

DIN EN 50090-3-3

ICS 97.120

**Elektrische Systemtechnik für Heim und Gebäude (ESHG) –
Teil 3-3: Anwendungsaspekte –
ESHG-Interworking-Modell und übliche ESHG-Datenformate;
Englische Fassung EN 50090-3-3:2009**

Home and Building Electronic Systems (HBES) –
Part 3-3: Aspects of application –
HBES Interworking model and common HBES data types;
English version EN 50090-3-3:2009

Systèmes électroniques pour les foyers domestiques et les bâtiments (HBES) –
Partie 3-3 : Aspects de l'application –
Modèle d'inter-fonctionnement des HBES et types de données communes;
Version anglaise EN 50090-3-3:2009

Gesamtumfang 75 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Beginn der Gültigkeit

Die von CENELEC am 2008-12-01 angenommene EN 50090-3-3 gilt als DIN-Norm ab 2009-09-01.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN EN 50090-3-3:2008-06.

Die Europäische Norm EN 50090-3-3:2009 besteht nur aus einer englischen Sprachfassung. Es kann davon ausgegangen werden, dass die Anwender dieser Norm über hinreichende englische Sprachkenntnisse verfügen. Für die meisten der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel der englischen Benennungen bedienen.

Bei der Anwendung von DIN EN 50090-3-3 können Patente berührt werden, die der europäischen Normungsorganisation CENELEC mitgeteilt worden sind. Das Vorwort auf Seite 2 im EN-Text enthält Informationen zur Lizenzvergabe.

Die Europäische Norm EN 50090-3-3:2009 „Elektrische Systemtechnik für Heim und Gebäude (ESHG) – Teil 3-3: Anwendungsaspekte – ESHG-Interworking-Modell und übliche ESHG-Datenformate“ wurde vom TC 205 „Elektrische Systemtechnik für Heim und Gebäude (ESHG)“ des Europäischen Komitees für Elektrotechnische Normung (CENELEC) mit Hilfe des CENELEC-Kooperationspartners Konnex Association erarbeitet.

Die vorliegende Norm ist Teil der Reihe DIN EN 50090 „Elektrische Systemtechnik für Heim und Gebäude (ESHG)“ mit der VDE-Klassifizierung VDE 0829. Dieser Teil der Norm steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen. Damit entfällt die VDE-Klassifizierung.

Für diese Norm ist das nationale Arbeitsgremium K 716 „Elektrische Systemtechnik für Heim und Gebäude (ESHG)“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Nationaler Anhang NA (informativ)

Zusammenhang mit Europäischen und Internationalen Normen

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Eine Information über den Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ist in Tabelle NA.1 wiedergegeben.

Tabelle NA.1

Europäische Norm	Internationale Norm	Deutsche Norm	Klassifikation im VDE-Vorschriftenwerk
EN 50090-1 (geplant)	—	—	—
EN 50090-3-2:2004	—	DIN EN 50090-3-2:2004-09	—
EN 50090-4-1:2004	—	DIN EN 50090-4-1:2004-06	—
EN 50090-4-2:2004	—	DIN EN 50090-4-2:2004-07	—

Nationaler Anhang NB (informativ)

Literaturhinweise

DIN EN 50090-3-2:2004-09, *Elektrische Systemtechnik für Heim und Gebäude (ESHG) – Teil 3-2: Anwendungsaspekte – Anwendungsprozess ESHG Klasse 1; Deutsche Fassung EN 50090-3-2:2004, Text Englisch*

DIN EN 50090-4-1:2004-06, *Elektrische Systemtechnik für Heim und Gebäude (ESHG) – Teil 4-1: Medienunabhängige Schicht – Anwendungsschicht für ESHG Klasse 1; Deutsche Fassung EN 50090-4-1:2004, Text Englisch*

DIN EN 50090-4-2:2004-07, *Elektrische Systemtechnik für Heim und Gebäude (ESHG) – Teil 4-2: Medienunabhängige Schicht – Transportschicht, Vermittlungsschicht und allgemeine Teile der Sicherungsschicht für ESHG Klasse 1; Deutsche Fassung EN 50090-4-2:2004, Text Englisch*

**Home and Building Electronic Systems (HBES) —
Part 3-3: Aspects of application —
HBES Interworking model and common HBES data types**

Systèmes électroniques pour les foyers
domestiques et les bâtiments (HBES) —
Partie 3-3: Aspects de l'application —
Modèle d'inter-fonctionnement des HBES
et types de données communes

Elektrische Systemtechnik
für Heim und Gebäude (ESHG) —
Teil 3-3: Anwendungsaspekte —
ESHG-Interworking-Modell und übliche
ESHG-Datenformate

This European Standard was approved by CENELEC on 2008-12-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in two official versions (English, French). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: Avenue Marnix 17, B - 1000 Brussels

Foreword

This European Standard was prepared by the Technical Committee CENELEC TC 205, Home and Building Electronic Systems (HBES), joined by the co-operating partner KNX Association.

The text of the draft was submitted to the Unique Acceptance Procedure and was approved by CENELEC as EN 50090-3-3 on 2008-12-01.

CENELEC takes no position concerning the evidence, validity and scope of patent rights.

KNX Association as Cooperating Partner to CENELEC confirms that to the extent that the standard contains patents and like rights, the KNX Association's members are willing to negotiate licenses thereof with applicants throughout the world on fair, reasonable and non-discriminatory terms and conditions.

KNX Association
De Kleetlaan 5, bus 11
B - 1831 Diegem

Tel.: + 32 2 775 85 90
Fax.: + 32 2 675 50 28
e-mail: info@knx.org
www.knx.org

Attention is drawn to the possibility that some of the elements of this standard may be the subject of patent rights other than those identified above. CENELEC shall not be held responsible for identifying any or all such patent rights.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2009-12-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2011-12-01

EN 50090-3-3 is part of the EN 50090 series of European Standards, which will comprise the following parts:

- Part 1: Standardization structure
 - Part 2: System overview
 - Part 3: Aspects of application
 - Part 4: Media independent layers
 - Part 5: Media and media dependent layers
 - Part 6: Interfaces
 - Part 7: System management
 - Part 8: Conformity assessment of products
 - Part 9: Installation requirements
-

Contents

	Page
Introduction	5
1 Scope	6
2 Normative references	6
3 Terms, definitions and abbreviations	7
3.1 Terms and definitions.....	7
3.2 Abbreviations	7
4 HBES Interworking model	7
4.1 Principles of HBES Interworking	11
4.2 Busload	12
4.3 Datapoint Type error handling	13
4.4 Interpretability of data and data integrity	15
5 General Functional Block Design and Implementation Rules	15
5.1 Introduction	15
5.2 Describe the Application Domain.....	15
5.3 Describe the Application	15
5.4 Describe the Functional Block	19
5.5 Describing the Datapoint Types.....	29
Annex A (informative) Common HBES data types	38
Figures	
Figure 1 – The HBES Interworking Model An Application Domain can contain one or more Applications	7
Figure 2 – The HBES Interworking Model An Application Model may contain one or more Functional Blocks.....	8
Figure 3 – Standard representation for Functional Blocks	8
Figure 4 – Datapoints indicated in Functional Blocks.....	9
Figure 5 – Functional Blocks grouped in devices and linked	9
Figure 6 – Functional Block with 5 Datapoints	10
Figure 7 – The information contained in a Datapoint Type definition.....	10
Figure 8 – Example of an Interworking specification.....	11
Figure 9 – Functional Block diagram (Example)	20
Figure 10 – Table listing separate datapoints of a functional block	21
Figure 11 – Specification form for Inputs and Outputs	22
Figure 12 – Specification form for Parameters and Diagnostic Data	29
Figure 13 – Example of multi-state datapoint.....	32
Figure 14 – Datapoint Type specification form.....	34
Figure A.1 – Structure of Datapoint Types	38
Figure A.2 – December 12, 2006 encoded according DPT_Date in an A_GroupValue_Write-frame (example on TP1).....	39

Tables

Table 1 – Use of heart-beat.....	12
Table 2 – Authorisation level names	27
Table 3 – Datatypes notation styles	35
Table A.1 – Compatibility rules	67

Introduction

Interworking between devices signifies that these products send and receive datagrams and are able to properly understand and react on them. This ability is provided without additional equipment (like translators or gateways).

NOTE Media couplers are needed if different media are used in an installation.

The market requires Interworking for a multi-vendor approach, this is, products from different manufacturers can interwork in a certain application segment or domain as well as across different applications (cross discipline Interworking).

Such an Interworking is only possible if a set of requirements is complied with as defined in an Interworking model. For this, Functional Blocks need to be defined, which in turn specify Datapoints and the communication mechanisms to be used. Such a set of requirements is referred to as "Application Interworking Specifications" (AIS).

AIS allow Interworking independent of the implementation by a manufacturer. Besides the advantages for the user (multi-vendor offer) Interworking also allows a broad OEM market and easy market access for niche-products providers. Furthermore Interworking allows the establishment of a common market infrastructure (i.e. common configuration tool, training, etc.)

1 Scope

This European Standard gives general guidelines and recommendations to ensure interworking between HBES devices made by different manufacturers. It also contains design guidelines for the design of Functional Blocks and new datapoint types, the building blocks of HBES interworking.

In this way, the standard can be used as a basis to design application specifications relative to an Application Domain. If designed and supported by a large group of manufacturers, such application specifications will ensure to end customers a high degree of interoperability between products based on the HBES Communication System of different manufacturers.

This European Standard is used as a product family standard. It is not intended to be used as a stand-alone standard.

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.

EN 50090-1 ¹⁾	Home and Building Electronic Systems (HBES) – Part 1: Standardization structure
EN 50090-3-2:2004	Home and Building Electronic Systems (HBES) – Part 3-2: Aspects of application – User process for HBES Class 1
EN 50090-4-1:2004	Home and Building Electronic Systems (HBES) – Part 4-1: Media independent layers – Application Layer for HBES Class 1
EN 50090-4-2: 2004	Home and Building Electronic Systems (HBES) – Part 4-2: Media independent layers – Transport layer, network layer and general parts of data link layer for HBES Class 1

¹⁾ Under consideration.

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the definitions given in EN 50090-1 apply.

3.2 Abbreviations

AIS	Application Interworking Specifications
COV	Change of Value
DMA	Direct Memory Access
DP	Data Point
DPT	Datapoint type
DPT ID	Datapoint type identifier
FB	Functional Block
GO	Group Object
IO	Interface Object
lsb	least significant bit
M	Mandatory
msb	most significant bit
MSB	Most Significant Byte
NA	Not Applicable
O	Optional
PDT	Property Data Type
PID	Property Identifier
OEM	Original Equipment Manufacturer

4 HBES Interworking model

HBES Interworking can be applied in many application domains. Each Application Domain can encompass one or more Applications.

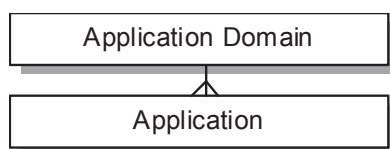


Figure 1 - The HBES Interworking Model
An Application Domain can contain one or more Applications

The Application Modelling or the process of analysing, deciding on the solution and specifying the model for such an Application needs to be agreed upon amongst manufacturers in Application Specification Groups.

Applications shall not be defined in terms of products, but analysed and split up into *Functional Blocks*, which communicate with one another. The term *Distributed Application* indicates this approach: the total functionality of an Application is spread over a number of Functional Blocks implemented in various devices in the network.

A Functional Block transports its data over the bus via one or more Datapoints (these are Inputs, Outputs, Parameters and Diagnostic Data).

A Functional Block thus describes the standard specification of the chosen solution for one given task of an application.

These Datapoints and their described functionality are implemented by the product developer.

The following picture shows the Interworking model as defined so far.

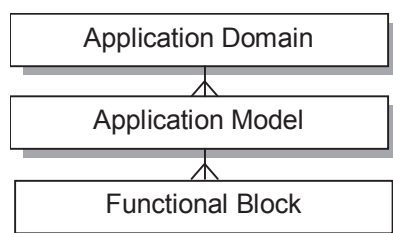


Figure 2 - The HBES Interworking Model
An Application Model may contain one or more Functional Blocks

The Functional Blocks shall be described as objects; this is a set of Datapoints and a well-defined behaviour.

The standard graphical representation for a Functional Block shall be the following:

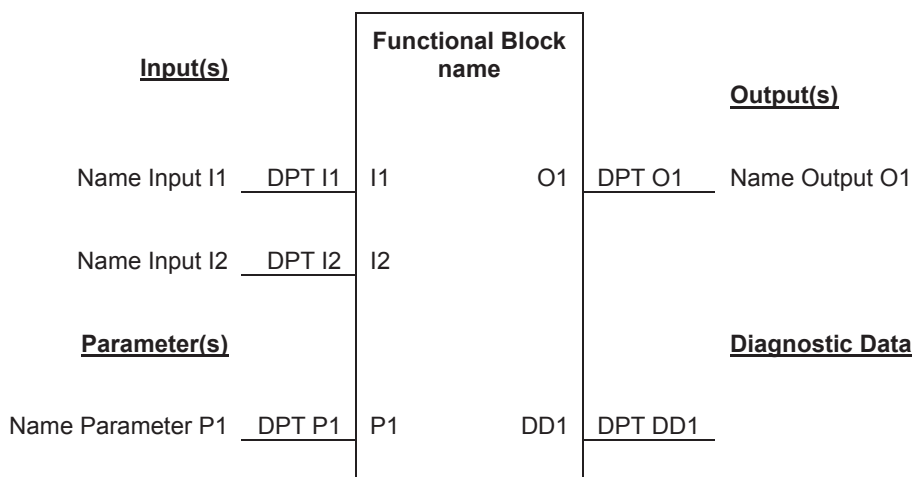


Figure 3 - Standard representation for Functional Blocks

The Datapoints that are typically Inputs shall be put on the left, the Outputs on the right, the Parameters on the left below the Inputs and the Diagnostic Data on the right below the Outputs. For each Datapoint, a name shall be given, an indication of its Datapoint Type and an abbreviation.

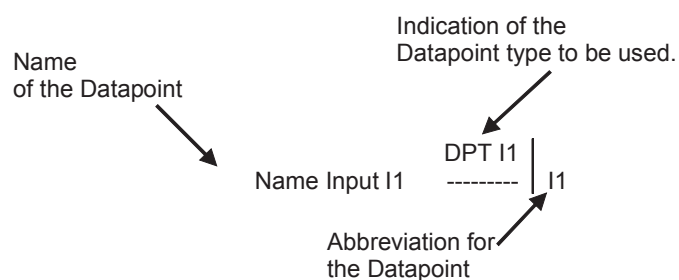


Figure 4 - Datapoints indicated in Functional Blocks

A manufacturer may group one or more of these Functional Blocks, of the same or of different Applications, to build a device.

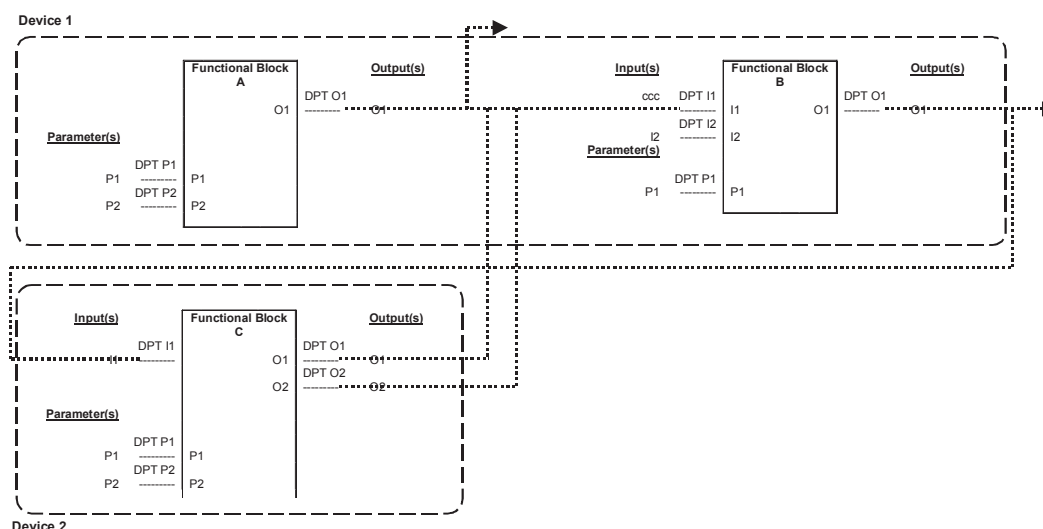


Figure 5 - Functional Blocks grouped in devices and linked

For every Functional Block its behaviour shall be specified. This fixes its handling of its Datapoints and physical inputs and outputs (e.g. a state machine of a dimming actuator).

A Datapoint is any interface over which data in the Functional Block can be set or received and/or transmitted (for its run-time operation). Every Functional Block may have one or more such Datapoints.

From the **communication** point of view, 4 classes of Datapoints can be distinguished (for more information see 5.3.3):

NOTE These classes of Datapoints are defined in EN 50090-4-2, EN 50090-4-1 and EN 50090-3-2:

- Group Object Datapoint
- Interface Object Property Datapoint
- Polling Value Datapoint
- Memory Mapped Datapoint

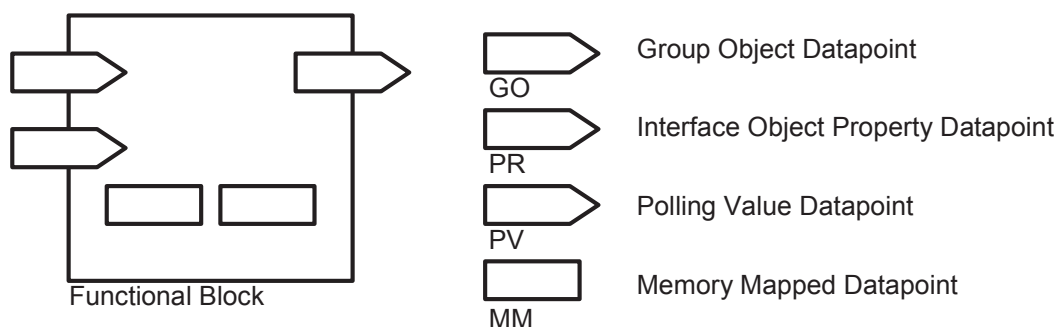


Figure 6 - Functional Block with 5 Datapoints

From the **access** point of view another differentiation of Datapoints is possible:

- Input: a value that is received and processed by the Functional Block.
- Output: the value resulting from the process of the Functional Block and to be provided to at least one other Functional Block.
- Parameter: a value that controls the process of handling the Input(s) and generating the values of the Output(s). This access type is typically non-volatile or saved at reset. It is usually set by management functions.
- Diagnostic Data: a value that represents the local or internal status information of a Functional Block. It is not used for runtime communication to Inputs or Outputs of other FBs but serves for visualisation on a central unit or operating station and, during installation, service and maintenance.

The data exchanged through Datapoints shall be standard in format, encoding, range and unit. They are defined in *Datapoint Types* (DPT).

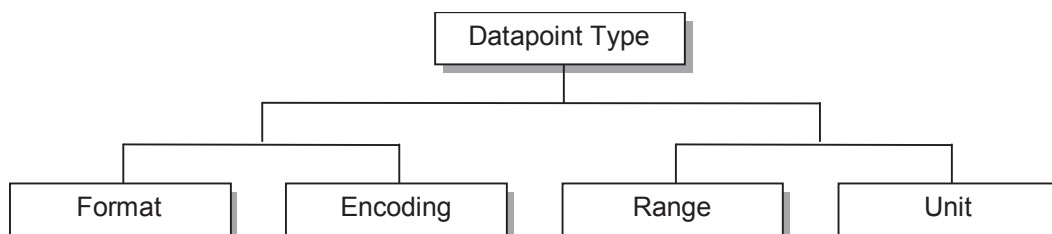


Figure 7 - The information contained in a Datapoint Type definition

When a given Datapoint Type is used to represent the value of a Datapoint in a Functional Block, it additionally obtains a specific semantic meaning.

EXAMPLE A Datapoint Type "Temperature" obtains the semantic value "Input Temperature Setpoint Value" when it is used in the Functional Block "Boiler Control".

The definition of a Datapoint Type shall consist of the following elements:

- *Format*: describes the sequence and length of the fields, each consisting of one or more bits, of which the Datapoint Type is built up.
- *Encoding*: describes how the data, that shall be transported using this Datapoint Type, is coded using the given format, possibly for each field.
- *Range*: describes the limitation of the values that may be encoded in this Datapoint Type, possibly for each field. This may be a minimum/maximum indication or an explicit list.
- *Unit*: indicates the unit of the information that can be transported, possibly for each field.

An example of this all is given in Figure 8.

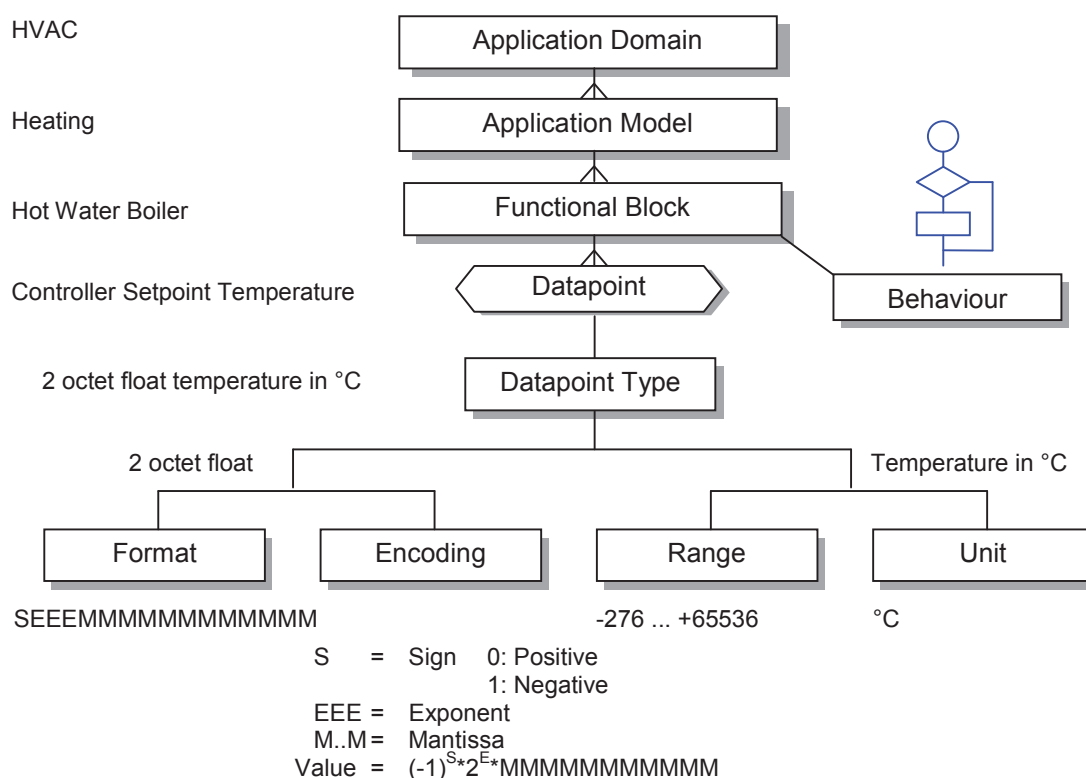


Figure 8 - Example of an Interworking specification

4.1 Principles of HBES Interworking

- HBES Interworking is primarily ensured by the use of standard Datapoint Types, which specify how to encode data that is exchanged between devices by using identical *communication mechanisms*.
- Functional Blocks can be used to **describe** the used Datapoint Types, communication mechanisms and behaviour for a given functionality.
- Functional Blocks may oblige the use of a specific class of Datapoints (in most cases Group Objects), implementation of an additional class of Datapoint (e.g. Interface Objects Property) may be optional.
- A list of the commonly used HBES DPTs is given in Annex A to this standard. A DPT can only be considered compliant to the one given in the Annex when the specified encoding, range and unit really complies with what is realised.

EXAMPLE For 8 bit unsigned integers, value shall indeed be numerical (e.g. no bit-field) and the entire specified range shall be supported without gaps.

- Reserved fields shall be set to the value as specified in the DPT specification. If reserved fields are set to a value other than allowed by the specification, the implementation of the DPT can not be considered as compliant. In case of designing a new DPT, unused (i.e. reserved fields) shall be set to 0b.
- It may be possible to use some of the DPTs in Annex A only in specific applications, others are more general purpose.
- Fields of an even octet size, e.g. 2 octets, should be positioned on even octet number positions within the DPT.
- For parameterization during device set-up and for diagnostic purposes, DMA parameters or HBES Interface Objects should be implemented instead of HBES Group objects.
- Datapoints should not be used bidirectionally.
- If an application provides status information, one (or more) separate Datapoint(s) should be used.
- It is recommended, that necessary time delays are located in the actuators or the application controller and not in the sensors.

4.2 Busload

4.2.1 Repetition rate

The repetition rate shall be selected very carefully as it influences the busload (risk of overload).

NOTE At the planning stage of an installation the possible busload shall be assessed.

The following aspects shall be covered.

4.2.2 Change of value and Delta value

If a device generates information on the bus depending on a Change Of Value (COV) condition, a limitation-mechanism (either by means of hardware [strap] or software features [parameter] shall be provided.

The application shall transmit the value only after the (sensor) value changes at least for the (minimum) Delta value. For fast changing sensor values, the application shall not transmit a new value before a minimum repetition time has elapsed.

4.2.3 Message on request

If this mode is selected, the same requirement as for delta value applies for the application sending the request.

4.2.4 Heart-beat

For sensor values with little changes, smaller than the Delta value or no change at all, the application shall transmit the value after the maximum repetition time heart-beat. Heart-beat communication denotes that

- the sender shall periodically send its data, and
- the receiver shall maintain a time-out timer to check for this periodic transmission and act in a specified way when no sensor signal is received within the given time-out.

Heart-beat can be used to increase application reliability as well as for life-check of a communication partner.

Table 1 specifies the use requirements of heart-beat.

Table 1 – Use of heart-beat

Signal	Sender / receiver	Required ?
sensor values	heart-beat at sender: time-out at receiver:	Y O
calculated values ^a	heart-beat at sender: time-out at receiver:	Y O
safety relevant values	heart-beat at sender: time-out at receiver:	Y Y
life check ^b	heart-beat at sender: time-out at receiver:	Y Y
trigger signal ^c	heart-beat at sender: time-out at receiver:	NA NA
user triggered signals ^d	heart-beat at sender: time-out at receiver:	O NA
Y = mandatory, O = optional, NA = not applicable. ^a By an automation process, scheduler, building management station. Whether time-out and heart-beat is required may depend on the application domain. ^b Datapoints used to check the presence of a partner device. ^c Datapoints using DPT_Trigger. ^d Signals sent exclusively on user interaction.		

4.2.5 Transmission priorities

The priorities should be selected very carefully to ensure fair bus access. The priorities for frames are defined in EN 50090-4-2.

The maximum priority that shall be used for run-time communication is “normal” (01b).

Usage of the priority “urgent” shall be reserved for specific applications only.

Usage of the priority “system” is reserved for communication for system configuration and Management Procedures.

4.2.6 Bus load considerations for Property Clients

When Property values in a Property Server are accessed (write/read) by a Property Client, then the bus load generated by this communication is fully controlled by the Property Client.

At runtime, the Property Client shall therefore guard the following rules to keep the bus load within limits:

- The Property Client shall not access a next Property value before the Property Server has responded to the previous Property access (A_PropertyValue_Response-PDU).
- While waiting for the response of one Property Server, the Property Client shall not address another Property Server. During configuration, this requirement does not apply: a Configuration Client may access more than one device at the same time.
- In subsequent accesses to Property values, in between the response from the Property Server and the next access to a Property value, the Property Client shall guard a longer interframe time than for low priority data. This will allow normal process data to access the bus meanwhile. This may in the Application in the Property client either be given automatically by the delays in processing the received property values, or may be handled explicitly by introducing small wait times of e.g. 1 ms.

4.3 Datapoint Type error handling

4.3.1 Scope

DPT error handling denotes the capability to properly handle and react on not-allowed or unexpected values of DPTs or fields of DPTs.

Senders – devices that provide the data – shall in general not send erroneous data. Receivers may react in several ways to unexpected values.

The rules for DPT error handling listed below apply to entire DPTs in case the DPT exists only of a single field and for each individual field in case the DPT is composed of two or more fields.

Consistency of data does not have to be tested and handled across different fields of structured DPTs.

EXAMPLE It is not required to check whether February 29 is a possible date in DPT_DateTime.

4.3.2 Requirements

4.3.2.1 Reserved fields

Sender: The DPT definition shall specify the value for this field. Usually, this is 0. The sender shall set this field to this value.

Receiver: The receiver shall check whether the field has the specified value. If not, the entire received DPT-value shall be ignored entirely.

EXAMPLE DPT_SceneControl (B₁r₁U₆, DPT_ID = 18.001) is used to call scenes in a device. Bit 6 in this DPT is reserved. If a device received a value encoded according DPT_SceneControl on its scene Input and bit 6 appears to be set, this received value shall be ignored and the device shall not react any further.

4.3.2.2 Not-supported fields

If a field in a DPT is not supported, then the handling shall be as follows:

- Sender: This field shall be set to its default value, which is mostly 0.
- Receiver: The value of this field shall be ignored.

4.3.2.3 Ranges for enumerated fields

Sender: The field shall never be set to a value out of the supported range.

Receiver: It shall be checked whether the value is supported. If the value is not supported, the DPT value shall be ignored.

EXAMPLE A burner in a heating installation may burn more than one fuel type, e.g. oil and gas. It may be possible to select the fuel type to be used by setting the Input FuelSelect. If a not supported value is received, e.g. "solid stated fuel", then the burner controller should ignore the received value.

4.3.2.4 Ranges for numerical values

The handling of the range shall be as follows:

- Sender: The sender shall not send values out of this range.
- Receiver: The receiver may support a subset of the range but shall have proper error handling for received values out of range.

Possible reactions are:

- to ignore the received value, or
- to use the value of the nearest boundary, this is the lowest - or highest supported value for the DPT

EXAMPLE The Input SAPBL in the shutter actuator allows setting the absolute position of a shutter expressed in millimetre. If a blinds actuator that controls blinds that have a maximal drive length of 1 500 mm receives a value of 2 000 mm on the Input SAPBL, it shall move the blinds to their maximal position.

The reaction can be specified in the error handling specification of the DP of the FB. If no such error handling is specified, then the reaction may be device specific.

Precise error handling can per DP be specified in the FB specifications, in the part "Exception handling" of the DP definition (see 5.4.2.1.6.9).

In case the faulty DP-value is set with an A_PropertyValue_Write-service, then the A_Property-Value_Response-PDU may contain either a negative response or the taken lowest or highest value.

4.4 Interpretability of data and data integrity

It shall at any time be possible to unambiguously identify the information transported in a frame solely by evaluating the contents of this frame. The interpretation of the data shall not depend on other frames (from the same or different senders), on the destination of the frame or the moment at which the frame is transmitted.

A frame transmitted by a DP shall always contain the same type and amount of information. To fulfil this requirement, it may be necessary to foresee one or more additional mask fields (bits) in the used DPT to mark one or more data fields as valid or not.

By this, it shall be possible to unambiguously specify the data transported in a frame by a DP according the DPT description: there will only be one definition of format and encoding. The length of (the fields of) the used DPT is thus fixed.

NOTE Exceptions to this fixed length are (character) strings and arrays. For such DPTs, the rule applies to each individual element of the string or array.

The following examples are not allowed.

EXAMPLE 1 Group Object 1 once transports two octets status information and another moment it transports single bit information ("ok", "not ok"). The size differs, also on the bus.

EXAMPLE 2 Group Object 1 is always two octets long, but once it contains a 16 bit counter value and once it contains a compound structure of two times DPT_Scaling.

5 General Functional Block Design and Implementation Rules

5.1 Introduction

The following steps shall be observed when HBES interworking models are designed:

- Describe the Application Domain;
- Describe the Application;
- Describe the Functional Block(s);
- Describe the Datapoint Types.

5.2 Describe the Application Domain

An Application Domain groups related Applications in a 1-or more level structure, depending on the complexity of the Application Domain.

The Application Designer shall document the Application Domain and/or a sub-structure of it for which the Application Model is intended.

5.3 Describe the Application

The Application Designer shall integrate in the Application Specification:

- the **functionality** covered by the proposed Application;
- the **Application Domain** for which the proposed Application is intended.

This shall happen in an inclusive as well as in an exclusive way. This means, that the designer shall describe which functionality is covered and if misunderstanding of the scope of the proposal is real, also of the functionality not covered by the solution. Note that this is not yet a description of the proposal itself, but only a description of the problem or functionality it handles.

Additional guidelines may be given for

- combining optional/mandatory features,
- interacting with other Applications, from the same and/or from other Application Domains.

At this point the Functional Block Designer shall also specify whether the proposal is an exclusive solution for the described functionality or if other solutions are also acceptable.

5.3.1 Conditions for Communication

The Designer shall consider for the entire application the conditions for communication:

Model 1: Data driven or more command oriented.

NOTE The basic run-time communication mechanism in HBES, i.e. Group Objects, favours the use of a data driven approach. Command based solutions restrict implementations more towards Interface Objects only.

Model 2: Event driven or polling or time driven.

NOTE This specifies the conditions on which the data is transmitted on the bus.

The selection of the communication model may influence the usability of the existing HBES system implementations and media as well as the communication of this application with other applications.

The Designer shall consider the used communication model and limit the restrictions on applicable implementations and media as much as possible.

5.3.2 Interpretability of Frames

Any communication between any Functional Blocks shall contain all information so that the receiving Functional Block(s) is/are able to interpret the frame independently of other frames.

EXAMPLE If the value cannot be interpreted as such or is too volatile in time compared to the transmission and processing speed. More interpretation can however be made possible by using structured Data Types.

Non-used bits in the frame data-field shall be set to 0 by the sender. The receiver shall ignore the value of those bits.

5.3.3 Communication Types

5.3.3.1 General

This paragraph lists considerations for the available communication types to be considered when designing Application specification. The selection of the Communication Type has repercussions on the effort for implementation (required memory) and the network mechanisms to be used when accessing the Datapoint (group communication messages versus connection-oriented communication, with possible preceding authorization).

When selecting the communication type, it shall be taken into account that the effect of an update on the behaviour of the application shall not depend on the communication mechanism over which the value is updated.

NOTE The update of the value of an Interface Object Property may not automatically set the Group Object's update flag. Applications depending on the functionality of this flag shall take care in the application functionality to cope with the above requirement.

5.3.3.2 Group Object

A Group Object mainly serves for run-time communication between Functional Blocks at field level.

NOTE So called "Horizontal Communication", opposite to "Vertical Communication", between field level devices and controllers on automation level.

It typically incorporates a single "atomic" Datapoint (possibly with implicit embedded field structure).

When specifying an Interface Object Property to be implemented as Group Object value, the following shall be taken into consideration and be specified:

- Group Objects are the interface for Shared Variables.

This means:

- For an output: that its information can be received by more than one input.
- For an input: that it can receive information from more than one output.

Group Objects thus support n-to-m communication relationships. If – for whatever reason - the application model foresees a limitation (1:m or n:1), this shall be stated in the Functional Block Datapoint description and in the resulting applications, installation and usage guidelines.

- Group Objects are unacknowledged.

This means:

- the interpretation of the communicated value shall be possible *independent of*
 - a previous frame,
 - a value of another Group Object.

In this view control fields inside the data field for interpretability of the remaining data are acceptable.

- Group Objects allow spontaneous transmission.

This means that the function using a Group Object as Input shall be designed in such a way that any unannounced change of its input value is properly handled.

5.3.3.3 Interface Object

Interface Objects are mainly intended for communication between a tool (PC or controller) and a device. Such communication, which is mainly focussed on system and application configuration, is also referred to as vertical communication.

The Interface Objects can in principle also be used for point-to-point communication between devices.

Devices may contain several Interface Objects.

An Interface Object groups several individual Datapoints into an explicit structure. Such a Datapoint may be an array of elements, all of which shall have the same Datapoint Type. Each element may have an implicit data field structure. The value of a Group Object may coincide with that of (an element of) an Interface Object Property. In that case, the reaction of the application when writing the Interface Object Property value shall be identical to the reaction when accessing the corresponding Group Object.

Interface Objects are primarily intended for vertical handling of device parameters by providing an abstract access mechanism independent of the implementation: a tool can simply scan the device's implemented Interface Objects until an empty response (end of list) ensues. Of the individual Interface Objects it can scan their Interface Object Properties and via a dedicated service it can determine the length of a distinct Interface Object Property's array of values.

Any Functional Block **may** be mapped onto an Interface Object: the Inputs, Outputs and Parameters are the Interface Object's Properties. Preferably for each Functional Block, one Interface Object shall be implemented.

All other device control and status variables that can not be assigned to a particular Functional Block shall also be grouped into one or more additional Interface Objects.

5.3.3.4 Memory Mapped Datapoint

These are normally the application parameters, which are written by a management client directly into the memory of the device, using the A_Memory_Write-service.

A memory mapped Datapoint requires from the client side – setting the parameter – substantial knowledge of the parameters. The client side is thus limited to powerful configuration tools and controllers.

Memory types like EEPROM and Flash-ROM only have a limited number of write cycles (approximately 100.000 times in the case of EEPROM at an ambient temperature of 25 °C). If the application requires frequent write events to such memory, this will eventually lead to malfunctioning devices and thus unacceptable reduced life-time of the device. Therefore it is not recommended to use these memory types to store data during run-time communication (e.g. Group Object implemented in EEPROM). It is preferred to limit direct memory access to application configuration purposes only.

If – in spite of this recommendation – objects are still implemented in the above mentioned memory types, the applicant shall draw the installer's attention to the conditions, under which the product's life cycle could be diminished.

5.3.4 Parameters

5.3.4.1 Introduction

Two types of parameters exist:

- Run-time Parameters;
- Configuration Parameters.

5.3.4.2 Group Object Parameters

It is allowed to implement parameters by means of a Group Object, instead of by the usual Memory Mapped Datapoints or Interface Object Property. This is however only allowed if

- none of the possible changes requires a renewed downloading, e.g. by breaking links or making them invalid, by breaking the Interworking,...
- the Functional Block still operates properly also if these Group Object(s) are not associated to a Group Address.

The designer of the Functional Block or the implementer shall take care that this condition is fulfilled.

NOTE It shall be borne in mind that such implementations may cause unwanted side effects.

5.3.4.3 Parameters modifying Interworking

An application program may contain parameters by means of which an installer can reduce the conformity otherwise offered by the application by default. For instance, full default compliance of an application to the State Machine of a Functional Block may be reduced by the help of a parameter. However, such parameters shall not be misused to turn the application into one no longer ensuring compliance to the Functional Block description.

5.3.4.4 Default values for Parameters

The detailed specification of a Parameter Datapoint in a Functional Block description may indicate a default value for this Parameter.

This value is however only a recommended value, not mandatory.

Such default parameter values can be used as ex-factory value and lower the configuration effort for the installer, or as value after a reset.

5.3.5 Diagnostic data

Diagnostic Datapoints are used to access current local/internal status information of a Functional Block that is not used for standard run-time Interworking.

EXAMPLE Local sensor values, local actuator values, information about the local state machine of a FB, locally calculated values such as current setpoints, ...

Diagnostic Datapoints are mainly used for

- visualisation (on a central unit, an operating station), and
- during installation, service and maintenance.

Diagnostic Datapoints are only accessed on request, this is, there is no permanent update of the data on the network. Consequently, there shall be no spontaneous transmission of diagnostic data on the bus (to save bandwidth) and diagnostic Datapoints shall be accessed in client/server mode.

5.4 Describe the Functional Block

5.4.1 Introduction

Once the application is defined, it shall be analysed and split up into one or more Functional Blocks.

When splitting up the application in Functional Blocks, the Designer shall verify whether the Functional Blocks Datapoints make sense to be distributed over the network and can also be shared with/interpreted by other applications.

The description style shall comply with the format specified in 5.4.2.

5.4.2 Abstract Functional Block Description

The Functional Blocks shall be described in the following fixed structure.

5.4.2.1 Functional Block "Functional Block Name"

5.4.2.1.1 Aims and Objectives

For the given Functional Block, this paragraphs gives the:

- Introduction: The designer shall bear in mind that the Functional Blocks shall be understandable and readable as stand-alone topic. Therefore, some introduction on the implemented functionality can be given in this clause.
- Scope: this clause shall explicitly state its usability for or its restriction to one or more applications or application domains.
- Environment: in this clause the interrelation of this Functional Block with other Functional Blocks can be described, in a drawing or textual description, to facilitate the understanding of its scope.
- Introduction to the Functional Specification: in this clause, typically in a textual way, the chosen solution for the 3 above topics can be described.