Interworking

KNX Association

Table of Contents

1 Introduction

Already under the predecessor system to KNX, i.e. EIB, the Association not only took care of the standardisation of the protocol but also laid down rules for the coding of the useful data inside telegrams.
Without standardisation of this aspect, devices of different manufacturers would be able to talk KNX but could still code the useful data contained in the tail of the telegram differently, e.g. manufacturer A would code temperatures as 1 byte and another as 3 bytes.
For a start such objects could then not be linked by means of ETS and secondly, these devices would not understand one another.
In order to ensure interworking between manufacturers and even products of different application domains, EIBA therefore laid down formats for common functions like switching, dimming, blinds control, integer and float values, percentage, date/time, HVAC modes, scene control, …. At the time of EIBA, these formats were referred to as EIS (EIB Interworking Standards).
When KNX came into being, the EIS were renamed into KNX standardized Data types. The most common data types were also standardized on a European level and integrated into the EN 50090 series as Part 3-3.
If a standardized format for a certain function exists, for certification the KNX manufacturer is obliged to use this format. Compliance to the format is also checked during the KNX interworking tests as carried out by KNX accredited third party test labs.

2 Advantages of Interworking

The benefits of Home and Building Control only become truly visible when devices of different manufacturers and different application domains interwork:

- Presence detector is part of the alarm system at night
- Room thermostat of Manufacturer A sets position of valves of Manufacturer B
- “All off” button of Manufacturer A switches the lights off, controlled by switching actuators of Manufacturer B, C, D, …;
- Scheduler of Manufacturer A ensures presence simulation, thereby controlling blinds of Manufacturer B

It goes without saying that this is an enormous benefit to end users. When a product line A is defective but Manufacturer A has discontinued product line, the end user can find a replacement at Manufacturer B.
A system with true interworking also attracts manufacturers of niche products, as one single manufacturer can simply not offer all possible HBES solutions, from lighting to HVAC to Load management, etc.
This in turn boosts the OEM market: what Manufacturer A does not produce himself, he can easily find at another HBES manufacturer and complete his offer by relabeling the products he buys from Manufacturer A.
Gateway solutions between KNX and proprietary or other standardized systems (e.g. DALI, BACnet, …) are easier to develop, as the proprietary coding can be easily mapped to common KNX data formats as described in the KNX standard.

Last but not least, it would have been impossible for KNX to establish the common market infrastructure that exists today:

- the fact that all products of all manufacturers can be linked to one working installation is the corner stone of the KNX one configuration tool (ETS) approach.
- It would have been impossible to establish the common training scheme for the education of persons interested in using the KNX technology in home or commercial projects. The training scheme of basic, advanced and tutor courses is worldwide standardized. In contrast to that, manufacturers of proprietary systems must establish each individually their own training schemes and users of such material must visit several courses if they wish to combine material of several manufacturers into an installation.

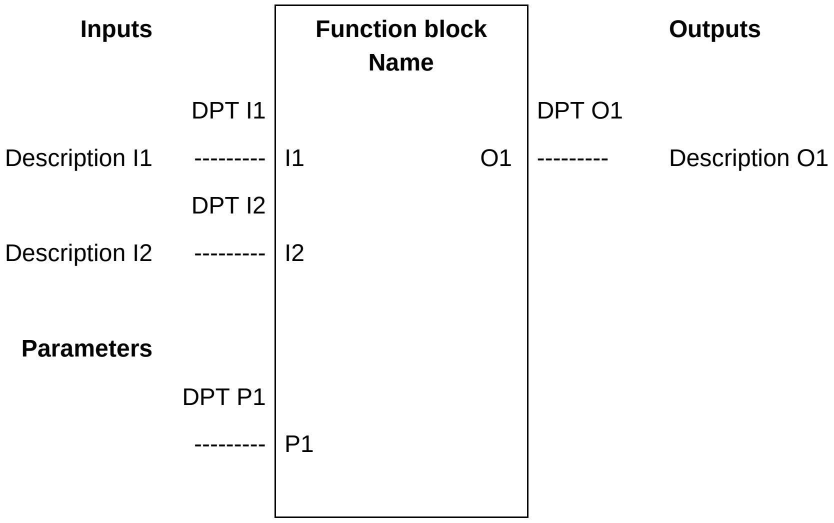
3 Principles of KNX Interworking

3.1 Introduction

For KNX devices that are programmed with the ETS, KNX requires that at least the group objects are coded according to KNX standardized data types. The coding of parameters that are described in the products' database descriptions can be manufacturer specific.

For some of the device types (e.g. like dimming, blind control or switching with priority), it is however necessary that actuators show a certain behaviour when data is sent to the available group objects. In this case, the specification of functional blocks becomes necessary and needs to be complied with during KNX product certification.

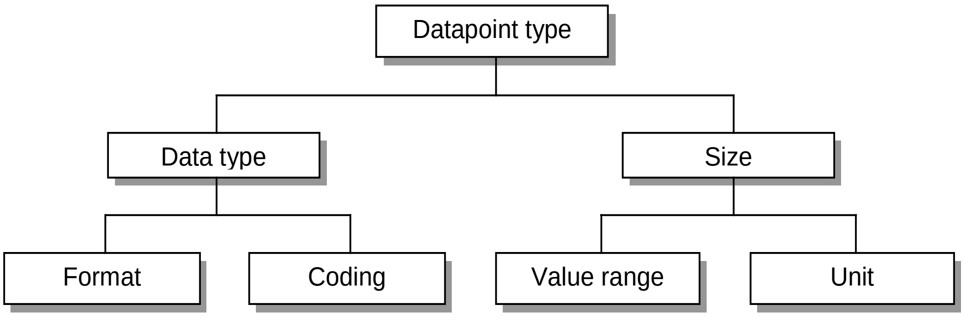
Functional blocks group a number of inputs, outputs and parameters. For this combination, a precise function description is provided.



Especially when devices support Easy installation, a clear definition of the description of each channel in the device in the form of functional blocks is necessary.

3.2 Coding of Datapoint types

3.2.1 Introduction



A group object is regarded as a certain type of datapoint. Also parameters can be regarded as datapoints.
The datapoint type is determined by the following four elements

1. Format: Which fields is the datapoint type composed of? Each field can consist of one or several bits.
2. Coding: How is the data coded?
3. Value range: Are there limitations to the value range? There may be different minimum and maximum values or valid values for each field or lists of valid values.
4. Unit: Which units does the data of the individual fields have?

3.2.2 Principle datapoint types classes

3.2.2.1 Introduction

Five classes can be distinguished according to their internal structure. More explanation is given in the next clauses.

3.2.2.2 Simple Types

Examples of simple types are: boolean values, pure numerical values.

3.2.2.3 Enumerated Types

Enumerations are used for datapoints with a clearly limited number of values and no precise hierarchical order. If a datapoint uses an enumerated data type, all possible states shall be described. If invalid / undefined values are sent, no malfunctions may occur.
Examples are: Toggling between “Comfort”, “Standby” and “Night” operating modes in room thermostats.

3.2.3 Structured Types

Structured datapoint types consist of several parts which are combined in a datapoint. Only fields which cannot be interpreted meaningfully on their own can be combined to a structured datapoint type.
Examples are : KNX relative dimming (DPT_Control_Dimming)

3.2.4 “Multi-state” Types

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

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

3.2.4.1 Definition

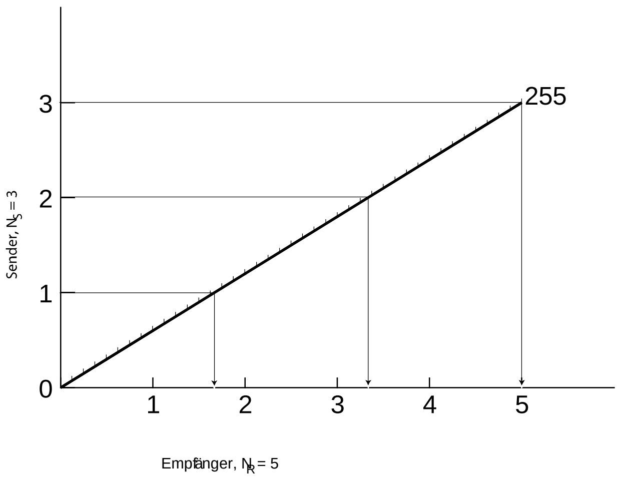
To be able to cover all cases in which transmitters and receivers operate with many different steps, they both convert their steps to a range between 0 and 255.
The following applies for the transmitter:

$$Value = \frac{required\ step}{number\ of\ steps} \cdot 255$$

This value is rounded up or down and sent as a byte without a sign.

The following applies for the receiver:

$$Step = \frac{received\ value \cdot number\ of\ steps}{255}$$



These datapoint types can only be used if

- the states have a clearly defined sequence,
- no exact linear conversion of the steps between transmitter and receiver is required

3.2.5 Status Types

Datapoint types for transmitting status information should meet two goals:

1. A device should be able to report its operating mode to other devices.
2. A device should be able to be switched to a specific operating mode.

To achieve this, further bits are sent as a mask in addition to the actual status information. The mask determines whether the data of the status fields should be used as valid commands or only for information purposes.
These datapoint types are primarily used in the area of heating technology.

4 KNX common datapoint Types

4.1 Introduction

This section lists some common KNX datapoint types.
All datapoint types have a unique identification consisting of two digits. The first digit denotes the format and the coding. The second digit represents the value range and the unit.
In the course of technical development, KNX members may submit suggestions for new datapoint types to the KNX Association. New datapoint types are also drawn up by the responsible KNX working group 'WGI' (Working Group Interworking) when elaborating new application descriptions and corresponding KNX standardised functional blocks.
Group objects complying to DPT types with a size of 6 bits or less only use the bits directly following the APCI (any unused bits are set to zero) in the KNX telegram. Any group objects complying to DPTs with a size greater than 6 bits will result in telegrams, where the 6 bits following the APCI are set to zero and the useful data padded after these unused bits.

4.2 Boolean Data

4.2.1 General

DPT ID 1.xxx is used for all possible applications in which two different values are to be set or sent.

4.2.2 Datapoint Types B_1

Format:	1 bit: B_1
octet nr	1
field names	| | | | | | | |b|
encoding	| | | | | | | |B|
Range:	b = {0,1}
Unit:	None.
Resol.:	(not applicable)

Datapoint Types

ID:	Name:	Encoding: b
1.001	DPT_Switch	0 = Off 1 = On
1.002	DPT_Bool	0 = False 1 = True
1.003	DPT_Enable	0 = Disable 1 = Enable
1.004	DPT_Ramp	0 = No ramp 1 = Ramp
1.005	DPT_Alarm	0 = No alarm 1 = Alarm
1.006	DPT_BinaryValue	0 = Low 1 = High
1.007	DPT_Step	0 = Decrease 1 = Increase
1.008	DPT_UpDown	0 = Up 1 = Down
1.009	DPT_OpenClose	0 = Open 1 = Close

Format:	1 bit: B_1
octet nr	1
field names	b
encoding	B
Range:	b = {0,1}
Unit:	None.
Resol.:	(not applicable)

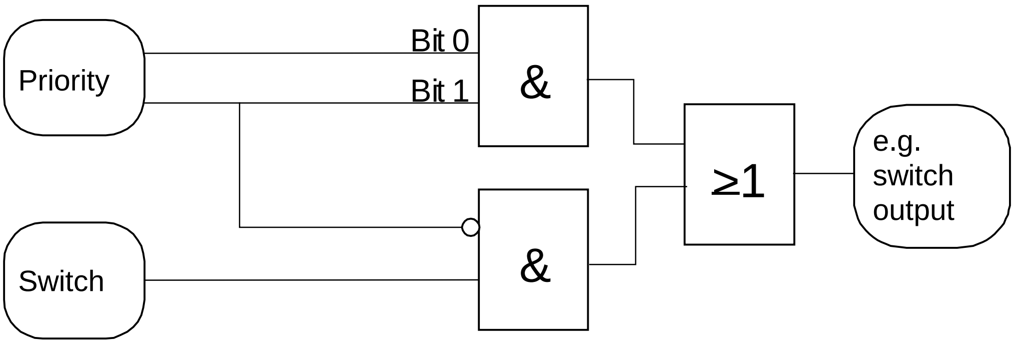
Datapoint Types

ID:	Name:	Encoding: b
1.010	DPT_Start	0 = Stop 1 = Start
1.011	DPT_State	0 = Inactive 1 = Active
1.012	DPT_Invert	0 = Not inverted 1 = Inverted
1.015	DPT_Reset	0 = no action (dummy) 1 = reset command (trigger)
1.016	DPT_Ack	0 = no action (dummy) 1 = acknowledge command (trigger), e.g. for alarming
1.017	DPT_Trigger	0, 1 = trigger
1.018	DPT_Occupancy	0 = not occupied 1 = occupied
1.019	DPT_Window_Door	0 = closed 1 = open
1.021	DPT_LogicalFunction	0 = logical function OR 1 = logical function AND
1.022	DPT_Scene_AB	0 = scene A 1 = scene B

4.3 1 Bit with priority control

4.3.1 General

These datapoint types are intended for applications offering a priority control that has precedence over the normal (manual) operation. In other words, if an actuator is switched via its priority datapoint, switching via its usual DPT 1.xxx group objects is disabled. Switching with priority is ensured via 2-bit datapoint types.
The interaction between the 2 and the 1 bit datapoint functions according to the underneath diagram.



4.3.2 Datapoint Types B_2

Format:	2 bit: B_2
octet nr	1
field names	c V
encoding	B B
Range:	c = {0,1}
	v = {0,1}
Unit:	None
Resol.:	(not applicable)

Datapoint Types

ID:	Name:	Encoding:	
		c	v
		0 = no control 1 = control	According to Type 1.xxx
2.001	DPT_Switch_Control		
2.002	DPT_Bool_Control		
2.003	DPT_Enable_Control		
2.004	DPT_Ramp_Control		
2.005	DPT_Alarm_Control		
2.006	DPT_BinaryValue_Control		
2.007	DPT_Step_Control		
2.010	DPT_Start_Control		
2.011	DPT_State_Control		
2.012	DPT_Invert_Control		

c	v	
0	0	No control
0	1	No control
1	0	Control. Function value 0
1	1	Control. Function value 1

<table>
<tr><td rowspan="3"><u>Format</u></td><td colspan="4">2 bit</td></tr>
<tr><td colspan="4">1</td></tr>
<tr><td colspan="4">C V</td></tr>
<tr><td><u>Coding</u></td><td colspan="4">See below</td></tr>
<tr><td><u>Range</u></td><td colspan="4">C = {0,1}
V = {0,1}</td></tr>
<tr><td><u>Unit</u></td><td colspan="4">-</td></tr>
<tr><td colspan="5">Datapoint types</td></tr>
<tr><td><u>ID:</u></td><td><u>Name:</u></td><td colspan="3"><u>Coding:</u></td></tr>
<tr><td rowspan="5"></td><td rowspan="5"></td><td>C</td><td>V</td><td></td></tr>
<tr><td>0</td><td>0</td><td>Without priority control</td></tr>
<tr><td>0</td><td>1</td><td>Without priority control</td></tr>
<tr><td>1</td><td>0</td><td>With priority control function corresp. V = 0</td></tr>
<tr><td>1</td><td>1</td><td>With priority control function corresp. V = 1</td></tr>
<tr><td>2.001</td><td>DPT_Switch_Control</td><td colspan="3"></td></tr>
</table>

4.4 3 Bit with Control

4.4.1 General

These datapoint types are amongst others used to realize relative dimming or moving blinds relative to a certain start position.

4.4.2 Datapoint DPT_Control_Dimming

Format:	4 bit: B_1U_3
octet nr	1
field names	c Step-Code
encoding	B U U U
Range:	c = {0,1}
	StepCode = [000b…111b]
Unit:	None
Resol.:	(not applicable)
Datapoint Types	
ID:	Name:
3.007	DPT_Control_Dimming

Data fields	Description	Encoding
c	Increase or decrease the brightness.	See 1.007 0 = Decrease 1 = Increase
StepCode	The amount of intervals into which the range of 0 % … 100 % is subdivided, or the break indication.	001b…111b: Step Number of intervals = $2^{(stepcode-1)}$ 000b: Break

4.4.3 Datapoint DPT_Control_Blinds

Format:	4 bit: B_1U_3
octet nr	1
field names	c Step-Code
encoding	B U U U
Range:	c = {0,1}
	StepCode = [000b…111b]
Unit:	none
Resol.:	(not applicable)
Datapoint Types	
ID:	Name:
3.008	DPT_Control_Blinds

Data fields	Description	Encoding
c	Move up or down.	See 1.008 0 = Up 1 = Down
StepCode	The amount of intervals into which the range of 0 % … 100 % is subdivided, or the break indication.	001b…111b: Step Number of intervals = $2^{\wedge(stepcode-1)}$ 000b: Break

NOTE This DPT can be used both for the relative positioning of the vertical blinds positions as well as for the relative positioning of the angle of the slats.

4.5 Character Set

4.5.1 General

DPT 4.xxx is defined for transmitting individual (text) characters. The coding sent on the bus corresponds to the coordinates of a look up table, containing the different characters.

4.5.2 Datapoint Types Character Set

Format	8 bit 1 AAAAAAAA
Coding	See below
Range	See below
Unit	-

Datapoint types

ID:	Name:	Range:	Coding:
4.001	DPT_Char_ASCII	[0…127]	See below. msb is always 0
4.002	DPT_Char_8859_1	[0…255]	See below

Coding:

4.001 DPT_Char_ASCII
4.002 DPT_Char_8859_1

AAAA	AAAA
MSN	LSN

LSN = Least Significant Nible

MSN = Most Significant Nible

MSN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LSN																
0	NUL	DLE		0	@	P	`	p				°	À	Ð	à	ð
1	SOH	DC1	!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2	STX	DC2	“	2	B	R	b	r			¢	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
4	EOT	DC4	$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5	ENQ	NAK	%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
6	ACK	SYN	&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
7	BEL	ETB	‘	7	G	W	g	w			§	·	Ç	×	ç	÷
8	BS	CAN	(	8	H	X	h	x			¨	¸	È	Ø	è	ø
9	HT	EM	)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A	LF	SUB	*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B	VT	ESC	+	;	K	[	k	{			«	»	Ë	Û	ë	û
C	FF	FS	,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü
D	CR	GS	-	=	M	]	m	}			-	½	Í	Ý	í	ý
E	SO	RS	.	>	N	^	n	~			®	¾	Î	Þ	î	þ
F	SI	US	/	?	O		o				¯	¿	Ï	ß	ï	ÿ

4.6 8 Bit without Sign

4.6.1 General

DPT 5.xxx is defined for transmitting unsigned values up to 255.

4.6.2 Scaled values

Format:	8 bit: U_8
octet nr	1
field names	Unsigned Value
Encoding	U U U U U U U U
Encoding:	binary encoded

msb lsb

U	U	U	U	U	U	U	U	
0	0	0	0	0	0	0	0	= range min. /off
0	0	0	0	0	0	0	1	= value “low”
⋮				⋮			⋮	
1	1	1	1	1	1	1	1	= range max.

Range: U = [0…255]

Datapoint Types

ID:	Name:	Range:	Unit:	Resol.:
5.001	DPT_Scaling	[0…100]	%	≈ 0,4 %
5.003	DPT_Angle	[0…360]	°	≈ 1,4°
5.004	DPT_Percent_U8	[0…255]	%	1 %

Examples

Datapoint Type	Encoded Value			Resolution
	50 %	100 %	255 %	
5.001	80h	FFh	Out of encodable range.	≈ 0,4 %
5.004	32h	64h	FFh	1 %

4.6.3 Non-Scaled values

Format:	8 bit: U_8			
octet nr	1			
field names	Unsigned Value			
Encoding	U U U U U U U U			
Encoding:	binary encoded			
Range:	UsignedValue = [0…255]			
Datapoint Types				
ID:	Name:	Range:	Unit:	Resol.:
5.010	DPT_Value_1_Ucount	[0…255]	counter pulses	1 counter pulse

4.7 8 Bit with Sign

4.7.1 General

DPT 6.xxx is defined for transmitting values from -128 up to +127.
Negative numbers are represented as two's complement. To do so, the binary representation of the positive number is inverted and 1 is added.

4.7.2 Datapoint Types V_8 - Signed Relative Value

Format:	8 bit			
octet nr	1			
field names	RelSigned Value			
encoding	V V V V V V V V			
Encoding:	Two's complement notation			
Range:	-128 … 127			
Datapoint Types				
ID:	Name:	Range:	Unit:	Resolution
6.001	DPT_Percent_V8	-128 % … 127 %	%	1 %
6.010	DPT_Value_1_Count	-128 … 127	counter pulses	1 counter pulse

4.8 2 Octet without Sign

4.8.1 General

DPT 7.xxx is defined for transmitting values up to 65535.

4.8.2 2-octet unsigned counter value

Format:	2 octets: U_{16}	
octet nr	2 MSB	1 LSB
field names	UnsignedValue	
encoding	UUUUUUUU	UUUUUUUU
Encoding:	Binary encoded value	
Range:	UnsignedValue = [0…65535]	

Datapoint Types

ID:	Name:	Range:	Unit:	Resol.:
7.001	DPT_Value_2_Ucount	[0…65 535]	pulses	1 pulse

4.9 2 Octet with Sign

4.9.1 General

DPT 8.xxx is defined for transmitting values from -32768 up to +32767. As for 1 octet with sign, negative values are transferred as two's complement.

4.9.2 2-octet signed counter value

Format:	2 octet: V_{16}	
octet nr	2 MSB	1 LSB
field names	SignedValue	
encoding	VVVVVVVV	VVVVVVVV
Encoding:	Two's complement notation	
Range:	SignedValue = [-32 768 … 32 768]	

Datapoint Types

ID:	Name:	Range:	Unit:	Resol.:
8.001	DPT_Value_2_Count	[-32 768 … 32 767] a)	pulses	1 pulse
8.010	DPT_Percent_V16	-327,68 % ... 327,67 %	%	0,01 %

a) Only for DPT_Value_2_Ucount, the value 7FFFh *can* be used to denote invalid data.

b) For DPT_Percent_, the value 7FFFh *shall* be used to denote invalid data.

4.10 2 Octet Floating Point Number

4.10.1 General

DPT 9.xxx is defined for transmitting floating point values. Various datapoint types have been defined for different physical variables.
Not all datapoint types use the maximum value range. Devices shall ignore invalid or undefined values.
The value to be transferred shall be coded in the mantissa. If the value multiplied by 100 (because of the resolution of 0,01) does not fit in the range of –2048 and +2047, the mantissa shall be divided by a factor, which constitutes the exponent. The sign bit indicates whether the value is a negative (S bit = 1) or a positive value (S bit =0). In case of negative values, the mantissa shall moreover be the two's complement of the corresponding positive value.

4.10.2 Datapoint Types 2-Octet Float Value

Format:	2 octets: F_{16}
octet nr	2 MSB 1 LSB
field names	FloatValue
encoding	M E E E E M M M M M M M M M M M
Encoding:	FloatValue = $(0{,}01 \cdot M) \cdot 2^{(E)}$
	E = [0 … 15]
	M = [-2 048 … 2 047], two's complement notation
	For all Datapoint Types 9.xxx, the encoded value 7FFFh shall always be used to denote invalid data.
Range:	[-671 088,64 … 670 760,96]

Datapoint Types

ID:	Name:	Range:	Unit:	Resol.:
9.001	DPT_Value_Temp	-273 °C … 670 760 °C	°C	1 °C
9.002	DPT_Value_Tempd	-670 760 K … 670 760 K	K	1 K
9.003	DPT_Value_Tempa	-670 760 K/h … 670 760 K/h	K/h	1 K/h
9.004	DPT_Value_Lux	0 Lux … 670 760 Lux	Lux	1 Lux
9.005	DPT_Value_Wsp	0 m/s … 670 760 m/s	m/s	1 m/s
9.006	DPT_Value_Pres	0 Pa … 670 760 Pa	Pa	1 Pa
9.007	DPT_Value_Humidity	0 % … 670 760 %	%	1 %
9.008	DPT_Value_AirQuality	0 ppm … 670 760 ppm	ppm	1 ppm
9.010	DPT_Value_Time1	-670 760 s … 670 760 s	s	1 s
9.011	DPT_Value_Time2	-670 760 ms … 670 760 ms	ms	1 ms
9.020	DPT_Value_Volt	-670 760 mV… 670 760 mV	mV	1 mV
9.021	DPT_Value_Curr	-670 760 mA … 670 760 mA	mA	1 mA
9.022	DPT_PowerDensity	-670 760 W/m^2 … 670 760 W/m^2	W/m^2	1 W/m^2
9.023	DPT_KelvinPerPercent	-670 760 K/% … 670 760 K/%	K/%	1 K/%
9.024	DPT_Power	-670 760 kW … 670 760 kW	kW	1 kW
9.025	DPT_Value_Volume_Flow	-670 760 l/h … 670 760 l/h	l/h	1 l/h

4.10.3 Example

A temperature value of - 30 degrees C can be calculated according DPT 9.001 as follows:

Step 1: Calculate the mantissa

Due to the resolution of 0.01, the value to be coded must be multiplied by 100: 30 x 100 = 3000

Step 2: Check if exponent is required

Mantissa is 11 bits, range is from + 2047 to -2048.
3000 is larger, therefore exponent is required.
Which exponent? $2^1 = 2$ is sufficient as 3000 : 2 = 1500, and this number can be coded in the mantissa.

Step 3: Code the mantissa:

Value:	1024	512	256	128	64	32	16	8	4	2	1
Number:	1	0	1	1	1	0	1	1	1	0	0

If the number is negative, then create a two's complement!

Output value: 101 1101 1100

Invert: 010 0010 0011
+1 1
----
010 0010 0100

Step 4: Code sign and exponent
Number is negative, therefore the S bit = 1
Exponent = 1, coded in four bits = 0001

Step 5: Final result

-30 = 1 0001 010 0010 0100

4.11 Time

4.11.1 General

DPT 10.001 is defined for transmitting the time of the day (e.g. cyclically by a system clock).

4.11.2 Datapoint type Time

Format:	3 octets: $N_3U_5r_2U_6r_2U_6$		
octet nr.	3 MSB	2	1 LSB
field names	Day | Hour	0 | 0 | Minutes	0 | 0 | Seconds
Encoding	N N N U U U U U	r r U U U U U U	r r U U U U U U
Encoding:	binary encoded		

Datpoint Types

ID:	Name:	Field:	Encoding:	Range:	Unit:	Resol.:
10.001	DPT_TimeOfDay	Day	1 = Monday … 7 = Sunday 0 = no day	[0…7]	none	none
		Hour	binary encoded	[0…23]	hours	h
		Minutes	binary encoded	[0…59]	minutes	min
		Seconds	binary encoded	[0…59]	seconds	s

4.12 Date

4.12.1 General

DPT 11.001 is defined for transmitting the date of the day (e.g. cyclically by a system clock). Please note that the day of the week is not transmitted in DPT 11.001.
It shall be noted that values shall be interpreted as follows by a receiver:
Year data ≥ 90 signifies year in the 20th century.
Year data < 90 signifies year in the 21st (this) century.
The coding therefore covers years between 1990 and 2089.
Example:

YYYYYYY = 99_d equals 1999
YYYYYYY = 0_d equals 2000
YYYYYYY = 4_d equals 2004

4.12.2 Datapoint type Date

Format:	3 octets: $r_3U_5r_4U_4r_1U_7$		
octet nr.	3 MSB	2	1 LSB
field names	0 0 0 Day	0 0 0 0 Month	0 Year
Encoding	r r r U U U U U	r r r r U U U U	r U U U U U U U
Encoding:	All values binary encoded.		

Datpoint Types

ID:	Name:	Field:	Range:	Unit:	Resol.:
11.001	DPT_Date	Day	[1…31]	Day of month	1 day
		Month	[1…12]	Month	1 month
		Year	[0…99]	Year	1 year

4.13 Date + Time

DPT 19.001 is defined for transmitting the date and time of the day.
The datapoint type combines and extends the DPT_TimeOfDay (10.001) and DPT_Date (11.001) and has a size of 8 bytes.
In this datapoint type, the year is coded as an unsigned byte and calculated as an offset to the year 1900. The period between 1900 and 2155 is thus covered with this datapoint type.

4.13.1 Datapoint type Time and Date

Format: 8 octets: $U_8[r_4U_4][r_3U_5][U_3U_5][r_2U_6][r_2U_6]B_{16}$

octet nr.	8 MSB	7	6	5
field names	Year	0 0 0 0 Month	0 0 0 DayOfMonth	DayOf-Week HourOfDay
Encoding	U U U U U U U U	r r r r U U U U	r r r U U U U U	U U U U U U U U
octet nr.	4	3	2	1 LSB
field names	0 0 Minutes	0 0 Seconds	F WD NWD NY ND NDoW NT SUTI	CLQ 0 0 0 0 0 0 0
Encoding	r r U U U U U U	r r U U U U U U	B B B B B B B B	B r r r r r r r

Datapoint Types

ID:	Name:
19.001	DPT_DateTime

Field	Description	Encoding	Range	Unit	Resol.:
Year	Year	Value binary encoded, offset 1900 0 = 1900 255 = 2155	[0…255]	year	1 year
Month	Month	Value binary encoded 1 = January … 12 = December	[1…12]	Month	1 month
DayOfMonth	D	Value binary encoded 1 = 1st day 31 = 31st day	[1…31]	none	none
DayOfWeek	Day of week	Value binary encoded 0 = any day 1 = Monday … 7 = Sunday	[0…7]	none	none
HourOfDay	Hour of day	Value binary encoded.	[0…24]	h	1 h
Minutes	Minutes	Value binary encoded.	[0…59]	min	1 min
Seconds	Seconds	Value binary encoded.	[0…59]	s	1 s
F	Fault	0 = Normal (No fault) 1 = Fault	{0,1}	none	none
WD	Working Day	0 = Bank day (No working day) 1 = Working day	{0,1}	none	none
NWD	No WD	0 = WD field valid 1 = WD field not valid	{0,1}	none	none
NY	No Year	0 = Year field valid 1 = Year field not valid	{0,1}	none	none
ND	No Date	0 = Month and Day of Month fields valid 1 = Month and Day of Month fields not valid	{0,1}	none	none
NDOW	No Day of Week	0 = Day of week field valid 1 = Day of week field not valid	{0,1}	none	none
NT	No Time	0 = Hour of day, Minutes and Seconds fields valid 1 = Hour of day, Minutes and Seconds fields not valid	{0,1}	none	none
SUTI	Standard Summer Time	0 = Time = UT+X 1 = Time = UT+X+1	{0,1}	none	none
CLQ	Quality of Clock	0 = clock without ext. sync signal 1 = clock with ext. sync signal	{0,1}	none	none

4.13.2 Comments

4.13.2.1 Year field

The year is encoded on 8 bits instead of on 7 bits as in DPT_Date.

4.13.2.2 Hour field

The encoding of the hour is within the range [0…24] instead of [0…23].
When the hour is set to "24", the values of octet 3 (Minutes) and 2 (Seconds) shall be set to zero. Messages with invalid values ("Hour = 24", Minutes and Seconds not zero) have to be ignored by the receiver.
In this way, it is possible to use this Datapoint Type to encode e.g. schedule programs. In daily schedule programs usually "end of day" is encoded as 24:00:00 and not 23:59:59; otherwise there would be a 1 s "break" at midnight.

Without the value 24:00:00 one can not differentiate between a full 24 h period and a 0 h period.

Examples:

- A daily program with 24 h comfort level is encoded as "start comfort: 00:00:00" and "end of comfort: 24:00:00".
- A daily program with 0 h comfort level (⇒ all day economy level) is encoded as "start comfort: 00:00:00" and "end of comfort: 00:00:00".

4.13.2.3 Fault field

"Fault" is set if one or more supported fields of the Date&Time information are corrupted. This is not the same as when the NY, ND, NW etc. attributes would be set (in this case the corresponding fields are not supported).
"Fault" is set e.g.

- After power-down, if battery backup of the clock was not sufficient
- After 1st start-up of the device (clock unconfigured)
- Radio-clock (DCF 77) had no reception for a very long time

"Fault" is usually cleared automatically by the device (producer) if the local clock is set or clock data is refreshed by other means (e.g. by reception of system clock message, reception of DCF 77 radio message etc.).

The receiver (e.g. a room unit, MMI) will interpret Date&Time with "Fault" as corrupted and will either ignore the message or show --:--:-- or blinking 00:00:00 (as known from Video recorders after power-up).

4.13.2.4 SUTI field

SUTI is only an attribute for information / visualisation. In the hour field, summer-time correction is already considered. Therefore no hour offset shall be added by the receiver if SUTI is set.

- SUTI = 0 standard time
- SUTI = 1 summer daylight saving time

4.13.2.5 NDoW field

- NDoW = 1 means that the “Day of Week”-field ddd is invalid and the ddd information shall be ignored. A Clock not supporting Day of Week information shall set NdoW = 1.
- NDoW = 0 and ddd = 0 means that the ddd-field is valid and that ddd is a wildcard. This encoding feature is thought for use in for instance scheduling information.

4.13.2.6 CLQ field

Bit 7 of the 1st byte is used for “Quality of Clock” bit (CLQ). The other bits of this byte are reserved for future extensions. Their values shall be 0. If this Datapoint Type is used for transmitting data, transmitters shall set the lower 7 bits to 0. Receivers shall check these bits to be 0.

Encoding

- 0: Clock without an external synchronisation signal.
  The device sending date&time information has a local clock, which can be inaccurate !
- 1: Clock with an external synchronisation signal (like DCF77, videotext, etc.).
  The device sending date & time information sends signals which are synchronised (time to time) with external date & time information.

The default value is 0.

Also an externally synchronised clock should send CLQ = 0 after start-up (until reception of first synchronisation signal) or after a synchronisation timeout.

4.14 4 Octet without Sign

4.14.1 General

DPT 12.xxx is defined for transmitting unsigned counter values up to 4294967295.

4.14.2 Datapoint Types 4-Octet Unsigned Value

Format:	4 octets: U_{32}		
octet nr	4 MSB 3 2 1 LSB		
field names	UnsignedValue		
encoding	UUUUUUUU UUUUUUUU UUUUUUUU UUUUUUUU		
Encoding:	Binary encoded		
Range:	UnsignedValue = [0…4 294 967 295]		
PDT	PDT_UNSIGNED_LONG		
Datapoint Types			
ID:	Name:	Unit:	Resol.:
12.001	DPT_Value_4_Ucount	counter pulses	1 pulse

4.15 4 Octet with Sign

4.15.1 General

DPT 13.xxx is defined for transmitting signed counter values from -2147483648 up to +2147483647, where negative values are transmitted as 2's complement.

4.15.2 Datapoint Types 4-Octet Signed Value

Format:	4 octets: V_{32}			
octet nr	4 MSB 3 2 1 LSB			
field names	SignedValue			
encoding	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV			
Encoding:	Two's complement notation			
Range:	SignedValue = [-2 147 483 648 … 2 147 483 647]			
PDT	PDT_LONG			
Datapoint Types				
ID:	Name:	Range:	Unit:	Resol.:
13.001	DPT_Value_4_Count		counter pulses	1 pulse

4.16 4 Octet Floating Point Number

4.16.1 General

DPT 14.xxx is defined for floating point values with greater accuracy. Various datapoint types have been defined depending on the different physical variables.
The IEEE floating point format is used in accordance with IEEE 754 so that

- higher values than for DPT 9.xxx can be transferred,
- compatibility to other systems using this format is ensured.

79 different datapoint types have been defined, of which some are given in the underneath paragraph.

4.16.2 Datapoint Types 4-Octet Float Value

Format:	4 octets: F_{32}			
octet nr.	4 MSB	3	2	1 LSB
field names	S Exponent	Fraction		
encoding	F F F F F F F F	F F F F F F F F	F F F F F F F F	F F F F F F F F
Encoding:	The values are encoded in the IEEE floating point format according IEEE 754.			
Range:	S (Sign) =	{0,1}		
	Exponent =	[0 … 255]		
	Fraction =	[0 … 8 388 607]		

Datpoint Types				
ID:	Name:	Unit:	Resol.:	Comment:
14.007	DPT_Value_AngleDeg	°	1 °	angle, degree
14.019	DPT_Value_Electric_Current	A	1 A	electric current
14.027	DPT_Value_Electric_Potential	V	1 V	electric potential
14.028	DPT_Value_Electric_PotentialDifference	V	1 V	electric potential difference
14.031	DPT_Value_Energy	J	1 J	energy
14.032	DPT_Value_Force	N	1 N	force
14.033	DPT_Value_Frequency	Hz = s^{-1}	1 Hz	frequency
14.036	DPT_Value_Heat_FlowRate	W	1 W	heat flow rate
14.037	DPT_Value_Heat_Quantity	J	1 J	heat, quantity of
14.038	DPT_Value_Impedance	Ω	1 Ω	impedance
14.039	DPT_Value_Length	m	1 m	length
14.051	DPT_Value_Mass	kg	1 kg	mass
14.056	DPT_Value_Power	W	1 W	power
14.065	DPT_Value_Speed	m s^{-1}	1 m s^{-1}	speed
14.066	DPT_Value_Stress	Pa = N m^{-2}	1 Pa	stress
14.067	DPT_Value_Surface_Tension	N m-1	1 N m-1	surface tension
14.068	DPT_Value_Common_Temperature	°C	1°C	temperature, common
14.069	DPT_Value_Absolute_Temperature	K	vK	temperature (absolute)
14.070	DPT_Value_TemperatureDifference	K	1 K	temperature difference
14.078	DPT_Value_Weight	N	1 N	weight
14.079	DPT_Value_Work	J	1 J	work

4.17 Access Control

4.17.1 General

DPT 15.000 is defined to represent or log access procedures.

4.17.2 Datapoint Type DPT_Access_Data

Format:	4 octets: $U_4U_4U_4U_4U_4U_4B_4N_4$			
octet nr.	4 MSB	3	2	1 LSB
field names	D_6 D_5	D_4 D_3	D_2 D_1	E P D C Index
encoding	U U U U U U U U	U U U U U U U U	U U U U U U U U	b b b b N N N N

Encoding:	$D_6, D_5, D_4, D_3, D_2, D_1$: binary encoded value N: binary encoded value E, P, D, C: See below
Range:	See below.
Unit:	Not applicable.
Resol.:	Not applicable.
Datapoint Types	
ID:	Name:
15.000	DPT_Access_Data

Field	Description	Encoding	Range
D6, D5, D4, D3, D2, D1	digit x (1…6) of access identification code. Only a card or key number should be used. System number, version number, country code, etc are not necessary. Ciphered access information code should be possible in principle. If 24 bits are not necessary, the most significant positions shall be set to zero.	Values binary encoded.	[0 … 9]
E	Detection error	0 = no error 1 = reading of access information code was not successful).	{0,1}
P	Permission (informs about the access decision made by the controlling device)	0 = not accepted 1 = accepted	{0,1}
D	Read direction (e.g. of badge) If not used (e.g. electronic key) set to zero.	0 = left to right 1 = right to left	{0,1}
C	Encryption of access information.	0 = no 1 = yes	{0,1}
Index	Index of access identification code (future use)	Value binary encoded.	[0 … 15]

EXAMPLE 1: Transmission of the access identification code “123456”, without error indication, permission accepted, badge read from left to right, no encryption and index 13.

<table>
<tr><td colspan="8">Octet 6</td><td colspan="8">Octet 7</td><td colspan="8">Octet 8</td><td colspan="8">Octet 9</td><td colspan="8">Octet 10</td><td colspan="8">Octet 11</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="4">APCI</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td colspan="4">D6</td><td colspan="4">D5</td><td colspan="4">D4</td><td colspan="4">D3</td><td colspan="4">D2</td><td colspan="4">D1</td><td>E</td><td>P</td><td>D</td><td>C</td><td colspan="4">Index</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="4">1</td><td colspan="4">2</td><td colspan="4">3</td><td colspan="4">4</td><td colspan="4">5</td><td colspan="4">6</td><td></td><td></td><td></td><td></td><td colspan="4">13</td></tr>
</table>

4.18 Character String

4.18.1 General

To transfer strings of characters, datapoint types 16.000 and 16.001 allow sending text of up to 14 characters. The coding of the individual characters corresponds to the datapoint types 4.001 and 4.002. The contents in both cases starts with the MSB.
Two data types exist: DPT 16.001 (DPT_String_ASCII – unused characters are set to value 00h) and DPT 16.002 (DPT_String_8859_1).

4.18.2 Example

"KNX is OK" is transmitted as:

K	N	X		i	S		O	K					
4B	4E	58	20	69	73	20	4F	4B	00	00	00	00	00

4.19 Scene Control

4.19.1 General

In KNX three different approaches exist for setting scenes

- Setting the scene conditions via ETS parameters and calling the desired parameterized scene by using the DPT_Scene_AB (1.022) 1 bit datapoint type. In this case, it is not possible that the user changes the scene.
- Setting the scene conditions of the connected actuators and storing this scene as a scene number in the connected actuators by using the DPT_Scene_Control. With the same DPT, scenes can thus be set and called.
- By using the DPT_SceneNumber: this DPT is identical to the DPT_Scene_Control, however it does not allow to store new scenes.

4.19.2 Datapoint Type Scene Number

Format:	1 octet: r_2U_6
octet nr.	1
field names	r | r | SceneNumber
encoding	0 | 0 | U | U | U | U | U | U
PDT:	PDT_GENERIC_01

Datapoint Types

ID:	Name:	Encoding:		Resol:	Range:
17.001	DPT_SceneNumber	Scene-Number	Value binary encoded	1	[0 … 63]

4.19.3 Datapoint Type DPT_SceneControl

Format:	1 octet: $B_1r_1U_6$			
octet nr.	1			
field names	C | R | Scene-Number			
encoding	B | r | U | U | U | U | U | U			
Unit:	Not applicable.			
Resol.:	Not applicable.			
Datpoint Types				
ID:	Name:	Encoding:		Range:
18.001	DPT_SceneControl	C	0 = activate the scene corresponding to the field Scene Number 1 = learn the scene corresponding to the field Scene Number	[0, 1]
		R	Reserved (0)	{0}
		Scene-Number	Scene number	[0 … 63]

4.20 Common HVAC Datapoint types

4.20.1 General

In earlier developments, the operating mode of room thermostat was set by one bit datapoint types.
Since some years, a general new DPT_HVACMode has been introduced, of which the use has become obligatory for new developments. The operating mode may be additionally set by single bit DPTs.
Next to this, a number of enumerations have been standardised for amongst others building occupancy and building mode.
Room thermostats inform on their status with the standardised DPT_HVACContrMode.

4.21 Datapoint Types N8

Format:	1 octet: N_8		
octet nr.	1		
field names	*field1*		
encoding	N | N | N | N | N | N | N | N		
Encoding:	Encoding absolute value N = [0 … 255]		
Unit:	none		
Resol.:	none		
PDT:	PDT_ENUM8 (alt: PDT_UNSIGNED_CHAR)		
Datapoint Types			
<u>**ID:**</u>	<u>**Name:**</u>	<u>**Encoding:**</u>	<u>**Range:**</u>
20.002	DPT_BuildingMode	0 = Building in use 1 = Building not used 2 = Building protection	[0 …3]
20.003	DPT_OccMode	0 = occupied 1 = standby 2 = not occupied 3 …255 not used; reserved	[0 … 3]

20.102	DPT_HVACMode	0 = Auto 1 = Comfort 2 = Standby 3 = Economy 4 = Building Protection 5 … 255 = reserved	[0 … 4]
20.105	DPT_HVACContrMode	0 = Auto 1 = Heat 2 = Morning Warmup 3 = Cool 4 = Night Purge 5 = Precool 6 = Off 7 = Test 8 = Emergency Heat 9 = Fan only 10 = Free Cool 11 = Ice 12 … 19 = reserved 20 = NoDem 21 … 255= reserved	{[0 … 11], 20}

5 Combination of DPTs into devices

5.1 General

For two very common device types, the combination of DPTs into devices has been standardized and is obligatory for certification.
These standards called ‘functional blocks’ are respectively the “Dimming actuator basic” and the “Sunblind Actuator basic”.

5.2 Functional Block – Dimming Actuator Basic

5.2.1 General

A dimming actuator shall per channel at least support three group objects complying to the underneath stated DPTs:

- Switch, DPT 1.001
- Relative dimming, DPT 3.007
- Absolute dimming, DPT 5.001

Any other group objects are optional.

5.2.2 Dimming Actuator Basic – Status Diagram

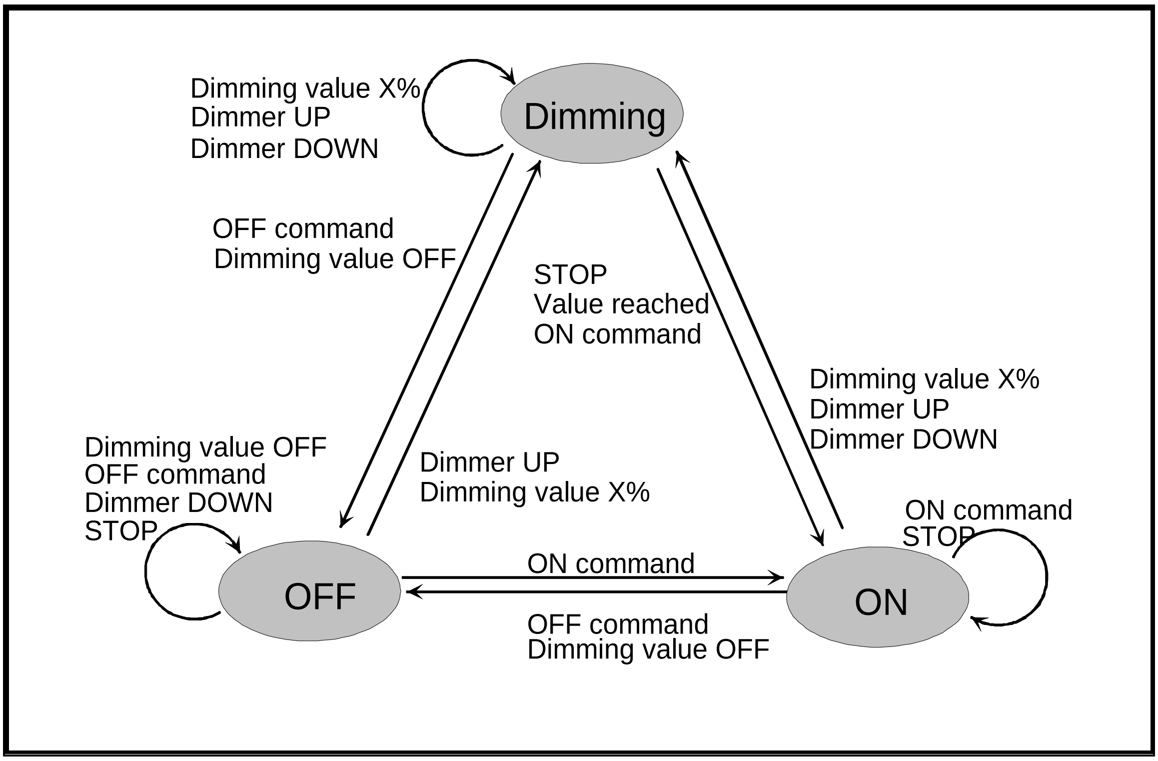
The basic function of the dimming actuator is described in the following status diagram.
A dimming actuator can find itself in one of three states – depending on which telegram(s) it receives:

State	Description
Off	Dimming actuator is switched off
On	Dimming actuator is switched on, at least the lowest possible brightness is set
Dimming	Dimmer controller is enabled, brightness is adjusted in the direction of the setpoint

The change from one state to another is triggered by so-called “events”. Events include:

Name	Meaning
OFF command	Switching object receives value 0
ON command	Switching object receives value 1
Dimming up	Dimming object receives a new value, upward dimming direction
Dimming down	Dimming object receives a new value, downward dimming direction
Stop	Dimming object receives value ‘Stop’
Dimming value OFF	Brightness obj. receives value 0
Dimming value X%	Brightness obj. receives value > 0
Value reached	Brightness has reached setpoint

The event “Value reached” is an internal event. The function of the application software of a dimming actuator is illustrated by the following status diagram. The ellipses represent the states while the arrows represent the events.
This behaviour is checked during KNX certification tests.



5.3 Functional Block – Sunblind Actuator Basic

5.3.1 General

The function “Drive Control” is used primarily for controlling blind and roller shutter motors.
A sunblind actuator shall per channel at least support two group objects complying to the underneath stated DPTs:

- StopStep UpDown, DPT 1.007
- Move UpDown, DPT 1.008

Any other group objects are optional.
Important: Group objects using this function shall not reply to read requests through the bus (Group Value Read messages). This restriction ensures that drives are not inadvertently set into motion. The READ flag of the group objects shall therefore be reset! This applies both to sensors AND actuators!
If devices are not of the type BCU1, the above is not needed, provided the update flag in the respective objects are not set.

5.3.2 Sunblind Actuator Basic – Status Diagram

An actuator for the control of blinds and shutters may be in one of four states, depending on the type of telegram(s) received:

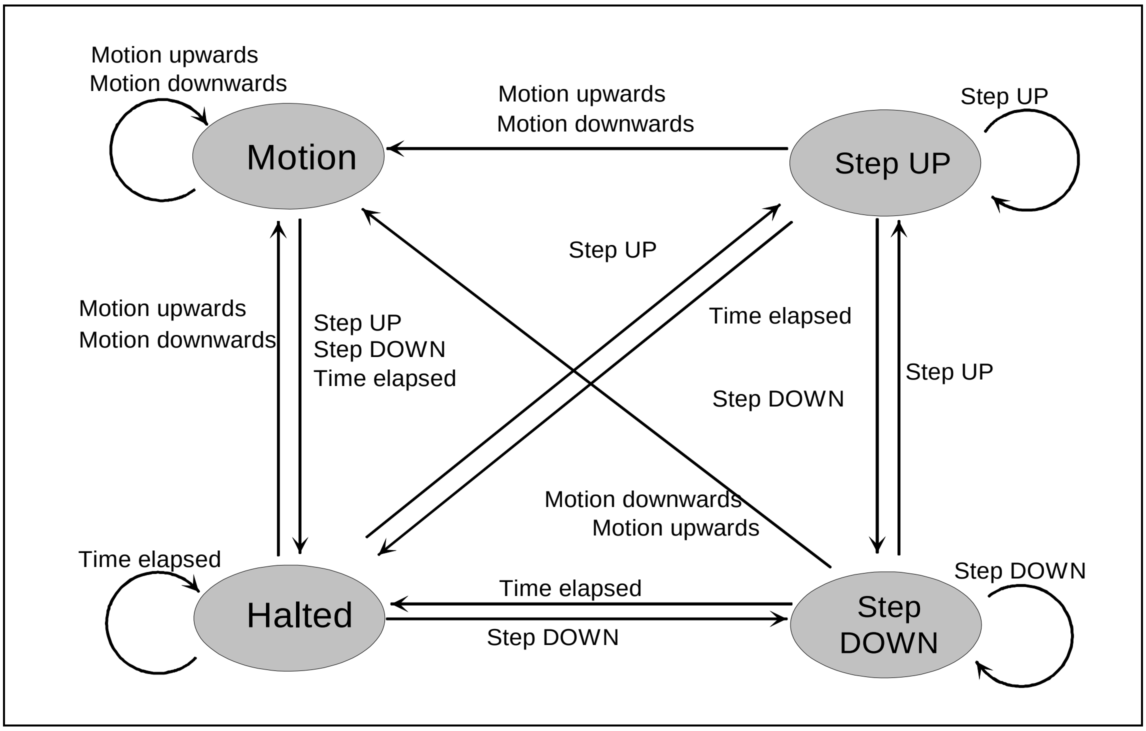
Status	Description
Stopped	No movement
In motion	The connected drive moves upwards or downwards
Step UP	The drive is in the step mode and is moved upwards by one step
Step DOWN	The drive is in the step mode and is moved downwards by one step

(Depending on the type of application, it is of course possible to declare other directions of movement than those stated above, e.g. right/left or forwards/backwards. Example: motor-driven sliding door.)

The following events trigger the changeover from one status to another:

Name	Meaning
Move UP	Move object receives value '0'
Move DOWN	Move object receives value '1'
Step UP	Step object receives value '0'
Step DOWN	Step object receives value '1'
Time elapsed	Time limit defined for the motion of one step or for the entire range of motions has run out.

The event 'Time elapsed' is an internal event. The operation of an actuator for the control of blinds and shutters is illustrated by the following status diagram. The ellipses represent the states whereas the arrows indicate the events. This behaviour is checked during KNX certification tests.


Motion upwards
Motion downwards
Motion
Motion upwards
Motion downwards
Step UP
Step UP
Step UP
Motion upwards
Motion downwards
Step UP
Step DOWN
Time elapsed
Time elapsed
Step DOWN
Step UP
Motion downwards
Motion upwards
Time elapsed
Halted
Time elapsed
Step DOWN
Step DOWN
Step
DOWN