



## User-Manual

For the  
COMM-TEC EIB Gateway  
CTG-EIB/NX (v1.1.4)



# Inhaltsverzeichnis

1	Introduction .....	3
1.1	Aims.....	3
1.2	Role of the EIB installer .....	3
1.3	EIB from the AMX programmer's point-of-view .....	3
1.4	EIB characteristics for assignment of the gateway .....	5
2	Hardware .....	6
2.1	Connection, Displays and Controls.....	6
2.2	Installation.....	7
2.2.1	Connection Gateway RS232 interface .....	7
3	Programming .....	8
3.1	Integrating the module .....	8
3.2	Analyzing feedback.....	8
3.3	The NetLinx module interface .....	9
3.3.1	Commands to the module .....	9
3.3.2	Feedback from module.....	11
3.3.3	Terminal Mode .....	12
3.3.4	Channels and levels .....	15
4	Data Types .....	15
5	Appendix A – Sample program .....	16
5.1	Notes EIB Table.....	16
5.2	Notes for programming .....	16
5.2.1	Example 1 – Structure of EIB table with functions from EIB_Tools.AXI .....	17
5.2.2	Example 2 – Structure of EIB-Table with SEND_COMMANDS .....	18
5.2.3	Example 3 – Enter Addresses from File .....	19
5.2.4	Example 4 – Main Program.....	20
6	Appendix B – EIB_Tools.AXI .....	21
7	Appendix C – Comparison with previous version .....	22
7.1	Structure / Generation of filter table in gateway .....	22
7.2	Controlling actuators .....	22
7.3	Feedback .....	23
8	Appendix D – What to do if it does not work? .....	24

Review History (of this document)

Datum	Kürzel	Kommentar
07.09.2006	MSZ/JSA	First Revision
26.10.2006	MSZ	Troubleshooting
22.11.2006	MSZ	Numbering of Chapters, additional Infos
27.11.2006	MSZ	New Infos, Terminal Outputs
02.02.2007	SLI/MSZ	Translation
26.03.2007	JSA/MSZ	Rev 1.0.7, changed Setting Date Format
11.06.2007	MSZ	Rev 1.0.9, EIS5
17.09.2007	MSZ/JSA	Rev 1.1.4, Data Types Text and HEXText

© COMM-TEC  
Siemensstrasse 14  
D-73066 Uhingen  
Tel.: +49 (0)7161/3000-0  
Fax: +49 (0)7161/3000-400

# 1 Introduction

## 1.1 Aims

The software module serves the connection of AMX NetLinx controls to the “European Installation Bus” EIB (“Instabus”). It provides an easy to use interface for developers..

## 1.2 Role of the EIB installer

It cannot be emphasized enough: for the connection to an EIB system solid EIB expert knowledge and close contact to a knowledgeable EIB installer is imperatively advisable. A wrongly set reading flag in an actuator or restrictively programmed line coupler are difficult to find without the right analysis tools.

The NetLinx module cannot configure an EIB system. The package serves contacting a functioning EIB and can only access bus elements with permitted use. For this reason it has to be arranged with the EIB installers, if all bus components are **permitted** to perform the desired functions.

## 1.3 EIB from the AMX programmer’s point-of-view

Analog to AMX systems the European Installation Bus is – as the name says – a bus system: all components are in principle connected to the same line and share the available bandwidth. The bus itself is a 2-core wire providing a power supply of 24V as well as the data transfer between devices. In contrast to AMX the EIB system is organized peripheral – there is no special master controlling communication, but every device may transmit data to any other device. A sophisticated protocol ensures, that only one device transmits at a time to avoid collisions as far as possible.

All communication is carried out by means of so-called telegrams. A telegram is a data package consisting of the following components:

Source ID – hardware address of the transmitting device

Destination address - group addresses of receiving devices

User data

A telegram can be transmitted to several destination devices at the same time, for instance to switch off all lights in a room at the same time. Thus there is a basic difference between both source and destination addresses:

A source address is a hardware address, meaning the address of the *device* transmitting the telegram. A destination address is a group address characterizing a *function*.

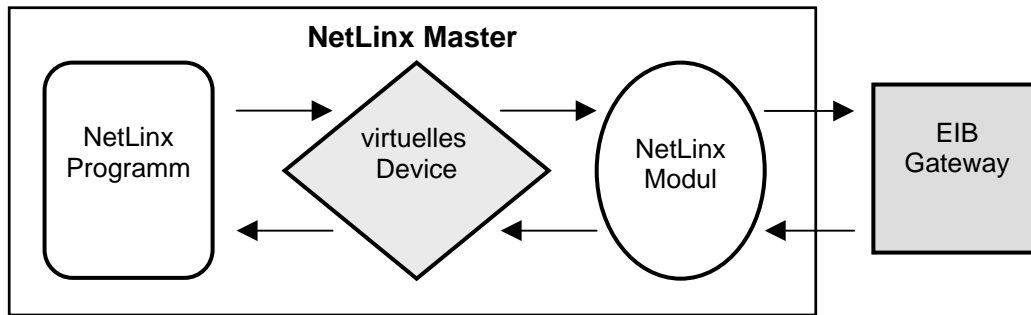
Thus each device connected to EIB has exactly one hardware address, but may have several group addresses. Furthermore, it is possible and common for *several* devices to respond to the same group address.

The EIB installer assigns both address types – the hardware addresses describing type and number of utilized devices and assigned during planning and installation. Hardware addresses are of no concern for the gateway.

The more important addresses for AMX are the group addresses.

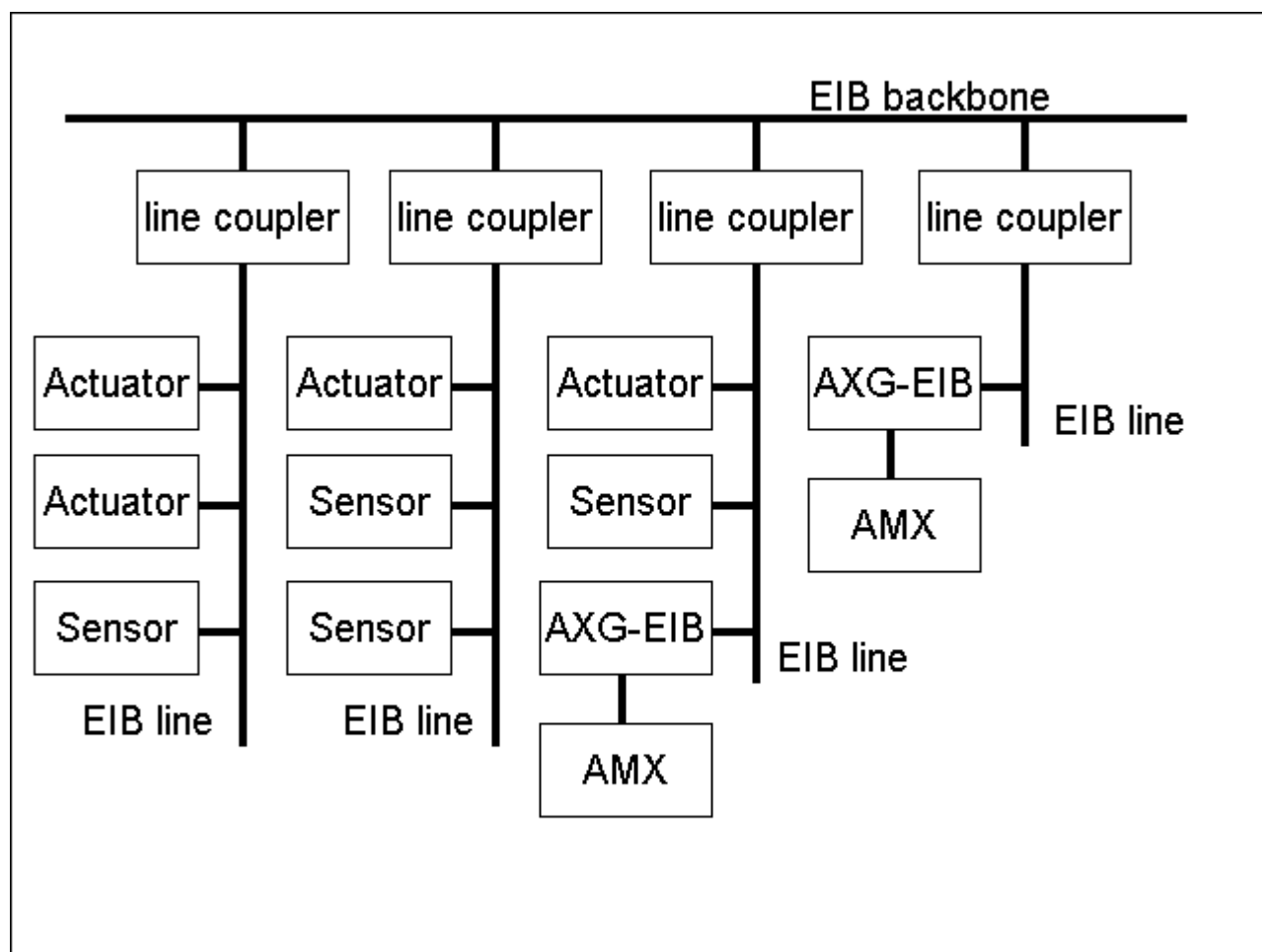
They define the functions an EIB installation can perform. Functions are thus simply actuated by transmitting a certain value to a group address.

The following diagram gives a graphic overview to the course of communication from the project-oriented NetLinx source code via the NetLinx COMM module to the EIB gateway.



**Figure 1: Course of communication**

## 1.4 EIB characteristics for assignment of the gateway



In principle the EIB gateway is a common EIB device and can be connected at any location with the EIB. In contrast to simple actuators and sensors it may be responsible for a multitude (up to 3,000) of group addresses – a normal dimmer for instance only responds to four addresses.

For this it is important to ensure, that the gateway is empowered to respond to all bus telegrams in question. Especially with the use of line couplers a careful planning is necessary. The following should be considered:

On the one hand the bus telegrams of course have to reach the gateway. If line couplers are inserted “between” gateway and the component to be controlled, the filter tables of the couplers are to be programmed to effectively pass on all relevant telegrams.

On the other hand, older bus couplers are slow – after a received telegram an EIB device needs a certain time until the next telegram can be received.

Especially scene modules can produce a flood of telegrams transmitted to all actuators participating in the scene. Because under normal circumstances these are different devices, the down time of the bus coupler does not matter – each coupler has enough time to recover before receiving a new telegram.

The situation with the gateway is different: normally all participating group addresses of a scene are of interest – the gateway needs a recovery time and can read all telegrams, even with high bus load. However, with the activation attention has to be paid, that the activated actuators are given enough time!

## 2 Hardware

### 2.1 Connection, Displays and Controls

Connections :



LEDs :	Busspannung	(Bus Voltage) flashes if bus voltage drops below 15V
	Bustelegramm	(Bus Telegram) displays telegrams on EIB
	RS232 TX	displays data transfer from gateway to AMX
	RS232 RX	displays data transfer from AMX to gateway
	Betrieb	(Power) illuminates if connected to power supply
	Prog.-LED	flashes if gateway is set to programming status from EIB
Pushbutton :	Prog.-Key	sets gateway into programming mode (from EIB)

## 2.2 Installation

### 2.2.1 Connection Gateway RS232 interface

A 5-core cable has to be used, because the gateway works mandatory with hardware handshake (RTS/CTS).

Connection assignment (valid for all serial ports of NetLinx series):

Gateway ( 9 pol SubD )		AMX ( Mini Phoenix ) ( NXI / NXC )		Gateway ( 9 pol SubD )		AMX ( 9 pol SubD ) ( NI Controller )	
Pin		Pin		Pin		Pin	
1				1			
2	—————	2		2	—————	2	
3	—————	3		3	—————	3	
4				4			
5	—————	1		5	—————	5	
6				6			
7	—————	5		7	—————	7	
8	—————	4		8	—————	8	
9				9			

We recommend using the supplied cable.

## 3 Programming

### 3.1 Integrating the module

The interface for module and communication with EIB gateway provides the virtual device vdvEIB. Communication with the gateway is **ONLY** via the virtual device.

A complete documentation of available commands and string feedback is contained in chapter "NetLinx module interface".

The communication module (COMM module) translates between standard command interface on NetLinx side and the protocol for EIB connection.

The communication module for EIB connection is integrated with the following source code line:

```
DEFINE_MODULE 'CTEIB6 mod' EIB6 (vdvEIB,  
                                dvEIB,  
                                lEIB_Value)
```

The parameters are as follows:

<b>vdvEIB</b>	- is the virtual device for communication with the module
<b>dvEIB</b>	- is the physical interface for EIB connection (gateway)
<b>lEIB_Value</b>	- is the central value array of the EIB actuators (type LONG!)

This Array has to be defined in the section DEFINE\_VARIABLE, and has to have a size of 3000 :

```
DEFINE_VARIABLE  
...  
LONG lEIB_Value[3000]  
...
```

The module also attends to the correct initialization of the serial interface, thus it is not necessary to set for instance the baud rate.

To access/analyze actuators on the bus, the EIB group addresses must be introduced to the program. This is done in an external file, mapping group address, type, poll conjunction and additional features on an "AMX number" between 1 and 3000. Communication with the actuators concerned is only made via this number.

This table is integrated with the following source code line:

```
#INCLUDE 'EIB_Table.axi'
```

An example for such a file and a sample program is contained in appendix A.

In addition the file EIB\_Tools.AXI should be integrated to have easy access to commonly used functions (see appendix B).

```
#INCLUDE 'EIB_Tools.axi'
```

### 3.2 Analyzing feedback

The feedback should be analyzed in a Data\_Event. Per feedback **one** DATA\_EVENT is actuated – exactly **one** feedback is in the DATA\_TEXT!

In case several feedback are coming in, the corresponding number of events is actuated.



## 3.3 The NetLinx module interface

### 3.3.1 Commands to the module

Commands to the module always take place per SEND\_COMMAND to the virtual device.  
The module knows the following commands:

Command	Description
ADD=<No>: <Type>: <GrpAdr> [:Flags]	<p>Add an EIB group address to list Note: Flags are optional</p> <p>&lt;No&gt; - 1 .. 3000 = AMX Number of Actuator &lt;Type&gt; - Actuator Type (Switch, Dim4, 1Byte, 2Byte, 3Byte, 4Byte, Text, HexText) &lt;GrpAdr&gt; - EIB group address in 2 or 3 grouped display &lt;Flags&gt; - EIS5 – Value is reported additionally as ASCII Float Value. The EIB Value is converted according to EIS5 Standard (only valid for 2Byte Actuators)</p> <ul style="list-style-type: none"><li>- Time – Value is reported additionally as ASCII Time (hh:mm:ss) (only valid for 3Byte Actuators)</li><li>- Date – Value is reported additionally as ASCII Date (MM/DD/YY) (only valid for 3Byte Actuators)</li><li>- PS – Actuator is automatically polled with Start of AMX System</li></ul> <p>Flags are separated by Commas</p> <p>Examples :</p> <pre>SEND_COMMAND vdvEIB, 'ADD=13:Switch:1/0/11' SEND_COMMAND vdvEIB, 'ADD=17:1Byte:4/7/12:PS' SEND_COMMAND vdvEIB, 'ADD=45:2Byte:3/0/11:EIS5' SEND_COMMAND vdvEIB, 'ADD=12:3Byte:2/1/101:TIME,PS'</pre>
DATE=<No>: <Datum>	<p>Setting the Date Note : only valid for 3Byte Actuators</p> <p>&lt;No&gt; - 1 .. 3000 = AMX Number of Actuator &lt;Date&gt; - date in format MM/DD/YY</p> <p>Example : SEND_COMMAND vdvEIB, 'DATE=17:14/08/06'</p>
DATE?<No>	<p>Request Date Note : only valid for 3Byte Actuators</p> <p>&lt;No&gt; - 1.. 3000 = AMX Number of Actuator</p> <p>Example : SEND_COMMAND vdvEIB, 'DATE?17'</p>
DEBUGON=<Level>	<p>Activating the debug reports</p> <p>With activated debug report all actuators of the terminal are listed, which can be accessed via EIB. This allows simple diagnostics.</p> <p>Example : SEND_COMMAND vdvEIB, 'DEBUGON=1'</p>
DEBUGOFF	<p>Deactivating the debug reports</p> <p>Example : SEND_COMMAND vdvEIB, 'DEBUGOFF'</p>
EIS5=<No>: <floating point value>	<p>Setting an EIS5 value Converts a floating-point value mapped in ASCII into 2Byte EIS5 value before transfer.</p> <p>&lt;No&gt; - 1 .. 3000 = AMX Number of Actuator &lt;Floating Point Value&gt; - Number in the range of. -671088.64 and 670760.96 Note : only valid for 2Byte Actuators</p> <p>Example : SEND_COMMAND vdvEIB, 'EIS5=12:24.3'</p>

Command	Description
EIS5?<No>	Request EIS5 Value Converts the 2Byte raw data into ASCII string with floating point notation Note : only valid for 2Byte Actuators  <No> - 1 .. 3000 = AMX Number of Actuator  Example : <code>SEND_COMMAND vdvEIB, 'EIS5?12'</code>
GET=<No> GET?<No>	Request Value of Actuator stored in the module Note : Creates no telegram on EIB (for synchronization of master-to-master connection only use POLL command) Not for Datatypes Text and HEXText  <No> - 1 .. 3000 = AMX Number of Actuator  Example : <code>SEND_COMMAND vdvEIB, 'GET=17'</code>
POLL=<No> POLL?<No>	Request current value of actuator (for synchronization of master-to-master connection only use POLL command)  <No> - 1 .. 3000 = AMX Number of Actuator  Example : <code>SEND_COMMAND vdvEIB, 'POLL=17'</code> Example : <code>SEND_COMMAND vdvEIB, 'POLL?17'</code>
POLLDELAY=<Value>	Set pause between (automatic) value requests  <Value> - 0-2 (default = 1) Note : 0 stands for very fast and should not be used, because otherwise the gateway would create a high bus load. For installations with slow bus couplers (BCU1), the value 2 should be selected.  Example : <code>SEND_COMMAND vdvEIB, 'POLLDELAY=2'</code>
SENDDelay=<Value>	Delay between commands to EIB. Value is the time in 1/10 sec. The value 0 deactivated the delay.
SET=<No>:<Value>	Set Actuator Note : Observe actuator type in value range! The module limits the value range automatically to max valid range of the accessed actuator  <No> - 1 .. 3000 = AMX Number of Actuator <Value> - Value to be set  Example : <code>SEND_COMMAND vdvEIB, 'SET=5:1'</code>
STARTDELAY=<Value>	Sets point in time, when the module is activated, time for booting of master. <Value> - time in seconds (default = 30, value should not be changed)  Example : <code>SEND_COMMAND vdvEIB, 'STARTDELAY=40'</code>
STATE?	Output of current module status in terminal
TIME=<No>:<Time>	Setzen der Uhrzeit Note : only valid for 3 Byte Actuators  <No> - 1 .. 3000 = AMX Number of Actuator <Time> - time in format hh:mm:ss  Example : <code>SEND_COMMAND vdvEIB, 'TIME=8:13:15:00'</code>
TIME?<No>	Request of time Note : only valid for 3 Byte Actuators  <Nr> - 1 .. 3000 = AMX Number of Actuator  Example : <code>SEND_COMMAND vdvEIB, 'TIME?8'</code>
UPLOAD	Synchronizing tables : Internal – Gateway Writes the current internal table to gateway – <b>only for debugging</b>
VERSION	Output of current module version in Terminal Example : <code>SEND_COMMAND vdvEIB, 'VERSION'</code>

Command	Description
WHEN=<No>:<No2>	<p>Definition von Polltriggern. Hinweis : Der Trigger wird ausgelöst, wenn die erste Adresse angesprochen wird, nicht nur bei einer Wertänderung der ersten Adresse.</p> <p>&lt;No&gt; - 1 .. 3000 = AMX Number of Actuator activating the Polling &lt;No2&gt; - 1 .. 3000 = AMX Number of Actuator to be polled</p> <p>Example : SEND_COMMAND vdvEIB, 'WHEN=32:12'</p>

### 3.3.2 Feedback from module

The feedback is always in STRING format.

The module can generate the following feedback:

String	Description
DATE=<No>:<Value>	<p>Feedback of Date Note : Is transmitted as <b>ADDITIONAL</b> feedback, if in actuator &lt;No&gt; the DATE flag is set.</p> <p>&lt;No&gt; - 1 .. 3000 = AMX Number of Actuator &lt;Value&gt; - Date string in format MM/DD/YY (AMX display)</p> <p>Example : DATE=17:08/14/06</p>
EIS5=<No>:<Value>	<p>Feedback of a value in ASCII floating point display. The actuator value to be coded according to EIS5. Note: Is transmitted as <b>ADDITIONAL</b> feedback, if in actuator &lt;No&gt; the EIS5 flag is set.</p> <p>&lt;No&gt; - 1 .. 3000 = AMX Number of Actuator &lt;Value&gt; - Floating Point Value(String), converted according to EIS Specification</p> <p>Example : EIS5=12:20.25</p>
ERRORM=	<p>Error message from gateway and/or bus. The message are only for information.</p>
SET=<No>:<Value>	<p>Report of a <b>value change</b> Notes: The feedback array (type LONG) is automatically updated, unchanged values are reported as VAL = (see below).</p> <p>&lt;Nr&gt; - 1 .. 3000 – AMX Number of Actuator &lt;Wert&gt; - new Value of Actuator (raw data)</p> <p>Example : SET=8:1</p>
TIME=<No>:<Value>	<p>Feedback of Time Note : Is transmitted as <b>ADDITIONAL</b> feedback, if in actuator &lt;No&gt; the time flag is set.</p> <p>&lt;No&gt; - 1 .. 3000 = AMX Number of Actuator &lt;Value&gt; - Time string in format hh:mm:ss</p> <p>Example : Time=18:09:55:30</p>
VAL=<No>:<Value>	<p>Feedback of an unchanged Value (for instance after GET or POLL) &lt;No&gt; - 1 .. 3000 AMX Number of Actuator &lt;Value&gt; - Value of Actuator</p>

### 3.3.3 Terminal Mode

For implementation and diagnostics several terminal functions are available. The following commands can be transmitted directly in monitor mode to the module. These commands are transmitted via SEND\_COMMANDs to the virtual device.

To visualize the text outputs on the monitor it must be activated with command

**"MSG ON"** and the following Send\_Command has to be sent to the virtual Device :

**SEND\_COMMAND <vdev>,'DEBUGON=1'**

Capitalization is ignored during input; additional parameters (if available) are separated with a **space** (in this documentation with " ").

Naturally, in monitor mode also the commands of the regular command interface are available (i.e. ADD=, SET=, etc.).

Feedback and hints generated in this way serve only diagnostics und should be switched off during runtime. Output messages are marked by the prefix "EIB".

Command	Description
ADR_<Value>	Definition of output format of EIB group address (Main/middle/sub group OR main group/sub group) <Value> - 2/3  Example : SEND_COMMAND vdevEIB, 'ADR 3 '
DEL_<Value>	Delete Actuator from Table  <Value> - 1 .. 3000 – AMX Number of Actuator  Example : SEND_COMMAND vdevEIB, 'DEL 3 '
HELP /?	Output of available Terminal Commands  Example : SEND_COMMAND vdevEIB, 'HELP'
LIST	List all entered actuators with AMX number, EIB group address, current value, set flags (if applicable) and resulting additional feedback values List sum of individual Typees, sum of all actuators  Example : SEND_COMMAND vdevEIB, 'LIST'
LIST_<Type>	List all entered actuators with AMX number, EIB group address, current value, set flags (if applicable) and resulting additional feedback values. Sum of all actuators of one Typee  <Type> - Data Type, where SW or SWITCH – 1Bit Actuators D4 or DIM4 – 4Bit Actuators 1B or 1BYTE – 1Byte Actuators 2B or 2BYTE – 2Byte Actuators 3B or 3BYTE – 3Byte Actuators 4B or 4BYTE – 4Byte Actuators  Example : SEND_COMMAND vdevEIB, 'LIST 1B'
LIST_<No>	List ONE actuator (AMX number) with EIB group address, current value, set flags (if applicable) and resulting additional feedback values  <No> - 1 .. 3000 – AMX Number of Actuator  Example : SEND_COMMAND vdevEIB, 'LIST 17'

Command	Description
LIST_<No>-<No2>	<p>List actuators in the range of &lt;No&gt; to &lt;No2&gt; (AMX numbers) with EIB group address, current value, set flags (if applicable) and resulting additional feedback values</p> <p>&lt;No&gt; - 1 .. 3000 – AMX Number of Actuator (Start)          &lt;No2&gt; - 1 .. 3000 – AMX Number of Actuator (End)</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST 17-24'</p>
LIST_FLAGS	<p>List ALL actuators with assigned flags in table with EIB group address, current value, set flags (if applicable) and resulting additional feedback values</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_FLAGS'</p>
LIST_LOAD_[<Filename>]	<p>Reads the entries in table written with LIST_SAVE from CF and back. The current table is replaced with the read one. File name is optional. If no file name is specified, the default file name is used. Default file name: EIBTableNX.TXT (In terminal connection with master the already available files on CF can be listed by entering "list" (no SEND_COMMAND to virtual device!!!).)</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_LOAD'</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_LOAD MyTable.txt'</p>
LIST_POLL	<p>List all poll triggers with AMX number and EIB group address</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_POLL'</p>
LIST_SAVE_[<Filename>]	<p>Writes the current EIB table, including poll trigger, as text file on CF. This file can be edited with simple text editor. The entries correspond with the structure of the regular table. Thus a table can be buffered, modified (i.e. delete or add actuators) and finally reconstructed with LOAD (see above) File name is optional. If no file name is specified, the default file name is used. Default file name: EIBTableNX.TXT (In monitor connection with master the already available files on CF can be listed by entering "list" (no SEND_COMMAND to virtual device!!!).)</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_SAVE'</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_SAVE MyTable.txt'</p>
LIST_SUM	<p>List sum of all types, sum of all actuators</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_SUM'</p>
LIST_WATCH	<p>List currently observed actuator with EIB group address, current value, set flags (if applicable) and resulting additional feedback values</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_WATCH'</p>
LIST_GAPS	<p>List free (unused) AMX numbers</p> <p>Example : SEND_COMMAND vdvEIB, 'LIST_GAPS'</p>
SEARCH <Groupaddress>	<p>Search for EIB group address          Note: Here 2 and 3 grouped mapping is accepted.          (Caution: The addresses 7 / 715 and 7 / 2 / 203 are i.e. identical EIB group addresses)</p> <p>Example : SEND_COMMAND vdvEIB, 'SEARCH 1/0/101'</p>
STATUS	<p>List general status information for:          AMX hardware, EIB module, gateway, active EIB table</p> <p>Example : SEND_COMMAND vdvEIB, 'STATUS'</p>

Command	Description
UPLOAD	Write Table to Gateway  Example : <code>SEND_COMMAND vdvEIB, 'UPLOAD'</code>
WATCH <No>	Activate observation function for actuator. All value changes are recorded on monitor with EIB group address, current value, set flags and resulting additional feedback values.  <No> - 0 .. 3000 – AMX Number of Actuator, with 0 standing for deactivation of observation  Example : <code>SEND_COMMAND vdvEIB, 'WATCH 12'</code>
WATCH_OFF	Deactivate observation function for actuator

### 3.3.4 Channels and levels

All addresses are available as channels. The current value is mapped to the corresponding channels of the virtual device. For value report "0" the channel is OFF, for all other values ON.

Channel	Description
1 ... 3000	Mapping of values (irrespective of EIB type)
3001	Gateway was detected, module ready for ADD = commands
3002	Gateway and module function correct

The first 200 addresses (actuators in EIB table) are available as levels. For every value change the current value is transmitted as level to the program, for instance to control a bar graph.

Level	Description
1 ... 200	Mapping of values of first 200 group addresses (irrespective of EIB type)

## 4 Data Types

'Switch'	-	Value '0' or '1'	(e.g. Off – On)
'4 Bit'	-	Value '0' to '15'	(e.g. relative dimming Dimmen – direction,interval)
'1 Byte'	-	Value '0' to '255'	(e.g. value absolute)
'2 Byte'	-	Value '0' to '65535'	(e.g. floating point value in EIS5 Notation)
'3 Byte'	-	3 Byte	(e.g. Date or Time)
'4 Byte'	-	4 Byte	
'Text'	-	1 to 14 ASCII Characters,	String automatically filled with spaces
'HEXText'	-	1 to 14 Byte Hexvalue in ASCII-Notation	

## 5 Appendix A – Sample program

### 5.1 Notes EIB Table

All actuators to be switched/set/controlled are to be introduced to the gateway. This is achieved with a table structure.

If this table is generated with a help file (i.e. 'EIB\_Table.AXI') it must only be observed, that all entries are edited, meaning the last Switch\_Case command (default:) is really edited last. There is to be no gap in the counter. For the entries itself the order is irrelevant.

Recommendation: Use the version with help function (**example 1**, see below). In this version less typing errors will occur and the compiler can perform several checks.

Poll triggers are only entered, if the polling and polled addresses are already entered. We recommend entering the poll triggers at the end.

Alternatively, a table may be loaded from the master's Compact Flash. The syntax is identical with SEND\_COMMAND "ADD=". The send command and device address are omitted (**example 3**, see below).

### 5.2 Notes for programming

Predefined functions are available for control to generate the SEND\_COMMANDs for the virtual device.

Recommendation: Use these functions, less typing errors will occur and the compiler can perform several checks.

These functions are in file **EIB\_Tools.AXI** (appendix B).



## 5.2.1 Example 1 – Structure of EIB table with functions from EIB\_Tools.AXI

```
PROGRAM_NAME='EIB_Table'

DEFINE VARIABLE
INTEGER nCounter

DEFINE_START

nCounter = 0
WAIT_UNTIL ([vdvEIB, 3001]) // Startdelay, Channel ON when Module ready
{
    nCounter = 1 // Start
}

#include 'EIB_Tools.axi'

DEFINE_PROGRAM

WAIT 1
{
    IF(nCounter)
    {
        SWITCH(nCounter)
        {
            CASE 1: EIBAdd(vdvEIB, 1, eibSwitch, '1/0/1', "") // Light 1 On/Off
            CASE 2: EIBAdd(vdvEIB, 2, eibSwitch, '1/0/2', "") // Light 2 On/Off
            CASE 3: EIBAdd(vdvEIB, 3, eibSwitch, '1/0/3', "") // Light 3 On/Off
            CASE 4: EIBAdd(vdvEIB, 4, eibSwitch, '1/0/11', "eibPollstart") // Light 2 Status
            CASE 5: EIBAdd(vdvEIB, 5, eibSwitch, '1/0/12', "eibPollstart") // Light 2 Status
            CASE 6: EIBAdd(vdvEIB, 6, eibSwitch, '1/0/13', "eibPollstart") // Light 3 Status

            CASE 7: EIBAdd(vdvEIB, 7, eibSwitch, '1/0/21', "") // Scene,1+2
            CASE 8: EIBAdd(vdvEIB, 8, eibSwitch, '1/0/22', "") // Scene 3+4

            CASE 9: EIBAdd(vdvEIB, 9, eibSwitch, '1/0/31', "") // Blinds, up/down
            CASE 10: EIBAdd(vdvEIB, 10, eibSwitch, '1/0/32', "") // Blinds shutter

            CASE 11: EIBAdd(vdvEIB, 11, eibSwitch, '1/0/111', "") // Dimmer, On/Off
            CASE 12: EIBAdd(vdvEIB, 12, eibDim4, '1/0/112', "") // Dimmer relative
            CASE 13: EIBAdd(vdvEIB, 13, eib1Byte, '1/0/113', "") // Dimmer absolute
            CASE 14: EIBAdd(vdvEIB, 14, eib1Byte, '1/0/114', "eibPollstart") // Dimmer read Value

            CASE 15: EIBAdd(vdvEIB, 15, eib2Byte, '1/0/201', "eibEIS5, ',' ,eibPollstart")
            CASE 16: EIBAdd(vdvEIB, 16, eib1Byte, '1/0/203', "") // analog Value

            CASE 17: EIBAdd(vdvEIB, 17, eib3Byte, '1/0/205', "eibTIME, ',' ,eibPollstart") // Time
            CASE 18: EIBAdd(vdvEIB, 18, eib3Byte, '1/0/206', "eibDATE, ',' ,eibPollstart") // Date

            CASE 19: EIBWhenPoll(vdvEIB, 1, 2) // Polltrigger
            CASE 20: EIBWhenPoll(vdvEIB, 1, 3) // Polltrigger

            CASE 21: EIBWhenPoll(vdvEIB, 11, 14) // Polltrigger
            CASE 22: EIBWhenPoll(vdvEIB, 12, 14) // Polltrigger
            CASE 23: EIBWhenPoll(vdvEIB, 13, 14) // Polltrigger
            CASE 24: EIBWhenPoll(vdvEIB, 14, 11) // Polltrigger

            DEFAULT: nCounter = 0
        }
        IF (nCounter) nCounter ++
    }
}
}
```

## 5.2.2 Example 2 – Structure of EIB-Table with SEND\_COMMANDS

```
PROGRAM_NAME='EIB_Table'

DEFINE VARIABLE
INTEGER nCounter

DEFINE_START

nCounter = 0
WAIT_UNTIL ([vdvEIB, 3001]) // Startdelay, Channel ON when Module ready
{
    nCounter = 1 // Start
}

DEFINE_PROGRAM

WAIT 1
{
    IF(nCounter)
    {
        SWITCH(nCounter)
        {
            CASE 1 : SEND_COMMAND vdvEIB, "'ADD=1:Switch:1/0/1'" // Light 1 On/Off
            CASE 2 : SEND_COMMAND vdvEIB, "'ADD=2:Switch:1/0/2'" // Light 2 On/Off
            CASE 3 : SEND_COMMAND vdvEIB, "'ADD=3:Switch:1/0/3'" // Light 3 On/Off
            CASE 4 : SEND_COMMAND vdvEIB, "'ADD=4:Switch:1/0/11:PS'" // Light 1 Status
            CASE 5 : SEND_COMMAND vdvEIB, "'ADD=5:Switch:1/0/12:PS'" // Light 2 Status
            CASE 6 : SEND_COMMAND vdvEIB, "'ADD=6:Switch:1/0/13:PS'" // Light 3 Status

            CASE 7 : SEND_COMMAND vdvEIB, "'ADD=7:Switch:1/0/21'" // Scene 1+2
            CASE 8 : SEND_COMMAND vdvEIB, "'ADD=8:Switch:1/0/22'" // Scene 3+4

            CASE 9 : SEND_COMMAND vdvEIB, "'ADD=9:Switch:1/0/31'" // Blinds up/down
            CASE 10 : SEND_COMMAND vdvEIB, "'ADD=10:Switch:1/0/32'" // Blinds shutter

            CASE 11 : SEND_COMMAND vdvEIB, "'ADD=11:Switch:1/0/111'" // Dimmer, On/Off
            CASE 12 : SEND_COMMAND vdvEIB, "'ADD=12:Dim4:1/0/112'" // Dimmer relative
            CASE 13 : SEND_COMMAND vdvEIB, "'ADD=13:1Byte:1/0/113'" // Dimmer absolute
            CASE 14 : SEND_COMMAND vdvEIB, "'ADD=14:1Byte:1/0/114:PS'" // Dimmer read Value

            CASE 15 : SEND_COMMAND vdvEIB, "'ADD=15:2Byte:1/0/201:EIS5,PS'" // analog Value
            CASE 16 : SEND_COMMAND vdvEIB, "'ADD=16:1Byte:1/0/203'" // analog Value

            CASE 17 : SEND_COMMAND vdvEIB, "'ADD=17:3Byte:1/0/205:Time,PS'" // Time
            CASE 18 : SEND_COMMAND vdvEIB, "'ADD=18:3Byte:1/0/206:Date,PS'" // Date

            CASE 19 : SEND_COMMAND vdvEIB, "'WHEN=1:2'" // Polltrigger
            CASE 20 : SEND_COMMAND vdvEIB, "'WHEN=1:3'" // Polltrigger

            CASE 21 : SEND_COMMAND vdvEIB, "'WHEN=11:14'" // Polltrigger
            CASE 22 : SEND_COMMAND vdvEIB, "'WHEN=12:14'" // Polltrigger
            CASE 23 : SEND_COMMAND vdvEIB, "'WHEN=13:14'" // Polltrigger
            CASE 24 : SEND_COMMAND vdvEIB, "'WHEN=14:11'" // Polltrigger

            DEFAULT: nCounter = 0
        }
        IF (nCounter) nCounter ++
    }
}
}
```

### 5.2.3 Example 3 – Enter Addresses from File

The address table can be read and generated from file on CF card. The reading of the file can for instance be started in the ONLINE section of the interface.  
(Description see chapter “Terminal mode”)

```
...  
DATA_EVENT[dvEIB]  
{  
  ONLINE :  
  {  
    SEND_COMMAND vdveIB, 'LIST LOAD MyTable.txt'  
  }  
}  
...
```

Note: Comments at the end of a line must be separated by at least one space und are initiated – corresponding with programming – with “//”.

Only one command per line is permitted. Leading spaces are ignored.

Lines starting with “//” are completely treated as a comment and not executed by the controlsystem.

```
// EIB - Tabelle fuer Project xy  
//  
// Group Addresses  
//  
ADD=1:Switch:1/0/1           // Light 1 On/Off  
ADD=2:Switch:1/0/2           // Light 2 On/Off  
ADD=3:Switch:1/0/3           // Light 3 On/Off  
ADD=4:Switch:1/0/11:PS       // Light 1 Status, poll on Start  
ADD=5:Switch:1/0/12:PS       // Light 2 Status, poll on Start  
ADD=6:Switch:1/0/13:PS       // Light 3 Status, poll on Start  
ADD=7:Switch:1/0/21           // Scene 1+2  
ADD=8:Switch:1/0/22           // Scene 3+4  
ADD=9:Switch:1/0/31           // Blinds up/down  
ADD=10:Switch:1/0/32          // Blinds shutter  
ADD=11:Switch:1/0/111        // Dimmer On/Off  
ADD=12:Dim4:1/0/112          // Dimmer relative  
ADD=13:1Byte:1/0/113         // Dimmer absolute  
ADD=14:1Byte:1/0/114:PS      // Dimmer read Value, beim Start abfragen  
ADD=15:2Byte:1/0/201:EIS5,PS // analog Value, poll on Start  
ADD=16:1Byte:1/0/203         // analog Value  
ADD=17:3Byte:1/0/205:Time,PS // Time, poll on Start  
ADD=18:3Byte:1/0/206:Date,PS // Date, poll on Start  
//  
// Polltrigger  
//  
WHEN=1:2                     // Polltrigger  
WHEN=1:3                     // Polltrigger  
WHEN=11:14                   // Polltrigger  
WHEN=12:14                   // Polltrigger  
WHEN=13:14                   // Polltrigger  
WHEN=14:11                   // Polltrigger
```

## 5.2.4 Example 4 – Main Program

### DEFINE\_DEVICE

```
dvEIB      = 5001:1:0
vdvEIB     = 33001:1:0

dvPanel    = 10001:1:0
```

### DEFINE\_CONSTANT

```
...
```

### DEFINE\_VARIABLE

```
VOLATILE LONG lEIB_Value[3000] // Feedbackarray for max 3000 Addresses
...
```

### DEFINE\_START

```
...
```

```
#INCLUDE `EIB_Table.AXI`
```

```
DEFINE_MODULE `CTEIB6_mod` GATEWAY_1 (vdvEIB, dvEIB, lEIB_Value)
```

### DEFINE\_EVENT

```
...
```

```
BUTTON_EVENT[dvPanel,1]
```

```
{
    PUSH :
    {
        EIBSet(vdvEIB,1,1)           // Light 1 ON
        EIBSet(vdvEIB,16,128)        // Ballast to 50%
        EIBSet(vdvEIB,12,10)         // Dimmer up
    }
    RELEASE :
    {
        EIBSet(vdvEIB,12,0)          // Dimmer Stop
    }
}
```

### DEFINE\_PROGRAM

```
...
```

```
[dvPanel,11] = lEIB_Value[4]        // Feedback for Light 1 (see EIB Table)
[dvPanel,12] = (lEIB_Value[16] > 127) // Button ON, when Ballast >= 50%
```

## 6 Appendix B – EIB\_Tools.AXI

We recommend not to use the send commands directly, but always use the functions of this include file. The compiler has the opportunity to avoid typing errors already during compiling. Additional typing is avoided.

This file also provides absolute terms for relative dimming:

```
EIB_DIM_UP = 9           // Brighter
EIB_DIM_DN = 1           // Dark
EIB_DIM_SP = 0           // Dimming Stop
```

The following functions are available in file EIB\_Tools.AXI for programming:

**EIBSet (<Virtual Device >,<AMXNo>,<Value>)**

Function: Sets actuator <AMXNo> to <Value>

The module limits the value range automatically to the maximum range of the selected actuator.

Example: EIBSet (vdvEIB,13,1)

**EIBGet (<Virtual Device>,<AMXNo>)**

Function: Gets the value of actuator <AMXNo> to <Value> stored in module

Example: nVAL = EIBGet (vdvEIB,13)

**EIBPoll (<Virtual Device>,<AMXNo>)**

Function: Polls the actuator <AMXNo>

Example: EIBPoll (vdvEIB,13)

**EIBAdd (<Virtual Device>,<AMXNo>,<Type>,<Group Address>,<Flags>)**

Function: Adds entry to EIB table (description of parameters see above)

Example: EIBAdd (vdvEIB,13,eib2Byte,'1/0/206',"eibPollstart")

**EIBWhenPoll (<Virtual Device>,<AMXNo1>,<AMXNo2>)**

Function: Adds a poll trigger. Value report from <AMXNo1> triggers polling on <AMXNo2>.

Example: EIBWhenPoll (vdvEIB,13,20)

## 7 Appendix C – Comparison with previous version

The general operation of hardware and software has not changed. A filter table has still to be generated and loaded into the gateway. Via this filter the incoming and outgoing telegrams are filtered. The control of actuators or feedback from sensors takes place by setting of values, which are assigned to certain addresses.

- in order to limit the busload created by AMX on the EIB,
  - approx. 10 telegrams are being released per second during polling
  - approx. 35 telegrams are being released per second on triggering (formerly approx. 10 telegrams)
- Telegrams are being read accurately, until the maximum busload is reached.
- resource-sparing operation of both Gateway and Driver. A permanent communication with the Gateway over the serial interface does not apply.
- the driver has been optimized for the ease of use in networked systems.
- the driver eases the examination of the triggering even without operational EIB (compare terminal command "DEBUGON").

## 7.1 Structure / Generation of filter table in gateway

Bisher :      SEND\_COMMAND vdvEIB, "ADD SW 4 1/0/65"

In the table structure now all actuators/sensors get a distinct number (1 ... 3000) independent of their type. Via this number the actuators/sensors are accessed. Additionally, several flags can be set, in the following example for instance “PS”. This means, that the corresponding actuator is immediately polled when starting the AMX (description or available flags see chapter “Module commands”).

```
NEW:      EIBAdd(vdvEIB,4,Switch,'1/0/65',"PS")    (recommended !)    or
          SEND COMMAND vdvEIB,"'ADD=4:Switch:1/0/1:PS'"
```

## 7.2 Controlling actuators

OLD:       SYSTEM CALL 'EIB NX Set Switch'(vdvEIB,1,4)

```
NEW:      EIBSet(vdvEIB,4,1)                                (recommended !)    or
          SEND COMMAND vdvEIB,``SET=4:1``
```

### 7.3 Feedback

There is now only ONE array to save the values of ALL actuators/sensors.

In addition feedback can be output in a readable ASCII display – depending on flags – meaning, the raw data are output as time string, date string, floating point display etc.

#### Example :

Feedback of a 2Byte value, converted according to EIS5 standard (i.e. temperature value). The corresponding actuator was entered in the filter table with flag "EIS5".

```
EIBAdd (vdvEIB, 15, eib2Byte, '1/0/201', "eibEIS5")
```

The virtual device will report two feedback with each value change (or as answer to a poll command):

String 1 from virtual device (value change):	'SET=15:3175'
String 2 from virtual device:	'EIS5=15:22.54'

or

String 1 from virtual device (no value change):	'VAL=15:3175'
String 2 from virtual device:	'EIS5=15:22.54'

## 8 Appendix D – What to do if it does not work?

The following table provides tips for error definition, in case it does not work.

This serves a quick error analysis **ON SITE**.

We recommend activating debugging mode during diagnostics to display additional error messages.

This is activated with monitor command "DEBUGON".

Error	Proposed solution / error definition
No controls possible, no feedback	Enter command "Status" in monitor mode. If the gateway is not detected, the cable is probably wrong / defective or the gateway has no power supply.
No controls possible, no feedback, according to "Status" the gateway is detected	Enter command "List" in monitor mode. Are all addresses entered? Are here feedback values displayed? Try to switch several addresses directly with "SET" (e.g. light). If it works, there is probably an error in the AMX program. Is here also no access possible, the reason is probably wrong group addresses. (Is the light still on in the adjacent building?)
The Module consistently reports "no STX at beginning of GW Feedback"	The Gateway is being recognized by the module, but during the attempt of reading out the version, the connection is getting interrupted again. Solution: Please check the cabling from AMX to the Gateway