RS-232 (serial) interface when programming (physical addresses, application programs) the bus devices using ETS 5. You can therefore remove it afterwards, because the system does not need it to operate.

3.7 Addressing Nodes (Devices)

All KNX devices that communicate (sensors, actuators, couplers, etc.) must have a unique physical address. Nodes also belong to a group so that they can exchange data with each other. The nodes in such a group have a group address in addition to their physical address, which can be used to communicate with them.

3.7.1 Physical Addresses

Each bus device (except the power supply) is assigned a unique physical address. This address is then loaded into the device (node) and stored permanently in its EEPROM during commissioning using ETS 5.

Addresses should be assigned at random, but should correspond to the layout of the building’s installation. In other words, nodes (devices) that are located next to each other should be assigned consecutive physical addresses, for example, 1.1.1, 1.1.2, 1.1.3, and so on. The physical addresses must be clearly and permanently labeled on the devices.

A device’s physical address clearly identifies it and, at the same time, provides the following information on the device’s topographical position within the whole system:

Area.Line.Node (A.L.N.)

The area, line and node parts of the physical address are separated by periods. This way you can distinguish between a physical address and a group address, which uses a slash (/). For a physical address, 2 bytes are available in a data frame as seen in Table 3.4.

<table>
<thead>
<tr>
<th>Table 3.4 Source address in a data frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>High byte</td>
</tr>
<tr>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
</tr>
<tr>
<td>A3 A2 A1 A0 L3 L2 L1 L0</td>
</tr>
<tr>
<td>Area</td>
</tr>
</tbody>
</table>
There can be
- $2^4 = 16$ areas
- $2^4 = 16$ lines
- $2^8 = 256$ nodes per line

giving a total of $2^{16} = 65,536$ nodes. However, this maximum capacity is rarely used. In particular, apart from couplers no other devices are connected to the main lines and the backbone line. This minimizes the delay to data transmitted across lines and areas caused by the internal data traffic from the nodes on a main line or the backbone.

The physical addresses are usually limited to:
- 15 areas (numbers 1–15)
- 12 lines per area (numbers 1–12)
- 64 nodes per line (63 sensors/actuators with numbers 1–63 and one line coupler with the node number 0)

This means that a maximum of $63 \times 12 \times 15 = 11,340$ sensors/actuators, plus $12 \times 15 = 180$ LCs and 15 BCs, giving a total of 11,535 nodes (Fig. 3.20).

This is more than enough for most projects in residential and commercial buildings. The majority of KNX projects usually only have a few dozen to a few hundred nodes.

### 3.7.1.1 Physical Addresses for Couplers and Line Repeaters

The following physical addresses are reserved for line and backbone couplers:
- A.L.0 for line couplers (e.g. 1.1.0, 1.2.0, ..., 1.12.0, 2.1.0, 2.2.0, ..., 15.12.0)
- A.0.0 for backbone couplers (e.g. 1.0.0, 2.0.0, ..., 15.0.0)

Line repeaters must be assigned a node number that is greater than zero, for example, 1.1.64.

### 3.7.1.2 Physical Addresses for Main Lines and Backbone Lines

#### Physical Adresses of Devices Connected to a Main Line

Sensors and actuators are not normally connected to the main line. This is because the resultant data traffic would interfere with the data traffic across the lines or areas.

As a result, a main line’s power supply unit usually only needs to supply one backbone coupler with power. For this reason, a maximum of 63 sensors/actuators can be connected to the main line. They are assigned the physical addresses A.0.x, where $x$ can vary from 1 to 255. The address A.0.0 is reserved for the backbone coupler.
Examples of addresses:

- 1.0.1–1.0.63
- 2.0.100–2.0.162.

If the KNX system comprises only one area and, therefore, does not need a backbone coupler, then a maximum of 64 sensors/actuators can be connected to the main line.

**Physical Adresses of Devices connected to a Backbone Line**

As with the main lines, sensors and actuators are not normally connected to the backbone line. This is because the resultant data traffic would interfere with the cross-area data traffic.

As a result, a backbone line’s power supply unit does not normally have to supply a bus device with power. For this reason, a maximum of 64 sensors/actuators can be connected to the backbone line. They are assigned the physical addresses 0.0.x, where x can vary from 1 to 255.

Examples of addresses:

- 0.0.1–0.0.64
- 0.0.100–0.0.163.

### 3.7.1.3 Examples of Physical Addresses

- The address 1.2.2 refers to the second node on the second line in the first area.
- 1.12.0 is a line coupler that connects the 12th line in the 1st area to the 1st area’s main line. The main line is referred to as the superordinate line and the line is referred to as the subordinate line.
- 2.0.0 represents a backbone coupler that connects the second area’s main line to the backbone line. The backbone line is the superordinate line and the main line is the subordinate line.

### 3.7.2 Group Addresses (Logical Addresses)

When commissioning a KNX system using ETS 5, specific devices must be addressed or programmed. For this reason, data frames are sent using a destination device’s unique physical address as the destination address. During the normal operating cycle of a KNX system data frames are sent using group addresses.
KNX distinguishes between two kinds of group addresses:

- Group addresses with a main group and a subgroup (two-level addressing)
- Group addresses with a main group, middle group, and subgroup (three-level addressing)

You can use ETS 5 to select the desired addressing type. A 16-bit field in the data frame is reserved for the group address, although only 15 bits are used.

### 3.7.2.1 Two-Level Addressing

For two-level addressing, the available 15 bits are used as follows (Table 3.5).

This means that

- \(2^4 = 16\) main groups (numbers 0–15)
- \(2^{11} = 2048\) subgroups (numbers 0–2047)

are possible. The group address for two-level addressing is written as

```
Main group/subgroup
```

In order to be able to identify a certain group, these main groups and subgroups are assigned names.

Examples for group addresses:

- 0/1 lightning central on/off
- 1/1 lightning living room on/off
- 1/2 lightning office on/off
- 2/1 blinds down/up.

### 3.7.2.2 Three-Level Addressing

For three-level addressing, the available 15 bits are used as follows (Table 3.6).

This means that

- \(2^4 = 16\) main groups (numbers 0–15)
- \(2^3 = 8\) middle groups (numbers 0–7)
- \(2^8 = 256\) subgroups (numbers 0–255)

are possible. The group address for three-level addressing is written as

```
Main group/Middle group/Subgroup
```

<table>
<thead>
<tr>
<th>Table 3.5</th>
<th>Two-level addressing</th>
</tr>
</thead>
<tbody>
<tr>
<td>High byte</td>
<td>Low byte</td>
</tr>
<tr>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
</tr>
<tr>
<td>0 M3 M2 M1 M0 S10 S9 S8</td>
<td>S7 S6 S5 S4 S3 S2 S1 S0</td>
</tr>
</tbody>
</table>

| Main group | Subgroup |
### Table 3.6 Three-level addressing

<table>
<thead>
<tr>
<th>High byte</th>
<th>Low byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7  D6  D5 D4  D3  D2  D1  D0</td>
<td>D7  D6  D5 D4  D3  D2  D1  D0</td>
</tr>
<tr>
<td>0  M3  M2  M1  M0  G2  G1  G0</td>
<td>S7  S6  S5  S4  S3  S2  S1  S0</td>
</tr>
</tbody>
</table>

Main group/middle group/subgroup

Examples for group addresses:
- 1/1/1 lightning living room ceiling on/off
- 1/1/2 lightning living room standing lamp on/off
- 1/2/1 lightning office ceiling on/off
- 1/2/2 lightning office desk on/off.

### 3.7.2.3 Main Groups 14 and 15

With both two and three-level addressing, there is not enough room in the couplers’ filter tables for the main groups 14 and 15 due to the limited amount of memory in the EEPROM. As a result, they are not usually included in the project design. If they are included, then the couplers must be configured accordingly.

The main group 0 is usually reserved for alarm functions, whereas the main groups 1–13 are reserved for various systems such as lighting and shutters/blinds.

### 3.8 Communication Objects

#### 3.8.1 Definition

A communication object is a memory area that is used for data exchange with other applications or devices by the application software of a KNX device in combination with the communication software (see Fig. 3.23).

#### 3.8.2 Characteristics of Communication Objects

##### 3.8.2.1 Structure

A communication object (C.Obj.) has a certain structure. For example it can be:
- a bit field (length 1 bit, 4 bit, 8 bit etc.)
- a variable (integer, float)
• a time or date
• a text (e.g. 14 ASCII letters).

3.8.2.2 Attributes

For each application program there is a certain number of C.Obj. that can be found in the device’s documentation. A C.Obj. has certain attributes. For example:

• number
• name
• function
• group address
• length (1 bit, 4 bit, 1 B etc.)
• flags: communication (C), read (R), write (W), transmit (T), update (U), read on initialization (I)

The flags of a C.Obj. are usually predefined (default settings) and should only be changed in special cases. For sensors and actuators, every flag besides the L-flag is usually already set. Only some devices have an I-flag.

Generally, the flags influence the access to the C.Obj. The meaning of the flags can be found in Table 3.7.

3.8.2.3 Access Methods

The C.Obj. is accessed by the ETS5, the system software and the application with certain services for reading and writing. A commonly used service in the application layer is e.g. A_GroupValue_Write, which is used to write the object value.
3.8.3 Communication Objects in Sensors

Sensor applications normally only use communication objects for sending data frames. This is shown by the words Telegr. in the function assigned to the communication object, for example, Telegr.switch or Telegr. relative dimming. As an example, Table 3.8 lists the communication objects for the Switch Dim LED application of a four-gang switch sensor in the switch/dimming sensor setting [9].

In the switch/dimming sensor default setting, the Switch Dim LED application has eight communication objects for four push buttons that are used to carry out the following functions [9]:

<table>
<thead>
<tr>
<th>No.</th>
<th>Type (bit)</th>
<th>Object name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Left push button—short</td>
<td>Telegr.switch</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Mid left push button—short</td>
<td>Telegr.switch</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Mid right push button—short</td>
<td>Telegr.switch</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Right push button—short</td>
<td>Telegr.switch</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Left push button—long</td>
<td>Telegr. relative dimming</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>Mid left push button—long</td>
<td>Telegr. relative dimming</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>Mid right push button—long</td>
<td>Telegr. relative dimming</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>Right push button—long</td>
<td>Telegr. relative dimming</td>
</tr>
</tbody>
</table>

Table 3.7 Flags

<table>
<thead>
<tr>
<th>Flag (Communication)</th>
<th>Flag set</th>
<th>Flag not set</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C.Obj. is connected to the bus</td>
<td>Acknowledgement of frames but C.Obj. is not changed</td>
</tr>
<tr>
<td>R (Read)</td>
<td>Value of C.Obj. can be read by bus</td>
<td>Value of C.Obj. can not be read by bus</td>
</tr>
<tr>
<td>W (Write)</td>
<td>Value of C.Obj. can be changed by bus</td>
<td>Value of C.Obj. can not be changed by bus</td>
</tr>
<tr>
<td>T (Transmit)</td>
<td>If the object’s value changes (for a sensor), the new value is transmitted by bus</td>
<td>Object value is transmitted by bus only during read commands</td>
</tr>
<tr>
<td>U (Update)</td>
<td>Objects value is updated by a value answer frame on the bus (service A_Value_Response). Value answer frame is a reaction to a value reading frame (service A_Value_Read) sent by e.g. a visualization</td>
<td>Objects value is not updated</td>
</tr>
<tr>
<td>I (Read on initialization)</td>
<td>C.Obj. reads the object’s value from the bus during initialization (only some devices)</td>
<td>C.Obj. does not read the objects value during initialization</td>
</tr>
</tbody>
</table>

Table 3.8 The communication objects for the Switch Dim LED function in a four-gang switch sensor
• Pressing and releasing the push buttons upper contact (the corresponding C-Obj. then contains a “1”) sends a switch ON data frame. Pressing and releasing the push button’s lower contact (the corresponding C-Obj. is assigned a “0”) sends a switch OFF data frame.
• Pressing and holding down a push button’s upper contact sends a brighter-dimming data frame. Pressing and holding the lower contact sends a darker-dimming data frame. As soon as the push button is released, a stop dimming data frame is sent.

3.8.4 Communication Objects in Actuators

Actuator applications normally only use communication objects to receive data frames and to execute functions. As an example, Table 3.9 shows the communication objects for the Switch Default Staircase function application of a six-gang switch actuator in normal operation mode [9].

They are used to process the following functions [9]:
• If the Switch function parameter has been set to normally opened contact, then the actuator switches the relay on when it receives a data frame with the value “1” (the corresponding C.Obj. then contains a “1”) and switches it off when it receives a data frame with the value “0” (the corresponding C.Obj. then contains a “0.”
• If the Switch function parameter has been set to normally closed contact, then the actuator switches the relay on when it receives a data frame with the value “0” and off when it receives a one with the value “1.”

3.8.5 Assigning Communication Objects to Group Adresses

3.8.5.1 General Rules

In order to realize a function like switching or dimming, the application of at least one sensor and the application of at least one actuator have to exchange data.

Table 3.9 The communication objects for the Switch Default Staircase function/3 in a six-gang switch actuator

<table>
<thead>
<tr>
<th>No.</th>
<th>Type (bit)</th>
<th>Object name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Output A</td>
<td>Switch</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Output B</td>
<td>Switch</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Output C</td>
<td>Switch</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Output D</td>
<td>Switch</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Output E</td>
<td>Switch</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Output F</td>
<td>Switch</td>
</tr>
</tbody>
</table>
Therefore both the transmitting and the receiving applications use a number of C.Obj. that have to share one group address. This is equivalent to a connection of the devices.

Assuming that nodes A, B and C are sensors and that nodes D, E and F are actuators with the respective C.Obj. as seen in Fig. 3.24.

In order to realize two different functions with ETS5, you need to create group addresses (1/1 and 1/2) that will be used to carry out the desired functions. The communication objects for all the bus devices required for the function must then be assigned to these group addresses. This can be done by dragging and dropping. Also note that the communication objects that are to be assigned together must all be the same type (length), for example, all 1 bit, 4 bit or 1 B. ETS 5 will not allow you to assign them together if they are not all of the same type.

In general, the following rules apply:

- A group address needs to include at least one transmitting and at least one receiving C.Obj.
- A transmitting C.Obj. can only be assigned to one group address.
- A receiving C.Obj. can be assigned to multiple group addresses.

Like the number of communication objects that can be assigned to a group address, there are only a limited number of communication objects and group addresses that can be used by an application. This information can be found in the technical data documentation that comes with each KNX device. An example is shown in Fig. 3.25. The number of communication objects varies depending on the application and the parameters that have been set. The number of functions assigned to the communication objects also varies from application to application.
3.8.5.2 Example

The following function is to be executed using KNX: The left push button of a four-gang switch sensor is to turn two lamp circuits on and off. Pressing the push button’s upper contact should switch all the lamps on and pressing the lower contact should turn them all off. A six-gang switch actuator’s outputs A and B will switch the lamp circuits.

In order to know, which C.Obj. are available, the correct applications have to be selected for 4-gang switch sensor and 6-gang switch actuator. It is also possible that setting the application parameters results in different C.Obj.

Therefore, the following steps are to be taken:

1. Select application
2. Set parameters of the application
3. Create group addresses
4. Assign communication objects to group addresses

Figure 3.26 shows how the two devices are connected. C.Obj. 0 (object name: *left push button—short*) is assigned to the *left push button*, C.Obj. 0 (object name: *Output A*) of the six-gang switch actuator is assigned to output (relay) A, and C.Obj. 1 (object name: *Output B*) of the six-gang switch actuator is assigned to output (relay) B. All C.Obj. are assigned to the same group address 1/1. After programming both devices, the system is ready for use.

Pressing the upper contact of the left push button sends a Switch ON data frame (with group address 1/1, the command *write* value, and the object value “1”), which closes the relay contacts for outputs A and B and turns on the lamps. And in same way, pressing the lower contact of the left push button, sends a Switch OFF data frame (with the group address 1/1, the command *write* value, and the object value “0”), which then opens the relay contact for outputs A and B and turns off the lamps.

This is assuming that the *Switch function* parameters for outputs A and B have both been set to *normally opened contact*, and also that the communication flag and the transmission has been set for C.Obj. 0 and 1 in the switch actuator. The C.Obj. then has a standard connection to the bus, which if the object value changes, a data