System Conformance Testing

8

Testing of API/PEI/EMI-IMI

6

Testing of EMI-IMI (Local Service Testing)

3

Summary

This document contains the specifications for testing of the External and Internal Message Interface.

Version 01.02.01 is a KNX Approved Standard.

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

Document Updates

Version	Date	Modifications
1.0	2001.10.15	Approved Standard
1.1	2005-02-25	Preparation for release for voting – integration of tests for cEMI – integration of AN019
1.1dedup1	2005-05-30	Small editorial corrections - - Submission to RfV
1.1_FV_a	2005-11-23	Resolution of comments of Release for voting – preparation for KonCert November plenary meeting
1.1_FV_b	2006-05	Resolution of last comment – preparation for final voting
1.1_FV_c	2006-09	Resolution of additional KonCert comment – preparation for final voting
1.1_AS	2007-04	No comments in final voting – publication as AS
1.2	2009-06	Readying document for publication in V2.0 of the KNX specifications
01.02.01	2013.10.24	Editorial updates for the publication of KNX Specifications 2.1.

Filename: 08_06_03 System Conformace Testing - EMI-IMI Tests v01.02.01 AS.docx
Version: 01.02.01
Status: Approved Standard
Savedate: 2013.10.24
Number of pages: 100

Contents

1 Introduction

Underneath only server tests are defined. Local Services are tested only on devices having a PEI. If the BDUT has no PEI, the tests are not relevant.

Notes :

1) All local services covered by these tests are specified in Vol. 3/6/3 of this handbook or Application Notes AN 33 (cEMI protocol specification) and AN 64 (Function Properties local management). These test specifications do not give any indication on which services shall be supported by which profile. For this, please consult Volume 6 as well as the relevant chapters of Volume 3/6/3.
2) The underneath frames if sent on the bus are depicted in TP1 format. For any other medium the telegram formats have to be adapted respectively the acceptance criteria according to the relevant Link Layer specifications. It shall moreover be taken into account that for particular media (e.g. TP0), a Link Layer Acknowledge is only sent when the appropriate flag is set in case of group or individual addressing (for further details see appropriate chapters in Volume 3/1).
3) For local services an error handling mechanism has not (yet) been defined.
4) The message codes (mc) are defined in Volume 3 part 6 chapter 3 (External Message Interface) or in Application Notes AN 33 (cEMI protocol specification) and AN 64 (cEMI adaptations, M_PropFunction local management)
5) The Internal Message Interface (IMI) is closely related to the external message interface (EMI), which is accessible by local services via the PEI: in BCU1 the EMI is identical to the IMI[1] (EMI1=IMI1), as in BCU2 with PEI type 10 (EMI2=IMI2). Only in BCU2 with PEI type 12 or 16 the PEI module translates between IMI2 and EMI1 and vice versa. The main difference between EMI/IMI1 and EMI/IMI2 lies in the different message codes and the fact that EMI/IMI2 contains all the services of EMI/IMI1 plus additional services..
6) underneath the terminology **direct** implies sending the message via a PC tool connected to an RS232 stuck directly to the BDUT, while **indirect** implies sending the message via a second PC tool connected to another RS232 (see figure underneath)
7) all tests shall be carried out once with and once without load generator.

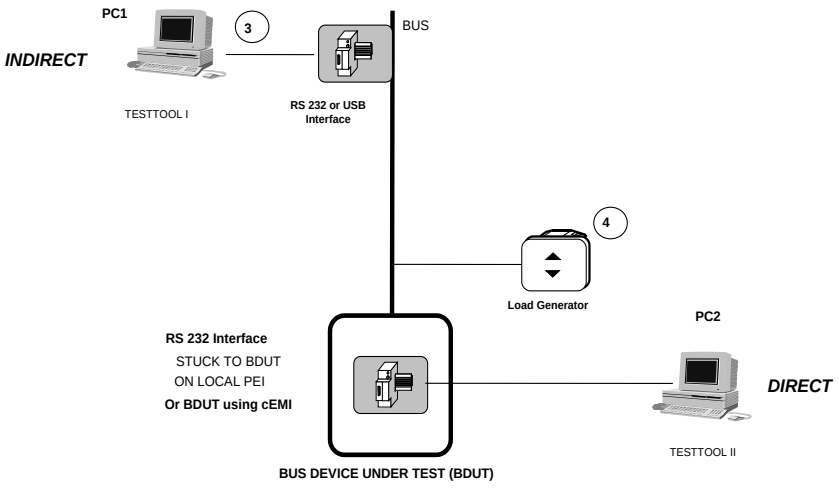


Figure 1: Test Set-up for local service testing

[1] the format is identical but the message codes are different (start with 4 for EMI upstream)

2 Testing of Busmonitor services

2.1 Additional Test-Setup

1. Set the BDUT in busmonitor-mode.

2.2 L_Busmon.ind-Service

• Step 1 (indirect)

Purpose

Check whether the BDUT correctly reports an unacknowledged frame on the bus.

Procedure

Send a frame on the bus that is not acknowledged.

Criteria of acceptance

The BDUT sends the L_Busmon.ind specified below:

Service	Trans.-Flags	Time		Data
Mc	XX_H	XX_H	XX_H	Frame sent (with checksum)

• Step 2 (indirect)

Purpose

Check whether the BDUT reports an acknowledged frame on the bus correctly.

Procedure

Send a frame on the bus that is acknowledged.

Criteria of acceptance

The BDUT sends the two L_Busmon.ind specified below:

1.

Service	Trans.-Flags	Time		Data
mc	XX_H	XX_H	XX_H	Frame sent (with checksum)

2.

Service	Trans.-Flags	Time		Data
mc	XX_H	XX_H	XX_H	ACK-character

2.3 L_PlainData.req Service

• Step 1 (direct)

Purpose

Check whether the BDUT sends a frame on the bus when it receives a L_PlainData.req.

Procedure

Send a L_PlainData.req message to the BDUT.

	unused	time				Data
m_code	00	00	00	00	00	BC AFFE 0000 60 C2

Criteria of acceptance

The BDUT sends the corresponding frame on the bus:

BC AFFE 0000 60 C2

3 Testing of Link Layer services

3.1 L_Data.req/con/ind Services

3.1.1 Test-Setup

1. Select the Link Layer as working layer for the BDUT.
2. Load the BDUT with the following address table: 02h,1000h, FFFEh
3. Install a device in the test set-up acknowledging only the Group Address FFFFh.

3.1.2 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a L_Data.req. Check whether the BDUT also transmits a frame L_Data.con.

Procedure

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read :

Service	Control Field	Source Address		Destination Address				
mc	0Ch	00h	00h	FFh	FFh	E1h	00h	00h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus corresponding to the L_Data.req:

Control Field	Source Address		Destination Address					Check octet
BCh	10h	00h	FFh	FFh	E1h	00h	00h	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following L_Data.con when it receives an IACK on the bus:

Service	Control Field	Source Address		Destination Address				
mc	BCh	10h	00h	FFh	FFh	E1h	00h	00h

- **Step 2 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a L_Data.req with the maximum value length.

Procedure

The BDUT receives a L_Data.request, e.g. Group address type - APCI=ValueWrite :

Service	Control Field	Source Address		Destination Address				
mc	0Ch	00h	00h	FFh	FFh	EFh	00h	80h

Value
81h,82h,83h,84h,85h,86h,87h,88h,89h,8Ah,8Bh,8Ch,8Dh,8Eh

Criteria of acceptance

The BDUT transmits the data packet on the KNX bus corresponding to the L_Data.req:

Control Field	Source Address		Destination Address				
BCh	10h	00h	FFh	FFh	EFh	00h	80h

Value	Check octet
81h,82h,83h,84h,85h,86h,87h,88h,89h,8Ah,8Bh,8Ch,8Dh,8Eh	XXh

The BDUT also sends an according L_Data.con service:

Service	Control Field	Source Address		Destination Address				
mc	BCh	10h	00h	FFh	FFh	EFh	00h	80h

Value
81h,82h,83h,84h,85h,86h,87h,88h,89h,8Ah,8Bh,8Ch,8Dh,8Eh

- **Step 3 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus with the correct bus access priority when it receives a L_Data.req.

Procedure

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read - System priority:

Service	Control Field	Source Address		Destination Address				
mc	00h	00h	00h	FFh	FFh	E1h	00h	00h

Criteria of acceptance

The BDUT transmits the data packet on the KNX bus corresponding to the L_Data.req:

Control Field	Source Address		Destination Address					Check octet
B0h	10h	00h	FFh	FFh	E1h	00h	00h	XXh

The BDUT also sends an according L_Data.con service:

Service	Control Field	Source Address		Destination Address				
mc	B0h	10h	00h	FFh	FFh	E1h	00h	00h

- **Step 4 (direct)**

Purpose

Check whether the BDUT sends a frame L_Data.con with negative confirm when it receives a L_Data.req and when the data packet is not acknowledged on the KNX bus .

Procedure

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read :

Service	Control Field	Source Address		Destination Address				
mc	0Ch	00h	00h	00h	01h	E1h	00h	00h

Criteria of acceptance

The BDUT sends the following L_Data.con (and an according frame on the bus plus repetitions):

Service	Control Field	Source Address		Destination Address				
mc	9Dh	10h	00h	00h	01h	E1h	00h	00h

- **Step 5 (direct)**

Purpose

Check whether the BDUT sends a frame L_Data.con with negative confirm when it receives a L_Data.req and when the data packet is acknowledged only by "BUSY" on the KNX bus .

Procedure

Configure a bus device, so that it generates "BUSY" signals for all received data packets on the KNX bus.

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read :

Service	Control Field	Source Address		Destination Address				
mc	0Ch	00h	00h	FFh	FFh	E1h	00h	00h

Criteria of acceptance

The BDUT sends the following L_Data.con (and an according frame on the bus plus repetitions):

Service	Control Field	Source Address		Destination Address				
mc	9Dh	10h	00h	FFh	FFh	E1h	00h	00h

- **Step 6 (indirect)**

Purpose

Check whether the BDUT transmits a L_Data.ind when it receives a data packet from the KNX bus.

Procedure

The BDUT receives a data packet which is acknowledged by an IACK :

Control Field	Source Address		Destination Address					Check octet
BCh	AFh	FEh	FFh	FEh	E1h	00h	80h	XXh

Criteria of acceptance

The BDUT sends the correct L_Data.ind:

Service	Control Field	Source Address		Destination Address				
mc	BCh	AFh	FEh	FFh	FEh	E1h	00h	80h

- **Step 7 (indirect)**

Purpose

Check whether the BDUT sends a L_Data.ind with a correct Repeat flag and bus access priority when it receives a data packet on the KNX bus .

Procedure

The BDUT receives a repeated data packet with system priority (which is acknowledged by an IACK):

Control Field	Source Address		Destination Address					Check octet
90h	AFh	FEh	FFh	FEh	E1h	00h	80h	XXh

Criteria of acceptance

The BDUT sends the correct L_Data.ind:

Service	Control Field	Source Address		Destination Address				
mc	B0h	AFh	FEh	FFh	FEh	E1h	00h	80h

- **Step 8 (indirect)**

Purpose

Check whether the BDUT sends a correct L_Data.ind when it receives a data packet with the maximum value length.

Procedure

The BDUT receives a data packet with the maximum value length (which is acknowledged by an IACK):

Control Field	Source Address		Destination Address				Value	Check octet
BCh	AFh	FEh	FFh	FEh	EFh	00h	81h,82h,83h,84h,85h,86h,87h,88h,89h,8Ah,8Bh,8Ch,8Dh,8Eh	XXh

Criteria of acceptance

The BDUT sends the correct L_Data.ind:

Service	Control Field	Source Address		Destination Address				Value
mc	BCh	AFh	FEh	FFh	FEh	EFh	00h	80h	81h,82h,83h,84h,85h,86h,87h,88h,89h,8Ah,8Bh,8Ch,8Dh,8Eh

- **Step 9 (indirect)**

Purpose

Check whether the BDUT does not send a L_Data.ind when it receives a data packet which is not acknowledged on the KNX bus (e.g. because the destination address is not in the address table).

Procedure

The following data packet is sent on the KNX bus :

Control Field	Source Address		Destination Address					Check octet
BCh	AFh	FEh	00h	01h	E1h	00h	80h	XXh

Criteria of acceptance

The BDUT does not transmit a L_Data.ind.

3.2 L_PollData.req/.con Services

3.2.1 Test-Setup

- 1. Select the Link Layer as working layer for the BDUT.
- 2. Load the BDUT with the following address table: 01h,1000h
- 4. install a device in the test set-up with polling group FFFEh and answer slot 02h, Data=01h.

3.2.2 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a L_PollData.req. Check whether the BDUT also transmits a L_PollData.con message with the correct data.

Procedure

The BDUT receives a L_PollData.req, number of slots requested = 05h:

Service	Control Field	Source Address		Polling Group		
mc	F0h	00h	00h	FFh	FEh	05h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Check octet	Polling Slots				
F0h	10h	00h	FFh	FEh	05h	XXh	FEh	FEh	01h	FEh	FEh

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following L_PollData.con:

Service		Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots				
mc		F0h	10h	00h	FFh	FEh	05h	FEh	FEh	01h	FEh	FEh

- **Step 2 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a L_PollData.req with maximum number of polling slots. Check whether the BDUT also transmits a L_PollData.con message with the correct data.

Procedure

The BDUT receives a L_PollData.req, number of slots requested = 0Fh:

Service	Control Field	Source Address		Polling Group		
mc	F0h	00h	00h	FFh	FEh	0Fh

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Check octet	Polling Slots
F0h	10h	00h	FFh	FEh	0Fh	XXh	FE FE 01 FE FE FE FE FE FE FE FE FE FE FE FE

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following L_PollData.con:

Service		Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots
mc		F0h	10h	00h	FFh	FEh	0Fh	FE FE 01 FE FE FE FE FE FE FE FE FE FE FE FE

4 Testing of Network Layer Services

4.1 N_Data_Individual.req/.con/.ind – Services

4.1.1 Test Setup

1. Select the Network-Layer as working layer for the BDUT.
2. Set the routing counter of the BDUT to the value 6.
3. Load the BDUT with the following address table: 01h, 1000h.

4.1.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with the correct routing counter and the address type “individual” when it receives an N_Data_Individual.req. Check also that the BDUT sends a N_Data_Individual.con.

Procedure

Send local N_Data_Individual.req services with routing counter = 6, 5, 4, 3, 2, 1, 0 to the BDUT, e.g. APCI=A_ADC_Read:

Service	Control Field	Source Address		Destination Address					
mc	0Ch	00h	00h	AFh	FEh	02h	41h	80h	00h

...

Criteria of acceptance

The BDUT sends corresponding data packets on the KNX bus with routing counter 6:

Control Field	Source Address		Destination Address						Check octet
BCh	10h	00h	AFh	FEh	62h	41h	80h	00h	XXh

The "Source Address" shall be the individual address of the BDUT and the destination address type shall be “individual”.

The BDUT also sends N_Data_Individual.con services with routing counter = 6:

Service	Control Field	Source Address		Destination Address					
mc	XXh	10h	00h	AFh	FEh	62h	41h	80h	00h

Control Field : legal values = BCh (if acknowledged) or 9Dh (if not acknowledged)

- **Step 2 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with routing counter = 7 when it receives an N_Data_Individual.req with routing counter = 7.

Procedure

Send a local N_Data_Individual.req service to the BDUT, e.g. APCI=A_ADC_Read, routing counter =7 :

Service	Control Field	Source Address		Destination Address					
mc	0Ch	00h	00h	AFh	FEh	72h	41h	80h	00h

Criteria of acceptance

The BDUT sends a corresponding data packet on the KNX bus:

Control Field	Source Address		Destination Address						Check octet
BCh	10h	00h	AFh	FEh	72h	41h	80h	00h	XXh

The "Source Address" shall be the individual address of the BDUT. The routing counter in the frame shall be 7.

The BDUT also sends an N_Data_Individual.con :

Service	Control Field	Source Address		Destination Address					
mc	XXh	10h	00h	FFh	FFh	72h	41h	80h	00h

Control Field : legal values = BCh (if acknowledged) or 9Dh (if not acknowledged)

- **Step 3 (indirect)**

Purpose

Check whether the BDUT sends an N_Data_Individual.ind when it receives a frame on the bus with address type “individual”.

Procedure

Send frames to the BDUT via the bus with routing counters 7, 6, 5, 4, 3, 2, 1, 0, e.g. APCI=A_ADC_Read:

Control Field	Source Address		Destination Address						Check octet
BCh	AFh	FEh	10h	00h	62h	41h	80h	00h	XXh

Criteria of acceptance

The BDUT sends N_Data_Individual.ind services with the routing counter identical to the routing counter in the received frames:

Service	Control Field	Source Address		Destination Address					
Mc	0Ch	FFh	FFh	10h	00h	62h	41h	80h	00h

The "Source Address" shall be the individual address of the BDUT

4.2 N_Data_Group.req / .con / .ind – Services

4.2.1 Test Setup

1. Select the Network-Layer as working layer for the BDUT.
2. Set the routing counter of the BDUT to the value 6.
3. Load the BDUT with the following address table: 02h, 1000h, FFFEh.

4.2.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with the correct routing counter and the address type “group” when it receives an N_Data_Group.req. Check also that the BDUT sends a N_Data_Group.con.

Procedure

Send N_Data_Group.req req services with routing counters 6, 5, 4, 3, 2, 1, 0, to the BDUT, e.g. APCI=A_GroupValue_Read :

Service	Control Field	not used		Destination Address				
mc	0Ch	00h	00h	FFh	FEh	01h	00h	00h

Criteria of acceptance

The BDUT sends corresponding frames on the KNX bus with routing counter = 6:

Control Field	Source Address		Destination Address					Check octet
BCh	10h	00h	FFh	FEh	E1h	00h	80h	XXh

...

The "Source Address" shall be the individual address of the BDUT, and the destination address type shall be “group”.

The BDUT also sends N_Data_Group.con services with routing counter = 6:

Service	Control Field	not used		Destination Address				
mc	XXh	10h	00h	FFh	FEh	E1h	00h	00h

Control Field : legal values = BCh (if acknowledged) or 9Dh (if not acknowledged)

- **Step 2 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with routing counter = 7 when it receives an N_Data_Group.req with routing counter = 7.

Procedure

Send an N_Data_Group.req service to the BDUT, e.g. APCI=A_GroupValue_Read, routing counter =7 :

Service	Control Field	not used		Destination Address				
mc	0Ch	00h	00h	FFh	FEh	71h	00h	80h

Criteria of acceptance

The BDUT sends a corresponding frame on the KNX:

Control Field	Source Address		Destination Address					Check octet
BCh	10h	00h	FFh	FEh	F1h	00h	80h	XXh

The routing counter in the frame shall be 7.

The BDUT also sends a N_Data_Group.con with routing counter = 7:

Service	Control Field	not used		Destination Address				
mc	XXh	10h	00h	FFh	FEh	F1h	00h	80h

Control Field: legal values = BCh (if acknowledged) or 9Dh (if not acknowledged)

- **Step 3 (indirect)**

Purpose

Check whether the BDUT sends an N_Data_Group.ind when it receives a frame on the bus with address type “group”.

Procedure

Send frames to the BDUT via the KNX bus with routing counters 7, 6, 5, 4, 3, 2, 1, 0, e.g. APCI= A_GroupValue_Read:

Control Field	Source Address		Destination Address					Check octet
BCh	AFh	FEh	FFh	FEh	E1h	00h	00h	XXh

Criteria of acceptance

The BDUT sends N_Data_Group.ind services containing the same routing counter as in the received frames:

Service	Control Field	unused		Destination Address				
mc	BCh	AFh	FEh	FFh	FEh	E1h	00h	00h

The "Source Address" shall be the individual address of the BDUT.

4.3 N_Data_Broadcast.req / .con / .ind – Services

4.3.1 Test Setup

1. Select the Network-Layer as working layer for the BDUT.
2. Set the routing counter of the BDUT to the value 6.
3. Load the BDUT with the following address table: 01h, 1000h.

4.3.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with the correct routing counter and the address type “group” when it receives an N_Data_Broadcast.req. Check also that the BDUT sends a N_Data_Broadcast.con.

Procedure

Send N_Data_Broadcast.req services to the BDUT with routing counters 6, 5, 4, 3, 2, 1, 0, e.g. APCI=A_IndividualAddress_Read:

Service	Control Field	not used		not used				
mc	0Ch	00h	00h	00h	00h	01h	01h	00h

Criteria of acceptance

The BDUT sends corresponding frames on the KNX with routing counter 6:

Control Field	Source Address		Destination Address					Check octet
BCh	10h	00h	00h	00h	E1h	01h	00h	XXh

The "Source Address" shall be the individual address of the BDUT and the destination address type shall be “group” with the destination address =0000h.

The BDUT also sends N_Data_Broadcast.con services with routing counter 6:

Service	Control Field	Source Address		Destination: Broadcast				
Mc	BCh	10h	00h	00h	00h	E1h	01h	00h

Control Field : legal values = BCh (if acknowledged) or 9Dh (if not acknowledged)

- **Step 2 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with routing counter = 7 when it receives an N_Data_Broadcast.req with routing counter = 7.

Procedure

Send an N_Data_Broadcast.req service to the BDUT, e.g. APCI=A_IndividualAddress_Read, routing counter =7 :

Service	Control Field	not used		not used				
mc	0Ch	00h	00h	00h	00h	71h	01h	00h

Criteria of acceptance

The BDUT sends a corresponding frame on the KNX bus with routing counter 7:

Control Field	Source Address		Destination Address					Check octet
BCh	10h	00h	00h	00h	F1h	01h	00h	XXh

The BDUT also sends an N_Data_Broadcast.con with routing counter 7:

Service	Control Field	Source Address		Destination: Broadcast				
mc	BCh	10h	00h	00h	00h	F1h	01h	00h

Control Field : legal values = BCh (if acknowledged) or 9Dh (if not acknowledged)

- **Step 3 (indirect)**

Purpose

Check whether the BDUT sends a N_Data_Broadcast.ind when it receives a frame on the bus with address type “group” and destination address = 0000h.

Procedure

Send broadcast frames on the KNX bus with routing counter 7, 6, 5, 4, 3, 2, 1, 0, e.g. APCI= A_IndividualAddress_Read:

Control Field	Source Address		Destination Address					Check octet
BCh	AFh	FEh	00h	00h	E1h	01h	00h	XXh

Criteria of acceptance

The BDUT sends N_Data_Broadcast.ind services with the same routing counters:

Service	Control Field	Source Address		Destination: Broadcast				
mc	BCh	AFh	FEh	00h	00h	E1h	01h	00h

4.4 N_PollData.req / .con – Services

4.4.1 Test Setup

1. Select the Network-Layer as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h, 1000h.

4.4.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a N_PollData.req. Check whether the BDUT also transmits a N_PollData.con message with the correct data.

Procedure

The BDUT receives a N_PollData.req, number of slots requested = 05h:

Service	Control Field	Source Address		Polling Group		
mc	F0h	00h	00h	FFh	FEh	05h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Check octet	Polling slots				
F0h	10h	00h	FFh	FEh	05h	XXh	FEh	FEh	01h	FEh	FEh

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following N_PollData.con:

Service	Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots				
mc	F0h	10h	00h	FFh	FEh	05h	FEh	FEh	01h	FEh	FEh

- **Step 2 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a N_PollData.req with maximum number of polling slots. Check whether the BDUT also transmits a N_PollData.con message with the correct data.

Procedure

The BDUT receives a N_PollData.req, number of slots requested = 0Fh:

Service	Control Field	Source Address		Polling Group		
mc	F0h	00h	00h	FFh	FEh	0Fh

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Check octet	Polling Slots
F0h	10h	00h	FFh	FEh	0Fh	XXh	FE FE 01 FE FE FE FE FE FE FE FE FE FE FE

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following N_PollData.con:

Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots
F0h	10h	00h	FFh	FEh	0Fh	FE FE 01 FE FE FE FE FE FE FE FE FE FE FE

5 Testing of Transport Layer services

5.1 T_Connect - Services

5.1.1 Test-Setup

1. Select the Transport-Layer as working layer for the BDUT.
2. Set the routing counter of the BDUT to the value 6.
3. Load the BDUT with the following address table: 01h,1000h
4. Install a device on the bus with individual address = AFFEh (sends IACKs).

5.1.2 Test-Steps

- **Step 1 (direct, IACK sent)**

Purpose

Check whether the BDUT sends a frame with TPDU Control Data Open on the KNX when it receives a T_Connect.req. Check also that the BDUT sends a T_Connect.con message when it receives an IACK.

Procedure

The BDUT receives a T_Connect.req.

Service	Unused			Destination Address	
mc	00h	00h	00h	AFh	FEh

Criteria of acceptance

The BDUT sends the TPDU Control Data Open on the KNX:

Control Field	Source Address		Destination Address				Check octet
B0h	10h	00h	AFh	FEh	60h	80h	XXh

The BDUT also sends a T_Connect.con:

Service	Control Field	connection Address		unused	must be 00h
mc	B0h	AFh	FEh	AFh	00h

The last octet must be 00h. The source address is the connection address.

- **Step 2 (direct, no IACK sent)**

Purpose

Check whether the BDUT sends a frame with TPDU Control Data Open on the KNX when it receives a T_Connect.req. Check also that the BDUT sends a T_Disconnect.ind message when it does not receive an IACK.

Procedure

The BDUT receives a T_Connect.req.

Service	Unused			Destination Address	
mc	00h	00h	00h	00h	01h

Criteria of acceptance

The BDUT sends the TPDU Control Data Open on the KNX (and repeats it three times):

Control Field	Source Address		Destination Address				Check octet
B0h	10h	00h	00h	01h	60h	80h	XXh

For the repetitions the Control Octet is 90h.

The BDUT also sends a T_Disconnect.ind:

Service	Control Field	unused		unused	must be 00h
mc	00h	10h	00h	00h	00h

The control field and the low octet of the Source and Destination Address must be 00h.

- **Step 3 (indirect)**

Purpose

Check whether the BDUT sends a T_Connect.ind when it receives a frame with TPDU Control Data Open from the KNX-Bus.

Procedure

The BDUT receives a frame with TPDU Control Data Open:

Control Field	Source Address		Destination Address				Check octet
B0h	AFh	FEh	10h	00h	60h	80h	XXh

Criteria of acceptance

The BDUT sends the correct T_Connect.ind:

Service	Control Field	Source address		unused	must be 00h
mc	B0h	AFh	FEh	10h	00h

5.2 T_Disconnect - Service

5.2.1 Test-Setup

1. Select the Transport-Layer as working layer for the BDUT.
2. Set the routing counter of the BDUT to the value 6.
3. Load the BDUT with the following address table: 01h,1000h
4. Install a device on the bus with individual address = 0001h (sends IACKs)

5.2.2 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a data packet on the KNX bus when it receives a T_Disconnect.req. Check also whether the BDUT sends a T_Disconnect.con.

Procedure

After establishing the transport connection, send a T_Disconnect.req to the BDUT:

Service			Connection number
mc	00h 00h	00h 00h	00h

Criteria of acceptance

The BDUT sends the data packet on the KNX with TPCI=close:

Control Field	Source Address	Destination Address			Check octet
B0h	10h 00h	AFh FEh	60h	81h	XXh

The BDUT also sends a T_Disconnect.con message.

Service	unused	unused	unused
mc	00h 00h 00h	00h 00h	

- **Step 2 (indirect)**

Purpose

Check that the BDUT sends a T_Disconnect.ind when it receives a frame with TPDU Control Data Close when a connection is open.

Procedure

Open a connection from device 0001h and send the following data packet on the KNX bus:

Control Field	Source Address	Destination Address			Check octet
B0h	AFh FEh	10h 00h	60h	81h	XXh

Criteria of acceptance

The BDUT sends the correct T_Disconnect.ind:

Service	unused	must be 00h
mc	00h AFh FEh AFh	00h

- **Step 3 (indirect)**

Purpose

Check that the BDUT shows no reaction when it receives a frame with TPDU Control Data Close with a source address different from the connection address when a connection is open.

Procedure

Open a connection from device 0001h and send the following data packet on the KNX bus:

Control Field	Source Address		Destination Address				Check octet
B0h	00h	02h	10h	00h	60h	81h	XXh

Criteria of acceptance

The BDUT shows no reaction:

Service	unused				
mc	00h	00h	00h	00h	00h

- **Step 4 (indirect)**

Purpose

Check that the BDUT shows no reaction when it receives a frame with TPDU Control Data Close when no connection is open.

Procedure

Send the following data packet on the KNX bus when no connection is open:

Control Field	Source Address		Destination Address				Check octet
B0h	00h	01h	10h	00h	60h	81h	XXh

Criteria of acceptance

The BDUT shows no reaction:

- **Step 5 (direct)**

Purpose

Check that the BDUT sends a T_Disconnect.ind and a frame on the bus with TPCI=close when a connection is open and a timeout occurs:

Procedure

Open a connection and wait for 6 seconds.

Criteria of acceptance

The BDUT sends the correct T_Disconnect.ind:

Service	unused				
mc	00h	00h	00h	00h	00h

The BDUT also sends the corresponding frame on the bus:

Control Field	Source Address		Destination Address				Check octet
B0h	10h	00h	AFh	FEh	60h	81h	XXh

5.3 T_Data_Connected - Service

5.3.1 Test-Setup

1. Select the Transport-Layer as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h,1000h
3. Install a device on the bus with individual address = 0001h (sends IACKs, T_ACKs).

5.3.2 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a frame with TPDU Numbered Data Packet on the KNX-Bus with correct bus access priority, correct sequence number and correct data when it receives a T_Data_Connected.req service. Check also that the BDUT sends a T_Data_Connected.con service after receiving a T_ACK.

Procedure

After establishing the transport connection the BDUT receives T_Data_Connected.req services.

1.

Service	Control Field					Length TSDU	APCI	
mc	00h	00h	00h	00h	00h	01h	03h	00h

2.

Service	Control Field					Length TSDU	APCI/Data				
mc	04h	00h	00h	00h	00h	04h	02h	81h	00h	C0h	80h

3.

Service	Control Field					Length TSDU	APCI/Data								
mc	08h	00h	00h	00h	00h	09h	02h	86h	00h	C0h	80h	81h	82h	83h	84h	85h

4.

Service	Control Field					Length TSDU
mc	0Ch	00h	00h	00h	00h	0Fh

APCI/Data															
02h	8Bh	00h	C0h	80h	81h	82h	83h	84h	85h	86h	87h	88h	89h	8Ah	8Bh

Criteria of acceptance

The BDUT sends the TPDUs Numbered Data Packet on the KNX-Bus in the following sequence:

1.

Control Field	Source Address		Destination Address					Check octet
B0h	10h	00h	00h	01h	61h	43h	00h	XXh

2.

Control Field	Source Address		Destination Address							Check octet
B4h	10h	00h	00h	01h	64h	46h	81h	00h	C0h	80h	XXh

3.

Control Field	Source Address		Destination Address												Check octet
B8h	10h	00h	00h	01h	69h	4Ah	86h	00h	C0h	80h	81h	82h	83h	84h	85h	XXh

4.

Control Field	Source Address		Destination Address		
BCh	10h	00h	00h	01h	6Fh

															Check octet
4Eh	8Bh	00h	C0h	80h	81h	82h	83h	84h	85h	86h	87h	88h	89h	8Ah	8Bh	XXh

The BDUT also sends T_Data_Connected:con services, when T_ACK frames are received on the bus (not after IACK!)

1.

Service	Control Field					Length TSDU	APCI	
mc	B0h	10h	00h	00h	00h	61h	03h	00h

2.

Service	Control Field					Length TSDU	APCI/Data				
mc	B4h	10h	00h	00h	00h	64h	02h	81h	00h	C0h	80h

3.

Service	Control Field					Length TSDU	APCI/Data									
mc	B8h	10h	00h	00h	00h	69h	02h	86h	00h	C0h	80h	81h	82h	83h	84h	85h

4.

Service	Control Field					Length TSDU
mc	BCh	10h	00h	00h	00h	6Fh

APCI/Data															
02h	8Bh	00h	C0h	80h	81h	82h	83h	84h	85h	86h	87h	88h	89h	8Ah	8Bh

- **Step 2 (direct)**

Purpose

Check whether the BDUT sends no frame on the KNX-Bus when it receives a T_Data_Connected.req service when no connection is open. Check also that the BDUT sends a T_Disconnect.ind service or shows no reaction.

Procedure

With no connection open the BDUT receives a T_Data_Connected.req service:

1.

Service	Control Field					Length TSDU	APCI	
mc	00h	00h	00h	00h	00h	01h	03h	00h

Criteria of acceptance

The BDUT sends a T_Disconnect.ind or shows no reaction.

Service	unused				
mc	00h	00h	00h	00h	00h

- **Step 3 (indirect)**

Purpose

Check whether the BDUT sends T_Data_Connected.ind services with correct bus access priority, correct connection and correct data when it receives frames with TPDU Numbered Data Packet.

Procedure

After establishing the transport connection, the following frames with TPDU Numbered Data Packet shall be sent to the BDUT via a certified RS232 :

Control Field	Source Address		Destination Address				Check octet
B0h	00h	01h	10h	00h	61h	43h 00h	XXh

Control Field	Source Address		Destination Address							Check octet
B4h	00h	01h	10h	00h	64h	46h	81h	00h	C0h	80h	XXh

Control Field	Source Address		Destination Address												Check octet
B8h	00h	01h	10h	00h	69h	4Ah	86h	00h	C0h	80h	81h	82h	83h	84h	85h	XXh

Control Field	Source Address		Destination Address		
BCh	00h	01h	10h	00h	6Fh

															Check octet
4Eh	8Ch	00h	C0h	80h	81h	82h	83h	84h	85h	86h	87h	88h	89h	8Ah	8Bh	XXh

Criteria of acceptance

The BDUT sends the correct T_Data_Connected.ind services in the following sequence:

1.

Service	Control Field	Source Address		Unused		Length TSDU
mc	X X X X 0 0 X X	00h	01h	10h	00h	61h

APCI/Data		
X X X X X X 1 1	00h	

2.

Service	Control Field	Source Address		Unused		Length TSDU
mc	X X X X 0 1 X X	00h	01h	10h	00h	64h

APCI/Data				
X X X X X X 1 0	81h	00h	C0h	80h

3.

Service	Control Field	Source Address	Unused	Length TSDU
mc	X X X X 1 0 X X	00h 01h	10h	00h	69h

APCI/Data																			
X	X	X	X	X	X	1	0	86h	00h	C0h	80h	81h	82h	83h	84h	85h			

Service	Control Field	Source Address	Unused	Length TSDU
mc	X X X X 1 0 X X	00h 01h	10h	00h	6Fh

APCI/Data																					
X	X	X	X	X	X	1	0	8Ch	00h	C0h	80h	81h	82h	83h	84h	85h	86h	87h	88h	89h	8Ah	8Bh

5.4 T_Data_Group Services

5.4.1 Test Setup

1. Select the Transport-Layer as working layer for the BDUT.
2. Set the routing counter of the BDUT to the value 6.
3. Load the following address table into the BDUT: 02h,1000h, FFFFh.
4. Install a device on the bus with individual address = 0001h (sends IACKs).

5.4.2 Test Steps

Step 1 (direct)

Purpose

Check whether the BDUT generates the correct frame after receiving a T_Data_Group.req. Check that the BDUT also sends a T_Data_Group.con.

Procedure

Send a T_Data_Group.req to the BDUT (e.g. APCI=A_GroupValue_Read):

Service	Control Field	Unused	Unused	Unused	cr_id
mc	$0C_H$	00_H	00_H	00_H	01

length	APCI		
01_H	00H	00H	

Criteria of acceptance

The BDUT sends a frame on the KNX bus including the Value Read information

Control Field	Source Address	Destination Address				Check octet
BC_H	10_H 00_H	FF_H FF_H	$E1_H$	00H	00H	XXh

The Source Address shall be the individual address of the BDUT.

The Destination Address shall be the Group Address of the specified connection number.

The BDUT also sends a T_Data_Group.con:

Service	Control Field	unused	Unused	cr_id
mc	BC_H	10_H	00_H	00_H	01_H

length	APCI	
$E1_H$	00h	00h

Step 2 (direct)

Purpose

Check whether the BDUT generates a T_Data_Group.con if cr_id is not found in the address table.

Procedure

Send a T_Data_Group.req to the BDUT (e.g. APCI=A_GroupValue_Read):

Service	Control Field	Unused	Unused	Unused	cr_id
mc	$0C_H$	00_H	00_H	00_H	03

length	APCI	
01_H	00H	00H

Criteria of acceptance

The BDUT sends sends a T_Data_Group.con, but no frame on the KNX.

Service	Control Field	unused	Unused	cr_id
mc	BD_H	00_H	00_H	00_H	03_H

length	APCI	
01_H	00h	00h

Step 3 (indirect)

Purpose

Check whether the BDUT generates a T_Data_Group.ind after receiving a Group Addressed frame from the KNX.

Procedure

The Server transmits a frame on the KNX including e.g. APCI=A_GroupValue_Read

Control Field	Source Address		Destination Address					Checkoctet
BC_H	00_H	01_H	FF_H	FF_H	$E1_H$	00H	00H	XXh

Criteria of acceptance

The BDUT sends a T_Data_Group.ind service to the PEI:

Service	Control Field	unused		Unused	Connection number
mc	BC_H	00_H	01_H	00_H	01_H
length	APCI				
$E1_H$	00H	00H			

5.5 T_Data_Broadcast Services

5.5.1 Test Setup

1. Select the Transport Layer as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h, 1000h.

5.5.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus when it receives a T_Data_Broadcast.req. Check also that the BDUT sends a T_Data_Broadcast.con.

Procedure

The BDUT receives a T_Data_Broadcast.req, e.g. APCI=A_IndividualAddress_Read:

Service	Control Field	not used		not used				
mc	0Ch	00h	00h	00h	00h	01h	01h	00h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus corresponding to the T_Data_Broadcast.req:

Control Field	Source Address		Destination Address					Check octet
BCh	10h	00h	00h	00h	E1h	01h	00h	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT also sends an T_Data_Broadcast.con :

Service	Control Field	Source Address		Destination: Broadcast				
mc	BCh	10h	00h	00h	00h	E1h	01h	00h

Control Field: legal values = BCh (if acknowledged) or 9Dh (if not acknowledged)

- **Step 2 (indirect)**

Purpose

Check whether the BDUT sends an T_Data_Broadcast.ind when it receives a frame on the bus with address type “group” and destination address = 0000h.

Procedure

The BDUT receives a frame on the KNX bus, e.g. APCI= A_IndividualAddress_Read:

Control Field	Source Address		Destination Address				Check octet
BCh	00h	01h	00h	00h	E1h	01h	00h	XXh

Criteria of acceptance

The BDUT sends a T_Data_Broadcast.ind service:

Service	Control Field	Source Address		Destination: Broadcast				
mc	BCh	00h	01h	00h	00h	E1h	01h	00h

5.6 T_Data_Individual Services

5.6.1 Test Setup

1. Select the Transport Layer as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h, 1000h.

5.6.2 Test Steps

- **Step 1 (direct)**

Purpose

Check that the BDUT sends a frame on the bus when it receives a T_Data_Individual.req service. Check that the BDUT also sends a T_Data_Individual.con service.

Procedure

The BDUT receives a T_Data_Individual.req, e.g. APCI=A_PropertyValue_Read:

Service	Control Field	Source Address		Destination Address								
mc	0Ch	00h	00h	AFh	FEh	05h	03h	D5h	00h	00h	00h	00h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus:

Control Field	Source Address		Destination Address								
BCh	10h	00h	AFh	FEh	65h	03h	D5h	00h	00h	00h	00h

The BDUT also sends a T_Data_Individual.con:

Service	Control Field	Source Address		Destination Address								
mc	BCh	10h	00h	AFh	FEh	65h	03h	D5h	00h	00h	00h	00h

- **Step 2 (indirect)**

Purpose

Check that the BDUT sends a T_Data_Individual.ind service when it receives an individual addressed frame with TPCI=unnumbered data.

Procedure

The BDUT receives a frame, e.g. APCI=A_PropertyValue_Read:

Control Field	Source Address		Destination Address								
BCh	AFh	FEh	10h	00h	65h	03h	D5h	00h	00h	00h	00h

Criteria of acceptance

The BDUT sends a T_Data_Individual.ind service:

Service	Control Field	Source Address		Destination Address								
mc	BCh	AFh	FEh	10h	00h	65h	03h	D5h	00h	00h	00h	00h

5.7 T_PollData Services

5.7.1 Test Setup

1. Select the Transport Layer as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h, 1000h.

5.7.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a T_PollData.req. Check whether the BDUT also transmits a T_PollData.con message with the correct data.

Procedure

The BDUT receives a T_PollData.req, number of slots requested = 05h:

Service	Control Field	Source Address		Polling Group		
mc	F0h	00h	00h	FFh	FEh	05h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots					Check octet
F0h	10h	00h	FFh	FEh	05h	FEh	FEh	01h	FEh	FEh	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following T_PollData.con:

Service	Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots				
mc	F0h	10h	00h	FFh	FEh	05h	FEh	FEh	01h	FEh	FEh

6 Testing of Application Layer Services

6.1 M_Connect.ind[2] Services

6.1.1 Test-Setup

1. Select the Management as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h, 1000h

- **Step 1(indirect)**

Purpose

Check whether the BDUT sends a M_Connect.ind when it receives a frame with TPDU Control Data Open from the KNX-Bus.

Procedure

The BDUT receives a frame with TPDU Control Data Open:

Control Field	Source Address		Destination Address				Check octet
B0h	AFh	FEh	10h	00h	60h	80h	XXh

Criteria of acceptance

The BDUT sends the correct M_Connect.ind:

Service	unused	source address		unused	
mc	B0h	AFh	FEh	10h	00h

6.2 M_Disconnect.ind[3] Services

6.2.1 Test-Setup

1. Select the Management as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h,1000h

[2] So far (end of 2001), M_Connect.req and M_Connect.con are not supported in devices

[3] So far (end of 2001), M_Disconnect.req and M_Disconnect.con are not supported in devices

- **Step 1 (indirect)**

Purpose

Check that the BDUT sends a M_Disconnect.ind when it receives a frame with TPDU Control Data Close:

Procedure

The following data packet is sent on the KNX bus :

Control Field	Source Address		Destination Address				Check octet
B0h	AFh	FEh	10h	00h	60h	81h	XXh

Criteria of acceptance

The BDUT sends the correct M_Disconnect.ind:

Service	unused				
44h	00h	AFh	FEh	AFh	00h

6.3 M_User_Data_Connected Services

The following tests constitute only a part of the possible tests, which could be applicable to test the A_USER_DATA-Services. These tests have been selected on the basis of the following criteria :

- test correct conversion of messages from bus to PEI and vice versa
- check whether variable message length is correctly handled

6.3.1 Test-Setup

- the read or written memory area of the BDUT is not read or write protected.
- the connection is set up
  1. Select the Management as working layer for the BDUT.
  2. Load the BDUT with the following address table: 01h,1000h
  3. Install a device on the bus with individual address = 0001h (sends IACKs).
  4. Ensure that the read or written memory area of the BDUT is not read or write protected.
  5. Set up the connection.

6.3.2 Test-Steps

- **Step 1 (indirect)**

Purpose

Check whether the BDUT generates the correct M_User_Data_Connected.ind if it receives a frame with e.g. APCI=A_UserMemory_Read via the bus.

Procedure

Generate a frame which contains the following UsrDataRead-PDU:

APCI			UserMsg		UsrDataRead						Length (of data in Octets)	Address	
1	0		1	1	0	0	0	0	0	0	1	High Octet	Low Octet

Criteria of acceptance

The BDUT generates a M_User_Data_Connected.ind as follows:

Service		Source address	unused	communication number	information length
mc	x x x x 1 1 x x	AFh FEh	00h	00h	x x x x 0 1 0 0

APCI			Length (of data in Octets)		
UserMsg		UsrDataRead		Address	
x x x x x x 1 0	1 1	0 0 0 0 0 0	1	High Octet	Low Octet

The contents of length, address and data shall comply with the corresponding fields in the transmitted frame.

- **Step 2 (direct)**

Purpose

Check whether the BDUT is able to correctly handle a M_User_Data_Connected.req with e.g. APCI=A_UserMemoryResponse. Check whether the BDUT also sends a M_User_Data_Connected.con.

Procedure

Send a M_User_Data_Connected.req to the BDUT with the following contents:

Service		unused	communication number	information length
mc	0 0 0 0 1 1 0 0	00h 00h 00h	00h	0 0 0 0 0 1 0 1

APCI			Length (of data in Octets)			
UserMsg		UsrDataResponse		Address		Data
0 0 0 0 0 0 1 0	1 1	0 0 0 0 0 1	1	High Octet	Low Octet	1 Octet

Criteria of acceptance

The BDUT shall generate a frame containing the following APCI:

APCI			Length (of data in Octets)			
UserMsg		UsrDataResponse		Address		Data
1 0	1 1	0 0 0 0 0 1	1	High Octet	Low Octet	1 Octet

The contents of length, address and data shall comply with the corresponding fields in the A_USER_DATA.req.

The BDUT also sends a M_User_Data_Connected.con message:

Service		unused	communication number	information length
mc	0 0 0 0 1 1 0 0	10h 00h 00h	00h	0 0 0 0 0 1 0 1

APCI			Length (of data in Octets)			
UserMsg		UsrDataResponse		Address		Data
0 0 0 0 0 0 1 0	1 1	0 0 0 0 0 1	1	High Octet	Low Octet	1 Octet

- **Step 3 (indirect)**

Purpose

Check whether the BDUT generates the correct M_User_Data_Connected.ind after it received a frame with APCI=A_UserMemory_Write for 11 octets via the bus.

Procedure

Send the following frame to the BDUT:

APCI										Length (of data in Octets)	Address		Data
UserMsg		UsrDataWrite											
1	0	1	1	0	0	0	0	1	0	0Bh	High Octet	Low Octet	11 Octets

Criteria of acceptance

The BDUT generates a M_User_Data_Connected.ind as follows:

Service										Source address		unused	communication number								information length
49h		x	x	x	x	c	c	x	x	High	Low	00h	00h		x	x	x	x	1	1	1	1

APCI										Length (of data in Octets)	Address		Data
UserMsg								UsrDataWrite					
x	x	x	x	x	x	1	0	1	1	0	0	0	0	1	0	0Bh	High Octet	Low Octet	11 Octets

The contents of length, address and data shall comply with the corresponding fields in the transmitted frame.

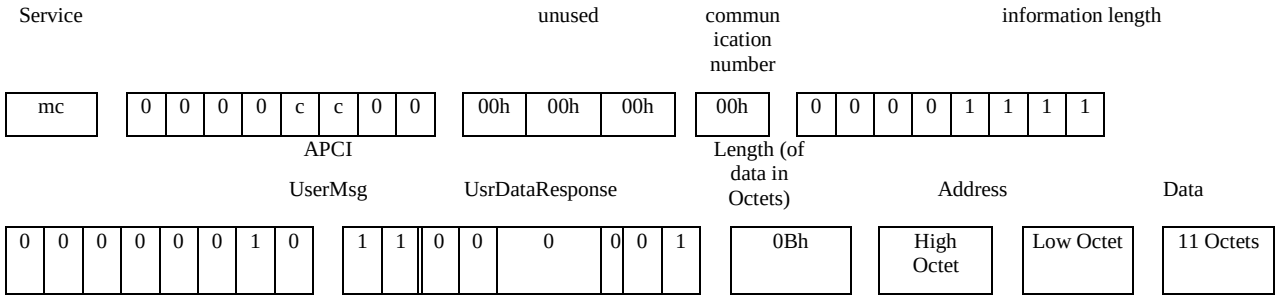
- **Step 4 (direct)**

Purpose

Check whether the BDUT is able to handle correctly a M_User_Data_Connected.req with APCI=A_UserMemory_Write with a data length of 11 octets. Check also that the BDUT sends a M_User_Data_Connected.con.

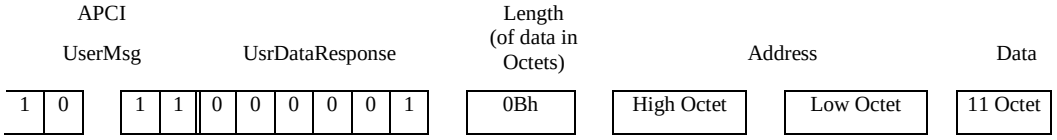
Procedure

Send a M_User_Data_Connected.req to the BDUT with the folowing contents :



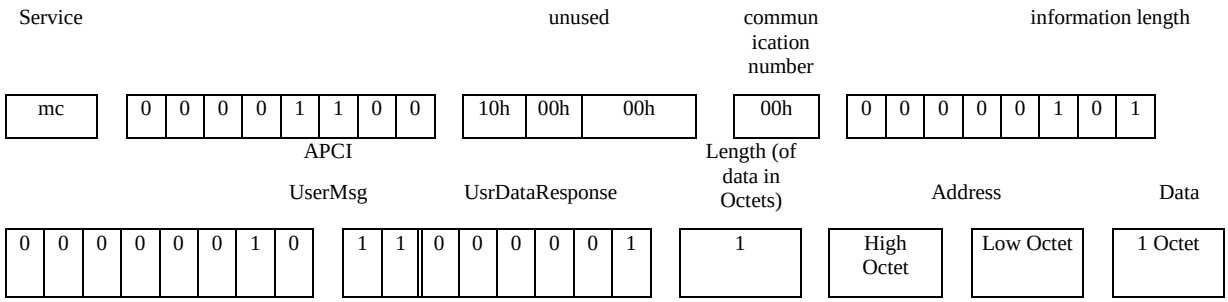
Criteria of acceptance

The BDUT shall generate a frame containing the following UsrDataResponse-PDU:



The contents of length, address and data shall comply with the corresponding fields in the M_User_Data_Connected.request.

The BDUT also sends a M_User_Data_Connected.con message:



6.4 A_Data_Group Services

6.4.1 Test Setup

1. Select the Application Layer as working layer for the BDUT.
2. Load the following address table into the BDUT: 02h,1000h, FFFFh
3. Load the following association table into the BDUT: 01h, 01h, 00h.
4. Install a device on the bus with individual address = AFFEh (sends IACKs).

6.4.2 Test Steps

Step 1 (direct)

Purpose

Check whether the BDUT generates the correct frame after receiving an A_Data_Group.req.Check that the BDUT also sends an A_Data_Group.con.

Procedure

Send an A_Data_Group.req to the BDUT (e.g. APCI=A_GroupValue_Read:

Service	Control Field	Unused	Unused	Unused	SAP-Nr.
mc	$0C_H$	00_H	00_H	00_H	00

length	APCI	
01_H	00H	00H

Criteria of acceptance

The BDUT sends a frame on the KNX bus including the Value Read information

Control Field	Source Address		Desination Address					Checkoctet
BC_H	10_H	00_H	FF_H	FF_H	$E1_H$	00H	00H	XXh

The Source Address shall be the individual address of the BDUT.

The Destination Address shall be the Group Address of the specified connection number.

The BDUT also sends a A_Data_Group.con:

Service	Control Field	Unused	Unused	Unused	SAP-Nr.
mc	BC_H	00_H	00_H	00_H	00_H

length	APCI	
$X1_H$	00H	00H

Step 2 (indirect)

Purpose

Check whether the BDUT generates a A_Data_Group.ind after receiving a Group Addressed frame from the KNX bus.

Procedure

The Server transmits a frame on the KNX bus including e.g. APCI=A_GroupValue_Read

Control Field	Source Address		Desination Address					Checkoctet
BC_H	AF_H	FE_H	FF_H	FF_H	$E1_H$	00H	00H	XXh

Criteria of acceptance

The BDUT sends an A_Data_Group.ind service:

Service	Control Field		Unused	SAP-Nr.
mc	BC_H	00_H 01_H	00_H	00_H

length	APCI		
$E1_H$	00H	00H	

6.5 M_User_Data_Individual Services

6.5.1 Test Setup

1. Select the Management as working layer for the BDUT.
2. Load the BDUT with the following address table: 01h, 1000h.

6.5.2 Test Steps

- **Step 1 (direct)**

Purpose

Check that the BDUT sends a frame on the bus when it receives a M_User_Data_Individual.req service. Check that the BDUT also sends a M_User_Data_Individual.con service.

Procedure

The BDUT receives a M_User_Data_Individual.req, e.g. APCI=A_PropertyValue_Read:

Service	Control Field	unused		Destination Address								
mc	0Ch	00h	00h	00h	01h	05h	03h	D5h	00h	00h	00h	00h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus:

Control Field	Source Address		Destination Address								
BCh	10h	00h	00h	01h	65h	03h	D5h	00h	00h	00h	00h

The BDUT also sends a M_User_Data_Individual.con:

Service	Control Field	unused		Destination Address								
mc	BCh	10h	00h	00h	01h	65h	03h	D5h	00h	00h	00h	00h

- **Step 2 (indirect)**

Purpose

Check that the BDUT sends a M_User_Data_Individual.ind service when it receives a individual addressed frame with TPCI=unnumbered data.

Procedure

The BDUT receives a frame, e.g. APCI=A_PropertyValue_Read:

Control Field	Source Address		Destination Address								
BCh	00h	01h	10h	00h	65h	03h	D5h	00h	00h	00h	00h

Criteria of acceptance

The BDUT sends a M_User_Data_Individual.ind:

Service	Control Field	Source Address		Destination Address								
mc	0Ch	00h	01h	10h	00h	65h	03h	D5h	00h	00h	00h	00h

6.6 A_PollData Services

6.6.1 Test Setup

1. Select the Application Layer as working layer for the BDUT.

2. Load the BDUT with the following address table: 01h, 1000h.

6.6.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives an A_PollData.req. Check whether the BDUT also transmits an A_PollData.con message with the correct data.

Procedure

The BDUT receives an A_PollData.req, number of slots requested = 05h:

Service	Control Field	Source Address		Polling Group		
mc	F0h	00h	00h	FFh	FEh	05h

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots					Check octet
F0h	10h	00h	FFh	FEh	05h	FEh	FEh	01h	FEh	FEh	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following A_PollData.con:

Service	Control Field	Source Address		Polling Group		Nr. of Slots	Polling Slots				
mc	F0h	10h	00h	FFh	FEh	05h	FEh	FEh	01h	FEh	FEh

6.7 M_InterfaceObj_Data Services

Note that only the M_InterfaceObj_Data.req service is usually implemented in the system firmware, .ind and .con services must be implemented in a user-specific callback function.

- **Step 1 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the EHBES bus when it receives an A_Obj_Data.req service with the connection type flag set to connection less communication.

Procedure

The BDUT receives an M_InterfaceObj_Data.req service (e.g. APCI=A_PropertyValue_Response):

Service	Control Field	Destination Address		communication type								
mc	0Ch	AFh	FEh	01h	00h	05h	03h	D6h	00h	00h	00h	00h

Criteria of acceptance

The BDUT sends a frame on the EHBES bus.

Control Field	Source Address		Destination Address									Check octet
BCh	10h	00h	AFh	FEh	65h	03h	D6	00h	00h	00h	00h	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT also sends an M_InterfaceObj_Data.con service to the user:

Service	Control Field	Destination Address		communication type								
mc	0Ch	AFh	FEh	01h	00h	05h	03h	D6h	00h	00h	00h	00h

- **Step 2 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the EHBES bus when it receives an M_InterfaceObj_Data.req service with the connection type flag set to connection oriented communication.

Procedure

Establish a transport layer connection from another bus device to the BDUT. The BDUT receives an M_InterfaceObj_Data.req service (e.g. APCI=A_PropertyValue_Response):

Service	Control Field	unused		communication type								
Mc	0Ch	00h	00h	00h	00h	05h	03h	D6h	00h	00h	00h	00h

Criteria of acceptance

The BDUT sends a frame on the EHBES bus.

Control Field	Source Address		Destination Address									Check octet
BCh	10h	00h	AFh	FEh	65h	43h	D6	00h	00h	00h	00h	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT also sends an M_InterfaceObj_Data.con service to the user:

Service	Control Field	unused		communication type								
Mc	0Ch	00h	00h	00h	00h	05h	03h	D6h	00h	00h	00h	00h

- **Step 3 (indirect)**

Purpose

Check whether the BDUT sends a M_InterfaceObj_Data.ind service to the external user if it receives a data packet on the EHBES bus destined for an externally located interface object sent in connection less mode.

Procedure

The BDUT receives a data frame on the bus. (e.g. APCI=A_PropertyValue_Read) with the connection type flag set to connection less communication:

Control Field	Source Address		Destination Address								Check octet
BCh	AFh	FEh	10h	00h	65h	03h	D6	00h	00h	00h	00h	XXh

Criteria of acceptance

The BDUT sends the following EMI message at the PEI:

Service	Control Field	Destination Address		communication type								
mc	0Ch	AFh	FEh	01h	00h	05h	03h	D6h	00h	00h	00h	00h

- **Step 4 (indirect)**

Purpose

Check whether the BDUT sends a M_InterfaceObj_Data.ind service to the external user if it receives a data packet on the EHBES bus destined for an externally located interface object sent in connection oriented mode.

Procedure

The BDUT receives a data frame on the bus (e.g. APCI=A_PropertyValue_Read)) with the connection type flag set to connection less communication:

Control Field	Source Address		Destination Address								Check octet
BCh	AFh	FEh	10h	00h	65h	43h	D6	00h	00h	00h	00h	XXh

Criteria of acceptance

The BDUT sends the following EMI message at the PEI:

Service	Control Field	Destination Address		communication type								
mc	0Ch	AFh	FEh	00h	00h	05h	03h	D6h	00h	00h	00h	00h

7 Testing of System Broadcast services

7.1 L_Data_SystemBroadcast.req/con/ind Services

7.1.1 Test-Setup

1. Select the Link Layer as working layer for the BDUT.

2. Set the Individual Address of the BDUT to 1101h.

7.1.2 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives a L_Data_SystemBroadcast.req. Check whether the BDUT also transmits a frame L_Data_SystemBroadcast.con.

Procedure

The BDUT receives a L_Data_SystemBroadcast.req, e.g. destination addess = broadcast - APCI=A_DomainAddress_Read :

Service	Control Field	Source Address		Destination Address				
Mc	0Ch	00h	00h	00h	00h	E1h	03h	E1h

Criteria of acceptance

The BDUT sends the data packet (and its repetitions) on the KNX bus corresponding to the L_Data_SystemBroadcast.req with the domain address = 0000h.

Control Field	Source Address		Destination Address					Check octet
BCh	11h	01h	00h	00h	E1h	03h	E1h	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following L_Data_SystemBroadcast.con.

Service	Control Field	Source Address		Destination Address				
Mc	9Ch	11h	01h	00h	00h	E1h	03h	E1h

- **Step 2 (direct)**

Purpose

Check whether the BDUT transmits a data packet (and its repetitions) on the KNX bus with the correct bus access priority when it receives a L_Data_SystemBroadcast.req.

Procedure

The BDUT receives a L_Data_SystemBroadcast.req, e.g. destination address = broadcast ,APCI=A_DomainAddress_Read, System priority :

Service	Control Field	Source Address		Destination Address				
mc	00h	00h	00h	00h	00h	E1h	03h	E1h

Criteria of acceptance

The BDUT transmits the data packet (and its repetitions) on the KNX bus corresponding to the L_Data_SystemBroadcast.req:

Control Field	Source Address		Destination Address					Check octet
B0h	11h	01h	00h	00h	E1h	03h	E1h	XXh

The BDUT also sends an according L_Data_SystemBroadcast.con service:

Service	Control Field	Source Address		Destination Address				
mc	90h	11h	01h	00h	00h	E1h	03h	E1h

- **Step 3 (indirect)**

Purpose

Check whether the BDUT transmits a L_Data_SystemBroadcast.ind when it receives a data packet from the KNX bus.

Procedure

The BDUT receives a data packet, which is acknowledged by an IACK (only applicable to PL devices with IACK capability):

Control Field	Source Address		Destination Address					Check octet
BCh	11h	01h	00h	00h	E1h	03h	E1h	XXh

Criteria of acceptance

The BDUT sends the correct L_Data_SystemBroadcast.ind:

Service	Control Field	Source Address		Destination Address				
mc	BCh	11h	01h	00h	00h	E1h	03h	E1h

- **Step 4 (indirect)**

Purpose

Check whether the BDUT sends a L_Data_SystemBroadcast.ind with a correct Repeat flag and bus access priority when it receives a data packet with the domain address = 0000h on the KNX bus.

Procedure

The BDUT receives a repeated data packet with system priority (which is acknowledged by an IACK - only applicable to PL devices with IACK capability):

Control Field	Source Address		Destination Address					Check octet
90h	7Fh	FEh	00h	00h	E1h	03h	E1h	XXh

Criteria of acceptance

The BDUT sends the correct L_Data_SystemBroadcast.ind:

Service	Control Field	Source Address		Destination Address				
Mc	90h	7Fh	FEh	00h	00h	E1h	03h	E1h

7.2 N_Data_SystemBroadcast.req / .con / .ind – Services

7.2.1 Test Setup

4. Select the Network-Layer as working layer for the BDUT.
5. Set the routing counter of the BDUT to the value 6.
6. Load the BDUT with the following address table: 01h, 1101h.

7.2.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with the correct routing counter and the address type “group” when it receives an N_Data_SystemBroadcast.req. Check also that the BDUT sends a N_Data_SystemBroadcast.con.

Procedure

Send N_Data_SystemBroadcast.req services to the BDUT with routing counters 6, 5, 4, 3, 2, 1, 0, e.g. APCI=A_IndividualAddress_Read:

Service	Control Field	not used		not used				
mc	0Ch	00h	00h	00h	00h	01h	03h	E1h

Criteria of acceptance

The BDUT sends corresponding frames on the KNX with routing counter 6:

Control Field	Source Address		Destination Address					Check octet
BCh	11h	01h	00h	00h	E1h	03h	E1h	XXh

The "Source Address" shall be the individual address of the BDUT and the destination address type shall be “group” with the destination address =0000h.

The BDUT also sends N_Data_SystemBroadcast.con services with routing counter 6:

Service	Control Field	Source Address		Destination: Broadcast				
mc	9Ch	11h	01h	00h	00h	E1h	03h	E1h

- **Step 2 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus with routing counter = 7 when it receives an N_Data_SystemBroadcast.req with routing counter = 7.

Procedure

Send an N_Data_SystemBroadcast.req service to the BDUT, e.g. APCI=A_DomainAddress_Read, routing counter =7:

Service	Control Field	not used		not used				
mc	0Ch	00h	00h	00h	00h	71h	03h	E1h

Criteria of acceptance

The BDUT sends the corresponding frames on the KNX bus with routing counter 7:

Control Field	Source Address		Destination Address					Check octet
BCh	11h	01h	00h	00h	F1h	03h	E1h	XXh

The BDUT also sends an N_Data_SystemBroadcast.con with routing counter 7:

Service	Control Field	Source Address		Destination: Broadcast				
mc	9Ch	11h	01h	00h	00h	F1h	03h	E1h

- **Step 3 (indirect)**

Purpose

Check whether the BDUT sends a N_Data_SystemBroadcast.ind when it receives a frame on the bus with address type “group” and destination address = 0000h and the domain address = 0000h.

Procedure

Send system-broadcast frames on the KNX bus with routing counter 7, 6, 5, 4, 3, 2, 1, 0, e.g. APCI= A_DomainAddress_Read:

Control Field	Source Address		Destination Address					Check octet
BCh	7Fh	FEh	00h	00h	E1h	03h	E1h	XXh

Criteria of acceptance

The BDUT sends N_Data_Broadcast.ind services with the same routing counters:

Service	Control Field	Source Address		Destination: Broadcast				
mc	BCh	7Fh	FEh	00h	00h	E1h	01h	00h

7.3 T_Data_SystemBroadcast Services

7.3.1 Test Setup

1. Select the Transport Layer as working layer for the BDUT.

3. Load the BDUT with the following address table: 01h, 1101h.

7.3.2 Test Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a frame on the bus when it receives a T_Data_SystemBroadcast.req. Check also that the BDUT sends a T_Data_SystemBroadcast.con.

Procedure

The BDUT receives a T_Data_SystemBroadcast.req, e.g. APCI=A_DomainAddress_Read:

Service	Control Field	not used		not used				
mc	0Ch	00h	00h	00h	00h	01h	03h	E1h

Criteria of acceptance

The BDUT sends the data packets on the KNX bus corresponding to the T_Data_SystemBroadcast.req:

Control Field	Source Address		Destination Address					Check octet
BCh	11h	01h	00h	00h	E1h	03h	E1h	XXh

The "Source Address" shall be the individual address of the BDUT.

The BDUT also sends an T_Data_SystemBroadcast.con :

Service	Control Field	Source Address		Destination: Broadcast				
mc	9Ch	11h	001h	00h	00h	E1h	03h	E1h

- **Step 2 (indirect)**

Purpose

Check whether the BDUT sends an T_Data_SystemBroadcast.ind when it receives a frame on the bus with address type “group” and destination address = 0000h and the domain address = 0000h.

Procedure

The BDUT receives a frame on the KNX bus, e.g. APCI= A_DomainAddress_Read:

Control Field	Source Address		Destination Address					Check octet
BCh	7Fh	FEh	00h	00h	E1h	03h	E1h	XXh

Criteria of acceptance

The BDUT sends a T_Data_SystemBroadcast.ind service:

Service	Control Field	Source Address		Destination: Broadcast				
mc	BCh	7Fh	FEh	00h	00h	E1h	03h	E1h

8 Testing of User services

8.1 U_ValueRead-Service

8.1.1 Additional Test-Setup

1. Load an application program with a Group Object table as described below:
   Length of Group Object table: 4

Group Object 0:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: UINT1		
Group Object 1:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: UINT7		
Group Object 2:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: UINT8		
Group Object 3:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: MAXDATA		

Ensure that the application program is not running:

For information:

Type	Action to stop application program
BCU1	set bit 2 = 1, bit 4 = 1 in octet $10D_H$
BCU2	write 02h [stop] to run state machine
CU(BIM112)	write 02h [stop] to run state machine

8.1.2 Test-Steps

• Step 1 (direct)

Purpose

Check whether the BDUT responds correctly when reading from a 1-bit Group Object.

Procedure

Set the RAM-flags of Group Object 0, as listed below:

Update = 1, Data Request = 0, Transmission Status = 00

Set the value of Group Object 0 to 1.

A client sends the U_ValueRead.req below

Service	Object number
mc	00_H

Criteria of acceptance

The BDUT responds with a U_ValueRead.con:

Service	Object number	RAM-flags	Value
mc	00_H	0 0 0 0 1 0 0 0	01_H

• Step 2 (direct)

Purpose

Check whether the BDUT responds correctly when reading from a 7-bit Group Object.

Procedure

Set the RAM-flags of Group Object 1, as listed below:

Update = 0, Data Request = 1, Transmission Status = 00

Set the value of Group Object 1 to $7F_H$.

A client sends the U_ValueRead.request below

Service	Object number
mc	01_H

Criteria of acceptance

The BDUT responds with a U_ValueRead.con:

Service	Object number	RAM-flags	Value
mc	01_H	0 0 0 0 0 1 0 0	$7F_H$

- **Step 3 (direct)**

Purpose

Check whether the BDUT responds correctly when reading from an 8-bit Group Object.

Procedure

Set the RAM-flags of Group Object 2, as listed below:

Update = 0, Data Request = 0, Transmission Status = 01

Set the value of Group Object 2 to 80_H.

A client sends a U_ValueRead.req:

Service	Object number
mc	02_H

Criteria of acceptance

The BDUT responds with a U_ValueRead.con:

Service	Object number	RAM-flags	Value
mc	02_H	0 0 0 0 0 0 0 1	80_H

- **Step 4 (direct)**

Purpose

Check whether the BDUT responds correctly when reading from a 14-octet Group Object.

Procedure

Set the RAM-flags of Group Object 3, as listed below:

Update = 0, Data Request = 0, Transmission Status = 00

Set the value of Group Object 3 to 81_H 82_H 83_H 84_H 85_H 86_H 87_H 88_H 89_H $8A_H$ $8B_H$ $8C_H$ $8D_H$ $8E_H$.

A client sends a U_ValueRead.req:

Service	Object number
mc	03_H

Criteria of acceptance

The BDUT responds with a U_ValueRead.con:

Service	Object number	RAM-flags
mc	03_H	0 0 0 0 0 0 0 0

Value
81_H 82_H 83_H 84_H 85_H 86_H 87_H 88_H 89_H $8A_H$ $8B_H$ $8C_H$ $8D_H$ $8E_H$

8.2 U_ValueWrite-Service

8.2.1 Additional Test-Setup

1. Load an application program with a Group Object table as described below:
   Length of Group Object table: 4

Group Object 0:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: UINT1		
Group Object 1:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: UINT7		
Group Object 2:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: UINT8		
Group Object 3:	Flags:	Transmission =	enabled
		Value memory type =	RAM
		Write =	enabled
		Read =	enabled
		Communication =	enabled
		Transmission Priority =	low
	Value type: MAXDATA		

2. Make sure the application program is not running.

For information:

Type	Action to stop application program
BCU1	set bit 2 = 1, bit 4 = 1 in octet $10D_H$
BCU2	write 02h [stop] to run state machine
CU(BIM112)	write 02h [stop] to run state machine

8.2.2 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT evaluates correctly a write operation on a 1-bit Group Object.

Procedure

Set the Group Object 0 in the BDUT, as listed below:

Update-Flag = 0, Data Request-Flag = 0, Transmission Status = 00
Value = 0

A client sends a U_ValueWrite.req:

Service	Object number	Write-Mask/RAM-flags	Value
mc	00_H	1 1 0 0 1 1 1 1	01_H

Criteria of acceptance

In the BDUT the Group Object 0 is set as specified below:

Update-Flag = 1, Data Request-Flag = 0, Transmission Status = 00
Value = 01_H

- **Step 2 (direct)**

Purpose

Check whether the BDUT correctly evaluates a write operation on a 7-bit Group Object.

Procedure

Set the Group Object 1 in the BDUT, as listed below:

Update-Flag = 0, Data Request-Flag = 0, Transmission Status = 00
Value = 0

A client sends a U_ValueWrite.req:

Service	Object number	Write-Mask/RAM-flags	Value
mc	01_H	1 0 1 0 1 1 1 1	$7F_H$

Criteria of acceptance

In the BDUT the Group Object 1 is set as specified below:

Update-Flag = 0, Data Request-Flag = 1, Transmission Status = 00
Value = $7F_H$

- **Step 3 (direct)**

Purpose

Check whether the BDUT evaluates correctly a write operation on a 8-bit Group Object.

Procedure

Set the Group Object 2 in the BDUT, as listed below:

Update-Flag = 0, Data Request-Flag = 0, Transmission Status = 00
Value = 0

A client sends a U_ValueWrite.req:

Service	Object number	Write-Mask/RAM-flags	Value
mc	02_H	1 0 0 1 1 1 0 1	FF_H

Criteria of acceptance

In the BDUT the Group Object 2 is set as specified below:

Update-Flag = 0, Data Request-Flag = 0, Transmission Status = 01
Value = FF_H

- **Step 4 (direct)**

Purpose

Check whether the BDUT evaluates correctly a write operation on a 14-octet Group Object.

Procedure

Set the Group Object 3 in the BDUT, as listed below:

Update-Flag = 0, Data Request-Flag = 0, Transmission Status = 00
Value = $0..0_H$

A client sends a U_ValueWrite.req:

Service	Object number	Write-Mask/RAM-flags
mc	03_H	1 0 0 0 1 1 1 1

Value
81_H 82_H 83_H 84_H 85_H 86_H 87_H 88_H 89_H $8A_H$ $8B_H$ $8C_H$ $8D_H$ $8E_H$

Criteria of acceptance

In the BDUT the Group Object 3 is set as specified below:

Update-Flag = 0, Data Request-Flag = 0, Transmission Status = 00
Value = 81_H 82_H 83_H 84_H 85_H 86_H 87_H 88_H 89_H $8A_H$ $8B_H$ $8C_H$ $8D_H$ $8E_H$

8.3 U_FlagsRead-Service

8.3.1 Additional Test-Setup

1. Load an application program with a Group Object table as described below:
   Length of Group Object table: 2

   Group Object 0: Flags: Transmission = enabled
   Value memory type = RAM
   Write = enabled
   Read = enabled
   Communication = enabled
   Transmission Priority = low
   Value type: UINT1

   Group Object 1: Flags: Transmission = disabled
   Value memory type = EEPROM
   Write = disabled
   Read = disabled
   Communication = disabled
   Transmission Priority = system
   Value type: UINT16

2. Make sure the application program is not running.

For information:

Type	Action to stop application program
BCU1	set bit 2 = 1, bit 4 = 1 in octet $10D_H$
BCU2	write 02h [stop] to run state machine
CU(BIM112)	write 02h [stop] to run state machine

- **Step 1 (direct)**

Purpose

Check whether the BDUT responds correctly when reading from the Group Object 0.

Procedure

Set the RAM-flags of Group Object 0, as listed below:

Update = 1, Data Request = 1, Transmission Status = 00

A client sends an A_FlagsRead.req:

Service	Object number
mc	00_H

Criteria of acceptance

The BDUT responds with the A_FlagsRead.confirm specified below:

Service	Object number	RAM-flags	EEPROM-flags	Value type
mc	00_H	0 0 0 0 1 1 0 0	1 1 0 1 1 1 1 1	00_H

- **Step 2 (direct)**

Purpose

Check whether the BDUT responds correctly when reading from the Group Object 1.

Procedure

Set the RAM-flags of Group Object 1, as listed below:

Update = 0, Data Request = 0, Transmission Status = 01

A client sends an A_FlagsRead.req:

Service	Object number
mc	01_H

Criteria of acceptance

The BDUT responds with an A_FlagsRead.con:

Service	Object number	RAM-flags	EEPROM-flags	Value type
mc	01_H	0 0 0 0 0 0 0 1	1 0 1 0 0 0 0 0	08_H

8.4 U_Event-Service

8.4.1 Test-Setup

1. Load an application program with the following Group Object table into the BDUT:

Length of communication object table is 0

Group Object 0: flags: transmission = enabled

		value memory type =	RAM
		write	enabled
		read	enabled
		communication	enabled
		transmission priority	low
	value type:		UINT1
Group Object 1:	flags:	transmission =	enabled
		value memory type =	RAM
		write	enabled
		read	enabled
		communication	enabled
		transmission priority	low
	value type:		UINT7
Group Object 2:	flags:	transmission =	enabled
		value memory type =	RAM
		write	enabled
		read	enabled
		communication	enabled
		transmission priority	low
	value type:		UINT8
Group Object 3:	flags:	transmission =	enabled
		value memory type =	RAM
		write	enabled
		read	enabled
		communication	enabled
		transmission priority	low
	value type:		MAXDATA

2. Setup the address table as follows

length	6
individual address	as it is
1st Group Address	0001h
2nd Group Address	0002h
3rd Group Address	0003h
4th Group Address	0004h
5th Group Address	0005h

3. Setup the association table as follows

length	7
1st entry	0100h
2nd entry	0200h
3rd entry	0301h

4th entry	0302h
5th entry	0403h
6th entry	0500h

4. Enable U_Event message generation (BCU1 / BCU2: set bit 6 in ConfigDes)

5. Make sure that the application program is not running

8.4.2 Test-Steps

- **Step 1 (indirect)**

Purpose

Check whether the BDUT generates a correct connection number if only one association for an object exists.

Procedure

Send a group frame with Group Address 0004h to the BDUT.

Criteria of acceptance

BDUT generates a U_Event.ind:

Service	Connection Number
mc	4

- **Step 2 (indirect)**

Purpose

Check whether the BDUT generates a correct connection number if a Group Address exists which is connected to two objects.

Procedure

Send a group frame with Group Address 0003h to BDUT.

Criteria of acceptance

The BDUT sends a U_Event.ind:

Service	Connection Number
mc	3

- **Step 3 (indirect)**

Purpose

Check whether the BDUT generates a correct connection numbers if more than one Group Address is connected to one object.

Procedure

Send group messages with the Group Addresses 0001h, 0002h, 0005h. Generate the group messages in the order given by the Group Addresses.

Criteria of acceptance

The following U_Event.ind messages shall be generated by BDUT in the given order.

Service	Connection Number
mc	1

Service	Connection Number
mc	2

Service	Connection Number
mc	5

8.5 U_UserData Services

U_Userdata services are application specific services which may be defined by the application programmer for communication between an external and an internal user (micro-processor). A set of message codes is reserved to indicate messages, which are sent by the PEI to the user. The internal user must send such a message to the PEI directly.

Tests should verify, that such a message from the external user arrives at the internal user and that a message sent by the internal user is sent to the external user by the PEI.

9 Testing of other default EMI services

9.1 PC_GetValue-Service

9.1.1 Additional Test-Setup

1. Disable any application program (e.g. in BCU1/BCU2 by setting the octet $10D_H = 00_H$).
2. From a starting address SA set the data to the following values: 11_H, 22_H, 33_H, 44_H, 55_H, 66_H, 77_H, 88_H, 99_H, AA_H, BB_H, CC_H, DD_H, EE_H, FF_H
   This starting address shall be

Type	Start address
BCU1/BCU2	0120h
CU 0701 (UsrEEPROM)	4100h

9.1.2 Test-Steps

• Step 1 (direct)

Purpose

Check whether the BDUT responds correctly when reading the minimum amount of data, i.e. 1 octet only.

Procedure

A client sends the PC_GetValue.req below

Service	Length	Address	
mc	01_H	SA high	SA low

Criteria of acceptance

The BDUT responds with the PC_GetValue.con specified below:

Service	Length	Address		Data
mc	01_H	SA high	SA low	11_H

• Step 2 (direct)

Purpose

Check whether the BDUT responds correctly when reading the maximum amount of data, i.e. 15 octets.

Procedure

A client sends the PC_GetValue.req below

Service	Length	Address	
mc	$0F_H$	SA high	SA low

Criteria of acceptance

The BDUT responds with the PC_GetValue.con specified below:

Service	Length	Address		Data
mc	$0F_H$	SA high	SA low	11_H 22_H 33_H 44_H 55_H 66_H 77_H 88_H 99_H AA_H BB_H CC_H DD_H EE_H FF_H

9.2 PC_SetValue -Service

9.2.1 Additional Test-Setup

1. Disable any application program (e.g. in BCU1/BCU2 by setting the octet $10D_H = 00_H$).
2. Set the data starting at starting address1 (SA1, UsrRAM) and at starting address2 (SA2, UsrEEPROM) to the value 00_H. For BCU1/BCU2 the starting addresses shall be: SA1=$0CE_H$ and SA2=120_H. For CU 0701 the starting addresses shall be: SA1=0700h and SA2=4100h.

9.2.2 Test-Steps

• Step 1 (direct)

Purpose

Check whether the BDUT correctly writes the minimum amount of data to RAM, i.e. 1 octet.

Procedure

A client sends the PC_SetValue.req below

Service	Length	Address		Data
mc	01_H	SA1 high	SA1 low	12_H

Then the client sends the PC_GetValue.req below

Service	Length	Address	
mc	01_H	SA1 high	SA1 low

Criteria of acceptance

The BDUT responds with the PC_GetValue.con specified below:

Service	Length	Address		Data
mc	01_H	SA1 high	SA1 low	12_H

- **Step 2 (direct)**

Purpose

Check whether the BDUT correctly writes the maximum amount of data to RAM, i.e. 15 octets.

Procedure

A client sends the PC_SetValue.req below

Service	Length	Address		Data
mc	$0F_H$	SA1 high	SA1 low	11_H 22_H 33_H 44_H 55_H 66_H 77_H 88_H 99_H AA_H BB_H CC_H DD_H EE_H FF_H

Then the client sends the PC_GetValue.req below

Service	Length	Address	
mc	$0F_H$	SA1 high	SA1 low

Criteria of acceptance

The BDUT responds with the PC_GetValue.con specified below:

Service	Length	Address		Data
mc	$0F_H$	SA1 high	SA1 low	11_H 22_H 33_H 44_H 55_H 66_H 77_H 88_H 99_H AA_H BB_H CC_H DD_H EE_H FF_H

- **Step 3 (direct)**

Purpose

Check whether the BDUT writes correctly the minimum amount of data to EEPROM, i.e. 1 octet.

Procedure

A client sends the PC_SetValue.req below

Service	Length	Address		Data
mc	01_H	SA2 high	SA2 low	12_H

Then the client sends the PC_GetValue.req below

Service	Length	Address	
mc	01_H	SA2 high	SA2 low

Criteria of acceptance

The BDUT responds with the PC_GetValue.con specified below:

Service	Length	Address		Data
mc	01_H	SA2 high	SA2 low	12_H

- **Step 4 (direct)**

Purpose

Check whether the BDUT writes correctly the maximum amount of data to EEPROM, i.e. 15 octets.

Procedure

A client sends the PC_SetValue.req below

Service	Length	Address		Data
mc	$0F_H$	SA2 high	SA2 low	11_H 22_H 33_H 44_H 55_H 66_H 77_H 88_H 99_H AA_H BB_H CC_H DD_H EE_H FF_H

Then the client sends the PC_GetValue.req below

Service	Length	Address	
mc	$0F_H$	SA2 high	SA2 low

Criteria of acceptance

The BDUT responds with the PC_GetValue.con specified below:

Service	Length	Address		Data
mc	$0F_H$	SA2 high	SA2 low	11_H 22_H 33_H 44_H 55_H 66_H 77_H 88_H 99_H AA_H BB_H CC_H DD_H EE_H FF_H

9.3 PEI_Identify Services

9.3.1 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT sends a PEI_Identify.con to the PEI correctly when it receives a PEI_Identify.req.

Procedure

An external user sends a PEI_Identify.req message:

Service
mc

Criteria of acceptance

The BDUT responds with a PEI_Identify.con message:

Service	Phys. Address		Serial Number					
mc	x	x	x	x	x	x	x	x

The individual address and the serial number of the BDUT are contained in the message.

9.4 PEI_Switch.req Service

9.4.1 Test-Steps

- **Step 1 (direct)**

Purpose

Check whether the BDUT switches to link layer when it receives an according PEI_Switch.req.

Procedure

An external user sends a PEI_Switch.req message

(LL=link layer, NL=network layer, TLG= group oriented transport layer, TLC=connection oriented transport layer, TLL=local transport layer, AL=application layer, MG=management, USR=user).

Service	System Status	LL	NL	TLG	TLC	TLL	AL	MG	PEI	USR	res.
mc	0	1	8	3	4	5	6	7	8	0	A

Check whether the BDUT has switched to link layer by sending a frame on the bus addressed to the BDUT.

Control Field	Source Address		Destination Address					Check octet
BCh	AFh	FEh	FFh	FEh	E1h	00h	80h	XXh

Criteria of acceptance

The BDUT sends the correct L_Data.ind:

Service	Control Field	Source Address		Destination Address				
mc	BCh	AFh	FEh	FFh	FEh	E1h	00h	80h

- **Step 2 (direct)**

Purpose

Check whether the BDUT switches to link layer in Busmonitor mode when it receives an according PEI_Switch.req.

Procedure

An external user sends a PEI_Switch.req message.

Service	System Status	LL	NL	TLG	TLC	TLL	AL	MG	PEI	USR	res.
mc	90h	1	8	3	4	5	6	7	8	0	A

Check whether the BDUT has switched to link layer in busmonitor mode by sending a frame on the bus addressed to the BDUT.

Criteria of acceptance

The BDUT sends the correct L_Busmon.ind:

Service	Trans.-Flags	Time		Data
mc	XX_H	XX_H	XX_H	Frame sent (with checksum)

- **Step 3 (direct)**

Purpose

Check whether the BDUT switches to TLC and TLG when it receives an according PEI_Switch.req.

Procedure

An external user sends a PEI_Switch.req message.

Service	System Status	LL	NL	TLG	TLC	TLL	AL	MG	PEI	USR	res.
mc	0	1	8	3	4	4	8	8	8	0	A

Check whether the BDUT has switched to TLC and TLG by sending group and individual addressed frames on the bus addressed to the BDUT.

Control Field	Source Address		Destination Address				Check octet
B0h	XXh	XXh	10h	00h	60h	80h	XXh

Control Field	Source Address		Destination Address					Check octet
BC_H	00_H	01_H	FF_H	FF_H	$E1_H$	00H	00H	XXh

Criteria of acceptance

The BDUT sends the correct T_Connect.ind and T_Data_Group.ind:

Service	Control Field	Unused			Connection number		
mc	XXh	XXh	XXh	XXh	00h	60h	80h

Service	Control Field		Unused	Connection number
mc	BC_H	00_H 01_H	00_H	01_H

length	APCI	
$E1_H$	00H	00H

- **Step 4 (direct)**

Purpose

Check whether the BDUT switches to AL when it receives an according PEI_Switch.req.

Procedure

An external user sends a PEI_Switch.req message.

Service	System Status	LL	NL	TLG	TLC	TLL	AL	MG	PEI	USR	res.
mc	0	1	8	3	4	5	6	7	8	8	A

Check whether the BDUT has switched to AL by sending a frame on the bus addressed to the BDUT.

Control Field	Source Address		Destination Address					Check octet
BC_H	00_H	01_H	FF_H	FF_H	$E1_H$	00H	00H	XXh

Criteria of acceptance

The BDUT sends an A_Data_Group.ind service:

Service	Control Field		Unused	SAP-Nr.
mc	BC_H	00_H 01_H	00_H	00_H

length	APCI	
$E1_H$	00H	00H

9.5 TM_Timer Service

- **Step 1 (direct)**

Purpose

Check whether the BDUT generates a TM_Timer.ind message when a timer expires.

Procedure

Start a timer the option “message generation” and the addressed module = “PEI”.

Criteria of acceptance

BDUT generates a TM_Timer.ind:

Format style 1 (Control Unit)

Message Format :

Message	timer number
mc	1 Octet

Format style 2 (BCU 2)

Message Format :

Message	timer parameter	Timer number	5 octets
mc	1 octet	1 octet	00 00 00 00 00

10 Testing of cEMI

10.1 Testing of Busmonitor and Raw Services

10.1.1 L_Busmon Service

10.1.1.1 Additional Test-Setup

Set the BDUT in busmonitor-mode.

KNX-USB-HID frame example: 011310000800080103000**0F600080134100101**
Response from BDUT: 01130F0008000701030000**F5000801341001**

10.1.1.2 Test Steps

• Step 1 (indirect)

EMI1/2 equivalent test: chapter 8-6-3, clause 2.2, Test-Step 1

Purpose

Check whether the BDUT correctly reports an unacknowledged frame on the bus.

Procedure

Send a frame on the bus that is not acknowledged.

Criteria of acceptance

The BDUT sends the L_Busmon.ind specified below:

Service	cEMI Additional Information (optional)												cEMI Service Information
	AddIL	Trans.-Flags								Time			Raw Data
mc	XX_H	03_H	1	f	b	p	v	l	s	s	s	04_H	2	XX_H XX_H	Frame sent (with checksum)

Check that the cEMI Additional information is set to 0 when not supported or when supported that

- frame error flag (f) is set, if a frame error is detected in one or several of the frame bits in the message.
- Bit error flag (b) is set, if an invalid bit is detected in one or several of the frame characters.
- Parity error flag (p) is set, if an invalid parity bit is detected in one or several of the frame bits.
- Lost flag (l) is set if at least one frame or frame piece was lost by the Data Link Layer in Busmonitor mode.
- The sequence counter is incremented with every received frame

• Step 2 (indirect)

EMI1/2 equivalent test: chapter 8-6-3, clause 2.2, Test-Step 2

Purpose

Check whether the BDUT correctly reports an acknowledged frame on the bus.

Procedure

Send a frame on the bus that is acknowledged.

Criteria of acceptance

The BDUT sends the two L_Busmon.ind specified below:

1.

Service	cEMI Additional Information (optional)												cEMI Service Information
	AddIL	Trans.-Flags								Time			Raw Data
mc	XX_H	03_H	1	f	b	p	v	l	s	s	s	04_H	2	XX_H XX_H	Frame sent (with checksum)

2.

Service	cEMI Additional Information (optional)												cEMI Service Information
	AddIL	Trans.-Flags								Time			Raw Data
mc	XX_H	03_H	1	f	b	p	v	l	s	s	s	04_H	2	XX_H XX_H	ACK-character

10.1.2 L_Raw Service

Current known implementations do not use this Service.[4]

10.1.2.1 Additional Test-Setup

Set the BDUT back to Link Layer-mode.

KNX-USB-HID frame example: 011310000800080103000**0F600080134100100**
Response from BDUT: 01130F0008000701030000**F5000801341001**

10.1.2.2 Test Steps

- **Step 1 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 2.3, Test-Step 1 (L_PlainData.req)

Purpose

Check whether the BDUT sends a frame on the KNX bus when it receives an L_Raw.req. Check whether the BDUT also transmits a frame L_Raw.con.

Procedure

Send an L_Raw.req message to the BDUT:

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data (Raw Frame to send, with checksum)
mc	XX_H					BC AFFE 0000 60 C2

[4] L-Raw Service is not applicable/possible for a TP-Uart based cEMI server device

Criteria of acceptance

The BDUT sends the corresponding frame on the bus

- when no additional information field is supported, the telegram shall be sent on the bus with the regular interframe time)

when a time delay (4 octet unsigned counter) is given in the supported additional information field (Type ID 05h), the telegram shall be sent when the free running system counter of the sending device is equal to the value given

BC AFFE 0000 60 C2

The BDUT sends the following L_Raw.con, after the frame is sent completely on the bus:

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data
mc	XX_H					BC AFFE 0000 60 C2

• **Step 2 (indirect)**

EMI1/2 equivalent test: chapter 8-6-3, clause 2.3 → not available

Purpose

Check whether the BDUT correctly reports a frame on the bus as a raw frame.

Procedure

Send a frame on the bus.

Criteria of acceptance

The BDUT sends the following L_Raw.ind when it sees a frame on the bus:

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Raw Data
mc	XX_H					Frame sent (with checksum)

10.1.3 Testing of Additional Information in L_Busmon & L_Raw Services

As far as supported by BDUT [5], additional information shall be tested in the following way:

• **Test Step 1 (indirect)**

EMI1/2 equivalent test: none

Purpose

Check whether the BDUT correctly reports frames with the expected additional information. This Test shall be done for all supported additional information types.

Procedure

Set BDUT to the communication mode (Busmonitor Mode or Raw Mode) needed for the test (direct). Then send frames (indirect) on the bus that are reported by BDUT on its cEMI interface.

[5] Please refer to the device profile and the manufacturer's product documentation (PICS/PIXIT) for supported additional info types.

Criteria of acceptance

The BDUT sends cEMI frames (with L_Busmon.ind or L_Raw.ind, according the current communication mode) with the expected additional information.

- **Test Step 2 (direct)**

EMI1/2 equivalent test: none

Purpose

Check whether the BDUT sends frames to the bus as expected according the additional information given in the cEMI frame (L_Raw.req).
This Test shall be done for all supported additional information types.

Procedure

Send L_Raw.req messages to the BDUT with the additional information type to be tested.

Criteria of acceptance

The BDUT sends the frames to the bus as expected, e.g. with time delay.

- **Test Step 3 (direct)**

EMI1/2 equivalent test: none

Purpose

Check whether the BDUT discards unsupported additional information types.

Procedure

Send L_Raw.req messages to the BDUT with unsupported additional information type(s)

Criteria of acceptance

The BDUT ignores the received additional information and sends the frames to the bus independent of the (unsupported) additional information given in the cEMI frame.

10.2 Testing of Link Layer services

10.2.1 L_Data Service

10.2.1.1 Test-Setup

1 Select the Link Layer as working layer for the BDUT.
KNX-USB-HID frame example: 011310000800080103000**0F600080134100100**
Response from BDUT: 01130F0008000701030000**0F5000801341001**

2 Install a device in the test set-up acknowledging only the Group Address FFFFh.

3 Load the BDUT with the following address table: 02h, 1000h, FFFEh.
Remark: when a cEMI server device sends systematic LL-Acks to Group Addressed frames (telegrams, in which the address type bit [AT-bit] is set to 1), then this step is not necessary.

10.2.1.2 Test-Steps

- **Step 1 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 1

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives an L_Data.req. Check whether the BDUT also transmits a frame L_Data.con.

Procedure

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read :

Service	cEMI Additional Information (optional)						cEMI Service Information
	AddIL						Data
mc	XX_H						BC E0 0000 FFFF 01 00 00

KNX-USB-HID frame example: 0113130008000B01030000**1100BCE00000FFFF010000**

Criteria of acceptance

The BDUT sends the data packet on the KNX bus corresponding to the L_Data.req:

Control Field	Source Address		Destination Address					Check octet
BC	xx	yy	FF	FF	E1	00	00	XX

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following L_Data.con when it receives an IACK on the bus[6]:

Service	cEMI Additional Information (optional)						cEMI Service Information
	AddIL						Data
mc	XX_H						BC E0 xxyy FFFF 01 00 00

KNX-USB-HID frame example: 0113130008000B01030000**2E00BCE0xxyyFFFF010000**
whereas xx yy is the BDUT's Individual Address

- **Step 2 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 2

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives an L_Data.req with the maximum value length.

Procedure

The BDUT receives an L_Data.req, e.g. Group address type - APCI=ValueWrite:

Service	cEMI Additional Information (optional)						cEMI Service Information
	AddIL						Data
mc	XX_H						BC E0 0000 FFFF 0F 00 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E

KNX-USB-HID frame example:
0113210008001901030000**1100BCE00000FFFE0F00808182838485868788898A8B8C8D8E**

[6] In the case where the cEMI server does not have an own individual address, it shall send back a negative confirm (also applies to following steps).

Note: this frame example is applicable as test for maximum value length only for devices that support standard short frames only. Test frame must be adapted for extended/long frames.

KNX-USB-HID frame example for a max. length single HID report (cEMI-USB):
01153D0008004101030000**11003CE00000FFFE37008081828384858687888 98A8B8C8D8E8F909192939495969798999A9B9C9D9E9F10111213141516171 8191A**: 1. Report
01260C**1B1C1D1E1F20212223242526**: 2. Report

Criteria of acceptance

The BDUT transmits the data packet on the KNX bus corresponding to the L_Data.req:

Control Field	Source Address		Destination Address				Value	Check octet
BC	xx	yy	FF	FF	EF	00	80	81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E	XX

The "Source Address" shall be the individual address of the BDUT.

The BDUT also sends an according L_Data.con service:

Service	cEMI Additional Information (optional)				cEMI Service Information
	AddIL				Data
mc	XX_H				BC E0 xxyy FFFF 0F 00 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E

- **Step 3 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 3

Purpose

Check whether the BDUT transmits a data packet on the KNX bus with the correct bus access priority when it receives an L_Data.req.

Procedure

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read - System priority:

Service	cEMI Additional Information (optional)				cEMI Service Information
	AddIL				Data
mc	XX_H				B0 E0 0000 FFFF 01 00 00

KNX-USB-HID frame example: 0113130008000B01030000**110090E00000FFFF010000**

Criteria of acceptance

The BDUT transmits the data packet on the KNX bus corresponding to the L_Data.req:

Control Field	Source Address		Destination Address					Check octet
B0	xx	yy	FF	FF	E1	00	00	XX

The BDUT also sends an according L_Data.con service:

Service	cEMI Additional Information (optional)	cEMI Service Information

	AddIL					Data
mc	XX_H					B0 E0 xxyy FFFF 01 00 00

KNX-USB-HID frame example: 0113130008000B010300002**E00B0E0xxyyFFFF010000**
whereas xx yy is the BDUT's Individual Address

- **Step 4 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 4

Purpose

Check whether the BDUT sends a frame L_Data.con with negative confirm when it receives an L_Data.req and when the data packet is not acknowledged on the KNX bus.

Procedure

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read, with Repeat-Flag set to "repetitions":

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data
mc	XX_H					BC E0 0000 1234 01 00 00

KNX-USB-HID frame example: 0113130008000B010300001**100BCE000001234010000**

Criteria of acceptance

The BDUT transmits the data packet on the KNX bus corresponding to the L_Data.req, with repetitions:

Control Field	Source Address		Destination Address					Check octet
BC	xx	yy	12	34	E1	00	00	XX
9C	xx	yy	12	34	E1	00	00	XX
9C	xx	yy	12	34	E1	00	00	XX
9C	xx	yy	12	34	E1	00	00	XX

The BDUT sends the following L_Data.con:

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data
mc	XX_H					BD E0 xxyy 1234 01 00 00

KNX-USB-HID frame example: 0113130008000B010300002**E00BDE0xxyy1234010000**
whereas xx yy is the BDUT's Individual Address

- **Step 5 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2 → not available

Purpose

Check whether the BDUT sends a frame L_Data.con with negative confirm when it receives an L_Data.req and when the data packet is not acknowledged on the KNX bus. Compared to Test Step 4, the frame shall not be repeated on the bus.

Procedure

The BDUT receives a L_Data.req, e.g. Group address type - APCI=A_GroupPropValue_InfoReportValue_Read, with Repeat-Flag set to “no repetitions”:

Service	cEMI Additional Information (optional) AddIL						cEMI Service Information Data
mc	XX_H						14 E4 00 00 04 11 07 07 EB 00 64 01 33 02 00

KNX-USB-HID frame example: 011319000800110103000**0110014E4000004110707EB006401330200**

Criteria of acceptance

The BDUT transmits the data packet on the KNX bus corresponding to the L_Data.req (**without** repetitions):

Control Field 1	Control Field 2	Source Address		Destination Address		Length				Check octet
34	E4	xx	yy	04	11	07	07	EB	00 64 01 33 02 00	XX

The BDUT sends the following L_Data.con:

Service	cEMI Additional Information (optional) AddIL						cEMI Service Information Data
mc	XX_H						**15** E4 xx yy 04 11 07 07 EB 00 64 01 33 02 00

KNX-USB-HID frame example: 011319000800110103000**0110015E4xxxx04110707EB006401330200**
whereas xx yy is the BDUT’s Individual Address

- **Step 6 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 5

Purpose

Check whether the BDUT sends a frame L_Data.con with negative confirm when it receives an L_Data.req and when the data packet is acknowledged only by "BUSY" or “NACK” on the KNX bus.

Procedure

Configure a bus device, so that it generates "BUSY" signals for all received data packets on the KNX bus.

The BDUT receives an L_Data.req, e.g. Group address type - APCI=A_GroupValue_Read, with Repeat-Flag set to “repetitions”:

Service	cEMI Additional Information (optional) AddIL						cEMI Service Information Data
mc	XX_H						BC E0 0000 FFFF 01 00 00

KNX-USB-HID frame example: 0113130008000B01030000**1100BCE00000FFFE010000**

Criteria of acceptance

The BDUT sends the following L_Data.con (and a corresponding frame on the bus plus repetitions):

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data
mc	XX_H					**BD** E0 xxyy FFFF 01 00 00

KNX-USB-HID frame example: 0113130008000B010300002**E00BDE0xxyyFFFF010000**
whereas xx yy is the BDUT's Individual Address

- **Step 7 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2 → not available

Purpose

Check whether the BDUT sends a frame L_Data.con with negative confirm when it receives an L_Data.req and when the data packet is acknowledged only by "BUSY" or "NACK" on the KNX bus. Compared to Test Step 6, the frame shall not be repeated on the bus.

Procedure

Configure a bus device, so that it generates "BUSY" signals for all received data packets on the KNX bus.

The BDUT receives an L_Data.req, e.g. Group address type - APCI=A_GroupPropValue_InfoReport, with Repeat-Flag set to "no repetitions":

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data
mc	XX_H					14 E4 00 00 04 11 07 07 EB 00 64 01 33 02 00

KNX-USB-HID frame example: 011319000800110103000**0110014E4000004110707EB006401330200**

Criteria of acceptance

The BDUT sends the following L_Data.con (and an according frame on the bus, **without** repetitions):

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data
mc	XX_H					**15** E4 xx yy 04 11 07 07 EB 00 64 01 33 02 00

KNX-USB-HID frame example: 011319000800110103000**0110015E4xxxx04110707EB006401330200**
whereas xx yy is the BDUT's Individual Address

- **Step 8 (indirect)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 6

Purpose

Check whether the BDUT transmits an L_Data.ind when it receives a data packet from the KNX bus.

Procedure

The BDUT receives a data packet, which is acknowledged by BDUT with an IACK:

Control Field	Source Address		Destination Address					Check Octet
BC	AF	FE	FF	FE	E1	00	80	XX

Criteria of acceptance

The BDUT sends the correct L_Data.ind:

Service	cEMI Additional Information (optional) AddIL		cEMI Service Information Data
mc	XX_H		BC E0 AFFE FFFE 01 00 80

KNX-USB-HID frame example: 0113130008000B01030000**2900BCE0AFFEFFFE010080**

- **Step 9 (indirect)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 7 [7]

Purpose

Check whether the BDUT sends an L_Data.ind with correct Repeat flag and bus access priority when it receives a data packet on the KNX bus.

Procedure

The BDUT receives a repeated data packet with system priority (which is acknowledged by an IACK):

Control Field	Source Address		Destination Address				Check Octet
90	AF	FE	FF	FE	E1	00	80	XX

Note: The EITT-UI gives the impression that the repetition flag can be set when sending frames onto the bus. But any frames sent to the bus (with EITT V2.3) over a BCU1/2-RS232 interface are sent by the data interface always with R-Flag set to '1' for 'not a repeated frame', i.e. the repetitions are controlled by the interface device (BAU) itself and not by the R-Flag in the Ctrl-Field within the EMI-frame.
Sending of above frame as "the first" telegram is not possible with a TP-UART based interface device either, as repetitions onto the bus are controlled by the TP- UART itself.
→ Use a "terminal-BCU".

Criteria of acceptance

The BDUT sends the correct L_Data.ind:

Service	cEMI Additional Information (optional) AddIL		cEMI Service Information Data
mc	XX_H		90 E0 AFFE FFFE 01 00 80

KNX-USB-HID frame example: 0113130008000B01030000**290090E0AFFEFFFE010080**

- **Step 10 (indirect)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 8

Purpose

Check whether the BDUT sends a correct L_Data.ind when it receives a data packet with the maximum value length.

[7] The test in HB chapter 8-3-6, *clause 3.1.2, Test-Step 7 is not meaningful concerning the R-Flag: the repetition flag always is set with by the BAU (BCU1/2 for an RS232 interface) independent of EITT setting.*

Procedure for a short frame

The BDUT receives a data packet with the maximum value length (which is acknowledged by an IACK):

Control Field	Source Address		Destination Address				Value	Check Octet
BC	AF	FE	FF	FE	EF	00	80	81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E	XX

Note: this frame example is applicable as test for maximum value length only for devices that support standard short frames only. Test frame must be adapted for extended/long frames.

Criteria of acceptance (short frame)

The BDUT sends the correct L_Data.ind:

Service	cEMI Additional Information (optional)				cEMI Service Information
	AddIL				Data
mc	XX_H				BC E0 AFFE FFFE 0F 00 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E

KNX-USB-HID frame example: 011321000800190103000**02900BCE0AFFEFFFE0F0080…8E**

Procedure for a long frame (with max. APDU-Length of 55)

The BDUT receives a data packet with the maximum value length (which is acknowledged by an IACK):

Control Field 1	Control Field 2	Source Address		Destination Address		Length		Value	Check Octet
3C	E0	AF	FE	FF	FE	37	00	**01 02 03 04 05 06 07 00 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 28 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36**	XX

Criteria of acceptance (long frame with a max. APDU-Lenth of 55)

The BDUT sends the correct L_Data.ind:

Service	cEMI Additional Information (optional)				cEMI Service Information
	AddIL				Data
mc	XX_H				BC E0 AFFE FFFE 37 00 01 02 03 04 05 06 07 08 … 33 34 35 36

KNX-USB-HID frame example:

1st USB-HID-Report:
01153D000800410103000**029003CE002FFFFFE3700800102030405060700090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F2021222324252627282 82A**

2nd USB-HID-Report:
01260C**2B2C2D2E2F30313233343536**

- **Step 11 (indirect)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.1.2, Test-Step 9

Purpose

Check whether the BDUT does not send an L_Data.ind when it receives a data packet, which is not acknowledged on the KNX bus (e.g. because the destination address is not in the address table or an individual-addressed frame does not belong to the BDUT's Individual Address).

Procedure

The following data packet is sent on the KNX bus (example for a Group Addressed frame):

Control Field	Source Address		Destination Address						Check Octet
BC	AF	FE	00	01	E1	00	80		XX

Criteria of acceptance

The BDUT does not transmit an L_Data.ind.

10.2.2 L_Poll_Data Service

Current known implementations do not use this Service.

10.2.2.1 Test-Setup

1. Select the Link Layer as working layer for the BDUT.
   KNX-USB-HID frame example: 011310000800080103000**0F600080134100100**
   Response from BDUT: 01130F0008000701030000**F5000801341001**

2. Load the BDUT with the following address table: 01h,1000h

3. Install a device in the test set-up with polling group FFFEh and answer slot 02h, Data=01h.

10.2.2.2 Test-Steps

- **Step 1 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.2.2, Test-Step 1

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives an L_Poll_Data.req. Check whether the BDUT also transmits an L_Poll_Data.con message with the correct data.

Procedure

The BDUT receives a L_Poll_Data.req, number of slots requested = 05h:

Service	cEMI Additional Information (optional) AddIL					cEMI Service Information Data
mc	XX_H					F0 00 0000 FFFE 05

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Check octet	Polling Slots				
F0	xx	yy	FF	FE	05	XX	FE	FE	01	FE	FE

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following L_Poll_Data.con:

Service	cEMI Additional Information (optional) AddIL	cEMI Service Information Data

mc	XX_H					F0 00 xxyy FFFE 05 FE FE 01 FE FE

- **Step 2 (direct)**

EMI1/2 equivalent test: chapter 8-6-3, clause 3.2.2, Test-Step 2

Purpose

Check whether the BDUT transmits a data packet on the KNX bus when it receives an L_Poll_Data.req with maximum number of polling slots. Check whether the BDUT also transmits an L_Poll_Data.con message with the correct data.

Procedure

The BDUT receives an L_Poll_Data.req, number of slots requested = 0Fh:

Service	cEMI Additional Information (optional)					cEMI Service Information
	AddIL					Data
mc	XX_H					F0 00 0000 FFFE 0F

Criteria of acceptance

The BDUT sends the data packet on the KNX bus and fills all unanswered slots with FEh:

Control Field	Source Address		Polling Group		Nr. of Slots	Check octet	Polling Slots
F0	xx	yy	FF	FE	0F	XX	FE FE 01 FE FE FE FE FE FE FE FE FE FE FE FE

The "Source Address" shall be the individual address of the BDUT.

The BDUT sends the following L_Poll_Data.con:

Service	cEMI Additional Information (optional)						cEMI Service Information
	AddIL						Data
mc	XX_H						F0 00 xxyy FFFE 0F FE FE 01 FE FE FE FE FE FE FE FE FE FE FE FE

10.2.3 Testing of Additional Information in Link Layer Services

As far as supported by BDUT [8], additional information shall be tested in the following way:

- **Test Step 1 (indirect)**

EMI1/2 equivalent test: none

Purpose

Check whether the BDUT correctly reports frames with the expected additional information. This Test shall be done for all supported additional information types.

Procedure

Set BDUT to Link Layer communication mode (direct). Then send frames (indirect) on the bus that are reported by BDUT on its cEMI interface.

Criteria of acceptance

The BDUT sends cEMI frames with the expected additional information.

- **Test Step 2 (direct)**

EMI1/2 equivalent test: none

Purpose

Check whether the BDUT sends frames to the bus as expected according the additional information given in the cEMI link layer frame.
This Test shall be done for all supported additional information types.

Procedure

Send Link Layer messages to the BDUT with the additional information type to be tested.

Criteria of acceptance

The BDUT sends the frames to the bus as expected, e.g. with time delay.

- **Test Step 3 (direct)**

[8] Please refer to the device profile and the manufacturer's product documentation (PICS/PIXIT) for supported additional info types. The contents of the additional information field are described in AN033.

EMI1/2 equivalent test: none

Purpose

Check whether the BDUT discards unsupported additional information types.

Procedure

Send link layer messages to the BDUT with unsupported additional information type(s)

KNX-USB-HID frame examples: 0113170008000F01030000**110401020001BCE00000FFFF010000**
0113190008001101030000**110605040000FFFFBCE00000FFFF010000**

Criteria of acceptance

The BDUT ignores the received additional information and sends the frames to the bus independent of the (unsupported) additional information given in the cEMI frame. The BDUT sends back an L_Data confirmation without the additional information types given in the L_Data request-

Expected KNX-USB-HID frame for the stimuli examples above:
0113130008000B01030000**2E00BCE002FFFFFF010000**

10.3 Testing of Network Layer Services

The cEMI protocol does not support any services on Network Layer.

10.4 Testing of Transport Layer Services

The cEMI protocol does not (yet) support any services on Transport Layer.

10.5 Testing of Application Layer Services

The cEMI protocol does not support any services on Application Layer.

10.6 Testing of Local Device Management Services

Tests are meaningful only “direct” (according to the figure in the introduction clause 1).

10.6.1 M_PropRead-Service

General remark on error handling: for the test steps with expected negative responses, the detailed error information (error type codes) as specified in the protocol specification is shown in the frame examples underneath. In all these cases, an “unspecified error” (error type ‘00h’) is allowed alternatively as the error return code to fulfil the minimum error handling requirements.

- **Step 1: Property Read with correct parameters**

Purpose

Check whether BDUT sends the local confirmation with expected correct data.

Procedure

The BDUT receives an M_PropRead.req, e.g. to Object Type Property of cEMI server Object:

Service	cEMI Local Service Information Data
mc	0008 01 01 1001

KNX-USB-HID frame example: 01130F0008000701030000**FC000801011001**

Criteria of acceptance

The BDUT sends the following M_PropRead.con:

Service	cEMI Local Service Information Data
mc	0008 01 01 1001 0008

KNX-USB-HID frame example: 0113110008000901030000**FB0008010110010008**

Repeat the test with other (correct) parameters:

- Test with other Object Type
- Test with Property ID ≠ 1
- Test with Count ≠ 1 (if applicable)
- Test with Start Index ≠ 001 (if applicable)
- Combinations of them

Note: Testability & Test frames are dependent on implemented features.

• Step 2: Property Read of array element 0 (current number of valid array elements)

Purpose

Check whether BDUT sends the local confirmation with expected correct data.

Procedure

The BDUT receives an M_PropRead.req, e.g. to Object Type Property of cEMI server Object:

Service	cEMI Local Service Information Data
mc	0008 01 01 1000

KNX-USB-HID frame example: 01130F0008000701030000**FC000801011000**

Criteria of acceptance

The BDUT sends the following M_PropRead.con:

Service	cEMI Local Service Information Data
mc	0008 01 01 1000 0001

KNX-USB-HID frame example: 0113110008000901030000**FB0008010110000001**

Repeat the test for (at least one) other existing properties, e.g. for the Group Address and association table properties.

- **Step 3: Property Read with illegal Object Type**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropRead.req, e.g. to Object Type Property of an inexistent Interface Object:

Service	cEMI Local Service Information Data
mc	0063 01 01 1001

KNX-USB-HID frame example: 01130F0008000701030000**FC006301011001**

Criteria of acceptance

The BDUT sends the following M_PropRead.con:

Service	cEMI Local Service Information Data
mc	0063 01 01 0001 07

KNX-USB-HID frame example: 011310000800080103000**0FB00630101000107**

- **Step 4: Property Read with illegal Object Instance**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropRead.req, e.g. to Object Type Property of an inexistent Object Instance:

Service	cEMI Local Service Information Data
mc	0008 02 01 1001

KNX-USB-HID frame example: 01130F0008000701030000**FC000802011001**

Criteria of acceptance

The BDUT sends the following M_PropRead.con:

Service	cEMI Local Service Information Data
mc	0008 02 01 0001 07

KNX-USB-HID frame example: 011310000800080103000**0FB00080201000107**

- **Step 5: Property Read with illegal Property ID**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropRead.req, e.g. to an inexistent Property within cEMI server Object:

Service	cEMI Local Service Information

	Data
mc	0008 01 32 1001

KNX-USB-HID frame example: 01130F0008000701030000**FC000801321001**

Criteria of acceptance

The BDUT sends the following M_PropRead.con:

Service	cEMI Local Service Information
	Data
mc	0008 01 32 0001 07

KNX-USB-HID frame example: 011310000800080103000**0FB00080132000107**

- **Step 6: Property Read with illegal Start Index**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropRead.req, e.g. to index 2 of the object type Property (cEMI server Object):

Service	cEMI Local Service Information
	Data
Mc	0008 01 01 1002

KNX-USB-HID frame example: 01130F0008000701030000**FC000801011002**

Criteria of acceptance

The BDUT sends the following M_PropRead.con:

Service	cEMI Local Service Information
	Data
Mc	0008 01 01 0002 09

KNX-USB-HID frame example: 011310000800080103000**0FB00080101000209**

- **Step 7: Property Read with illegal Count**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropRead.req, e.g. 2 elements of the object type Property (cEMI server Object):

Service	cEMI Local Service Information Data
mc	0008 01 01 2001

KNX-USB-HID frame example: 01130F0008000701030000**FC000801012001**

Criteria of acceptance

The BDUT sends the following M_PropRead.con:

Service	cEMI Local Service Information Data
mc	0008 01 01 0001 09

KNX-USB-HID frame example: 01131000080008010300 00**FB00080101000109**

Repeat this test step with an existing property, with start index = 0 and count ≠ 1.
Remark: For array element with start index = 0 (number of active array elements), only count = 1 is allowed.

10.6.2 M_PropWrite-Service

General remark on error handling: for the test steps with expected negative responses, the detailed error information (error type codes) as specified in the protocol specification is shown in the frame examples underneath. In all these cases, an “unspecified error” (error type ‘00h’) is allowed alternatively as the error return code to fulfil the minimum error handling requirements.

- **Step 1: Property Write with correct parameters**

Purpose

Check whether BDUT sends the local confirmation with expected correct data.

Procedure

The BDUT receives an M_PropWrite.req, e.g. a switch to busmonitor mode:

Service	cEMI Local Service Information Data
mc	0008 01 34 1001 01

KNX-USB-HID frame example: 01131000080008010300 00**F600080134100101**

Criteria of acceptance

The BDUT sends the following M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0008 01 34 1001

KNX-USB-HID frame example: 01130F0008000701030000**F5000801341001**

Read back the property value with M_PropRead Service to check if the property Value was written.

Repeat the test with other (correct) parameters:

- Test with other Object Type (if applicable)
- Test with other Property ID (if applicable)
- Test with Count ≠ 1 (if applicable)
- Test with Start Index ≠ 001 (if applicable)
- Combinations of them

Note: Testability & Test frames are dependent on implemented features.

• Step 2: Property Write to array element 0 (current number of valid array elements)

Purpose

Check whether BDUT sends the local confirmation with expected correct data.

Note: test is applicable only for a BDUT that supports at least one array-structured property with more than 1 array element, and with write-access enabled for this property.

Procedure

The BDUT receives an M_PropWrite.req, e.g. set Group Address table length to 0:

Service	cEMI Local Service Information
	Data
mc	0001 01 17 1000 0000

KNX-USB-HID frame example: 011311000800090103000**0F60001011710000000**

Criteria of acceptance

The BDUT sends the following M_PropWrite.con:

Service	cEMI Local Service Information
	Data
mc	0001 01 17 1000

KNX-USB-HID frame example: 01130F0008000701030000**F5000101171000**

Read back the property with M_PropRead Service to check if the property value was written.

Repeat the test for (at least one) other existing array-structured properties with write-access enabled.

• Step 3: Property Write to array element with array index > current valid nr. of elements

Purpose

Check whether BDUT changes the current number of valid array elements if written to a formerly unused element.

Note: test is applicable only for a BDUT that supports at least one array-structured property with more than 1 array element, and with write-access enabled for this property.

Procedure

The BDUT receives an M_PropWrite.req, e.g. write Group Addresses to the (now empty) Group Address table property:

Service	cEMI Local Service Information Data
mc	0001 01 17 2001 1000 FFFE

KNX-USB-HID frame example: 0113130008000B01030000**F60001011720011000FFFE**

Criteria of acceptance

The BDUT sends the following M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0001 01 17 2001

KNX-USB-HID frame example: 01130F0008000701030000**F5000101172001**

Read back the number of current valid array-elements of the written property with M_PropRead Service to check if the current valid number of array-elements was updated to the expected value (2 in the example above):

Stimuli:

Service	cEMI Local Service Information Data
mc	0001 01 17 1000

KNX-USB-HID frame example: 01130F0008000701030000**FC000101171000**

Expected Response:

Service	cEMI Service Information Data
mc	0001 01 17 1001 0002

KNX-USB-HID frame example: 0113110008000901030000**FB0001011710000002**

If applicable, repeat the test for (at least one) other existing array-structured properties with write-access enabled.

- **Step 4: Property Write with illegal Object Type**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req to a property of a non-existent interface object, e.g. try to write to PID_Table of an inexistent Interface Object (usually, write access is enabled for this property within objects that have this property):

Service	cEMI Local Service Information Data
mc	0063 01 17 1001 1000

KNX-USB-HID frame example: 011311000800090103000**0F60063011710011000**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0063 01 17 0001 07

KNX-USB-HID frame example: 011310000800080103000**0F500630117000107**

- **Step 5: Property Read with illegal Object Instance**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req to a property of a non-existent instance of an existent interface object, e.g. try to switch to busmonitor mode by writing to PID_CommMode of 2nd instance of the cEMI server object:

Service	cEMI Local Service Information Data
mc	0008 02 34 1001 01

KNX-USB-HID frame example: 011310000800080103000**0F600080234100101**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0008 02 34 0001 07

KNX-USB-HID frame example: 011310000800080103000**0F500080234000107**

- **Step 6: Property Write with illegal Property ID**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req to non-existent property within an existent interface object, e.g. try to write to Property 50 of the cEMI server object:

Service	cEMI Local Service Information Data
mc	0008 01 32 1001 00

KNX-USB-HID frame example: 011310000800080103000**0F600080132100100**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0008 01 32 0001 07

KNX-USB-HID frame example: 011310000800080103000**0F500080132000107**

- **Step 7: Property Write with illegal Start Index**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req to an existent property but with array-start index out of range, e.g. try to write to PID_CommMode within cEMI server object:

Service	cEMI Local Service Information Data
mc	0008 01 34 1002 00

KNX-USB-HID frame example: 011310000800080103000**0F600080134100200**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0008 01 34 0002 09

KNX-USB-HID frame example: 011310000800080103000**0F500080134000209**

- **Step 8: Property Write with illegal Count**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req to an existent property but with an illegal count (number of array elements to write), e.g. try to write 2 elements to PID_CommMode within cEMI server object:

Service	cEMI Local Service Information Data
mc	0008 01 34 2001 0102

KNX-USB-HID frame example: 011311000800090103000**0F60008013420010102**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0008 01 34 0001 09

KNX-USB-HID frame example: 011310000800080103000**0F500080134000109**

Repeat this test step with an existing property, with start index = 0 and count ≠ 1, if applicable.
Remark: For array element with start index = 0 (number of active array elements), only count = 1 is allowed.

Procedure

Example: Try to write 2 elements to Group Address table starting from array-index 0:

Service	cEMI Local Service Information Data
mc	0001 01 17 1000 0001 1000

KNX-USB-HID frame example: 0113130008000B0103000**0F6000101172000000011000**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information Data
mc	0001 01 17 0001 09

KNX-USB-HID frame example: 011310000800080103000**0F500010117000009**

- **Step 9: Property Write to a read-only Property**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req to a read-only property, e.g. try to write to PID_MediaType within cEMI server object:

Service	cEMI Local Service Information
	Data
mc	0008 01 33 1001 00

KNX-USB-HID frame example: 011310000800080103000**0F600080133100100**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information
	Data
mc	0008 01 33 0001 05

KNX-USB-HID frame example: 011310000800080103000**0F500080133000105**

- **Step 10: Property Write with a Value out of allowed range**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req, e.g. try to switch to an invalid communication mode:

Service	cEMI Local Service Information
	Data
mc	0008 01 34 1001 1F

KNX-USB-HID frame example: 011310000800080103000**0F600080134100011F**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information
	Data
mc	0008 01 34 0001 01

KNX-USB-HID frame example: 011310000800080103000**0F500080134000101**

Note: Error code 03 is also allowed in the example shown above.

- **Step 11: Property Write with a Data Type Conflict**

Purpose

Check whether BDUT sends the local confirmation with count = 0 and the corresponding error code.

Procedure

The BDUT receives an M_PropWrite.req, e.g. try to switch to busmonitor mode with a wrong property data type (wrong length):

Service	cEMI Local Service Information
	Data
mc	0008 01 34 1001 0001

KNX-USB-HID frame example: 011311000800090103000**0F60008013410010001**

Criteria of acceptance

The BDUT sends the following negative M_PropWrite.con:

Service	cEMI Local Service Information
	Data
mc	0008 01 34 0001 08

KNX-USB-HID frame example: 011310000800080103000**0F500080134000108**

- **Additional Test Steps: Error Codes**

Purpose

If applicable[9], check whether BDUT sends the local confirmation with count = 0 and expected error codes as far as not yet tested in test steps 1 to 11.

Procedure

The BDUT receives an M_PropWrite.req that causes an error with the error code to be tested

Criteria of acceptance

The BDUT sends the negative M_PropWrite.con with the expected error code:

10.6.3 M_PropInfo-Service

M_PropInfo Service is an optional feature. Check for the device profile / product information from the manufacturer, if the service is implemented in the BDUT.

- **Test Step**

Purpose

Check whether the BDUT transmits an M_PropInfo.ind with correct frame structure and data.

Procedure

Stimulate BDUT to send an M_PropInfo.ind

Criteria of acceptance

The BDUT sends the correct M_PropInfo.ind:

Service	cEMI Local Service Information
	Data
mc	*OT;Inst;PID;start-Idx/count;data*

KNX-USB-HID frame example: 011310000800080103000**0F7…..**

If applicable, repeat the test for (at least one) other properties.

[9] Please refer to manufacturer's product documentation (PICS/PIXIT) for a list of supported error codes

10.6.4 M_Reset-Service

M_Reset is a mandatory feature of a cEMI server device.

- **Step 1: Reaction to an M_Reset.req**

Purpose

Check whether BDUT reacts as expected after a reset from the client.

Procedure

The BDUT receives an M_Reset.req:

Service

mc

KNX-USB-HID frame example: 011309000800010103000**0F1**

Note: The M_Reset.req message may result in a breakdown of a connection on the level of the host protocol, in which cEMI is used. Example: KNXnet/IP.

Criteria of acceptance

The reaction to an M_Reset.ind is cEMI server specific. The cEMI server device shall react to an M_Reset.ind as stated by the product supplier in the product documentation (PICS/PIXIT).

If applicable, the BDUT sends an M_Reset.ind[10]:

Service

mc

KNX-USB-HID frame example: 011309000800010103000**0F0**

Step 2: behaviour after a power-on

Purpose

Check whether BDUT reacts as expected after Power-On of the BDUT.

Procedure

- Disconnect BDUT's power supplying connection and reconnect

Criteria of acceptance

The behaviour after a supply power-on is cEMI server specific. The cEMI server device shall react to a supply power on as stated by the product supplier in the product documentation (PICS/PIXIT).

[10] Not applicable for KNXnet/IP devices.

If applicable, the BDUT sends an M_Reset.ind[11]:

Service
mc

KNX-USB-HID frame example: 011309000800010103000**0F0**

10.6.5 M_FuncPropCommand-Service

- **Step 1: FunctionPropertyCommand with correct parameters**

Purpose

Check whether the BDUT sends the local confirmation with expected correct data and Return_code = 00h.

Procedure

The BDUT receives an M_FuncPropCommand.req, e.g. clear routing table:

Service	cEMI Local Service Information
	Data
mc	0006 01 38 00 01

KNXnet/IP frame example: 06100310001204810400**F8000601380001**

Criteria of acceptance

The BDUT sends the following M_FuncPropStateResponse.con:

Service	cEMI Local Service Information
	Data
mc	0006 01 38 **00** 01

KNXnet/IP frame example: 06100310001204810400**FA000601380001**

- **Step 2: FunctionPropertyCommand with incorrect data**

Purpose

Check whether the BDUT sends the local confirmation with Return_code ≠ 00h.

Procedure

The BDUT receives an M_FuncPropCommand.req, e.g. with faulty data:

Service	cEMI Local Service Information
	Data
mc	0006 01 38 00 08

KNXnet/IP frame example: 06100310001204810400**F8000601380008**

[11] Not applicable for KNXnet/IP devices and USB Interface devices (powered from USB-side) due to reasons on protocol/management level.

Criteria of acceptance

The BDUT sends the following M_FuncPropStateResponse.con:

Service	cEMI Local Service Information
	Data
mc	0006 01 38 **FF** 08

KNXnet/IP frame example: 06100310001204810400**FA00060138FF08**

- **Step 3: FunctionPropertyCommand with illegal parameters**

Purpose

Check whether the BDUT sends the local confirmation without data and Return_code

Procedure

The BDUT receives an M_FuncPropCommand.req to a property with property data type ≠ PDT_Function:

Service	cEMI Local Service Information
	Data
mc	0006 00 01 00 01

KNXnet/IP frame example: 06100310001204810400**F8000600010001**

Criteria of acceptance

The BDUT sends the following M_FuncPropStateResponse.con:

Service	cEMI Local Service Information
	Data
mc	0006 00 01

KNXnet/IP frame example: 06100310001004810400**FA00060001**

Repeat the test with other illegal parameters:

- Test with illegal Object Type (an inexistent interface object)
- Test with illegal Object Instance (e.g. instance >1 if this does not exist in the BDUT)
- Test with illegal Property ID

10.6.6 M_FuncPropStateRead-Service

- **Step 1: FunctionPropertyStateRead with correct parameters**

Purpose

Check whether the BDUT sends the local confirmation with expected correct data and Return_code = 00h.

Procedure

The BDUT receives an M_FuncPropStateRead.req, e.g. test if routing table is cleared:

Service	cEMI Local Service Information Data
mc	0006 01 38 00 01

KNXnet/IP frame example: 06100310001204810400**F9000601380001**

Criteria of acceptance

The BDUT sends the following M_FuncPropStateResponse.con:

Service	cEMI Local Service Information Data
mc	0006 01 38 **00** 01

KNXnet/IP frame example: 06100310001204810400**FA000601380001**

- **Step 2: FunctionPropertyStateRead with incorrect data**

Purpose

Check whether BDUT sends the local confirmation with Return_code ≠ 00h.

Procedure

The BDUT receives an M_FuncPropStateRead.req e.g. with faulty data:

Service	cEMI Local Service Information Data
mc	0006 01 38 00 08

KNXnet/IP frame example: 06100310001204810400**F9000601380008**

Criteria of acceptance

The BDUT sends the following M_FuncPropStateResponse.con:

Service	cEMI Local Service Information Data
mc	0006 01 38 **FF** 08

KNXnet/IP frame example: 06100310001204810400**FA00060138FF08**

- **Step 3: FunctionPropertyStateRead with illegal parameters**

Purpose

Check whether the BDUT sends the local confirmation without data and Return_code.

Procedure

The BDUT receives a M_FuncPropStateRead.req to a property with property data type ≠ PDT_Function:

Service	cEMI Local Service Information Data
mc	0006 00 01 00 01

KNXnet/IP frame example: 06100310001204810400**F9000600010001**

Criteria of acceptance

The BDUT sends the following M_FuncPropStateResponse.con:

Service	cEMI Local Service Information Data
mc	0006 00 01

KNXnet/IP frame example: 06100310001004810400**FA00060001**

Repeat the test with other illegal parameters:

- Test with illegal Object Type (an inexistent interface object)
- Test with illegal Object Instance (e.g. instance >1 if this does not exist in the BDUT)
- Test with illegal Property ID

10.7 Testing of cEMI Server's Interface Objects

10.7.1 General Test Guideline

Interface Objects inside BDUT shall be tested as following:

- Read from read-only properties
- Try to read from non-existent properties
- Write to properties with write-access enabled
- Try to write to read-only properties
- Try to write to non-existent properties

Always check for correct/expected data.

Test examples of read/write requests that cause a negative confirmation are given in the property server tests according clauses 10.6.1& 10.6.2.

10.7.2 Device Object

Tests shall cover:

- Reading from all implemented (read-only) properties.
- Writing to (and reading back from) all implemented properties with write-access enabled.
- At least once each negative test (try to read from a non-existent property, try to write to a read-only property and try to write to a non-existent property).

The information, which properties are implemented in the BDUT shall be taken from the cEMI server device profile, respectively from the manufacturer's product documentation.

10.7.3 cEMI Server Object

Tests shall cover:

- Reading from all implemented properties
- Writing to and reading back from all implemented properties with write-access enabled.
- At least once each negative test (try to read from a non-existent property, try to write to a read-only property and try to write to a non-existent property).

The information, which properties are implemented in the BDUT shall be taken from the cEMI server device profile, respectively from the manufacturer's product documentation.

10.7.4 Other Interface Objects

In order to limit the (certification) test efforts, sample tests are sufficient here.
A reasonable set of properties shall be tested. The sample tests shall cover:

- Reading from a sample set of implemented properties
- Write to and read back from a sample set of properties with write-access enabled.