Application Note 157/12 v01

Title:	**Mask 57B0h**	
Status:		**Date:**
	Draft Proposal	2012.11.12
Transitional period:	Immediate effect after Final Voting.	
Date:	2012.11.12	

Subject: Specification of Mask 57B0h

Documents **Modified**

[01] Chapter 3/5/3 “Configuration Procedures”
[02] Volume 6 “Profiles v1.8”

Referred

[03] Chapter 3/2/6 “KNX IP”
[04] Chapter 3/3/4 “Transport Layer”
[05] Chapter 3/3/7 “Application Layer”
[06] Chapter 3/4/1 “Application Interface Layer””
[07] Chapter 3/5/1 “Resources”
[08] Chapter 3/5/2 “Management Procedures” v1.4
[09] Chapter 3/6/3 “External Message Interface”
[10] Chapter 3/8/2 “KNXnet/IP Core”
[11] Chapter 3/8/3 “KNXnet/IP Management”
[12] Chapter 3/8/5 “KNXnet/IP Routing”
[13] Chapter 8/3/4 ““Transport Layer Tests” v1.0 AS of 2002.02.05”
[14] AN115 “Mask 5705h”
[15] AN127 “Master Reset”

Document updates

Version	Date	Modifications
TFIP007-01	2010.12.02	Document creation based on TFIP001-11 mask 5705h_20100713_editbyTP.doc
TFIP007-02	2011.04.18	Update according to the comments of SDB
TFIP007-03	2011.06.17	Update according to the TelCo (2011.06.14)
TFIP007-04	2012.05.31	Update according to the TelCo (2011.06.21)
TFIP007-05	2012.07.04	Update according to the TelCo (2012.06.05)
TFIP007-06	2012.07.27	Update according to the TelCo (2012.07.05)
TFIP007-07	2012.07.30	Update according to the TelCo (2012.07.30)
TFIP007-08	2012.08.17	Update according to the TelCo (2012.08.07)
TFIP007-09	2012.09.20	Update of load procedure based on documentation of Dr. Gütter (IT GmbH)
AN157 v01	2012.11.09	Preparation of the Draft Proposal.

Contents

1 General

1.1 Scope

🗎 *This clause is not intended for integration in the KNX Specifications.*

In Application Note AN115 ([14]) the Mask Version 5705h is specified as the first Profile for end devices for usage on the KNX IP medium. The Mask 5705h is limited to 255 Group Objects. As especially KNX IP devices are quite complex an additional solution is required without this rigid limitation.

To avoid the definition of a complete new Profile from scratch, it is proposed to reuse the existing Profile System B also for KNX IP. This AN specifies the necessary steps to implement a KNX IP device based on the Profile of mask 57B0h. In addition to AN115, which defines the first Mask Version for KNX IP and Chapter 3/2/6 “KNX IP” ([03]), which defines the medium-specific Physical Layer and Data Link Layer services for the KNX IP medium, this document defines the Profile for mask 57B0h devices (System B for KNX IP medium).

2 Specification

2.1 Terms and definitions

🗎 *This clause is not intended for integration in the KNX Specifications.*

This document does not introduce any new terms or definitions.

2.2 Stack and communication

🗎 *This clause is not intended for integration in the KNX Specifications.*

This document does not introduce neither modify any stack or communication specifications.

2.3 Resource definition or used Resources

🗎 *This clause is not intended for integration in the KNX Specifications.*

This document does not introduce neither modify any Resources.

2.4 Management Procedures

🗎 *This document does not change or introduce any Management Procedures.*

2.5 Configuration Procedures

🗎 *This clause shall be integrated in Chapter 3/6/3 “Configuration Procedures” ([01]).*

2.5.1 Merge points

The support of Mask 57B0h devices in the Management Client (ETS) shall foresee the support of merge points, which shall be used to integrate application specific Management Procedures into the default Configuration Procedures.

Legend

Symbol	Description
O	The merge point is optional.
M	The merge point is mandatory.

The following MergeIDs are defined for this Mask 57B0h.

MergeID	O/M	Description
1	O	This is used for a complete download before any modifying action is taken. A typical use is to verify some Properties or the state of the device.
2	M	This is executed just after switching the Application Program 1 Load State Machine to Loading. It is expected to contain the necessary segment allocation: DMP_LoadStateMachineWrite_R_Co_IO(object_index = OIDX_-APPLICATION_PROGRAM_1, data = {event = 0Bh, length, mode, fill}) Additionally, a Property Value may be written for subsegmentation: DMP_InterfaceObjectWrite_RCo(object_index=_APPLICATION_PROGRAM_1, PID= PID_MCB_TABLE, start_index, element_count, data). For Partial Download it is essential, that bit 0 of the mode octet is 0 (do not fill) at least if parameters are available.
3	M	This is executed just after switching the Application Program 2 Load State Machine to Loading. It is expected to contain the necessary segment allocation: DMP_LoadStateMachineWrite_R_Co_IO(object_index = OIDX_APPLICATION_PROGRAM_2, data = {event = 0Bh, length, mode, fill}) Additionally, a Property Value may be written for subsegmentation: DMP_InterfaceObjectWrite_RCo(object_index=_APPLICATION_PROGRAM_2, PID= PID_MCB_TABLE, start_index, element_count, data). For Partial Download it is essential, that bit 0 of the mode octet is 0 (do not fill) at least if parameters are available.
4	M	This is executed after all segments are allocated. It is expected to contain load controls necessary to write the Application Program data including parameters for Application Program 1. Typically, this will be a DMP_MemWrite_RCoV.
5	M	This is executed after all segments are allocated. It is expected to contain load controls necessary to write the Application Program data including parameters for Application Program 2. Typically, this will be a DMP_MemWrite_RCoV.
6	O	This is executed after all Load State Machines are switched to Loaded. A typical use is to write additional parameters via Properties that are accessible only after loading the Application Program.

MergeID	O/M	Description
7	O	This is executed immediately before the final Restart/Disconnect. If differential download is supported, this part should contain LoadImage records for the MCB (memory control block) tables (for Application Program 1: Interface Objects 1 to 4, for Application Program 2: Interface Object 5)

2.5.2 Load Control implementation

The following table gives the implementation of the used “Load Controls”.

Load Control	Implementation	Remarks

LdCtrlConnect	DM_Connect(flags=0)
	DM_Authorize(flags=0, key=project_key)
		This load control will be ignored if a connection is already established.
		Authorization is performed in the form DM_Authorize2-_RCo [1].

LdCtrl-Disconnect	DM_Disconnect(flags=0)	ETS may ignore this load control if it plans to access the device immediately again.
LdCtrlRestart	DM_Restart(flags=0)	
LdCtrlUnload	DM_LoadStateMachineWrite(event=-Unload) Invalidate cached base pointer	In all System B load procedures, Interface Object addressing is done via Object Index.
LdCtrlLoad	DM_LoadStateMachineWrite(event=Load) Invalidate cached base pointer	
LdCtrl-LoadCompleted	DM_LoadStateMachineWrite(event=-LoadCompleted)	
LdCtrl-RelSegment	DM_LoadStateMachineWrite(event=-AllocRelSegment [2]) Invalidate cached base pointer	
LdCtrl-WriteRelMem	If base pointer not yet determined DM_InterfaceObjectRead(PID_TABLE_-REFERENCE) DM_MemWrite(...)/DM_UserMemWrite(...) depending on address	

[1] In [02] clause 3.5.2, System B is not listed in the paragraph with the Profiles. That indication is however informative. Please refer to [01] to read which Management Procedures are mandatory for which Profile.

[2] In [02] clause 3.27.1 LSM event is not listed.

Load Control	Implementation	Remarks
LdCtrlLoad-ImageRelMem	If base pointer not yet determined DM_InterfaceObjectRead(PID_TABLE_-REFERENCE) DM_MemRead(...)/DM_UserMemRead(...)depending on address	
LdCtrlWriteProp	DM_InterfaceObjectWrite(...)	
LdCtrlReadProp	DM_InterfaceObjectRead(...)	
LdCtrlCompare-Prop	DM_InterfaceObjectVerify(...)	

2.5.3 Input

System B requires the definition of fixed constants for the Object Indexes.

Constants

OIDX_ADDRESS_TABLE = 1	This is the index of the Interface Object holding the Group Address Table.
OIDX_ASSOCIATION_TABLE = 2	This is the index of the Interface Object holding the Association Table.
OIDX_GROUPOBJECT_TABLE = 3	This is the index of the Interface Object holding the Group Object Table.
OIDX_APPLICATION_PROGRAM_1 = 4	This is the index of the Interface Object holding the Application Program 1.
OIDX_APPLICATION_PROGRAM_2 = 5	This is the index of the Interface Object holding the Application Program 2.

Application specific

GrAT.length:	The length of the Group Address Table, according the number of GAs assigned by the user.
AscT.length:	The length of the Group Object Association Table, according to the number of Group Address associations configured by the user.
GrObjT.length:	The length of the Group Object Table, according to the number of Group Objects.

From device

GrAT.start:	The base address of the Group Address Table.
AscT.start:	The base address of the Association Table.
GrObjT.start:	The base address of the Group Object Table.

From user

IA:	Individual Address of the Management Server (device) to be configured.

2.5.4 Complete Download

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.2)
01	`<LdCtrlConnect />`		01
		Reading DD0 is part of DM_Connect, Verfying is done in the general pre-download verification step (see below)	02
		Authorize: Always done implicitly by LdCtrlConnect if device supports authorization	03
		Verify Manufacturer: Done in the general pre-download verification step (see below)	04
02	`<LdCtrlMerge MergeId="1" />`		
03	`<LdCtrlUnload LsmIdx="5" />`	Any Errors here are ignored if only AP 1 shall be loaded (see below)	05
04	`<LdCtrlUnload LsmIdx="4" />`		
05	`<LdCtrlUnload LsmIdx="3" />`		
06	`<LdCtrlUnload LsmIdx="2" />`		
07	`<LdCtrlUnload LsmIdx="1" />`		
08	`<LdCtrlLoad LsmIdx="5" />`	Not executed if only AP 1 shall be loaded	06 Part 1
09	`<LdCtrlMerge MergeId="3" />`	Not executed if only AP 1 shall be loaded Expected to contain <LdCtrlRelSegment LsmIdx=”5” ... /> and optionally writing Property PID_MCB_TABLE for sub-segmentation	06 Part 2
10	`<LdCtrlLoad LsmIdx="4" />`		07 Part 1

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.2)
11	`<LdCtrlMerge MergeId="2" />`	Expected to contain <LdCtrlRelSegment LsmIdx="4" ... /> and optionally writing Property PID_MCB_TABLE for sub-segmentation	07 Part 2
12	`<LdCtrlLoad LsmIdx="3" />`		08 Part 1
13	`<LdCtrlRelSegment LsmIdx="3" Size="*" Mode="0" Fill="0" />`	Size = current size of Group Object table	08 Part 2
14	`<LdCtrlLoad LsmIdx="1" />`		09 Part 1
15	`<LdCtrlRelSegment LsmIdx="1" Size="*" Mode="0" Fill="0" />`	Size = current size of Group Address Table	09 Part 2
16	`<LdCtrlLoad LsmIdx="2" />`		10 Part 1
17	`<LdCtrlRelSegment LsmIdx="2" Size="*" Mode="0" Fill="0" />`	Size = current size of Group Object Association Table	10 Part 2
18	`<LdCtrlMerge MergeId="5" />`	Expected to contain the load controls necessary to write the Application Program data including parameters for AP2.	06 Part 3 and 4
19	`<LdCtrlMerge MergeId="4" />`	Expected to contain the load controls necessary to write the Application Program data including parameters for AP1.	07 Part 3 and 4
20	`<LdCtrlWriteRelMem Object Index="3" Offset="0" Size="*" Verify="true" />`	Size = current size of Group Object table	08 Part 3 and 4
21	`<LdCtrlWriteRelMem Object Index="2" Offset="0" Size="*" Verify="true" />`	Size = current size of association table	10 Part 3 and 4
22	`<LdCtrlWriteRelMem Object Index="1" Offset="0" Size="*" Verify="true" />`	Size = current size of Group Address Table	09 Part 3 and 4

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.2)
23	`<LdCtrlWriteProp Object Index="1" PID="53" Count="*" Verify="true" />`	Only for Masg Version 17B0h Count = current size of group responder table	10 Part 5
24	`<LdCtrlWriteProp Object Index="5" PID="13" Verify="true" InlineData="*" />`	Not executed if only AP 1 shall be loaded InlineData = ApplicationID of AP2	06 Part 5
25	`<LdCtrlWriteProp Object Index="4" PID="13" Verify="true" InlineData="*" />`	InlineData = ApplicationID of AP1	07 Part 5
26	`<LdCtrlLoadCompleted LsmIdx="5" />`	Not executed if only AP 1 shall be loaded	06 Part 6
27	`<LdCtrlLoadCompleted LsmIdx="4" />`		07 Part 6
28	`<LdCtrlLoadCompleted LsmIdx="3" />`		08 Part 5
29	`<LdCtrlLoadCompleted LsmIdx="2" />`		10 Part 6
30	`<LdCtrlLoadCompleted LsmIdx="1" />`		09 Part 5
31	`<LdCtrlMerge MergeId="6" />`		
32	`<LdCtrlMerge MergeId="7" />`	Expected to contain a reading of the Property PID_MCB_TABLE if differential download shall be supported	06,07,10 Part 7 08,09 Part 6
33	`<LdCtrlWriteProp Object Index="0" PID="73" Verify="true" InlineData="*" />`	Only for Mash Version 17B0h.	Not in [01]
34	`<LdCtrlRestart />`		Not in [01]

Pre-download verification step

ETS shall verify if it is allowed to load the data into the device right at the start and outside of the Configuration Procedure implementation. This includes:

- Reading the Device Descriptor Type 0 from the installed device and verifying whether it is identical to or compatible with the Device Descriptor of the device that it holds in its project information and that it intends to configure.
- Reading the Manufacturer Identifier from the installed device and verifying whether it is identical to or compatible with the Manufacturer Identifier of the device that it holds in its project information and that it intends to configure.
- A list of additional Resources of which the value of the installed device and the value stored in the project for the device that it indends to configure shall be identical. For System B these shall be the Properties PID_ORDER_INFO (identical), PID_VERSION (identical or higher), and PID_HARDWARE_TYPE (identical) of the Device Object. The comparison is against the product data (parameter value in Application Program); if no such parameter exists the comparison shall be skipped.

Post-download steps

If an authorization access key is configured in the project, all devices supporting authorization shall be locked with this key. This is implemented outside of the Configuration Procedure implementation. If fact, locking shall already be done immediately after assigning the Individual Address.

Remarks on differences

- Splitting up the different parts of steps 06 to 10 in [01] is done in order to catch errors early.
- If no AP2 is present in the product database entry, errors accessing the AP2 Interface Object are ignored.

🗎 *This has been introduced because it was not clear from the handbook, whether or not the AP2 Interface Object is mandatory for System B devices.*

- It should be clarified if the procedure ends with Restart (ETS) or Disconnect ([01]).
- In contrast to [01], ETS does not evaluate PID_TABLE_REFERENCE to check if the allocation succeeded, but looks at the Load State and expects it to be ‘error’ after an unsuccessful allocation. To be clarified.

2.5.5 Partial Download “Group Communication”

In [01], this corresponds a combination of the cases “Partial Download of the “Group Object Table”” (denoted C here), “Partial Download of the “Group Address Table”” (D) and “Partial Download of the “Group Object Association Table”” (E). ETS does not offer these cases as separate procedures.

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 C-E)
01	`<LdCtrlConnect />`		C01, D01, E01
02	`<LdCtrlUnload LsmIdx="3" />`		C05
03	`<LdCtrlUnload LsmIdx="2" />`		E05
04	`<LdCtrlUnload LsmIdx="1" />`		D05
05	`<LdCtrlLoad LsmIdx="3" />`		C06 Part 1
06	`<LdCtrlRelSegment LsmIdx="3" Size="*" Mode="0" Fill="0" />`	Size = current size of Group Object table	C06 Part 2
07	`<LdCtrlLoad LsmIdx="1" />`		D06 Part 1
08	`<LdCtrlRelSegment LsmIdx="1" Size="*" Mode="0" Fill="0" />`	Size = current size of Group Address Table	D06 Part 2
09	`<LdCtrlLoad LsmIdx="2" />`		E06 Part 1
10	`<LdCtrlRelSegment LsmIdx="2" Size="*" Mode="0" Fill="0" />`	Size = current size of association table	E06 Part 2
11	`<LdCtrlWriteRelMem Object Index="3" Offset="0" Size="*" Verify="true" />`	Size = current size of Group Object table	C06 Part 5
12	`<LdCtrlWriteRelMem Object Index="2" Offset="0" Size="*" Verify="true" />`	Size = current size of Group Object Association Table	E06 Part 5
13	`<LdCtrlWriteRelMem Object Index="1" Offset="0" Size="*" Verify="true" />`	Size = current size of Group Address Table	D06 Part 5

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 C-E)
14	<LdCtrlWriteProp Object Index="1" PID="53" Count="*" Verify="true" />	Only for 17B0 Count = current size of group responder table	Missing in [01]
15	<LdCtrlLoadCompleted LsmIdx="3" />		C06 Part 7
16	<LdCtrlLoadCompleted LsmIdx="2" />		E06 Part 7
17	<LdCtrlLoadCompleted LsmIdx="1" />		D06 Part 7
18	<LdCtrlMerge MergeId="7" />		
19	<LdCtrlWriteProp Object Index="0" PID="73" Verify="true" InlineData="*" />	Only for 17B0	Not in [01]
20	<LdCtrlRestart />		Not in [01]

Pre-download verification step:

- Before a Partial Download is performed, ETS will check if the Application Program(s) are loaded and the ApplicationIDs of AP1 (and AP2 if present) are identical.

Remarks on differences

- C06/D06/E06 Part 6 is probably an error since the communication tables do not have a Property PID_PROGRAM_VERSION.
- Splitting up the different parts of step C-E06 in [01] is done in order to catch errors early.
- ETS does not currently implement differential download for this case, i.e. the communication tables are always loaded completely. Reasons: the moderate size of typical tables and the fact that an added or removed group association affects half of the Group Address and Group Object Association Table in the average.
- It should be clarified if the procedure ends with Restart (ETS) or Disconnect) [01])
- In contrast to [01] ETS does not evaluate PID_TABLE_REFERENCE to check if the allocation succeeded, but looks at the Load State and expects it to be ‘error’ after an unsuccessful allocation. To be clarified.

2.5.6 Partial Download “Parameters”

In [01], this corresponds to combination of the cases “Partial Download of the ‘Application Program 2’ ” (denoted A here) and “Partial Download of the ‘Application Program 1’ ” (B).

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 A-B)
01	<LdCtrlConnect />		A01, B01
02	<LdCtrlLoadImageProp Object Index="4" PID="7" />	See below	
03	<LdCtrlLoadImageProp Object Index="5" PID="7" />	Not executed if only AP 1	
04	<LdCtrlUnload LsmIdx="5" />	Not executed if only AP 1	A05
05	<LdCtrlUnload LsmIdx="4" />		B05
06	<LdCtrlLoad LsmIdx="5" />	Not executed if only AP 1	A06 Part 1
07	<LdCtrlMerge MergeId="3" />	Not executed if only AP 1 Expected to contain <LdCtrlRelSegment LsmIdx=”5” ... /> and optionally writing Property PID_MCB_TABLE for sub-segmentation. The allocation must be done with bit#0 of the mode octet = 0 (do not fill).	A06 Part 2
08	<LdCtrlCompareProp Object Index="5" PID="7" InlineData="*" />	Not executed if only AP 1 * = Data read in 03 See below	
09	<LdCtrlLoad LsmIdx="4" />		B06 Part 1

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 A-B)
10	<LdCtrlMerge MergeId="2" />	Expected to contain <LdCtrlRelSegment LsmIdx=”4” ... /> and optionally writing Property PID_MCB_TABLE for sub-segmentation. The allocation must be done with bit#0 of the mode octet = 0 (do not fill).	B06 Part 2
11	<LdCtrlCompareProp Object Index="4" PID="7" InlineData="*" />	* = Data read in 02 See below	
12	<LdCtrlMerge MergeId="5" />	Not executed if only AP 1 Expected to contain the load controls necessary to write the Application Program data including parameters for AP2	A06 Part 5
13	<LdCtrlMerge MergeId="4" />	Expected to contain the load controls necessary to write the Application Program data including parameters for AP1	B06 Part 5
14	<LdCtrlWriteProp Object Index="5" PID="13" Verify="true" InlineData="*" />	Not executed if only AP 1	A06 Part 6
15	<LdCtrlWriteProp Object Index="4" PID="13" Verify="true" InlineData="*" />		B06 Part 6
16	<LdCtrlLoadCompleted LsmIdx="5" />	Not executed if only AP 1	A06 Part 7
17	<LdCtrlLoadCompleted LsmIdx="4" />		B06 Part 7
18	<LdCtrlMerge MergeId="6" />		

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 A-B)
19	<LdCtrlMerge MergeId="7" />	Expected to contain a reading of the Property PID_MCB_TABLE if differential download shall be supported	A06/B06 Part 8
20	<LdCtrlWriteProp Object Index="0" PID="73" Verify="true" InlineData="*" />	Only for 17B0	Not in [13]
21	<LdCtrlRestart />		Not in [13]

Pre-download verification step

- Before a Partial Download is performed, ETS will check if the Application Program(s) are loaded and the ApplicationIDs of AP1 (and AP2 if present) are identical.
- In addition, ETS compares the CRCs read in the previous download with the CRCs in the device. If a mismatch is found, a standard Partial Download is performed instead of a differential download.
  Therefore, CRC comparison is not part of the actual Configuration Procedure in ETS.

Remarks on differences

- In addition to the CRC comparison, ETS also checks if the base pointers before and after the Unload/Load/Alloc sequence are identical (Steps 02, 03, 08, 09).
  If the base pointer would be different, Partial Download will not work and thus an error is reported by ETS in this case.
  Thus, it is a requirement for devices that want to support Partial Download, that after unloading AP1+AP2 and then allocating the same amount of memory again with mode “do not fill” the same memory range with unchanged data is allocated again and the same base pointers are returned.
- The CRC comparison in [01] seems to be wrong:
  At this stage, the LSM is in state ‘loaded’, so the CRCs in PID_MCB_TABLE may be incorrect (the device is supposed to calculate the CRC on the transition to the ‘loaded’ state).
  ETS reads and compares the CRCs as part of the pre-download checks before the actual load procedure runs so this is not a problem for ETS.
- Splitting up the different parts of step C-E06 in [01] is done in order to catch errors early.
- If an allocation error occurs, ETS does not try to recover as described in A08-12 and B08-11 but gives up and displays an error message.
- It should be clarified if the procedure ends with Restart (ETS) or Disconnect) [01])
- In contrast to [01], ETS does not evaluate PID_TABLE_REFERENCE to check if the allocation succeeded, but looks at the Load State and expects it to be ‘error’ after an unsuccessful allocation. To be clarified.

2.5.7 Partial Download “Group Communication and Parameters”

As this is just a combination of the other Partial Download cases, no detailed discussion is done here.

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 A-E)
01	<LdCtrlConnect />		
02	<LdCtrlLoadImageProp Object Index="4" PID="7" />		
03	<LdCtrlLoadImageProp Object Index="5" PID="7" />		

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 A-E)
04	<LdCtrlUnload LsmIdx="5" />		
05	<LdCtrlUnload LsmIdx="4" />		
06	<LdCtrlUnload LsmIdx="3" />		
07	<LdCtrlUnload LsmIdx="2" />		
08	<LdCtrlUnload LsmIdx="1" />		
09	<LdCtrlLoad LsmIdx="5" />		
10	<LdCtrlMerge MergeId="3" />		
11	<LdCtrlCompareProp Object Index="5" PID="7" InlineData="*" />		
12	<LdCtrlLoad LsmIdx="4" />		
13	<LdCtrlMerge MergeId="2" />		
14	<LdCtrlCompareProp Object Index="4" PID="7" InlineData="*" />		
15	<LdCtrlLoad LsmIdx="3" />		
16	<LdCtrlRelSegment LsmIdx="3" Size="*" Mode="0" Fill="0" />		
17	<LdCtrlLoad LsmIdx="1" />		
18	<LdCtrlRelSegment LsmIdx="1" Size="*" Mode="0" Fill="0" />		
19	<LdCtrlLoad LsmIdx="2" />		
20	<LdCtrlRelSegment LsmIdx="2" Size="*" Mode="0" Fill="0" />		
21	<LdCtrlMerge MergeId="5" />		
22	<LdCtrlMerge MergeId="4" />		
23	<LdCtrlWriteRelMem Object Index="3" Offset="0" Size="*" Verify="true" />		

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.3 A-E)
24	<LdCtrlWriteRelMem Object Index="2" Offset="0" Size="*" Verify="true" />		
25	<LdCtrlWriteRelMem Object Index="1" Offset="0" Size="*" Verify="true" />		
26	<LdCtrlWriteProp Object Index="1" PID="53" Count="*" Verify="true" />		
27	<LdCtrlWriteProp Object Index="5" PID="13" Verify="true" InlineData="*" />		
28	<LdCtrlWriteProp Object Index="4" PID="13" Verify="true" InlineData="*" />		
29	<LdCtrlLoadCompleted LsmIdx="5" />		
30	<LdCtrlLoadCompleted LsmIdx="4" />		
31	<LdCtrlLoadCompleted LsmIdx="3" />		
32	<LdCtrlLoadCompleted LsmIdx="2" />		
33	<LdCtrlLoadCompleted LsmIdx="1" />		
34	<LdCtrlMerge MergeId="6" />		
35	<LdCtrlMerge MergeId="7" />		
36	<LdCtrlWriteProp Object Index="0" PID="73" Verify="true" InlineData="*" />		
37	<LdCtrlRestart />		

2.5.8 Partial Download “Cfg”

This type of Partial Download is for quickly updating parameters affecting the behaviour of the device on the medium, like:

- “Repeater present” flag for PL
- IP configuration for IP enabled devices

The procedure is defined for 17B0 only.

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [13]
01	<LdCtrlConnect />		Not in [01]
02	<LdCtrlWriteProp Object Index="0" PID="73" Verify="true" InlineData="*" />		
03	<LdCtrlRestart />		

2.5.9 Unload

Nr	Device Management Procedure (Load Control)	Remarks	Corresponds to [01] (2.4.4)
01	<LdCtrlConnect />		01
02	<LdCtrlUnload LsmIdx="1" />		05
03	<LdCtrlUnload LsmIdx="2" />		
04	<LdCtrlUnload LsmIdx="3" />		
05	<LdCtrlUnload LsmIdx="4" />		
06	<LdCtrlUnload LsmIdx="5" />	Errors will be ignored here (to support devices without AP2 object)	
07	<LdCtrlWriteProp Object Index="0" PID="73" Verify="true" InlineData="FF" />	Only for 17B0	Not in [01].
08	<LdCtrlDisconnect />		

Remarks on differences

- ETS never verifies the Manufacturer Identifier on unload. This is because a device can be unloaded without project context or even without product data.

2.5.10 Setting of IP Properties

Goal

The goal of this Configuration Procedure is to write the IP system Properties as defined in the KNX specification [11] that the user of the ETS configured in device and line settings into KNXnet/IP and KNX IP devices.

Prerequisites

1. The routing multicast address is a valid class D IP address.
2. Device address, network mask and gateway address are valid and match together.
3. The target device already has a known Individual Address assigned.

Inputs

Constants

OID_KNXNETIP_PARAMETER	The Object Type of the KNXnet/IP Parameter Object
PID_PROJECT_INSTALLATION_ID	The Property Identifier of the project installation Property
PID_ADDITIONAL_INDIVIDUAL_ADDRESSES	The Property Identifier of the Property holding the array of additional Individual Addresses
PID_IP_ASSIGNMENT_METHOD	The Property Identifier of the Property containing the method of IP address assignment
PID_IP_ADDRESS	The Property Identifier of the IP address Property for manual address assignment
PID_SUBNET_MASK	The Property Identifier of the subnetwork mask Property for manual IP address assignment
PID_DEFAULT_GATEWAY	The Property Identifier of the Property containing the default gateway for manual assignment
PID_ROUTING_MULTICAST_ADDRESS	The Property Identifier of the routing multicast address Property
PID_TTL	The Property Identifier of the Property containing the multicast TTL
PID_FRIENDLY_NAME	The Property Identifier of the friendly name string Property

Product Data:

noAIAs	The number of additional Individual Addresses supported by this device.

Project:	
ProjID, InstID	The project ID and installation ID.
MC	The routing multicast address valid for this device.
TTL	The TTL for multicast communication in this device.
User:	
Name	The friendly name of this device.
IA	The Individual Address of the target device.
AIAs[]	The array of additional Individual Addresses.
IPAMethod	The selected method of IP address assignment.
IPA	The manually configured IP address.
SN	The manually configured IP subnetwork mask.
GW	The manually configured default gateway address.

Procedure

The following Configuration Procedure is inserted into the System B Configuration Procedure.

If the System B Configuration Procedure contains a DMP_Restart, it is inserted immediately before the first found DMP_Restart (the DMP_Restart is thereby removed). Otherwise it is inserted before the final DMP_Disconnect.

This Configuration Procedure is executed at a Complete Download or at a Partial Download “Cfg” (ETS resets the Cfg flag if the user changes the IP configuration).

```
      /* Establish a Transport Layer connection to the remote device. */
DMP_Connect_RCo(IA, connection-oriented)

      /* Check if the project installation ID needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_PROJECT_INSTALLATION_ID, start_index=1,
      noElements=1, data=rPIID)
If (rPIID != PIID(ProjID,InstID))
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_PROJECT_INSTALLATION_ID, start_index=1,
      noElements=1, data=PIID(ProjID,InstID))
Endif

      /* Check if the routing multicast address needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID=PID_ROUTING_MULTICAST_ADDRESS, start_index=1,
      noElements=1, data=rMC)
If (rMC != MC)
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID=PID_ROUTING_MULTICAST_ADDRESS, start_index=1,
      noElements=1, data=MC)
      restartRequired = true
Endif

      /* Check if the multicast TTL needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID=PID_TTL, start_index=1, noElements=1,
      data=rTTL)
If (rTTL != TTL)
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID=PID_TTL, start_index=1, noElements=1, data=TTL)
      restartRequired = true
Endif
```

```
      /* Check if the friendly name needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_FRIENDLY_NAME, start_index=1,
      noElements=1, data=rName)
If (rName != Name)
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_FRIENDLY_NAME, start_index=1,
      noElements=1, data=Name)
Endif

      /* Check if the IP assignment method needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_IP_ASSIGNMENT_METHOD, start_index=1,
      noElements=1, data=rIPAMethod)
If (rIPAMethod != IPAMethod)
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_IP_ASSIGNMENT_METHOD, start_index=1,
      noElements=1, data= IPAMethod)
      restartRequired = true
Endif

If(IPAMethod==fixed)

      /* Check if the IP address needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_IP_ADDRESS, start_index=1, noElements=1,
      data=rIPAMethod)
If (rIPAMethod != IPAMethod)
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_IP_ADDRESS, start_index=1, noElements=1,
      data= IPAMethod)
      restartRequired = true
Endif

      /* Check if subnetwork mask needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_SUBNET_MASK, start_index=1, noElements=1,
      data=rSN)
If (rSN != SN)
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_SUBNET_MASK, start_index=1, noElements=1,
      data=SN)
      restartRequired = true
Endif

      /* Check if the default gateway needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_DEFAULT_GATEWAY, start_index=1,
      noElements=1, data=rGW)
If (rGW != GW)
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_DEFAULT_GATEWAY, start_index=1,
      noElements=1, data= GW)
      restartRequired = true
Endif
Endif

      /* Check if the additional Individual Address needs to be written to the device. */
DMP_InterfaceObjectRead_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_ADDITIONAL_INDIVIDUAL_ADDRESSES,
      start_index=1, noElements= noAIAs, data=rAIA[])
For i=1 to noAIAs
If (rAIA[] != AIAs[i])
      DMP_InterfaceObjectWrite_R(flags=2, dataBlockStartAddress=0, object_type=0,
      object_index=OID_KNXNETIP_PARAMETER, PID= PID_ADDITIONAL_INDIVIDUAL_ADDRESSES,
      start_index=i, noElements=1, data= AIAs[i])
Endif
Endfor

      /* Restart the device if necessary. */
If (RestartRequired)
      DMP_Restart_RCo()
Else
      DMP_Disconnect_RCo()
Endif
```

2.6 Profile definition

2.6.1 Goal

🗎 *This clause is not intended for integration in the KNX Specifications.*

For the implementation of System B on KNX IP as a KNX medium, a new Profile for this device model has to be created. This Profile will be based on the existing descriptions for System B and replace the Data Link Layer and Physical Layer part as done for mask 5705.

A Mask Version including the medium identifier for KNX IP and based on the existing mask 07B0h Profile will be defined in this Application Note. The Profile of this new mask 57B0h shall be identical to the Profile 07B0h introduced in AN057 “System B” with the addition of the KNXnet/IP specific Properties and procedures necessary for KNXnet/IP Routing protocol access.

This Profile describes the features of a device necessary for operation. The aim is to guarantee runtime Interworking between all devices in the system. The main components for this objective are the support of group oriented (multicast) communication.

In this document, the following legend is used.

Symbol	Description
M	Mandatory
C^n	Conditions are specified under note “n”
O	Optional
X	not allowed
n/a	not applicable
?	not yet defined (editorial indication)

2.6.2 Common Profile

KNX IP devices with Mask Version 57B0h shall comply with the common Profile requirements for “all end devices”.

2.6.3 Specific parts

KNX IP devices with Mask Version 57B0h shall comply with the specific Profile requirements for “System B” devices in general and mask 07B0h devices in particular.

2.6.4 Medium dependent layers

This Profile describes the requirements on a device in order to guarantee compliance with one of the standardised communication media of the system. Compliance with one of these Profiles is a prerequisite for both runtime and configuration Interworking.

2.6.4.1 KNX IP medium dependent layers

2.6.4.1.1 Overview

Feature		Mask 57B0h
1	Physical Layer	M
2	Data Link Layer	M

2.6.4.1.2 Physical Layer

The Physical Layer services for the operation of KNX IP devices are described in [03].

2.6.4.1.3 Data Link Layer

The Data Link Layer services for the operation of KNX IP devices are described in [03].

2.6.5 Configuration and Management (S-Mode Server)

2.6.5.1 Goal

These Profiles describe the requirements on an S-Mode device that are relevant for configuration as a Management Server accessed only via the bus. The objective is to guarantee Interworking with the configuration tool (ETS).

2.6.5.2 Communication

		Mask 57B0h
1	TL - broadcast	M
2	TL - connection oriented	M
3	TL - connection oriented minimal	C [1]
4	TL - connectionless	M

[1] Mandatory for the additional Individual Address, if KNXnet/IP Tunnelling is implemented.

2.6.5.2.1 TL - connection oriented

• Mask 57Bh0	
Specification	**Test**
[04] All features of the following clauses are mandatory except for the coding of the internal service primitives. - §1.6 “Point-to-Point, Connection-Oriented Communication Mode” - §2 “TPDU” - §3.7 “T_Connect service” - §3.8 “T_Disconnect service” - §3.9 “T_Data_Connected service” - §4 “Parameters of Transport Layer” - §5.1 “States” - §5.2 “Actions” - §5.3.3 “Style 3”	[13]

2.6.5.3 Device Management

In this clause all general requirements on a device concerning the mechanisms used for access by the Management Client are described.

Feature	**Mask 57B0h**
1. Direct Memory Access	M
2. DMA on User Memory	M
3. Line Coupler services	n/a
4. Verify Mode	M
5. Interface Object Handling	M
6. Reduced Interface Objects	-
7. Load and Run State Machines	
a) Realisation Type 1	M
b) Realisation Type 2	O
8. Hardware Specific Parameters	n/a
9. RAM cleared	-
10. User EEPROM	M
11. Restart	
a) Connection-oriented	M
b) Connectionless	O
c) Master Reset	M
12. Authorization	M
Nr of access levels	4
13. Memory Control Table	O

🗎 *KNX IP devices with Mask Version 57B0h comply with the device management requirements for "System B" devices in general and mask 07B0h devices in particular. See [02] for the Profile specification of mask 07B0h.*

2.6.5.3.1 Connectionless restart

Specification	Test
[08] - §3.7.2 DM_Restart_RCl	

2.6.5.4 Device Identification

	Feature	Mask 57B0h
1	Device Descriptor Service	
2	Connection oriented	M
3	Connection less	M
4	Device Descriptor Type 0	M
5	Device Descriptor Type 2	O
6	Identification of Hardware[3]	M
7	Identification of Application	M

🗎 *KNX IP devices with Mask Version 57B0h comply with the device identification requirements for "System B" devices in general and mask 07B0h devices in particular. See [02] for the Profile specification of mask 07B0h.*

2.6.5.5 Device Individualisation

	Feature	Mask 57B0h
1	Programming Mode	
	1.a Connection oriented	M
	1.b Connection less	O
2	KNX Serial Number	
	2.a Client initiated	M
	2.b Server initiated	O
3	Domain Address Assignment	n/a
4	Local Assignment	n/a
5	Distributed Address Assignment	n/a

[3] The Property PID_HARDWARE_TYPE (PID = 78) in the Device Object shall identify the hardware.

2.6.5.5.1 connection oriented

a) Realisation Type 1 - Property based • Mask 57B0h	
Specification	**Test**
[08] - §2.2 “NM_IndividualAddress_Read” - §2.3 “NM_IndividualAddress_Write” [4)]	
Programming Mode Control • via HMI: device selection and indication of Programming Mode • via bus: [07] - §4.3.5 “PID_PROGMODE” [08] - §3.22.2 “DMP_InterfaceObject-Write_R” - §3.23.2 “DMP_InterfaceObject-Verify_R” - §3.24.2 “DMP_InterfaceObject-Read_R”	

🗎 *KNX IP devices with Mask Version 57B0h shall comply with the device individualisation requirements for “System B” devices in general and mask 07B0h devices in particular. See [02] for the Profile specification of mask 07B0h.*

2.6.5.6 Device Linking

	Mask 57B0h
1 Address Table	M
2 Association Table	M
3 Routing Table	n/a
4 Linking via Properties	n/a
5 Direct Link	n/a

2.6.5.6.1 Group Address Table

a) Group Address Table – Realisation Type 7 • mask 57B0h	
Specification	**Test**
[07] - §4.9.7 “Group Address Table – Realisation Type 7”	

[4)] Implies connection-oriented TL and Application Layer services for accessing the Device Descriptor.

2.6.5.6.2 Association Table

🗎 *EDITOR NOTE The Realisation Type is unclear from the working document. It refers to a "Realisation Type 7", which does not exist.*

a) Group Object Association Table – Realisation Type 6 • System 300 • System B • mask 57B0h?	
Specification	**Test**
[07] - §4.10.4 "Group Object Association Table – Realisation Type 6"	
b) Group Address Table – Realisation Type 8 • Mask 5705h • mask 57B0h?	
Specification	**Test**
[07] - §4.10.5 "Group Object Association Table – Realisation Type 8"	

🗎 *KNX IP devices with Mask Version 57B0h comply with the device linking requirements for "System B" devices in general and mask 07B0h devices in particular. See [02] for the Profile specification of mask 07B0h.*

2.6.5.7 Application Handling

Feature	**Mask 57B0h**
1 Group Object Table	M
2 Application Program & Parameters	O
3 Application Specific Parameters	M
4 Application Programming Interface	O
5 Functional Parameters	n/a

2.6.5.7.1 Group Object Table

a) Group Object Table – Realisation Type 7 • System BMask 57B0h	
Specification	**Test**
[07] - §4.15.5 "Group Object Table – Realisation Type 7"	

🗎 *KNX IP devices with Mask Version 57B0h comply with the application handling requirements for "System B" devices in general and mask 07B0h devices in particular. See [02] for the Profile specification of mask 07B0h.*

2.6.6 KNXnet/IP

2.6.6.1 IP Protocols

		Mask 57B0h
1	ARP	M
2	RARP	O
3	Support of fixed IP address	M
4	Auto Configuration	
a	BOOTP client	M
b	DHCP client	M
c	Auto-IP	O
5	UDP	M
6	TCP	O
7	ICMP	M
8	IGMP	M
1	BOOTP/DHCP: Either one shall be implemented by a KNX IP device.	

2.6.6.2 KNXnet/IP services families

	Feature	Mask 57B0h
1	Core	M
2	Device Management	
a	Version 1 (cEMI Property Access)	M
b	Version 2 (cEMI Transport Layer)	M
3	Tunnelling	O
4	Routing	X
5	Remote Logging	n/a
6	Remote Configuration	O
7	Object Server	n/a

[1] See [09] for details.

2.6.6.3 KNXnet/IP Management

2.6.6.3.1 Interface Objects

Mask 57B0h devices shall serve all mandatory Interface Objects of System B mask 07B0h devices plus additionally the KNXnet/IP Parameter Object that includes the IP parameters for the KNXnet/IP device’s service container.

	Interface Object	Mask 57B0h
0	Device Object	M
1	Addresstable Object	M
2	Association Table Object	M
3	Applicationprogram Object	M
4	Interfaceprogram Object	O
9	Group Object Table Object	M
11	KNXnet/IP Parameter Object	M

2.6.6.3.2 Properties in the Interface Objects

2.6.6.3.2.1 General (informative)

🗎 *This clause is informative and will not be integrated in the KNX Specifications.*

Mask 57B0h devices shall serve the Properties in the Interface Objects described in the following clause.

Legend

The term “Data” is used for Properties that are accessible via A_PropertyValue_Read and A_PropertyValue_Write.

The term “Local” is used for Properties that are accessible via M_PropRead and M_PropWrite.

See [02] for detailed information.

In the tables the access levels are noted as “read access level”/”write access level”. “m” denotes the read access level, “n” denotes the write access level.

Symbol.	**Description**	
x	It is not allowed to implement this Property as data Property.	
(m/n)	The Property is optional. If it is implemented, then m is the recommened default read access level and n is the recommended default write access level.	
(m/(n))	The Property is optional. If it is implemented, then m is the recommended default read access level. The Property may be writable with the recommended default access level n, but may be read-only as well.	
(m/x)	The Property is optional. If it is implemented then the Property is read-only; the Property may also be writeable, but only with the access levels 0 (system manufacturer) or 1 (product manufacturer). m is the recommended default read access level.	
m/n	The Property is mandatory. m is the recommended default read access level and n is the recommended default write access level.	
m/(n)	The Property is mandatory. The Property may be read-only. If the Property is writeable, then the recommended write access level is n.	
m/x	The Property is mandatory. m is the recommended default read access level. The Property shall be read-only; the Property may also be writeable, but only with the access levels 0 (system manufacturer) or 1 (product manufacturer).	

NOTE 1 The access levels to Interface Objects are specified in [05] clause 3.4.7 “A_Authorize_Request-service”.

NOTE 2 This AN uses the access levels from 0 to 3. For the relation to access levels 0 to 15 as used in certain Profiles, please refer to [06].

2.6.6.3.2.2 Device Object

Nr.	Property		Mask 57B0h
1	PID_OBJECT_TYPE	Data, Local	3/x
2	PID_OBJECT_NAME	Data, Local	(3/3)
8	PID_SERVICE_CONTROL	Data, Local	(3/3)
9	PID_FIRMWARE_REVISION	Data, Local	(3/x)
11	PID_SERIAL_NUMBER	Data, Local	3/x
12	PID_MANUFACTURER_ID	Data, Local	3/x
14	PID_DEVICE_CONTROL	Data, Local	3/3
15	PID_ORDER_INFO	Data, Local	3/x
16	PID_PEI_TYPE	Data, Local	(3/x)
17	PID_PORT_CONFIGURATION	Data, Local	(3/3)
18	PID_POLL_GROUP_SETTINGS	Data, Local	X
19	PID_MANUFACTURER_DATA	Data, Local	(3/x)
21	PID_DESCRIPTION	Data, Local	(3/3)
25	PID_VERSION	Data, Local	3/x
51	PID_ROUTING_COUNT	Data, Local	3/3
52	PID_MAX_RETRY_COUNT	Data, Local	X
53	PID_ERROR_FLAGS	Data, Local	(3/3)
54	PID_PROG_MODE	Data, Local	3/3
55	PID_PRODUCT_ID	Data, Local	(3/x)
56	PID_MAX_APDU_LENGTH	Data, Local	3/x
57	PID_SUBNET_ADDR	Data, Local	3/0
58	PID_DEVICE_ADDR	Data, Local	3/0
62	PID_OBJECT_VALUE	Data, Local	(3/3)
63	PID_OBJECT_LINK	Data, Local	(3/3)
65	PID_PARAMETER	Data, Local	(3/3)
66	PID_OBJECT_ADDRESS	Data, Local	(3/3)
70	PID_DOMAIN_ADDRESS	Data, Local	X
71	PID_IO_LIST	Data, Local	3/0
72	PID_MGT_DESCRIPTOR_01	Data, Local	X

Nr.	Property		Mask 57B0h
73	PID_PL110_PARAM	Data, Local	X
75	PID_RECEIVE_BLOCK_TABLE	Data, Local	X
76	PID_RANDOM_PAUSE_TABLE	Data, Local	X
77	PID_RECEIVE_BLOCK_NR	Data, Local	X
78	PID_HARDWARE_TYPE	Data, Local	(3/3)
79	PID_RETRANSMITTER_NUMBER	Data, Local	X
80	PID_SERIAL_NR_TABLE	Data, Local	X
81	PID_BIBATMASTER_ADDRESS	Data, Local	X
83	PID_DEVICE_DESCRIPTOR	Data, Local	3/x

2.6.6.3.3 Group Address Table Object

Nr.	Property		Mask 57B0h
1	PID_OBJECT_TYPE	Data, Local	3/x
2	PID_OBJECT_NAME	Data, Local	(3/3)
5	PID_LOAD_STATE_CONTROL	Data Local	3/3
7	PID_TABLE_REFERENCE	Data Local	3/x
23	PID_TABLE	Data Local	3/(3)
27	PID_MCB_TABLE	Data Local	3/3
28	PID_ERROR_CODE	Data Local	3/x
53	PID_GROUP_RESPONDER_TABLE	Data Local	X

2.6.6.3.3.1 PID_LOAD_STATE_CONTROL (PID=5)

Load Control	Subtype	Description	Mask 57B0h
00h		No Operation	O
01h		Start Loading	M
02h		Load Completed	M
03h		Additional Load Controls	
	00h	Absolute Code/Data Allocation	n/a
	01h	Absolute Stack Allocation	n/a
	02h	Segment Control Record	n/a
	03h	Task Pointer Record	n/a
	04h	Task Control Record-1	n/a
	05h	Task Control Record-2	n/a
	0A	Relative Allocation	n/a
	0B	Large Relative Allocation	M
04h		Unload	M

The Property PID_LOAD_STATE_CONTROL (PID=5) is also valid for the Association Table Object (2.6.6.3.4), the Group Object Table Object (2.6.6.3.5) and the Application Program Object (2.6.6.3.6).

2.6.6.3.4 Association Table Object

Nr.	Property		Mask 57B0h
1	PID_OBJECT_TYPE	Data, Local	3/x
2	PID_OBJECT_NAME	Data, Local	(3/3)
5	PID_LOAD_STATE_CONTROL	Data, Local	3/3
7	PID_TABLE_REFERENCE	Data, Local	3/x
23	PID_TABLE	Data, Local	3/(3)
27	PID_MCB_TABLE	Data, Local	3/3
28	PID_ERROR_CODE	Data, Local	3/x

2.6.6.3.5 Group Object Table Object

Nr.	Property		Mask 57B0h
1	PID_OBJECT_TYPE	Data, Local	3/x
2	PID_OBJECT_NAME	Data, Local	(3/3)
5	PID_LOAD_STATE_CONTROL	Data, Local	3/3
7	PID_TABLE_REFERENCE	Data, Local	3/x
23	PID_TABLE	Data, Local	3/(3)
27	PID_MCB_TABLE	Data, Local	3/3
28	PID_ERROR_CODE	Data, Local	3/x
51	PID_GRP_OBJTABLE	Data, Local	(3/3)
52	PID_EXT_GRPOBJREFERENCE	Data, Local	(3/3)

2.6.6.3.6 Application Program Object

Nr.	Property		Mask 57B0h
1	PID_OBJECT_TYPE	Data, Local	3/x
2	PID_OBJECT_NAME	Data, Local	(3/3)
5	PID_LOAD_STATE_CONTROL	Data, Local	3/3
6	PID_RUN_STATE_CONTROL	Data, Local	(3/3)
7	PID_TABLE_REFERENCE	Data, Local	3/x
13	PID_PROGRAM_VERSION	Data, Local	3/3
16	PID_PEI_TYPE	Data, Local	3/3
27	PID_MCB_TABLE	Data, Local	3/3
28	PID_ERROR_CODE	Data, Local	3/x
51	PID_PARAM_REFERENCE	Data, Local	(3/x)

System B foresees two Application Programs; this table is valid for the Interface Objects of Application Program 1 and Application Program 2.

2.6.6.3.7 KNXnet/IP Parameter Object

Nr.	Property		Mask 57B0h
1	PID_OBJECT_TYPE	Data, Local	3/x
51	PID_PROJECT_INSTALLATION_ID	Data, Local	3/3
52	PID_KNX_INDIVIDUAL_ADDRESS	Data, Local	3/3
53	PID_ADDITIONAL_INDIVIDUAL_ADDRESSES	Data, Local	(3/3)
54	PID_CURRENT_IP_ASSIGNMENT_METHOD	Data, Local	(3/x)
55	PID_IP_ASSIGNMENT_METHOD	Data, Local	3/3
56	PID_IP_CAPABILITIES	Data, Local	3/1
57	PID_CURRENT_IP_ADDRESS	Data, Local	3/x
58	PID_CURRENT_SUBNET_MASK	Data, Local	3/x
59	PID_CURRENT_DEFAULT_GATEWAY	Data, Local	3/x
60	PID_IP_ADDRESS	Data, Local	3/3
61	PID_SUBNET_MASK	Data, Local	3/3
62	PID_DEFAULT_GATEWAY	Data, Local	3/3
63	PID_DHCP_BOOTP_SERVER	Data, Local	(3/x)
64	PID_MAC_ADDRESS	Data, Local	3/x
65	PID_SYSTEM_SETUP_MULTICAST_ADDRESS	Data, Local	3/x
66	PID_ROUTING_MULTICAST_ADDRESS	Data, Local	3/3
67	PID_TTL	Data, Local	3/3
68	PID_KNXNETIP_DEVICE_CAPABILITIES	Data, Local	3/x
69	PID_KNXNETIP_DEVICE_STATE	Data, Local	(3/x)
70	PID_KNXNETIP_ROUTING_CAPABILITIES	Data, Local	(3/x)
71	PID_PRIORITY_FIFO_ENABLED	Data, Local	x
72	PID_QUEUE_OVERFLOW_TO_IP	Data, Local	(3/x)
73	PID_QUEUE_OVERFLOW_TO_KNX	Data, Local	X
74	PID_MSG_TRANSMIT_TO_IP	Data, Local	(3/x)
75	PID_MSG_TRANSMIT_TO_KNX	Data, Local	X
76 77	PID_FRIENDLY_NAME Reserved	Data, Local	3/3

2.7 Identifiers and discovery

2.7.1 Medium Type code

The medium type code for KNX IP shall be 5.

2.7.2 Device Descriptor Type 0

Device Descriptor Type 0 shall have the value 57B0h for devices with this Profile.

2.7.3 KNXnet/IP discovery

Mask 57B0h devices shall answer to KNXnet/IP discovery requests. Discovery and description response messages shall not show support for the routing services family. The value 20h shall be used as medium type in the device information DIB.

🗎 *DPT_Media in chapter 3/7/2 “Datapoint Types” has to be extended with the new value 20h for KNX IP*

3 Impact and dependencies

3.1 System specification (“Handbook”) dependencies

The new Profiles for mask 57B0h will be integrated into the Volume 6 “Profiles” ([02]).

3.2 Configuration interworking

Mask 57B0h implementations are in configuration interworking compatible with mask 07B0h with the addition of the configuration of the KNXnet/IP Parameter Object.

3.3 Run-time Interworking

Mask 57B0h implementations are in runtime interworking exactly compatible with System B.

3.4 Integration and common tool impact

3.4.1 General requirements

ETS must know of the new Mask Version and the implications on configuration interworking of this type of devices.

ETS shall support merge points in the support of mask 57B0h.

ETS should provide the user with new common dialogs for IP address assignment and configuration of the KNXnet/IP Parameter Object.

ETS must know of the KNX IP medium type in its base data and provide the IP medium as an option for the medium type of Main Lines and Lines in the topology.

ETS must know of the topology constraints with KNX IP lines and prevent the user from configuring misplaced lines.

3.4.2 Device Info

For mask 57B0h ETS shall display the contents of the following Properties and memory locations if available.

General	
Mask Version	Value of Device Descriptor Type 0
Individual Address	Value from Database of Tool Software
Manufacturer Identifier	Device Object / PID_MANUFACTURER_ID
Order Info	Device Object / PID_ORDER_INFO
Serial Number	Device Object / PID_SERIAL_NUMBER
Hardware Type	Device Object / PID_HARDWARE_TYPE
Program Mode	Device Object / PID_PROGMODE
Firmware Revision	Device Object / PID_FIRMWARE_REVISION
Application Program	
Program	Value from Database of Tool Software
Device Type	Application Program Object / PID_PROGRAM_VERSION
Version	Application Program Object / PID_PROGRAM_VERSION
Load State	Application Program Object / PID_LOAD_STATE_CONTROL
Run State	Application Program Object / PID_RUN_STATE_CONTROL
Application Program 2	
Program	Value from Database of Tool Software
Device Type	Interfaceprogram Object / PID_PROGRAM_VERSION
Version	Interfaceprogram Object / PID_PROGRAM_VERSION
Load State	Interfaceprogram Object / PID_LOAD_STATE_CONTROL
Run State	Interfaceprogram Object / PID_RUN_STATE_CONTROL
Group Communication	
Load State Addresstable	Addresstable Object / PID_LOAD_STATE_CONTROL
Load State Assoc.Table	Associationtable Object / PID_LOAD_STATE_CONTROL
Load State GrpObj.Table	Group Object Table Object / PID_LOAD_STATE_CONTROL

KNX IP

Device Name	KNXnet/IP Object / PID_FRIENDLY_NAME
Device Capabilities	KNXnet/IP Object / PID_KNXNETIP_DEVICE_CAPABILITIES
MAC Address	KNXnet/IP Object / PID_MAC_ADDRESS
Current IP Assignment	KNXnet/IP Object / PID_CURRENT_IP_ASSIGNMENT_METHOD
IP Address	KNXnet/IP Object / PID_CURRENT_IP_ADDRESS
Subnet Mask	KNXnet/IP Object / PID_CURRENT_SUBNET_MASK
Default Gateway	KNXnet/IP Object / PID_CURRENT_DEFAULT_GATEWAY
DHCP Server[1]	KNXnet/IP Object / PID_DHCP_BOOTP_SERVER
IP Assignment Method[2]	KNXnet/IP Object / PID_IP_ASSIGNMENT_METHOD
Manual IP Address[3]	KNXnet/IP Object / PID_IP_ADDRESS
Manual Subnet Mask[3]	KNXnet/IP Object / PID_SUBNET_MASK
Manual Default Gateway[3]	KNXnet/IP Object / PID_DEFAULT_GATEWAY
Routing Multicast Addr.	KNXnet/IP Object / PID_ROUTING_MULTICAST_ADDRESS
Multicast TTL	KNXnet/IP Object / PID_TTL
Messages to IP	KNXnet/IP Object / PID_MSG_TRANSMIT_TO_IP
Additional Ind. Addr.[4]	KNXnet/IP Object / PID_ADDITIONAL_INDIVIDUAL_ADDRESSES

[1] Display only if current IP address assignment method is DHCP/BOOT
[2] Display only if different from current IP address assignment method
[3] Display only if assignment method is manual and IP parameters differ from current ones
[4] Display only if KNXnet/IP Tunnelling is implemented

3.5 Risks and compatibility issues

None.

4 Test procedures

4.1 Test with EITT

4.1.1 General requirements and overview

Mask 57B0h devices shall be submitted to the same tests as “System B“ devices, respectively mask 07B0h. See [02].

Exception: Data Link Layer Test

Only the following test cases from the Data Link Layer Test are applicable to mask 57B0h devices.

Test case number	Test case describtion	Mask 57B0h
10.2.1.1	Valid and Invalid Control field – Receive	M [5)]
10.2.1.2	Priorities: Send	M
10.2.2	Source address	M
10.2.3.1	Individual Address	M
10.2.3.2	Unused Individual Address	M
10.2.3.3	Used Group Address	M
10.2.3.4	Unused Group Address	M
10.2.3.5	Send telegrams	M
10.2.4.1	Info Length: Send	M
10.2.4.2	Info Length: Receive	M

[5)] Necessary extension of EITT.

4.1.2 Test setup

The following is required for the setup:

- PC with EITT Version 3.1 (Using KNXnet/IP Routing Interface) [6].
- KNX IP Device mask 57B0h, DUT
- 10/100MBit (Managed) Switch
- Optional: PC with Wireshark

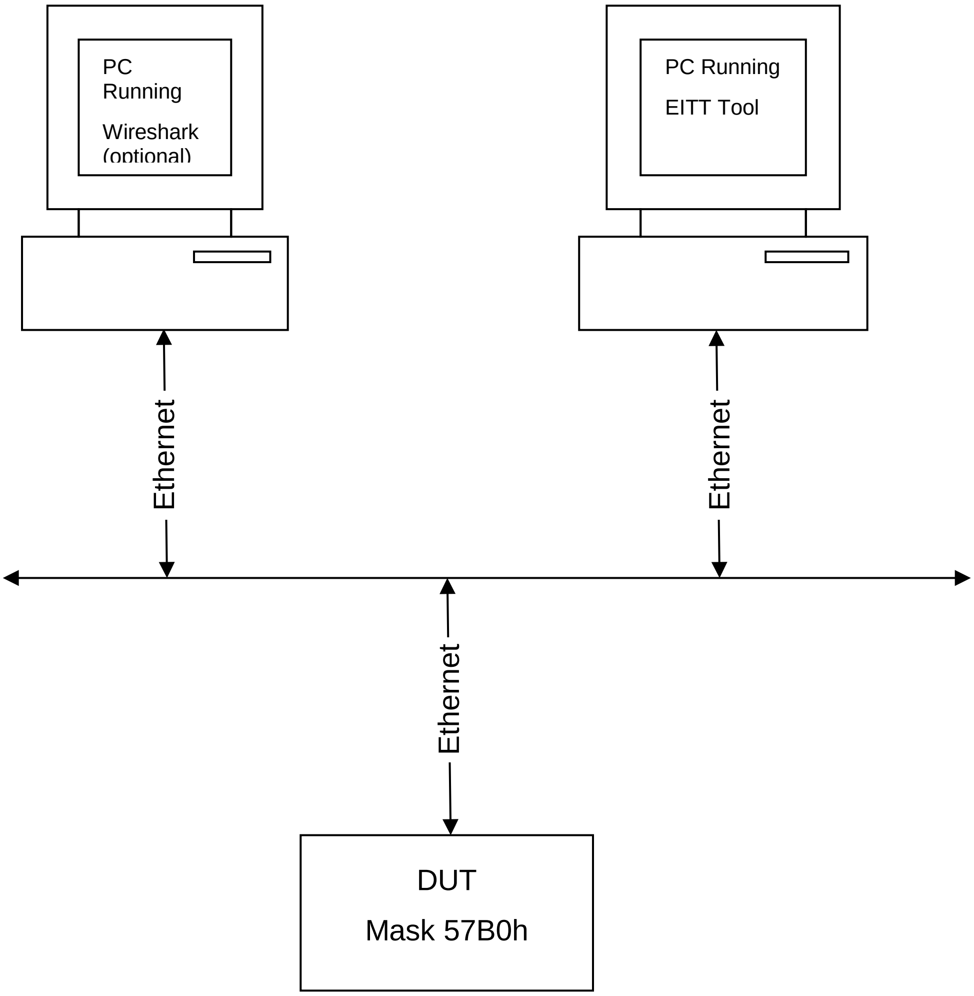

Figure 1 - Test setup for EITT test

4.2 Validation Tool

4.2.1 Test cases

The following table defines the Validation Tool test cases that are or applicable for mask 57B0h devices. All other test cases are mandatory for mask 57B0h.

[6] Transport Layer tests require 2 KNXnet/IP routing connections. This is not testable with EITT 3.1 today. An update of EITT is requested.

Section	Test case numbe r	Test case describtion	Mask 57B0h
Core (Unspecific)	10101	Undefined Discovery Code	M
Core (Unspecific)	10204	Undefined Control Code	M
Core (Search Request)	10201	Standard Case	M
Core (Search Request)	10202	Invalid Version	M
Core (Search Request)	10203	Invalid Header Size	M
Core (Search Request)	10205	Incomplete Message	M
Core (Search Request)	10206	Oversized Message	M
Core (Description Request)	10301	Standard Case	M
Core (Connect Request)	10401	Standard Case	M
Core (Connect Request)	10402	Invalid Connection Type	M
Core (Connectionstate Request)	10501	Standard Case	M
Core (Connectionstate Request)	10502	Invalid Channel	M
Core (Connectionstate Request)	10503	Bus connection interrupted	n/a
Core (Connectionstate Request)	10802	Time Out	M
Core (Disconnect Reques)	10601	Standard Case	M
Core (Disconnect Reques)	10602	Invalid Channel	M
Device Management (Connect Request)	20101	Multiplicity	M
Device Management (Configuration Request)	20203	Standard Case	M
Device Management (Configuration Request)	20204	Read mandatory device Properties	n/a
Device Management (Configuration Request)	20205	Write to read-only device Property	M
Device Management (Configuration Request)	20206	Read nonexisting device Property	M
Device Management (Configuration Request)	20207	Get/set programming mode of device	M
Device Management (Configuration Request)	20208	Get/ set programming mode by memory access	n/a
Device Management (Configuration Request)	20209	Change Individual Address	n/a

Section	Test case numbe r	Test case describtion	Mask 57B0h
Device Management (Configuration Request)	20210	Change Individual Address by IP Property	M
Device Management (Configuration Request)	20201	Invalid Endpoint	M
Device Management (Configuration Request)	20202	Unconnected endpoint	M
Device Management (Configuration Request)	20211	Repeat and timeout after missing ACK	M
Device Management (Configuration Request)	20212	Bus connection interrupted	n/a
Tunneling (Connect Request)	30101	Standard Case	O
Tunneling (Connect Request)	30102	Multiplicity	O
Tunneling (Connect Request)	30105	cEMI Raw Mode	O
Tunneling (Connect Request)	30107	KNX Busmonitor Mode	O
Tunneling (Connect Request)	30108	Invalid KNX layer code	O
Tunneling (Connect Request)	30204	Standard Case Tunneling to KNX	O
Tunneling (Connect Request)	30205	Standard Case Tunneling from KNX	O
Tunneling (Connect Request)	30206	Not increased Sequence counter	O
Tunneling (Connect Request)	30207	Sequence counter increased by two	O
Tunneling (Connect Request)	30208	Standard Case Tunneling from KNX Busmonitor	O
Tunneling (Connect Request)	30209	Standard Case Tunneling from KNX Raw Mode	O
Tunneling (Connect Request)	30210	Repeat and timeout after missing ACK	O
Tunneling (Connect Request)	30211	Broadcast telegram tunneled to KNX	O
Tunneling (Connect Request)	30212	Broadcast telegram tunneled from KNX	O
Tunneling (Connect Request)	30213	Point-to-point telegram tunneled to KNX and back	O
Tunneling (Connect Request)	30215	Group Address telegram tunneled to KNX	O
Tunneling (Connect Request)	30216	Group Address telegram tunneled from KNX	O
Tunneling (Connect Request)	30110	Tunnel Addresses Standard Case	O
Tunneling (Connect Request)	30111	Tunnel Addresses Uniqueness	O

Section	Test case number	Test case describtion	Mask 57B0h
Tunneling (Connect Request)	30112	Tunnel Addresses Assignment Method	O
Tunneling (Connect Request)	30301	Standard Case NAT Compatible Tunneling to KNX	O
Tunneling (Connect Request)	30302	NAT Compatible Tunneling to KNX with IP address set	O
Tunneling (Connect Request)	30303	NAT Compatible Tunneling to KNX with port number set	O
Tunneling (Connect Request)	30304	Standard Case NAT Compatible Tunneling from KNX	O
Tunneling (Connect Request)	30305	NAT Compatible Tunneling from KNX with IP address set	O
Tunneling (Connect Request)	30306	NAT Compatible Tunneling from KNX with port number set	O
Routing (Routing Indication)	40201	Standard Case 1	n/a
Routing (Routing Indication)	40202	Standard Case 2	n/a
Routing (Routing Indication)	40203	Changed multicast address, Case 1	n/a
Routing (Routing Indication)	40204	Changed multicast address, Case 2	n/a
Routing (Routing Indication)	40205	Property PID_MSG_TRANSMIT_TO_KNX	n/a
Routing (Routing Indication)	40206	Property PID_MSG_TRANSMIT_TO_IP	n/a
Routing (Routing Indication)	30201	Mixed Case 1	n/a
Routing (Routing Indication)	30202	Mixed Case 2	n/a
Routing (Routing Indication)	30203	Mixed Case 3	n/a
Routing (Routing Lost Message)	40101	Router overflow	n/a
Routing (Routing Lost Message)	40102	Continuous overflow	n/a

4.2.2 Test setup

The following is required for the setup:

- KNX IP Device mask 57B0h, DUT
- 10/100MBit (Managed) Switch
- PC with Validation Tool
- Optional: PC with Wireshark

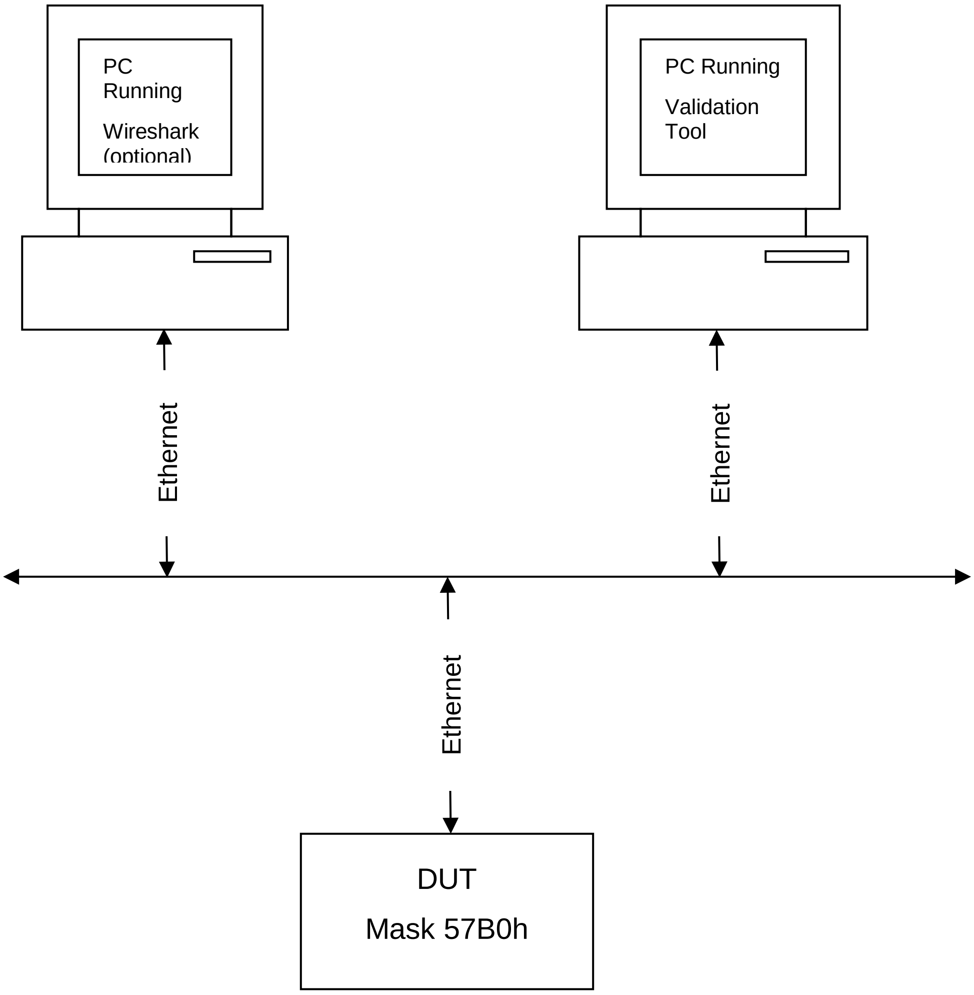

Figure 2 - Test setup for Validation Tool

4.3 Test Application Interface

The Application Interface shall be tested through a manufacturer specific test.