KNX Association

Serial Data Transmission and KNX Protocol

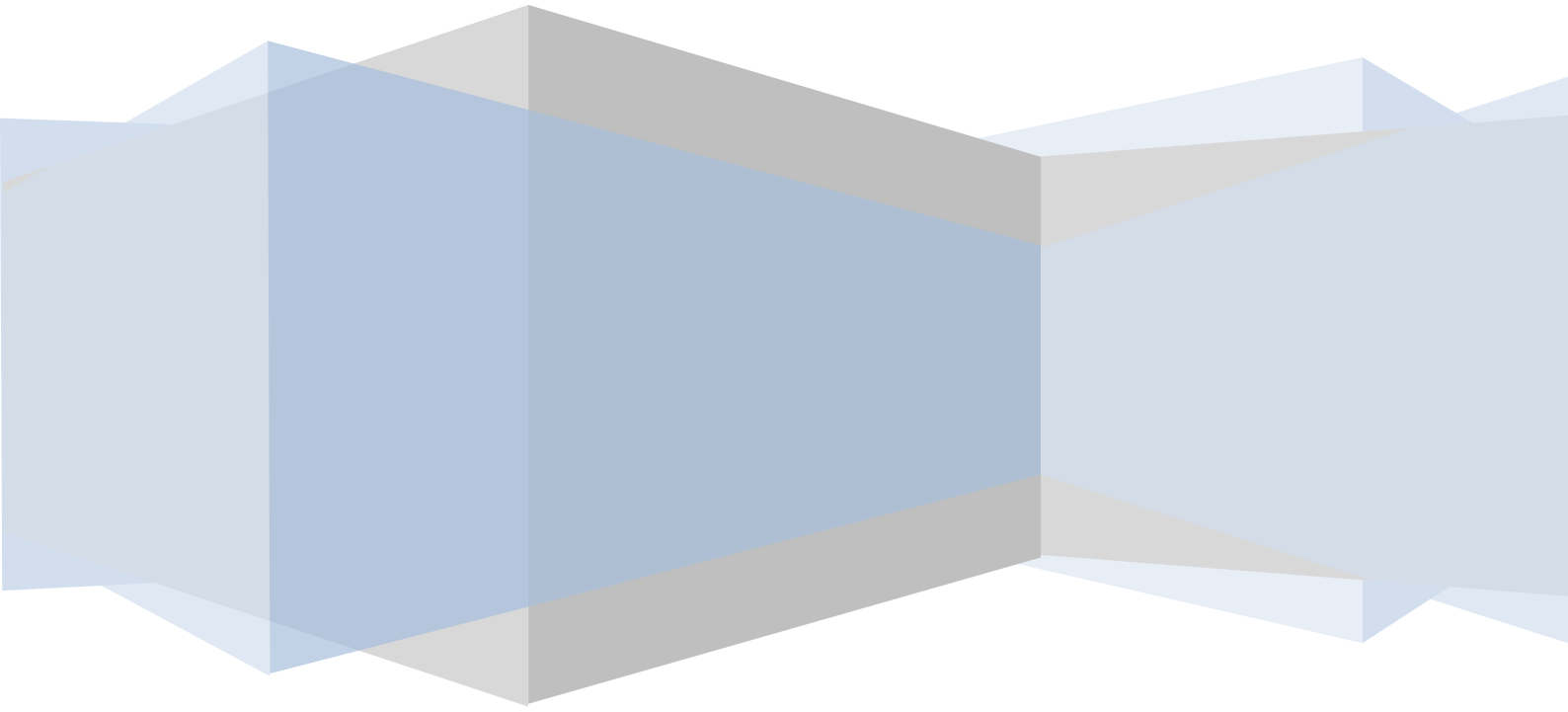


Table of Contents

1 Fundamentals of Serial Data Transmission

1.1 Introduction and Terminology

1.1.1 Hierarchical Problem Division

Hierarchical division expediently solves the variety of tasks that arise from the automation of machining or operational procedures. The hierarchical levels take on the following tasks for example:

Main computer:	All Manufacturing (or Material) Requirements Planning (MRP) and Production Control is done at this stage. This is, among other things, manufacturing planning resulting from requirements for production planning / scheduling as well as central production control. It is therefore necessary to feed condensed production-specific data from the process level back to the main computer.
Process computer/ Control centre:	These control one section of the manufacturing process. In car manufacturing, such plant sections could be the car body production line, the paint line, the assembly of the drive mechanics, etc. The process computer is responsible for data acquisition and compression, for monitoring and optimisation.
Field level:	This could for example be a programmable (logic) controller (PLC) with all the sensors and actuators, or intelligent sensors and actuators, connected to it.

1.1.2 Hierarchical Structures

The hierarchical division of tasks leads to the hierarchical structure of the communications networks used in plants and installations. The demands made on the network are different at each level.

The operations level handles large data volumes but on the other hand requires no time-critical data transmissions. Relatively small amounts of data are processed on the field level but uncontrollable delays in data transmission cannot be tolerated at this level of the automated installation.

1.1.3 Where in this Hierarchy do we find the Building Systems Buses?

The networks used in automated building installations are typical field bus systems (sensor-actuator-bus).

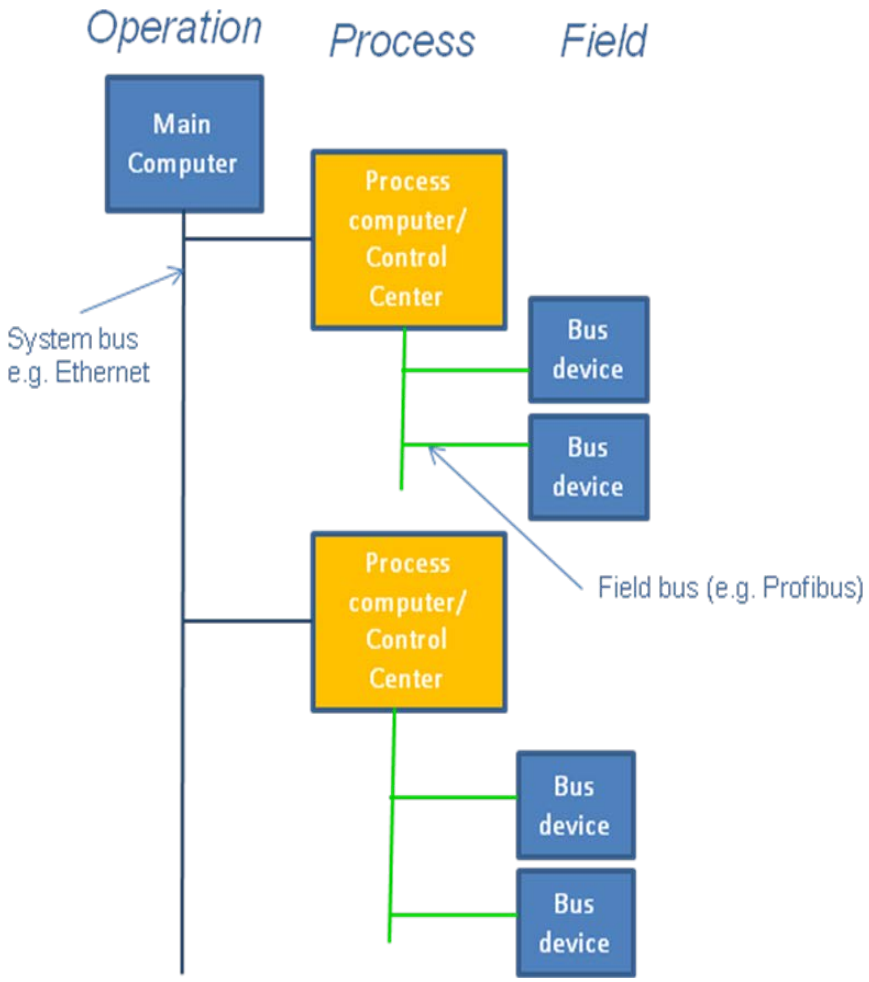


Figure 1: Where do we find the building systems buses?

1.1.4 Classification of Serial Buses

According to their methods of data transmission and bus access:

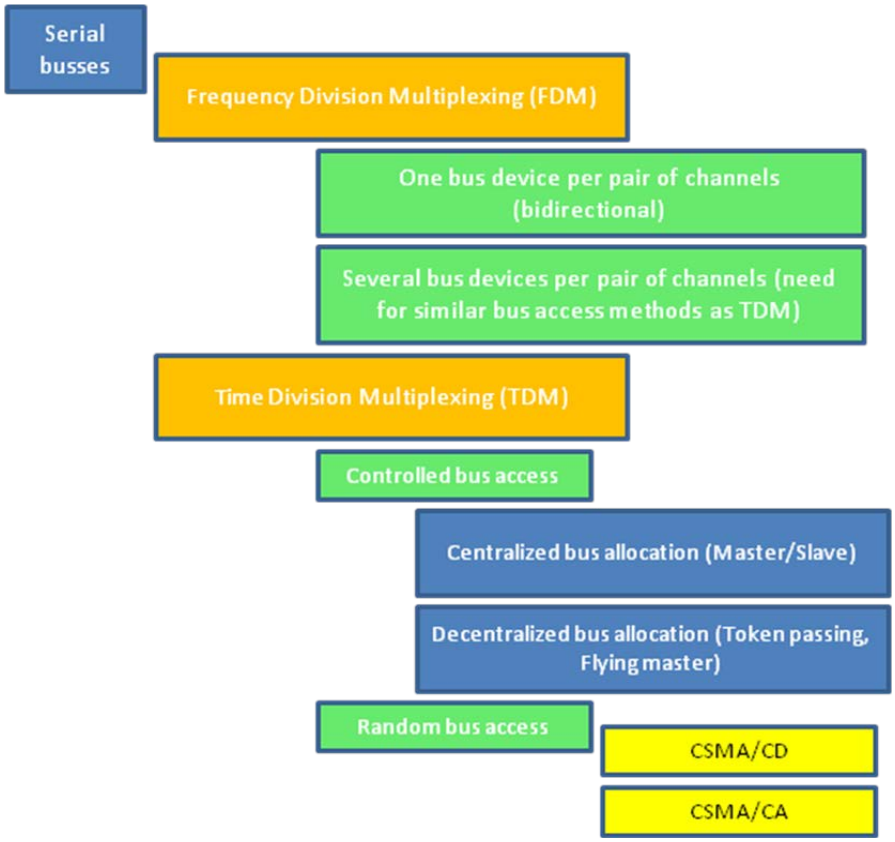


Figure 2: Classification of Serial buses

Frequency Division Multiplexing (FDM)
The messages to be transmitted modulate the carrier frequencies, either by amplitude or frequency modulation. One bus line can therefore transmit several channels (different frequencies) each of which allows data traffic independently of the others.

Time Division Multiplexing (TDM)
Certain time slices are allocated to each message or channel, i.e. the bus devices may only use the bus one after the other. With TDM therefore, access to the bus must be controlled in order to prevent simultaneous bus access of several bus devices.

- centralised bus allocation (Master/Slave)
- decentralised bus allocation (e.g. Token Passing, Flying Master)
- combination of both access control methods (e.g. PROFIBUS)
- CSMA/CD (Carrier Sense Multiple Access with Collision Detection e.g. Ethernet)
- CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance e.g. KNX,CAN)

How the Shape of the Signal (Bit Representation) Changes with Modulation

A carrier frequency is modulated by the binary information to be transmitted:

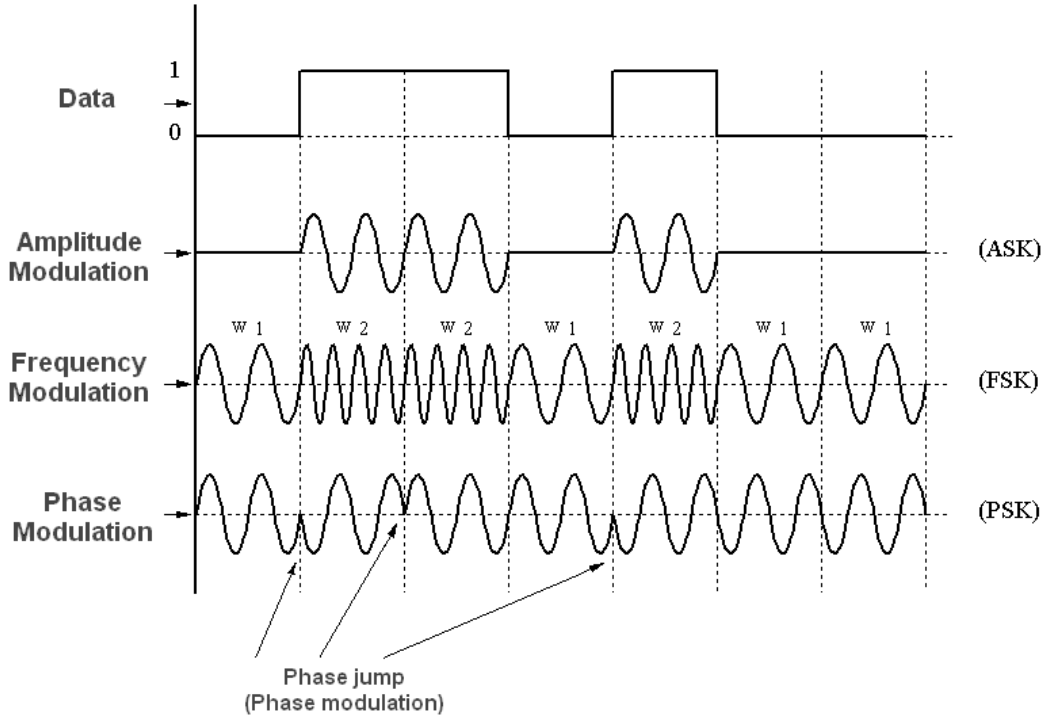


Figure 3: Change of shape of signal with modulation (Bit representation)

Modulation is used both for Frequency Division and Time Division Multiplexing.

How the Shape of the Signal (Bit Representation) Changes with Baseband Transmission

With baseband transmission the binary information is transmitted in the form of rectangular pulses. This type of data transmission is used in KNX.

Typical formats of baseband transmission

Example: Consider the bit sequence 00101100

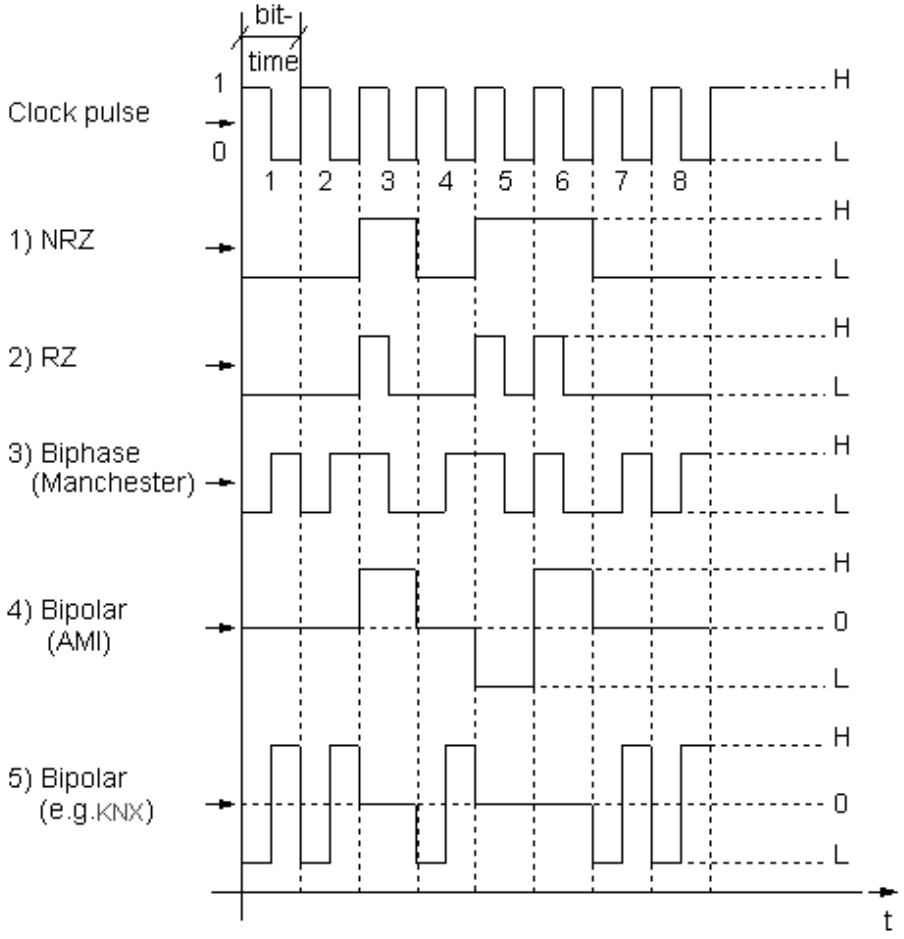


Figure 4: Change of shape of signal with Baseband transmission (Bit representation)

NRZ = non return to zero
RZ = return to zero

1.1.5 Error Detection Techniques

- Parity (VRC - vertical redundancy check)
  One bit per character, even or odd; only suitable for character-by-character data transmission as used, for example, in KNX; usually applied to all start-stop transmission methods; often referred to as vertical parity. Parity checks only detect odd bit errors and not the even ones.
- Horizontal/longitudinal parity (LRC - longitudinal redundancy check)
  A check digit is used which contains exactly as many bits as the characters of the transmitted information. The bits of the check digit are derived from the associated bit positions of all characters of the information already transmitted and appended to the actual information in the form of (even or odd) parity bits.
- Cross check
  Combination of vertical and horizontal parity checking. This type of transfer check is capable of correcting single-bit errors and detecting double-bit errors. Example: KNX with even vertical and odd horizontal parity.

Example 1

Running number	**Character**								**Even vertical parity**
1	1	0	1	1	1	1	0	0	1
2	0	1	0	1	0	0	0	1	1
3	0	0	0	0	1	0	1	0	0
4	0	**(1)**	0	0	1	0	0	0	**[1]**
5	0	0	0	0	0	0	0	1	1
6	1	1	1	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	1
Odd horizontal parity	0	**[1]**	1	1	0	0	0	0	

There is a single-bit error in the 4th character. This error is detected by the vertical parity check, whereas the position of the incorrect bit is recognised by the horizontal parity check. This makes it possible to correct errors.

Example 2

Running number	Character								Even vertical parity
1	1	0	1	1	1	1	0	0	1
2	0	1	0	1	0	0	0	1	1
3	0	0	0	0	1	0	1	0	0
4	0	**1***)	0	0	**0***)	0	0	0	1
5	0	0	0	0	0	0	0	1	1
6	1	1	1	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	1
Odd horizontal parity	0	**1****)	1	1	**0****)	0	0	0	

*) Bit error **) Bad parity

The 4th character contains a 2-bit error which remains undetected by the vertical parity check but is recognised by the horizontal parity check.

- Cyclical check (CRC - cyclical redundancy check)
  With this error detection method, a check digit is generated as a result of an extensive polynominal calculation. The CRC procedure detects even complex errors.

Comparison of the Different Error Detection Techniques

Method	Reduction of undetected bit errors by factor (approx.):
Vertical parity	10^2
Horizontal parity	10^2
Cross check	10^4
Cyclical check	10^5

2 The OSI Reference Model (ISO 7498)

2.1 Introduction to the OSI Reference Model

2.1.1 Why are Standards Necessary?

When data processing systems and communication networks of different manufacturers were first introduced, there was no compatibility between them. Intercommunication between systems of different levels and standards was therefore difficult and only possible with great efforts. The multifarious networks on the market were designed for very different purposes in the industry and business sectors. Each network was a self-contained system closed to communication with another.

2.1.2 Objective of the OSI Reference Model

The International Standardisation Organisation (ISO) in the seventies therefore decided to formulate a standard architecture for computer networks and information processing. The aim was to…

- develop a standardised model in data communications on the basis of which distributed networks and systems could talk to each other,
- achieve compatibility among networks without insisting on implementation of the complete model for each network,
- achieve open communication (Open System Interconnection OSI) among all users of a heterogeneous computer network.

2.2 The Principle of the OSI Reference Model (OSI)

OSI generally specifies the functions required for data communication and the interaction between these functions. The OSI model stipulates that higher-level functions build on lower-level functions according to well-defined rules. The result is therefore a hierarchical concept consisting of layers.

During the decoding of a telegram by the bus device, each layer decodes part of the telegram. During the creation of a telegram by a bus device, the grouped functions inside an OSI layer all contribute for a part to the telegram.
The above can be respectively compared to peeling off the layers of an onion to arrive to the core (= useful information) or adding onion layers to the core information.

2.2.1 The Seven Layers of the OSI Reference Model

- Layer 1: Physical Layer
  Generation of electrical signals, depending on the transmission medium used; connector pin assignment, control signals, bit coding.
- Layer 2: Data Link layer
  Layer for connections and procedures; line protocol, synchronisation, data security, error handling.
- Layer 3: Network Layer (also referred to as Packet Level)
  Determination of suitable paths for data transmission and switching the links involved (routing), packet control, transport protocol through the entire network.
- Layer 4: Transport Layer
  End-to-end control, transport control
- Layer 4 is responsible for the transport of the entire data volume (i.e. of all the blocks or packets); packet organisation, multiplexing.
- Layer 5: Session Layer (also referred to as Communication Layer)
  Control of logical connections, link to the application process, buffer management *(not used in KNX)*
- Layer 6: Presentation Layer
  Data format conversions, data compression, message coding *(not used in KNX)*.
- Layer 7: Application Layer (also referred to as Process Layer)
  Can be freely defined by the application.

2.2.2 The Two Types of Communication

- Connection-oriented Communication:

The established connection constitutes a logical link between two communication partners. Both partners regard this connection as a link that only exists between them (point-to-point [P2P] connection) and that can only be used by them.

The server (= the device wishing to establish a communication) must establish this connection according to some specified criteria (quality, cost, time delays, etc.) and uses resources to do so. In addition, a protocol or parameters, such as the block length of the data, may be defined. When the connection is established, the actual data exchange (data transport) is executed according to the protocol.

The connection may be aborted at any time by any of the partners (disconnection). The resources used for the link are then released and made available to other functions.

Such communication is highly secure, but is not very bandwidth effective (i.e. a lot of communication is needed to reach only one single communication partner).
When considering the OSI model, during this communication the Transport Layer as well as the Link Layer will generate messages confirming reception of data.
This type of communication can be compared to talking to one single person and checking after each question, whether he/she understood the question.

Use in KNX:

Downloading group addresses, parameters and/or application program from the PC (= ETS) into the KNX device

- Connectionless Communication

This type of communication uses no connection. It is therefore not possible for the involved devices

- to agree upon a protocol or parameters
- to fix criteria such as service quality, cost, etc.

As there is no connection, the address must be sent together with the useful data after each data request. There is no confirmation of the data received and the device sending the information (=server) does not guarantee the correct order of the transmitted data blocks.

Such communication is not secure, but very bandwidth effective: there is no need to establish or break down a communication connection and with one single message it is possible to reach all devices included in the installation. However, only one common acknowledgement of receipt is received. It can therefore not be ensured that all devices have correctly received the sent message.

When considering the OSI model, during this communication only the Link will generate acknowledgements of receipt.

This type of communication can be compared to talking to a group of persons and continuing with the next question if at least one person confirms that he/she understood the message.

Use in KNX:

- Normal operating mode of the KNX (telegrams with group addresses, Multicast, one to many communication)
- Broadcast (one to all communication)

2.2.3 Protocol Data Units (PDUs)

The Protocol data Unit of an OSI layer constitutes

- the data that is contributed by this Layer to the telegram when created;
- The data that is decoded by this Layer from the telegram when received;

2.3 Principle Functions of the Layers; Implementation in KNX

2.3.1 General

The underneath description focuses on the realization of KNX on the TP1 medium. For deviations to this for different media, please consult the relevant chapter in this documentation.

2.3.2 Application Layer (Layer 7)

The application layer basically provides functions for two tasks:

- Assisting the application program in sending and receiving useful information

  Implementation in KNX:
  KNX is a bus using objects for communication (called 'group objects'[1], similar to CAN bus). A group object can be, for instance, the switching state, the daylight intensity or the temperature in a room.
  The application program of a sensor "measures" the physical quantity (e.g. contact state, lux value, temperature, etc) and writes the value of this quantity in the appropriate group object. At the same time it will request to the system software to send the new value of the group object on the bus in order to inform the communication partners of the sensor of this updated value.
  The Application Layer of the addressed actuator(s) will ensure that new received value is written in the group object(s) concerned and subsequently inform the application program(s) of the actuator(s) of the update. The program(s) reads the value of the actuator's group object and executes the required function accordingly. This can be the switching of a relay, reducing the light intensity of a lamp, operating a valve, etc.

  Application programs in a KNX network communicate therefore by way of group objects. Hence no connection is established for data communication (refer to "Connectionless Communication").

  Note that the application program is unable to retrieve the source of the update of the group objects!!

- Assisting devices in understanding configuration messages called 'management services'.

  During the commissioning phase of bus devices, connection-oriented communication is established between devices to download the application programs. It is the Application that will help to understand these configuration commands.

[1] Previously referred to as 'communication objects'.

2.3.3 Presentation Layer (Layer 6)

This layer of the OSI model has the task of shielding layer 7 and the application program from the duty of having to deal with different forms of representation of the transmitted data. It interprets and, if necessary, adjusts or converts the syntax of the message.
For example, a file containing the particulars of a group of persons must be transmitted from one bus device to another. The first device uses data records containing fields in the order "name, first name", whereas the second device uses the reverse order. In this case it would be the task of the presentation layer to ensure that each device receives the data record in the correct order; layer 6 must therefore automatically switch the order of the data record fields as required.
With field buses in general, and therefore also in KNX, the problem of converting the form of representation in the transmitted data does not arise. Hence, KNX devices *do not need* an OSI layer 6.

2.3.4 Session Layer (Layer 5)

The principle task of layer 5 according to OSI is to control the communication between two communication partners. This involves:

- opening a dialog,
- closing a dialog,
- aborting or interrupting a dialog,
- continuing a dialog at a later stage,
- picking up a dialog at a defined, backdated point of time in the event of detected errors, so that a faulty data transmission can be repeated correctly.

As the messages exchanged in field buses are short (measured values), these buses require no explicit communication control. Hence, KNX devices *do not need* an OSI layer 5.

2.3.5 Transport Layer (Layer 4)

- **Function during connectionless Communication**

For connectionless communication, the Transport layer in KNX has the following function:

- When sending, ensuring that the value of the changed group object is sent by the link layer with the correct associated (sending) group address.
- When receiving, ensuring that the value of all group objects is updated to which the received group address is linked.

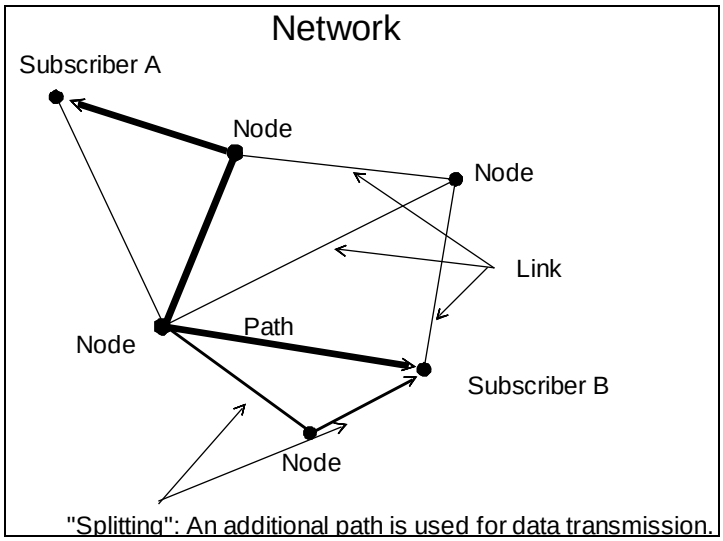
In connectionless communication, the Transport Layer is in other words responsible for checking the Association table loaded in bus devices. This table constitutes the relation between the supported group objects and the attributed group addresses.

- Function during connection-oriented Communication

In order to establish a connection-oriented communication, the initiating device will send a Transport Layer connection message, using the individual address[2] of the receiver as the destination address.
During an established connection oriented communication between two communicating bus devices, the Transport Layer of both devices will make use of the Transport Layer "ACK" (positive confirmation) or "NACK messages" (negative confirmation) to acknowledge or reject received telegrams.
On reception of a NACK message from Transport Layer, rejected telegrams (negative acknowledgement) are repeated up to three times. The repetition parameter is fixed in the system software.
Connection oriented communication is monitored by means of timers. If a telegram cannot be transmitted within the set time, or if neither "ACK" nor "NACK" are received from the communication partner, then an established connection is automatically cleared down.
The connection oriented communication is also monitored with an additional sequence number (between 0 and 15). If the sequence number does not have the expected value, the receiver will automatically clear down the established connection.

2.3.6 Network Layer (Layer 3)

A network is usually a combination of nodes which are interconnected by individual links according to a defined topology. The principal task of the OSI layer 3 (network layer) of such a network is to find suitable paths for data transmission, switch the links involved and ensure that the telegrams are directed toward the destination. This is often referred to as "routing".



"Splitting": An additional path is used for data transmission.

The nodes in between subscriber A and Subscriber B must ensure a "switching service", so that the message sent by A reaches B. This is comparable to relaying calls in a public telephone network.

[2] Previously referred to as physical address.

A KNX Twisted Pair (KNX-TP) network has the following structure:

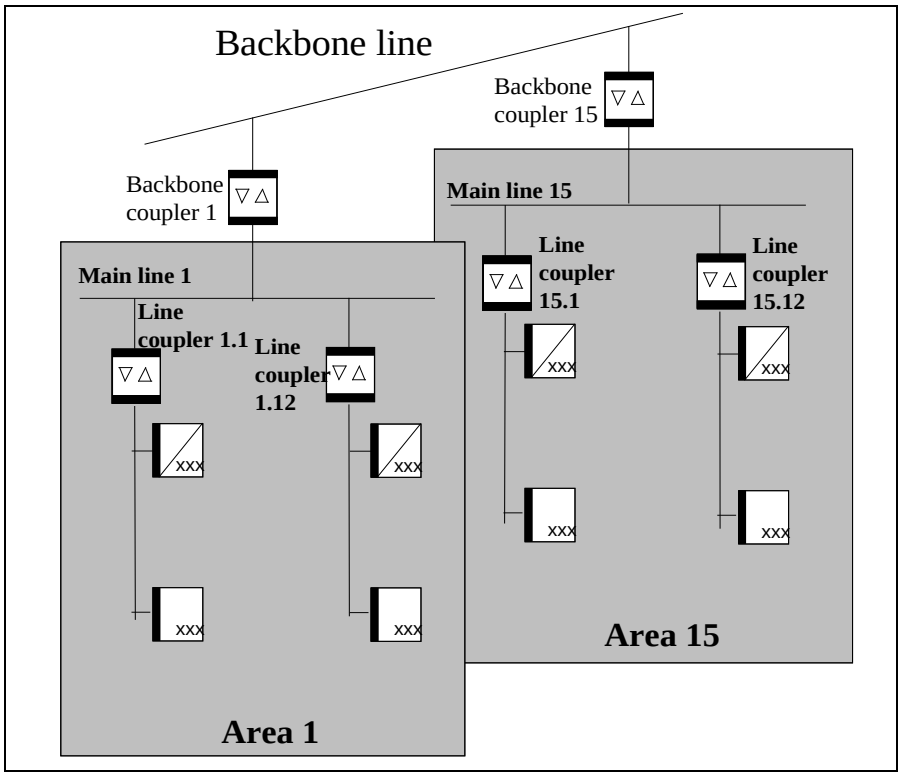


Figure 5: KNX TP network structure

The individual segments of the KNX-TP network constitute the links, whereas the backbone couplers and line couplers are the nodes of the network.

Loops between two lines are not permitted in a KNX-TP network. As the structure of the KNX network is relatively uncomplicated, routing can be solved by fairly simple means.

When a bus device is sending a telegram, the network layer will insert the routing counter value (typically stored in EEPROM) in the sent telegram. This routing counter value is only evaluated by the network layer of backbone or line couplers in the installation (not by normal end devices).

- In the case where the value is 0, in all cases the received telegram will not be routed by the receiving backbone or line couplers
- In the case where the value is between 1 and 6, the received telegram will be routed by the received backbone or line coupler, depending on:
  - In connectionless communication: whether the group address used in the telegram as destination address is contained in its filter table;
  - In connection-oriented communication, whether the individual address used in the telegram as destination address is a device located in the line or area, which is located at “the other side” of the backbone/line coupler where the telegram was originally received. During routing, the coupler will decrement the received value of the routing counter.
- In the case where the value is 7, in all cases the received telegram will be routed by the receiving backbone or line couplers.

It is checked during certification conformity testing whether KNX devices only use routing counter values equal to or lower than 6. The use of routing counter 7 is restricted to ETS only.
The routing of telegrams in the above described way, ensure that telegrams are only routed to lines and areas where the telegram is needed (or on its way to its final destination). This considerably reduces bus load.
The routing mechanism works in a similar way in media couplers between TP and PL, PL backbone couplers, IP routers ... For more information, see chapter ‘couplers’.

2.3.7 Data Link Layer (Layer 2)

General Task of Link Layer
According to OSI, this layer has the task of ensuring the "error-free" transmission of telegrams within a link, i.e. between two nodes of a network (read for KNX: line or backbone couplers) or between a bus device and a node.
Hence information such as synchronisation characters, sequence numbers, error check field and other control characters are included in telegrams, in addition to the actual data to be transmitted.

General Structure of KNX Link Layer Telegram
The figure below illustrates the structure of a KNX telegram transmitted by layer 2.

Control field	**Source address**	**Receiver address**	**N_PDU**	**Check field**
8 bit	16 bit	16 bit		8 bit

The source address is the individual address of the bus device.
The receiver address can either be a group address (connectionless communication) or an individual address (connection-oriented communication): this is indicated by the first bit in the N_PDU (see later).
The N_PDU is the information that is destined to/included in the telegram by the OSI layers above Link Layer (in the case of KNX: Network, Transport and Application Layer).
The other fields are explained in detail underneath.

Structure of control field

The control field, comprising of 8 bits, has the following structure: Please note that inside a character, D0 is the first bit to be sent on the bus.

D7	D6	D5	D4	D3	D2	D1	D0
1	**0**	R	**1**	*P*	*P*	**0**	**0**

The **bold values** of the control field must be set to the indicated value: if not, the Link Layer will reject the telegram as an invalid KNX telegram.
D0 and D1 constitute the preamble bits of the telegram: the preamble bits ensure that any spikes that may occur on the bus line are not misinterpreted as start bits.
The bits marked with *P* determine the sending priority of the telegram: four levels are defined:

00	Priority 1	System functions
10	Priority 2	Alarm functions
01	Priority 3	Normal mode, high priority
11	Priority 4	Normal mode, low priority

Note: The term "priority" refers to the priority defined at layer 7 for the group objects concerned. This priority is passed through all layers down to layer 2.

The bit D5 (R) indicates whether the telegram is a repeated telegram (= value 0) or not (= value 1).
As the zero overrides the one (see Layer 1), telegrams of higher priority are handled first in the event of a collision. The same applies to telegrams that are repeated; they too will be handled with (higher) priority.

Structure of Source address

The underneath figure shows the structure of the individual address of the sender.
If two devices belonging to one and the same electrical segment (see Layer 1) transmit simultaneously, their telegrams differ in at least one item: the individual address of the sender. As a result, a collision will occur, at the latest, upon transmission of the sender address. The bus device whose telegram has a 0 and occupies the first position in the sender address while all the other devices are simultaneously sending a 1, can transmit its telegram without any interruption (CSMA/CA, again note the sending sequence of the bits: data bit 0 is sent first, line and area addresses are sent before the device address).

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Area 0= backbone 1 to 15 = area				Line 0 = main line 1 to 15 line				Device address 0 = coupler 1 to 64= line Above 64: line extension, other line segment							

Structure of Receiving address

The receiving address is an individual address as indicated above for point-to-point connections (connection-oriented communication). For multicast or broadcast addressing (connectionless communication), addresses called “group addresses” are used as receiver addresses and have the following structure:

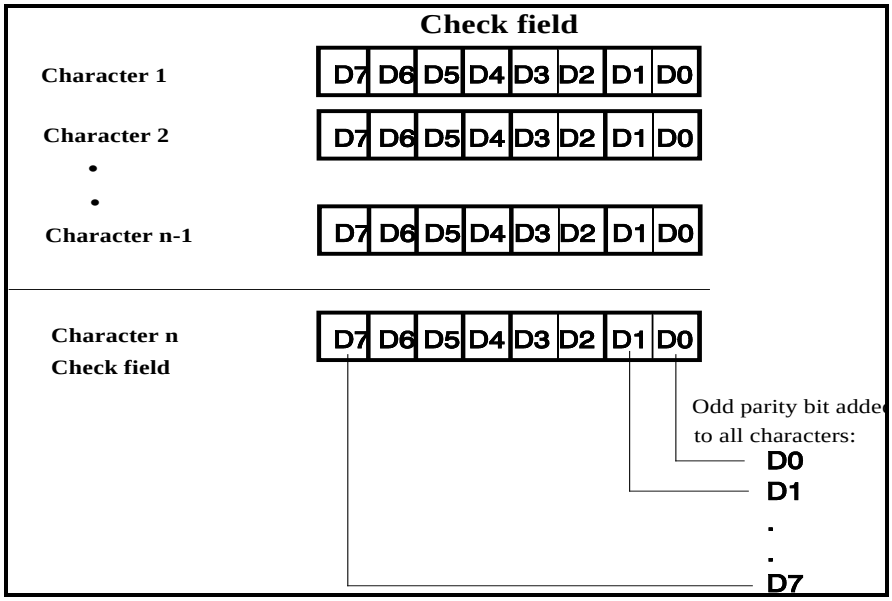
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	Main Group				Subgroup										
	Main Group				Middle Group			Sub Group							

D7
T

The group address is coded on 15 bits (Bit D15 is a reserved bit).
The first row shows how the group address is represented in a two level hierarchy (main/sub), while the second in a three level hierarchy (main/middle/sub). Note that physically on the bus, there is no difference between a two and a three level group address!
The bit D7 in the next character (marked ‘T’) determines whether the preceding address is of the type group (=value 1) or individual (=value 0).

Structure of the check field

The check field is generated according to the cross check method:



Acknowledgment of telegrams by Link Layer

To ensure transmission acknowledgement, it is necessary that the Link Layer of the addressed bus devices or the line/backbone coupler immediately send an acknowledgement all within a specified time frame (Immediate Acknowledgement / Immediate Not Acknowledge ("IACK", "INACK")).

The acknowledgement of the type "BUSY" controls the data flow. It must be transmitted in the same time frame as IACK, INACK by the device that is momentarily busy.

If the sending layer 2 receives an INACK or BUSY message, or a faulty or no IACK acknowledgement, then the telegram or message will be repeated.

Repetition telegrams are marked as such to ensure that bus devices that have acknowledged a transmission with IACK, do not send this message to the upper layers (i.e. Network Layer) a second time.

The following formats for Immediate Acknowledgements are defined:

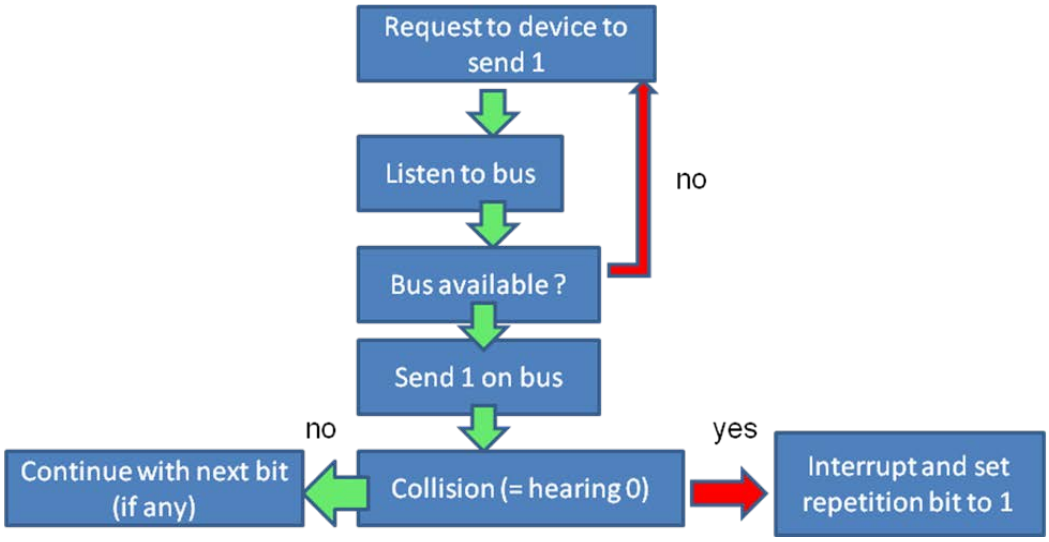
- Negative Acknowledgement: coding 0Ch (00001100b)
- Busy Acknowledgement: coding C0h (11000000b)
- Positive Acknowledgement: coding CCh (11001100b)

All the addressed devices acknowledge *simultaneously* (group acknowledgement). If several devices of one electrical segment (link) acknowledge differently, due to the selected coding the BUSY message overwrites the IACK and INACK messages. INACK overwrites IACK.

Error detection and bus access control

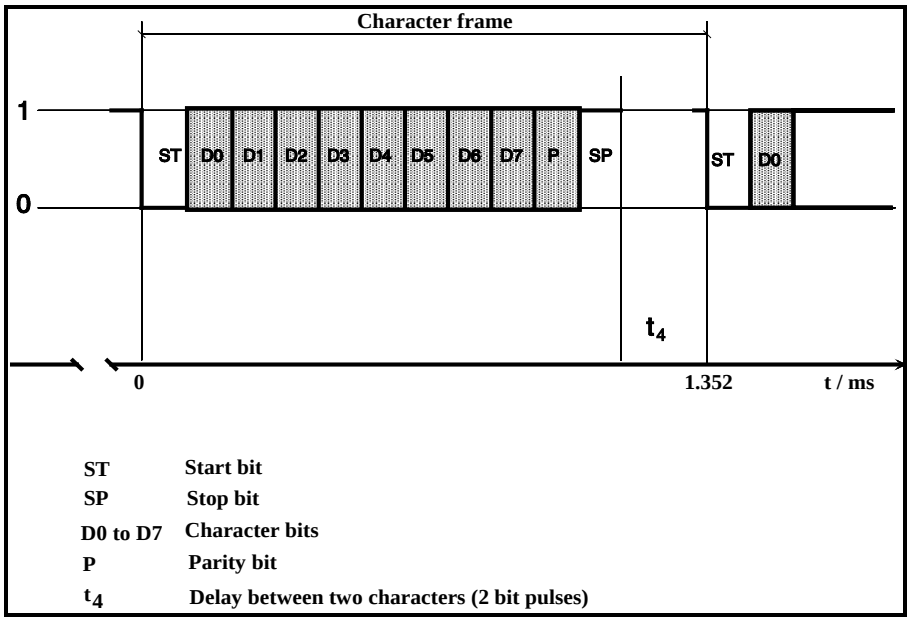
In KNX, layer 2 uses the cross-checking method for error detection. This is a combination of horizontal and vertical parity checks.

CSMA/CA is used for bus access control. Information on the status of the bus is conveyed to layer 2 via layer 1 ("Bus available" or "Collision").



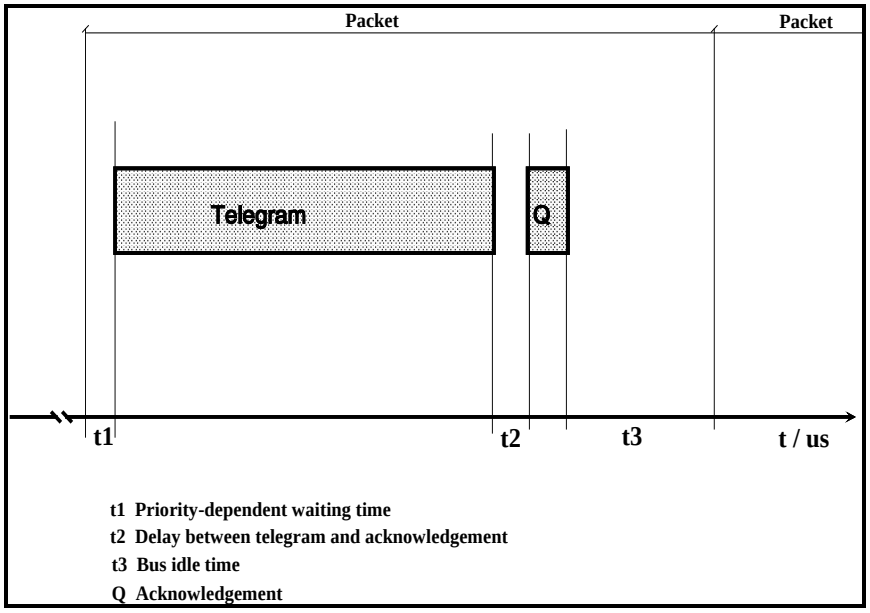
Telegrams of the first and second priorities as well as repetition telegrams can be transmitted immediately when the Layer 1 indicates that the bus is available, whereas messages of the lower priorities are delayed by the time t_1 (= 3 bit pulses).

Since the transmitted character is heard bit-by-bit, the (or several) device(s) which simultaneously sends a 0 bit will not notice collision and will therefore continue to send as if no collision had occurred. This means that no time is lost since there is always an incoming telegram.
The bits of the telegrams are transmitted character by character (1 character = 8 bits) according to the Start-Stop procedure (even parity, 1 stop bit). A delay t_4 of 2 bit pulses separates one character from the next. This means that a total of 13 bit pulses are required per character.



The protocol foresees a transmission break t_2 of at least one character width after the sending of the entire telegram. The next character width is reserved for the reception of acknowledgement, which must be sent by the addressed bus devices of the link or of the line coupler / backbone coupler / line repeater.

A bus idle time t_3 of 50 bit pulses is observed after the transmission of the acknowledgement (IACK, INACK or BUSY).



The total time t_{tot} required for the transmission of the telegram and the acknowledgement and the observance of the bus idle time is calculated as follows:

$$t_{tot} = t_1 + t_3 + (n+2)*13*bitpulses,$$
$$8 \le n \le 23$$
$$20ms \le t_{tot} \le 40ms$$

2.3.8 Physical Layer (Bit Transmission, Layer 1)

The lowest layer in the OSI Reference Model is called the physical layer. This layer is concerned with the nature of the signals. It has the task of identifying the bits received from layer 2, converting them to physical signals such as voltages, currents and electromagnetic waves (radio or optical signals) and finally transmitting these signals over the bus transmission media (copper cable, optical fibre etc).

Principal aim of the services provided by layer 1 is to shield layer 2 from the physical means used for bit transmission. It is thus ensured that the upper layers of the network remain independent of the transmission physics used, so that principally the transmission media could be changed without having any effect upon the upper layers.
The physical layer contains the protocol, defines the cabling and wiring between devices and contains the specifications for electromechanical components such as plugs and connectors.

The bit transmission layer of the KNX network is characterised as follows:

The KNX-TP network comprises one or several electrical segments. Each segment has one or two power supplies but - as per definition - does not contain any line couplers.

Technical data of an electrical segment:

- Random topology.
- Total capacitance of one segment:
  without bus devices, line coupler, line repeater: 100 nF max.
  with bus devices, line coupler, line repeater: 120 nF max.
  (measured at 10 KHz)
- Bus line resistance between power supply and bus device, line coupler or line repeater: 25Ω max.
  Bus line resistance between two bus devices, line couplers or line repeaters: 50Ω max.
- Minimum resistance between two power supplies: 15Ω
  Minimum bus line length between two power supplies: 200 m.
- Voltage drop on bus line between power supply and bus device or line coupler: 5 V max.
  This criterion determines how the bus devices may be physically arranged in the bus line of a segment. It is not permitted, for instance, to have a conglomeration of 64 bus devices installed at one end of a 350 m bus line and the power supply at the other.
- Maximum bus line length of a segment: 1000 m
  Maximum bus line length between two devices: 700 m
  Max. bus line length between power supply and bus device: 350 m
- No terminating resistances required.
- The bus devices are supplied with a rated voltage of 24 V DC through the bus.
- Max. number of bus devices of an electrical segment: 64.

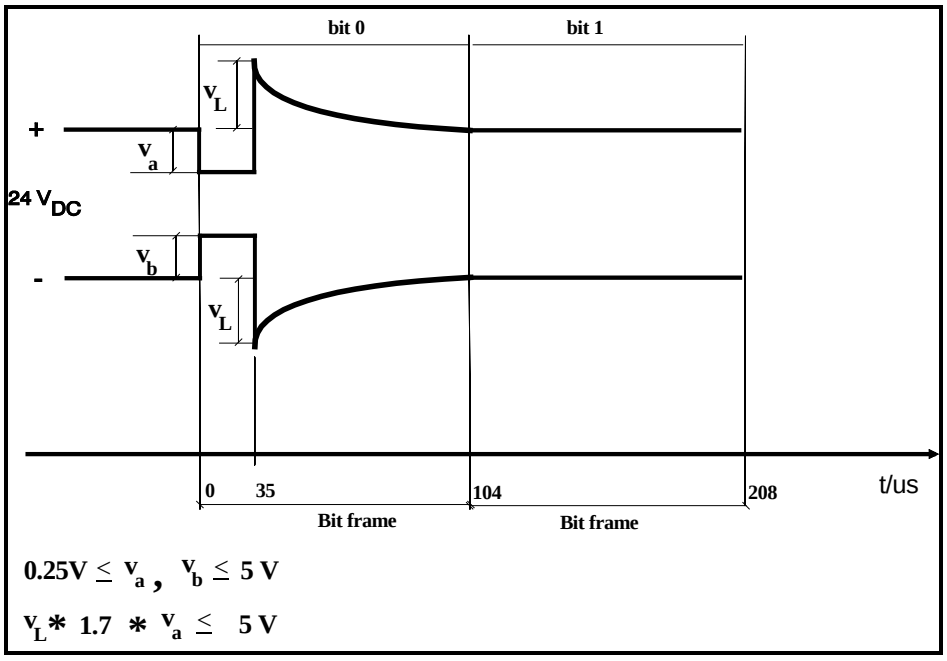
Technical data of the bus line:

Type	Twisted Pair, two pairs
Load resistance per line	max . 37 Ω/km (Loop 74 Ω/km)
Load capacitance line / line	max 100 nF/km (800 Hz)
Screening	Thin strip with drain terminal
Number of twists	min. 5/m
Line diameter	0.8 mm

TP cable which meets the KNX requirements in Volume 9 of the KNX specifications (e.g. YCYM 2x2x0.8 or J-Y(St)Y 2x2x0.8) can be authorised (without the KNX logo) or certified (with the KNX logo) by KNX Association. Only the green standard TP cable guarantees the cable lengths above. The maximum length of all other cables per line segment is indicated in the data sheet of the respective cable.
If distribution boards are used, these may house a data rail in addition to the bus line.

Transmission method used:
Time division multiplexing, baseband, symmetrical

Bit representation:



Absence of DC current, not self-timing.

Transmission speed:
9.6 Kbit/s

Transmission delay:
To ensure collision detection and correction, the transmission delay must not exceed 10 µs. This condition is automatically met if you

- use only the authorised bus cables
- observe the maximum distance of 700 metres between any two bus devices and
- do not install more than 64 bus devices in one electrical segment.

Bus coupling method:
Repeater, symmetrical

Connection techniques used for cables:
Bus terminal (terminal block) with four terminals per line for connection to the bus device.
Overvoltage protection terminal in connection with the bus terminal.
Data rail-to-wire connector for connecting the bus line to the data rail.

Connection technique used for data rails:
Bus devices designed for DIN rail mounting have a contact block containing spring-finger connectors which establish the connection to the conductors of the data rail.

3 KNX Telegram structure

As already indicated above, each layer will contribute to a part of the telegram when a device generates a telegram and will decode part of the telegram when receiving.
Or in other words:

- When receiving a telegram, it will process the relevant information and pass on the rest to the upper layers;
- When creating a telegram, it will insert its relevant information before it passes on the data to the lower layers.

As already indicated, the general structure of a telegram is as follows:

<table>
<tr><td>Control field</td><td>Source address</td><td>Receiver address</td><td colspan="3">N_PDU</td><td>Check field</td></tr>
<tr><td rowspan="2">8 bit</td><td rowspan="2">16 bit</td><td rowspan="2">16 bit</td><td>8 bits</td><td colspan="2">T_PDU</td><td rowspan="2">8 bit</td></tr>
<tr><td></td><td>6 bits</td><td>A_PDU</td></tr>
</table>

The N-PDU, as the part of the LL telegram destined for/created by the Network Layer, contains the following information:

- The **first bit** indicates whether the receiver address needs to be interpreted as a
  - Individual addressed telegram (value 0)
  - Group addressed telegram (value 1)

- The following **three bits** indicates the value of the routing counter, which will play a role in the routing of the telegram across line and backbone couplers
- The following **4 bits** indicate the length of the actual useful information in the telegram (or 'payload')
- The rest of the N-PDU is actually the T-PDU

The T-PDU, as the part of the LL telegram destined for/created by the Transport Layer, contains the following information:

- The first **two bits** indicate the type of Transport Layer communication

Coding	Type of communication
00	**U**nnumbered **D**ata **P**acket (UDP)
01	**N**umbered **D**ata **P**acket (NDP)
10	**U**nnumbered **C**ontrol **D**ata (UCD)
11	**N**umbered **C**ontrol **D**ata (NCD)

- The next **4 bits** indicate – *but only in the case of the communication type numbered* - the sequence number. In the case of unnumbered communication, these bits have no meaning (typically set to 0).
- The rest of the T-PDU is actually the A-PDU

The A-PDU, as the part of the LL telegram destined for/created by the Application Layer, has the following significance:

- In case the first two bits in the T-PDU were of the type *UCD*, the next two bits have the following meaning:
  - If the value is **00**: by means of the telegram, a transport layer point to point connection is opened from the indicated sender to the indicated receiver.
  - If the value is **01**: by means of the telegram, an existing transport layer point to point connection between the indicated sender and the indicated receiver is terminated/broken down.
- In case the two first bits in the T-PDU were of the type NCD, the next two bits have the following meaning:
  - If the value is **10**: by means of the telegram, the transport layer of the indicated sender *positively confirms* to the indicated receiver reception of a previously received telegram.
  - If the value is **11**: by means of the telegram, the transport layer of the indicated sender *negatively confirms* to the indicated receiver reception of a previously received telegram.
- In case the two bits in the T-PDU were of the type UDP or NDP, the following bits indicate the APCI.
  An APCI is a 4 bit code to distinguish the different application layer services. Depending on the indicated 4 bit code, the data that follows has a different meaning.

4 Different APCI Codings

4.1 General

The task of the application layer is to manage the values of the group objects on behalf of the application program, process group telegrams as well as carry out management functions.
In run-time communication (i.e. after configuration), the typical APCIs used are:

APCI	Name
0000	GroupValueRead
0001	GroupValueResponse
0010	GroupValueWrite

These APCIs are used in conjunction in multicast communication, i.e. the target addresses of these telegrams are group addresses, useful data is coded according to the KNX Interworking Data types.

Configuration of bus devices is ensured via management functions, which are typically stored in the ROM of the bus device. For such management functions, typically connection-oriented (point-to-point) communication or broadcast (group address 0/0) is used.

Very commonly used APCIs are:

APCI	Name
0011	IndividualAddrWrite
0100	IndividualAddrRequest
0101	IndividualAddrResponse
0110	AdcRead
0111	AdcResponse
1000	MemoryRead
1001	MemoryResponse
1010	MemoryWrite
1011	UserMessage
1100	MaskVersionRead
1101	MaskVersionResponse
1110	Restart
1111	Escape

Some other specific APCI's are defined to:

- Configure line and backbone couplers
- Attribute keys for (memory) access protection[3]
- Grant authorization to access memory[3]
- Handling of the system ID (domain address) in case of PL devices
- Handling of the serial number of a bus device[3]
- Addressing (properties of) KNX interface objects of a remote device[3]

The KNX interface objects were introduced in BCU 2 or BIM M 112 to enable other devices or tools to learn about the properties of a device without needing detailed knowledge about the device (i.e. without needing to know which memory location in the bus device can access which particular property). KNX interface objects group together values and parameters from one function of a device (in most cases this corresponds to one channel).

Apart from three system interface objects (such as the device object with general device information e.g. order number), each device has a number of application interface objects that have been implemented by the manufacturer (e.g. generally one per channel). The values of properties of these interface objects can also be present as group objects or as an array of elements (in the case of a parameter).

Using specific APCIs, a tool or device can first examine which KNX interface objects a particular device supports. As interface objects have been standardised by KNX Association and have a unique identifier, the tool is able to identify the type of the device. A tool can read out the (partially standardised) properties for each interface object or overwrite them if required (e.g. if the property that determines the dimming speed is found, this value can if required be increased or reduced).
In the case of the services "UserMessage" (APCI 1011) and "Escape" (APCI 1111), the 6 bits following the APCI are to be interpreted as an extension to the APCI.

4.2 Detailed Explanation of Several APCIs

Note: the underneath shaded bits are the respective APCI codings (as given in the above table).

GroupValueRead
Requests the group object(s) of the bus devices that respond to the group address as indicated as target address to transmit the contents of this (or these) object(s).

	00		00	xxxxxx

x...x: no significance[4]

GroupValueResponse
Response to a GroupValueRead, containing the requested data. The target address of this telegram is the *sending* group address that is assigned to the read group object(s).

[3] not supported by BCU1 compliant devices

[4] Typically 0b

a. telegram format for useful data of up to 6 bits:

	00		01	dddddd

b. telegram format for useful data larger than 6 bits:

	00		01	xxxxxx	dddddddd	. . .	Dddddddd

x...x: no significance[4]

d...d: data

GroupValueWrite

Write operation to and from group object(s): The contents of a group object is transmitted onto the bus, together with the *sending* group address assigned, e.g. when a push button sensor is being operated or the contents of a group object is overwritten, if the indicated target address is the assigned group address (e.g. relay switches on or off)..

a. telegram format for useful data of up to 6 bits:

	00		10	dddddd

b. telegram format for useful data larger than 6 bits:

	00		10	xxxxxx	dddddddd	. . .	dddddddd

x...x: no significance[4]

d...d: data

IndividualAddressWrite

Setting the individual address of those bus devices of a KNX installation that are currently in the programming mode (the programming LED of these devices light up). The target address in the telegram is 0/0 (broadcast).

	00		11	xxxxxx	BBBBLLLL	TTTTTTTT

x...x: no significance[4]

BBBBLLLL TTTTTTTT: individual address to be assigned

IndividualAddressRequest

Reads the individual address of all those bus devices within a KNX installation that are currently in the programming mode (the programming LEDs of these devices light up). The target address in the telegram is 0/0 (broadcast).

	01		00	Xxxxxx

x...x: no significance[4]

IndividualAddressResponse
Response to an IndividualAddressRequest of all those bus devices of a KNX installation that are currently in the programming mode (their programming LED lights up). The target address in the telegram is 0/0 (broadcast).

	01	01	Xxxxxx

x...x: no significance[4]

AdcRead
Requests the addressed bus device to carry out a number of A to D conversions, calculate the sum and return the result, of the indicated A to D channel.

	01	10	Aaaaaaa	ccccccc

a...a: number of the A to D channel to be read
c...c: number of A to D conversions to be carried out and summed

AdcResponse
Response to AdcRead containing the requested data.

	01	11	aaaaaa	ccccccc	hhhhhhhh	LLLLLLLL

a...a: number of the A to D channel read
c...c: number of A to D conversions carried out and summed
h...h: high byte of the result
L...L: low byte of the result

MemoryRead
Requests the addressed bus device to transmit its memory contents, starting at the specified address.

	10	00	XXLLLL	aaaaaaaa	bbbbbbbb

L...L: length of the memory area to be read (bytes)
a...a: starting address of the memory area to be read (high byte)
b...b: starting address of the memory area to be read (low byte)

MemoryResponse
Response to MemoryRead containing the requested data.

10	01	XXLLLL	aaaaaaaa	bbbbbbbb	dddddddd	. . .	dddddddd

L...L: length of the memory area read (bytes)
a...a: starting address of the memory area read (high byte)
b...b: starting address of the memory area read (low byte)
d...d: contents of the memory area read

If part of the requested data lies in a protected memory area or does not exist, then the bus device returns the message L...L =0 and no data at all.

Memory Write
Writes a continuous data block into the memory of the addressed bus device, starting at the specified address.

10	10	XXLLLL	aaaaaaaa	bbbbbbbb	dddddddd	. . .	dddddddd

L...L: number of bytes to be written
a...a: target address of the 1st byte to be written (high byte)
b...b: target address of the 1st byte to be written (low byte)
d...d: data to be written
If part of the requested data lies in a protected memory area or does not exist, then the addressed bus device ignores the write request.
Some bus devices automatically emit a MemoryResponse immediately after the write instruction (mode called 'verify').

User Message
Initiates the exchange of data between two microcontrollers in the bus device[5].

10	11	dddddd	. . .	Dddddddd

d...d: data

MaskVersionRead
Requests the addressed bus device to return information on its supported system profile by means of its mask type and mask version.

11	00	Xxxxxx

x...x: no significance[4]

[5] Rather an exception in current devices

MaskVersionResponse
Response to MaskVersionRead containing the requested data.

	11	01	xxxxxx	mmmmmmmm	vvvvuuuu

m...m: mask type (first nibble indicates medium, second nibble indicates main profile)
v..vu..u: mask version (first nibble main version, second nibble subversion)
For more information on the existing system profiles, see appropriate paragraphs in Basic Course, Part 'Bus devices'
x...x: no significance[4]

Restart
Resets the bus device of the addressed bus device. Whether or not the application module, too, is reset, depends on the respective bus device to which the message was sent.

	11	10	xxxxxx

x...x: no significance[4]

Escape
In case the APCI has the underneath value, the exact meaning of the APCI is indicated in the six bits following the Escape APCI.

	11	11	eeeeee

e...e: extended APCI bits

M_BitWrite (in case of eee eee = 010 000)
This service allows individual bits in the memory to be manipulated: In this way it is possible to change up to 1 to 48 bits in the memory of the bus device

M_AuthorizeRequest/Response (in case of eee eee = 010 001, respectively 010 010)
These services allow accessing a bus device with memory access-protection. 16 different access levels are possible. A 32 bit number (FFFF FFFF) is required to be granted access to memory. If no access protection is used, the number remains at FFFF FFFF and all the access levels are enabled.

The process is started by an M_ AuthorizeRequest message which contains the number. The device that receives the message compares the number with its table and enables the corresponding access levels. If the number is not in the table, the device disables all memory access. The bus device replies with an M_ AuthorizeResponse; this reply contains the information about to which level access has been granted.

M_SetKeyRequest/Response (in case of eee eee = 010 011 respectively 010 100)
These services allow to write keys for the various access levels to bus devices with memory access protection (16 levels are possible, 0 is the highest level and 15 is the lowest).
A code is set by an M_SetKeyRequest, indicating the key for the respective access level. The receiving device tests whether the current access level is at least as high as the one that is to receive the indicated key. If that is the case, the code is set and an M_SetKeyResponse is returned together with the access level. If the test proves negative, an M_SetKeyResponse is returned together indicating the error code FFh.

5 Interpretation of a telegram sequence: Allocation of an individual address

Explanation of the print-out in the following pages showing the recorded telegrams when downloading an individual address in a bus device. The columns Date, Time and Priority are not shown.

No.0	Using a Transport Layer Open, the ETS program tries to establish a transport layer connection to the bus device with the individual address 1.1.2, i.e. it attempts to establish a point-to-point connection.
No.1-3	No device responds with an IACK message and the telegram is therefore repeated three times.
No.4	The ETS tries to read the Mask version of the bus device with individual address 1.1.2.
No.5-7	No bus device responds with an IACK. For this reason the telegram is repeated three times.
No.8	The attempt to open a Transport layer P2P connection by the ETS program is broken down again by means of a Transport Layer Close Telegram.
No.9-11	As no IACK is received in the agreed time interval, the telegram is repeated three times.
No.12-15	It can now be assumed from the above that the individual address 1.1.2 does not yet exist in the KNX installation. The ETS program can now proceed with programming this address. ETS now sends an IndividualAddressRead Request Telegram cyclically, requesting those bus devices whose programming LED lights up to return their individual address. The target address of this telegram is 0/0/0 (reserved address for broadcast); therefore all bus devices connected to the bus simultaneously respond with an IACK message. (The IACK Telegram, the telegram code in Hex CC, is marked as ACK in the column “Type” of the print-out).
No.16-17	As the programming button has been pressed at bus device with individual address 3.1.4, it will respond to the IndividualAddressRead Request Telegram with an IndividualAddressResponse Telegram. The source address contained in this telegram is the device’s current individual address; the target address is the group address 0/0/0 (broadcast). An IACK follows.

No.18-19 The ETS sends an IndividualAddressWrite Telegram. The KNX device that is in the programming mode adopts the transmitted address as its new address and acknowledges with an IACK message.

No.20-21 The ETS program attempts to open a P2P connection to the bus device with the address 1.1.2 using a Transport Layer Open Telegram. As this address is now available in the KNX installation, the addressed bus device returns the acknowledgement IACK (layer 2 acknowledgement).

No.22 The ETS requests the bus device that has just been addressed to send information on its mask type and mask version. The sequence number of the data packet is 0.

No.23 Upon arrival of the telegram, the device sends the acknowledgement IACK.

No.24-25 The bus device 1.1.2 confirms the arrival of the data telegram by returning a Transport Layer acknowledgement telegram (layer 4 acknowledgement) to the PC. The PC's KNX serial interface acknowledges with IACK (layer 2 acknowledgement).

No.26 The bus device responds to the Mask Version Read by returning a Mask Version Response. The PC confirms with IACK. This ACK is not recorded (timing problems in the recording bus device)

No.27-28 The PC confirms the arrival of the telegram by returning a Transport Layer acknowledgement telegram (layer 4 acknowledgement) to the device with address 1.1.2 (has gone missing here). The latter acknowledges with ACK.

No.29-30 The ETS program resets device no. 1.1.2 (with sequence number 1). Because of this the Program LED is switched off. The device returns an IACK acknowledgement

No.31 The device returns a Transport Layer acknowledgement. The IACK acknowledgement of the PC has not been recorded.

No.32-33 The ETS closes the P2P connexion with bus device 1.1.2. The bus devices respond with a IACK.

No.34,35. The ETS tries to read a System ID (Powerline)

No.36,37 The ETS tests whether any programming LEDs are still on. All bus devices respond with IACK.

No.38-80 The ETS checks the newly programmed bus device. In order to this, a P2P connexion will be established two times and the Manufacturer code (EEPROM address 0x0104), the data direction of Port A (EEPROM Address 0x010C) and the impedance (Port 4 of ADC) of the possibly plugged-in application module will be assessed.

Nr.	Telegramm	Wiederholung	Quelladresse	Zieladresse	Typ	Pakettyp	Wert
0	B0 11 FA 11 02 60 80 57		01.01.250	01.01.002	Kontroll	Kontrollpaket	Öffnen
1	90 11 FA 11 02 60 80 77	Ja	01.01.250	01.01.002	Kontroll	Kontrollpaket	Öffnen
2	90 11 FA 11 02 60 80 77	Ja	01.01.250	01.01.002	Kontroll	Kontrollpaket	Öffnen
3	90 11 FA 11 02 60 80 77	Ja	01.01.250	01.01.002	Kontroll	Kontrollpaket	Öffnen
4	B0 11 FA 11 02 61 43 00 95		01.01.250	01.01.002	Maskenversion lesen (verbindungsorientiert)	Numeriertes Datenpaket (0)	
5	90 11 FA 11 02 61 43 00 B5	Ja	01.01.250	01.01.002	Maskenversion lesen (verbindungsorientiert)	Numeriertes Datenpaket (0)	
6	90 11 FA 11 02 61 43 00 B5	Ja	01.01.250	01.01.002	Maskenversion lesen (verbindungsorientiert)	Numeriertes Datenpaket (0)	
7	90 11 FA 11 02 61 43 00 B5	Ja	01.01.250	01.01.002	Maskenversion lesen (verbindungsorientiert)	Numeriertes Datenpaket (0)	
8	B0 11 FA 11 02 60 81 56		01.01.250	01.01.002	Kontroll	Kontrollpaket	Schli&eßen
9	90 11 FA 11 02 60 81 76	Ja	01.01.250	01.01.002	Kontroll	Kontrollpaket	Schli&eßen
10	90 11 FA 11 02 60 81 76	Ja	01.01.250	01.01.002	Kontroll	Kontrollpaket	Schli&eßen
11	90 11 FA 11 02 60 81 76	Ja	01.01.250	01.01.002	Kontroll	Kontrollpaket	Schli&eßen
12	B0 11 FA 00 00 E1 01 00 44		01.01.250	00/0	Adress Anfrage (Broadcast)	Datenpaket	
13	CC				ACK		
14	B0 11 FA 00 00 E1 01 00 44		01.01.250	00/0	Adress Anfrage (Broadcast)	Datenpaket	
15	CC				ACK		
16	B0 31 04 00 00 E1 01 40 DA		03.01.004	00/0	Adress Rückmeldung (Broadcast)	Datenpaket	
17	CC				ACK		
18	B0 11 FA 00 00 E3 00 C0 11 02 94		01.01.250	00/0	Physikalische Addresse (Broadcast)	Datenpaket	01.01.002
19	CC				ACK		
20	B0 11 FA 11 02 60 80 57		01.01.250	01.01.002	Kontroll	Kontrollpaket	Öffnen
21	CC				ACK		
22	B0 11 FA 11 02 61 43 00 95		01.01.250	01.01.002	Maskenversion lesen (verbindungsorientiert)	Numeriertes Datenpaket (0)	
23	CC				ACK		
24	B0 11 02 11 FA 60 C2 15		01.01.002	01.01.250	Kontroll	Numeriertes Kontrollpaket (0)	ACK
25	CC				ACK		
26	B0 11 02 11 FA 63 43 40 00 12 C5		01.01.002	01.01.250	Maskenversion Rückmeldung (verbindungsorientiert)	Numeriertes Datenpaket (0)	Maskenversion: 0x0012
27	B0 11 FA 11 02 60 C2 15		01.01.250	01.01.002	Kontroll	Numeriertes Kontrollpaket (0)	ACK
28	CC				ACK		
29	B0 11 FA 11 02 61 47 80 11		01.01.250	01.01.002	Neustart (verbindungsorientiert)	Numeriertes Datenpaket (1)	
30	CC				ACK		
31	B0 11 02 11 FA 60 C6 11		01.01.002	01.01.250	Kontroll	Numeriertes Kontrollpaket (1)	ACK
32	B0 11 FA 11 02 60 81 56		01.01.250	01.01.002	Kontroll	Kontrollpaket	Schli&eßen
33	CC				ACK		
34	B0 11 FA 00 00 E1 03 E1 A7		01.01.250	00/0	System ID lesen	Datenpaket	
35	CC				ACK		
36	B0 11 FA 00 00 E1 01 00 44		01.01.250	00/0	Adress Anfrage (Broadcast)	Datenpaket	
37	CC				ACK		
38	B0 11 FA 11 02 60 80 57		01.01.250	01.01.002	Kontroll	Kontrollpaket	Öffnen
39	CC				ACK		
40	B0 11 FA 11 02 61 43 00 95		01.01.250	01.01.002	Maskenversion lesen (verbindungsorientiert)	Numeriertes Datenpaket (0)	
41	CC				ACK		
42	B0 11 02 11 FA 60 C2 15		01.01.002	01.01.250	Kontroll	Numeriertes Kontrollpaket (0)	ACK
43	CC				ACK		
44	B0 11 02 11 FA 63 43 40 00 12 C5		01.01.002	01.01.250	Maskenversion Rückmeldung (verbindungsorientiert)	Numeriertes Datenpaket (0)	Maskenversion: 0x0012

Nr.	Telegramm	Wiederholung	Quelladresse	Zieladresse	Typ	Pakettyp	Wert
45	B0 11 FA 11 02 60 C2 15		01.01.250	01.01.002	Kontroll	Numeriertes Kontrollpaket (0)	ACK
46	CC				ACK		
47	B0 11 FA 11 02 63 46 01 01 04 96		01.01.250	01.01.002	Speicher lesen (verbindungsorientiert)	Numeriertes Datenpaket (1)	Anzahl Bytes: 1 - Adresse 0x0104
48	CC				ACK		
49	B0 11 02 11 FA 60 C6 11		01.01.002	01.01.250	Kontroll	Numeriertes Kontrollpaket (1)	ACK
50	B0 11 02 11 FA 64 46 41 01 04 01 D0		01.01.002	01.01.250	Speicher Rückmeldung (verbindungsorientiert)	Numeriertes Datenpaket (1)	Anzahl Bytes: 1 - Adresse 0x0104 - Daten: 01
51	CC				ACK		
52	B0 11 FA 11 02 60 C6 11		01.01.250	01.01.002	Kontroll	Numeriertes Kontrollpaket (1)	ACK
53	CC				ACK		
54	B0 11 FA 11 02 63 4A 01 01 0C 92		01.01.250	01.01.002	Speicher lesen (verbindungsorientiert)	Numeriertes Datenpaket (2)	Anzahl Bytes: 1 - Adresse 0x010C
55	CC				ACK		
56	B0 11 02 11 FA 60 CA 1D		01.01.002	01.01.250	Kontroll	Numeriertes Kontrollpaket (2)	ACK
57	B0 11 02 11 FA 64 4A 41 01 0C 00 D5		01.01.002	01.01.250	Speicher Rückmeldung (verbindungsorientiert)	Numeriertes Datenpaket (2)	Anzahl Bytes: 1 - Adresse 0x010C - Daten: 00
58	CC				ACK		
59	B0 11 FA 11 02 60 CA 1D		01.01.250	01.01.002	Kontroll	Numeriertes Kontrollpaket (2)	ACK
60	CC				ACK		
61	B0 11 FA 11 02 60 81 56		01.01.250	01.01.002	Kontroll	Kontrollpaket	Schli&eßen
62	CC				ACK		
63	B0 11 FA 11 02 60 80 57		01.01.250	01.01.002	Kontroll	Kontrollpaket	Öffnen
64	CC				ACK		
65	B0 11 FA 11 02 61 43 00 95		01.01.250	01.01.002	Maskenversion lesen (verbindungsorientiert)	Numeriertes Datenpaket (0)	
66	CC				ACK		
67	B0 11 02 11 FA 60 C2 15		01.01.002	01.01.250	Kontroll	Numeriertes Kontrollpaket (0)	ACK
68	B0 11 02 11 FA 63 43 40 00 12 C5		01.01.002	01.01.250	Maskenversion Rückmeldung (verbindungsorientiert)	Numeriertes Datenpaket (0)	Maskenversion: 0x0012
69	CC				ACK		
70	B0 11 FA 11 02 60 C2 15		01.01.250	01.01.002	Kontroll	Numeriertes Kontrollpaket (0)	ACK
71	CC				ACK		
72	B0 11 FA 11 02 62 45 84 0A 1E		01.01.250	01.01.002	ADC lesen (verbindungsorientiert)	Numeriertes Datenpaket (1)	Port Nr.: 4 - Anzahl der Wiederholungen: 10
73	CC				ACK		
74	B0 11 02 11 FA 60 C6 11		01.01.002	01.01.250	Kontroll	Numeriertes Kontrollpaket (1)	ACK
75	B0 11 02 11 FA 64 45 C4 0A 00 00 58		01.01.002	01.01.250	ADC Rückmeldung (verbindungsorientiert)	Numeriertes Datenpaket (1)	Port Nr.: 4 - Anzahl der Wiederholungen: 10 - Summe der Werte 0
76	CC				ACK		
77	B0 11 FA 11 02 60 C6 11		01.01.250	01.01.002	Kontroll	Numeriertes Kontrollpaket (1)	ACK
78	CC				ACK		
79	B0 11 FA 11 02 60 81 56		01.01.250	01.01.002	Kontroll	Kontrollpaket	Schli&eßen
80	CC				ACK		

6 KNX Easy Installation Principles

6.1 General

ETS is the best known and by far most used way to design and program a KNX installation. This is called System Mode (S-Mode). KNX also offers alternatives: KNX devices can also be configured without ETS. This is called *Easy Mode* (E-Mode) and the devices are E-Mode devices. There exist three submodes: *Controller Mode* (Ctrl-Mode), *Push Button Mode* (PB-Mode) and *Logical Tag Extended Mode* (LTE-Mode).

6.2 E-Mode channels

KNX guarantees the compatibility and runtime Interworking independently of the configuration mode by fixing elementary functions in so-called *Functional Blocks*. These specify the behaviour and coding (Datapoint Types), but still allow for a certain freedom. LTE-Mode even directly bases on Functional Blocks for the description, linking and communication between its devices.
Ctrl-Mode and PB-Mode however base their description and linking on a standard selection of Datapoints and Parameters of one or more Functional Blocks. These are called *E-Mode Channels*. Every Group Object additionally gets one of more identifiers that specify its functionality (e.g. switch, time, temperature). These are called *Connection Codes*. During the configuration, Group Objects with the same Connection Code can be linked.

6.2.1 Ctrl-Mode

In Ctrl-Mode, one single, central controller takes over the role of ETS. This controller assigns a unique Individual Address to each E-Mode device that it finds. A free Individual Address is searched in the same way as ETS checks whether an Individual Address is occupied or not. Next, of all devices, the device capabilities (E-Mode Channels and parameters) are read out and shown on the display, so that the installer can select the functionality and can indicate which channels should be linked. The controller decides on the Group Addresses and parameter values and downloads the devices, again much in a way as done by ETS, as shown in the underneath figure.

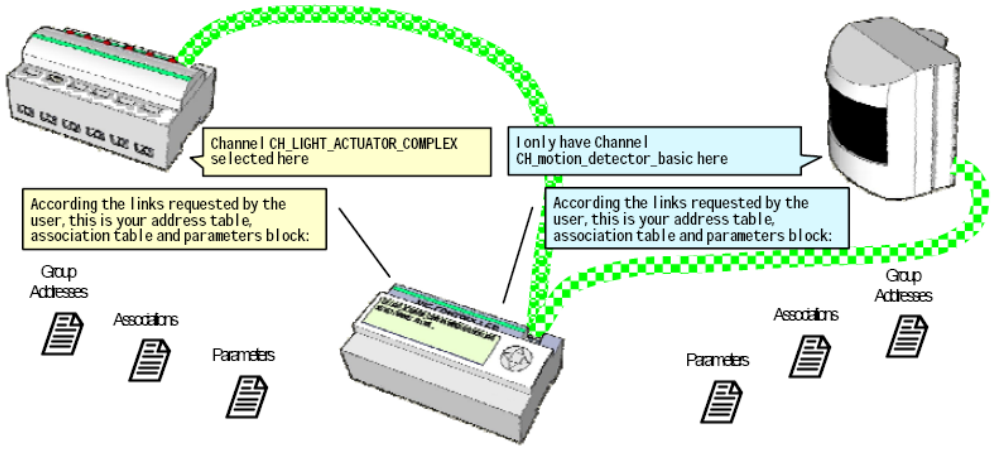


Figure 6: Ctrl-Mode

6.2.2 PB-Mode

In PB-Mode, the configuration starts with taking Individual Addresses. Every separate device searches for itself a free Individual Address. Even further on, no 3rd party - like ETS or a central controller - is needed. Instead, the PB-Mode themselves configure each other. The installer firstly selects an actuator Channel (typically by pressing a push button on it) and then a sensor E-Mode Channel. The underneath figure sketches how these two Channels (devices) then negotiate whether or not a link shall be established, and the parameters and Group Addresses to be used for this. The installer can repeat the procedure for this or another actuator Channel and other sensor Channels, always two by two. The devices do not write each other's Address Table or parameter memory. Instead, PB-Mode device have some dedicated services and commands to do this without knowing the mask and memory map of the communication partner.

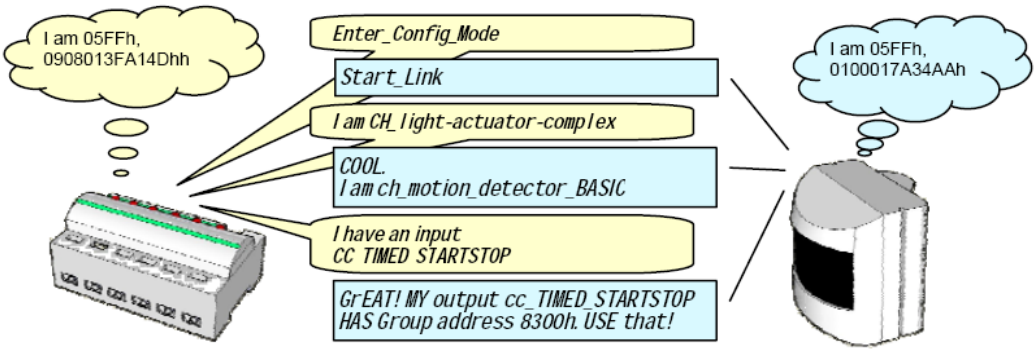


Figure 7: PB-Mode

6.2.3 LTE-Mode

In LTE-Mode, the application model is fixed: the relation of any device to any other device is defined on beforehand; the binding is “static”. This is originally designed for HVAC applications.
The LTE-Mode devices (or Channels) only need to be assigned to a *zone*: this is the information about the location (e.g. room, floor) or sometimes about the functionality to which the transmitted data belongs. If the same functionality is installed twice or more, then different zones are used. The zone is a structured piece of information; an output uses it as the destination address in its telegrams. An input only listens to the telegrams sent on the zone(s) to which it is assigned itself.
LTE-Mode does not use the classic Group Objects. Instead, it uses Properties. Also the frame format differs from the well-known group telegrams. Consequently, LTE telegrams cannot be understood by other KNX devices; therefore, LTE-devices additionally exhibit classic Group Objects that can be linked with ETS and provide a rich subset with the same data.

6.3 Usability of E-Mode

KNX E-Mode devices are available both for KNX TP and for KNX RF; so far, no implementations have been launched for KNX PL or KNX IP.

6.3.1 Installation size

Ctrl-Mode and PB-Mode installations are in size limited to a single Line. With more devices, the overview would easily be lost. It would also require dedicated Couplers and a higher computational power in the devices and the Controller.
LTE-installation can exceed the size of a Line. The newer Couplers have parameters to allow the filtering based on zoning information (next to the Filter Table).

6.3.2 Co-existence with S-Mode

E-Mode Lines separate themselves from S-Mode Lines by using a reserved Line Address (0.2 for TP and 0.5 for RF).
Conflicts in the use of Group Addresses are avoided by questioning network wide whether a Group Address is used (PB-Mode on TP) or using predefined extended Group Addresses (these are Group Addresses also containing the unique KNX Serial Number, used in KNX RF). Possibly, the Group Address values can be used from a reserved range.

6.4 Applications and Interworking

6.4.1 Applications

The more than hundred already defined E-Mode Channels support applications beyond simple input/output commands. E-Mode Channels are available for the following functions: control of lights and shutters and blinds, scenes, weather sensors, logical gates, clocks, room temperature control, valve control and many other.

6.4.2 Runtime - and Configuration Interworking

The common Functional Block basis guarantees that at runtime S-Mode, Ctrl-Mode, E-Mode and the Group Objects of LTE-Mode use the same group messages with information encoded in the same way. However, at Configuration time the fundamentally different strategies for linking in Ctrl-Mode, PB-Mode and LTE-Mode make that these devices cannot be linked directly to one another. This can only be achieved by reading out the E-Mode installation and re-using or modifying the Group Addresses. This functionality is planned for ETS 4.