2019

TAPKO TECHNOLOGIES GMBH

SIM-KNX

Serial Interface for KNX

Technical & Application Description

Content

1 Product Description

SIM-KNX is an easy to use serial interface for connecting to KNX. It contains the certificated KNX communication system for data format conversion. Access to KNX is realized by a serial ASCII protocol.

The SIM-KNX modules and SIM-KNX devices are designed to connect a controller or other non-KNX devices to the KNX bus system. The hardware contains a microcontroller and provides galvanic isolation between KNX and serial connection.

In this document, physically addressed telegrams are named Physical Telegrams.

In this document, group oriented telegrams are named Group Telegrams.

1.1 SIM-KNX Hardware Structure

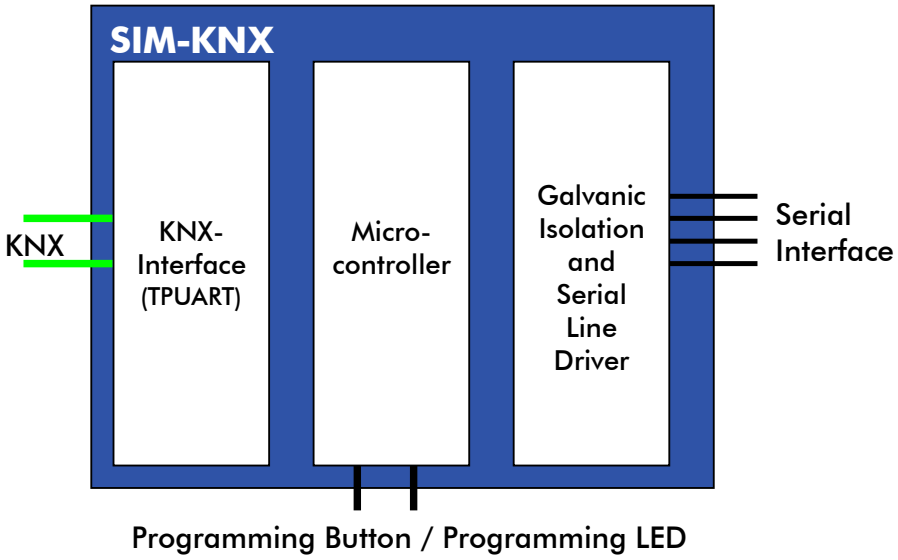

Figure 1: Block Diagram of SIM-KNX Hardware

Non-volatile data is stored in internal flash memory. So, some commands do have a negative impact on the flash memory lifetime.

1.2 Connection Pins

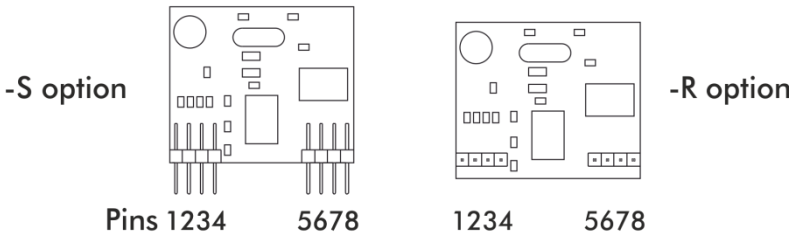

Figure 2: Pin Assignment

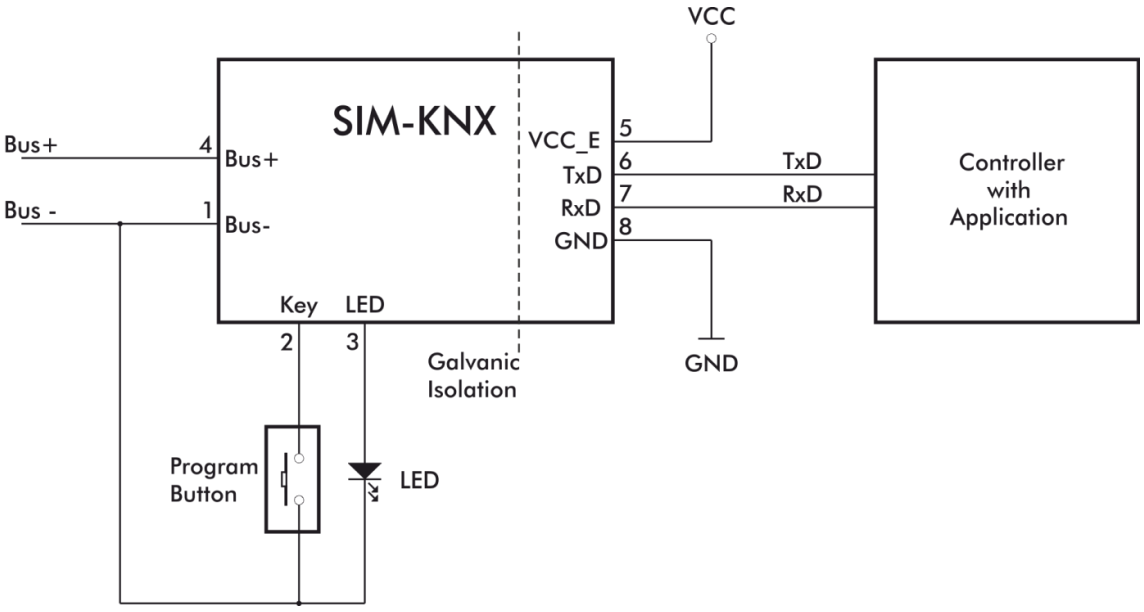

Figure 3: Connection Digram

Table 1: Pin Function

Pin No.	Pin Name	Description
1	Bus-	Negative bus pin
4	Bus+	Positive bus pin
2	Key	Connector for KNX Programming Button
3	LED	Connector for KNX Programming LED
8	GND	Ground
7	RxD	Input of the serial interface
6	TxD	Output of the serial interface
5	VCC_E	Power input for the galvanic isolated part

1.3 Feature Summary

Application Interface:

- serial asynchronous interface
- 3V to 5V interface
- 3-wire interface
- ASCII protocol
- configurable baud rate and transmission parameters
- access to KNX group communication objects (runtime communication)
- access to KNX interface objects (configuration)
- configurable indication when group communication value was received

KNX features (RAW Mode + Interoperability Mode):

- device model 0701
- integrated mechanism for configuration via KNX
- read requests from KNX are internally processed
- two different numbers of group objects: 128 or 254

KNX group communication objects (RAW Mode):

- transparent transmission of group communication object data
- data conversion not active
- telegram generation is controlled from serial interface side
- configuration via serial interface

KNX group communication objects (Interoperability Mode):

- support of EIB/KNX datapoint types (EIS/DPT)
- data conversion for group object values (e.g. temperature -> DPT5)
- configurable sending conditions for all group communication objects
- configuration via ETS database entry/KNX bus and serial interface
- indication when data received, value changed, positive/negative edges (DPT1)
- cyclic (time can be configured from 3 to 255 seconds, 3 to 255 minutes)
- complex sending conditions are available for certain datapoint types:
  - send on value difference
  - integrated threshold switch
    (to trigger a group communication object on threshold value passing)

Transparent Mode

- receiving of all group oriented telegrams
- sending to all group addresses
- no filtering of telegrams
- no data type restriction to a specific group address for sending
- unnecessary protocol oriented information is removed
- also suitable for tracing and logging of telegrams

1.4 Resources in RAW and Interoperability Mode

1.4.1 SIM-KNX128

Number of group addresses: 254

Number of associations: 254

Number of communication objects: 128

Size of application parameters: 512 byte

Table 2: SIM-KNX128 Communication Objects

Communication Object	Max. Size	Comments
0 – 15	4 bytes	Can be used for complex sending conditions
16 – 63	4 bytes	
64 – 111	1 byte	
112 – 127	14 bytes	

1.4.2 SIM-KNX250

Number of group addresses: 254

Number of associations: 254

Number of communication objects: 254

Size of application parameters: 512 byte

Table 3: SIM-KNX250 Communication Objects

Communication Object	Max. Size	Comments
0 – 15	14 bytes	Can be used for complex sending conditions
16 – 111	4 bytes	
112 – 127	14 bytes	
128 - 253	4 bytes	

1.5 Electrical Specification

Voltages refer to GND potential. Currents flowing into the pins are determined positive.

Table 4: Absolute Maximum Ratings

Symbol	Parameter	Min	Typ.	Max	Unit
V_{ISO}	Isolation voltage	4000			V
V_{BUS}	Bus voltage (Bus+ to Bus-)	-45		45	V
V_{VCC}	Reference voltage	-0.5		5.5	V
V_{RxD}	Voltage on pin RxD	-0.5		V_{VCC}	V
	Storage temperature	-40		85	°C

Table 5: Recommended Working Conditions

Symbol	Parameter	Min	Typ.	Max	Unit
V_{BUS}	Bus voltage (Bus+ to Bus-)	20	28	33	V
V_{VCC}	Reference voltage	3		5	V
I_{VCC}	Current (at V_{VCC}=5,5V)			1.5	mA
V_{RxD}	Voltage on pin RxD	-0.5		V_{VCC}	V
I_{BUS}	Bus power consumption		5		mA
I_{LED}	LED power consumption		2.6		mA
I_{Key}	Key power consumption		33	50	µA
I_{RxD}	Current at pin RxD		-7		mA
$V_{RxD(low)}$				0.7	V
$V_{RxD(high)}$	with I_{TXD} < 100µA	V_{VCC} –1.5			V
I_{TxD}	Current at pin RxD (at V_{VCC}=5.5V)		-0.1	-0.8	mA
$V_{TxD(low)}$		0		0.7	V
$V_{TxD(high)}$	with I_{TxD} < 100µA	V_{VCC} –0.8		V_{VCC}	V
	with I_{TxD} < 50µA	V_{VCC} –0.4		V_{VCC}	V
	Total power dissipation			1	W
	Working temperature	-5		60	°C

1.6 Mechanical Specification

Dimensions shown here are specified in mm. Measures in technical drawings have a tolerance of 0.5 mm. Recommended measures are accurate within 0.1 mm.

1.6.1 Technical Drawings

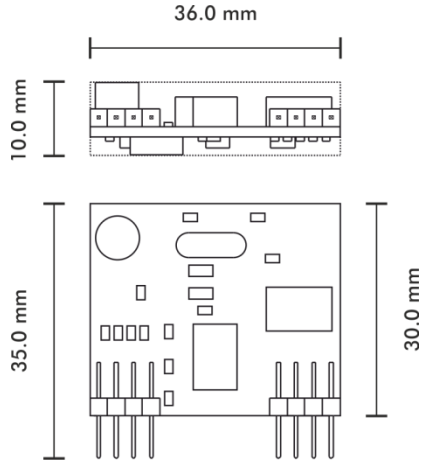

Figure 4: Straight Connector (-S option)

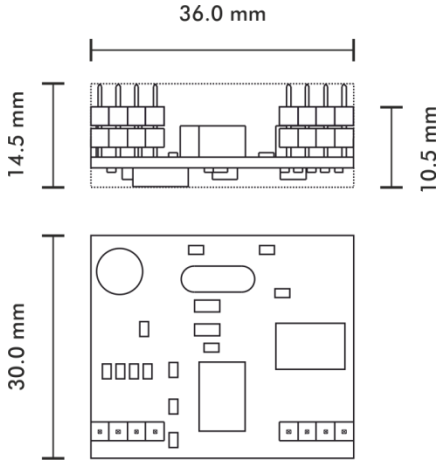

Figure 5: Right Angle Connector (-R option)

1.6.2 Recommended Footprint

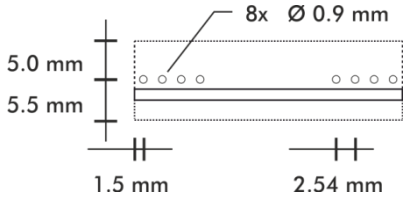

Figure 6: Footprint for -S option

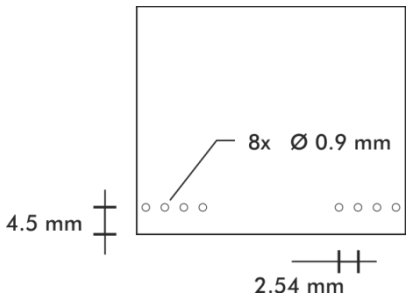

Figure 7: Footprint for -R option

1.6.3 Necessary Isolation Area between Circuitries

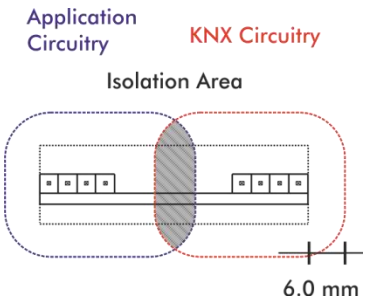

Figure 8: Necessary Isolation Area for -S option

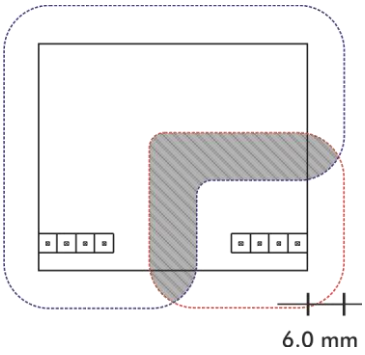

Figure 9: Necessary Isolation Area for -R option

1.7 Command Overview

This chapter gives an overview of the commands that can be processed by SIM-KNX. Their usage is also stated. Complete information concerning a command is shown by tables in chapter 34 Serial protocol (grey hyperlinks guide to the command table of interest).

Certain commands modify the device internal memory. It is strongly recommended to not use these commands on a permanent basis. Command tables of chapter 4 indicate the relevant commands.

1.7.1 General Commands

Table 6: General Commands

Command	Usage
dag	Get physical address
das *physicalAddress*	Set physical address
dpg	Get programming mode
dps *progMode*	Set programming mode
dr	Restart device
dsg	Get device state
dvg	Get version
dts *data*	Device transparent set
gci	Reset to manufacturer default settings
pdg *(index count)*	Get parameter data
idg *(ioIndex propertyID)* **idg** *(ioIndex propertyID elementIndex)*	Get interface object data
ids *(ioIndex propertyID) data* **ids** *(ioIndex propertyID elementIndex) data*	Set interface object data

1.7.2 Commands in RAW Mode and Interoperability Mode

Table 7: RAW/Interoperability Mode Commands

Command	Usage
ods *(objectNr) data*	Set object data (RAW Mode)
odg *(objectNr)*	Get object data (RAW Mode)
odt *(objectNr)*	Send group telegram
odr *(objectNr)*	Send group read telegram
ofg *(objectNr)*	Get RAM flags
ovs *(objectNr) data*	Set object value (Interoperability Mode)
ovg *(objectNr)*	Get object value (Interoperability Mode)
ogs *(objectNr) group*	Set sending group address
oga *(objectNr) group*	Add group address
ogd *(objectNr) group*	Delete group address
ogg *(objectNr)*	Get group addresses
ocs *(objectNr) DPT objectType comFlags sendConfig rcvConfig time*	Set object configuration
ocg *(objectNr)*	Get object configuration
dus	Set event generation
gug	Return the update flag of all group object
gcg	Return the valueChanged flag of all group object
gtg	Return the timeout flag of all group object

Table 8: RAW/Interoperability Mode Indications

gui	Receive the global update flag
oui	Receive update data for a communication object
dsi	Receive device state

1.7.3 Commands in Transparent Mode

Table 9: Transparent Mode Commands

Command	Usage
dts *data*	Device transparent set
tds *(dest) data* **tds** *(dest length) data*	Send data by GroupValueWrite telegram
trs *(dest)*	Send request by GroupValueRead telegram
tes *(dest) data* **tes** *(dest length) data*	Send read response by GroupValueResponse telegram
tdi *(source dest length) data*	Send GroupValueWrite indication
tri *(source dest)*	Send GroupValueRead indication
tei *(source dest length) data*	Send GroupValueResponse indication
tdc *(source dest length) data*	Send GroupValueWrite confirmation
trc *(source dest)*	Send GroupValueRead confirmation
tec (source dest length) data	Send GroupValueResponse confirmation
tdn *(source dest length) data*	Send GroupValueWrite negative confirmation
trn *(source dest)*	Send GroupValueRead negative confirmation
ten *(source dest length) data*	Send GroupValueResponse negative confirmation

1.7.4 Explanation of Command Table Buildup

The command tables contained in chapter 4 have following general buildup.

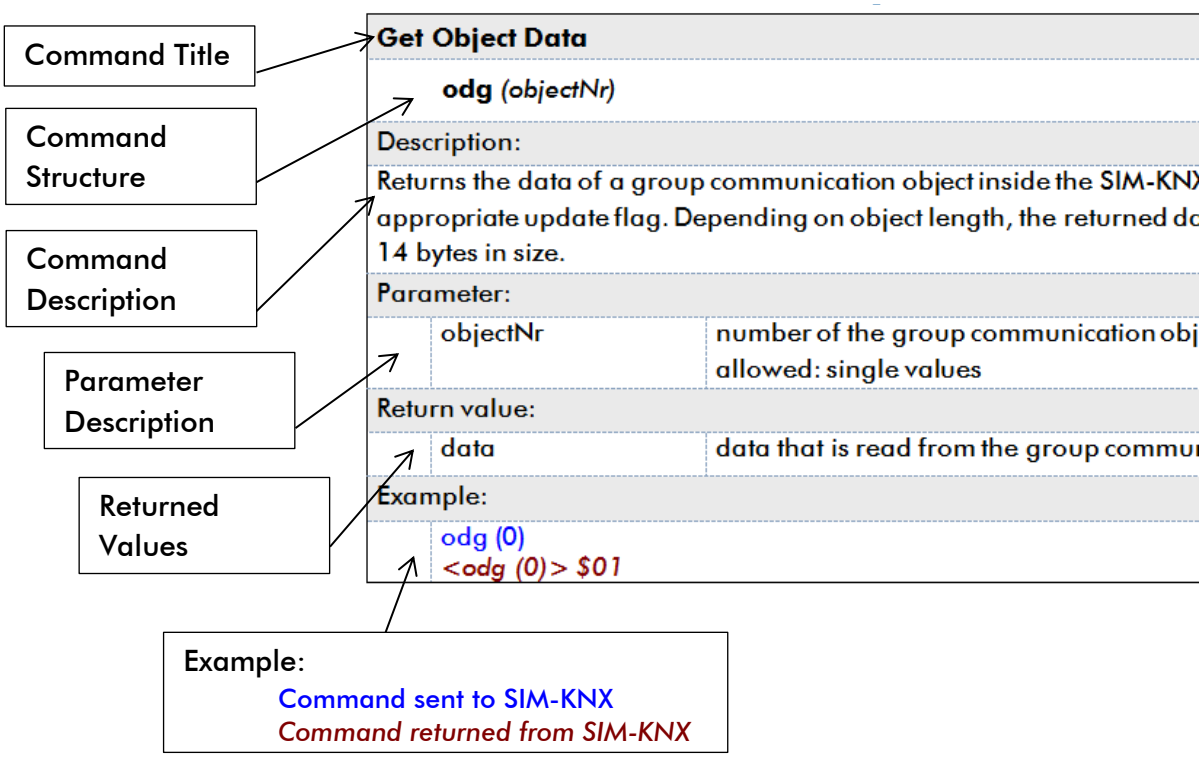

Figure 10: Command Table Example

2 Operational Description

2.1 Introduction

In general, there are two ways of operating with SIM-KNX:

- SIM-KNX behaves like a normal device having its own physical address, an address table, communication objects and full device management.
- SIM-KNX has an additional bypass for group oriented communication. It is active in the lower layers. All group oriented traffic is routed to the serial interface side without filtering and without any communication object linkage.

2.2 SIM-KNX Software Structure

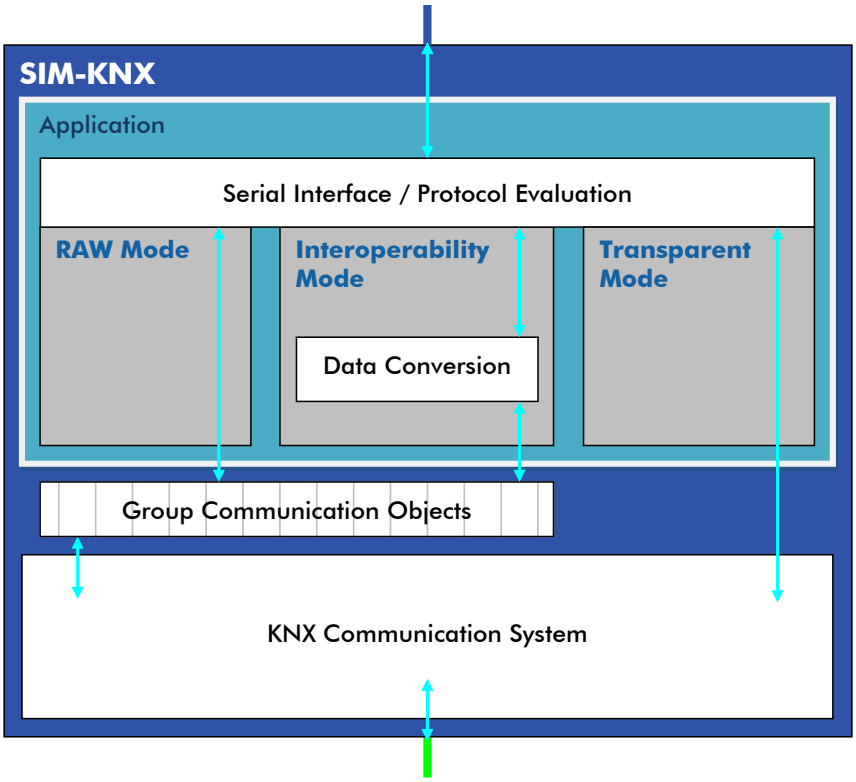

Figure 11: Block Diagram of SIM-KNX Software

2.3 Operating Modes

2.3.1 RAW Mode

In RAW Mode, SIM-KNX transparently transfers data between serial interface and KNX. Data transmission is controlled from serial interface side. Due to configuring RAW Mode usually from serial interface side, no external tool is additionally required.

2.3.2 Interoperability Mode

In Interoperability Mode, SIM-KNX converts the data sent from serial interface to KNX to a KNX data format. According to the configuration settings, data transmission is controlled by SIM-KNX. Configuring Interoperability Mode can be done via both KNX side (with ETS) and the serial interface side.

 For some datapoint types complex sending conditions are available.

2.3.3 Transparent Mode

Transparent Mode enables sending to and receiving from all group addresses without filtering. There are no data type limitations for sending.

 Maximum object data size is 14 bytes.

 Transparent Mode is suitable for tracing the group oriented telegram traffic.

2.4 Operation with Communication Objects (RAW Mode/Interoperability Mode)

Runtime communication on KNX is accomplished via communication objects. The SIM-KNX memory contains a set of communication objects. Group Telegrams are received and values of associated group addresses are stored in the corresponding communication objects. SIM-KNX always stores the last received communication object value. This value can be read out by the serial interface.

Regarding the other direction, the serial interface transfers a value to a communication object of SIM-KNX and SIM-KNX sends the value, according to its sending conditions, to the KNX bus.

It is also possible to read values from KNX side. Therefore, SIM-KNX can be triggered by the serial interface to send a read request on the bus. When a 'value read' was received from KNX, the response is handled inside SIM-KNX and the stored communication object values are sent to the bus.

When an object value was received or changed, SIM-KNX is able to create an indication on the serial interface.

For data exchange via communication objects SIM-KNX offers two modes, RAW Mode and Interoperability Mode. The operating mode can be selected for each communication object independently.

2.4.1 RAW Mode

In RAW Mode, raw data is exchanged with SIM-KNX by the serial interface. SIM-KNX has no knowledge about format and semantics of exchanged data. Only the communication object size is known by SIM-KNX.


Usually, RAW Mode is used for configuring SIM-KNX via the serial interface.

Configuring communication objects in RAW Mode can be done by setting the DPT to 0 with the command **ocs** and then setting the object size with the object type parameter.

2.4.2 Interoperability Mode

In Interoperability Mode, SIM-KNX knows communication object size and datapoint type (DPT). As they are standardized to guarantee interworking of all devices, DPTs are defined by format and usage. A short list of mainly used DPTs is shown in chapter 2.4.5.


SIM-KNX supports a wide range of datapoint types.

Configuring communication objects in Interoperability Mode can be done by setting the DPT with the command **ocs**. The object type parameter is then ignored by SIM-KNX.

2.4.3 Communication Object Features

All configuration data of SIM-KNX is stored in non-volatile memory. A device restart or power down cannot cause configuration data changes.

Communication object values are stored in volatile memory. After a restart or power down of SIM-KNX all present object values are lost.

SIM-KNX communication objects have following features:

- Standard features of communication objects as defined by the device model type:

  1. Enable sending and receiving of communication object values
  2. Enable receiving of values read from KNX side
  3. Setting of priorities

- Specific features of all SIM-KNX communication objects:

  1. Format conversion in Interoperability Mode
  2. Object values can be sent to KNX
     - after receiving a value from the serial interface side
     - after changing a value at the serial interface
     - for DPT1: on changes of positive edge, negative edge and/or both
     - cyclically
  3. Indications to the serial interface can be generated
     - on receiving a value from KNX
     - on changing a received value from KNX
     - when the receive-timeout was elapsed

- Complex sending conditions for SIM-KNX, only available for the first block of communication objects (COs 0-15):

  1. Only configurable by ETS
  2. Sending to KNX after a value change of certain amount
  3. Threshold switch for triggering a communication object after crossing pre-set threshold values.

2.4.4 Transferring Communication Object Data by the Serial Interface

The data of a communication object is set with the commands

- **ods** in RAW Mode,
- **ovs** in Interoperability Mode.

For receiving the communication object data the commands are

- **odg** in RAW Mode,
- **ovg** in Interoperability Mode.

In RAW Mode, the format in which the data is transferred to is hex bytes. In Interoperability Mode, the data format depends on the DPT that is used and can be, for example, a simple number, a float value or a string. Data formats are described in the list of DPTs.

2.4.5 Most Commonly Used Datapoint Types

KNX datapoint types and their usage are standardized. All definitions of these datapoint types can be found in the KNX handbook "Volume 3 / Part 7 / Chapter 2: Datapoint Types" available from KNX Association.

- **DPT1 (1BIT) ON / OFF**

This DPT is used to switch on (1) or switch off (0) lights, relays, etc. It is also used to move blinds up (0) and down (1). Further usages like enable / disable are also defined.

The serial interface of SIM-KNX transfers the data as single values 0 or 1.

- **DPT3 DIMMING CONTROL**

This DPT is defined as a 4-bit communication object. The values of this DPT are interpreted as follows: *C VVV*

Data format of SIM-KNX: *C* {0,1}: control (0=off, 1=on)
V {0...7}: value

- **DPT9 TEMPERATURE**

This DPT is defined as a 16-bit floating point value.

Data format of SIM-KNX (Interoperability Mode): single value

Due to data conversion of values, it may happen that the value that is read back from SIM-KNX is not exactly the original one that was written.

- **DPT5 SCALING**

This DPT is used for absolute dimming, absolute blind position, value of valves, etc.

On KNX, values are transmitted as 1-byte values (0…255). 100% is coded as 255.

In Interoperability Mode, values are transmitted as single values in the range of 0 to 100.

Due to data conversion of values, it may happen that the value that is read back from SIM-KNX is not exactly the original one that was written.

- **DPT10 TIME**

On KNX, time is transmitted as a 3-byte value.

Data format of SIM-KNX: 4 kinds of values: *w h m s*

- **DPT11 DATE**

This DPT is similar to DPT10.

SIM-KNX uses following interpretation:

Octet 3 contains value ≥ 90 -> interpreted as 20th century

Octet 3 contains value < 90 -> interpreted as 21st century

This format covers the range 1990 to 2089.

Data format of SIM-KNX: 3 kinds of values: *d m y*

2.4.6 Configuration of Communication Objects

Configuring communication, i.e. the communication objects of tables 2/3, can be done

- via the serial interface
- with the application specific ETS database entry from KNX side.

2.4.6.1 Configuring Objects via the Serial Interface

Configuring communication objects is divided in 3 parts:

1. Assignment of group addresses
2. Setting the communication parameters
3. Configuring the indications

Assignment of group addresses:

Group addresses for the communication objects are configured with the commands:

- **ogs** (set sending group address)
- **oga** (add group address)
- **ogd** (delete group addresses)

Each communication object can be associated with several group addresses. One of these associated group addresses always is the sending group address for sending object values (on the bus). All other associated group addresses are used to receive the object values.

Receiving group addresses are set by the command **oga** (e.g. **oga (1) 1/0/0**). Group addresses can be transferred as hex number ($1000) or in ETS format (2/0/0).

The sending group address is set by **ogs**. When there was one sending group address present before sending this command, this sending group address remains the receiving group address. When the sending group address was deleted or no group address is marked for sending, one of the receiving group addresses automatically becomes the sending group address.

Single group addresses can be deleted by the command **ogd**. To delete all group addresses associated to one communication object the command **ogd (0) "all"** can be used.

The number of group addresses which can be associated to the communication objects is limited by the global address resources and the association table.

For each group address one address table entry is used, no matter how many communication objects are associated to this group address.

For each association between group address and communication object one entry in the association table is used.

Setting the communication parameters

Parameters for communication objects are configured by the command **ocs**. The current communication object configuration can be retrieved via **ocg**.

Commands have following parameters:

- ***DPT*** data point type
- ***objectType*** type of the communication object
- ***flags*** configuration flags
- ***sendConfig*** configuration when the value is sent
- ***rcvConfig*** configuration when the indications will be sent
- ***time*** cycle time

The parameters ***DPT*** and ***objectType*** set operation mode and size/format of a communication object.

Setting the data format for Interoperability Mode is done by selecting the desired data point type with the parameter ***DPT***. Then the parameter ***objectType*** is not used.

To set the size of a communication object in RAW Mode, the parameter ***DPT*** must be set to 0 and the parameter ***objectType*** must be set to the required size.

 The total object data size for communication objects is limited.

The ***flags*** parameter contains the configuration flags of a communication object which are also displayed in ETS. Sending and receiving of communication object values can be controlled by this flags.

The ***sendConfig*** parameter configures the behavior, when to send an object value on the KNX bus. Depending on this parameter, the object value is sent when a value was received from the serial interface, or only when it was changed, or e.g. cyclically.

Via the ***rcvConfig*** parameter it can be configured which indication is sent by the serial interface.

The ***time*** parameter sets the time period for cyclic sending and for the receive-timeout. If this parameter is set to 0, cyclic sending and receive-timeout are switched off.

If one parameter of the command **ocs** not needs to be modified, it is possible to replace this parameter by a '*'.

2.4.6.2 Configuring Objects by ETS Database Entry

Settings that can be set by the serial interface can also be set with the application-specific ETS database entry. Here, complex sending conditions are additionally available.

Required information to build the ETS database entry can be found in chapter 6 ETS.

2.5 Operation without Communication Objects (Transparent Mode)

In Transparent Mode, SIM-KNX activates a bypass to channel all group oriented messages from the lower layers directly to the serial interface without filtering. All requests coming from the serial interface are directly sent to the lower layers.

Care must be taken when Transparent Mode is switched on because the bypass activation won´t deactivate the object handling. Erasing all communication objects with the command **gci** prevents unpredictable interference between objects and incoming messages.

Transparent Mode can be switched on or off with the command **dts**.

It is recommended to use Transparent Mode only with a baud rate of 38,400 baud.

2.5.1 Transparent Mode Communication

To use Transparent Mode, it is essential to understand the communication mechanism for group oriented communication. In the following, three communication situations are described in detail. The appropriate ASCII commands for the serial interface are indicated by green letters.

Situation1: One device wants to distribute a value (simplest situation)

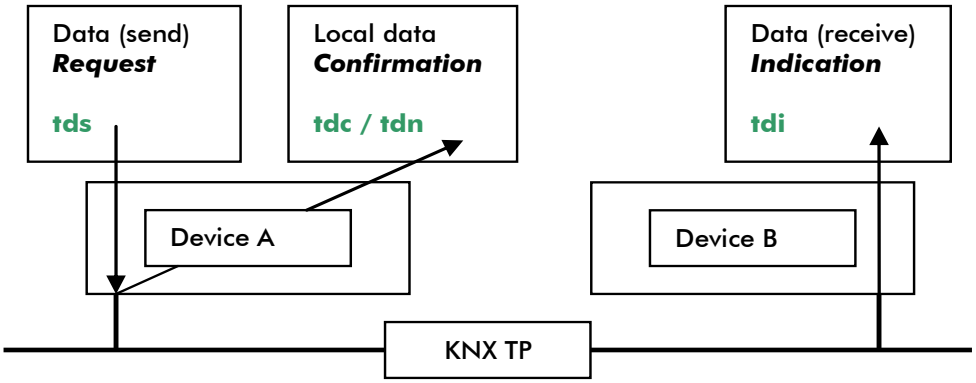

Figure 12: Communication Situation Case1

A ‘send data request’ is sent to SIM-KNX. This request is forwarded to the bus by device A. When local confirmations are enabled, messages plus the information of successful or faulty transmission are returned to the serial interface. (This procedure bases on the result of immediate acknowledge reception on the KNX bus after transmission of the message.) The remote device B receives the message from device A and indicates this fact to its own serial interface by sending a data indication.

Situation2: A device is requested to send its actual value

This request for data is called a 'read request'. A local confirmation is sent by the serial interface once the transmission is completed. The remote device B receives the request and sends a read indication to its own serial interface.

This step alone does not yet result in getting the requested value.

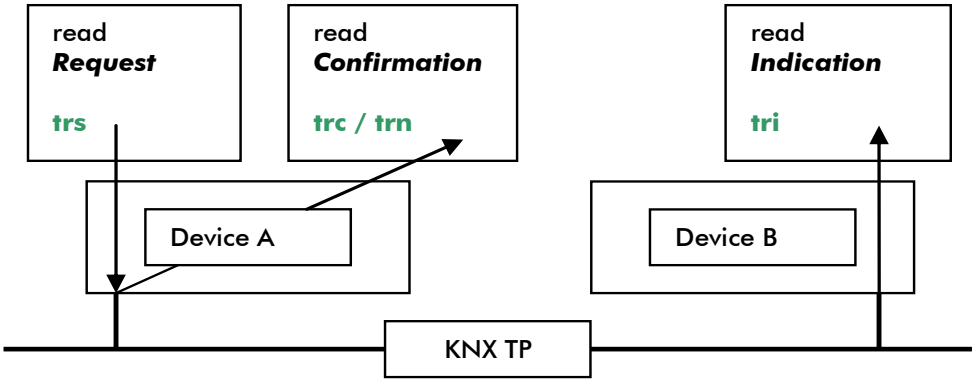

Figure 13: Communication Situation Case2

Situation3: A device sends the response as a result to the request of Situation2

The remote device B sends a response message from its serial interface to the KNX bus. In the remote device B it is locally confirmed and the confirmation message is distributed to all other devices connected with the KNX bus system. Device A receives this confirmation message and sends a response indication via its serial interface.

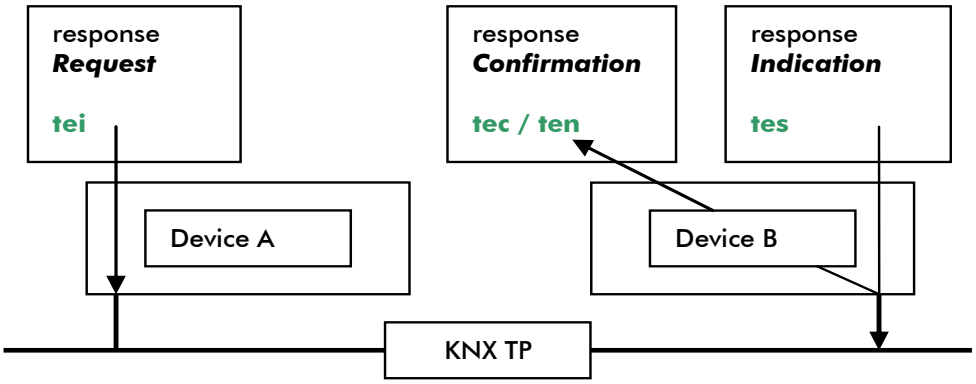

Figure 14: Communication Situation Case3

2.5.2 Transparent Mode Configuration

In Transparent Mode, the local confirmations can be suppressed by an appropriate setting of the **dts** command parameters.

2.6 Device Information

2.6.1 Physical Address

Physical address and its assignment are completely handled inside SIM-KNX. Usually, there is no need to set or read the physical address by the serial interface. If required anyway, there is the possibility to have it read and set by the commands **dag** and **das**.

2.6.2 Programming Mode

Programming Mode, Programming LED and Programming Button are handled inside SIM-KNX. Usually, there is no need to set or read the Programming Mode state by the serial interface. If required anyway, there is the possibility to have it read and set by the commands **dpg** and **dps**.

2.6.3 Other Device Information

SIM-KNX also provides other device information like version (**dsg**) and state of the internal application (**dvg**).

2.7 Flash Memory

In flash memory the non-volatile data is stored. Lifetime of the internal flash memory is limited and certain commands modify the internal flash memory. So, it is recommended to use these commands not in excess meaning only for configuration purpose and not on a permanent basis.

Table 10: Commands that modify Internal Flash Memory

Command	Usage
das *physicalAddress*	Set physical address
gci	Reset to manufacturer default settings
dus	Set event generation
ids *(ioIndex propertyID) data* **ids** *(ioIndex propertyID elementIndex) data*	Set interface object data
ogs *(objectNr) group*	Set sending group address
oga *(objectNr) group*	Add group address
ogd *(objectNr) group*	Delete group address
ocs *(objectNr) DPT objectType comFlags sendConfig rcvConfig time*	Set object configuration

During processing a command that modifies the flash memory it is not possible to process a further command. It is urgently recommended to wait at least 1 s to send a further command. Commands sent anyway will be lost.


Between sending of two commands that modify flash memory it is important to wait at least 1 s.

3 Examples for using SIM-KNX

3.1 without ETS Database Entry

In this case the complete configuration of SIM-KNX can be done by the serial interface:

- Configuration of communication objects
- Assignment of Group Addresses

3.2 with ETS Database Entry

In this case all configurations of SIM-KNX can be done with the database entry. No configuration must be done by the serial interface.

During development, it may be necessary that SIM-KNX is configured by the serial interface.

3.3 in Transparent Mode

After switching on Transparent Mode, e.g. with the command **dts**, from the KNX bus to the serial interface all group oriented messages are passed through in the form of *Transparent Data Indication* (normal writing coming from the bus), *Transparent Read Indication* (asking a sensor its value) or *Transparent Response Indication* (the answer on a request from the sensor). The information passed to the serial interface is the Physical Address of the originating device (source), the destination Group Address and the data (containing length information) itself.

From the serial interface a *Transparent Data Send* (normal writing to the bus), a *Transparent Response Send* (answer to a request from the bus) and a *Transparent Read Send* (sending a request for a value to the bus) can be sent.

In this example no local confirmations are generated and the hex format is used for displaying received messages.

For getting the Transparent Mode status, see chapter 5 Implemented Application Interface Object and property ID 51.

4 Serial protocol

4.1 Communication Parameter Settings

Serial interface default settings

- 9600 baud
- 8 databit
- no parity
- 1 stopbit
- no hardware handshake

Changing the settings

The settings are represented by 2 bytes of flash memory. They can be changed by ETS (see chapter 6.2 Parameter) and with the Application Interface object ID50 (see chapter 5 Implemented Application Interface Object).

Detailed description of the 2 bytes

Bit 0, 1:	handshake (reserved for future use)
Bit 2-7:	not used
Bit 8:	number of databits (0=7 bits, 1=8 bits)
Bit 9, 10:	parity (0=none, 1=even, 2=odd)
Bit 11:	number of stopbits (0=1 stopbit, 1=2 stopbits)
Bit 12-15:	baud rate following baud rates are possible: 0 = 1200 bps 1 = 2400 bps 2 = 4800 bps 3 = 9600 bps (preferred) 4 = 19200 bps 5 = 38400 bps


After changing one or more of these settings, a restart is necessary.

Example for changing the baud rate

The baud rate is changed to 38400 baud:

ids (5 50) $51 $00

4.2 Syntax

4.2.1 General Syntax (command based)

Syntax is command based and uses the ACSII char set. Commands are terminated by <CR>. <LF> is ignored.

General syntax for commands

command<CR>
command (parameter)<CR>
command (parameter) data<CR>

A command contains up to 3 parts

- *command*
  The command itself. It defines the kind of operation that is is executed.
- *parameter*
  specifies the element to be manipulated. For example, one of the group communication objects.
- *data*
  contains the values that are transmitted to SIM-KNX. For example, the value of a group communication object or the physical address.

4.2.2 Values

There are different types of values. For every command, their usage is defined.

- **Single value** can either be a 1 byte value, a 2 byte value or a 4 byte value specified in one of following formats:
  decimal: *1234*
  hexadecimal: *$1234*
  binary: *%10101*
- **Hex stream** defines a sequence of hexadecimal bytes:
  #12345678
- **String** must be enclosed in quotation marks:
  "Hello"
- **Wildcard**
  In some commands it is allowed to use '*' as wildcard.
- **Group address in ETS format** can be used for group address manipulation:
  1/234
  1/2/34

4.2.3 Strings from SIM-KNX

Three types of strings are sent from SIM-KNX:

- Responses
- Indications
- Error messages

4.2.3.1 Responses

Responses follow the general syntax for commands.

General syntax for responses

If echoing the received command string is active, response syntax is:

<commandstring>returnValues<LF><CR>

If echoing the received command string is not active, response syntax is:

returnValues<LF><CR>

A response contains 2 parts

- *commandstring*
  is a general generic response for all commands. It can be configured whether the received command string will be returned or not.
- *returnValues*
  are the data that were requested. Values and their data format depend on the executed command.

Changing the responses in RAW/Interoperability Mode

The settings are stored in the flash memory. They can be changed by ETS (see chapter 6.2 Parameter) and with the Application Interface object ID52 (see chapter 5 Implemented Application Interface Object). Following settings are possible:

Bit 0: 0 = response without command string
1 = response with command string
Bit 1: 1 = print "ok" if no data are following

Table 11: Examples for Changing the Responses in RAW/Interoperability Mode

Setting: 0x00	Setting: 0x01	Setting: 0x02	Setting: 0x03
odg (0) *$01*	odg (0) *<odg (0)>$01*	odg (0) *$01*	odg (0) *<odg (0)>$01*
odt (0)	odt (0) *<odt (0)>*	odt (0) *ok*	odt (0) *<odt (0)>* *ok*

4.2.3.2 Indications

Indications are sent without requests. They are independent from the response syntax. Indications are terminated by <LF><CR>. For details see 4.4.6 Indications.

4.2.3.3 Error Messages

An error message is sent after an invalid command was received. Explanations of error codes can be found in chapter 4.6 Error Codes.

An error message contains 3 parts

- Keyword *"!error"*
- *error number*
- *received command*

Error Message Example

!error $0215 : <abc>

4.3 Command Reference (General)

4.3.1 Accessing Interface Objects

Get interface object data	
idg *(ioIndex propertyID)* **idg** *(ioIndex propertyID elementIndex)*	
Description:	
Get the data of one interface object property. The property is selected via ioIndex and propertyID. If the property is implemented as an array the elements are selected via elementIndex.	
Parameter:	
ioIndex	index to the interface object allowed: single values
propertyID	ID of the property allowed: single values
elementIndex	element of the property allowed: single values is set to 1, if skipped
Return value:	
data	Data, which are read from the property.
Example	
idg (5 52) *<idg (5 52)>$01*	

Set interface object data	
ids *(ioIndex propertyID) data* **ids** *(ioIndex propertyID elementIndex) data*	
Description:	
Set the data of one interface object property. The property is selected via *ioIndex* and *propertyID*. If the property is implemented as an array the elements are selected via *elementIndex*. This command modifies the internal flash memory. It is recommended to use this command only for configuration purpose and not on a permanent basis.	
Parameter:	
ioIndex	index of the interface object allowed: single values
propertyID	ID of the property allowed: single values
elementIndex	element of the property allowed: single values is set to 1, if skipped
data	data that is written to the property, only for one element. allowed: single values, hex stream for elements of size >1 byte
Example:	
ids (5 52) 1 *<ids(5 52) 1>*	

4.3.2 Device Settings

Get physical address	
dag	
Description:	
Get the physical address of SIM-KNX.	
Parameter:	
-	
Return value:	
physicalAddress	physical address as one single value
Example:	
dag *<dag>$ffff*	

Set physical address	
das *physicalAddress*	
Description:	
Set the physical address of SIM-KNX. This command modifies the internal flash memory. It is recommended to use this command only for configuration purpose and not on a permanent basis.	
Parameter:	
physicalAddress	physical address allowed: single values
Example:	
das $1508 *<das $1508 >*	

Get programming mode	
dpg	
Description:	
Get the Programming Mode state of SIM-KNX. This state is also indicated by the LED.	
Parameter:	
-	
Return value:	
progMode	state of Programming Mode 0: off 1: on
Example:	
dpg *<dpg>$01*	

Set programming mode	
dps *progMode*	
Description:	
Set the state of Programming Mode of SIM-KNX.	
Parameter:	
progMode	state of Programming Mode 0: off 1: on allowed: single values
Example:	
dps 1 *<dps 1>*	

Restart device	
dr	
Description:	
Execute a restart of SIM-KNX, after a delay of 50ms.	
Parameter:	
-	
Example:	
dr *<dr>* *gui $01*[1]	

Get device state	
dsg	
Description:	
Get various states of SIM-KNX.	
Parameter:	
-	
Return value:	
bit0	0: normal operation 1: Transparent Mode
bit1	1: application loaded
bit2	1: application is running
bit7	1: device is in Programming Mode
Example:	
dsg *<dsg>$06*	

[1] if the global indication for restart is set

Get version	
dvg	
Description:	
Get the version of SIM-KNX.	
Parameter:	
-	
Return value:	
deviceVersion	single value, which shows the kind and the version of this device. high byte: general type of the device: 00: bus coupling via TP1 media low byte: version of this device
protocolVersion	version of the software protocol high byte: main version of this protocol. When this version changes, the protocol may have incompatible changes. low byte: sub version of this protocol. Higher sub versions are always upward compatible.
activeObjects	number of activated communication objects
Example:	
dvg *<dvg>$0001 $0001 $80*	

Reset to manufacturer default settings
gci
Description:
Clear all settings that were done by ETS and the serial interface, and execute a restart. This command modifies the internal flash memory. It is recommended to use this command only for configuration purpose and not on a permanent basis.
Example:
gci

4.3.3 Parameter

Get parameter data	
pdg *(index)* **pdg** *(index count)*	
Description:	
Get the parameter settings that can only be written by ETS.	
Parameter:	
index	index of the parameter
count	number of parameters to be read out is set to 1 if skipped
Return value:	
data	
Example:	
pdg (10 4) *<pdg (10 4)>$01 $02 $03 $04* pdg (11) *<pdg (11)>$02*	

4.4 Command Reference (RAW/Interoperability Mode)

4.4.1 Configuration

Set event generation	
dus *globalEvent*	
Description:	
Set the configuration of the global event generation. This command modifies the internal flash memory. It is recommended to use this command only for configuration purpose and not on a permanent basis.	
Parameter:	
globalEvent	send global Event bit 0: at restart bit 3: if global update flag is set bit 4: if global changed flag is set bit 6: if timeout on serial interface has occured bit 7: if global receive-timeout is set
Example:	
dus $01 *<dus $01>*	


For getting the status see property ID 128 in chapter 5 Implemented Application Interface Object.

4.4.2 Accessing Group Communication Objects (RAW Mode)

Set object data	
ods *(objectNr) data*	
Description:	
Set the data of a group communication object inside SIM-KNX. Depending on the sending condition, transmission is automatically initiated. This command deletes the appropriate update flag. Depending on object length, the data size of this command is from 1 byte up to 14 bytes.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
data	data that is written to the group communication object. allowed: single values, hex stream (Take care that the correct length of the data is set.)
Example:	
ods (0) 1 *<ods (0) 1>* ods ($1) $0 $1 $2 *<ods ($1) $0 $1 $2>* ods (1) #000102 *<ods (1) #000102>*	

Get object data	
odg *(objectNr)*	
Description:	
Get the data of a group communication object inside SIM-KNX. This command deletes the appropriate update flag. Depending on object length, the returned data can be from 1 byte up to 14 bytes in size.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
Return value:	
data	data that is read from the group communication object.
Example:	
odg (0) *<odg (0)> $01*	

4.4.3 Accessing Group Communication Objects (Interoperability Mode)

Set object value	
ovs *(objectNr) data*	
Description:	
Set the value of a group communication object inside SIM-KNX. Depending on the send condition, transmission is automatically initiated. The data and its format depend on the configured datapoint type.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
data	data that is written to the group communication object. allowed: single values, hex stream (depends on data point type)
Example:	
ovs (0) 1 *<ovs (0) 1>* ovs ($1) 2100 *<ovs ($1)>2100>*	

Get object value	
ovg *(objectNr)*	
Description:	
Get the value of a group communication object inside SIM-KNX. The data and its format depend on the configured datapoint type.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
Return value:	
data	data that is read from the group communication object. Its data format depends on the datapoint type.
Example:	
ovg (0) *<ovg (0) >1* ovg ($1) *<ovg ($1)>2100>*	

4.4.4 Accessing Group Communication Objects (RAW/ Interoperability Mode)

Send group telegram	
odt *(objectNr)*	
Description:	
Start the transmission of the group communication object value on KNX as 'A_ValueWrite' without checking the sending condition.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
Example:	
odt (0) *<odt (0)>*	

Send group read telegram	
odr *(objectNr)*	
Description:	
Start the request of the group communication object value. An 'A_ValueRead' is transmitted on KNX.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
Example:	
odr (1) *<odr (1)>*	

Get RAM flags	
ofg *(objectNr)*	
Description:	
Get the RAM flags of the object.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
Return value:	
RAM flags	RAM flags of the object.
Example:	
ofg (1) *<ofg (1)> $01*	

Structure of the RAM flags

The RAM flags contain the information about the communication status of the communication object. Usually, only the three indications update, value changed and receive-timeout are of interest to the user application.

Table 12: Structure of RAM Flags

Bit	7	6	5	4	3	2	1	0
Flag	receive-timeout	not used	not used	value changed	update	read	transmission	transmission
	I	0	0	C	U	R	T	T

Table 13: Flag Description

Flag	Flag Name	Description
T	transmission	Information about the state of value transmission on KNX.
R	read	Flag to trigger the sending of 'value read request'. This flag is used in combination with the transmission state.
U	update	This flag indicates a value was received from the bus.
C	value changed on update	This flag indicates the value that was received from the bus changed the communication object value.
I	receive-timeout	This flag is set, when no value was received within the configured time.
0	-	-not used-

Return the update flags of all group objects	
gug	
Description:	
Get the update flags of all objects.	
Return value:	
updateFlags	packed update flags
Example:	
gug *<gug>$01 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00*	

Return the valueChanged flags of all group objects	
gcg	
Description:	
Get the valueChanged flags of all objects.	
Return value:	
valueChangedFlags	packed valueChanged flags
Example:	
gcg *<gcg>$04 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00*	

Return the timeout flags of all group objects	
gtg	
Description:	
Get the timeout flags of all objects.	
Return value:	
timeoutFlags	packed timeout flags
Example:	
gtg *<gtg>$10 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00*	

4.4.5 Configuring Group Communication

4.4.5.1 Group Addresses

Set sending group address	
ogs *(objectNr) group*	
Description:	
Set the sending group address of a group communication object. The previous sending group address, if existing, will further be used, but as the receiving group address. This command modifies the internal flash memory. It is recommended to use this command only for configuration purpose and not on a permanent basis.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
group	sending group address (as one single value) allowed: single values and ETS format
Example:	
ogs (0) $affe *<ogs (0) $affe>* ogs (0) 21/2046 *<ogs (0) 21/2046>* ogs (0) 21/7/254 *<ogs (0) 21/7/254>* ogs (0) 21/7/$fe *<ogs (0) 21/7/$fe>*	

Add group address	
oga *(objectNr) group*	
Description:	
Add a group address to a group communication object as the receiving group address. When there was no group address associated with the object before, this group address automatically becomes the sending group address. This command modifies the internal flash memory. It is recommended to use this command only for configuration purpose and not on a permanent basis.	
Parameter:	
objectNr	number of the group communication object which is manipulated. allowed: single values (according to tables in chapter 1.4)
group	group address (as one single value) allowed: single values and ETS format
Example:	
oga (0) $affe *<oga (0) $affe>* oga (0) 21/2046 *<oga (0) 21/2046>* oga (0) 21/7/254 *<oga (0) 21/7/254>* oga (0) 21/7/$fe *<oga (0) 21/7/$fe>*	

Delete group address	
ogd *(objectNr) group*	
Description:	
Delete one or all group addresses of a group communication object. When a sending group address is deleted, the next group address automatically becomes the sending group address. This command modifies the internal flash memory. It is recommended to use this command only for configuration purpose and not on a permanent basis.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
group	group address (as one single value) allowed: single values and string "all"
Example:	
ogd (0) $4711 *<ogd (0) $4711>* ogd (0) 8/1809 *<ogd (0) 8/1809>* ogd (0) 8/7/17 *<ogd (0) 8/7/17>* ogd (1) "all" *<ogd (1) "all">*	

Get group addresses	
ogg *(objectNr)*	
Description:	
Get the group addresses of a group communication object. The first group address is the sending group address.	
Parameter:	
objectNr	number of the group communication object that is read. allowed: single values (according to tables in chapter 1.4)
Return value:	
group	group addresses as single values in hex format
Example:	
ogg (0) *<ogg (0)>$4711 $affe*	

4.4.5.2 Group Communication Objects

Set object configuration

ocs *(objectNr) DPT objectType comFlags sendConfig rcvConfig time*

Description:

Set the configuration of a group communication object.
To configure the communication object in Interoperability Mode,
set the DPT and have the object type set to 0.
To configure the communication object in RAW Mode,
set the DPT to 0 and set the length by the object type parameter.
When the wildcard '*' is used, the parameter will not be changed.

Parameter:

objectNr	number of the group communication object that is manipulated allowed: single values (according to tables in chapter 1.4)
DPT	datapoint type allowed: single values, * see also part Supported Datapoint Types (DPT)
objectType	object type allowed: single values, * see also part Group Object Types (objectType)
comFlags	configuration flags of the group communication object allowed: single values, * see also part Structure of Configuration Flags (comFlags)
sendConfig	configuration when value is sent allowed: single values, * see also part Send Configuration (sendConfig)
rcvConfig	configuration when indications is sent allowed: single values, * see also part Receive Configuration (rcvConfig)
time	delay time When time is set to 0, the send/receive timeout is disabled. The time base can be changed by sendConfig. allowed: single values, *

Example:

```
ocs (0) 1 0 $df $0001 $0001 0
<ocs (0) 1 0 $df $0001 $0001 0>

ocs ($1) 9 0 $df $0002 $0004 120
<ocs ($1) 9 0 $df $0002 $0004 120>

ocs ($1) 9 * * $0002 * 120
<ocs ($1) 9 * * $0002 * 120>
```

Get object configuration	
ocg *(objectNr)*	
Description:	
Get the configuration of the group communication object.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to tables in chapter 1.4)
Return value:	
DPT	data point type see also part Supported Datapoint Types (DPT)
objectType	object type see also part Group Object Types (objectType)
comFlags	configuration flags of the group communication object see also part Structure of Configuration Flags (comFlags)
sendConfig	configuration when the value is sent see also part Send Configuration (sendConfig)
rcvConfig	configuration when indication is sent see also part Receive Configuration (rcvConfig)
time	delay time
Example:	
ocg(0) *<ocg(0)> 1 0 $df $0001 $0001 0*	

Supported Datapoint Types (DPT)

The following table gives an overview of the DPTs that are supported by SIM-KNX. For information about their usage, please also check the documents from KNX Association, especially the Interworking Datapoint Types System Specifications.

Table 14: DPTs supported by SIM-KNX

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
1	DPT 1 "1-bit"	Format:	*b* *b* {0,1}: value (0=off, 1=on)
		Object Size:	1 bit
		Usage:	switch on / off, move up / down, enable / disable, …
		Example:	ovg (0) *<ovg (0) >1*
2	DPT 2 "1-bit controlled"	Format:	*c v* *c* {0,1}: control (0=off, 1=on) *v* {0,1}: value (0=off, 1=on)
		Object Size:	2 bit
		Usage:	control
		Example:	ovg (1) *<ovg (1)>1 1*
3	DPT 3 "3-bit controlled"	Format:	*c StepCode* *c* {0,1}: control (0=off, 1=on) *StepCode* {000b…111b}: value Interval: $2^{(StepCode-1)}$
		Object Size:	4 bit
		Usage:	dimming control
		Example:	ovg (9) *<ovg (9)>1 5*
4	DPT 4 "character set"	Format:	A_8 character string
		Object Size:	1 byte
		Usage:	text, single charaters, ASCII
		Example:	ovg (120) *<ovg (120)>"h"*

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
5	DPT 5 "8-bit unsigned value"	Format:	U_8 unsigned value {0...100}, allowed: single values (DPT5.001 DPT_Scaling)
		Object Size:	1 byte
		Usage:	scale (0...100 %), absolute dimming
200		Format:	U_8 unsigned value {0...360}, allowed: single values (DPT5.003 DPT_Angle)
		Object Size:	1 byte
		Usage:	angle (0...360°)
201		Format:	U_8 unsigned value {0...255} (DPT5.010 DPT_Value_1_Ucount)
		Object Size:	1 byte
		Usage:	counter pulses
		Example:	ovg (211) *<ovg (211)>54*
	can be used with complex sending conditions		
6	DPT 6 "8-bit signed value"	Format:	V_8 signed value {-128...127}
		Object Size:	1 byte
		Usage:	percent, counter pulses
		Example:	ovg (288) *<ovg (288)>127*
7	DPT 7 "2-octet unsigned value"	Format:	U_{16} unsigned value {0...65535}
		Object Size:	2 byte
		Usage:	counter pulses
		Example:	ovg (76) *<ovg (76)>45698*
8	DPT 8 "2-octet signed value"	Format:	V_{16} signed value {-32768...32767}
		Object Size:	2 byte
		Usage:	counter pulses, percent
		Example:	ovg (86) *<ovg (86)>-25789*
9	DPT 9 "2-octet float value"	Format:	signed float value {-671088.64...670760.96}
		Object Size:	2 byte
		Usage:	temperature, temperature change, brightness, wind speed, pressure, humidity, air quality, air flow, time, voltage, current, power, power density, ...
		Example:	ovg (99) *<ovg (99)>92815.27*
	can be used with complex sending conditions		

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
10	DPT 10 "time"	Format:	$N_3\ U_5\ U_6\ U_6$ N_3 {0...7}: weekday (1=Monday, 7=Sunday, 0=no day) U_5 {0...23}: hour U_6 {0...59}: minute U_6 {0...59}: second
		Object Size:	3 byte
		Example:	ovg (111) *<ovg (111)>4 21 39 11*
11	DPT 11 "date"	Format:	$U_5\ U_4\ U_7$ U_5 {1...31 }: day (valid number of days for the month) U_4 {1...12}: month U_7 {0...99}: year (<90 interpreted as 21th century)
		Object Size:	3 byte
		Example:	ovg (37) *<ovg (37)>25 10 2087*
12	DPT 12 "4-octet unsigned value"	Format:	U_{32} unsigned value {0...4294967295}
		Object Size:	4 byte
		Usage:	counter pulses
		Example:	ovg (66) *<ovg (66)>1234567*
13	DPT 13 "4-octet signed value"	Format:	V_{32} signed value {-2147483648...2147483647} (range is not validated)
		Object Size:	4 byte
		Usage:	counter pulses, flow rate, energy, long time
		Example:	ovg (51) *<ovg (51)>-1234327*
14	DPT 14 "4-octet float value"	Format:	F_{32} float value
		Object Size:	4 byte
		Usage:	value
		Example:	ovg (92) *<ovg (92)>5642.736*
15	DPT 15 "access data"	Format:	*usage not recommended, reserved for Functional Block*
		Object Size:	4 byte
		Usage:	Access identification

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
16	DPT 16 "string"	Format:	A_{112} character string {maximum length: 14 characters} Response always has 14 characters (unused characters were set to 0 <NULL>)
		Object Size:	14 byte
		Usage:	Text, string, fixed length, ASCII (valid for communication objects 112-127)
		Example:	ovg (123) *<ovg (123)>"Hello"*
18	DPT 18 "scene control"	Format:	$C\ U_6$ C {0,1}: 0=control, 1=learn U_6 {0...63}: scene number
		Object Size:	1 byte
		Example:	ovg (870) *<ovg (870)>1 36*
19	DPT 19 "date time"	Format:	$U_8\ U_4\ U_5\ U_3\ U_5\ U_6\ U_6\ B_9$ U_8 {0...255}: year (0 = 1900) U_4 {1...12}: month U_5 {1...31}: day of month U_3 {0...7}: day of week (1=Monday, 7=Sunday, 0=no day) U_5 {0...24}: hour of day U_6 {0...59}: minutes U_6 {0...59}: seconds B_9 {0,1} (valid for communication objects 112-127)
		Object Size:	8 bytes
		Example:	ovg (19) *<ovg (19)>5 3 2020 2 9 8 4 0000001 00000001*
20	DPT 20 "absolute value"	Format:	N_8 N_8 {0...255}: unsigned value
		Object Size:	1 byte
		Usage:	Encoding absolute value
		Example:	ovg (10) *<ovg (10)>225*
21	DPT 21 "8-bit set"	Format:	B_8 B_8 {0,1}
		Object Size:	1 byte
		Usage:	General status and device control
		Example:	ovg (77) *<ovg (77)>10110010*

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
22	DPT 22 "16-bit set"	Format:	*usage not recommended, reserved for Functional Block*
		Object Size:	2 byte
		Usage:	DHW control, HVAC status
23	DPT23 "Enum8"	Format:	N_2
		Object Size:	1 byte
		Usage:	OnOff action, alarm reaction, UpDown action

DPT Examples

Table 15: Dimming Control Examples

Command	Value binary (Control Value)	Action
ovs(9) $1 1	1 001	1/1 brighter (dim to on)
ovs(9) 0 $1	0 001	1/1 darker (dim to off)
ovs(9) 1 8	1 100	⅛ brighter
ovs(9) 0 $3	0 011	¼ darker
ovs(9) 1 $0	1 000	stop dimming

$$Value = \frac{1}{2^{stepcode-1}}$$

Table 16: Temperature[2] Examples

Command
ovs(9) –12.75
ovs(9) 37

Table 17: Scaling Examples

Command	Usage	Action
ovs(9) $13	Wind direction	26.82 °
ovs(9) $A5	Relative Brightness	64.71 %
ovs(9) $DC	Counter	220

Table 18: Time Examples

Command	Action
ovs(9) 1 12 45 59	Monday, 12:45:59
ovs(9) 5 $A $1F $2F	Friday, 10:31:47

Table 19: Date Examples

Command	Action
ovs(9) 24 12 56	24.12.2056
ovs(9) $F $B $5C	15.11.1992

[2] Due to value conversion, it may happen that the value read back from SIM-KNX not exactly equals the originally written one.

Group Object Types (objectType)

Table 20: Group Object Types according to the KNX Handbook Resource Definition

Value (code)	Size	Used Memory
0	1 bit	1 byte
1	2 bit	1 byte
2	3 bit	1 byte
3	4 bit	1 byte
4	5 bit	1 byte
5	6 bit	1 byte
6	7 bit	1 byte
7	1 octet	1 byte
8	2 octets	2 byte
9	3 octets	3 byte
10	4 octets	4 byte
11	6 octets	6 byte
12	8 octets	8 byte
13	10 octets	10 byte
14	14 octets	14 byte

Structure of Configuration Flags (comFlags)

The Configuration Flags contain the information about communication. They are identical to the „Edit Object“ dialog flags in ETS.

Table 21: Structure of Configuration Flags

Bit	7	6	5	4	3	2	1	0
Flag	read response enable	transmit enable	not used	write enable	read enable	communication enable	priority	priority
	U	T	0	W	R	C	P	P

Table 22: Flag Description

Flag	Flag Name	Description
P	priority	Possible values: 3 = low 2 = urgent 1 = high 0 =system (do not use!)
C	communication enable	Enable communication of the object.
R	read enable	Object value can be read from the KNX bus.
W	write enable	Enable receiving from KNX.
T	transmit enable	Enable sending to KNX.
U	read response enable	Object value is updated by a ‘Read Response’. In ETS, this flag is called “Update Flag”.
0	-	-not used-

Default configuration setting is “0xD3”.

Defining the direction of telegrams:

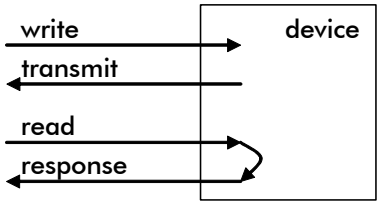

Figure 15: Telegram Direction Definition

Table 23: Default Configuration of the Flag Settings

Bit	7	6	5	4	3	2	1	0
Flag	U	T	0	W	R	C	P	P
Set	yes	yes	-	yes	no	no	yes	yes

Send Configuration (sendConfig)

Table 24: Byte containing Send Conditions Configuration

Bit	Explanation
0	send on receive 1: sends the object value when the value was received from the serial interface
1	send on change (not DPT1) 1: sends the object value when the value received from the serial interface is different than the current value
1	send on falling edge (DPT1 only) 1: sends the object value when the value changes from 1 to 0
2	send on rising edge (DPT1 only) 1: sends the object value when the value changes from 0 to 1
3	reserved (0)
4	reserved (0)
5	reserved (0)
6	select the timer usage 0: send timer 1: receive timer
7	select the time base (timer is activated when time > 0) 0: time in seconds 1: time in minutes
8	reserved (0)
9	reserved (0)
10	reserved (0)
11	reserved (0)
12	reserved (0)
13	reserved (0)
14	reserved (0)
15	reserved (0)

Receive Configuration (rcvConfig)

Table 25: Byte containing Receive Conditions Configuration

Bit	Explanation
0	set usage of indication 0: single indication 1: global indication
1	set format of single indication 1: send object value with indication
2	reserved (0)
3	reserved (0)
4	reserved (0)
5	reserved (0)
6	reserved (0)
7	reserved (0)
8	reserved (0)
9	reserved (0)
10	reserved (0)
11	indication on received value
12	indication on changed value
13	reserved (0)
14	reserved (0)
15	indication on timeout

4.4.6 Indications

Following commands are sent from the module without any requests. They are independent from the response syntax.

Receive the global update flag	
gui	
Description:	
Get the global update flag.	
Return value:	
global updateFlag	bit 0: application restart bit 3: update flag bit 4: changed flag bit 6: timeout on serial interface has occured bit 7: receive-timeout
Example:	
gui $01	

Receive the update data for a communication object	
oui	
Description:	
Indicate data changes on a communication object. Update flag, transmit flag and timeout flag are cleared after indication.	
Return value:	
groupObjectNo	number of the group communication object that was updated
flags	RAM flags of the communication object
(Object Value)	actual communication object value; sending this value depends on rcvConfig.
Example:	
oui $01 $10 oui $01 $10 50	

Receive device state	
dsi	
Description:	
Get the actual state of SIM-KNX.	
Return value:	
bit0	0: normal operation 1: Transparent Mode
bit1	1: application loaded
bit2	1: application is running
bit7	1: device is in Programming Mode
Example:	
dsi $06	

4.5 Command Reference (Transparent mode)

Device transparent set		
dts *data*		
Description:		
Configure Transparent Mode.		
Parameter:		
data	bit0:	1 = enable Transparent Mode 0 = disable Transparent Mode
	bit1:	1 = send Ack on every group telegram (recommended) 0 = no Ack on every group telegram
	bit2:	1 = generate confirmation on serial interface 0 = no confirmation on serial interface
	bit3:	not used
	bit4:	format of address 0 = hex format 1 = format depends on bit5
	bit5:	1 = address as 2-level group address 0 = address as 3-level group address
Example:		
dts 7		

Transparent data send	
tds *(dest) data* **tds** *(dest length) data*	
Description:	
Send a 'Value Write'.	
Parameter:	
dest	ETS group address
length	0 = 1...6 bit (default) 1 = 1 byte ...
data	data that is sent to destination; allowed: single values
Example:	
tds (2/0/0) 2	

Transparent response send	
tes *(dest) data* **tes** *(dest length) data*	
Description:	
Send a 'Value Response'.	
Parameter:	
dest	ETS group address
length	0 = 1...6 bit (default) 1 = 1 byte ...
data	data that is sent to destination; allowed: single values
Example:	
tes ($1000) $01 tes (2/0/1 2) $01 $02	

Transparent send read request	
trs *(dest)*	
Description:	
Send a 'Value Read'.	
Parameter:	
dest	ETS group address
Example:	
trs ($1000)	

Transparent read indication	
tri *(source dest)*	
Description:	
Get a 'Value Read' indication.	
Parameter:	
source	physical address
dest	ETS group address
Example:	
tri ($ffff $1000)	

Transparent data indication	
tdi *(source dest length) data*	
Description:	
Get a 'Value Write' indication.	
Parameter:	
source	physical address
dest	ETS group address
length	0 = 1...6 bit 1 = 1 byte ...
data	data that is received from destination
Example:	
tdi ($ffff $17d0 $06) $01 $00 $00 $00 $00 $01	

Transparent response indication	
tei *(source dest length) data*	
Description:	
Get a 'Value Response' indication.	
Parameter:	
source	physical address
dest	ETS group address
length	0 = 1...6 bit 1 = 1 byte ...
data	data that is received from destination
Example:	
tei ($ffff $17d0 $00) $01	

Transparent read confirmation	
trc *(source dest)*	
Description:	
Get a 'Value Read' confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	physical address
dest	ETS group address
Example:	
trc ($ffff $1000)	

Transparent data confirmation	
tdc *(source dest length) data*	
Description:	
Get a 'Value Write' confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	physical address
dest	ETS group address
length	0 = 1...6 bit 1 = 1 byte ...
data	data that is received from destination
Example:	
tdc ($1508 $0002 $00) $01	

Transparent response confirmation	
tec *(source dest length) data*	
Description:	
Get a 'Value Response' confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	physical address
dest	ETS group address
length	0 = 1...6 bit 1 = 1 byte ...
data	data that is received from destination
Example:	
tec ($1508 $0002 $00) $01	

Transparent read negative confirmation	
trn *(source dest)*	
Description:	
Get a 'Value Read' negative confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	physical address
dest	ETS group address
Example:	
trn ($1508 $1000)	

Transparent data negative confirmation	
tdn *(source dest length) data*	
Description:	
Get a 'Value Write' negative confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	physical address
dest	ETS group address
length	0 = 1...6 bit 1 = 1 byte ...
data	data that is received from destination
Example:	
tdn ($1508 $1000 $00) $01	

Transparent response negative confirmation	
ten *(source dest length) data*	
Description:	
Get a 'Value Response' negative confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	physical address
dest	ETS group address
length	0 = 1...6 bit 1 = 1 byte ...
data	data that is received from destination
Example:	
ten ($1508 $1000 $00) $01	

4.6 Error Codes

Table 26: Error Codes Explanation

errCode	Internal Name	Explanation	Possible Reasons
$0010	TRANSMIT_FAIL	requested transmission of an object failed	• object is already transmitting
$0101	APPL_STOP	requested command is denied, because application was stopped	• illegal memory access via KNX bus • incomplete ETS download • active ETS download
$0102	NO_OBJ	requested object number is invalid	
$0103	NUMBER_EXP	number format of the expected value is missing	
$0104	VALUE_RANGE	value is out of range	
$0105	COMMAND_END	end of command expected	
$0106	TYPE_RANGE	given object type is not allowed	• requested object type exceeds the object size
$0109	TO_MUCH_VALUES	bytestream is too long	
$010b	NUMBER_OR_COMMA	number or comma is expected	
$010d	NUMBER_OF_INDEX	too many indices are given	
$010e	LINK_WRITE	modification of group addresses failed	• deleting a non-existing address • maximum number of addresses reached
$010f	NO_OF_VALUES	number of values in bytestream is incorrect	
$0110	SYNTAX_ERROR	invalid syntax	
$0111	STRING_EXP	expected parameter is a string	
$0112	STRING_SIZE	size of string is too large	
$0113	SOE_GROUPADDR_RANGE	invalid group address	
$0114	SOE_KEY_ALREADY_EXIST	security key is already set	
$0210	TYPE_EXPECTED	command expected	• serial string does not start by command
$0215	ILLEGAL_OPERATION	unknown command	
$0216	NO_PROP	interface object property not found	

errCode	Internal Name	Explanation	Possible Reasons
$0217	IO_ELEMENT	element index is too large for the property	
$0218	RO	property is 'Read Only'	
$0219	TYPE_MISMATCH	object type and DPT are not compatible	
$0220	DPT_NOT_FOUND	given DPT is not supported	
$0221	WRONG_PARAMETER	given parameter is invalid	
$0222	ACCESS_DENIED	accessing the given command is denied	• some commands are disabled after an ETS download
$0223	NO_PARAMETER	the requested 'User Parameter' does not exist	
$0224	TO_MUCH_PARAMETER	too many 'User Parameters' are requested	
$0225	IO_ERROR	property access has failed	

5 Implemented Application Interface Object

The interface object of SIM-KNX can be accessed from KNX side with standard property access mechanisms and from the serial interface side with the commands **ids** (*ioIndex propertyID elementIndex*) *data*) and **idg** (*ioIndex propertyID elementIndex*). For access from the KNX bus tools like the "Device editor" provided with the ETS can be used.


To access the application interface object for both commands **ids** and **idg** the *ioIndex* always is 5.

Table 27: Object Index: 5

Property ID	Type Read/Write	Element Index	Usage
1	PDT_UNSIGNED_INT R		**Object Type** (0xF000)
50	PDT_GENERIC_02 RW		**Parameter of the serial interface** For a detailed description of the 2 bytes used to store the serial interface configuration see chapter 4.1 Communication Parameter Setting. *Example:* *Set default values to the serial interface* ids (5 50) $31 $00
51	PDT_GENERIC_02 RW		**Parameter for Transparent Mode** (see chapter 3.3 in Transparent Mode) *Example:* *Read the Transparent Mode setting* idg (5 51)
52	PDT_GENERIC_01 RW		**Syntax options** (see chapter 4.2.3 Strings from SIM-KNX) *Example:* *Enable responding of "OK if response contains no data"* ids (5 52) $02
53	PDT_GENERIC_02 R		Reserved
54	PDT_GENERIC_01 RW		**Control replacement of text on receiving from the serial interface** 0 = disable text replacement 1 = enable text replacement Original strings and replaced strings are defined in property 136. *Example:* *Enable* replacement of text ids (5 54) $01

Property ID	Type Read/Write	Element Index	Usage
55	PDT_GENERIC_01 RW		**Control the sending of user-defined strings on the serial interface** (standard indication replacement) after receiving group oriented communication objects. 0 = disable 1 = enable
60	Array PDT_GENERIC_06 RW		**Object configuration** Every *elementIndex* configures one communication object Element structure is as follows: • byte 0: Supported Datapoint Types (DPT) • byte 1: delay time • byte 2: upper byte of Send Configuration (sendConfig) • byte 3: lower byte of Send Configuration (sendConfig) • byte 4: upper byte of Receive Configuration (rcvConfig) • byte 5: lower byte of Receive Configuration (rcvConfig)
		1	**Configuration for communication object no. 0**
		…	…
		128 (254)	**Configuration for communication object no. 127 (253)**
96	Array PDT_GENERIC_06 RO		**Manufacturer serial number** *Example:* *Get the serial number (123456789ABC)* idg (5 96) *<idg (5 96)> $00 $72 $01 $02 $03 $04*
128	PDT_GENERIC_01 RW		**Global event generation** (see chapter 4.4.1 Configuration) *Example:* *Read the global event generation setting* idg (5 128)
130	PDT_GENERIC_01 RW		Reserved
132	Array PDT_GENERIC_06 RW		Reserved
133	Array PDT_GENERIC_01 RW		Reserved

Property ID	Type Read/Write	Element Index	Usage
134	Array PDT_GENERIC_01 RW		**User-defined parameter (ETS and local) in non-volatile memory** (See also command **pdg**) *Example:* *Read byte 4 of parameter* idg (5 134 4) *Example:* *Write byte 4 with $D6* ids (5 134 4) $D6
135	Array PDT_GENERIC_01 RW		**Complex sending conditions**
136	Array PDT_GENERIC_01 RW		**String replacements** **(Original strings and replacement strings)** Replacement of complete strings and string fragments that were received from the serial interface before command string is processed for execution. If string is shorter than 32 bytes it is terminated by '0'. For further explanation of the function see "Using SIM-KNX with Doepke fingerprintsensor" (contained in the SIM-KNX EVALUATION KIT documentation CD).
		1...32	1st original string that is to be replaced
		33...64	1st replacement string that is inserted instead of the original one
		65...96	2nd original string that is to be replaced
		97...128	2nd replacement string that is inserted instead of the original one
		...	...
		1985...2016	32nd original string that is be replaced
		2017...2048	32nd replacement string that is inserted instead of the original one

Property ID	Type Read/Write	Element Index	Usage
137	Array PDT_GENERIC_01 RW		**Replacements of indication text on reception** Each replacement canbe described as a structure having 35 bytes: • bytes 1-32: string that is sent • byte 33: object number where indication is replaced • byte 34: value = 0: send string regardless of which value was received value ≠ 0: send string if received value equals the reference value • byte 35: reference value
		1...35	Replacement structure 1
		36...70	Replacement structure 2
		...	...
		1086...1120	Replacement structure 32
140[3]	PDT_GENERIC_01 RW		**Control of the communication timeout supervision of the serial interface - time** • value = 0: disable timeout supervision • value ≠ 0: enable timeout supervision • value is the timeout value *Example:* ids (5 140) $10 If enabled, a timeout is detected as soon the given time expires and no communication on the serial interface takes place. The timeout flag is set and the given object with the given value is sent to the bus. After communication resumes, it depends on configuration if this changed situation is sent to the bus or not. A global update indication that informs about the previous lost communication is sent by the serial interface.

[3] available since protocol version $0106

Property ID	Type Read/Write	Element Index	Usage
141[3]	PDT_GENERIC_02 RW		**Configuration 1 of the communication timeout supervision of the serial interface – behaviour** 1st byte: not used 2nd byte: • bit 7: 0 = time (value of byte 1) in seconds 1 = time (value of byte 1) in minutes • bit 6, 5, 4, 3, 2, 1 not used • bit 0: 0 = do nothing when communication resumes 1 = clear timeout flag when communication resumes and after timeout restart After communication resumes, timeout supervision starts automatically, if enabled (property 140). After a value change of the timeout flag (number in property 142) the object with the value of property 143 is sent.
142[3]	PDT_GENERIC_02 RW		**Configuration 2 of the communication timeout supervision of the serial interface – object number** 1st byte: not used 2nd byte: number of communication object that is used for sending the alarm message "communication lost".
143[3]	PDT_GENERIC_02 RW		**Configuration 3 of the communication timeout supervision of the serial interface – object value** 1st byte: this value is used when timeout flag is set 2nd byte: this value is used when timeout flag is cleared
144[4]	PDT_GENERIC_01 RW		**Device State Generation** Configuration of automatic sending by command **dsi**.
224	PDT_GENERIC_01 RW		Reserved

[4] available since protocol version $0111

6 ETS

6.1 Group Objects

Number of Group Addresses: 254

Number of associations: 254

Number of communication objects: 128 / 254

Address of association table: 0x41FF

Address of communication object table: 0x43FC

6.2 Parameters

Address	Internal Name	Description
0x47FA	serParam	
0x47FC	syntaxOptions	
0x47FE	globalEventGeneration	
0x47FF	serialTimeout	
0x4800	securityLevel	
0x4801	enableReplacement	
0x4802	sendString	
0x4804	setTransparentMode	
0x4806	serialTimeoutConfiguration	
0x4808	setSerialTimeoutObject	
0x480A	serialTimeoutOnValue	
0x480B	serialTimeoutOffValue	
0x480C	deviceStateEventGeneration	
0x4846 + 6*x + 0	objectConfig[x].objEisTypes	Configuration for object "x": Datapoint type (DPT) x is in the range of 0...127
0x4846 + 6*x + 1	objectConfig[x].repTime	Configuration for object "x": Datapoint type (DPT) x is in the range of 0...127
0x4846 + 6*x + 2	objectConfig[x].sendConfig	Configuration for object "x": Datapoint type (DPT) x is in the range of 0...127
0x4846 + 6*x + 4	objectConfig[x].rcvConfig	Configuration for object "x": Datapoint type (DPT) x is in the range of 0...127

Address	Internal Name	Description
0x4E3C + 32*x + 0	complexSendCondition[x].config	Complex sending condition for object “x”: x is in the range of 0...15
0x4E3C + 32*x + 2	complexSendCondition[x].diffSendValue	Complex sending condition for object “x”: x is in the range of 0...15
0x4E3C + 32*x + 6	complexSendCondition[x].hysteresesOn	Complex sending condition for object “x”: x is in the range of 0...15
0x4E3C + 32*x + 10	complexSendCondition[x].hysteresesOff	Complex sending condition for object “x”: x is in the range of 0...15
0x4E3C + 32*x + 14	complexSendCondition[x].objectIndex	Complex sending condition for object “x”: x is in the range of 0...15
0x4E3C + 32*x + 15	complexSendCondition[x].value0	Complex sending condition for object “x”: x is in the range of 0...15
0x4E3C + 32*x + 16	complexSendCondition[x].value1	Complex sending condition for object “x”: x is in the range of 0...15
0x503C ... 0x523B	userParameter	
0x523C + 64*x + 0	sendOnString[x].originalString	String replacement: x is in the range of 0...32
0x523C + 64*x + 32	sendOnString[x].replaceString	String replacement: x is in the range of 0...32
0x5A3C + 35*x + 0	sendStringInd[x].sendString	Send string on received object “x”: x is in the range of 0...32
0x5A3C + 35*x + 32	sendStringInd[x].objectNr	Send string on received object “x”: x is in the range of 0...32
0x5A3C + 35*x + 33	sendStringInd[x].evaluateValue	Send string on received object “x”: x is in the range of 0...32
0x5A3C + 35*x + 34	sendStringInd[x].value	Send string on received object “x”: x is in the range of 0...32

7 SIM-KNX Product Range

SIM-KNX 128-S	SIM-KNX Module with 128 communication objects and straight connector
SIM-KNX 128-R	SIM-KNX Module with 128 communication objects and right angle connector
SIM-KNX 250-S	SIM-KNX Module with 254 communication objects and straight connector
SIM-KNX 250-R	SIM-KNX Module with 254 communication objects and right angle connector
SIM-KNX 128 USB	SIM-KNX Desktop Device with 128 communication objects for USB connection
SIM-KNX 128 RS232	SIM-KNX Desktop Device with 128 communication objects for RS232 connection
SIM-KNX 250 USB	SIM-KNX Desktop Device with 254 communication objects for USB connection
SIM-KNX 250 RS232	SIM-KNX Desktop Device with 254 communication objects for RS232 connection

SIM-KNX 128 DINRAIL	SIM-KNX Rail-mounted Device with 128 communication objects for RS232 connection
SIM-KNX 250 DINRAIL	SIM-KNX Rail-mounted Device with 254 communication objects for RS232 connection
SIM-KNX 128 USB EVAL (see document "SIM-KNX EVAL")	SIM-KNX Evaluation board with mounted SIM-KNX 128-R Module for USB connection
SIM-KNX 128 RS232 EVAL (see document "SIM-KNX EVAL")	SIM-KNX Evaluation board with mounted SIM-KNX 128-R Module for RS232 connection
SIM-KNX 250 USB EVAL (see document "SIM-KNX EVAL")	SIM-KNX Evaluation board with mounted SIM-KNX 254-R Module for USB connection
SIM-KNX 250 RS232 EVAL (see document "SIM-KNX EVAL")	SIM-KNX Evaluation board with mounted SIM-KNX 254-R Module for RS232 connection

8 Glossary

- **Communication Object:**
  see Group Communication Object
- **Datapoint Types (DPT):**
  Standardized format for transmitting data via KNX. The complete list of DPTs is available at KNX Association.
- **Group Address:**
  Group addresses are used to link group communication objects.
  (see Group Communication Object)
- **Group Object:**
  see Group Communication Object
- **Group Communication Object:**
  Group communication objects contain the datapoints which are transmitted via runtime communication. One or more group addresses can be assigned to one group communication object. Always, one of these assigned group addresses is the sending group address. This group address is used to send the group communication object value to the KNX bus. The remaining assigned group addresses, if available, are used to receive values.
  The sending group address can be set with the command **ogs** and by ETS. When the sending group address is deleted, the next group address (of the assignment table) becomes the sending one.


Other terms for group communication object are group object and communication object.

- **Individual Address:**
  see Physical Address
- **Physical Address:**
  This address is the unique device address inside the KNX system. This address is independent of the group addresses and is used for device configuration.


Another term for physical address is individual address.

9 FAQ

9.1 Starting with SIM-KNX

- **What knowledge do I need before starting with SIM-KNX?**

  It is not necessary having detailed knowledge about KNX and protocols.

  But it is very helpful knowing KNX from the installer´s side. Then, you know the basic principles of KNX and the general installation tool ETS. Such knowledge is required to develop your product.

- **Can I use SIM-KNX without connection to KNX?**

  No. SIM-KNX is bus-powered and needs the KNX connection to be functional.

- **Which tools are required to design devices including SIM-KNX?**

  No specific tools are required.

  But it is recommended to use the ETS – then you have the same possibilities as your customer or the installer: changing parameters, monitor telegrams, ...

  If you want to develop the database entry by yourself, you need the manufacturer extension of ETS.

- **What do I need to start developing with SIM-KNX?**

  1. SIM-KNX / SIM-KNX EVALuation board
  2. A communication device that is connected to the serial interface of SIM-KNX, like a PC with a terminal program running or a microcontroller.
  3. A device which reacts on and creates KNX telegrams, preferably within a small KNX system. This may be an arbitrary KNX device or another SIM-KNX device.
  4. It is recommended to use the ETS together with a KNX interface for monitoring telegrams on the bus and for configuring other KNX devices.

- **Where can I get introduced to KNX?**

  Comprehensive information about KNX can be found in the download area of www.knx.org.

  You will also find introductions to KNX on our SIM-KNX product CD (contained in the SIM-KNX evaluation kit).

- **How can I find the correct datapoint type (DPT)?**

  A complete list of defined DPTs and their usage is provided by KNX Association. DPTs for common applications are listed in this document (see also next chapter).

9.2 Configuration of SIM-KNX

- **How can SIM-KNX be configured?**

  SIM-KNX can either be configured via the serial interface or by the ETS.

- **Is it required to configure SIM-KNX each time after restart?**

  No. SIM-KNX stores the configuration data in its non-volatile memory.

- **Is it necessary to configure SIM-KNX via the serial interface, when I want to use an ETS database entry?**

  No. All configuration data can be downloaded by ETS

- **How are devices configured by the installer?**

  Mainly the ETS is used by installers to configure KNX devices.

- **How can I find the correct configuration string?**

  As part of the demo application (Windows and Excel) a configuration manager is included. Here you can set your requirements and the configuration manager calculates the configuration string.

  The configuration manager is described in the SIM-KNX EVAL document.

- **What are the most common used DPTs?**

  DPT 1 "1-bit" is used for switching on and off, enable/disable …

  DPT 5 "8-bit unsigned value" is used for procentual dimming (0-100%).

  DPT 9 "2-octet float value" is used for temperature in units of °C.

- **Where can I find a complete list of DPTs?**

  Complete information about DPTs is contained in the KNX System Specifications, Interworking, Datapoint Types (Chapter 03_07_02). A freely available version is KNX System Specifications Interworking Datapoint Types v01.05.00.

9.3 KNX Certification / KNX Membership

- **What are the benefits of KNX certification?**

  In general, you gain a much higher acceptance on the market.

  You are allowed to use the KNX logo on your product, in related documents and for advertising and commercial material.

  You can distribute the ETS database entry as product database.

- **Why is the SIM-KNX module/device not certified?**

  To be precise, SIM-KNX is not a complete KNX end device.

  The communication system that is contained in SIM-KNX is KNX-certified. It is used in various KNX products in high volume.

  Due to KNX rules, not only the communication system but also the application interworking is KNX-certified. Also the datapoint types that are used will be checked during a certification process. The correct usage of datapoint types cannot be guaranteed by SIM-KNX. This is part of the individual application buildup.

- **What is necessary if I want to certify a product that makes use of SIM-KNX?**

  According to the KNX rules, following requirements have to be fulfilled:

  1. You must be a KNX member.
  2. You need a quality management system.
  3. The product must be fully consistent with the KNX specification.

- **Where can I get information about KNX membership and KNX certification?**

  General information about KNX membership and KNX certification can be found on the KNX homepage (www.knx.org) or can be retrieved from the certification department of KNX Association in Brussels.

  The TAPKO KNX Testlab offers full support for KNX certification. A document that describes all the necessary testing and TAPKO services in detail is available.

9.4 SIM-KNX and ETS

- **How can I use SIM-KNX together with the ETS?**

  There are various possibilities to use SIM-KNX together with the ETS. The choice depends on the effort and certification stage you want to reach for your product.

  Stage 1: generic project database provided by TAPKO
  Stage 2: specific project database provided by yourself
  Stage 3: specific KNX-certified product database

  If you decide to use a database at an early stage, you are not bound to it. You can change it to another stage later.

- **What is the difference between product database entry and project database?**

  The product database is only related to certified KNX products. This is the usual way for manufacturers to distribute the KNX database for their products.

  Project databases are used to export/transfer projects between ETS installations. Product databases can be contained in uncertified products. This way can be used to distribute database entries for uncertified products.

- **Are there differences to be regarded by the installer for product/project database handling?**

  Yes, the product databases can be selected in the ETS product catalogue by choosing the manufacturer among further criteria.

  With project databases the user has to import the distributed project, and then copy the device of interest into his project.

- **What tool do I need to create a database entry for the ETS?**

  To create an ETS database entry, you need the ETS manufacturer extension. It is available for KNX members and can be retrieved from KNX Association.

9.5 KNX Information and Product Support

- **Who offers support for my development?**

  Of course, TAPKO offers full support for your developments, no matter at what stage.

  One of TAPKO´s core businesses is universal support for all kinds of KNX developments, problems and tasks.

- **Where can I get further information about the KNX system?**

  You can get further information from the KNX Association in Brussels (www.knx.org) and the KNX National organisations (e.g. ZVEI in Germany: www.knx.de).

SIM-KNX

Product:

Serial Interface for KNX

Doctype:

Technical & Application Description

Release Number / Release Date:

R1.4 / May 2019

Editor:

Peter Hauner

Web:

www.tapko.de/sim-knx

Contact:

info@tapko.de

Telephone:

+49 941 30747-0

TAPKO Technologies GmbH
Im Gewerbepark A15
93059 Regensburg
Germany