KNX Cookbook

KNX development on basis of existing system components

Components by Weinzierl Engineering

Summary

This document is a development help for KNX newcomers.

This document describes the development of a KNX device based on existing KNX components provided by the company Weinzierl Engineering.

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

Document updates

Version	Date	Modifications
1.0.0	2011.05.13	Preparation of the Final version.
1.01.01	2011.08.11	Update according final Manufacturer Tool release.
01.01.02	2013.10.14	Editorial updates for the publication of KNX Specifications 2.1.

References

[01] Chapter 3/1/2 “Glossary”
[02] Chapter 3/6/2 “Physical External Interface”
[03] Volume 6 “Profiles”

Filename: 02_05_03 Components by Weinzierl v01.01.02.docx
Version: 01.01.02
Status: Final version
Savedate: 2013.10.14
Number of pages: 17

Contents

1 Developing applications on the basis of Weinzierl Component KNX BAOS

1.1 Needed software

- Installed MT4 and valid license.
- Installed ETS4 and valid license.
- Weinzierl KNX BAOS Starter Kit.
- KnxBaosClient SW (included in the Starter Kit).

1.2 About the Weinzierl KNX BAOS Starter Kit

- BAOS = Bus Access and Object Server
- This product is based on the Weinzierl KNX BAOS Modul 820 device, in summary:
  - Profile Class = System 7, see [03].
  - interface = serial asynchronous
  - FT1.2 based protocol, see [02].
- The Weinzierl KNX BAOS Starter Kit contains:
  - 1 development board (PCB)
  - 2 KNX BAOS 820 Modules
  - 1 Power Supply (9 V)
- KNX BAOS Modules come delivered with 250 pre-defined GOs
- KNX BAOS Modules have additionally 16 freely to use parameters, which however will not be discussed here because they are not needed for our sample. In summary:
  - readable via the UART interface
  - address = 4900h
  - length per parameter = 1 byte

2 Weinzierl KNX BAOS Solution

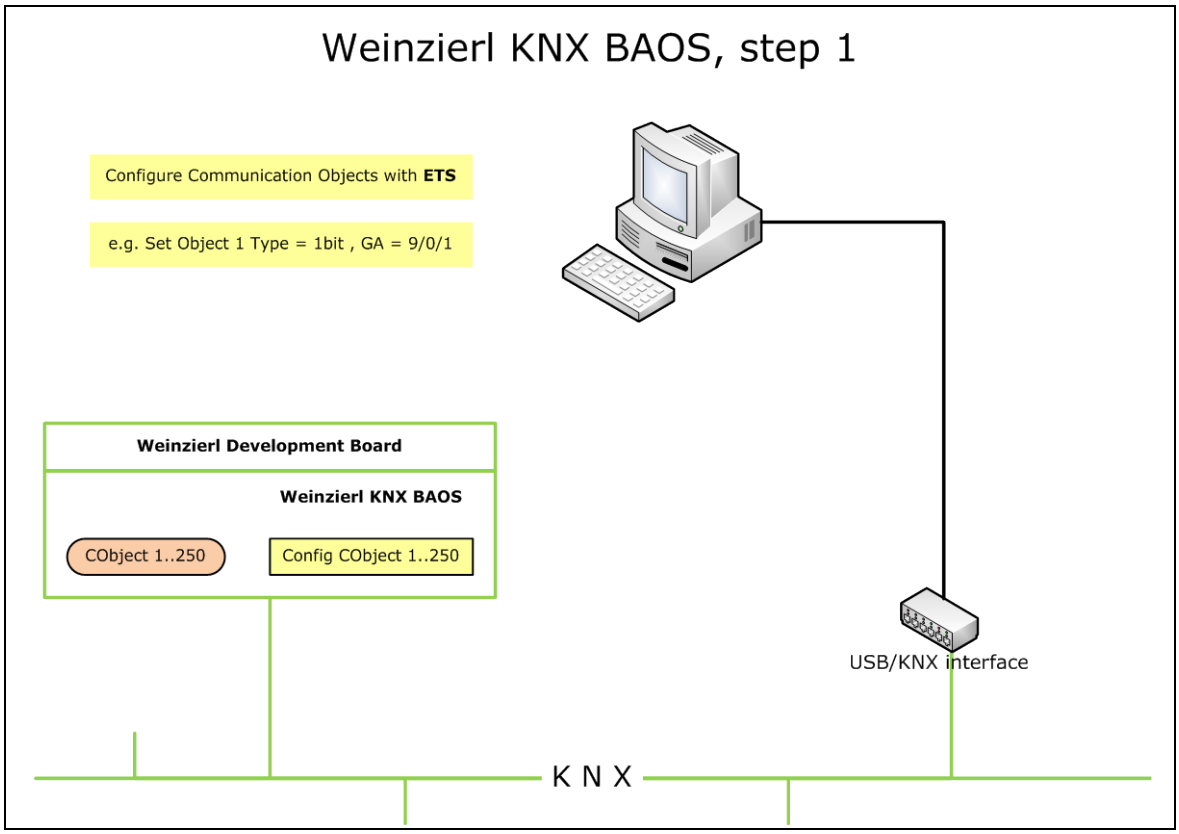
2.1 What you should know about the Weinzierl KNX BAOS GOs

- 250 pre-defined GOs available
- GO Table address = 4400h
- Association Table address = 4200h
- GA Table address = 4000h
- The length of every single GO can be configured, within the following boundaries:
  - GO [0] is not available
  - GO [1..32] -> length = 1 bit .. 14 bytes
  - GO [33..250] -> length = 1 bit .. 4 bytes
- In our sample however we will only use the first 8 GOs.

2.2 Concept of the Weinzierl KNX BAOS Solution

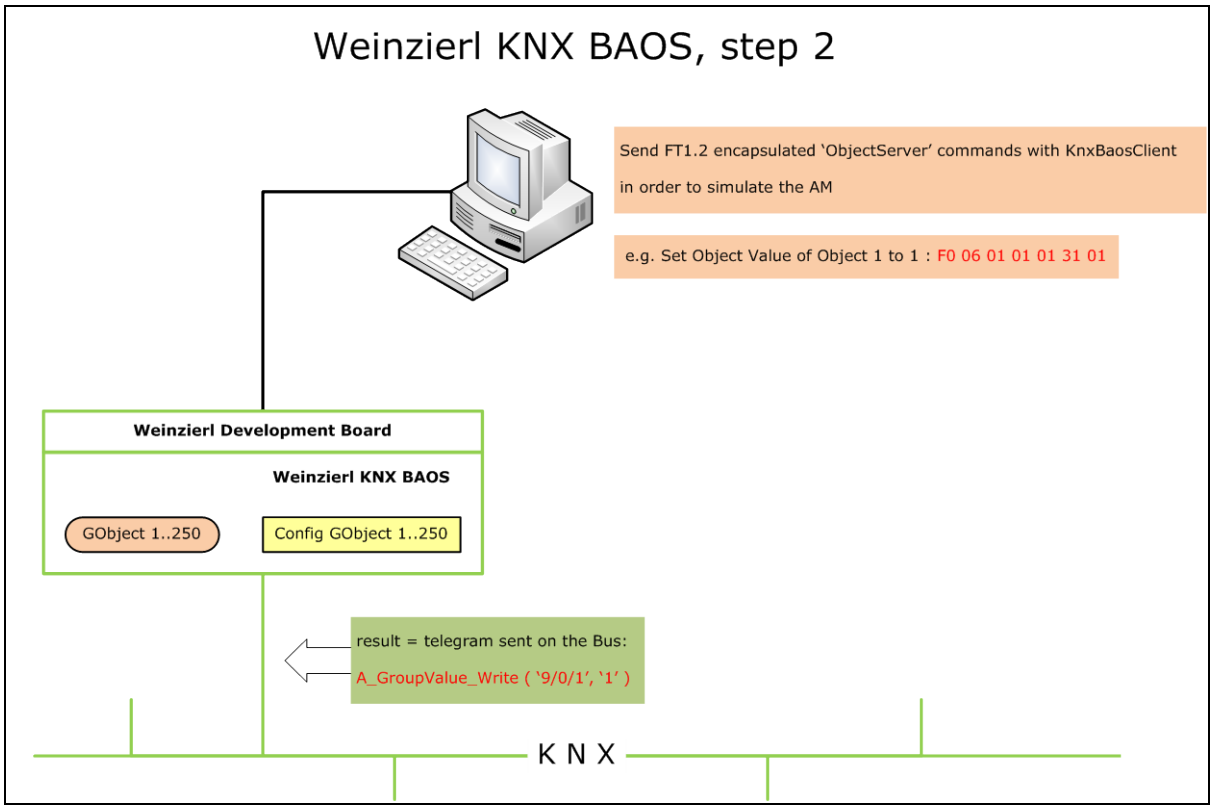
2.2.1 Step 1

- Use the Development Kit = KNX BAOS + Development Board
- Configure the GOs with ETS



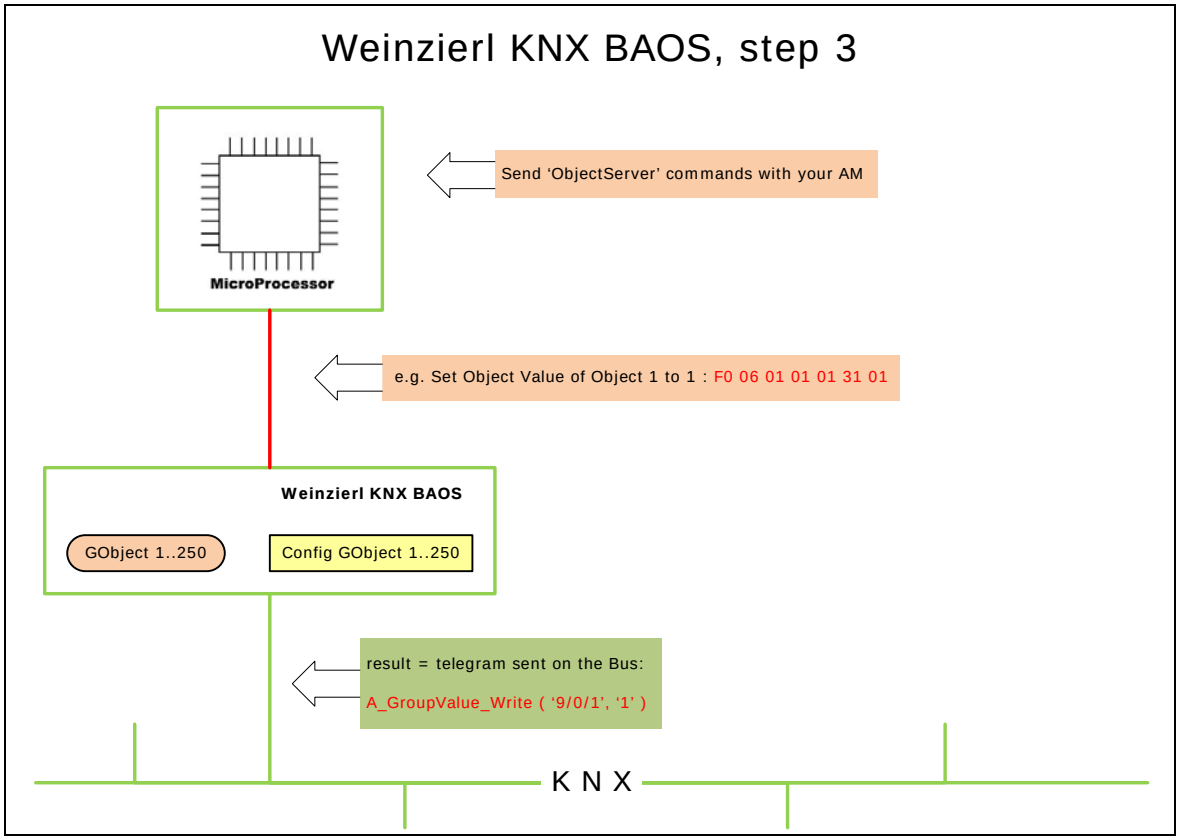
2.2.2 Step 2

- Simulate the AM by means of ‘ObjectServer’ commands.
- Send FT1.2 encapsulated commands from the PC to the Development Board via RS232.
- The connection with the Development Board is established via UART, which is part of the KNX BAOS.
- Check for all GOs.
- For more details about the ObjectServer Protocol please contact www.weinzierl.de



2.2.3 Step 3

- Replace the evaluation device by a regular KNX BAOS device.
- Connect your AM via UART.
- Make sure that the AM sends the same 'ObjectServer' commands as in step 2.
- Make sure to send these commands as FT1.2.
- Check for all GOs



2.3 Phase 1: Evaluation

- Use the Weinzierl KNX BAOS Module + Evaluation Board.
- Create an ETS4 product with MT4 in order to pre-configure the required GOs for the AP.
  - Define the individual types.
  - Set up the required parameter structure for the actual GO setup.
- Add the product to an ETS4 database/project.
  - Link its GOs with GAs.
  - Set its parameters.
  - Set its GOs flags.
  - Program the device (with ETS via KNX).
- Use the KnxBaosClient SW to check whether ETS4 did configure the device correctly.
  - Connect it to both KNX (TP) and your computer (RS232).
  - Play by sending BAOS protocol commands via the KnxBoasClient SW.

- Use the KnxBaosClient SW in order to simulate the AM -> sending BAOS protocol commands in order to set the values of the individual GOs.

2.4 Phase 2: AM integration

Reaching this stage really means you should get in contact with Weinzierl.

The work to be done in this phase comes basically down to the following.

- Dismount the KNX BAOS Module from the Evaluation Board.
- Integrate the KNX BAOS Module with the AM.
- Test: make sure the same commands are sent as during the evaluation phase.
- Certify the entire product, this counts for both:
  - the HW, being the Weinzierl KNX BAOS Module with integrated AM
  - the AP

3 In practice

3.1 Group Object vs. Communication Object

- First of all, the terms ‘Group Object’ and ‘Communication Object’ are synonyms.
- ‘Communication Object’ is used in MT, ETS and other tools.
- ‘Group Object’ is the only term used in the rest of the KNX specifications and is therefore considered as the only correct one.
- Both terms will however be used in this Cookbook because it is here were practice and theory meet. ‘Communication Object’ will only be used when absolutely necessary, e.g. when the context is MT.

3.2 Install the KnxBaosClient application

- Link: http://www.weinzierl.de/
- Screenshot

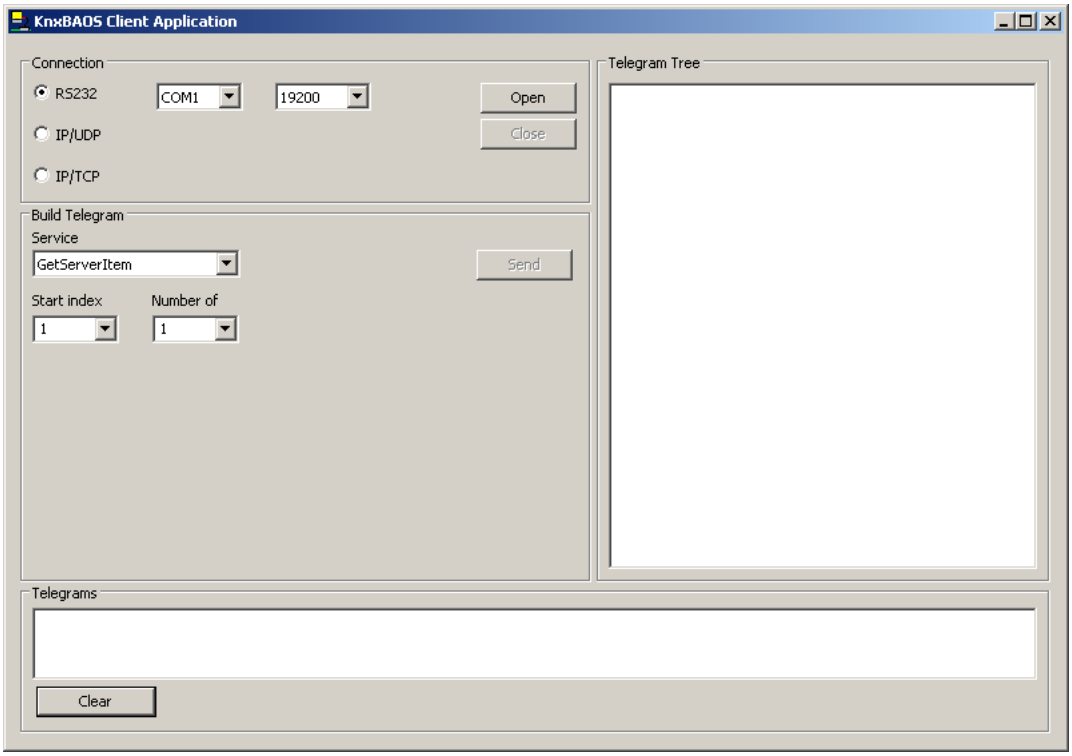


Figure 1 – KnxBAOS Client screenshot

Connect and Play

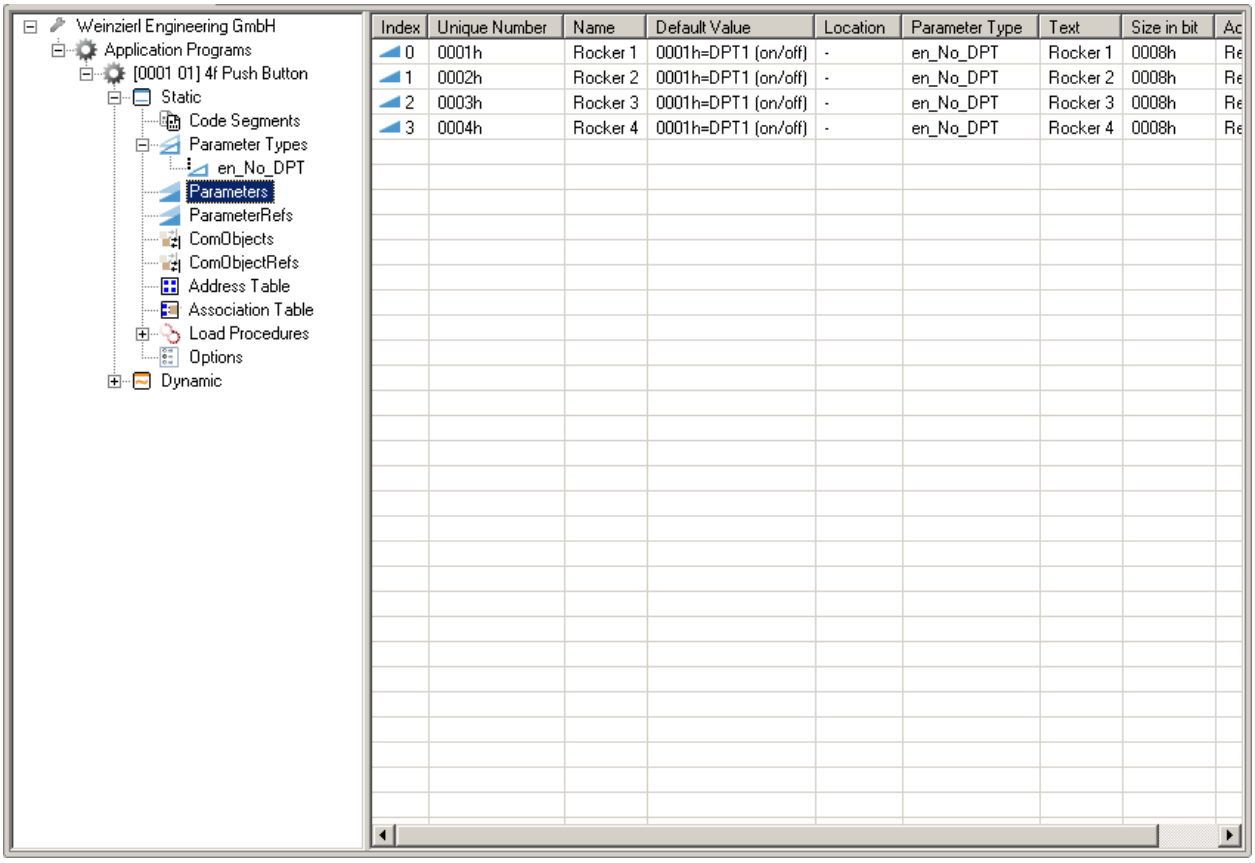
- Wire the device, for RS232 use the connector indicated as ‘K20 BAOS’.
- Open the KnxBaosClient software (see Figure 1).
- Select RS232, your COM port, 19200.
- Click the Open button.
- Select Service = GetServerItem, Start index= 1, Number= 1 then click Send. This is sufficient to test the connection.
- The device’s reaction can be seen in the Telegrams sections, any reaction obviously means the connection is OK.
- Any telegram (sent or received) can be analyzed within the section Telegram Tree of this application.

3.3 Create an ETS4 product with MT4

- Add Project
  - KNX Manufacturer Tool Project
  - Name = KNXproduct100203-01
  - Target ETS Version = ETS4
- Add Hardware
  - Serial Number = SN17.06.2010-01
  - Name = PCB SN17.06.2010-01
  - Version = 1
- Define Product (= commercial realization of HW)
  - Order Number = 4f-PB SN17.06.2010-01
  - Product Name = 4 fold Push Button V1.0
  - Language
- Add Application Program
  - Number = 1 & Version = 1
  - Name = 4f Push Button
  - Mask Version = 7.1
- Open Hardware
  - select Application Programs
  - add new Hardware2Program
  - Select 4f Push Button
- AP Properties
  - Dynamic Table Management = True
- Open the AP, in static
- Import binary data -> select BAOS_870.s19, this will:

- Add Code Segments
  - Segment 0: Address = 4000h, Size = 01FFh
  - Segment 1: Address = 4200h, Size = 01FFh
  - Segment 2: Address = 0700h, Size = 05E6h
  - Segment 3: Address = 4400h, Size = 0408h
  - Segment 4: Address = 4810h, Size = 00F0h
  - Segment 5: Address = 4900h, Size = 0010h
- Add Address Table
  - Code Segment -> [4000h], Max Entries = 00FEh, Offset=0
- Add Association Table
  - Code Segment -> [4200h], Max Entries = 00FFh, Offset=0
- Add Load Controls
  - LC00: Connect
  - LC01: Unload, LSM Index=1
  - LC02: Unload, LSM Index=2
  - LC03: Unload, LSM Index=3
  - LC04: Load, LSM Index=1
  - LC05: AbsSegment, Access=F2h , Address=4000h, LSM Index=1, MemType=3, SegFlags=80h, SegType=0, Size=01FFh
  - LC06: TaskSegment, Address=4000h, ApplicationNumber=0, ApplicationVersion=0, LSM Index=1, Manufacturer ID=0, PEI Type=0
  - LC07: LoadCompleted, LSM Index=1
  - LC08: Load, LSM Index=2
  - LC09: AbsSegment, Access=F2h , Address=4200h, LSM Index=2, MemType=3, SegFlags=80h, SegType=0, Size=01FFh
  - LC10: TaskSegment, Address=4200h, ApplicationNumber=0, ApplicationVersion=0, LSM Index=2, Manufacturer ID=0, PEI Type=0
  - LC11: LoadCompleted, LSM Index=2
  - LC12: Load, LSM Index=3
  - LC13: AbsSegment, Access=F1h , Address=0700h, LSM Index=3, MemType=2, SegFlags=0, SegType=0, Size=05E6h
  - LC14: AbsSegment, Access=F1h , Address=4400h, LSM Index=3, MemType=3, SegFlags=80h, SegType=0, Size=0408h
  - LC15: AbsSegment, Access=FFh , Address=4810h, LSM Index=3, MemType=3, SegFlags=80h, SegType=0, Size=00F0h
  - LC16: AbsSegment, Access=FFh , Address=4900h, LSM Index=3, MemType=3, SegFlags=0, SegType=0, Size=0010h
  - LC17: TaskSegment, Address=4800h, ApplicationNumber=0, ApplicationVersion=0, LSM Index=3, Manufacturer ID=0, PEI Type=0
  - LC18: TaskCtrl1, Address=0, Count=0, LSM Index=3
  - LC19: LoadCompleted LSM Index=3
  - LC20: Restart
  - LC21: Disconnect

- Add Parameter Types
  - Restriction : name = en_No_DPT, range: DPT1 (on/off) = 1, DPT3 (dimming) = 3, No Function = 99
- Add Parameters
  - Virtual : Rocker [1..4], en_No_DPT , default = DPT1
- Add Communication Objects
  - Code segment = 4400h, offset = 0
  - Object #0..8 : Name = object [0..8] , Object Size = 14 bytes
- Add Communication Objects References
  - Object #1..4 : Communication Object = object [1..4], Name = object [1..4] DTP1, Object Size = 1 bit, Text = on/off
  - Object #5..8 : Communication Object = object [1..4], Name = object [5..8] DTP3, Object Size = 4 bit, Text = dimming
- Result:



Index	Unique Number	Name	Default Value	Location	Parameter Type	Text	Size in bit	Ac
0	0001h	Rocker 1	0001h=DPT1 (on/off)	-	en_No_DPT	Rocker 1	0008h	Re
1	0002h	Rocker 2	0001h=DPT1 (on/off)	-	en_No_DPT	Rocker 2	0008h	Re
2	0003h	Rocker 3	0001h=DPT1 (on/off)	-	en_No_DPT	Rocker 3	0008h	Re
3	0004h	Rocker 4	0001h=DPT1 (on/off)	-	en_No_DPT	Rocker 4	0008h	Re

In the dynamic part of the AP:

- Add Channel: Name = Channel 1, Text = Rocker 1
  - Add ParameterBlock: Name = Page1, Text = parameter
  - Add Choose: Parameter = Rocker 1
  - Add When: default = False, test: <99

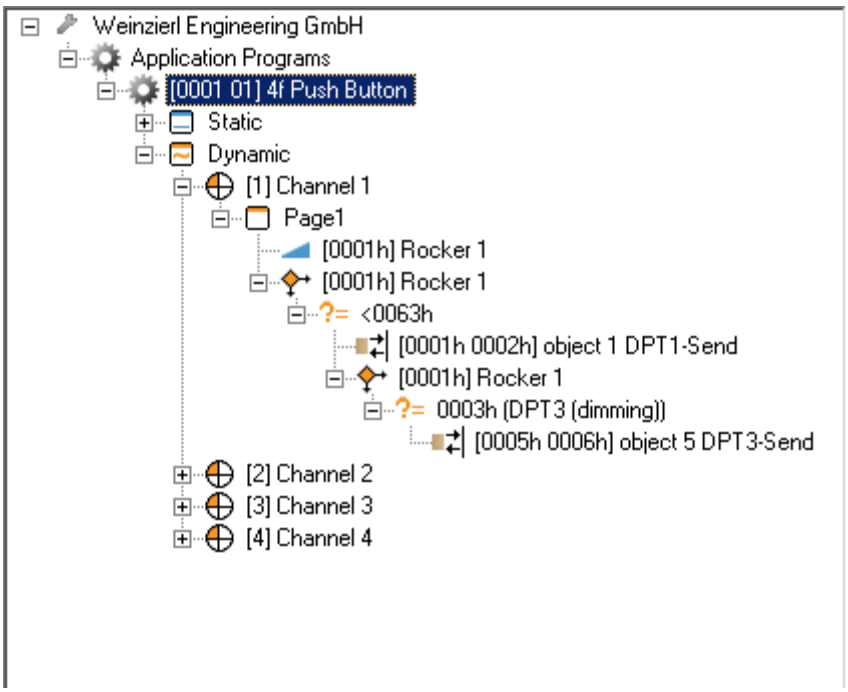
NOTE 1 '99' is the value for 'No Function'

    - Add ComObjectRefRef: object 1 DPT1
    - Add Choose: Parameter = Rocker 1
      - Add When: default = False, test: 3

        NOTE 2 Rocker 1 has dim function => add dim object

      - Add ComObjectRefRef: object 5 DPT3
- Add Channel: Name = Channel 1, Text = Rocker 2
  - Add ParameterBlock: Name = Page1, Text = parameter
  - Add Choose: Parameter = Rocker 2
  - Add When: default = False, test: <99
    - Add ComObjectRefRef: object 2 DPT1
    - Add Choose: Parameter = Rocker 2
      - Add When: default = False, test: 3
      - Add ComObjectRefRef: object 6 DPT3
- Add Channel: Name = Channel 1, Text = Rocker 3
  - Add ParameterBlock: Name = Page1, Text = parameter
  - Add Choose: Parameter = Rocker 3
  - Add When: default = False, test: <99
    - Add ComObjectRefRef: object 3 DPT1
    - Add Choose: Parameter = Rocker 3
      - Add When: default = False, test: 3
      - Add ComObjectRefRef: object 7 DPT3
- Add Channel: Name = Channel 1, Text = Rocker 4
  - Add ParameterBlock: Name = Page1, Text = parameter
  - Add Choose: Parameter = Rocker 4
  - Add When: default = False, test: <99
    - Add ComObjectRefRef: object 4 DPT1
    - Add Choose: Parameter = Rocker 2
      - Add When: default = False, test: 3
      - Add ComObjectRefRef: object 8 DPT3

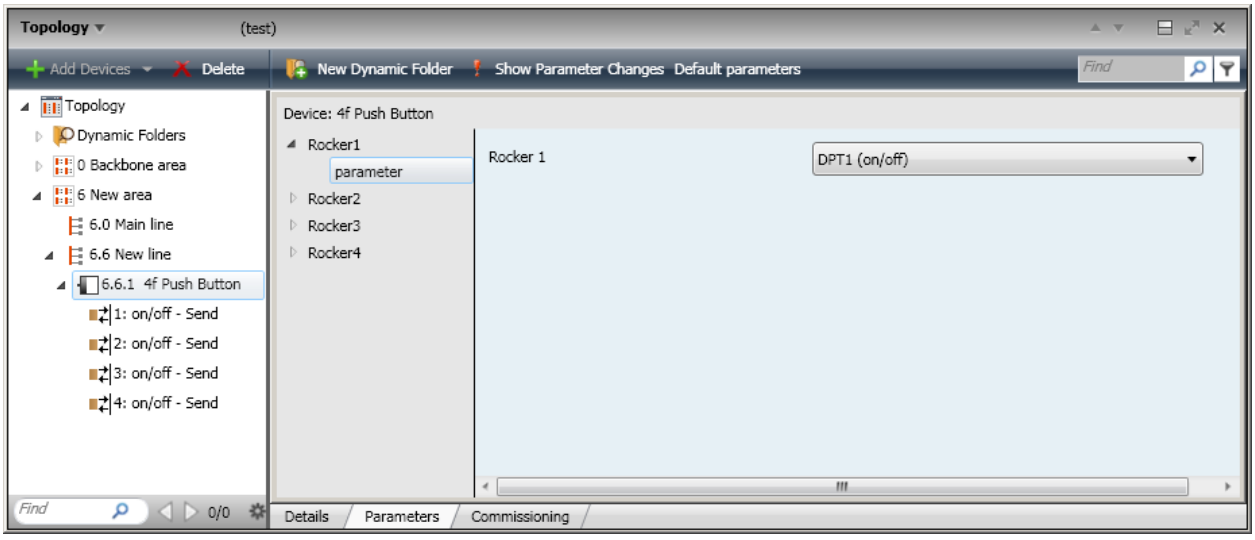
- Result:



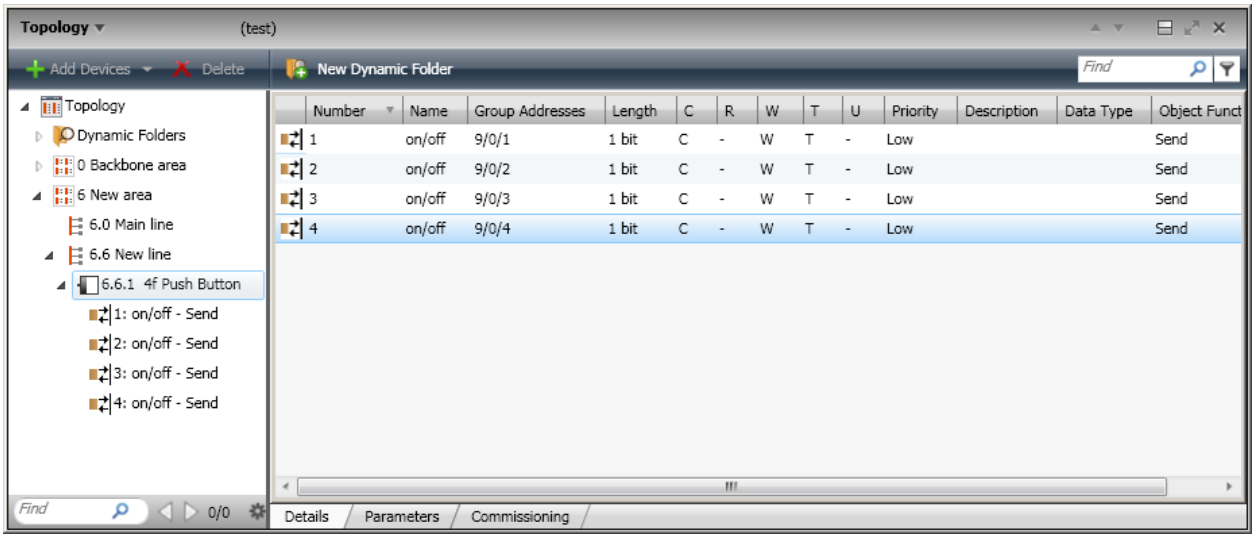
3.4 Import into ETS4

- build the project in MT4
- create a test project via the ‘Edit’ menu in MT4
- import the test project in ETS4
- the device is then visible in the ‘Device’ panel under ‘All Devices’
- link its GOs with GAs, in our example:
  - 9/0/1, 9/0/2, 9/0/3, 9/0/4 for object 1..4 (on/off)
  - 9/1/1, 9/1/2, 9/1/3, 9/1/4 for object 5..8 (dimming)
- set the individual GOs flags
- program the device (with ETS via KNX)

- ETS4 Screenshots
  - Parameter for Rocker1

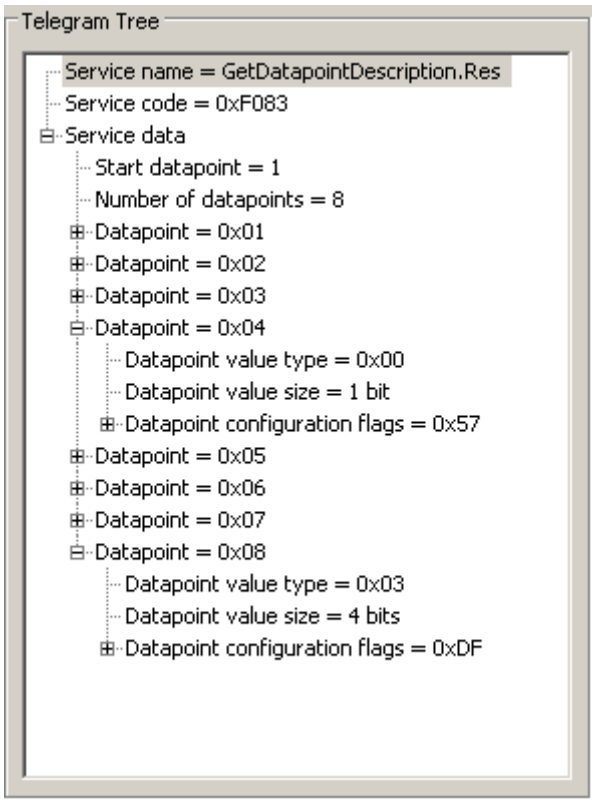


  - Group Objects



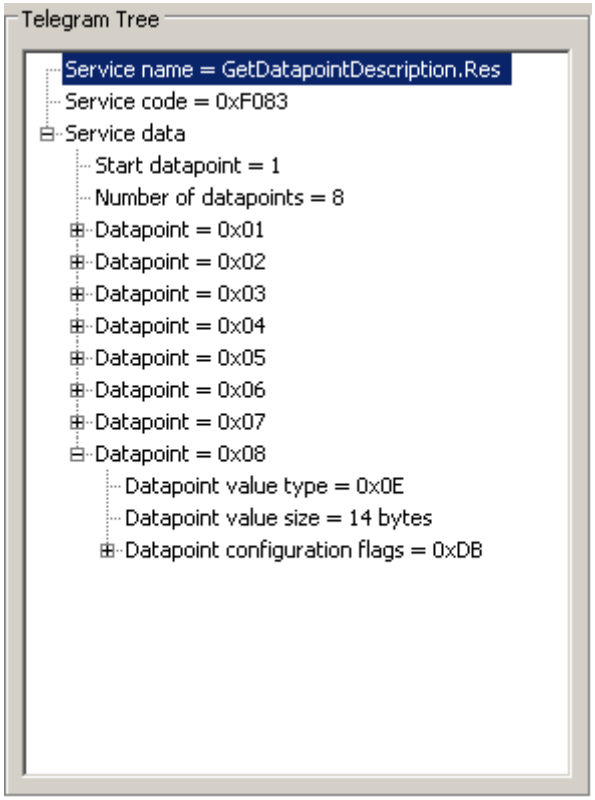
3.5 Use KnxBaosClient to check whether ETS did configure the device correctly

- Connect via RS232.
- Send the following command.
  - Service = GetDatapointDescription
  - Start index = 1
  - Number of = 8
  - example: all Rockers set to DPT3.007 (relative dimming)
  - Result can be checked in the ‘Telegram Tree’ area



EXAMPLE Datapoint = 0x04 is GO 4, which size was indeed set to 1 bit
EXAMPLE Datapoint = 0x08 is GO 8, which size was indeed set to 4 bits

- Setting Rocker 4 back to DPT1 -> size of GO 8 = 14 bytes (not used)



3.6 Use KnxBaosClient in order to simulate the AM

- Connect via RS232.
- Send the following commands.
  - Service = SetDatapointValue
  - Start index = 1
  - Number of = 1
  - Command = Set new value and send on bus
  - Data 0 = 01
  - The expected result is an A_GroupValue_Write telegram for GA 9/0/1 on the bus.
  - This can be checked in three ways:
    - with an actual actuator - the outputs should switch on – or,
    - with the ETS4 Group- or Busmonitor
    - with the command
      - Service = GetDatapointValue
      - Start index = 1
      - Number of = 1
  - Repeat for all other GOs.

