A Powerful Tool for EIB/KNX Development

Key words:
Busmonitor, EIB/KNX-Development, EIB/KNX-Stack;

1 General
The quality of any work is greatly dependent on the availability of efficient tools. When developing components for a bus system, it is important to have a detailed view of what’s going on in the devices and throughout the entire system. Digital oscilloscopes and logic analysers are very helpful when working with hardware and in-circuit emulators, and source-level debuggers will help you find bugs in the software or firmware. Bus monitors are used to analyse the behaviour of a bus device and the interworking of the system.
The Net’n Node software is a comprehensive tool set for EIB/KNX development. It runs on the current versions of MS Windows. Not only is it a bus monitor program, it also features a set of very efficient tools for analysing bus devices and system behaviour, and is able to handle the standard EIB/KNX device models defined today.
The following paragraphs describe the Net’n Node features, giving an overview of the tools included and highlighting the benefits of using Net’n Node during EIB/KNX development.
2 History

Several development tools are available for the EIB/KNX bus. Perhaps the most frequently used tool is the EIB bus monitor from Siemens, which was written during the development of the EIB system about 15 years ago. To this day, the tool runs in DOS mode and cannot be used with any current versions of MS Windows. Newer features of the EIB/KNX system have not been implemented in the program. Figure 1 shows the telegram view of the EIB monitor.

Another well-known tool is the EIB-IDE, also from Siemens. It was specifically designed for writing applications for bus access units (BCUs and BIMs) available as OEM components from Siemens. It was not designed as a bus monitor. The current version runs in a 16-bit environment and the communication does not work on Windows 2000 or XP.

A special monitor is available from EIBA. The EITT (EIB Interworking Test Tool) is used for certification. With the EITT you can define and run linear telegram sequences. Device-specific tools are not included. Also, the ETS (EIB Tool Software) comes with some bus analyser features. However, the capabilities offered by this software are generally insufficient for development.

Initially, Net'n Node was developed for internal use in our company because none of the available development tools fulfilled our needs. Net'n Node proved to be an essential tool for testing during the implementation of our EIB/KNX stack, based on TP-UART (Twisted Pair Universal Asynchronous Receive and Transmit IC) and a microcontroller. The first versions of Net'n Node were only released as part of our EIB/KNX development kit together with our stack implementation. Now Net'n Node is also available as a stand-alone product.

3 Program Framework

The basis for the efficiency of Net'n Node is the very powerful program framework. It is specifically designed for packet-oriented communication software. It runs in a 32-bit environment of MS Windows and has a multi-threaded event-driven architecture. Thus, the user interface is not blocked while a service, such as a download, is running.
The framework is able to handle multiple ports simultaneously. It is possible, for example, to use one port to program a bus device and a second port in bus monitor mode to log the telegrams on the bus. In addition to the connection to the EIB, an RS-485 interface and a serial terminal are also available. As a special feature, the program has the ability to work as a “PEI spy”. Using a special hardware adapter, you can listen to the communication between other computers or microcontrollers with a BCU. This is often a great help when debugging devices and applications that use a serial PEI protocol. Figure 2 shows the main window of Net’n Node with two ports active.

The telegrams transmitted and received via different ports can be shown in separate telegram windows, or they can be combined in a single window to analyse the timing relations of different ports.

4 Bus Access

In the current release of Net’n Node, a 32-bit serial driver is included for Windows 98 and ME to handle PEI16 (EMI1). Our serial driver for Windows 2000 and XP is no longer included because it would conflict with the Falcon driver (used by ETS NG) on the same computer. Interface devices that use PEI10 (FT1.2 / EMI2) can be used with all supported versions of Windows.

To avoid problems with the serial port, we recommend use of a USB data interface. Net’n Node supports the KNX standard for USB interfaces. Most implementations of KNX-USB data
interfaces currently entering the market were developed by Weinzierl Engineering GmbH. A development sample is shown in Figure 3.

![KNX-USB Data Interface](image)

**Figure 3: A KNX-USB data interface (development sample from Weinzierl Eng. GmbH)**

## 5 Sending and Receiving Telegrams

The simplest but nevertheless very important task of a bus monitor is to receive telegrams from the bus and to send frames. To send a single telegram, *Net'n Node* supports two different kinds of dialogs. The first can be used to send group object value very comfortably. Figure 4 and Figure 5 show some examples.

![Send a two byte float](image)

**Figure 4: Send a two byte float**

![Send a time value](image)

**Figure 5: Send a time value**

Figure 6 shows a more flexible dialog. With this dialog, any kind of frame can be assembled manually and sent to the local bus access unit to be transmitted on the bus. For most services defined in the EIB/KNX stack, individual masks have already been prepared in the program.
When receiving telegrams from the bus, it is very useful to get a detailed description of the information it contains. Figure 7 shows a receive window in which the description of telegrams has been switched on. The information displayed in the window corresponds to the layers of the OSI/ISO reference model defined for EIB/KNX.

The contents of the telegram window and all information can be stored on a disk or copied into the PC clipboard and reused, for example, in a text editor.

### 6 System-Oriented Tools

System-oriented tools do not address a single device but rather offer analyser features for system aspects. For example, Figure 8 shows a dialog for scanning a bus line to find all devices connected in the system. The Test ACK tool shown in Figure 9 checks which addresses (group or physical) will be positively acknowledged in the bus.
Also, the assignment of the individual addresses to a bus device is available as a system-oriented tool. A physical address can be programmed using the learning mode of the target device or via the serial number if supported by the device model.

7 Device-Oriented Tools

The broadest range of tools implemented in Net'n Node address an individual device. The device models defined today are quite complex. Therefore, you need a set of sophisticated tools to program and to test bus devices. The following device-oriented tools are currently implemented:

- Access protection
- Switch LED
- Reset device
- Read device state
- Read AD converter
- Edit device memory
- Edit interface objects

All device-oriented tools have the same controls to set the target address. The address can be filled in manually in hex format as well as in decimal format, as used in ETS. In addition, a target address can be read via the bus when activating the learning mode of the target device. In the same manner, a new individual address may be assigned to the target device.
Figure 10 shows the dialog for handling the interface objects (properties). With the **Scan** button, the complete set of property descriptors will be read automatically. The object structure will be displayed as a tree. Of course this dialog allows reading and writing of single or multiple object values.

Figure 11 shows another complex tool. It is called **Device State** and shows all important settings of an addressed device. When reading the device state, the device descriptor type 0 (mask version) is read first, followed by the items defined for this device model, which are read and displayed in a list.

Figure 10: “Edit Properties” tool  
Figure 11: “Device State” tool

### 8 Group-Oriented Tool

A group-oriented tool addresses a group via a multicast. Figure 12 shows the **Read Object Value** tool. The address format used by the dialog can be selected (hexadecimal, 2-layer or 3-layer decimal). The service sends a value read request to the group set at the top of the dialog and waits for the value read responses. The responses will be listed in a table. When clicking on a single response frame in the list, the dialog offers any interpretation possible for the received value length.
9 Device Programming

During the first years of EIB, only the BCU1 was available. This 001x device model is quite easy to understand and the complete device management system is realised by means of direct memory access in an area of 256 bytes. The download procedure for BCU1 devices is defined implicitly and it is the same for all devices.

Now, the 002x and 070x device models are also part of the EIB/KNX specification. These models have a broad set of new features and a complex application-specific load procedure. To handle all device models currently specified, a comfortable device management system has been included in Net’n Node. It consists of a view of several pages and a download system.

Figure 13 displays the Memory page showing the complete EEPROM area in a hexadecimal view. To get an overview of byte usage, the address table, the association table and the communication object table can be highlighted using buttons.

Certain other details of the device configuration, such as the pointer to the communication tables, are displayed on the Details page. The tables themselves can be displayed on a separate page called Tables. It shows the address table, the association table and the communication object table.
To edit the contents of the tables, a special dialog as shown in Figure 15 is available. It can be opened for each communication object and allows the addition and removal of group addresses. The address table and the association table will be updated automatically when closing the dialog using the OK button.

When working with the more complex device models, 002x (BCU2) or 070x (BIM112), the described pages are, of course, also available. Additional pages are included as well. One of these is called Interface Objects. This page extracts the structure of the user interface objects and displays them in a tree.

A special challenge is the handling of the download procedure of these device models. For each application, the download procedure must be individually defined. When creating a new application with EIB-IDE, assembler macros are used to generate pseudo code for non-existing memory locations. Together with the standard memory content, this pseudo code will be stored in a single s19 file. The pseudo code contains the load commands and can only be
interpreted by a few tools. Net’n Node is able to read this code and to display the content in a graphical user interface. In this view, the complete content can be edited. Figure 16 shows the page with the load controls. Of course, the download procedure can be started immediately when changing to the “Download” page.

Figure 16: View of load controls

Generating a device configuration is necessary not only when working with standard BCUs and BIMs. It must also be created to define the management procedures for devices based on TP-UART and an own stack implementation.

As already noted, Net’n Node was developed simultaneously with the stack implementation from Weinzierl Engineering GmbH. In fact, Net’n Node fulfils the very important task of creating the device configurations for new devices based on our stack. All of the information regarding the memory map, the settings and the load procedure is defined in the source code of the application and it is therefore included in the compiler output. Net’n Node is now able to read the compiler output (the binary and the map file) and to create the complete device configuration without any additional user input. Figure 17 shows the import and export features for device configurations.
Figure 18 shows the import dialog for selecting the type of data source. This device configuration can be viewed, edited and tested immediately via a download. At the end, the new configuration can be exported as an s19 file compatible with the ETS manufacturer tool.

![Figure 18: Dialog to select the import format](image)

After the s19 file is imported into the ETS database, the device description has to be completed by setting certain strings and defining the parameters.

### 10 Sequencer and Job Programmer

Device management is a fairly complex topic in EIB/KNX. The most important sequences for handling bus devices have already been prepared. However, you may need to define your own management sequences for special configuration or test purpose. As seen before, load controls can be used to define individual load procedures. It is precisely this mechanism that is used to configure the sequencer. Thus, you can define your sequence at the application layer. This means you do not have to worry about transport layer frames (T-ACK). The communication system uses an internal stack to handle the lower layers. Figure 19 shows a sequence for unloading a device. These sequences can, of course, be stored on a disk and reused at any time.

![Figure 19: The sequencer view](image)

Individually defined sequences are often an issue during device production. Before delivery, special configurations must be programmed into the devices. For example, when using OEM BCUs or BIMs, the manufacturer code must be written into the device memory. Naturally, this is not a problem for the Net’n Node sequencer. However, the standard user interface is not
a very ergonomic production environment. So we have added a new feature that we call the *Job Programmer*.

![Job "Download" active](image)

Figure 20: The job programmer

The job programmer is a special user interface for *Net’n Node*. In this view, sequences that have already been defined and standard downloads can be executed repeatedly. It can be run in full screen mode and the user has no access to edit the sequences or to start other tools. The download process of the job programmer can be started manually by the operator or automatically by another computer or program. After each run, the result is written to a log file for the documentation of the production. Note: The job programmer is not accessible in the standard edition of *Net’n Node* but has to be ordered separately.

### 11 Conclusion

*Net’n Node* is a powerful tool for EIB/KNX development. It combines sophisticated management and analyser features with a state-of-the-art user interface. All parts of the program are optimised to accelerate the development process. And further development of *Net’n Node* has not ceased. We are working intensively to include new features and improve the user interface.

*Net’n Node* is included in our development kit together with our stack implementation based on TP-UART and a microcontroller (e.g. MSP430 or AT-Mega). In addition, it is now also available as a stand-alone product with a company licence.

### 12 Bibliography