 61158-5:2003, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 5: Application layer service definition*

IEC 61784-1:2003, *Digital data communications for measurement and control – Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems*

IEC 61915:2003, *Low-voltage switchgear and controlgear – Principles for the development of device profiles for networked industrial devices*

IEC 61987 (all parts), *Industrial-process measurement and control – Data structures and elements in process equipment catalogues.*

ISO/IEC 10646:2003, *Information technology – Universal Multiple-Octet Coded Character Set (UCS).*

ISO/IEC 80000 (all parts), *Quantities and units*

ISO 15745-2:2003*, Industrial automation systems and integration – Open systems application integration framework – Part 2: Reference description for ISO 11898-based control systems*

ISO 15745-3:2003*, Industrial automation systems and integration – Open systems application integration framework – Part 3: Reference description for ISO 61158-based control systems*

ISO 15745-4:2003*, Industrial automation systems and integration – Open systems application integration framework – Part 4: Reference description for Ethernet-based control systems*

IEEE Std 754-1985 (R1990), *IEEE Standard for Binary Floating Point Arithmetic*

REC-xml-20001006, *Extensible Markup Language (XML) 1.0 Second Edition – W3C Recommendation 6 October 2000*

REC-xmlschema-1-20010502, *XML Schema – Part 1: Structures – W3C Recommendation 02 May 2001*

REC-xmlschema-2-20010502, *XML Schema – Part 2: Data types – W3C Recommendation 02 May 2001*

UML V1.5, *OMG – Unified Modeling Language Specification (Version 1.5, March 2003).*

———————

blank

BSI — British Standards Institution

BSI is the independent national body responsible for preparing British Standards. It presents the UK view on standards in Europe and at the international level. It is incorporated by Royal Charter.

Revisions

British Standards are updated by amendment or revision. Users of British Standards should make sure that they possess the latest amendments or editions.

It is the constant aim of BSI to improve the quality of our products and services. We would be grateful if anyone finding an inaccuracy or ambiguity while using this British Standard would inform the Secretary of the technical committee responsible, the identity of which can be found on the inside front cover. Tel: +44 (0)20 8996 9000. Fax: +44 (0)20 8996 7400.

BSI offers members an individual updating service called PLUS which ensures that subscribers automatically receive the latest editions of standards.

Buying standards

Orders for all BSI, international and foreign standards publications should be addressed to Customer Services. Tel: +44 (0)20 8996 9001. Fax: +44 (0)20 8996 7001. Email: orders@bsi-global.com. Standards are also available from the BSI website at http://www.bsi-global.com.

In response to orders for international standards, it is BSI policy to supply the BSI implementation of those that have been published as British Standards, unless otherwise requested.

Information on standards

BSI provides a wide range of information on national, European and international standards through its Library and its Technical Help to Exporters Service. Various BSI electronic information services are also available which give details on all its products and services. Contact the Information Centre. Tel: +44 (0)20 8996 7111. Fax: +44 (0)20 8996 7048. Email: info@bsi-global.com.

Subscribing members of BSI are kept up to date with standards developments and receive substantial discounts on the purchase price of standards. For details of these and other benefits contact Membership Administration. Tel: +44 (0)20 8996 7002. Fax: +44 (0)20 8996 7001. Email: membership@bsi-global.com.

Information regarding online access to British Standards via British Standards Online can be found at http://www.bsi-global.com/bsonline.

Further information about BSI is available on the BSI website at http://www.bsi-global.com.

Copyright

BSI
389 Chiswick High Road
London
W4 4AL