



System Specifications

3

Interworking

7

Interworking Model

1

Summary


This Chapter specifies the general Interworking Model. It is a guideline for designing Functional Blocks and Datapoint Types to the Application Groups.

The Interworking relevant certification requirements are given in this document.

Version 01.04.01 is a KNX Approved Standard.

This document is part of the KNX Specifications v2.1.

Document updates

| Version | Date | Description |
|----------|------------|---|
| AS 1.0 | 2002.01.03 | Compilation of the Approved Standard. |
| DP 1.1 | 2006.06.08 | <ul style="list-style-type: none">• Update of DPT specification and implementation guidelines.• Reorganised and extended DPT error handling.•  AN040 "Private Messaging" integrated.• Integration of Parameters and Diagnostic Data: definition and specification style.• Integration of E-Mode channel design and implementation guidelines.• Detailed review. |
| DV 1.2 | 2006.12.20 | <ul style="list-style-type: none">• Integration of WGI resolution of comments from Release for Voting.• Preparation of the Draft for Voting. |
| AS 1.3 | 2007.03.14 | <ul style="list-style-type: none">• Preparation of the Approved Standard. |
| AS 1.3 | 2007.10.01 | <ul style="list-style-type: none">• S12 "Channel Codes" clause 7 "System Initialisation" in 4.8.• Editorial correction. |
| AS 1.4 | 2009.06.25 | <ul style="list-style-type: none">• Preparation for publication in the KNX Specifications v2.0. |
| 01.04.01 | 2013.10.28 | <ul style="list-style-type: none">• Editorial updates for the publication of KNX Specifications 2.1. |

References

| | | |
|------|---------------|-------------------------------|
| [01] | Volume 3 | "System Specifications" |
| [02] | Chapter 3/3/2 | "Data Link Layer General" |
| [03] | Chapter 3/3/7 | "Application Layer" |
| [04] | Chapter 3/4/1 | "Application Interface Layer" |
| [05] | Chapter 3/7/2 | "Datapoint Types" |
| [06] | Chapter 3/7/3 | "Standard Identifier Tables" |
| [07] | Volume 5 | "Certification Manual" |
| [08] | Volume 6 | "Profiles" |
| [09] | Volume 7 | "Application Descriptions" |
| [10] | Part 10/1 | "Logical Tase Extended" |

Filename: 03_07_01 Interworking Model v01.04.01 AS.docx
Version: 01.04.01
Status: Approved Standard
Savedate: 2013.10.28
Number of pages: 59

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 5 |
| 1.1 | Introduction to Interworking..... | 5 |
| 1.2 | Interworking Model | 5 |
| 1.3 | Principles of KNX Interworking | 9 |
| 1.4 | Structure of the Interworking Specifications | 10 |
| 2 | Meta-Rules for Interworking | 11 |
| 2.1 | Extensions to the KNX system – private messaging | 11 |
| 2.1.1 | Introduction..... | 11 |
| 2.1.2 | Use cases for extensions | 11 |
| 2.1.3 | Communication stack layer 1 to 7 | 12 |
| 2.1.4 | Linking – communication set-up | 12 |
| 2.1.5 | Parameterization | 13 |
| 2.1.6 | Communication for testing, diagnostics and discovery purposes..... | 15 |
| 2.1.7 | Runtime..... | 16 |
| 2.1.8 | Conditions and tasks of the Certification Department..... | 21 |
| 2.2 | Recommendations..... | 21 |
| 2.3 | Busload | 21 |
| 2.3.1 | Repetition rate..... | 21 |
| 2.3.2 | Change of value and Delta value | 21 |
| 2.3.3 | Message on request..... | 21 |
| 2.3.4 | Heart-beat..... | 22 |
| 2.3.5 | Transmission priorities..... | 22 |
| 2.3.6 | Bus load considerations for Property Clients..... | 23 |
| 2.3.7 | Requirements for KNX Bus devices powered by Ancillary Power Supply (APS) | 23 |
| 2.4 | Datapoint Type error handling..... | 24 |
| 2.5 | Interpretability of data and data integrity | 25 |
| 3 | General Functional Block Design and Implementation Rules..... | 26 |
| 3.1 | Describe the Application Domain | 26 |
| 3.2 | Describe the Application | 26 |
| 3.2.1 | Conditions for Communication..... | 26 |
| 3.2.2 | Interpretability of Frames | 27 |
| 3.2.3 | Communication Types | 27 |
| 3.2.4 | Parameters..... | 29 |
| 3.2.5 | Diagnostic data..... | 30 |
| 3.3 | Describe the Functional Block..... | 30 |
| 3.3.1 | Introduction..... | 30 |
| 3.3.2 | Abstract Functional Block Description..... | 31 |
| 3.4 | Describe the Datapoint Types..... | 43 |
| 3.4.1 | Description..... | 43 |
| 3.4.2 | General Requirements for Datapoint Types | 43 |
| 3.4.3 | Simple Data Types..... | 43 |
| 3.4.4 | Enumerated Datapoint Types..... | 44 |
| 3.4.5 | Structured Datapoint Types | 44 |
| 3.4.6 | Multi-State Datapoint Types..... | 45 |
| 3.4.7 | Status Information..... | 46 |
| 3.4.8 | Datapoint Type Specification | 48 |

| | | |
|----------------|---|-----------|
| 4 | General Channel design and implementation rules | 53 |
| 4.1 | Introduction..... | 53 |
| 4.2 | Describe the Application Domain | 53 |
| 4.3 | Steps in the design of a channel..... | 53 |
| 4.4 | Describe the Channel | 54 |
| 4.5 | Channel Code Rules | 57 |
| 4.5.1 | General classification..... | 57 |
| 4.5.2 | Old ranges - for information | 57 |
| 4.6 | Private channels | 58 |
| 4.7 | Manufacturer specific Connection Codes..... | 58 |
| 4.8 | System initialisation..... | 58 |
| Annex A | (informative) Examples of Functional Block presentation styles | 59 |
| A.1 | Example 1 | 59 |

1 Introduction

1.1 Introduction to Interworking

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 ¹⁾).

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 the Interworking model. Therefore specifications are defined for Functional Blocks, which in turn specify Datapoints and the communication mechanisms to be used. This set of requirements is called "Application Interworking Specifications" (AIS).

The 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 tool, training...).

Interworking is a certification relevant feature and the main message of the KNX logo.

Therefore products not offering Interworking and/or disturbing Interworking cannot bear the KNX logo (thus excluding cluster-subsystems).

The list of specifications may be extended by submitting proposals to the KNX Association.

This document is intended for both the designer of new specification proposals as well as for the person implementing these specifications into KNX products.

1.2 Interworking Model

Application Interworking Specifications are defined for various Application Domains. Each Application Domain encompasses one or more Applications.

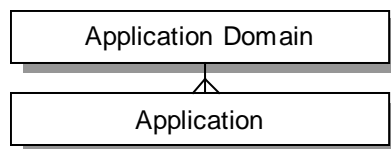


Figure 1 - The KNX Interworking Model
An Application Domain contains one or more Applications

The Application Modelling – this is the process of analysing, deciding on the solution and specifying and agreeing on the model for such an Application – is the responsibility of the various Application Specification Groups of the KNX Association.

The Applications are not 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).

¹⁾ Media couplers are needed if different media are used in an installation.

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.

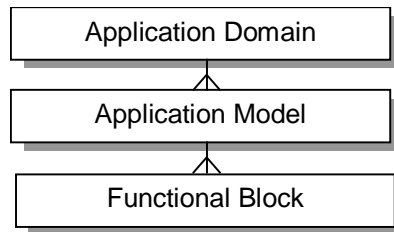


Figure 2 - The KNX Interworking Model
An Application Model contains one or more Functional Blocks

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

The standard graphical representation for a Functional Block is the following:

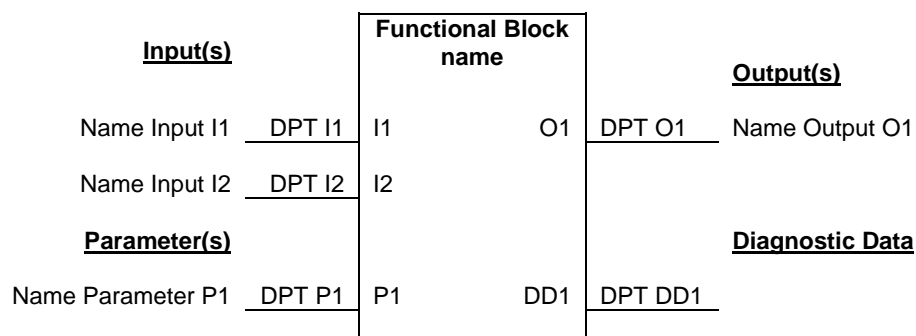


Figure 3 - Standard representation for Functional Blocks

The Datapoints that are typically Inputs are 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 is given, an indication of its Datapoint Type and an abbreviation.

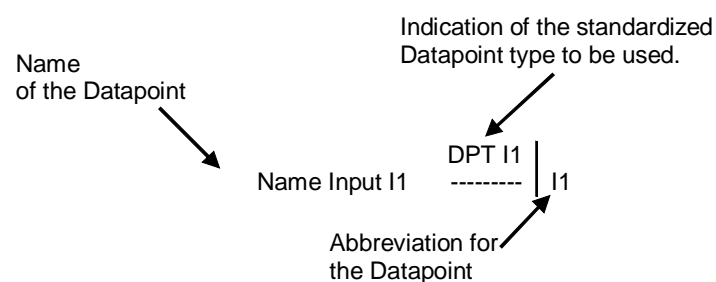


Figure 4 - Datapoints indicated in Functional Blocks

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

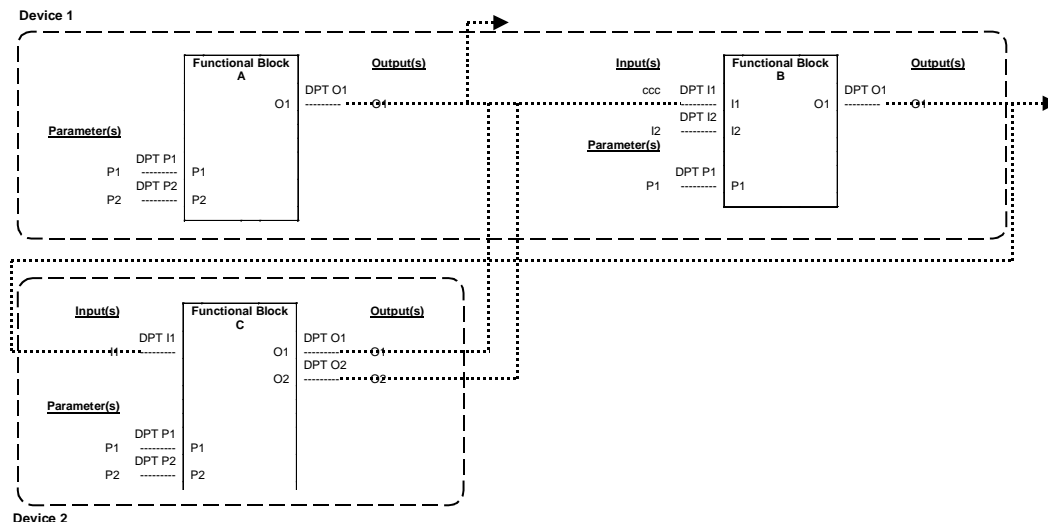


Figure 5 - Functional Blocks grouped in devices and linked

For every Functional Block its behaviour is 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 ²⁾ can be differentiated (for more information see clause 3.2.3 “Communication Types”):
 1. Group Object Datapoint
 2. Interface Object Property Datapoint
 3. Polling Value Datapoint
 4. Memory Mapped Datapoint

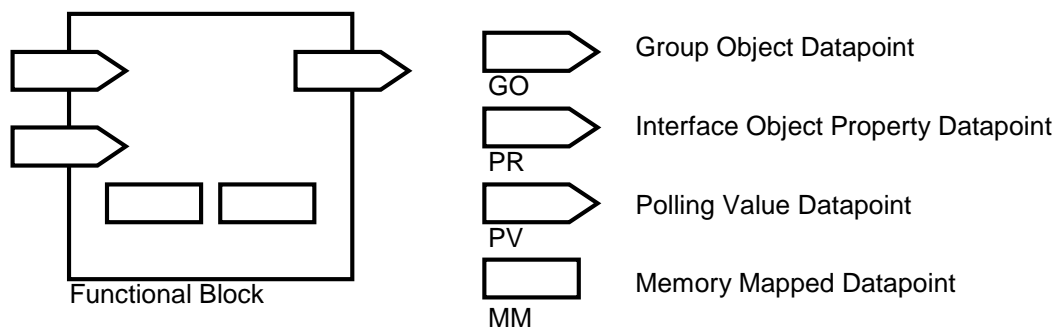


Figure 6 - Functional Block with 5 Datapoints

²⁾ These classes of Datapoints are defined in [01], [02] and [03].

- From the **access** point of view another differentiation of Datapoints is possible:
 1. **Input:** a value that is received and processed by the Functional Block.
 2. **Output:** the value resulting from the process of the Functional Block and to be provided to at least one other Functional Block.
 3. **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.
 4. **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 are standard in format, encoding, range and unit. They are defined in *Datapoint Types* (DPT).

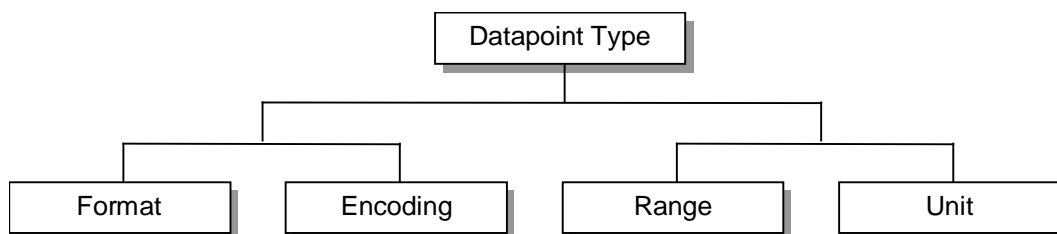


Figure 7 - The information contained in a Datapoint Type definition

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:

1. **Format** describes the sequence and length of the fields, each consisting of one or more bits, of which the Datapoint Type is built up.
2. **Encoding** describes how the data, that shall be transported using this Datapoint Type, is coded using the given format, possibly for each field.
3. **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.
4. **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.

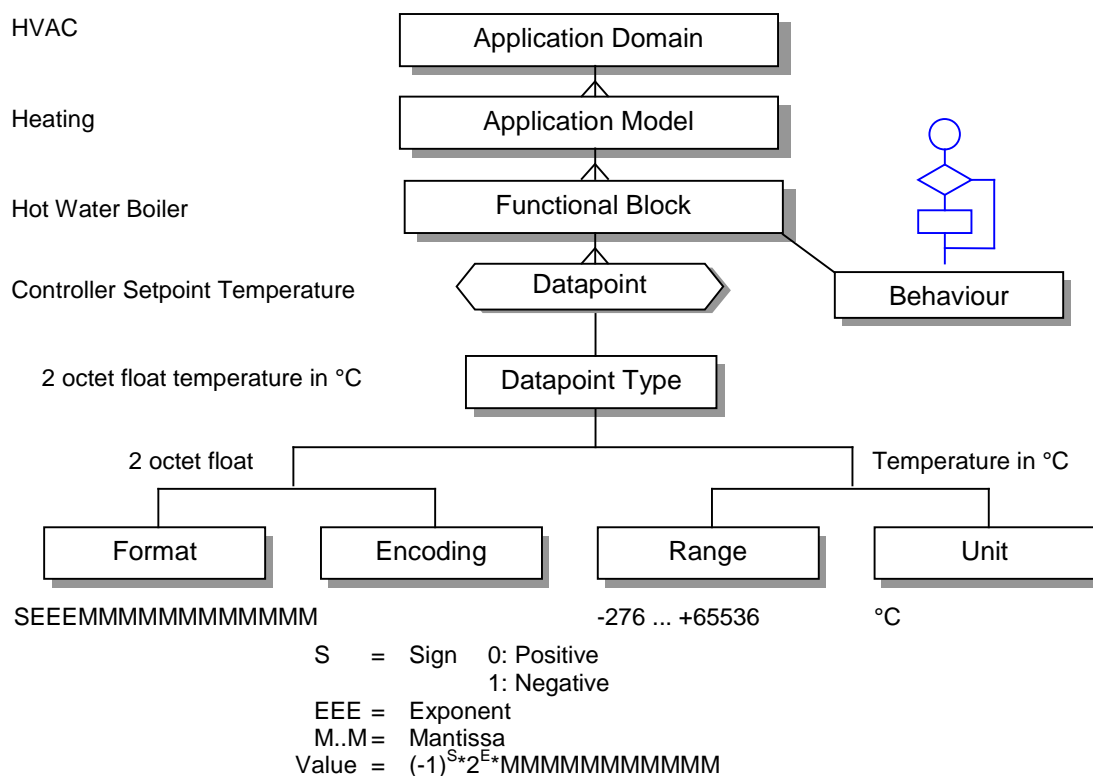


Figure 8 - Example of an Interworking specification

1.3 Principles of KNX Interworking

- As outlined in clause 2 “Meta-Rules for Interworking”, KNX 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 are used to **describe** the used Datapoint Types, communication mechanisms and behaviour for a given functionality.
- In most cases, Functional Blocks oblige the use of a specific class of Datapoints (in most cases Group Objects). In some cases, the implementation of an additional class of Datapoint (e.g. Interface Objects Property) is optional.
- In some cases, the use of a standardised Functional Block for the realization of a given function will be mandatory. In other cases, its use may be recommended.

1.4 Structure of the Interworking Specifications

The KNX Interworking specifications define the standard elements of the Interworking Model (see Figure 6):

1. Application Models
2. Functional Blocks
3. Datapoint Types

This structure is reflected:

- in the design and implementation guidelines in the following clauses, and
- in the following documents:
 - Chapter 3/7/2 "Datapoint Types" ([05]),
 - Chapter 3/7/3 "Standard Identifier Tables" ([06]) and
 - Volume 7 "Application Descriptions" ([09]).

2 Meta-Rules for Interworking

2.1 Extensions to the KNX system – private messaging

2.1.1 Introduction

2.1.1.1 General

With extensions to the standard KNX system is meant any alternative technical solution that goes beyond the standard solutions described in the KNX specifications.

2.1.1.2 Extensions and Certification

If a product is developed that does not take into account the underneath conditions for extensions, the product cannot be KNX certified.

In the case where **KNX certification is not envisaged**, the following applies.

- **For members of KNX Association**, extensions shall be based on the manufacturer specific range of user messages (APCI values 02F8h to 02FEh – see [03]) and the APCI shall be directly followed by the respective KNX Association member code before the actual payload (ASDU) is sent.
- **For non-members**, it is recommended that extensions are based on the manufacturer specific range of user messages (APCI values 02F8h to 02FEh – see [03]) and that the APCI is directly followed by the escape KNX Association member code (FFFFh) before the actual payload (ASDU) is sent.

2.1.2 Use cases for extensions

In case of extensions to the standard KNX system, first of all a distinction has to be made between factory and field use.

In the case of factory use, extensions to the standard KNX system are not subject to submission (neither for information nor for certification) to KNX Association. However, the manufacturer shall in all cases ensure that such functionality used during the production process is by default not accessible for users in the field (unless also part of the mechanisms for test/diagnostics, see later).

In the case of field use, extensions to the standard KNX system are always subject to submission to KNX Association. The following sub- use cases can be distinguished in the case of field use:

- linking, including address assignment, enrolment,
- parameterisation, i.e. parameter setting;....
- test, i.e. temporary communication from and to a device to check its proper reaction to external stimulation.
- diagnostics, i.e. temporary communication from and to a device to check its status in a functioning installation
- discovery, i.e. read out during installation and operation of e.g. online device descriptions, text, company specific device identifier, version information,
- runtime, i.e. any other communication

Broadcast communication shall never be used for extensions in the standard system.

2.1.3 Communication stack layer 1 to 7

2.1.3.1 General

The communication stack mechanisms are defined in [01]. Allowed combinations of these mechanisms are defined in [08]. Any deviation from either Volume is subject to approval by KNX Association.

2.1.3.2 Reserved fields in the protocol

The encoding of some of the services of several layers and several communication layers leaves some values unused. These values are foreseen for future extensions of the KNX protocol. They shall not be used.

2.1.4 Linking – communication set-up

2.1.4.1 General

The linking mechanisms are defined in [01] (Standard system) and [10] (Extended System). Allowed combinations of these mechanisms are defined in [08]. Any deviation from either Volume is subject to approval by the KNX Association.

By linking communication it is understood any communication as specified for the various KNX Configuration Modes that serves for setting up the network configuration and the device configuration as regards its communication to other devices, i.e. typically the linking of Group Objects, Polling Groups and/or Interface Objects.

The procedures used for linking shall be exclusively used for that purpose, as given for the Configuration Mode, and not be used for different purposes. Moreover, these procedures shall be used only as specified for the Configuration Mode, without any deviation or extension.

2.1.4.2 S-Mode plug-ins and stand-alone tools

Owing to the fact that stand-alone tools may use resources that are not reported to ETS (which may cause conflicts in installations), stand-alone S-Mode tools are not recommended for future developments. Instead, appropriate ETS plug-ins should be developed. When a plug-in is developed, the conditions for certification as set out in [08] shall be complied with.

2.1.4.3 E-Mode

2.1.4.3.1 Ctrl-Mode and PB-Mode

Linking shall be based exclusively on the evaluation of the Connection Codes and channel types. No other mechanisms shall be used to either limit or extend the linking capability. It is not allowed to use standardised Channel Codes respectively channel types for other purposes than laid down in the channel definitions.

E-Mode configurators (e.g. controller) shall support all channel types and Connection Codes of the relevant application domain as declared by the manufacturer.

For E-mode, manufacturer specific ranges are defined for Connection Codes (F0h to FFh) respectively channel types (1F00h to 1FEFh) that go beyond those standardised. Channel Codes in this area can only be interpreted together with the KNX Association member code contained in the field Application Manufacturer of DD2. These can be used until for the desired functionality a standardised channel exists or standardisation of such functionality as a channel is not envisaged in KNX Association.

2.1.4.3.2 LTE

As LTE uses zoning information instead of channels, additional LTE zoning parameters are permitted in order to allow manufacturer specific zoning information. (i.e. communication of a device with others belonging to another zone). These parameters are Properties of Interface Objects, whereby the Property Identifiers shall be chosen in the company specific range.

2.1.5 Parameterization

2.1.5.1 During device set-up

2.1.5.1.1 Definition:

This parameterisation is typically done either during the manufacturing process and/or during installation. These parameters are not modified during runtime. Mechanisms used for this purpose shall be DMA and/or Properties of (Reduced) Interface Objects.

2.1.5.1.2 Device set up via DMA parameters

Owing to the engineering nature of S-Mode configuration and its product databases, it is allowed for parameters standardised in a Functional Block to use non-standard Datapoint Types if implemented as DMA parameters in S-Mode.

2.1.5.1.3 Device set up via Properties of (Reduced) Interface Objects

2.1.5.1.3.1 Adding manufacturer specific parameters to standard Interface Objects

For such parameters, the Property Identifiers of such Properties of a standard Interface Object shall be chosen in the manufacturer specific range (see [06] – from 155 to 255).

It is recommended to use standard Datapoint Types for these Properties, e.g. in view of possible future common tool support.

2.1.5.1.3.2 Adding manufacturer specific Interface Object Types with manufacturer specific Properties

These parameters as Properties of a non-standard Interface Object shall be chosen in the manufacturer specific range, both for the Property Identifier as well as the Object Type (see [06] – for Property Identifiers from 51 to 255 and Object Types from 50 001 to 65 535).

It is recommended to use standard Datapoint Types for these Properties, e.g. in view of possible future common tool support.

2.1.5.1.4 Device set up via channels

Manufacturer specific extensions as regards parameters may be realised via manufacturer specific Connection Codes (F0h to FFh). These can be used until for the desired functionality a standard channel exists or standardisation of such functionality as a channel is not envisaged in KNX Association.

It is recommended to use standard Datapoint Types for these extensions.

2.1.5.2 During Runtime

2.1.5.2.1 Definition

These parameters are set during operation, e.g. by the end user or operators (room unit, building management station, visualisation software ...). Such parameters can include setpoints, scheduler and calendar programs, threshold values... Mechanisms used for this purpose shall be either Group Objects, Properties of (Reduced) Interface Objects and/or LTE runtime Properties.

2.1.5.2.2 Parameterisation via Group Objects during Runtime

The requirements as set out in clause 2.1.7 apply.

2.1.5.2.3 Parameterisation via Properties of (Reduced) Interface Objects during Runtime

2.1.5.2.3.1 Adding manufacturer specific parameters to standard Interface Objects

These parameters as Properties of a standard Interface Object shall be chosen in the manufacturer specific range (see [06] – from 155 to 255).

It is recommended to use standard Datapoint Types for these Properties, e.g. in view of possible future common tool support. The use of non-standard Datapoint Types for these Properties is only possible if approved by the KNX Certification Department and/or Working Group Interworking (WGI) at the time of registration.

2.1.5.2.3.2 Adding manufacturer specific Interface Object Types with manufacturer specific Properties

These parameters as Properties of a non-standard Interface Object shall be chosen in the manufacturer specific range, both for the Property Identifier as well as the Object Type (see [06] for Property Identifiers from 51 to 255 and Object Types from 50 001 to 65 535).

It is recommended to use standard Datapoint Types for these Properties, e.g. in view of possible future common tool support. The use of non-standard Datapoint Types for these Properties is only possible if approved by the KNX Certification Department and/or Working Group Interworking (WGI) at the time of registration.

2.1.5.2.4 Parameterisation via channels during runtime

Manufacturer specific extensions as regards parameters during runtime may be realised via manufacturer specific Connection Codes (F0h to FFh). These can be used until for the desired functionality a standardised channel exists or standardisation of such functionality as a channel is not envisaged in KNX.

It is recommended to use standard Datapoint Types for these extensions. The use of non-standard Datapoint Types is only possible if approved by the KNX Certification Department and/or Working Group Interworking (WGI) at the time of registration.

2.1.5.2.5 Parameterisation via LTE runtime Properties

For such parameterisation, the mechanisms as defined in [10] clause 7.6.8 ‘LTE-HEE private data’ shall be used. It is recommended to use standard Datapoint Types for these extensions. The use of non-standard Datapoint Types is only possible if approved by the KNX Certification Department and/or Working Group Interworking (WGI) at the time of registration.

2.1.6 Communication for testing, diagnostics and discovery purposes

Communication for testing, diagnostics and discovery purposes is in principle not restricted to a particular communication mechanism (except broadcast – see above), in order to enable implementation of testing, discovery and diagnostics functionality also on smaller system platforms. However, it is strongly recommended to use point-to-point communication for testing, discovery and diagnostics purposes.

The *communication partner* of the device determines the implementation of testing, discovery and diagnostic functionality.

- In the case where the communication partner is a **management client** (e.g. building management station or BMS functionality realised in visualisation software, permanent part of the installed system), the following solutions are allowed, in decreasing order of preference.
 - a) Use a standard Interface Object Type and a standard Property Identifier (in this case, the Datapoint Type is standard by definition).
 - b) Use a standard Interface Object Type and non-standard Property Identifier and standard Datapoint Type (if approved for use as a Property, see 2.1.7.2.1.2).
 - c) Use a non-standard Interface Object Type, non-standard Property Identifier and standard Datapoint Type (if approved by use as a Property, see 2.1.7.2.1.2).
 - d) Use a Group Object with standard Datapoint Type (if approved for general purpose use, see 2.1.7.2.1.2).
 - e) Use a non-standard Interface Object Type, non-standard Property Identifier and non-standard Datapoint Type – this is only possible if approved by the KNX Certification Department and/or Working Group Interworking (WGI) at the time of registration.
 - f) Use a Group Object with non-standard Datapoint Type – this is only possible if approved by the KNX Certification Department and/or Working Group Interworking (WGI) at the time of registration.
- In the case where the communication partner is a manufacturer specific test tool (e.g. hand held diagnostic device or test software, non-permanent temporary participant in the installed system), it is allowed to use the manufacturer specific range of user messages (02F8h to 02Feh – see [04] clause 2), where the APCI shall be directly followed by the respective KNX member code before the actual payload (ASDU) is sent – subject to validation by the Certification Department and/or Working Group Interworking (WGI).
- In the case where the communication partner is a field device, the requirements of runtime communication apply.

NOTE During diagnostics, it may happen that company specific purely device internal objects react on Interface Object scan. Although it shall be declared by the manufacturer whether such internal objects exist, they are not subject to certification.

2.1.7 Runtime

Table 1 - General conditions for extensions to the standard system in runtime

| Usage of | Conditions |
|-------------------------------|------------|
| Standard DPT | 2.1.7.2.1 |
| Non-standard DPT | 2.1.7.2.2 |
| Standard GO in FB | 2.1.7.2.4 |
| Non-standard GO in FB | 2.1.7.2.3 |
| Standard Property in FB | 2.1.7.2.5 |
| Non-standard Property in FB | 2.1.7.2.6 |
| Standard Interface Object | 2.1.7.2.7 |
| Non-Standard Interface Object | 2.1.7.2.8 |
| Standard Service | None |
| Non-Standard Service | None |

Table 2 - Specific conditions for extensions to the standard system in runtime

| Usage where | Standard? | Usage of | | | |
|----------------|-----------|-----------|------------|-----------|-----------|
| | | DPTs | | GO in FB | |
| | | Yes | No | Yes | No |
| DPTs | Yes | | | | |
| | No | | | | |
| GO in FB | Yes | 2.1.7.3.1 | 2.1.7.3.2 | | |
| | No | 2.1.7.3.3 | 2.1.7.3.4 | | |
| Property in FB | Yes | 2.1.7.3.5 | | 2.1.7.3.6 | |
| | No | 2.1.7.3.9 | 2.1.7.3.10 | 2.1.7.3.7 | 2.1.7.3.8 |
| Functionality | No | 2.1.7.2.9 | | | |

Legend:

| | |
|--|--|
| | This usage is mandatory. |
| | This usage is not allowed. |
| | This usage is subject to approval by KNX Association (Certification Department/WGI). |
| | This usage is highly recommended. |
| | Meaningless combination |
| | See detailed conditions |

2.1.7.1 Priority of conditions

The ‘General Conditions’ in Table 1 above gives general conditions that apply for the respective column, before the specific conditions apply.

2.1.7.2 General conditions

2.1.7.2.1 Usage of standard DPTs

Whenever a standardised DPT can be used, it shall be used.

2.1.7.2.1.1 How can a standard DPT become a non-standard DPT?

- Range
 - Limiting the range
Limiting the range to a range smaller than specified for the DPT or other than specified for the usage of the DPT in a FB makes the DPT a non-standard DP, if implemented in an Input Datapoint.
 - Extending the range
Extending the range to a range larger than specified for the DPT is only allowed for numerical DPTs, under the condition that the functionality of the FB remains unchanged, regardless of the type of Datapoint.
- Reserved fields

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 (e.g. coding value of 64 seconds in DPT_TimeOfDay – DPT 10.001), the DPT turns into a non-standard DPT.

2.1.7.2.1.2 Usage limitations

The standard DPT specifications specify the usage of the DPT. Please refer to the proper use indications in the column “Use” in the DPT specifications in [05].

- “General”

The DPT can be used without any limitation.
- “FB only”

The DP is only approved for usage within one or more specific FBs.

For further information see also [05] clause 1.

Any other usage, regardless of the type of Datapoint is subject to approval by the KNX Association Certification Department, which can call for Working Group Interworking (WGI) approval.

2.1.7.2.2 Usage of non-standard DPTs

Irrespective of further detailed conditions for the usage of non-standard DPTs, the Datapoint Type design guidelines, as laid down in clause 3.4 shall be applied.

EXAMPLE If a non-standard DPT is composed of multiple fields, the smaller fields shall be positioned on the lower significant positions.

Unused, i.e. reserved fields (bits) shall be set to 0b.

Fields of an even octet size, e.g. 2 octets, should be positioned on even octet number positions within the non-standard DPT.

Usage of non-standard DPTs is subject to approval by the KNX Certification Department, which can call for Working Group Interworking (WGI) approval.

Each requested non-standard DPT shall be described in English in a stand-alone document with the standard form for DPT descriptions, as described in clause 3.4.8.1. It is not required that a full-fledged FB description be provided, but aspects of the DP definition form ³⁾ (clause 3.3.2.1.6.1) shall be provided in plain text to explain the intended use of the non-standard DPT and the grounds why no standard DPT can be used.

2.1.7.2.3 Usage of non-standard Group Objects in a FB

Usage of non-standard Group Objects in standard FBs is only allowed under the following conditions.

1. The Group Objects shall be an extension of the FB. It shall not result from the combination of the FB with functionality that would be modelled in separate FBs.
2. The functionality achieved by the non-standard Group Object shall not compete neither conflict with the functionality of any standard Group Object or Property. Specifically, it shall not break the Interworking (functionality) of the standard Group Objects or Properties.

Usage outside these conditions is subject to approval by the KNX Certification Department, which can call for Working Group Interworking (WGI) approval.

2.1.7.2.4 Usage of standard Group Objects in a FB

Implementation of standard mandatory and/or optional Group Objects shall comply with the FB specifications, in particular if the FB specifications foresee a conditional implementation.

2.1.7.2.5 Usage of standard Properties in a FB

Implementation of standard mandatory and/or optional Properties shall comply with the Functional Block specification (including the standard Property Identifier), in particular if the FB specifications foresee a conditional implementation. The Property Identifiers shall always be within the range 51 to 154.

2.1.7.2.6 Usage of non-standard Properties

Implementation of non-standard Properties shall always be located in the manufacturer specific range, i.e. 155 to 255.

2.1.7.2.7 Usage of standard Interface Objects

If an intended functionality shall or may be implemented as an Interface Object according to Functional Block specifications, the standard Object Type shall be used. The Object Type range shall always be within the range 100 to 50 000 (as laid down for each Object Type in [09]).

2.1.7.2.8 Usage of non-standard Interface Objects

If an intended functionality has not yet been standardised as an Interface Object according to Functional Block specifications, the used Object Type shall always be located in the manufacturer specific range, i.e. 50 001 to 65 535. If a standard Interface Object exists that covers the intended functionality, a non-standard Interface Objects shall never be used. As regards the use of Properties in this non-standard Interface Object, the rules as laid down in 2.1.7.3.9 and 2.1.7.3.10 apply.

2.1.7.2.9 New Functionality

Always applies to functionality that has not (yet) been standardised by the KNX Association.

In this case the general conditions apply.

³⁾ Aspects as input/output, number of communication partners, transmission conditions...

2.1.7.3 Detailed conditions

2.1.7.3.1 Usage of a standard DPT in a standard DP of a FB

It is mandatory to use the standard DPT for this DP as specified in the FB specification.

It is not allowed to use other, standard or non-standard, DPTs for the realisation of a DP in a FB, other than the one specified in the FB. The DPT as specified for the DP in the FB specification shall be used, without limitation in range other than the one specified.

2.1.7.3.2 Usage of a non-standard DPT in a standard DP of a FB

This is not allowed.

2.1.7.3.3 Usage of a standard DPT in a non-standard DP of a FB

The usage of standard DPTs for non-standard DPs is highly recommended. It will however be specifically investigated whether the specified encoding, range and unit really complies with what is realised. If it is found not to, the DPT is considered a non-standard DPT.

EXAMPLE 1 For 1 bit values, it will be checked whether DPT_Switch really switches on and off.

EXAMPLE 2 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. During conformity tests, the minimum, maximum and a number of random middle values inside this range will be checked.

2.1.7.3.4 Usage of non-standard DPTs in a non-standard Group Object of a FB

Usage is subject to approval by the KNX Association Certification Department, which can call for Working Group Interworking (WGI) approval. The general conditions for non-standard DPT apply, as specified in 2.1.7.2.2.

2.1.7.3.5 Usage of standard DPTs for standard Properties

The DPT as specified for the standard Property shall be used. Usage of other DPTs, standard or non-standard, is not allowed.

2.1.7.3.6 Realisation of a Datapoint as standard Group Object and a standardised Property

This means that according to the Functional Block specification, data that is accessible as a Group Object may/shall be also implemented as a Property. The implementation shall be according the Functional Block specification: this can either be not allowed, optional or mandatory.

2.1.7.3.7 Realisation of a Datapoint as standard Group Object and a non-standard Property

This means that according to the Functional Block specification, data that is accessible as a Group Object is also implemented as a non-standard property. This is allowed: for conditions, see 2.1.7.2.6.

2.1.7.3.8 Use of non-standard Group Object in a non-standard Property

This means that data that is accessible as a non-standard Group Object is also implemented in a non-standard Property. The general conditions for non-standard Group Objects continue to apply (see 2.1.7.2.3) as well as the conditions for non-standard Properties (see 2.1.7.2.6).

2.1.7.3.9 Use of standard DPT in a non-standard Property

This is highly recommended. The general conditions for non-standard Properties apply (see 2.1.7.2.6).

2.1.7.3.10 Use of non-standard DPT in a non-standard Property

Usage is subject to approval by the KNX Association Certification Department, which can call for approval by Working Group Interworking (WGI). The General conditions for non-standard DPT apply (see 2.1.7.2.2) and for non-standard Properties (see 2.1.7.2.6).

2.1.7.3.11 LTE runtime Properties

For such communication, the mechanisms as defined in [10] clause 7.6.8 “LTE-HEE private data” shall be used. It is recommended to use standard Datapoint Types for these extensions. These mechanisms shall only be used for the realisation of additional non-standard features and are subject to validation by the Certification Department.

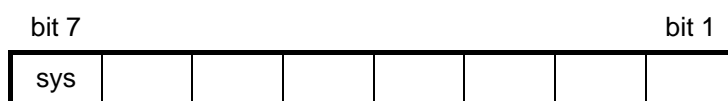
2.1.7.3.12 Support of legacy systems

In order to facilitate the development of gateways with legacy systems, it is allowed to transmit data realised via a standard DPT also in addition with non-standard DPTs. This shall be appropriately documented by the applicant and validated by the Certification Department.

2.1.7.3.13 Polling

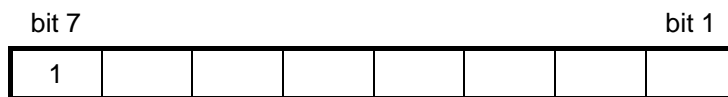
The following conditions shall be met:

2.1.7.3.13.1 - General Structure of Polling Octet



Bit 7: System code flag 0 = application
 1 = system

2.1.7.3.13.2 System Codes for Polling Information



FEh BAU not responding, set by polling master
 FFh Application not running

2.1.7.3.13.3 General Polling Interworking Standard for Applications



Bit 5, 6 Application Alarm Class

00b no alarm

01b Application warning

10b Application detected error (i.e. application program running, but device not working properly; e.g. sensor defect)

11b Application Alarm the properly working device has detected an alarm (e.g. fire alarm)

Bit 4 to 0 This bits are application specific standard. These bits are to be set to 0 until further polling formats are defined.

2.1.8 Conditions and tasks of the Certification Department

According to the certification procedure as set out in [07], compliance with the above requirements for system extensions shall be provided by the manufacturer by providing the appropriate filled forms and annexes at the time of registration of the relevant device.

In case the relevant device for registration by the Certification Department is rejected (e.g. where it is established that the above rules are not met), the following KNX Association competent bodies can be addressed for appeal:

- Working Group Interworking
- KNX Technical Board
- Management Board

In the case where the Management Board confirms the decision of the Certification Department, no further appeal is possible and the submitted device has to be upgraded to comply with the requirements.

2.2 Recommendations

1. Datapoints should not be used bidirectionally.
2. If an application provides status information, one (or more) separate Datapoint(s) should be used.
3. It is recommended, that necessary time delays are located in the actuators or the application controller and not in the sensors.

2.3 Busload

2.3.1 Repetition rate

This 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.

2.3.2 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 manufacturer shall describe this mechanism.

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 is not allowed to transmit a new value before a minimum repetition time has elapsed.

2.3.3 Message on request

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

2.3.4 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 acts 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 3 specifies the use requirements of heart-beat.

Table 3 – Use of heart-beat

| signal | sender / receiver | Standard Mode LTE standard mode interface | LTE-HEE |
|--|--|---|----------|
| sensor values | heart-beat at sender: time-out at receiver: | M O | M O |
| calculated values ^{a)} | heart-beat at sender: time-out at receiver: | M O | M O |
| safety relevant values | heart-beat at sender: time-out at receiver: | M M | M M |
| life check ^{b)} | heart-beat at sender: time-out at receiver: | M M | M M |
| trigger signal ^{c)} | heart-beat at sender: time-out at receiver: | NA NA | NA NA |
| user triggered signals ^{d)} | heart-beat at sender: time-out at receiver: | O NA | O NA |
| M = 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. | | | |

2.3.5 Transmission priorities

The priorities should be selected very carefully to ensure fair bus access. The priorities for frames are defined in [02].

The maximum priority that is allowed for run-time communication is “normal” (01b). If for a Datapoint the “normal” priority is specified, an implementation can use either the “normal” or the “low” priority. If for a Datapoint the “low” priority is specified, an implementation shall only use the “low” priority.

Usage of the priority “urgent” is only allowed for implementations of Datapoints of Functional Blocks or Channels for which this is specified explicitly. Any other usage is subject to approval by the KNX Association Certification Department, which can call Working Group Interworking for assistance.

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

2.3.6 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.

1. 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).
2. 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.

3. 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.

2.3.7 Requirements for KNX Bus devices powered by Ancillary Power Supply (APS)

If an Application Module (AM) of a KNX bus device is powered by an APS, it shall not disturb the bus with e.g. BUSY acknowledgments in case of receiving frames if this Application Module is not active.

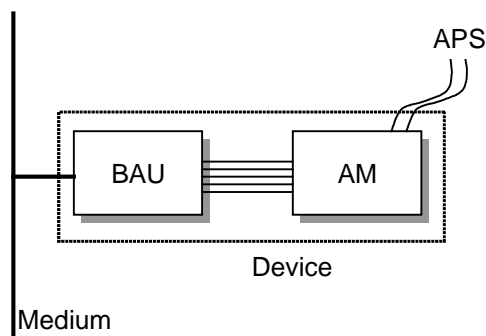


Figure 9 - AM with APS

2.4 Datapoint Type error handling

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.

It is not required that consistency of data is 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.

Requirements

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.

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.

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 with a value encoded according DPT_FuelType (N₈, DPT_ID = 20.100). If a not supported value is received, e.g. "solid stated fuel", then the burner controller should ignore the received value.

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, according DPT_Length_mm (U₁₆, DPT_ID = 7.011). If a blinds actuator that controls blind that have a maximal drive length of 1500 mm receives a value of 2000 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 3.3.2.1.6.9).

Property services

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.

In case the faulty DP-value is transmitted via an A_PropertyValue_InfoReport-service, then the receiver may either ignore the value or take the lowest or highest value.

2.5 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 standard 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.

Standard DPs and FBs automatically comply with this requirement.

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.

3 General Functional Block Design and Implementation Rules

Following considerations shall lead to a proper outlining and understanding of the proposal by the designers when drawing up new Interworking proposals:

1. Describe the Application Domain
2. Describe the Application
3. Describe the Functional Block(s)
4. Describe the Datapoint Types

3.1 Describe the Application Domain

The document [09] of these KNX Specifications lists all standard KNX Applications as models, which can be implemented.

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.

3.2 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. If exclusive, the Functional Blocks becomes certification relevant.

3.2.1 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 KNX, using 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 KNX 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.

3.2.2 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 ⁴⁾ or is too volatile in time compared to the transmission and processing speed.

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.

3.2.3 Communication Types

These clauses list considerations for the available communication types to be considered when designing Application specification. The selection of the Communication Type has not only 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) but also on assessment for certification.

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 In current implementations (situation 05/2006), the update of the value of an Interface Object Property does not 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.

3.2.3.1 Group Object

A Group Object mainly serves for run-time communication between Functional Blocks at field level⁵. 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:

1. 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.

2. Group Objects are unacknowledged.

This means:

- ◆ ²
 - 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.

⁴⁾ Though more interpretation can be made possible by using structured Data Types.

⁵⁾ So called "Horizontal Communication", opposite to "Vertical Communication", between field level devices and controllers on automation level.

3. Group Objects allow spontaneous transmission.

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

3.2.3.2 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.
- The KNX Specifications already specify a number of system parameters (address and association table, device info, ...) in terms of Interface Objects. If implemented as Interface Objects, these system parameters shall comply with the format and coding as specified in the KNX Specifications.
- 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.

Remark:

- An additional Interface Object may be implemented in each device or in the ETS-database to map the correspondence of implemented Group Objects to Properties of Interface Objects. The structure of this object is laid down in the [04] clause 4.4 "Group Objects Indirection".

3.2.3.3 Memory Mapped Datapoint

- These are normally the application parameters, which are written by a management client (ETS or controller) 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 (ETS®) 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.

3.2.4 Parameters

Two types of parameters exist:

1. Run-time Parameters
2. Configuration Parameters

3.2.4.1 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

1. a global or approved Datapoint Type is used.
2. none of the possible changes requires a renewed downloading, e.g. by breaking links or making them invalid, by breaking the Interworking,...
3. 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 has to be borne in mind that such implementations may cause unwanted side effects.

3.2.4.2 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, parameters of an application program may not be used to allow installers to modify a default complying application program in a fully non-complying one (e.g. replacing Group Objects with standard Datapoint Types to Group Objects with non-standard Datapoint Types).

Such parameters may only be realized as memory mapped parameters set by the ETS and not by parameters implemented as Interface Object Property.

3.2.4.3 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.

E-Mode channel definitions may however specify mandatory parameter values.

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.

3.2.5 Diagnostic data

Diagnostic Datapoints are used to access current local/internal status information of a Functional Blocks 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 actual 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.

State-of-the-art diagnostic data can be standardised.

However, diagnostic data accessing Functional Block internal data or data related to manufacturer- and implementation specific mechanisms need not to be standardised.

3.3 Describe the Functional Block

3.3.1 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 take into account,

- whether the result can be realistically integrated in existing BAUs or will require external process power;
- that 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 clause 3.3.2 "Abstract Functional Block Description".

3.3.2 Abstract Functional Block Description

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

3.3.2.1 Functional Block "Functional Block Name"

3.3.2.1.1 Aims and Objectives

For the given Functional Block, these clauses give the:

a) Introduction

The designer has to 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 here.

b) Scope

It shall explicitly state its usability for or its restriction to one or more applications or application domains.

c) Environment

Here as well, 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.

d) Introduction to the Functional Specification

Here, typically in a textual way, the chosen solution for the 3 above topics can be described. A comparison to alternative rejected solution may as well be at its place here.

3.3.2.1.2 Functional Specification

This clause is the first detailed specification of the inputs, outputs and parameters. It may depict a communication flow, a flow chart, picture, state machine, ...

3.3.2.1.3 Constraints

The "Constraints" shall not be a further description of the scope. It only relates to the chosen solution. It lists limitations inherent to the chosen solution and lists hints for implementation and practical use. This may include limitation to certain communication media or allow sleeping devices, etc. it may also give guidelines for the combination of multiple instances of this or different Functional Blocks in a single device, specifically concerning the availability of Datapoints.

3.3.2.1.4 Functional Block Diagram

If any abbreviations are used in the Functional Block Description, they should first be described.

The presentation style should look as given in the example in Figure 10. This presentation style may vary. Alternative possible styles are given in Annex A. It is important that the Functional Block diagram clearly groups the Inputs, the Outputs, the Parameters, possible diagnostic data and hardwired inputs and outputs. The abbreviations of the DPs shall be given and possibly an indication of mandatory and optional DPs.

| Sunblind Actuator Basic (SAB) | | | |
|--|---------|---|--------|
| Inputs | | Outputs | |
| Move Up/Down | (MUD) | Info Move Up Down (IMUD) | |
| StopStep UpDown | (SSUD) | Current Absolute Position Blinds Length (CAPBL) | |
| Dedicated Stop | (STOP) | Current Absolute Position Blinds Percentage (CAPBP) | |
| Preset Position | (PP) | Current Absolute Position Slat Percentage (CAPSP) | |
| Set Absolute Position Blinds Percentage | (SAPBP) | Current Absolute Position Slats Degrees (CAPSD) | |
| Set Absolute Position Blinds Length(SAPBL) | | Valid Current Absolute Position (VCAP) | |
| Scene Learn Mode Enable | (SLME) | | |
| Set Absolute Position Slats Percentage | (SAPSP) | | |
| | | | |
| Set Absolute Position Slats Degrees | (SAPSD) | | |
| Scene Number | (SN) | | |
| Forced | (FO) | | |
| Wind Alarm | (WA) | | |
| Rain Alarm | (RA) | | |
| Frost Alarm | (FA) | | |
| additional I/Os | | Parameters | |
| Sensor for detecting END Positions | | Reversion Pause Time | (RPT) |
| | | Move Up/Down Time | (MUDT) |
| | | Slat Step Time | (SST) |
| | | Preset Position Time | (PPT) |
| | | Preset Position Percentage | (PPP) |
| | | Preset Position Length | (PPL) |
| | | Preset Slat Percentage | (PSP) |
| | | Preset Slat Angle | (PSA) |
| | | Blinds Position for Scene Control | (BPSN) |
| | | Slats Position for Scene Control | (SPSN) |
| | | Storage Function for Scene Number | (SFSN) |
| | | Reaction on Wind Alarm | (RWA) |
| | | Heartbeat Wind Alarm | (HWA) |
| | | Reaction on Rain Alarm | (RRA) |
| | | Heartbeat Rain Alarm | (HRA) |
| | | Reaction on Frost Alarm | (RFA) |
| | | Heartbeat Frost Alarm | (HFA) |
| | | Maximum Slat Move Time | (MSMT) |
| | | Enable Blinds Mode | (EBM) |

Figure 10 – Functional Block diagram (Example)

3.3.2.1.5 Datapoints

The description of the separate Datapoints (inputs, outputs and parameters) starts with the table as below, which gives an overview as in the Functional Block Description, extended with

- the Name,
- the indication whether the Datapoint is mandatory or optional and
- the PID to be used when the Datapoint is implemented as Interface Object Property.
(The Interface Object PID and Property Name are assigned by the KNX Association.)

The first row in the table below indicates the Interface Object Type that shall be used if the Functional Block is implemented as Interface Object ⁶⁾.

| ID | Name | Abbr. | Description | Datapoint Type | M/O |
|---------------------|----------------------|-------|-------------------------|----------------|-----|
| 1 | PID_OBJECT_TYPE | | Object Type | PropDataType | M |
| Input(s) | | | | | |
| ID | Name | Abbr. | Description | Datapoint Type | M/O |
| xa | PID_NAME_INPUT_1 | I1 | Description Input 1 | ID_DPT_I1 | M |
| xb | PID_NAME_INPUT_2 | I2 | Description Input 2 | ID_DPT_I2 | M |
| Output(s) | | | | | |
| ID | Name | Abbr. | Description | Datapoint Type | M/O |
| xd | PID_NAME_OUTPUT_1 | O1 | Description Output 1 | ID_DPT_O1 | M |
| Parameter(s) | | | | | |
| ID | Name | Abbr. | Description | Datapoint Type | M/O |
| xf | PID_NAME_PARAMETER_1 | P1 | Description Parameter 1 | ID_DPT_P1 | M |

⁶⁾ As follows from the general definition of Interface Object, see [04], this Object Type is contained in the value of the first property of the Interface Object. Its PID is therefore always “1”.

3.3.2.1.6 Detailed specification of inputs and outputs

3.3.2.1.6.1 General form

Each input and Output shall be specified using the form as laid down in Figure 11.

| | | | | | | | |
|--------------------------------|-------------------------------------|--------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|-------------------------------------|
| DP Name: | | | Abbr.: | | | Mandatory | <input type="checkbox"/> |
| FB Name: | | | | | Can be internal | <input type="checkbox"/> | |
| Description | | | | | | | |
| | | | | | | | |
| Datapoint Type | | | | | | | |
| DPT_Name: | | | | | | | |
| DPT Format: | | | | | DPT_ID: | | |
| Field | Description | | | Supp. | Range | Unit | Default |
| Field1 | | | | | | | |
| Field2 | | | | | | | |
| ... | | | | | | | |
| Access Type | | | | | | | |
| ◆ Input | | | | | | | |
| N → this | <input checked="" type="checkbox"/> | 1 → this | <input type="checkbox"/> | | | | |
| Spontaneous | <input checked="" type="checkbox"/> | Cyclically: | <input type="checkbox"/> | Time-out: | | | |
| Request | <input type="checkbox"/> | Polling: | <input type="checkbox"/> | Period: | | | |
| ◆ Output | | | | | | | |
| this → M | <input checked="" type="checkbox"/> | this → 1 | <input type="checkbox"/> | | | | |
| Spontaneous | <input checked="" type="checkbox"/> | COV: | <input checked="" type="checkbox"/> | Δ-Value: | | | |
| | | Cyclic | <input checked="" type="checkbox"/> | Period: | | | |
| Request | <input checked="" type="checkbox"/> | | | | | | |
| Communication Type | | | | | | | |
| ◆ Group Object Datapoint | | | | | | Mandatory: | <input checked="" type="checkbox"/> |
| Default Group Address: | | | | | | | |
| ◆ Property | | | | | | Mandatory: | <input type="checkbox"/> |
| DP Address: (in the server) | IO Type(ID): Start-Index: | Property ID: N° of elements | | | | | |
| Property access: | Read only | <input type="checkbox"/> | Read/Write | <input checked="" type="checkbox"/> | | | |
| Protection *) | Read level | - | Write level | - | | | |
| Dynamics | | | | | | | |
| Power down: | Save: | <input type="checkbox"/> | | | | | |
| Power up: | Value: | No initialisation: | <input checked="" type="checkbox"/> | Default value: | <input type="checkbox"/> | | |
| | | Saved value: | <input type="checkbox"/> | Current value (not for input): | <input type="checkbox"/> | | |
| | Transmit on bus (only for output): | | <input type="checkbox"/> | Read from bus (only for input): | <input type="checkbox"/> | | |
| Exception Handling | | | | | | | |
| | | | | | | | |
| Special Features | | | | | | | |
| | | | | | | | |

Figure 11 – Specification form for Inputs and Outputs

3.3.2.1.6.2 Datapoint identification

◆ **DP Name**

This field shall contain the full name of the Datapoint.

◆ **Abbreviation**

This field shall contain the abbreviation of the Datapoint name.

This shall be the abbreviation that is used in the Functional Block Diagram and in the DP list.

◆ **FB Name**

For completeness of the DP identification, the name of the Functional Block that this DP is part of, shall be repeated here.

3.3.2.1.6.3 Implementation requirements

◆ **Mandatory**

This checkbox shall be marked to indicate that the DP shall be implemented in order to be compliant with the FB specification. If this checkbox is not marked, then implementation of this DP is optional. The setting of this checkbox shall be in line with the indication about whether or not the DP is mandatory in the Datapoint List and in the Functional Block diagram.

◆ **Can be internal**

If this FB is in a device implemented together with one or more other FBs that are the normal communication partners of this DP in this FB, according to the standard application model, then it may be meaningful to leave out this DP implementation: its functionality may be realised by purely device internal communication. If this checkbox is marked, this DP may be not implemented. If this checkbox is not marked, this DP shall always be implemented.

This setting is independent of whether a DP is mandatory or optional.

3.3.2.1.6.4 Datapoint description

This field shall contain the full text description of the functionality of the DP. It may repeat part of the functional specification (see 3.3.2.1.2) of the FB and shall give closer DP specific requirements.

3.3.2.1.6.5 Applied DPT and DPT usage

◆ **DPT_Name, DPT_Format, DPT_ID**

These fields shall contain the DPT_Name, DPT_Format and DPT_ID of the Datapoint Type that shall be used for this Datapoint. This information shall be copied unmodified from

- an accepted Datapoint type as given in [05], or
- a new DPT that is at first introduced in this FB; in this latter case the DPT specification shall be provided at the end of the document.

EXAMPLE Name "DPT_Switch" with ID "1.001" and format b₁

For each field in the used DPT a row shall be specified specifying its fieldname, its functional description, support, range, unit and default value.

◆ **Field**

The cells underneath the label "Field" shall contain the name of the fields, in subsequent rows, as many as there are fields in the used DPT. These names shall be copied unchanged from the DPT specification and serve for proper referencing.

◆ **Description**

The cells underneath description shall contain the specific functional specification of this field of the DPT for this specific Datapoint.

◆ Support

The cells underneath *Support* shall indicate whether the specific field shall be supported for this use of the DPT for this DP. This is possible only if the DPT is composed of two or more fields. The possible settings are “M” (mandatory), “O” (optional) and NA (not applicable).

Please refer to clause 2.4 for the handling of reserved fields in sender and receiver.

◆ Range

The cells underneath *Range* shall specify per field the range that shall be supported for the usage of the field of the DPT in this DP.

It is allowed to limit the range given by the used DPT for a DP. If the default range of the DPT is used, it is not necessary to fill in this field. The limitation of the range is a decision of the designer. The implementation shall comply with the indicated range (without limitation).

It is not allowed to extend the range beyond the range specified in the DPT specification.

For clear understanding, the range specification shall include the unit.

EXAMPLES {0, 1}, [-273°C ...670 760°C]

Please refer to clause 2.4 for the handling of ranges in sender and receiver.

◆ Unit

The cell(s) underneath *Unit* shall specify the unit(s) of the (field of) the used DPT. This is only repeated here from the DPT specification for better understanding of the DP description. It shall thus comply with the unit of the referred DPT.

EXAMPLE °C

◆ Default

The cell(s) underneath *Default* shall contain the default values of (the fields of) the DPT.

If this cell is empty for a field, then no default value is mandatory. The implementer is free to use a – free selectable – default value.

If a default value is filled in here, then this default value shall be used by the implementer.

EXAMPLE 20°C

For clear understanding, the default value specification shall include the unit.

3.3.2.1.6.6 Access Type

3.3.2.1.6.6.1 General

This part of the DP description form shall specify whether the DP is an Input or an Output.

NOTE For Parameters and Diagnostic data, a different form is used.

If a DP is an Input, then the rows for specifying an Output shall be removed and vice-versa.

It is possible - but strongly advised against - that a DP is used both as Input as well as Output. The DP is in this case a bidirectional DP and the rows for both Input as well as for Output shall be filled in.

3.3.2.1.6.6.2 Input

Exactly one of the checkboxes **N → this** and **1 → this** shall be set.

◆ N → this (“N to this”)

The marking of this checkbox shall indicate that this input is typically set by more than one client.

◆ 1 → this (“One to this”)

The marking of this checkbox shall indicate that this input is typically set by only a single client.

NOTE Group Objects always allow a DP to be set from multiple senders. This cannot be prevented by the KNX system. However, these indications indicate the typical communication as specified by the application model. Also with GOs it is possible to have a 1 → this communication model if the application model only foresees one sender.

This indication is relevant for acceptance of usability of certain Datapoint Types as specified in clause 3.4.2 "General Requirements for Datapoint Types".

EXAMPLE In case a DP is formatted as a bit field and multiple senders are foreseen (N → this) it is very likely that a bit mask will be required to allow one sender to exactly change these bits which it intends to change without changing any bits possibly already accessed by other senders.

This setting is not relevant for the implementer. The actual number of associations to the Datapoint results from the projecting of the application by the electrical installer.

◆ Spontaneous, Cyclically and Time-out

The setting *Spontaneous* in an Input shall indicate that the value of the Datapoint is updated by a client without request by the logics in the Functional Block itself.

This is the most typical communication way for inputs.

The setting *Cyclically* shall mean that the application model prescribes that the Input be written cyclically.

This is a typical behaviour for automatic processes and is less typical for sensor-actuator communication.

The cell *Time-out* may only be filled in if the preceding cell *Cyclically* is also set.

If no new write access to this Input has occurred during a period larger than the value specified by the Time-out, then the application shall apply a certain error handling as laid down in 3.3.2.1.6.9.

◆ Request, polling, period

The setting *Request* shall indicate that the logics in the Functional Block shall actively ask the communication partners in the network for an update of its Datapoint value.

EXAMPLE An application can do this by setting the read request flag for the Group Object.

This is a possible, but uncommon communication way. It can be used after start-up of a device to get up-to-date values as soon as possible, or as part of the error handling in case the periodic spontaneous setting times out (see above).

The setting *Polling* shall indicate that the DP will send requests on the network for an update of its value.

This can be done for Group Objects as well as for Properties.

The setting *Period* may only be filled in if the preceding cell *Polling* is also set. It shall indicate the time between subsequent polling requests.

3.3.2.1.6.6.3 Output

Exactly one of the checkboxes This → M and This → 1 this shall be set.

◆ This → M ("This to M")

The marking of this checkbox shall indicate that the Output typically communicates with more than one receiver (multicast).

◆ This → 1 ("This to 1")

The marking of this checkbox shall indicate that the Output typically communicates with only a single receiver.

◆ Spontaneous, COV and Δ-Value, Cyclic and Period

The setting *Spontaneous* for an Output shall mean that the logics in the FB shall decide to send this DP value on the bus without external request, but due to internal conditions.

NOTE If the DP is realised as a GO, this means that the GO-value is transmitted using the service A_GroupValue_Write.

This is the most typical communication way for Outputs.

Such internal conditions that lead to the transmission of the Output value can be:

- an operation on the HMI of the appliance (e.g. a rocker of a push button is pressed), or
- a change of signal of an input (e.g. a binary or analogue input changes value), or
- an internal calculation of a variable, possibly triggered by the reception of new values on one or more Inputs, results in new data for the Output,
- etc.

The setting *COV* (“Change of Value”) shall mean that the Output value shall only be transmitted on the bus if it changes value. This is the most typical setting for an Output. This is typical for physical event triggered sending, like human interaction on a push button or threshold level passing.

The cell *Δ-Value* (“Delta value”) shall specify the minimal value over which the data of the Output shall change before it shall be transmitted on the bus. A *Δ-Value* may only be specified if the setting *COV* is set. It allows for preventing too frequent transmissions on the bus that may be caused by slightly fluctuating Output values.

EXAMPLE The Temperature Output of a FB Room Temperature Sensor has a *COV*-value of 0,2°C.

The cell *Min repetition time* (“minimal repetition time”) shall specify the shortest period in between two successive transmissions of the Output value. It is not allowed to transmit the Output value with shorter intervals than the value specified here. *Min repetition time* can be set with or without setting a *Δ-Value*.

EXAMPLE The Temperature Output of a FB Room Temperature Sensor has a *Min Rep Time* of 10 s. This avoids from all subsequent data (heat demands, boiler commands ...) to be calculated and transmitted on the bus too frequently.

The setting *Cyclic* shall specify whether or not the Output value shall be transmitted periodically on the bus or not. If set, the cell *Period* shall specify the transmission period. *Cyclic* can be set independently of the *COV*-condition.

EXAMPLE The Temperature Output of a FB Room Temperature Sensor shall transmit its value periodically with a repetition period of 15 min, also if it has not changed compared to the previous transmission.

The setting *Request* shall specify whether or not the DP value shall be transmitted on the bus after request from the bus.

This is a less frequently used but not uncommon communication way for an Output. It can be set independently of the setting of *Spontaneous* transmission condition. It is most meaningful for sensor data, to allow a communication partner after power-up to immediately request an up-to-date value of its input.

3.3.2.1.6.7 Communication Type

3.3.2.1.6.7.1 General

A Datapoint can in KNX be realised as

- a Group Object,
- a Property (element) of an Interface Object,
- a memory mapped value, or
- a Fast Polling Value.

There are no requirements towards the format of memory mapped Datapoints: it is not required that these be compliant with any standard DPT.

Fast Polling is not supported on all media and is therefore not part of this standard DP specification form.

Therefore, this part “Communication” type of the DP specification form only allows classifying a DPT to be implemented as either Group Object and/or Property (element) of an Interface Object.

If it is not allowed to implement the Datapoint according either one of the communication types then this part of the DP specification form shall be removed.

If it is optionally allowed, but not mandatory, to implement the DP according either communication type, then this relevant part of the DP definition form shall be filled in, but the check box *Mandatory* shall not be checked.

If the implementation of a Datapoint according a certain communication type is mandatory, then this relevant part shall be filled in as well and the check box *Mandatory* shall be checked.

Typically, only one communication type is mandatory and the other communication type is not allowed.

3.3.2.1.6.7.2 Group Object Datapoint

◆ Default Group Address

This field allows specifying whether a device implementing this Group Object Datapoint may be delivered ex-factory already linked to a certain Group Address. This allows for a simple plug&play possibility and gives a guarantee of finding certain information always at the same Group Address.

The assignment of default Group Addresses is rather exceptional and is subject to strict supervision by the TF Interworking of KNX Association.

3.3.2.1.6.7.3 Property

◆ DP address

This part shall specify how the Datapoint can be addressed as Property – or element of a Property.

The IO type shall be the Interface Object Type that has been assigned to the FB. All Property DPs within the FB thus have the same value for this field *IO type*.

The *Property ID* shall list the Property Identifier that shall be used for this Property. The Property Identifier shall be unique within the Functional Block. These Property Identifiers should be given successive values starting from 51 and upwards.

Most commonly, a Property will consist of only one single value ⁷⁾. In this case, the fields *Start_Index* and *N^o of elements* shall both contain the value 1.

It is however possible that other Datapoint values occupy the lower array element(s) of the Property value ⁸⁾. In this case, the field *Start-Index* shall contain the appropriate value and the Datapoint value shall be accessible at this array index.

It is also possible that Datapoint value occupied more than one Property array element. In this case, the field *No of elements* shall contain the appropriate number and the Datapoint value shall be accessible in the implementation as the set of these array elements.

If thus the array aspect of Property values is used and the DPT value occupies two or more array elements, then each of these array elements shall be encoded according the indicated DPT even if the DPT contains multiple fields.

EXAMPLE DPT_Current_Set_F16[3] (221.001) is formatted as F₁₆F₁₆F₁₆. These three temperature values shall be implemented and addressed as one single Property Value.

◆ Property access

This part of the DP definition shall specify whether the Property shall be Read-only or whether the Property may be written as well. The setting of this field influences the field *write-enable field* in the A_PropertyDescription_Response-PDU.

⁷⁾ This counts even if the DPT for coding this Property Value is of a structured DPT, e.g. DPT_DateTime or DPT_TempRoomSetpSetShiftF16[3].

⁸⁾ Note that the KNX specification requires that the Property Value array elements be filled in starting with array index zero and increasing without gaps for further array elements. Please refer to the specification of the array aspects of Properties in [04].

◆ Protection

The fields Read level and Write level shall specify whether or not it is required that the Property client shall authorize for a certain access level for accessing the Property value and the recommended access level for reading and writing. The names as specified in Table 4 in shall be used:

Table 4 – Authorisation level names

| Access Level Name | Interpretation | Authorisation level |
|---------------------|---|---------------------|
| • No access | set by manufacturer at factory | 0, 1 |
| • Management Access | typically ETS (tool on start-up) | 2 |
| • Run-time Access | visualization and management (tool at run time) | 3 |
| • Free Access | field level communication | no authorisation |

The implementer and the installer configuring the application, restrict access to the various levels by locking with keys. The higher the access level, the less people will know the corresponding keys. There is no key for the level Free Access.

The designer shall be aware that the selection of the access level in combination with the 'access' attributes leads to different frame sequences on the bus: a protected Property can only be accessed after authorisation.

Please refer to [04] for the coding of the access levels. The used access level shall be indicated in the PIXIT.

3.3.2.1.6.8 Dynamics

3.3.2.1.6.8.1 Power-down

If the "Save" setting is checked, the implementer shall make the Datapoint value to be saved in non-volatile memory in case of a reset of the device. This is typical for parameter access types – which are normally already stored in non-volatile memory - but may be meaningful for certain Inputs, e.g. defining a setpoint.

3.3.2.1.6.8.2 Power-up

These check boxes shall specify the behaviour of the application concerning the value of the Datapoint after power up. Exactly one check box shall be checked.

◆ No initialisation

This check box shall be checked if there is no standard power up behaviour specified for the Datapoint.

This is most common for output Datapoints that are internally calculated by the application and cannot be read by other devices and for input Datapoints that are only evaluated on reception of a new value.

◆ Default value

This check box shall be checked if the Datapoint shall at power up initialise with its default value. It shall be taken care of that the field(s) Default preceding in this Datapoint definition are filled in.

◆ Saved value

This check box shall be checked if the Datapoint shall at power up initialise with the value saved at power down.

This setting requires that the check box Save in the above part “Power down” is checked as well.

◆ Current value (not for Input)

This check box shall be checked if the Datapoint shall at power up initialise with the correct value. The application shall take care that the Output value is immediately calculated from the Inputs or hardwired inputs (sensors).

This setting makes most sense if the Output value can be read from the bus, or if the setting Transmit on bus is set.

◆ Transmit on bus (only for Output)

This check box shall be checked if the value of the output Datapoint shall be transmitted on the bus after power up.

This shall allow synchronising the remote communication partners of the Datapoint value.

◆ Read from bus (only for input)

This check box shall be checked if the value of the input Datapoint shall be initialised by requesting it from a server via the bus.

This is typical for an Input access type, especially when it is implemented as a Group Object parameter.

3.3.2.1.6.9 Exception handling

This section shall specify all exception handling, e.g. reaction on reception of a value out-of-range, send- or receive time-outs, failing responses to read requests, etc.

This section guarantees Interworking or at least a defined behaviour in exceptional situations. The specified behaviour shall be followed by the implementer.

This field shall always be present, but it is not required to prescribe any exception handling.

3.3.2.1.6.10 Special features

This section shall specify all further normative or informative details about the Datapoint. Any specific settings of the preceding Datapoint definition can be documented here more exhaustively and referred to by table footnotes.

3.3.2.1.7 Detailed specification of Parameters and Diagnostic Data**3.3.2.1.7.1 General form**

Each Parameter and Diagnostic Data shall be specified using the form as laid down in Figure 12. Opposite to Figure 11, there are no optional parts in this form: no part can be left out.

This form specifies the Parameter or Diagnostic Data to be accessed as Property.

Note however that according clause 2.1.5.1.2, it is allowed for an S-Mode appliance to implement parameters – also standard parameters – memory mapped and not comply with the required DPT. For E-Mode devices, the realisation is specified in the channel definitions.

| | | | | | | | |
|--------------------------------|--------------------|------------------------------------|---------------|--|------------------------------------|--|-------------------------------------|
| FB: | | Property Name (Server): | | Mandatory | <input type="checkbox"/> | Optional | <input checked="" type="checkbox"/> |
| Description: | | | | | | | |
| | | | | | | | |
| DPT: | Name | | DPT_ID | | Datatype format: | | |
| Field | Description | | Sup. | Range | Unit | Resol. | Default |
| | | | | | | | |
| Communication: | | | | | | | |
| DP Address: (in the server) | | Object Type: Start-Index: | | Property ID: N° of elements | | | |
| Property access: | | Read only <input type="checkbox"/> | | Read/Write <input checked="" type="checkbox"/> | | | |
| Protection | | Read level | | Write level | | | |
| Exception Handling: | | Value after Power-up: | | Stored Value <input type="checkbox"/> | Act Value <input type="checkbox"/> | Default Value <input type="checkbox"/> | |
| | | | | | | | |
| Special Features: | | | | | | | |
| | | | | | | | |

Figure 12 – Specification form for Parameters and Diagnostic Data

All fields in this form have the same meaning as specified in Figure 11.

◆ **FB**

This field shall contain the abbreviation of the Functional Block name. Opposite to the form specified in Figure 11 for Inputs and Outputs, the Functional Block name is not given in full.

◆ **Property Name (server)**

This shall be the name of the Parameter or Diagnostic Data.

◆ **Default**

This field may specify a default value for the (fields of the) parameter.

Please refer to clause 3.2.4.3 concerning the interpretation of default values for parameters.

◆ **Exception Handling**

As exception handling, only the Value after Power-up is specified. If none of the following check-boxes is ticked, then there is no standard exception handling.

- Stored value

If this check-box is ticked than at power-up, the Parameter or Diagnostic Data shall be initialised with the value saved at power down.

- Curr. Value

If this check-box is ticked, then the Diagnostic Data shall be initialised with the current value. The internal process in the device shall thus at power up calculate the correct and up-to-date value and provide it via this Diagnostic Datapoint.

This check-box is not applicable for Parameters.

- Default Value

If this check-box is ticked, then the Parameter or Diagnostic Data shall be initialised with the default values for all the fields, if these are specified.

3.4 Describe the Datapoint Types

3.4.1 Description

This clause describes the guidelines for designing new Datapoint Types when needed for a Functional Block or an Application.

As much as possible, existing Data Types (format, encoding) and Datapoint Types shall be re-used.

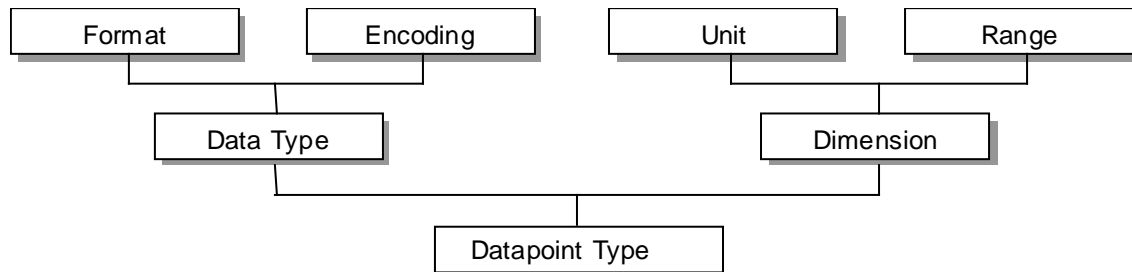


Figure 13 - Design and Structure of Datapoint Types

3.4.2 General Requirements for Datapoint Types

3.4.2.1 Physical Values

A physical value consists of the value itself, this is the data encoded according to some Datapoint Type, and the Dimension. The data is transmitted via the bus. The dimension is not transmitted.

The encoding of the values for transmission is described in the appropriate Datapoint Types.

3.4.2.2 Concatenation

Concatenation means that multiple Datapoint Type values are transported together in a single data-field in a frame over the bus.

- The concatenation of multiple values, of different Datapoint Types or not, individually destined for different Functional Blocks, located in the same or different devices, is not allowed.

This prevents

1. that data is sent on the bus of which one device interprets only one field and another device interprets a second field, and so on.
2. that data is sent on the bus for which one Functional Block in a device listens to the first field in the data and another Functional Block in a device listens to a second field in the data.

EXAMPLE In this way, it is prevented that the 2 relays in a two way blinds actuator are switched with one frame by having each of them react on a different part of the received data.

This definition of concatenation shall not be mistaken for the definition of “Structured Datapoint Types” (see clause 3.4.5) or “Status Information” (see clause 3.4.7).

As a Functional Block shall entirely be implemented in one single device and must not be split up over 2 or more devices, identical channels in a device are therefore clearly separate but identical Functional Blocks.

- Concatenation of one Datapoint Type destined for a single Functional Block in one or more devices is allowed for the exchange of Interface Object Property Values and Polling Values.

3.4.3 Simple Data Types

For the design of single Datapoint Types it is recommended to use as much as possible existing data formats from international standards.

3.4.4 Enumerated Datapoint Types

3.4.4.1 Grounds and Usage Domain

Enumerated Datapoint Types are used for representing Datapoints, which can only have a limited number of values.

3.4.4.1.1 Sample Application

A heating installation may need a setting by man or a central controller information about the operation mode it is to work in, for instance "Day", "Night", "Economy", "Alarm".

3.4.4.2 Definition and Conditions

3.4.4.2.1 Definition

Enumerated Datapoint Types encode Datapoints which only have a very limited range of possible values between which no order can be defined.

3.4.4.2.2 Conditions

Enumerated Datapoint Types shall not be used for encoding values, which have a clear and unambiguous hierarchical structure.

3.4.4.3 Coding

See clause 3.4.5.1 "Coding".

The Functional Block receiving an enumerated Datapoint Type shall verify the range. Illegal values shall not lead to malfunction of the device.

3.4.4.4 Consequences

Not defined.

3.4.4.5 Implementation

An enumerated data type may as a field be part of a Status Datapoint Type, as explained in clause 3.4.7 "Status Information".

EXAMPLE The encoding of the day of the week in the KNX Datapoint Type "-DPT_Date".

3.4.5 Structured Datapoint Types

Structured Datapoint Types are composed of more than one field.

Only those fields shall be grouped together which are necessary for the interpretation of one or more of the other fields or which can not be interpreted separately.

EXAMPLE The KNX Datapoint Type "DPT_Control_Dimming", the C-bit field ("Control") indicates whether the value encoded in the VVV-bits field shall be decremented or incremented.

3.4.5.1 Coding

A data type shall be chosen with the appropriate length. Unused bits needed to fill to an octet boundary shall be at the most significant positions and set to zero.

The larger fields shall be grouped at the less significant bit positions, the smaller fields (single bits) at the more significant positions.

3.4.6 Multi-State Datapoint Types

3.4.6.1 Grounds and Usage Domain

Multi-state Datapoint Types are intended for transmitting data

- of which the encodable values follow a hierarchical sequence and
- of which all encodable values are meaningful

3.4.6.1.1 Sample Application

A fan-controller can drive the fan from standstill over 5 positions up to a maximum speed.

3.4.6.2 Definition and Conditions

3.4.6.2.1 Definition

The sender maps the output value it wants to transmit on a range between 0 and 255 as follows:

$$value_transmitted = \frac{S_x}{N_s} \times 255$$

S_x = sender step to be transmitted

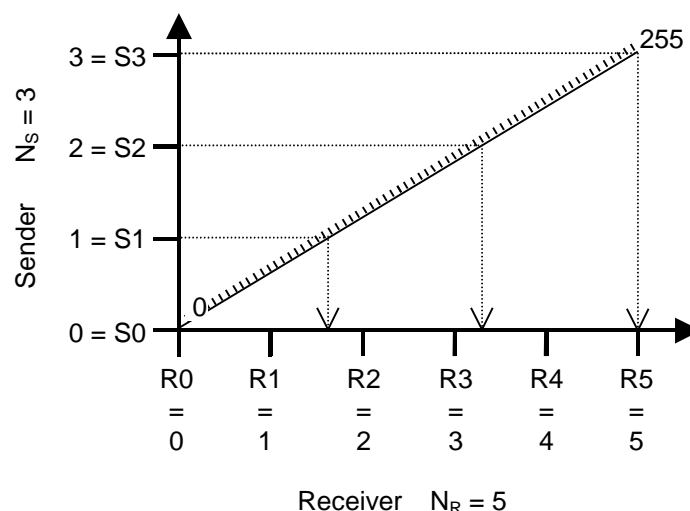
N_s = number of steps in the sender

This value is rounded to a 1-octet unsigned integer and transmitted on the bus. The receiver maps the received value to the range it can handle as follows:

$$set_value = \frac{received_value \times N_R}{255}$$

N_R = number of steps in the receiver

Example: A sender with 3 steps ($N_s=3$) controls a receiver with 5 steps ($N_R=5$). Step 2 ($S_x=2$) of the sender is thus transmitted as $\frac{2}{3} \times 255 = 170$. The receiver interprets this as $\frac{170 \times 5}{255} = 3$ and moves to its level 3.



The Datapoint Type to be used shall be DPT_Scaling (see [05]).

3.4.6.2.2 Conditions

1. This Multi-state is only usable if there is a hierarchic relation between the states.
2. No conditions are defined concerning the linearity of
 - the conversion of the physical or measured value to the value Ax in the sensor
 - the conversion of the value Bx to the physical output value the actuator

3.4.6.3 Coding

Please refer to the definition.

3.4.6.4 Consequences

This mapping and coding allows an N-fold sensor to control an M-fold actuator allowing:

- the off-state of the sensor to be interpreted as off by the actuator
- the full range value of the sensor to set the actuator to its full range even if it has a smaller range

3.4.6.5 Implementation

It is recommended to implement some hysteresis for both sensor – not to send continuously - and actuator – not to switch state continuously. This hysteresis and timing is not part of the Datapoint Type definition or the mapping. If applicable for the Datapoint, the designer shall add a guideline in the Datapoint description field.

3.4.7 Status Information

3.4.7.1 Grounds and Usage Domain

This clause defines the conditions for defining new Datapoint Types for transmitting device status information.

The status information defined here is intended for both

- a) allowing a device to **transmit** its status to one or more other devices
- b) allowing a device to **set** devices into a certain operation mode

This difference in usage leads to different coding and transmission conditions (see below).

Note: On Field Level communication, the run-time data exchanged between field level devices does not need to include device (sensor-) status information. Solely for the vertical communication to automation level some additional device status information may be required.

3.4.7.2 Definition and Conditions

3.4.7.2.2 Conditions

The status information is either sent:

- spontaneously
- on request

3.4.7.3 Coding

The coding is identical as in clause 3.4.5.1 "Coding".

3.4.7.3.1 Masks

3.4.7.3.1.1 Grounds

Imagine a status setting encoded in a 2- bit field:

| | |
|----|----|
| b1 | b0 |
| B | A |

Sender X sets B to 1 and A to 1. Sender Y subsequently wants to set only A to 0, but has to transmit the whole value, thus sends 0 for B and thus unintentionally overwrites the setting of sender X.

To avoid this situation, the status coding is preceded by mask bits indicating which data fields are valid and shall be evaluated:

| | | | |
|----|----|----|----|
| b3 | b2 | b1 | b0 |
| mB | MA | B | A |

$m\alpha = 0$ field α not to be updated/invalid
 $m\alpha = 1$ field α to be updated/valid

If now sender X sets B to 1 and A to 1, it sends 1111b. If sender Y again only wants to set A to 0 and not touch B, it sends 011x0, where x is don't care.

3.4.7.3.1.2 Usage

The use of mask-bits is

mandatory only when the status information is used to

- 1) set an operation mode and the communication relation is N-to-N or N-to-1.

not mandatory when the status information is used to

- 1) transmit the status of a device or
- 2) set an operation mode in a 1-to-N "Multicast" communication relation
(this means the application foresees only one sender)
- 3) set an operation mode in a 1-to-1 "Multicast" communication relation
(this means the application foresees a private communication between one sender and one receiver)

⇒ The designer shall explicitly specify the usage case in the Functional Block Datapoint description.
The implementer shall include this intended usage in the user guidelines.

3.4.8 Datapoint Type Specification

3.4.8.1 Datapoint Type specification form

Every Datapoint Type shall be specified by filling in the form specified in Figure 14.

| | | | | | | |
|------------------------|---|------------------|---------------|-----------------|----------------|-------------|
| <u>Format:</u> | n octets: N_8 | | | | | |
| <u>octet nr.</u> | 1 | | | | | |
| <u>field names</u> | field1 (MSB) | | ... | field N (LSB) | | |
| <u>encoding</u> | B B B B B B B B | | | V V V V V V V V | | |
| <u>Encoding:</u> | Encoding absolute value $N = [0 \dots 255]$ | | | | | |
| <u>Range:</u> | See below | | | | | |
| <u>Unit:</u> | none | | | | | |
| <u>Resol.:</u> | none | | | | | |
| <u>PDT:</u> | PDT_ENUM8 (alt.: PDT_UNSIGNED_CHAR) | | | | | |
| Datapoint Types | | | | | | |
| <u>ID:</u> | <u>Name:</u> | <u>Encoding:</u> | <u>Range:</u> | <u>Unit:</u> | <u>Resol.:</u> | <u>Use:</u> |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| Field | Description | Encoding | Range | Unit | Resol.: |
|---------|-------------|----------|-------|------|---------|
| field 1 | | | | | |
| ... | | | | | |
| field N | | | | | |

Figure 14 – Datapoint Type specification form

3.4.8.2 Explanation of the field in the DPT specification form

3.4.8.2.1 Format

◆ Length specification

This field shall firstly indicate the total length of the DPT. For DPTs shorter than 8 bit, the precise length shall be given in bit, thereby not counting unused, this is, reserved bits. For DPTs longer than 8 bit, the length shall be expressed in octets, even if there are unused bits within these octets. Possible lengths can thus be 1 bit, 2 bit, 3 bit, 4 bit, 5 bit, 6 bit, 7 bit, 1 octet, 2 octets, 3 octets, ...

◆ Format specification

The next information in the format field shall be a precise indication of the datatype and length of each of the fields that compose Datapoint Type.

- Field datatype specification

The datatype of a field can be one of the options as specified in Table 5.

Table 5 – Datatypes notation styles

| Abbreviation | Datatype and use |
|----------------|---|
| A | Character <u>Use:</u> Textual characters |
| A[n] | String of n characters |
| B | Boolean / Bit set <u>Use:</u> All boolean fields. |
| F | Floating point value <u>Use:</u> All floating point encodings. The fields in the Datapoint Type definitions use the notations E and M. |
| E | Exponent The exponent of the float data. |
| M | Mantissa The mantissa of the float data. |
| N | eNumeration <u>Use:</u> All values that have an enumeration relationship. |
| U | Unsigned value <u>Use:</u> All integer data that can only have positive values. |
| V | Signed value, 2's complement <u>Use:</u> All integer data that can have both positive as well as negative values. |
| Z ₈ | Standardised Status/Command B8. <u>Use:</u> Status/Command indication for HVAC DPTs, as specified for the HVAC DPTs in [05]. |

- Field length specification

The length of the field shall be expressed in bits, placed in subscript next to the datatype notation.

- Examples

- B₁ A single bit boolean field.
- N₃ A three bit enumerated field.
- U₁₆ A 16 bit unsigned integer field.

3.4.8.2.2 Octet nr

The next row in the Datapoint Type specification form shall be the indication of the octet number. The least significant octet shall be numbered 1. The Datapoint Type specification shall specify the fields/octets of the Datapoint Type starting with the higher significant fields/octets.

NOTE This is also the order in which the fields/octets shall be transmitted on the bus.

3.4.8.2.3 Field names

The next row identifies each of the composing fields with a short, unique name.

Though it is not recommended, a field may cross octet boundaries.

EXAMPLE The following excerpt of the definition of DPT_Version shows fields that cross the octet boundaries.

| | | | | | | | | | | | | | | |
|----------------|--|---|---|----------------|---|---|-----------------|------------------|---|---|---|---|---|---|
| <u>Format:</u> | 2 octets: U ₅ U ₅ U ₆ | | | | | | | | | | | | | |
| octet nr. | 2 _{MSB} | | | | | | | 1 _{LSB} | | | | | | |
| field names | Magic Number | | | Version Number | | | Revision Number | | | | | | | |
| encoding | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

Reserved fields need not to be given a name.

3.4.8.2.4 Encoding

The field is encoded according the format of one of the datatypes as listed above.

For numerical values, the field value is simply the binary encoding of the numerical value of the coded data. In this case, the form mentions here “Value binary encoded”

EXAMPLE In the Datapoint Type DPT_Percent_U8 (5.004), the numerical value of 35 % is transmitted as 100011b.

There may be a more complex relationship between the field value and the coded value. This is the case in the following situations:

◆ Enumerated Datapoint Types

In this case all allowed values are to be listed, with their respective meaning. Values that can be encoded but cannot be allowed shall be indicated as “Not allowed.”

◆ Resolutions

EXAMPLE In DPT_TimePeriod10MSec (7.003), a time period of 5 sec, this is 5 000 msec, is transmitted as a numerical value 500, unit ms and resolution 10 ms.

In case of numerical values with a unit, where the resolution is not equal to the unit, this part of the DPT specification shall indicate this factor.

◆ Linear mappings

EXAMPLE In DPT_Scaling (5.001) a percentage value of 20 % is multiplied by $\frac{255}{100}$, this is 51.

The field value is thus 33h (110011b).

3.4.8.2.5 Range, Unit and Resolution

This information shall be specified for each field of the Datapoint Type. One or more of these fields can be specified:

- in the Datapoint Type table header (indicated as (A) in Figure 14),

Then, this specification is valid for all Datapoint Types specified under “Datapoint Types” in the table.

EXAMPLE All 2 octet float Datapoint Types (F₁₆), specified in clause 3.10 in [05], have the same value range -671 088,64 ... 670 760,96. They however differ in unit and resolution.

- or in the DPT definition in the lower part of the DPT table (indicated as (B) in Figure 14),
- or in a dedicated table underneath the DPT table (indicated as (C) in Figure 14).

This last option is most convenient for DPTs that consist of multiple fields.

◆ Unit

In order to avoid confusion, there shall be a unit indication for the range and the resolution as well.

EXAMPLE 1 The following excerpt of the definition of DPT_TimePeriodMin

| <u>ID:</u> | <u>Name:</u> | <u>Range:</u> | <u>Unit:</u> | <u>Resol.:</u> | <u>Use:</u> |
|------------|-------------------|--------------------------------|--------------|----------------|-------------|
| 7.006 | DPT_TimePeriodMin | 0 min ... 65535 min (≅ 45,5 d) | min | 1 min | G |

It is recommended to use SI-units, or take over units from already existing KNX DPTs.

◆ Range

This shall be the indication of value range that can be encoded, not the binary data range that can be supported by the format.

EXAMPLE 2 Specify 0 % to 100 % and not 0 ... FFh.

EXAMPLE 3 Specify -327,68 s ... 327,67 s and not -32 768 ... 32 768.

| <u>ID:</u> | <u>Name:</u> | <u>Range:</u> | <u>Unit:</u> | <u>Resol.:</u> | <u>Use:</u> |
|------------|----------------------|--------------------------|--------------|----------------|-------------|
| 8.002 | DPT_DeltaTimeMsec | -32 768 ms ... 32 767 ms | ms | 1 ms | G |
| 8.004 | DPT_DeltaTime100Msec | -3 276,8 s ... 3 276,7 s | ms | 100 ms | G |

Both above DPT are encoded as V_{16} and have ms as unit. For DPT_DeltaTime100Msec, the value to be encoded shall be expressed in ms and divided by 100. This makes that DPT_DeltaTime100Msec has a range 100 times larger than DPT_DeltaTimeMsec, but has a resolution that is 100 times larger as well.

EXAMPLE 4 DPT_Scaling and DPT_Percent_U8 are both unsigned 1 octet formats. The data range is thus 0 ... 255 for both DPTs.

| <u>ID:</u> | <u>Name:</u> | <u>Range:</u> | <u>Unit:</u> | <u>Resol.:</u> | <u>PDT:</u> | <u>Use:</u> |
|------------|----------------|-----------------|--------------|----------------|--|-------------|
| 5.001 | DPT_Scaling | [0 % ... 100 %] | % | ≈ 0,4 % | PDT_SCALING (alt.: PDT_UNSIGNED_CHAR) | G |
| 5.004 | DPT_Percent_U8 | [0 %...255 %] | % | 1 % | PDT_UNSIGNED_CHAR | FB |

In both cases, the data range is 0 to 255 and both DPTs have % as unit. However, due to the difference in encoding, DPT_Scaling supports a value range from 0 % to 100 %, whereas DPT_Percent_U8 supports a value range of 0 % to 255 %.

◆ Resolution

The resolution shall specify the accuracy that can be achieved for encoding the value.

3.4.8.2.6 Property Datatype (PDT)

This field shall specify the Property Datatype that shall be used in case the DPT is used for encoding the value of a Property of an Interface Object.

Some implementations of the Interface Object Server do not support the full range of Property Datatype from 00h to 3Fh. Therefore, if a PDT above 1Fh is specified, the DPT definition shall additionally specify the alternative DPT that shall be used for such implementations, as specified in [06].

3.4.8.2.7 Use

This field specifies the allowed use for this DPT. The following optional are possible:

| Indication | Allowed use |
|------------------|--|
| G | The DPT can be used without limitation – of course limited to the information values for which it is specified – thus also for runtime communication and for Group Objects. |
| FB | The DPT shall not be used in any other context than in the FB in which it is used. Use of this DPT in implementations that do not comply with this FB is regarded as a non-standardised DPT, subject to approval by the KNX Certification Department, as specified in 2.1.7.2.1.2. |
| HVAC, TU, ... | The DPT can only be used in the Application Domain(s) and Functional Block(s) that are indicated by these abbreviations. This indication is identical to the above indication “FB” but moreover gives a helpful reference to the application where the DPT is used. These abbreviations can be found in the specifications of the applications in [09]. |

3.4.8.2.8 Extended DPT descriptions

This standard DPT definition may be followed by further detailed, textual specifications if necessary.

4 General Channel design and implementation rules

4.1 Introduction

The following considerations shall lead to a proper outlining and understanding of the proposal by the designers when drawing up new Interworking proposals.

1. Describe the Application Domain
2. Inputs for channel specification
3. Describe the Channel(s)
4. Describe the Parameter Types

4.2 Describe the Application Domain

The document [09] of these KNX Specifications lists all standard KNX Application models that can be implemented. An Application Domain groups related Applications in a one - or more level structure, depending on the complexity of the Application Domain.

The application designer shall document the Application Domain and its substructure for which the Application Model is intended.

4.3 Steps in the design of a channel

To design a channel, the following steps are followed.

1. Select the FB or combination of FBs that are needed to accomplish the intended channel functionality and describe that functionality (with reference to FB description).
2. Use the mandatory Datapoints from the selected FBs and select additional optional DPs from these FBs that are needed to accomplish the intended channel functionality.
3. Sort the Datapoints to have Input Datapoints preceding Output Datapoints.
4. In case the FB specifies a bidirectional Datapoint, create a separate Input and Output.

NOTE This should be a rare exception. Clauses 2.2 and 3.3.2.1.6.6.1 strongly advise against the use of bidirectional Datapoints.
5. Assign the Connection Codes according to the existing ones or the Datapoint Types. (to be developed).
6. Set the E-Mode flags of the Datapoints according the connection rules.
7. Select the parameters from the FBs.
8. For PB-Mode with optimized parameters, build a new parameter type that is a combination of existing FB parameters may optionally be drafted.
9. Set the parameter bit offset, taking care of the alignment.
10. Give the newly designed channel a new name - which should reveal the functionality - that shall start with "CH_".

EXAMPLE CH_Light_Setpoint_Controller_Info

11. Select the classification.

4.4 Describe the Channel

Every channel is described according the following layout.

- **Name:** The chosen channel name shall be given here. It shall with “CH_”.
- **ID:** The newly chosen Channel Code shall be given here.
- **Classification:** Here, the classification of the channel shall be given, according one of the following classes:
 - sensor, or
 - actuator, or
 - controller.

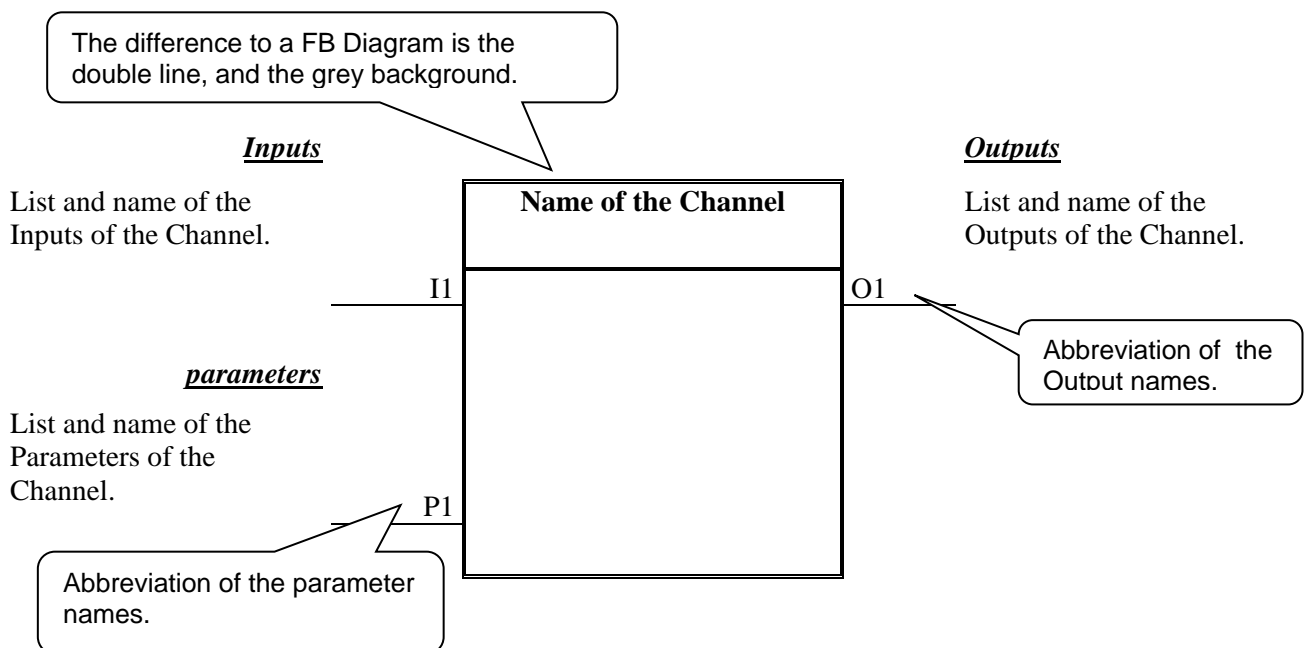
Please refer to clause 4.5 “Channel Code Rules”.

- **Functional Blocks:** Here, the list of Functional Blocks used to achieve the channel functionality shall be given. This list relates the channel to one or more FBs. For each FB, its Interface Object Type and its name shall be given.

EXAMPLE

- 384 - User HVAC Room Settings (UHRS)
- 321 – Room Temperature Sensor (RTS)

- **Graphical representation:**



- **Datapoint list**

List of the Datapoints used by the Channel. (In case of grouping Functional Blocks the internal Datapoints should not be present in this list).

| Index | FB Datapoint ID | Name | Sub-Unit | Main CC | Additional CCs | Flags (i/o,x,v, ...) |
|-------|-----------------|------|----------|---------|----------------|----------------------|
| | | | | | | |

- Index:** Index of the Datapoint in the Channel. According the connection rules, the Datapoints will be connected in the order of the index.
- FB / DP Name:** This shall identify the Datapoint by referring to its name or abbreviation from the Functional Block description.
- Name:** This shall be the name of Datapoint.
- Sub-Unit:** When having a channel with several Datapoints with the same Connection Code, there is a need to designate one given Datapoint when establishing the links. A Sub-Unit defines one set of such Datapoints.
- EXAMPLE A channel with a logical OR functionality will have the same Input and Output Connection Code and thus needs to have the possibility to connect the "input part" and the "output part" separately.
- Main CC:** Here, the Connection Code of the Datapoint shall be filled in. This value will be used with the connection rules during the link procedure.
- Additional CCs:** Here, "compatible" connections codes can be listed. At least the data format shall be the same.
- Flags:** Here, the E-Mode flags for the Datapoint shall be specified. These will be used while applying the connection rules during the link procedure.

Specification of E-Mode Datapoint flags:

| Flag | Meaning | Description |
|------|--|---|
| V | Visualisation Datapoint | This Datapoint shall be connected to one and only one Output Datapoint and one or more Input Datapoints (1-to-N connection). |
| X | Datapoints connectable only once | This Datapoint shall have exactly one connection. Only one Group Address shall be assigned to this Datapoint (1-to-1 connection). |
| XC | Datapoint connectable only once with an information if it is connected. Need a specific parameter "used input bitstream". | Only one Group Address shall be assigned to this Datapoint (1-to-1 connection). If the Datapoint is connected, the corresponding bit in the parameter "used input bitstream" shall be set. The application shall know the connected Datapoints via this parameter. |
| O | Output Datapoint | The value of this Datapoint shall be sent. |
| I | Input Datapoint | The value of this Datapoint shall be received. |
| L | Localisation Datapoint | This Datapoint shall be used for the localisation procedure in the Ctrl-Mode. |
| T | Adaptive Datapoint | For this Datapoint, only the format is specified (1 bit, 8 bit, 16 bit) and this can be adapted during the calculation of the link. This requires an additional definition in the subblock list to know the possibilities of adaptation. |
| LA | Localisation Actuator | This Datapoint shall be used for the optional localisation procedure for actuators in the Ctrl-Mode. |

- **Parameter table**

This table shall list the parameters that shall be implemented for this Channel. The important fields are the index for access via Properties (Properties with PID 101 to 132 of Device Object 9) and the bit offset for calculation of the parameter block.

| Index | FB Parameter ID | Name | Recommended default value | Bit-Offset |
|-------|-----------------|------|---------------------------|------------|
| | | | | |

Index: The cell shall specify the index of the Parameter in the channel.

FB Parameter ID: This identifier shall indicate the corresponding Parameter of the used FB(s).

Name: This shall be the name of the Parameter.

Recommended default value: This is the recommended default value for this Parameter. A manufacturer may choose another value in an implementation.

Bit offset: This shall indicate the position of the parameter in the parameter block of the channel. The value is from 0 to ...

| Channel block | Octet 0 | | | | | | | | Octet 1 | | | | | | | |
|---------------|---------|---|--------------------------|---|---|---|---|---|--------------------|---|----|----|----|----|----|----|
| Bit offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Parameter | | | 03 timer duration | | | | | | 02 On delay | | | | | | | |

- **Sub-Block list**

This list shall only be given if there are Sub-Blocks in the Channel. It shall list all Sub-Blocks. This list shall be used in case there are “adaptive” Datapoints. This list shall contain the possibilities of adaptation of the Datapoint put in “associated input Datapoint” or “associated output Datapoint”.

| Sub-Block index | Connection Code | Corresponding Parameter index | Adapting Parameter value | Input Datapoint | Output Datapoint |
|-----------------|-----------------|-------------------------------|--------------------------|-----------------|------------------|
| | | | | | |

⁹⁾ These can be realised by using Reduced Interface Objects, as specified in [04].

| | |
|--|--|
| <u>Sub-block index:</u> | Index of the Sub-Block in the Channel. |
| <u>Connection Code:</u> | Connection Code associated to the Datapoint index of the Sub-Block in the channel |
| <u>Associated Input Datapoint:</u> | Index of the Input Datapoint (really implemented) of the Sub-Block. |
| <u>Associated Output Datapoint:</u> | Index of the Output Datapoint (really implemented) of the Sub-Block. |
| <u>Corresponding parameter index:</u> | Index of the parameter used for storage of the "adaptation of the Sub-Block". |
| <u>Adapting parameter Value:</u> | indicates the value to put in the parameter selection for this block when the adaptation result corresponds to this line of the table. |

4.5 Channel Code Rules

4.5.1 General classification

Channel Codes are delivered by the KNX Association Working Group Interworking.

| Channel Code range | Classification / Domain Application |
|--------------------|-------------------------------------|
| 400h to 43Fh | Binary Switch On/Off |
| 440h to 47Fh | Dimming Actuator |
| 480h to 4BFh | Shutter |
| 4C0h to 4FFh | Anti-Intrusion |
| 500h to 57Fh | Heating |
| 120h to 1FFh | Other |

4.5.2 Old ranges - for information

| Channel Code range | Classification |
|--------------------|-------------------|
| 000h to 0FFh | sensors |
| 100h to 1FFh | actuators |
| 200h to 2FFh | functional module |
| 300h to 3FFh | generic channels |

4.6 Private channels

There will always be the situation where a manufacturer needs an additional channel type (and code).

In this situation a manufacturer shall bring his proposal to KNX Association Working Group Interworking and ask for standardisation.

While this process is ongoing, the manufacturer should have the possibility to already use the function.

Additionally, if the channel is not of common interest, it may not be standardised. Also in this case, the manufacturer should have the possibility to use the function.

To avoid collisions or malfunction between such channels of different manufacturers a very small range of Channel Codes should be reserved. Channel Codes in this range can only be interpreted together with the manufacturer-ID. (In start link for PB-Mode and in DD2 for other E-Modes).

Just as for Interface Object Types and Property Identifiers the standard may provide a range of Channel Codes for private channels.

This range for private Channel Codes shall be located at the end of the Channel Code range: 1F00h to 1FEFh.

4.7 Manufacturer specific Connection Codes

The same considerations lead to the result, that for Connection Codes a similar rule should be introduced.

The Connection Code range will be: F0h to FFh.

General remark:

Even if such a range for manufacturer specific Channel Codes or Connection Codes exists, the same rules as for Datapoint Types apply, this is, existing Connection Codes, Datapoint Types... shall be used where possible.

4.8 System initialisation

After power-up or reset, a device with an output Datapoint of which it is marked that its value shall be sent on initialisation, shall send its value within 3 minutes after power-up.

Recommendation: For starting time, use the Device Address (8 lower bits of the Individual Address) divided by 2 in seconds, or use a random value.

Annex A

(informative)

Examples of Functional Block Diagram presentation styles

A.1 Example 1

