

PERSONAL PRIVACY MANAGEMENT IN THE KNX-AWARE HOME

Luis A. Ramon Surutusa¹, Susana Alcalde Bagüés^{1,2}, Carlos Fernández-Valdivielso¹ and Ignacio R. Matías¹

Public University of Navarra¹
Department of Electrical and Electronic Engineering
Navarra, Spain

Siemens AG, Corporate Technology²
Munich, Germany

November 2008

Abstract In this work, we introduce the UCPF's user interface called *Privacy Manager*. The User Centric Privacy Framework (UCPF) was introduced in the International KNX Conference 2006 [1] as a novel mechanism to protect privacy information of the inhabitants of the KNX-Aware House. The Privacy Manager incorporates a set of applications designed especially to meet the requirement of being *user friendliness*, and make privacy management an attractive task for the inhabitants of the KNX home. Our first prototype provides inhabitants with the means to: i) customize permissions on the disclosure of their personal data, ii) control interactions with pull and push context-aware services, iii) negotiate obligations on the usage of the data once transmitted, iv) be aware of privacy related issues such as granted and denied permissions, v) apply alternative privacy mechanisms to access control as *white lying* and *obfuscation*, vi) adhere to enterprise privacy policies base on a contractual relationship with an enterprise or organization. Thus, the future of the KNX standard should be related with means to provide privacy in the home environment, and therefore, with means to allow a user managing his privacy preferences.

1. Introduction

Privacy is a prime concern in today's information society and one of the most challenging topics to consider when designing smart spaces, characterized by its ubiquitous intelligence and personalized services. Therefore, we developed the User-centric Privacy Framework (UCPF) to act as the privacy manager of the

inhabitants of a KNX-Aware House and assist them in interactions with inside-home and outside Context-aware Mobile Services (CAMS).

However, it was still needed a solution to allow users to administrate the UCPF. In general, there is a strong requirement for friendly, easy-to-use interfaces, aimed at non-expert users, to manage personal privacy. Individuals are often not aware of

potential privacy risks and mostly make decisions casually and out of habit when dealing with privacy issues. With the additional problem, that they are already confronted with difficulties and thus struggling to manage their privacy "online". Not many people would disagree with the observation that it would be beneficial or even necessary to empower users to decide on the exchange of personal data on a much simpler, automatic, and fine-grained level than possible today to prevent them from losing their privacy to enterprises and data sellers, in the upcoming era of so-called Pervasive computing.

In this paper, we present our solution, the UCPF's user interface, called *Privacy Manager*. Simplicity and user friendliness were the two key concepts that guided its design. Our first prototype includes six applications: "Customize Permissions", "Customize your Services", "Organization Policies", "White Lying", "Obligations" and "Privacy State". They allow now inhabitants to manage privacy by themselves, avoiding overwhelming them with the troublesome task of creating and administrating their privacy preferences.

2. Scenario

The following example scenario illustrates the type of privacy rules a user could specify with our Privacy Manager interface.

Ivan lives in his KNX-Aware House with his wife Maria and his two daughters. The house is designed for making life more comfortable and secure. The UCPF system has been installed, in the residential gateway, and all the family members can now store their privacy preferences and administrate their own personal privacy.

Ivan travels frequently but he does not trouble anymore with planning his trips. Instead, he is subscribed to the "Journey Planer" service. Thus, after setting up his personal and the company preferences and keeping up to date his travelling calendar, the "Journey Planer" arranges automatically all his trips. Ivan relies on the UCPF to disclose his calendar appointments filtered. Only those appointments with event type "travel" are revealed.

Ivan has met lot of people, all around the world, and he likes to keep in touch with many of them, networking is important for Ivan's work. Ivan uses one of the known "Friend Finder" services to track his contacts. He likes to get alerts about the people that are in the city. However, Ivan only wants to reveal his position when he is not working and to people that is in the same area of the city as him.

Finally, Ivan uses a "Restaurant Finder" application to request for known restaurants when he is out in a new city, with this application he can automatically see the restaurants registered in the area where he is located. The UCPF always disclose his location to the restaurant finder with accuracy area, without revealing his exact position.

3. Background: UCPF and SeT

A first prototype of the UCPF was developed to be installed on the residential gateway for the Siemens Smart Home Lab [2]. The smart home provides an ideal environment to deploy and test a trusted personal enforcement point, mostly oriented towards a controlled and small number of users. The residential gateway enables access to home-based services from inside and outside the home environment. The incorporation of the UCPF adds privacy control and context brokering as separate functionalities and allows inhabitants to interact with encountered CAMS. Part of its design was also incorporated into the privacy context manager system of the IST project CONNECT [6] and tested during its field trial in Oulu (Finland).

The UCPF seeks to give as much control as possible to its users, in the administration of personal privacy. It works as a privacy broker between user (target), context provider (data collector), service provider (data consumer) and the recipients of data (Subject) as is shown in Figure [1].

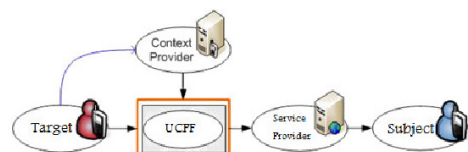


Figure 1 - UCPF role in the information exchange chain.

The UCPF architecture has been designed according to a set of 12 requirements identified to address our main concern; keeping balance between the following three key issues: privacy protection, service usability and user manageability.

The UCPF consists of five core services implemented as bundles within the OSGi framework namely, the Sentry or policy system, the System Manager Interface (SMI), the Context Handler (CH), the Obligation Manager (OM), and the Noise Module (NM). Apart from them, the UCPF incorporates a web registry (Sentry Registry) and a set of applications designed specially to be managed by common users; the *Privacy Manager*.

One of the first things we needed to address during the design of the UCPF was how to collect users' preferences, regarding the use and exchange of personal sensitive information. The best way known to express user preferences is with privacy rules. The table 1 shows the preferences of Ivan, from our scenario, in the form of rule restrictions.

Service	Subject	User context	Subject context	Resource	Transfor.
Journey Planer	-	Event: "travel"	-	Calendar	-
Friend Finder	Group friends	Activity: "no working"	In same city user	Location and Activity	-
Restaurant Finder	-	-	-	Location	Location: Area

Table 1: Rules scenario

After analyzing relevant related work in the area of policy languages, we concluded that there is not a single language, which fulfills all our requirements. The criteria used to evaluate the expressiveness and functionality of policy languages were: i) an ontology-based description; ii) access control functionality; iii) implementation of usage control; iv) support of role-value constraints; v) application of disclosure level control; and vi) distinction of different types of interactions service-user, e.g passive and active role.

We developed the SenTry language (SeT) to meet our requirements and be used in the UCPF. SeT is built on top of the Web Ontology Language (OWL) [11] and the Semantic Web Rule Language (SWRL) [10] as a combination of instances of our policy ontology (SeT ontology) and SWRL rules.

The SeT ontology describes the classes and properties associated with the policy domain in OWL DL using a unique XML namespace. OWL provides considerable expressive power to cover our requirements and flexibility to deal with different data collectors. However, it also has some limitations, which mainly stem from the fact that is not possible to capture relationships between a certain property and another in the domain [12]. We included SWRL rules in the specification of SeT, to reason about OWL individuals, primarily in terms of classes and properties, and overcome the limitations on the OWL language in the definition of role-value constraints [13].

4. GUI design

At this point, the UCPF is a system ready to get users' privacy preferences and start managing personal privacy, by properly distributing users' data to selected services. Now, we need to address the following questions: How users of the UCPF could configure the system?; How could they define who, when, and under which circumstances a service can access to which data?; How could they monitor their privacy state?; and even how could inhabitants negotiate obligations on secondary use, or set up the granularity of information to be disclosed?

The main challenge here was that while the design and specification of the SeT language was done using the editing tools of the Protégé-OWL environment [9], which allows expert users to create new OWL classes, properties and individuals, and together with the SWRLTab the edition and evaluation of SWRL rules. It was clear that for a common user, this is not a convenient environment to use, since it requires previous knowledge of OWL language. Moreover, In general the definition of OWL individuals and SWRL rules only with the provided tools is a non-intuitive and cumbersome task and should be hidden to non-expert users. We need an environment that allows inhabitants to add new rules into the system in a way that the generation of new OWL and SWRL individuals is completely transparent for them.

Nowadays, everybody is used to deal with computers in their daily life: at office,

at home, at cash points, etc. Individuals know how to manage information using graphic interfaces and mainly keyboards and mouses to populate data. A GUI seems to be the best solution to deal with privacy management. Therefore, we developed the *Privacy Manager*, the UCPF GUI, to control and administrate personal privacy.

The idea of building user friendly and easy-to-use interfaces has been followed by designers since the beginning. In fact, there exists an area dedicated to human-computer interaction that, since 1960, studies the requirements needed for designing high quality human-computer interfaces [5]. In [8] the main design principles for GUIs are collected. Below, we listed those followed in the implementation of the *Privacy Manager*.

In advance: A good application design means that the application "thinks in advance"; it is able to advance what a user may want to do next, and display the information and tools he may need. In the same context, it is recommended to show default values, when possible, to guide the user in the process of inserting data.

The *Privacy Manager* has been developed following this criterion, displaying tools only when needed and providing with different degrees of default setups, from policies and rules to simple parameters.

Autonomy: It is a good and recommended practice to enable free exploration of the interface, allowing users to navigate though the different applications and windows, and to discover available options and tools. In other words, users should be able to "click" around without getting into trouble. This makes users feel comfortable with the GUI, and in general reduces initial learning.

In the *Privacy Manager* we keep always visible the logout and home button, to allow user to go back or exit the application at any moment. The application also is provided with information and question messages that inform users about the consequences of pressing any button.

Consistency: This concept is related to the idea that two different things must be done in two different ways; while the same

thing, e.g. exit an application, must follow a similar way in different windows. It is important to keep a similar feeling to avoid pushing user to figure out how realize something that they already know.

In the *Privacy Manager* we implemented known features such as switching from one field to other with the *Tab* key, accepting changes by pressing *Enter*, accessing the menu always on the top left hand side of the window, or redirecting to home by pressing the icon "house", etc.

User effectiveness: An ultimate goal of any user interface it is to improve the performance of its users in the process of realizing a particular task. It is especially important that an application avoids losing information from its users, due to potential user mistakes, or unavoidable failures such as electrical shutdowns.

In the *Privacy Manager* there is not real danger of losing important information. The process of adding a new rule into the system, involves a maximum of 6 quick and consecutive steps. As the rule is saved at the end, a user could only lose the information of that last rule, which is easy to check and solve. Furthermore, useful tools for speeding up tasks have been included; a user is able to establish multiple preferences in one step, just by selecting some extra options rather than repeating tasks once and again.

Latency reduction: Other good design principle is to have heavy process running in the background, e.g., in a separate thread. The idea is to keep the interface always ready to interact with its users. Otherwise, a user could get frustrated and decide not to use the application anymore, if it is constantly busy resolving internal task.

In those situations where we cannot avoid that the application is busy finishing a task. If the task takes between half second and two seconds, it is recommended to use an animated pointer. On the other hand, for those tasks that take longer than two seconds, the best solution is to add a progress bar. Both measures have been implemented in the *Privacy Manager*.

Initial learning: The easier to use an application the better to gain users.

Nowadays, there exist lots of alternative applications for performing the same task. Thus, users are ready to decline a GUI just if they feel tired or frustrated during a first try. Although there is not an alternative solution to the *Privacy Manager*, it is important always to favour initial learning. Because of that, as part of our outgoing work, we plan to test the usability of the application with a selection of 20 or more potential users, and evaluate the initial learning time of the interface.

However, there exist a trade-off between user-friendliness and flexibility, thus a compromise solution must be found. In the design of the *Privacy Manager* the number of restrictions that a user could apply, in the definition of a rule, was limited to those we thought were easier to apply in real world scenarios. As a result, the use of the interface limits the expressiveness of the policy language to favour user friendliness. Nevertheless, we are aware that due to the novelty of the concept developed, the use of the interface may require some extra time to learn how to manage it.

5. GUI Implementation

The *Privacy Manager* is designed to be used in a familiar environment, by a small and controlled number of users. The six proposed applications are shipped as bundles, hosted on an OSGi framework (in the residential gateway), and connected remotely via SOAP with the offered UCPF services. The first prototype provides a single interface for all the inhabitants of a KNX-home. Part of our ongoing work is to develop a second version of the GUI, implemented as a web application, to favour mobility of its users outside the home environment.

The access screen, shown in Figure [2], is the first window users find after starting the *Privacy Manager*. This window is used to login but also to register a new user into the system.



Figure 2 – Login and registration screen.

Once a user is registered, he or she can access offered applications, just by introducing username and password. Pressing the button “Login” takes a user to the next window, the home window.

The home window, shown in Figure [3], displays six different options to manage personal privacy. Two of them, the “White Lying” and “Organization Policies” are still work in progress. The other four: “Customize Permissions”, “Customize your Services”, “Privacy State”, and “Obligations” are already implemented. Each of the applications is delimited by its own frame, which includes a picture and a brief explanation to help users understand the functionality of the application. In this window, a user can easily discover how to access an application by just moving the mouse around. Then, the selection area, extended to the whole panel, is highlighted with a darker colour. The “home” offers the possibility of logoff and also removing the user from the system by pressing on the “trash” icon.

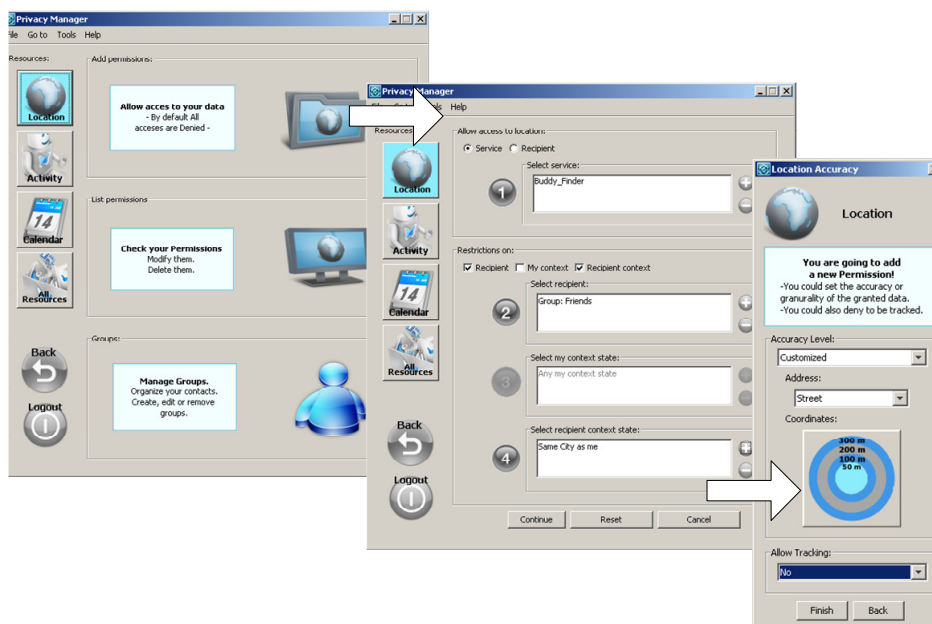


Figure 3 - Home screen.

5.1. Customize your Permissions.

The first selectable option, depicted at the upper left corner of the home screen, is the application that allows users to manage positive permissions.

Figure 4 - Customize Permissions screens.



An especial feature of this application is that there are a default configuration for all users, even if a user does not introduce any rule (permission) at all, the user is still protected, the system returns always a "deny" value which is enforced when there is no a positive permission, added by the user. With this application, shown in Figure [4] a user can add new permissions, and check, update and delete those

previously created. It also provides a special tool to manage contacts groups, which can be used to group individuals into roles.

The process of adding a new permission into the system is the following: first the user has to press a resource button among location, activity, and calendar or select all resources, and press the "folder" icon shown in the window on the left hand side of Figure [4]. Then, a new screen opens, the one indicated with an arrow, which is designed to guide a user through the definition of the different elements of the rule. First of all, the user has to select from a list of registered services to which of them the positive permission should apply, then he can use three different selection screens to add restrictions on the disclosure of that resource to that service. A user can define restrictions: on the recipient, e.g. a particular person, group or organization, on the state of user's context information, e.g. location, activity, calendar event, equal or different to a selected value, and finally on the recipient context state.

By pressing the button "Continue" the next screen is open, which is used now to set up the quality/accuracy of the data to be granted and to allow or disallow a tracking action. With that and after pressing the button "Finish" a new permission is added into the system.



Figure 5 – Editing rules screen.

In our example scenario, this application is used to regulate the exchange of information with “Journey Planer” and “Friend Finder” services. A user can easily set up the service, resource and recipients of the data, and add restrictions base on his context, e.g. my activity is not working, and base on the recipient context e.g., the recipient is in the same city as me. When the UCPF gets a request from these services, it evaluates the constraints added to decide whether or no to grant the action requested.

Once, a new permission is added into the system, the user can check it using the same application. The screen in Figure [5] displays permissions in a table, which can be easily adapted to user needs, dragging and dropping columns and sorting rows by different fields. From this window a user can also edit or delete any permission.

5.2. Customize your Services.

The second option to customize permissions on users' resources is the application “Customize your Services”, depicted at the upper right corner of the home screen. This application simplifies the process of adding new rules. Users just need to configure the accuracy of the resources, in the screen shown in Figure [6], per each of the registered services. These rules do not include other kind of restrictions. If one of these rules is enforced, the service always get grant permission, the rule only controls the transformation,

which sets the desired accuracy of the data to be disclosed.

These rules are not always evaluated; they only apply when a user has triggered the service request before, by actively asking that service. In the implementation of the UCPF we favour to keep balance between privacy protection and service usability, it is fair that if a user requests a service he provides the information needed to offer such service.

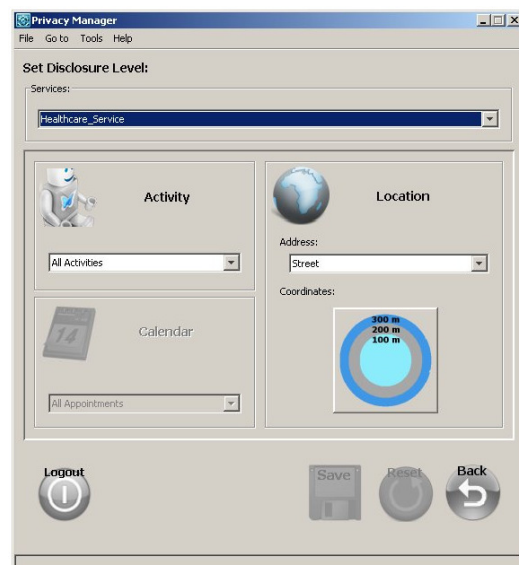


Figure 6 - Customize services screen.

In our scenario, this type of rule is used with the “Restaurant Finder” service, the rule only is evaluated when Ivan sends a request to the service. It that case, the UCPF always discloses Ivan's location with accuracy “area”.

5.3. Privacy State.

The application “Privacy State” was created to meet the system requirement of allowing a user to be aware of his privacy state. The application displays the information related with personal data issues. It the current version, it lists granted and denied permissions sorted by date, and allows to check the rule, which was enforced. In the future, it should provide functionalities to track existing agreements on obligations sets, check the state of unfulfilled obligations, and monitor notifications sent to and received from a service. We would like also to extend this

application by incorporating alarms on potential privacy risk.

5.4. White Lying.

The White lying application is the user front end for setting a white lying state in the WLG component. Following the principle of service usability this tool is only available in passive interactions with services, for further details see [4]. In passive interactions the application shall propose for a selected recipient a virtual context (location-activity) to be disclosed instead the real one. A virtual context is compiled based on existing rules, the location of the recipient, the present and future location of the user, and minimum distance parameters. Although the WLG is already implemented its interface is part of our ongoing work.

5.5. Obligations.

The UCPF is designed to carry on a negotiation process on a set of obligations, to control secondary use of data. That negotiation occurs between the UCPF and a service, before revealing information.

There are five system obligations and eight negotiable obligations predefined in the UCPF, for more details see [7]. System Obligations are negotiated upon registration of a new service and they are not configured by users. They control unwanted disclosures, monitor changes on the service privacy policy and enable the access to collected data. Negotiable obligations are managed by users with the application Obligations, shown in Figure [7]. In that application, obligations are grouped in sets of three obligations, optimal, acceptable and minimum, one for each of the three negotiation rounds allowed.

These three obligations of a set do not need to be different; they may differ only on the event parameter e.g., the timeout to delete data from a repository: 5 days, 1 month or 1 year. The application offers a selection of five predefined sets, namely: 1) "Limiting retention time", 2) "Controlling data accesses". 3) "Controlling data disclosures", 4) "Controlling data disclosures same subject", 5) "Limiting number disclosures per day to same subject". If a user selects one or more of these sets for a registered service, they

need to be negotiated before disclosing any new data to that service.

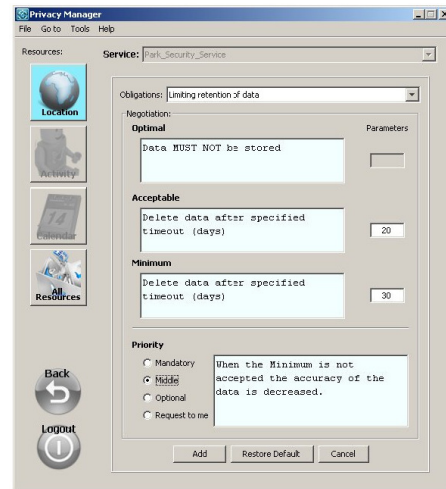


Figure 7 - Add obligation screen.

5.6. Organization Policies.

Organization Policies in our system are policies created by an organization or an enterprise aimed at regulating the exchange of one or more individual's resources, e.g. nurses location during working time. They are used exclusively during *binding interactions*, see [3].

The management of these types of policies entails two complementary processes: i) First, an organization creates a new policy targeted to one or more users of the UCPF. The constraints that an organization policy can apply are the same as the ones used in the definition of user rules. In this case, as an organization is not a user of the UCPF, it is needed an external web application, offered by the Sentry registry, to be used by the organization during the specification of the policy. ii) Second, a user of a UCPF can use now the "Organization Policies" application and download the policy from the Sentry Registry. The application allows users edit and check the constraints included in an organization policy, in the same way that they check a personal rule, in Figure [5], and either accept or reject it. Once an organization policy is accepted, it will be evaluated each time that a request from such organization is sent to the UCPF.

6. Conclusions and outlook.

In this document, we presented our *Privacy Manager* interface, developed to allow non-expert users to interact with the User-Centric Privacy Framework (UCPF) [2] and define and administrate their privacy preferences. The interface offers six different applications to control related aspects of personal privacy mainly aimed at inhabitants of the KNX home.

Part of our ongoing and future work is to complete the implementation of our first prototype, and to realize an extensive field study, with a selection of 20 or more potential users. Thus, evaluating different usability aspects of our *Privacy Manager*. Our main interest is to learn the difficulties that users might face managing personal privacy with our proposed applications.

7. References

- [1] S. Alcalde Bagüés, A. Zeidler, C. Fernández Valdivielso, and I. R. Matías. "Policy-based approach for user privacy protection in the KNX Aware House". International Scientific Conference 2006, Vienna.
- [2] S. Alcalde Bagüés, A. Zeidler, C. Fernández Valdivielso, and I. R. Matías. "Sentry@Home - Leveraging the smart home for privacy in pervasive computing". International Journal of Smart Home. Vol. 1 No. 2., 2007.
- [3] S. Alcalde Bagüés, A. Zeidler, C. Fernández Valdivielso, and I. R. Matías. "Towards personal privacy control". In proceedings of OTM Workshops (2) 2007: 886-895, Springer-Verlag, LNCS..
- [4] S. Alcalde Bagüés, A. Zeidler, C. Fernández Valdivielso, and I. R. Matías. "Disappearing for a while - Using white lies in pervasive computing". In Proceedings of the 6th ACM workshop on Privacy in electronic society 2007, Alexandria, Virginia, USA.
- [5] JCR Licklider and W. Clark, "On-Line Man Computer Communication". In proceedings of the Spring Joint Computer conference, vol. 21, 113-128 San Francisco, California, 1962.
- [6] Susana Alcalde Bagüés, Jelena Mitic and Elisabeth-Anna Emberger. The CONNECT Platform: "An Architecture for Context-Aware Privacy in Pervasive Environments". In the IEEE CS of the International Workshop on Secure and Multimodal Pervasive Environments 2007.
- [7] S. Alcalde Bagüés, J. Mitic, Andreas Zeidler, M. Tejada, C. Fernandez Valdivielso, I. R. Matias. "Obligations: Building a Bridge between Personal and Enterprise Privacy in Pervasive Computing". In proceedings of the 5th International Conference on Trust, Privacy and Security in Digital Business. 2008 Springer-Verlag, LNCS.
- [8] W. Hansen, "User Engineering Principles for Interactive Systems". In proceedings of the Fall Joint Computer Conference, 523-532, AFIPS Press, 1971.
- [9] Protégé SWRL Development Environment FAQ: <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab>, last access October 2008.
- [10] Martin O'Connor et al. "Supporting Rule System Interoperability on the Semantic Web with SWRL". In In Proceedings of the 4th International Conference on The Semantic Web U ISWC 2005. LNCS 3729, 2005.
- [11] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. "Owl web ontology language reference. w3c recommendation" , February 2004.
- [12] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. "OWL rules: A proposal and prototype implementation". J. of Web Semantics, 3(1):23-40, 2005.
- [13] Andrzej Uszok, Jeffrey M. Bradshaw, Matthew Johnson, Renia Jeffers, Austin Tate, Jeff Dalton, and Stuart Aitken. KAoS Policy Management for Semantic Web Services. In IEEE Intelligent Systems, Vol. 19, No. 4, July/August, p. 32-41., 2004.



www.knx.org

A New Tool Suite for KNX System Components

Dr.-Ing. Andreas Kinne
Siemens AG
Industry Sector
Building Technologies
Electrical Installation Technology
Regensburg
kinne.andreas@siemens.com

Abstract

KNX is a distributed system and thus relies on the computation power in each network node. In recent years microcontroller technology has evolved quickly providing more computational power and more memory space to the applications. The BIM M13x family and the KNX chipset are based on a state-of-the-art microcontroller and replace legacy components.

The migration of a product from a legacy BIM to the BIM M13x family is easy from the hardware point of view because the BIM families are pin compatible. Migrating towards the new KNX chipset requires some more hardware effort, as the product PCBs need to be adapted.

In any case the application program has to be re-written; however it is advantageous if the ETS database entry can be re-used for the modified product. The advantage is that the hardware change will not be recognised by the product customer. This allows e.g. to replace a product in an installation without the need to modify the ETS project.

In order for the ETS database entry to remain unchanged, the new application program has to be written in a specific way. Based on a new tool suite it is presented how such an application can be developed in an efficient way.

The new tools enable easy project generation, output file post-processing and download via KNX TP1 into the target device. In addition the tools allow analysis of the bus traffic during configuration and runtime.



Overview

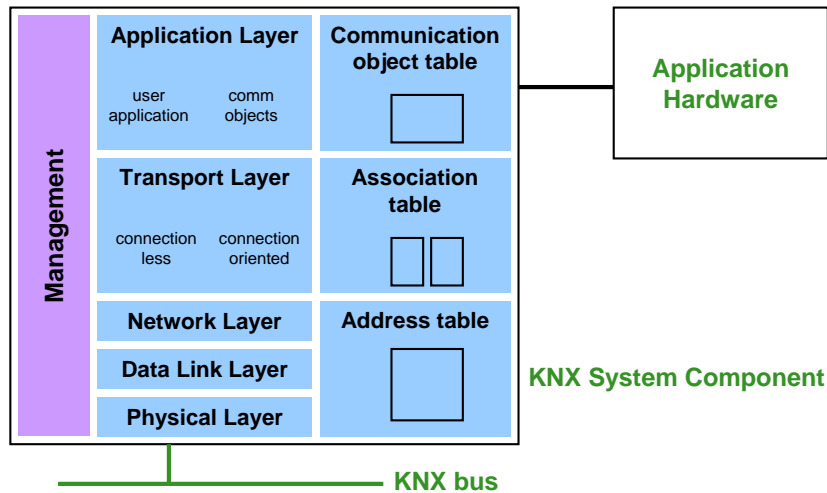
- **KNX System Components**
- **Tools in the Development Process**
- **BIM Replacement**
- **Serial communication**
- **Conclusion**

This presentation will cover the following topics: firstly an overview of available KNX system components will be given. Then the tools used in the process of application code development will be explained. An important item is the replacement of a legacy BIM by a member of the BIM M13x family or by the chipset, while the existing ETS-database entry can be used. Finally the usage of serial protocols is explained

System Components in KNX



■ Design of a KNX product



Dr.-Ing. Andreas Kinne
Siemens AG | BT ET

KNX: The world's only open STANDARD for Home & Building Control

Page No. 3
November 08

A product, which is able to communicate over the KNX bus consists of an application hardware on the one hand. In the example of a switch actuator, the application hardware will typically consist of a number of relays and an electronic circuit to control these relays.

On the other hand a KNX system component is required, which contains a microcontroller, on which the system software runs. A part of the system software is the KNX communication stack, which is implemented according to the OSI model consisting of several OSI layers. The physical layer is responsible for communication on the bus medium (e.g. twisted pair). The data link layer compares addresses of incoming frames with the address table and sends the frame only to the next upper layer in case of a match. The network layer analyses whether a frame is sent connectionless or connection oriented and activates the according part of the transport layer. In the connectionless part of the transport layer the group oriented frames are mapped to the communication objects according to the association table. The user application provides the application functionality of the product and interacts with the communication objects. If a frame is sent, the message is handed down the stack layer by layer in a similar way.

We will see in the following which kinds of KNX system components are available.

System Components in KNX

- **Bus Interface Module (BIM)**
 - pcb module for mounting inside target device
 - almost no external components required (programming button + LED + application hardware)
 - KNX standard hardware + system software

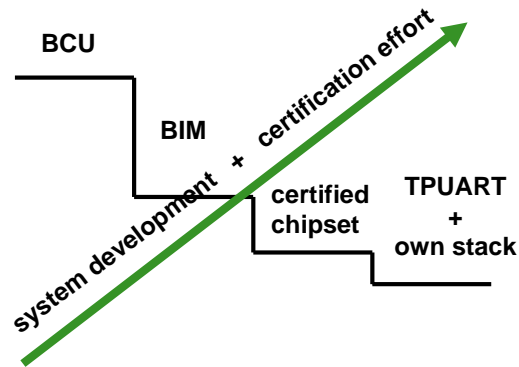
- **Chipset**
 - transceiver ASIC + KNX microcontroller
 - requires additional components
 - KNX system software



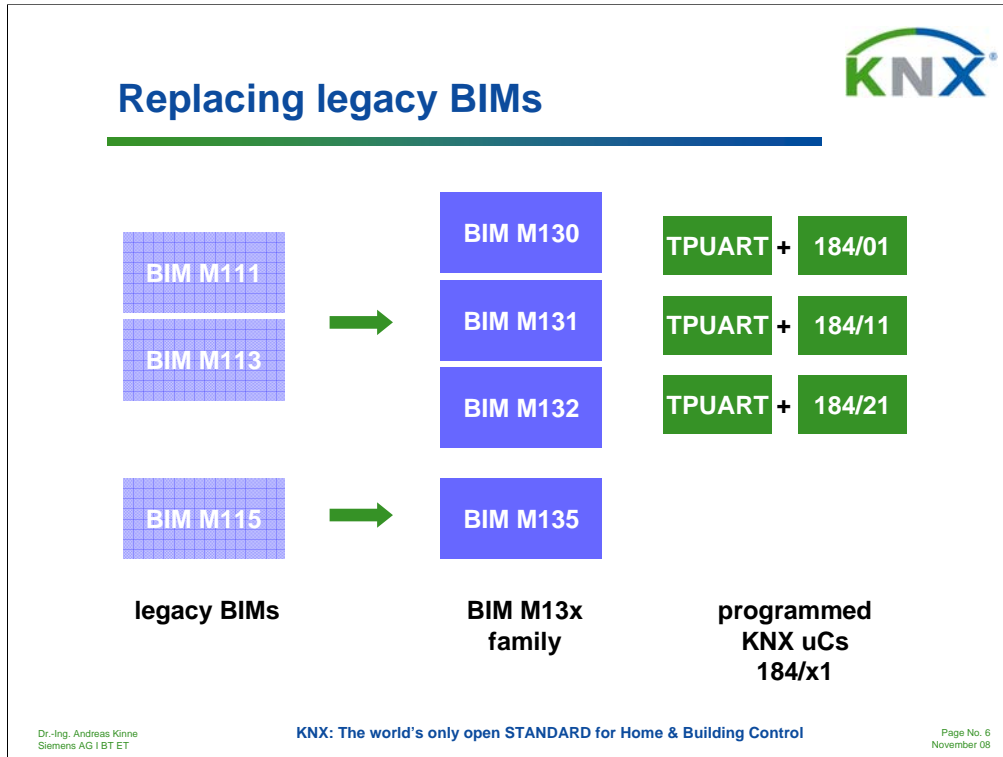
The Bus Interface Module (BIM) is a pcb module designed to build it into a target device. It contains the entire circuit required for KNX bus communication over the twisted pair medium. The only external components are the programming button and the programming LED, which come into addition of the application hardware of the device. The BIM is able to supply the KNX device with power from the bus by the voltages 5V and 20V. The BIM hardware is KNX standardised and contains KNX standardised system software.

The chipset is a bundle of the TPUART transceiver and the KNX microcontroller (KNX uC). For the chipset a number of external components are required. The chipset contains KNX standardised system software.

System Components in KNX



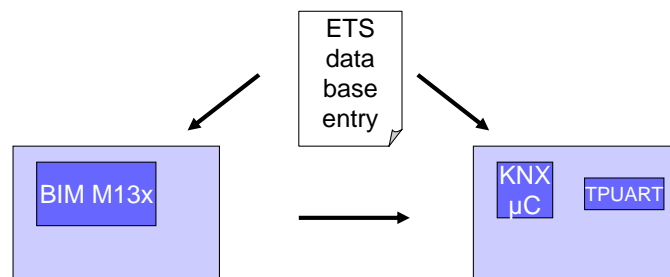
Scaling an application from a bus coupling unit (BCU) over a BIM, a chipset towards the usage of TPUART together with a self-developed KNX stack results in a decrease of component costs along with an increase of system development and KNX certification costs. The costs difference between a BCU and a BIM is bigger than the difference between the BIM and the chipset, which itself is bigger than the small difference between the chipset and the TPUART / own-developed stack solution.



The members of the new BIM M13x family are pin-compatible towards the legacy BIMs M111, M113, M115. The only legacy BIM which cannot be replaced by a BIM M13x, is the M112. The BIM M135 is equivalent to M115 with regard to its extended temperature range. With the additional effort of redesigning the host pcb, the BIM M13x family members can be replaced by solutions using the chipset consisting of the TPUART and the KNX uCs in the following way: 184/01 is the uC of the BIM M130, 184/11 is the uC of M131, and 184/21 is the uC of M132. BIM M135 may be replaced by 184/01, if the additional components are chosen such as to support the extended temperature range.

Reduction of system costs

- Product migration from BIM M13x to chipset (TPUART + KNX uC)
- No change of application software + ETS database entry
- cost reduction of product



In the case of migration from a BIM M13x towards the chipset the application software as well as the ETS database entry both do not need to be modified. The only modification is required on the hardware in order to accommodate the chipset and supporting components.



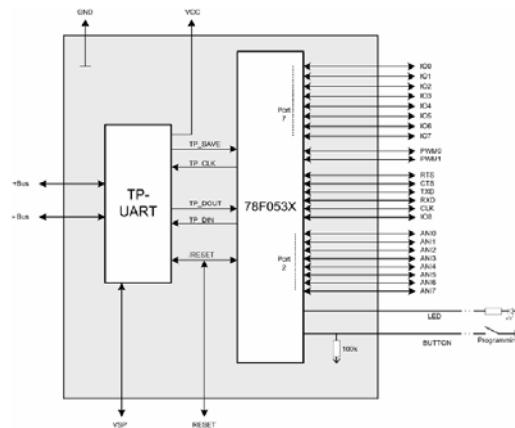
BIMs and KNX processors

BIM M13x family	KNX micro-controller	Memory available for the BIM internal application program	
		Flash	RAM
BIM M130	184/01	8 KByte	200 byte
BIM M131	184/11	16 KByte	1.2 KByte
BIM M132	184/21	48 KByte	5.2 KByte
BIM M135 *	184/01	8 KByte	200 byte

* extended temperature range -25...+70 °C, all other BIMs have -5...+45 °C

This overview displays the BIM M13x members along with the equivalent KNX microcontrollers, the available flash and RAM memory sizes, and the specified temperature range.

Chipset: Hardware



- Available ports: port2, port7, 1 UART, 2 PWM
- power supply: 5V
- Programming of application in target hardware via KNX bus or via programming interface

Dr.-Ing. Andreas Kinne
Siemens AG I BT ET

KNX: The world's only open STANDARD for Home & Building Control

Page No. 9
November 08

The chipset provides the following interfaces to the application hardware: 5V power supply, two ports of IO lines, one hardware UART (universal asynchronous receive / transmit) interface, two PWM (pulse width modulation) lines. The KNX uCs are shipped with the pre-loaded KNX system software.

The application program may be downloaded either via the KNX bus, or via the programming interface. The latter must be attached to the pcb during device manufacturing and provides the advantage of a higher data transfer rate. The advantage of a shorter programming time comes into play especially for larger applications.

Development Process

- **Development Toolpackage contains:**
- **Evaluation Board with BIM M132 functionality**
- **NEC Minicube2 USB programming interface**
- **IAR Embedded Workbench (size limited version)**
- **BIM Tools software**



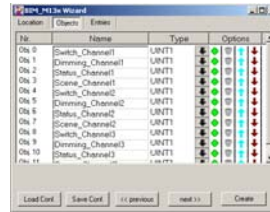
We will now have a closer look at the application program development process. The development tool package contains everything needed to write an application program: the evaluation board, the USB programming interface (“minicube”), a size limited version of the IAR embedded workbench, and the BIM Tools software. The embedded workbench supports application software code up to the maximum memory size of 16kbytes, i.e. applications for the BIM M130 and M131 can be written with this version without any restrictions. Only for application code intended for the BIM M132 the full version of the IAR embedded workbench is required.

BIM Tools Software

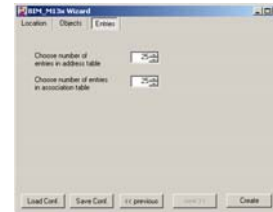
■ Project Generation by BIM Wizard



Project setup



specification of communication objects



specification of table lengths

The BIM Tools is a software designed to support the application development process for the BIM M13x and the chipset. A function in the BIM Tools is the BIM Wizard, which generates the code framework for an application program. The user needs to input some project information, like the path where the project shall be located. Furthermore, the communication objects required in the application need to be specified. Finally, the lengths of the address and association tables need to be given.

Code generation

- **Generated code contains:**
- **ComObjects, values + RAM flags**
- **Address + association tables**
- **code fragments for**
 - main: routine calling the main loop
 - init: called once on startup
 - save: called once at power down
 - unload: called once if there was an unload of the application program
- **Application info block**

Based on these inputs, the BIM Wizard generates code, which contains the communication object values, the RAM flags, the address and association tables, and some code fragments for some routines: the “main” routine contains the main loop, which will run quasi-continuously, cyclically interrupted by the operating system. “init” is the initialisation routine called once at startup of the device. “save” is called once at power down and serves typically to store some variables into non-volatile memory. “unload” is called once, when and unload is performed on the device, typically from a tool like ETS. The usage of “unload” will be explained on the next slide. The application info block contains finally some variables and pointers important for the application program.

Unload



- **Allows defined behaviour upon „unload“ in ETS**
- **Typical example: „unload“ terminates bus interaction, but local functionality remains**
- **Property in Device Object controls, whether application continues after unload**
- **API function IsApplicationLoaded tells the application, if it is loaded**
- **unload routine serves to go to a „safe state“, when unload occurs.**

The following types of behaviour are possible for the application program:

1. The full application program is downloaded by ETS. After unload the application program completely ceases to operate and no application functionality remains in the device.

2. Only part of the application is downloaded by ETS and after unload some basic functionality remains in the device, e.g. local operation of the device.

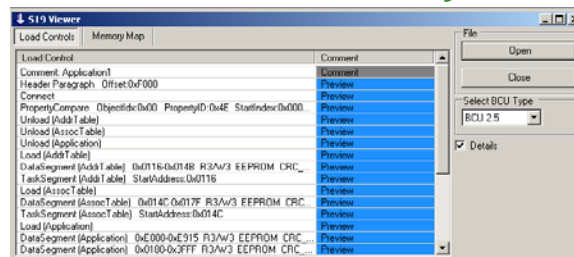
In the second case after unload the application program continues to be called by the operating system. This is managed by an application property in the device interface object.

Furthermore, the application needs to know the load status. Depending on this, the application will perform full functionality if the application is loaded, or reduced functionality, if the application is unloaded. The application finds out about the load status by the API function IsApplicationLoaded.

The unload routine mentioned previously serves to provide a safe state, when unload occurs, e.g. in this way it can be avoided that unload occurs while the relays of a switch actuator are being switched.

Producing downloadable code

- **Load controls: instructions controlling download via KNX bus.**
- **Compiler code contains no load controls**
- **BIM Tools contain S19 modifier to add load controls**
- **S19 viewer allows to analyse S19 files**



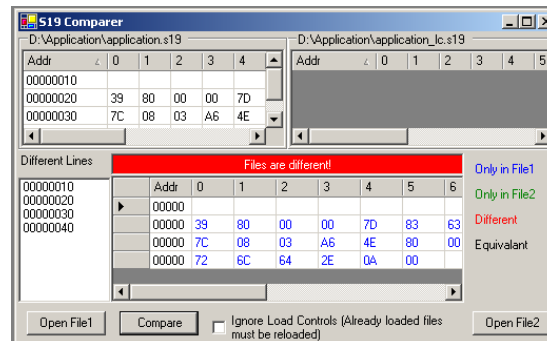
When the application program is ready and compiled, it is not yet possible to download the machine code (S19 file) into the controller via the KNX bus for the following reasons:

1. the compiler produces code located at the real memory addresses, whereas for a download via the bus virtual addresses are required (virtual addresses are based on the Motorola memory map).
2. the load controls are missing (load controls are special load instructions for the device, e.g. operating the load state machines).

The S19 modifier which is part of the BIM Tools maps the addresses to virtual addresses and adds the load controls, generating a downloadable S19 file.

Download / File Comparison

- BIM tools support S19 download
- S19 comparer allows comparison of two s19 files



The BIM Tools provide an S19 downloader which can be used to download an application program into the target device via the KNX bus.

The S19 comparer is a tool to check whether two S19 files are identical, and to display any differences. Also S28 and S37 files can be processed.



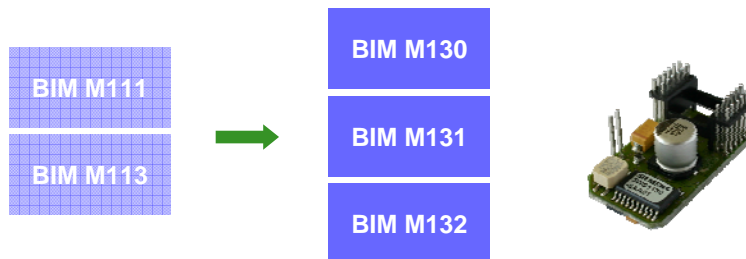
Additional Features

- **EIBcalc calculates address conversion:**
- **Dec / Hex / Bin / ASCII**
- **KNX Data types**
- **Network addresses: individual, group, IP-address**

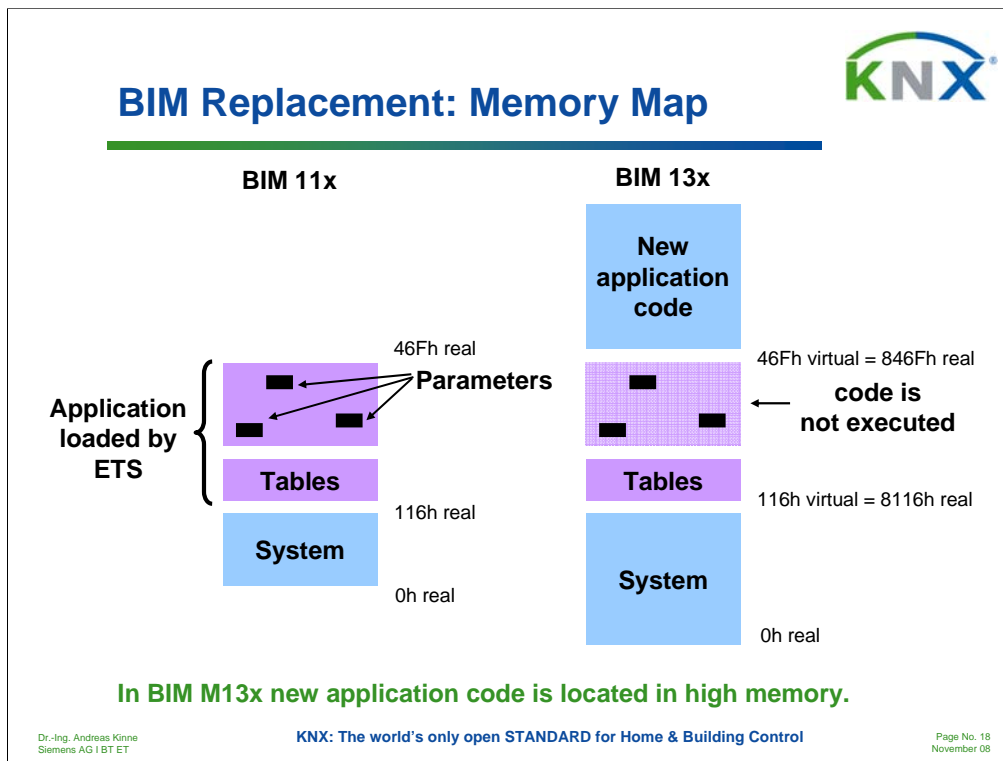
The BIM Tools contain EIBcalc, a useful tool for address conversion. It can convert data between decimal, hex, binary and ASCII values. Furthermore KNX data types can be calculated. Finally the different formats of KNX individual and group addresses and IP addresses can be converted.

BIM Replacement

- BIM M13x is pin compatible to legacy BIMs M111, M113, M115
- allows replacement of BIM in existing device
- Firmware needs to be re-written
- can existing ETS database entry be used?



A legacy BIM M11x may be simply replaced in an application by a member of the new BIM M13x family, because they are pin-compatible (except BIM M112). However, the application program of the BIM M11x is not able to run on a BIM M13x, because the uC has a different architecture, i.e. new firmware needs to be written. In order to make the BIM replacement invisible for the device user, there is a possibility to write the new firmware in a way, so that the existing ETS database entry can be further used.



In the BIM M11x the tables are located at memory addresses from 116h, followed by the application code. The application parameters may be located anywhere within the application code memory area. The highest application code address is 46Fh, which is available in BIM M113.

In BIM M13x, we need to differentiate between real memory addresses, and the virtual addresses, which follow the memory map of a BIM M11x microcontroller from Motorola. Usually the real address can be calculated by adding 8000h to the corresponding virtual address.

When an existing ETS database entry is loaded into a BIM M13x, it will be placed at memory locations starting from 116h virtual = 8116h real. The new code must be placed above the old application, i.e. at real addresses above 846Fh. In order to evaluate the loaded parameters, the new application must access the parameters at their real memory locations.



BIM Replacement

- **Replace BIM M11x by BIM M13x (or chipset)**
- **In manufacturing, load new application code into high memory**
- **Download existing database entry into modified device**
- **Old HC05 code is not executed**
- **New application code re-uses parameters and tables downloaded as part of the old application**

If a BIM M11x is replaced in a device by a BIM M13x or a chipset, the new application code needs to be loaded into the uC during manufacturing.

Step-by-step

- **Generate new application with BIM Wizard with same number and same type of CommObjects**
- **Open project with IAR workbench**
- **Write new application code which acts like existing application**
- **Memory offset for parameters:**
new address = old address + 8000h
- **Check AppID of loaded application**
U._ReadBCU2Adr100(...)
- **Start application only if correct AppID was loaded**

These are a step-by-step instructions how to generate new application code for BIM M13x using an existing ETS database entry:

New code must be generated using the BIM Wizard with the same number and type of communication objects. Using IAR Workbench, the new code must be written in such a way, that it has the same functionality as the existing code.

For parameters the offset of 8000h in order to obtain their real addresses must be observed.

The new application should check, whether the correct ETS database entry was loaded into the device by checking its AppID. This is established by calling U._ReadBCU2Adr100(...). Only if the AppID is correct the application should start operating.



Step-by-step

- **Device Descriptor (Mask Version): BIM supports 2.5 and 2.1**
- **2.1 is required for use with ex. database entry**
- **Device descriptor 2.1 must be set during manufacturing**
- **Adjustment of linker settings allows normal debugging of application code**

ETS only loads an existing BIM M11x database entry into the device, if the device descriptor (mask version) is identical to the one of the old device. Normally BIM M13x has device descriptor 2.5, but for our purpose it has to be adapted to 2.1. This is done by writing to a property during manufacturing.

Only if linker settings are adjusted, debugging of the new application code with the standard debug tools is possible.

Step-by-step



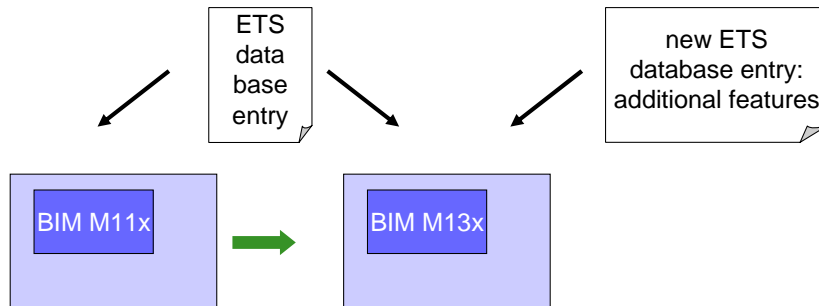
- **KNX registration: new registration required**
- **KNX certification: new application needs to be tested in a KNX accredited test lab by repeating at least one of the interworking/functionality test cases of the original product**



For the adapted device where the BIM has been replaced, new registration and certification at KNX association is necessary. Certification however is not required with the full effort as for a completely new device: only one of the interworking / functionality test cases needs to be repeated at a certified test lab.

Addition of new features

- Device provides same features as before BIM replacement
- New features require new ETS database entry
- New application may overwrite pre-loaded application



In the previous slides the process of replacing a BIM while keeping the functionality and the ETS database entry was explained. Now for such a device it may be required to issue a new ETS database entry which provides additional features in the application. For this purpose new application code can be contained in the ETS database entry overwriting the existing code stored in device memory.



Using serial communication

- **BIM / KNX-controllers contain hardware UART for application use**
- **allows two-controller solutions where required**
- **Handshake (only for legacy applications), FT1.2 protocol are part of BIM/KNX-uC firmware**
- **custom protocol can be easily implemented**

Finally in we will present how serial communication can be used by the application code within a BIM M13x / chipset. The microcontroller provides a hardware UART for application usage, while a second UART is used by the system to operate with TPUART.

Serial communication may be used e.g. for controller / controller communication in a two-controller solution. There are three options for the choice of protocol:

- for legacy solutions the system provides PEI 16 handshake protocol
- the system also provides FT1.2
- a custom protocol can be implemented by the application developer.

Implementation of custom UART protocol

- **In init routine: register receive + transmit interrupts**

```
U._IntRegister(UART0_TxD, INTST0_vect);  
U._IntRegister(UART0_RxD, INTSR0_vect);
```

- **Interrupt handlers for receive and transmit interrupts**

```
void UART0_TxD() // writes next transmit byte into TXS0  
{  
    if (TrmCount > 0)  
    {  
        TrmCount--;  
        TXS0 = TrmBuf[TrmCount];  
    }  
}  
  
void UART0_RxD() // reads next receive byte from RXB0  
{  
    if (RcvCount < 2)  
    {  
        RcvBuf[RcvCount] = RXB0;  
        RcvCount++;  
    }  
}
```

- **In main routine: handling of TrmBuf[] and RcvBuf[]**

For the custom protocol the following needs to be done:

In the init routine the UART receive and transmit interrupts need to be registered. For this purpose the function `U._IntRegister` is called twice, registering the functions `UART0_TxD` and `UART0_RxD` as interrupt handlers.

The interrupt handlers simply write a single byte to the transmit register in case of the transmit handler, and read a single byte from the receive register in case of the receive handler.

Now only remains to handle the buffers `TrmBuf[]` and `RvcBuf[]` in the cyclic part of the main routine, providing the protocol functionality.



Conclusion

- **BIM and chipset (TPUART + KNX uC) enable easy connection of a target device to the KNX bus.**
- **Memory size and temperature range can be selected as appropriate for application**
- **BIM tools support development process**
- **replacement of BIM M11x by BIM M13x or chipset allows re-use of existing ETS database entry.**
- **Serial communication possible with FT1.2 or custom protocol.**

We have seen how the BIM M13x and the chipset provide an easy way to connect a device to the KNX bus. Memory size and temperature range can be selected according to the application requirements.

The BIM Tools have been presented, which is a suite of tools supporting the development process. If a BIM M11x or a legacy chipset (FZE106x + HC05) is replaced by a BIM M13x or by a new chipset (TPUART + KNX uC), new application code needs to be written. If certain rules are observed, and if the new application provides the same functionality and contains the same number and type of communication objects, the ETS database entry of the existing device can be re-used.

Finally it was shortly demonstrated, how serial communication based on FT1.2 or a custom serial protocol can be used.

KNX IP – using IP networks as KNX medium

Dipl.-Ing. Hans-Joachim Langels
Siemens AG
Siemensstraße 10, 93055 Regensburg
Tel: 0941-790-2992
E-Mail: hans-joachim.langels@siemens.com
www.siemens.de/gamma
www.din-bauportal.de/siemens

Abstract

The versatility of KNX is based on its protocol but also on its different media: twisted pair (TP), power line (PL), and radio frequency (RF).

KNXnet/IP as standardized in EN 13321-2 was introduced in 2004 to enable usage of the ubiquitous IP networks for data transmission between KNX subnetworks (→ KNXnet/IP Routing), with supervisory systems (→ KNXnet/IP Tunneling), and with configuration tools (→ KNXnet/IP Device Management).

As an answer to the desire to have KNX devices use existing IP networks as a "medium" for peer-to-peer communication KNX Task Force IP has defined KNX IP, which is based on KNXnet/IP Routing.

This paper summarizes the discussions in the KNX Task Force IP, which led to the current state of definitions for KNX IP, and covers open issues on KNX IP still to be resolved.

Introduction

The ubiquitous presence of the Internet Protocol (IP) has led to the definition of KNXnet/IP. KNXnet/IP provides the means for point-to-point connections (KNXnet/IP Tunneling) for ETS and/or between a supervisory system and a KNX installation.

KNXnet/IP Device Management provides configuration of KNXnet/IP devices through the network effectively reducing the time required for configuration.

It also defines how lines or areas may interconnect using IP networks via KNXnet/IP Routing. Yet, KNXnet/IP Routing in itself only defines how KNXnet/IP routers communicate with each other using IP networks.

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

The desire is to have KNX devices use the IP networks as a "medium" for peer-to-peer communication.

The KNX system topology for twisted pair is hierarchical as depicted in figure 1.

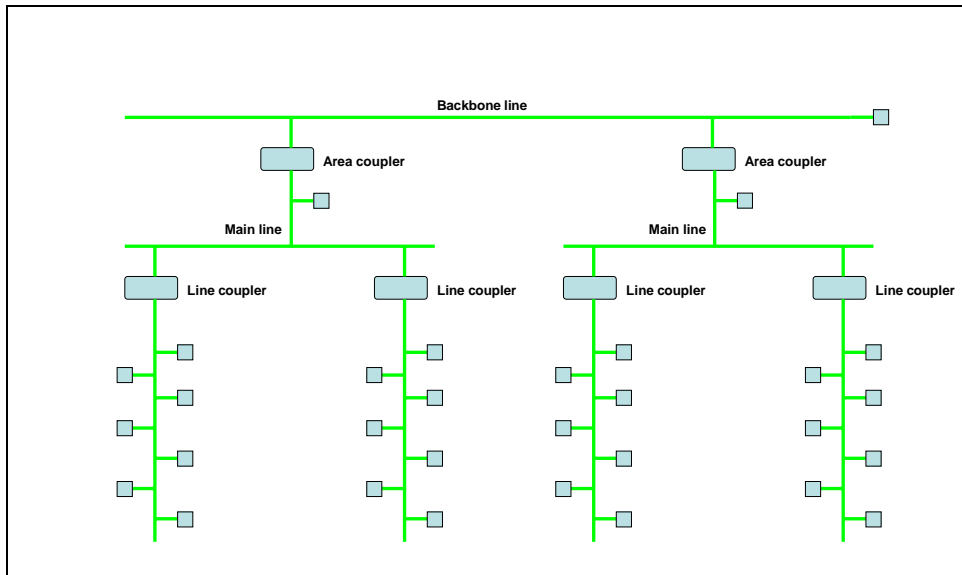


Figure 1

With KNXnet/IP routers replacing the area couplers the topology changes as depicted in figure 2.

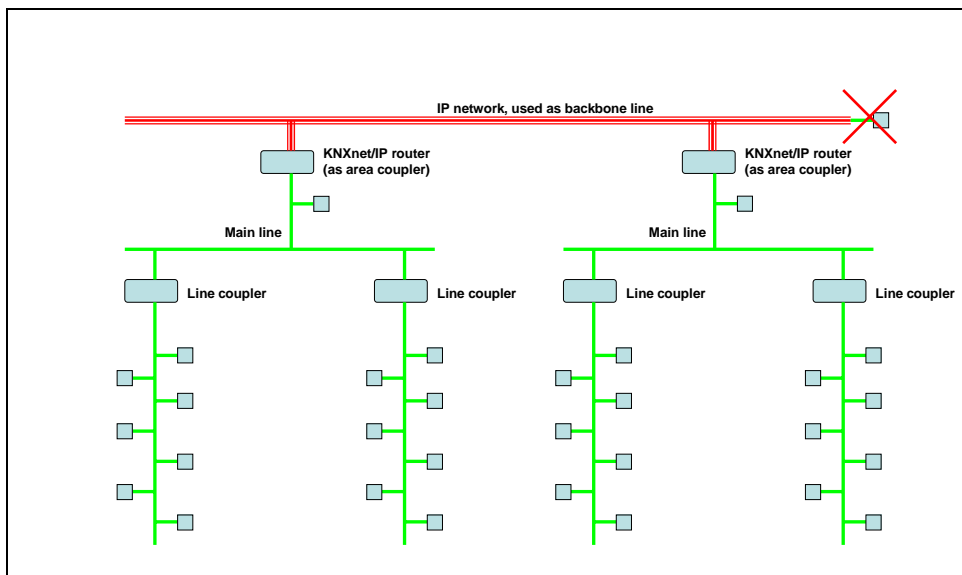


Figure 2

The IP network is used as the backbone for KNX. The communication protocol between KNXnet/IP routers is the KNXnet/IP Routing protocol. A KNXnet/IP router is defined as a device that is on one side connected to a KNX subnetwork (either KNX TP, KNX PL, or KNX RF) and on the other side is connected to an IP network, that is used as the backbone line.

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

Further collapsing the system topology, the line couplers are replaced by KNXnet/IP routers and the backbone couplers are omitted as depicted in figure 3.

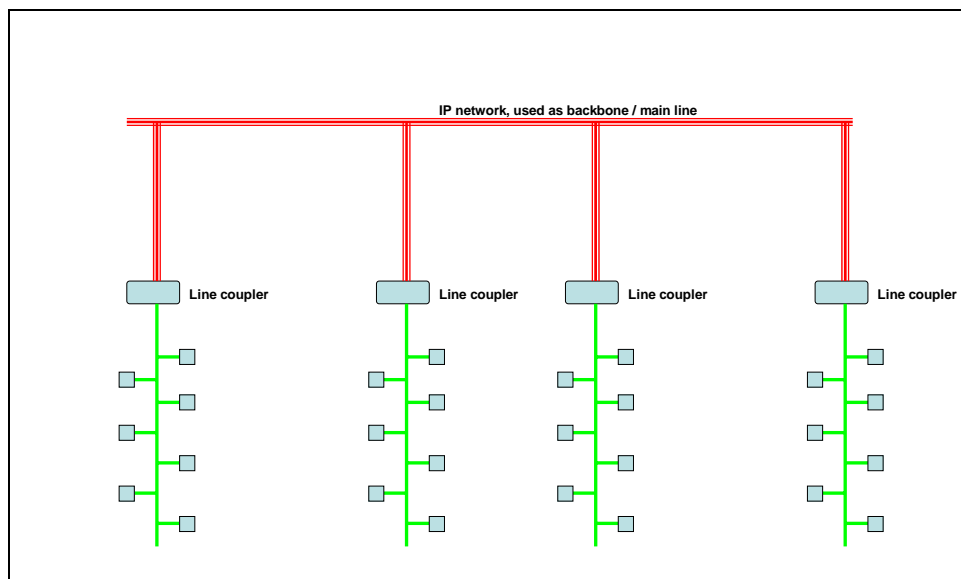


Figure 3

While the traditional KNX TP topology allows for KNX devices to be on the KNX backbone line or the main line the initial KNXnet/IP topology does not define KNX devices other than KNXnet/IP routers or interfaces to be connected to the IP network. It also assumes that KNXnet/IP devices have two interfaces: one to the KNX subnetwork and one to the IP network.

Users and manufacturers expressed the desire to use the IP network as a native medium for KNX. Consequently, these KNX IP devices would only have one physical interface i.e. the interface to the IP network.

KNX Task Force IP Design Objectives

KNX Task Force IP developed the KNXnet/IP protocol following a few design objectives:

- maintain the simplicity and scalability of the KNX system
- use management procedures already supported by ETS
- enable reusing existing stack implementations wherever possible
- add capabilities that enhance the KNX system for advanced applications

Following these design objectives the KNXnet/IP Routing protocol allowed quick development of KNXnet/IP routers that operate like a line coupler. This enables replacing line couplers with KNXnet/IP routers while simplifying the system topology as depicted in Figure 3. The changes to ETS were minimal, which means that installers could start using

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

KNXnet/IP routers quickly. Yet, the KNXnet/IP protocol defines a number of features that may be used by future ETS versions.

KNX IP

With the design principles in mind KNX Task Force IP has worked on how to use existing IP networks as a medium for KNX.

As a first step a few terms need to be defined before going into further details.

- A KNXnet/IP Server is a KNX device that has physical access to a KNX network and implements the KNXnet/IP Server protocol to communicate with KNXnet/IP Client or other KNXnet/IP Servers (in case of routing) on an IP network channel. A KNXnet/IP Server is by design always also a KNX node.
- A KNXnet/IP Client is an application that implements the KNXnet/IP Client protocol to get access to a KNX Subnetwork over an IP network channel.
- A KNXnet/IP Router is a special type of KNXnet/IP device that routes KNX protocol packets between KNX Subnetworks.
- KNX IP is the term used when the Internet Protocol (IP) is utilized as a KNX medium.
- KNX IP device describes a KNX device using the Internet Protocol (IP) as the only KNX medium.
- A KNX IP Router is a special type of KNX IP device that routes KNX protocol packets between its associated KNX IP Subnetwork and other KNX Subnetworks.

Following the design objectives KNX Task Force IP made a fundamental decision to use the existing KNXnet/IP Routing protocol as the basis for KNX IP.

Why?

KNX does not require the source of a KNX telegram to know who is supposed to receive that telegram. Based on KNX group addressing it allows the sender to transmit to an unknown number of recipients, which can be added or removed any time without affecting the sender configuration. Likewise, the number of sources sending to the same group address is not limited either, again without affecting the configuration of the recipients. How can this simplicity and scalability be transposed to IP networks?

In principle, either of these two IP addressing methods could be used: unicast or multicast.

If unicasting were used any KNXnet/IP router or KNX IP device would have to be configured for sending its data to a pre-defined list of other IP devices. This increases the engineering effort (new management procedures) and requires more resources in the devices but also of the network. Sending telegrams via unicast datagrams multiplies the number of datagrams by the number of recipients. Depending on the IP network characteristics and the number of recipients scalability is an issue. Unicast datagrams may be acknowledged, which ensures

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

delivery across the network under different network conditions. This is why KNXnet/IP Tunneling and KNXnet/IP Device Management protocols use IP unicast addressing.

IP multicast offers features similar to KNX group addressing. A multicast datagram is sent onto the network and the same multicast datagram may be received by one or many recipients at the same time. Hence, using multicast allows for scalability and retains the simplicity of the KNX system. This is why KNXnet/IP Routing uses multicast addressing. Following the objective of reusing existing stack implementations KNX Task Force IP decided to use KNXnet/IP Routing as the foundation for KNX IP communication.

Figure 4 shows the evolution of the KNXnet/IP router from the line coupler.

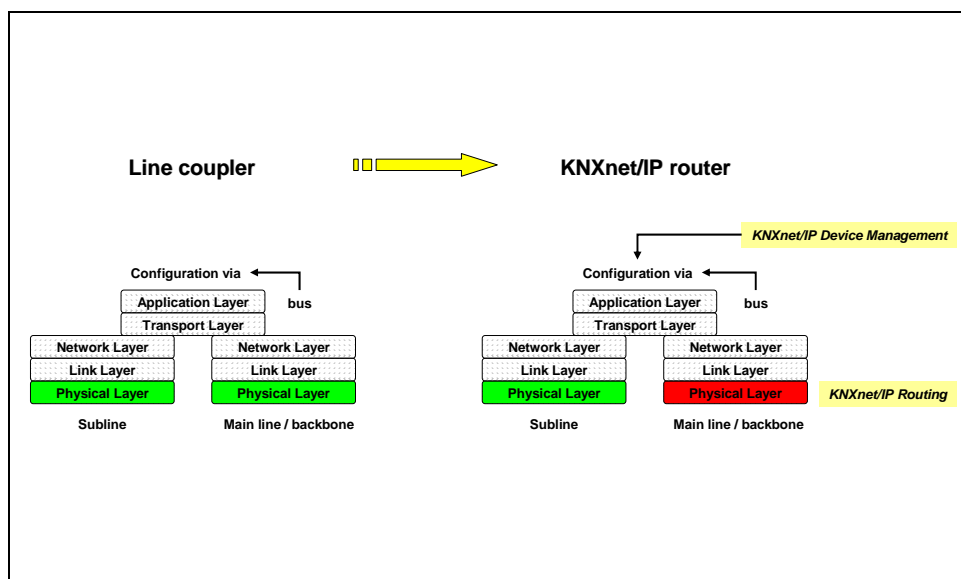


Figure 4

In a similar way KNX IP devices evolve from KNX devices (Figure 5). The existing stack and management procedures are retained but the transport has been changed from TP to KNXnet/IP. Configuration can be achieved through the KNX stack (via "bus"). In this case these configuration telegrams are sent via KNXnet/IP Routing. The second configuration option is available via KNXnet/IP Device Management. Originally, KNXnet/IP Device Management used cEMI property services for configuration. This is not sufficient to support existing management procedures for configuration. To retain these management procedures and the associated telegrams cEMI was enhanced by a cEMI Transport Layer (cEMI TL).

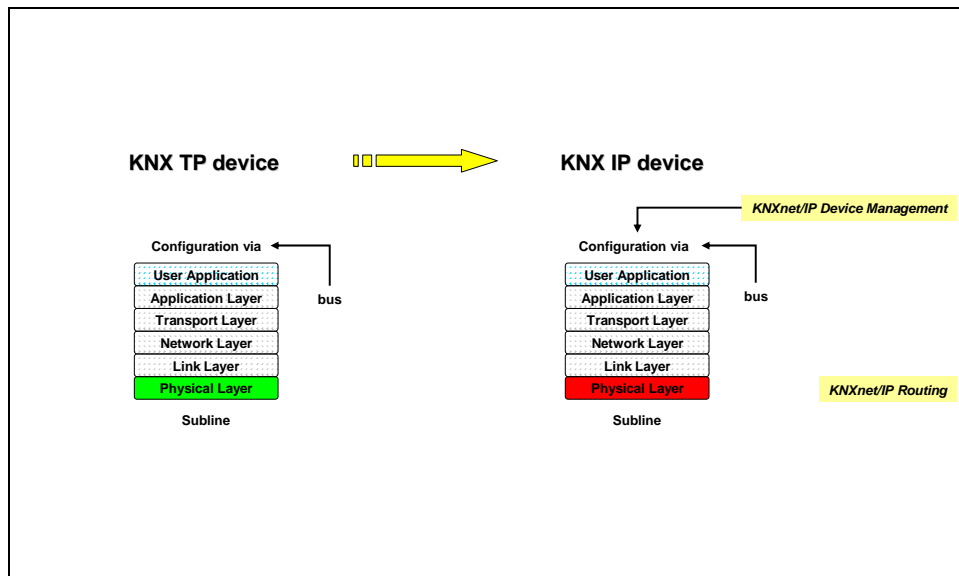


Figure 5

Figure 6 shows a schematic of a KNX IP device with the cEMI Transport Layer. The device may be either configured via the normal Transport Layer or the cEMI Transport Layer. The switch-over between these two Transport Layers is dependent on the connection established for KNXnet/IP Device Management. As long as such a connection is established the cEMI Transport Layer is activated.

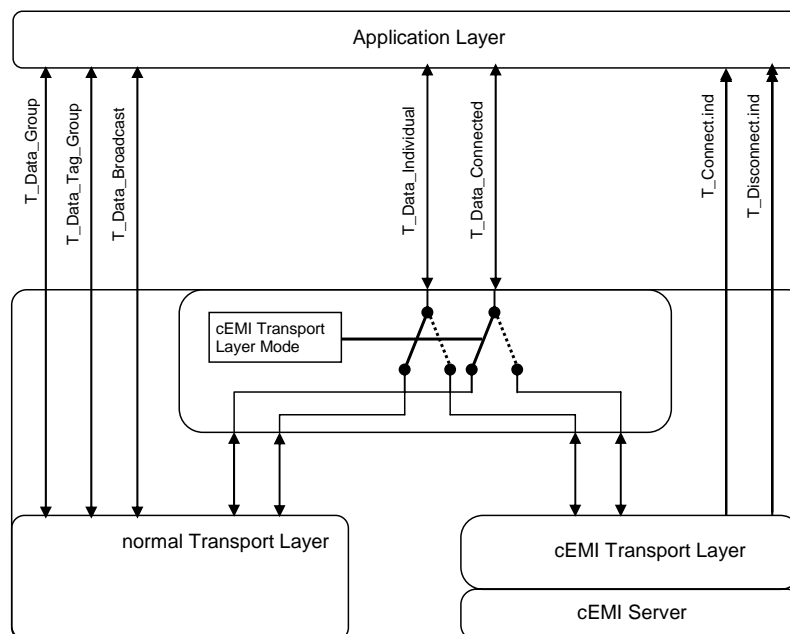


Figure 6

Configuration via KNXnet/IP Device Management is advantageous compared to configuration via KNXnet/IP Routing and the normal Transport Layer because KNXnet/IP Device Management establishes a point-to-point connection between ETS and the KNX IP

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

device. While configuration via KNXnet/IP Tunneling has a one second time-out and configuration via KNXnet/IP Routing with normal Transport Layer has a three second time-out, configuration via KNXnet/IP Device Management has a time-out of ten (10) seconds making it suitable for long-distance configuration of KNX IP devices.

Issues

As described above KNX IP uses KNXnet/IP Routing as the communication basis. This has a number of implications that will be dealt with in this section.

Figure 7 depicts a schematic of a KNXnet/IP router that shows possible communication bottle-necks in a KNXnet/IP router or KNX IP device.

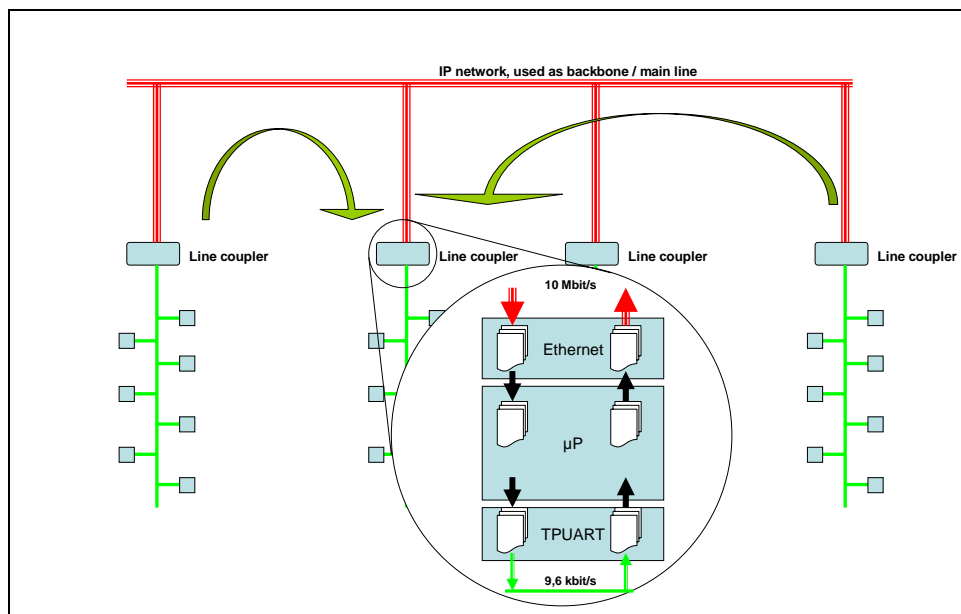


Figure 7

Following the direction of communication there are these possible locations for loss of telegrams / datagrams:

- (A) Routing from KNX subnetwork to KNXnet/IP Routing Multicast Address
- (B) Transmission across the IP network
- (C) Reception from the IP network
- (D) Interface within KNXnet/IP router or KNX IP device between Ethernet interface and micro processor
- (E1) Interface within KNXnet/IP router to KNX subnetwork
- (E2) Interface within KNX IP device between KNX IP communication stack and application

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

(A) Routing from KNX subnetwork to KNXnet/IP Routing Multicast Address

A KNXnet/IP router forwards telegrams from its KNX subnetwork to the IP network using the KNXnet/IP Routing Multicast Address to route the telegrams to other KNXnet/IP routers. Except for occasional network load conditions, forwarding KNX telegrams to the IP network is not critical as the IP network speed is at least 1000 times higher than that of the KNX subnetwork. Because of the capabilities of the KNX TP subnetwork the maximum number of telegrams forwarded from a KNX subnetwork to KNXnet/IP Routing Multicast Address is limited to 50 telegrams per second.

Under a more general system performance perspective it was decided to limit the transmission rate of KNX IP devices to 50 ROUTING_INDICATION datagrams per second.

(B) Transmission across the IP network

KNXnet/IP Routing is based on IP multicast addressing. Under certain network configurations or load conditions of devices or the network, datagrams in an IP network may be lost or even discarded by network components (switches, routers). If KNX IP datagrams are lost in an IP network this cannot be detected by the sender because multicast datagrams are not acknowledged.

Different transmission rates (10 Mbit/s, 100 Mbit/s, 1000 Mbit/s) may lead to traffic conditions in network routers and switches that can only be addressed by existing network methods (e.g. 8802.3x). If these network traffic conditions cannot be addressed then potentially KNX IP datagrams may be lost.

The KNXnet/IP Routing protocol itself has no influence on the performance or behavior of network components. Thus these are out of scope with respect to KNX IP.

Yet, there are measures that can be taken by the system integrator.

Any communication with supervisory systems or ETS should use KNXnet/IP Tunneling, which is an acknowledged point-to-point connection. This excludes loss of datagrams due to the network. KNXnet/IP routers by default support KNXnet/IP Tunneling. At this time, this is not required for KNX IP devices but it makes sense for a manufacturer to implement KNXnet/IP Tunneling in all KNX IP devices.

(C) Reception from the IP network

In general, the hardware design of a KNXnet/IP router or KNX IP device includes a physical 8802.3 (Ethernet) interface (often a specific Ethernet chip), a microprocessor, and in case of a KNXnet/IP router a KNX subnetwork interface like the TPUART.

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

If the Ethernet interface selected by a manufacturer is not suitable for the network data rate (e.g. 10 Mbit/s) this is an issue that is in the responsibility of the manufacturer. An issue with the Ethernet interface performance cannot be removed by whatever protocol measures would be applied.

(D) Interface within KNXnet/IP router or KNX IP device between Ethernet interface and microprocessor

Datagrams are exchanged between each of the three design units depicted in Figure 7 as datagrams are received from the IP network.

KNX IP devices receive datagrams via the IP network transceiver (“Ethernet chip”), which forwards them to the microprocessor. Depending on the hardware and software design of the interface between the network transceiver and the microprocessor the effective data transmission rate between these two parts inside a KNX IP device may be lower than the actual transmission rate on the communication network. Consequently, datagrams may queue up in the Ethernet chip before getting to the microprocessor when the data rate from the IP network exceeds the data rate to the microprocessor. This internal receiving transmission rate limitation may cause the loss of datagrams between network transceiver and microprocessor. This is a hardware, firmware, operating system, and/or application software issue that cannot be solved in general.

The only known measure may be using 8802.3x, which pauses communication between a switch and an end device (e.g a KNX IP device). Not all Ethernet network interface chips support this fairly new feature. It is also unproven that this would be a solution.

Unless 8802.3x is a viable solution, a deficiency of the KNX device design cannot be removed by protocol measures unless the system performance as a whole is reduced. This would be against the intention using the IP network to enhance overall KNX system performance.

As product design is a manufacturer specific responsibility and because of existing devices in the market, KNX IP can only gradually achieve higher system performance by encouraging product improvements from today’s performance levels to future higher performance levels.

Supporting such an approach is the definition of different performance classes that intend to give an indication of the device performance with respect to receiving and processing datagrams.

To ensure a minimum system performance any KNX IP device or KNXnet/IP Router shall be capable of receiving and processing a minimum number of ROUTING_INDICATION datagrams per second on an assigned multicast address. Ideally, any KNX IP device or

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

KNXnet/IP Router should be capable of receiving and processing at least 12 750 ROUTING_INDICATION datagrams¹⁾ per second on an assigned multicast address. This number enables KNX IP devices or KNXnet/IP Routers to receive and process datagrams sent by up to 255 KNX IP devices or KNXnet/IP Routers transmitting at a rate of 50 ROUTING_INDICATION datagrams per second.

The following table defines KNX IP performance classes and their application to different device classes.

Table 1 – KNX IP Performance Classes

KNX IP or KNXnet/IP product	KNX IP Performance Class and number of ROUTING_INDICATION datagrams per second to receive and process		
	A at least 12 750 (per multicast address)	B at least 4 250	C at least 1 000
1. KNX IP Router	M	X	X
2. KNXnet/IP Router	O	O	M
3. KNX IP device	O	O	M
4. Diagnostic software	M	X	X
5. ETS	M	X	X

Symbol	Description
M	Mandatory
O	Optional
X	not allowed

(E1) Interface within KNXnet/IP router to KNX subnetwork

Because KNXnet/IP Routing datagrams are not acknowledged there is no indication on a datagram-by-datagram level about possible overload conditions in one or more KNXnet/IP routers. Because the data rate from the IP network exceeds the data rate to the KNX subnetwork and depending on the configuration a KNXnet/IP Router could receive more datagrams from the LAN than it can send to the KNX Subnetwork. This could lead to an

¹⁾ IP datagram length: 64 octets.

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

overflow of the LAN-to-KNX queue and subsequent loss of one or more KNXnet/IP telegrams because they could not be transferred from the network buffer to the queue.

Flow control needs to be introduced for KNXnet/IP Routers and KNX IP devices to avoid the loss of datagrams due to overflowing queues in KNXnet/IP Routers and KNX IP devices.

Limiting the data rate of sending devices is not a solution for flow control as it does not guarantee that the incoming queue on a specific device (e.g. a KNXnet/IP Router) does not overflow because it is receiving telegrams to be sent onto the local Subnetwork from more than one sending device. The solution is for a receiving device to indicate to all other devices that its incoming queue is filling up and it may lose datagrams if they do not stop sending.

In this case the KNXnet/IP router needs to be able to signal other KNXnet/IP routers or KNX IP devices that its internal buffer to the KNX subnetwork is about to overflow. This signal has been introduced as ROUTING_BUSY, which is sent to the KNXnet/IP Routing Multicast Address.

The buffer overflow warning indication consists of a fixed length data field of four octets. It is used to indicate that the IP receive buffer has filled up to a point where the buffered incoming messages may take at least 100 ms to be sent to the KNX Subnetwork. The structure of the buffer overflow warning indication frame is shown in Figure 8.

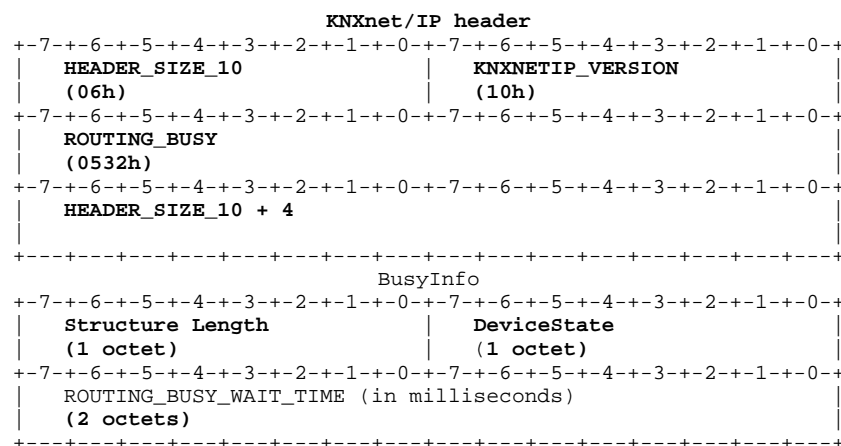


Figure 8 – ROUTING_BUSY frame binary format

ROUTING_BUSY is intended to take care of potential datagram losses due to temporary datagram rate differences between the IP network and a KNX subnetwork. Permanent or frequent datagram flooding of a KNXnet/IP router is a matter of system configuration, which is the responsibility of the system integrator.

ROUTING_BUSY does not take care of product performance issues due to product design limitations with regards to data exchange between network transceiver (Ethernet chip) and microprocessor. This is the responsibility of the manufacturer of a KNXnet/IP router or KNX IP device.

KNX Scientific Conference 2008

KNX IP – using IP networks as KNX medium

(E2) Interface within KNX IP device between KNX IP communication stack and application

The interface within a KNX IP device between the KNX IP communication stack and the application may be prone to similar performance issues as described under (E1) for KNXnet/IP routers. The solution for this situation is the same i.e. ROUTING_BUSY.

In general, it is expected that this case is a rare exception that manufacturers would endeavor to avoid.

Conclusions and Outlook

Following these few design objectives

- maintain the simplicity and scalability of the KNX system
- use management procedures already supported by ETS
- enable reusing existing stack implementations wherever possible
- add capabilities that enhance the KNX system for advanced applications

KNX Task Force IP defined the basics for KNX IP starting from the existing KNXnet/IP Routing protocol.

The influence of IP network performance on the KNX IP network performance and the KNX IP device performance may be dependent on the transmission rate, on network components (switches, routers), or on traffic on the network shared with other users (e.g.office applications). To-date there is an ongoing debate within KNX Task Force IP on how network performance and KNX IP device performance influences the overall KNX IP system performance, and what measures, if any, can be taken to influence the overall KNX IP system performance positively.

Evaluation of the KNX IP protocol definition and assumed KNX IP device performance under different network configurations and network load conditions will provide valuable input for KNX Task Force IP with respect to communication reliability as well as on scalability beyond 255 KNX IP devices. This is why KNX Association has asked the Technical University of Vienna for simulation of KNX IP to verify the concept and provide substantiated guidance on further definitions for KNX IP.