

# **CTEIB4**

## **AMX and NetLinx go Instabus®**



## **Bedienungsanleitung für das Comm-Tec EIB-Gateway AXG-EIB und der Ansteuersoftware CTEIB4**

Version 4  
Stand: 1.Juni 2002

© COMM-TEC  
Siemensstr. 14, 73066 Uhingen  
Tel.: +49/7161/3000-0  
Fax: +49/7161/3000-400

# Inhaltsverzeichnis

<b>INHALTSVERZEICHNIS .....</b>	<b>2</b>
<b>VORWORT.....</b>	<b>3</b>
ZIELSETZUNG .....	3
ROLLE DES EIB-INSTALLATEURS .....	3
EIB AUS DER SICHT DES AMX-PROGRAMMIERERS .....	3
EIB-BESONDERHEITEN BEIM EINSATZ DES GATEWAYS .....	4
<b>VORWORT ZUR REVISION 4.00.....</b>	<b>5</b>
<b>SYSTEMBESCHREIBUNG.....</b>	<b>6</b>
HARDWAREKOMPONENTEN .....	6
SOFTWAREKOMPONENTEN.....	7
GRUPPEN, ADRESSEN UND FORMATE .....	9
TRIGGER.....	10
<b>INSTALLATION UND INBETRIEBNAHME.....</b>	<b>11</b>
INSTALLATION DES GATEWAYS.....	11
INSTALLATION DER RS232-SCHNITTSTELLE.....	11
VERBINDUNG GATEWAY-RS232 SCHNITTSTELLE .....	12
EINBINDEN DES MODULS .....	12
GRUPPENADRESSEN IN DAS MODUL EINTRAGEN .....	12
VERSORGUNG DES MODULS AUS DEM HAUPTPROGRAMM.....	13
<b>KOMMUNIKATION MIT DEM MODUL .....</b>	<b>13</b>
SYSTEM_CALLS .....	13
SETZEN VON WERTEN .....	14
RÜCKMELDUNGEN .....	14
AKTIVE ABFRAGEN (POLLING) .....	14
BEFEHLE AN DAS MODUL .....	15
DATENKONVERTIERUNG .....	15
"TERMINAL"-MODUS .....	16
<b>BEFEHLSSATZ DES MODULS.....</b>	<b>16</b>
ADD – GRUPPENADRESSE EINTRAGEN.....	16
ADR – FORMAT DER ADRESSDARSTELLUNG UMSCHALTEN.....	17
DELTA – DIFFERENTIALÜBERTRAGUNG EIN/AUS .....	17
LIST – ANZEIGE DER BESTEHENDEN VERKNÜPFUNGEN .....	17
NOPOLL - LÖSCHEN EINES POLL-TRIGGERS .....	18
POLL - AKTIVE WERTABFRAGE.....	18
RESEND - ALLE WERTE ERNEUT SCHICKEN.....	18
RESET - GATEWAY-RESET AUSLÖSEN .....	18
SET - GRUPPENADRESSE SCHREIBEN .....	18
STATUS – STATUSINFORMATION ANZEIGEN .....	19
UPLOAD - GATEWAY-FILTERTABELLE NEU LADEN .....	19
WATCH - ÜBERWACHUNG EINER ADRESSE .....	19
WHEN .. POLL - SETZEN EINES POLL-TRIGGERS.....	20
<b>ANHANG.....</b>	<b>21</b>
A WERTE FÜR DIM4-AKTOREN.....	21
B BEISPIEL: HAUPTPROGRAMM UND INCLUDE DATEI .....	22
C FEHLERMELDUNGEN.....	26

# Vorwort

## Zielsetzung

Das Softwarepaket CTEIB4 dient der Ankopplung von AMX NetLinx-Steuerungen an den "European Installation Bus" EIB ("Instabus"). Es stellt eine einfach zu verwendende Schnittstelle für den Entwickler zur Verfügung.

## Rolle des EIB-Installateurs

Es kann nicht genug betont werden: beim Anschluß an ein EIB-System ist solides EIB-Fachwissen und der enge Kontakt zu einem kundigen EIB-Installateur dringend anzuraten. Ein falsch gesetztes Lese-Flag in einem Aktor oder ein restriktiv programmierter Linienkoppler können ohne gute Analyse-Tools wirklich schwer zu finden sein.

CTEIB4 kann keinesfalls ein EIB-System konfigurieren. Das Paket dient zur Kontaktaufnahme mit einem funktionsfähigen EIB, und kann nur auf Buselemente zugreifen, deren Benutzung auch gestattet ist. Schon aus diesem Grund sollte mit den EIB-Installateuren geklärt werden, ob alle Buskomponenten die gewünschten Funktionen auch ausführen *dürfen*.

## EIB aus der Sicht des AMX-Programmierers

Analog zu AMX-Systemen ist der European Installation Bus – wie der Name schon sagt – ein Bussystem: alle Komponenten sind im Prinzip an der selben Leitung angeschlossen und teilen sich die verfügbare Bandbreite. Der Bus selbst ist ein zweiadriges Kabel, das sowohl eine Versorgungsspannung von 24V bereitstellt, als auch dem Datentransport zwischen den Geräten dient. Im Gegensatz zu AMX ist ein EIB-System aber dezentral organisiert – es gibt keinen speziellen Master, der die Kommunikation steuert, sondern jedes Gerät darf an jedes andere Gerät Daten senden. Ein ausgeklügeltes Protokoll stellt dabei sicher, daß immer nur ein Gerät gleichzeitig sendet, und Kollisionen weitestgehend vermieden werden.

Sämtliche Kommunikation erfolgt dabei in Form sogenannter Telegramme. Ein Telegramm ist ein Datenpaket, das aus folgenden Komponenten besteht:

- Absenderkennung – Hardware-Adresse des sendenden Gerätes
- Zieladresse – Gruppenadresse der empfangenden Geräte
- Nutzdaten

Ein Telegramm kann dabei an mehrere Zielgeräte gleichzeitig gesendet werden, z.B. um sämtliche Lampen in einem Raum gleichzeitig auszuschalten. Es besteht also ein grundlegender Unterschied zwischen Quell- und Zieladressen:

Eine Quelladresse ist eine Hardware-Adresse, also die Adresse des *Gerätes* das das Telegramm sendet. Eine Zieladresse ist eine Gruppen-Adresse, eine Adresse, die eine *Funktion* beschreibt.

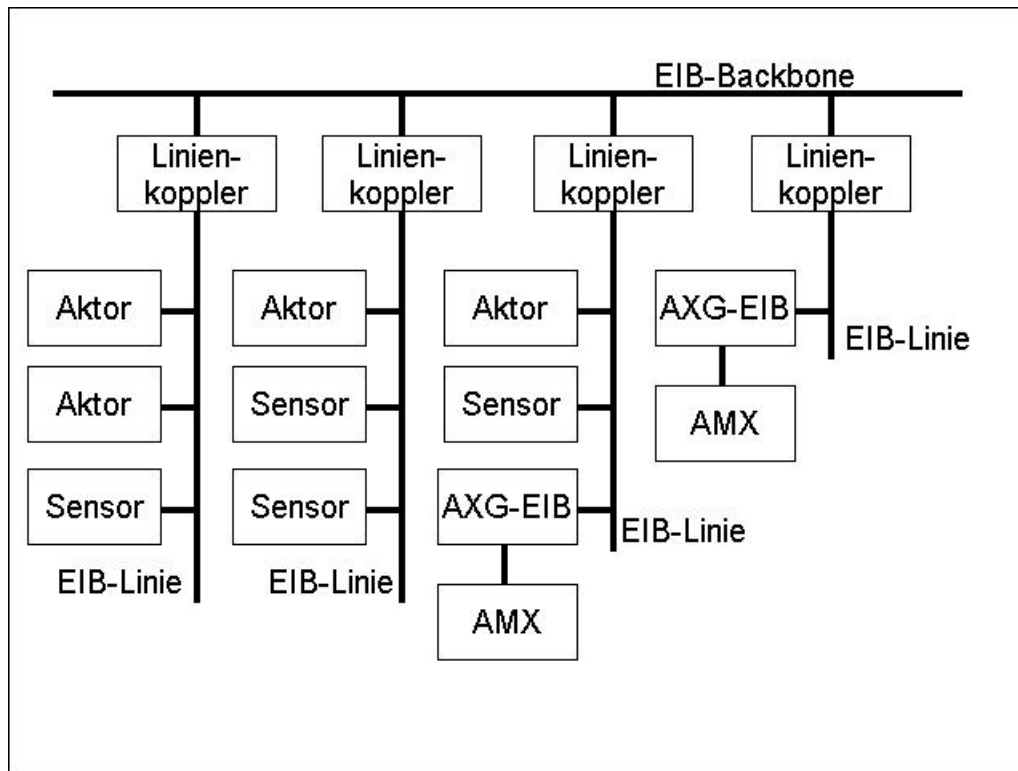
Jedes Gerät am EIB hat also genau eine Hardware-Adresse, kann aber durchaus mehrere Gruppenadressen haben. Ebenso ist es möglich, daß *mehrere* Geräte auf die selbe Gruppenadresse reagieren.

Beide Adressarten werden vom EIB-Installateur vergeben – die Hardware-Adressen beschreiben Art und Anzahl der verwendeten Geräte, und werden während der Planung und Installation zugeteilt. Für Hardware-Adressen interessiert sich das Gateway überhaupt nicht.

Die wesentlich wichtigeren Adressen für AMX sind die Gruppenadressen.

Sie legen die Funktionen fest, die eine EIB-Installation ausführen kann. Funktionen werden also schlicht dadurch ausgelöst, daß einer Gruppenadresse ein bestimmter Wert gesendet wird.

## EIB-Besonderheiten beim Einsatz des Gateways



Prinzipiell ist das EIB-Gateway AXG-EIB zwar ein gewöhnliches EIB-Gerät, kann also an jeder Stelle mit dem EI-Bus verbunden werden. Im Gegensatz zu einfachen Aktoren und Sensoren ist es aber unter Umständen für sehr viele (bis zu 1530) Gruppenadressen zuständig – ein normaler Dimmer reagiert z.B. nur auf vier Adressen.

Daher ist unbedingt darauf zu achten, daß das Gateway eine reelle Chance hat, auf alle Bus-Telegramme zu reagieren, die von Interesse sind. Insbesondere beim Einsatz von Linienkopplern ist eine sorgfältige Planung im Vorfeld unabdingbar. Folgende Überlegungen sollten berücksichtigt werden:

Zum Einen müssen Bustelegramme das Gateway natürlich erreichen können. Sind Linienkoppler „zwischen“ Gateway und zu steuernden Komponenten eingefügt, müssen die Filtertabellen der Koppler so programmiert sein, daß auch wirklich alle relevanten Telegramme durchgereicht werden.

Zum anderen sind EI-Busankoppler träge – nach jedem empfangenen Telegramm benötigt ein EIB-Gerät eine gewisse Zeit, bis das nächste Telegramm aufgenommen werden kann.

Speziell Szenenbausteine können aber eine Flut von Telegrammen erzeugen, die an die - an der Szene beteiligten - Aktoren gesendet werden. Da das normalerweise *verschiedene* Geräte sind, fällt die „Totzeit“ der Busankoppler nicht ins Gewicht – jeder Koppler hat genügend Zeit, sich zu erholen, bevor ein neues Telegramm an ihn empfangen wird.

Nicht so beim Gateway: üblicherweise sind *sämtliche* beteiligten Gruppenadressen einer Szene von Interesse – der Gateway-Busankoppler sollte also die Gelegenheit haben, *alle* Telegramme mitzulesen

Problem: die erste Quittung reicht, und die muss nicht unbedingt vom Gateway stammen!

Tatsächlich kann es vorkommen, dass ein an der Szene beteiligter Aktor ein Telegramm quittiert, das Gateway aber nichts davon mitbekommt, weil es noch mit der Verarbeitung des vorherigen beschäftigt ist... Sollte das vorkommen, gibt es zwei Taktiken:

1. Der Szenebaustein wird so programmiert, daß nicht die maximale Busbandbreite (ca. 50 Telegramme/Sekunde) ausgenutzt wird, sondern nur etwa zehn Telegramme pro Sekunde gesendet werden.
2. Das Gateway bekommt eine eigenen Linie, wird also von den „Zielaktoren“ isoliert. Dann kann der Linienkoppler sicherstellen, daß alle Pakete *an* das Gateway auch *vom* Gateway quittiert werden (sofern der Busankoppler des Linienkopplers schnell genug ist ;-)).

## Vorwort zur Revision 4.00

Mit der neuen NetLinx Controller Generation haben Sie nun die Möglichkeit, ohne zusätzlich notwendige Hardware in gewohnter Weise auf den EIB zugreifen zu können. Die leistungsfähigen NetLinx Controller können jetzt die nicht unerhebliche Arbeit, die bisher von der AXB-232++ erledigt wurde, in einem abgesetzten Software Modul erledigen.

Hauptkriterien zur Anwendung dieses Moduls waren ein möglichst reibungsloser Umstieg auf das neue Produkt, unter Berücksichtigung der bereits erworbene Kenntnisse im Umgang mit den früheren Revisionen.

Sie können weiterhin Ihre AMX – Anbindung an EIB-System wie bisher in gewohnter Weise realisieren. Die 232++ - interne Software funktioniert nun als Software-emulierte Hardware.

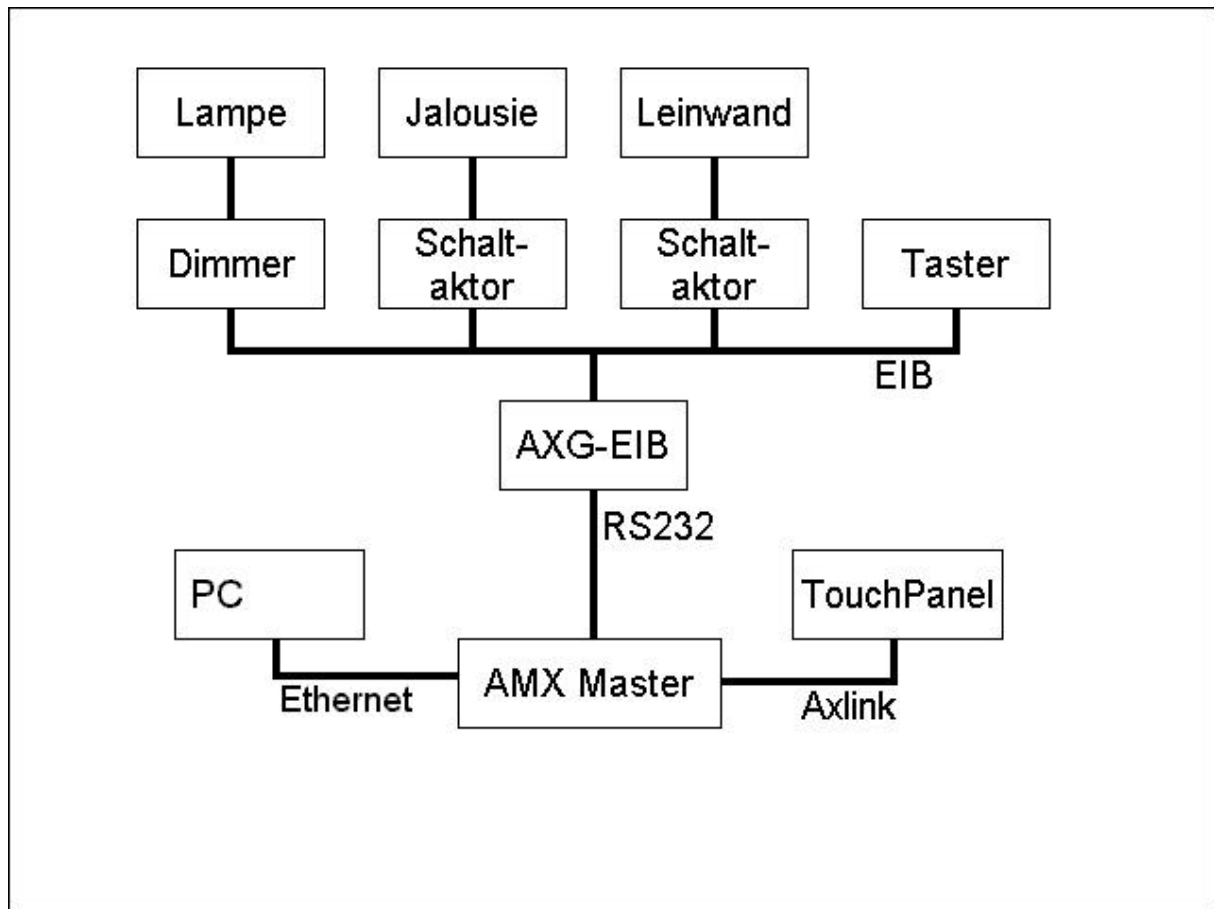
Die Datenfelder mit den aktuellen Switch-, Dim4-, 1Byte-, 2Byte-, 3Byte-, 4Byte-Werten müssen wie aus der Version CTEIB3 gewohnt als Integer-Arrays deklariert werden (siehe Beispiel im Anhang B). Die mitgelieferten SYSTEM\_CALLs haben sich in ihrer Funktion nicht geändert, lediglich der Einsatz der Rückmeldecalls ist nun nicht mehr notwendig.

Dieses Software-Modul inclusive der dazugehörigen SYSTEM\_CALLs arbeitet nur mit Gateways ab der Version 3.01 und Controllern der NetLinx Generation.

Uhingen, Juli 2002

# Systembeschreibung

## Hardwarekomponenten



Zur Anbindung eines AMX-Systems an EIB sind mehrere Hardware-Komponenten notwendig:

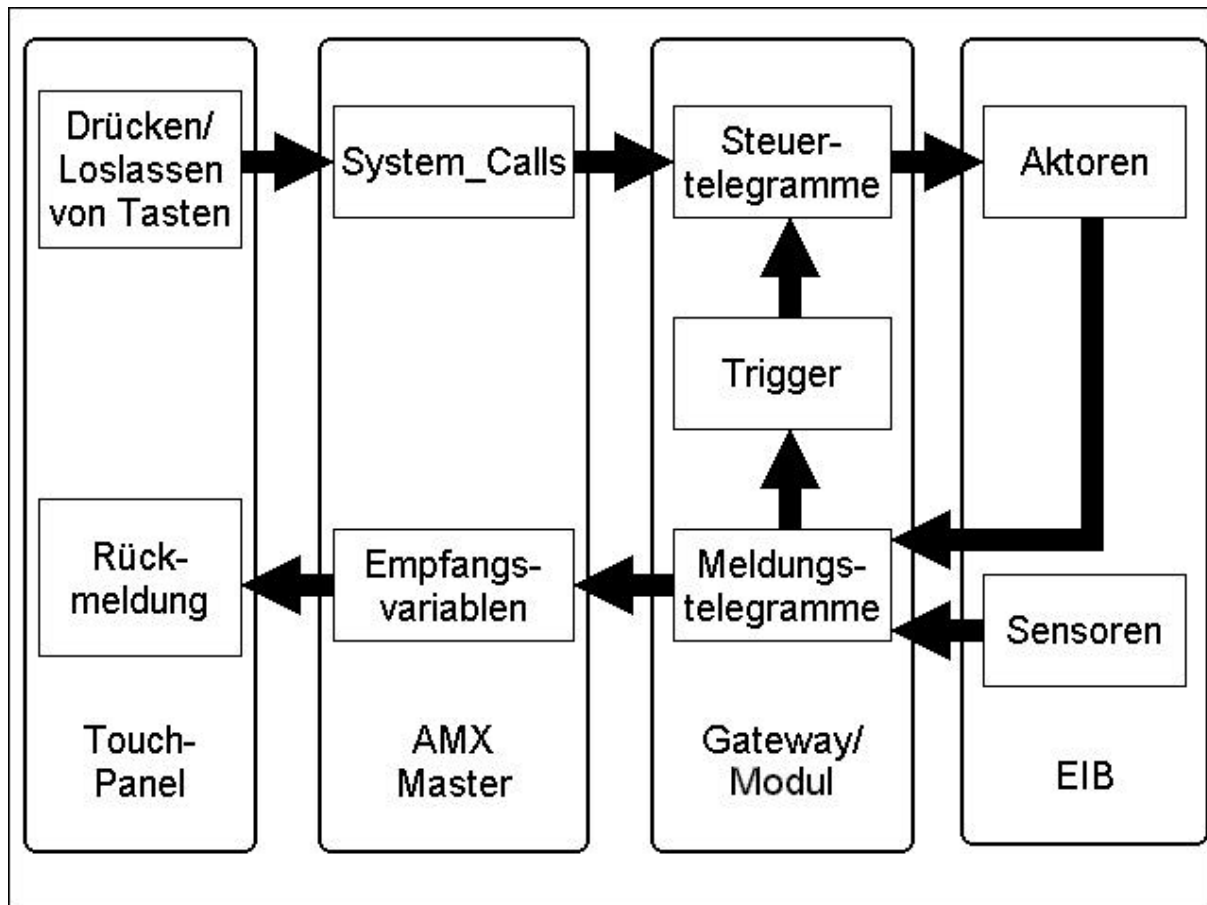
- ein **AMX-Master** der NetLinX Generation (ab Ver. 2.0.70), in dem das Hauptprogramm Steueranweisungen erzeugt und Rückmeldungen vom EIB entgegennimmt (nachfolgend „Master“ genannt).
- eine **RS232-Schnittstelle**
  - auf einem integrierten Controller (NXI-<xy>) (ab Ver. 1.0.20) oder
  - eine NXC-COM2 (Einschubkarte) in einem Kartenträger (NXF-<xy>) (ab Ver. 1.0.9) oder
  - eine NXC-COM2 (Einschubkarte) in einem Busdevice (NXM-COM2) zur Verwaltung der EIB-Meldungen und –Befehle.
- ein **EIB-Gateway** AXG-EIB (ab Ver. 3.01), zur physikalischen Verbindung mit dem EIB, enthält den Busankoppler BA und einen konfigurierbaren Paketfilter (nachfolgend „Gateway“)

Die kompletten Konfigurationsdaten des EIB liegen im Modul, nicht zwingend im Hauptprogramm. Bei jeder Änderung der Adressdaten und/oder Typen aktualisiert das Modul automatisch die Daten des Paketfilters im Gateway AXG-EIB. Dadurch ist gewährleistet, dass alle verwendeten Meldungen auch passieren dürfen (sofern sie EIB-seitig erzeugt werden...), und dass kein Datenmüll aus nicht (mehr) verwendeten Gruppen das System belastet. Bei jeder neuen Kontaktaufnahme mit dem Gateway wird die Gateway-intern generierte Prüfsumme mit einer gespeicherten Version im Modul verglichen und gegebenenfalls eine Neukonfiguration des Gateways eingeleitet.

Das Verfahren ähnelt AMX-seitig also mehr dem Umgang mit einem TouchPanel: sämtliche Aktionen laufen über Kanal-Nummern, Level-Nummern und Text-Nummern mit fester Bedeutung ab; das Aussehen der einzelnen Knöpfe und Seiten spielt für das Programm keine Rolle.

Ebenso wie beim TouchPanel werden aber auch hier die Anwendungsdaten in einem zwar batteriegepufferten, dennoch aber flüchtigen Speicher gehalten. Es empfiehlt sich daher, nach jeder Änderung der Konfiguration eine maschinenlesbare Version aufzubewahren, um im Notfall den Betrieb schnell wieder aufnehmen zu können.

## Softwarekomponenten



Das Programmpaket CTEIB4 umfaßt das eigentliche Modul und mehrere Programmbibliotheken (System\_Calls), die in eigene Programme eingebunden werden können.

Das Modul ist fertig übersetzt (kompiliert), und muß zur Installation nur noch ins Hauptprogramm eingebunden werden. (siehe Beispiel im Anhang B)

AMX-seitig stellt sich das Gateway durch die Verwendung des Moduls als "ganz normales" Busdevice dar, man könnte sich also auch "ganz normal" über Send\_Command/Send\_String-Befehle unterhalten.

Durch die Telegrammstruktur des EIB und die Vielfalt der möglichen Meldungen und Befehle sind jedoch die Anforderungen an die Empfangs- und Sendelogik inklusive Paket-Verarbeitung relativ hoch. Nicht zuletzt aus Performancegründen (Reaktionszeit) sollte daher auf die mitgelieferten Bibliotheksfunktionen zurückgegriffen werden.

Sie stellen eine Reihe von Funktionen bereit, die einfach - per System\_Call-Aufruf - in eigene Programme eingebunden werden können. Durch einen modularen Aufbau werden nur die wirklich benötigten Programmteile eingebunden, um die Ressourcen des Masters (Speicher, Rechenzeit, Buslast) möglichst zu schonen.

## Datenmodell

EIB stellt mehrere EIS-genormte Datentypen bereit, die verschiedenartigste Aufgaben wahrnehmen sollen. Neben simplen Ein/Aus-Schalt-Typen gibt es spezielle Datentypen für vorzeichenlose Zahlen in verschiedener Länge, vorzeichenbehaftete Zahlen, Fließkommawerte, Prozentangaben, Datum, Uhrzeit, Windrichtungen und alle möglichen sonstigen meßbaren Dinge. Das braucht in der (AMX-)Praxis kein Mensch, und CTEIB4 verwendet ein vereinfachtes Datenmodell, das die häufigsten Anwendungsfälle abdeckt, gleichzeitig aber völlig transparent ist, um beliebige Zugriffe zu erlauben.

Insgesamt stehen 6 verschiedene Datentypen zur Verfügung:

<b>Switch</b>	Binärer Schalter, Ein/Aus
<b>Dim4</b>	4-Bit Dimm-Aktor. Häufig verwendeter Ansteuertyp für Dimmer
<b>1Byte</b>	Sämtliche EIB-Datentypen mit einem Byte (8 Bit) Länge
<b>2Byte</b>	2-Byte-EIB-Typen (16 Bit)
<b>3Byte</b>	3-Byte-EIB-Typen (24 Bit)
<b>4Byte</b>	4-Byte-EIB-Typen (32 Bit)

Die beiden erstgenannten sind definierte EIB-Typen, die wohl mindestens 50% der Funktionen der meisten Anlagen "erschlagen" dürften.

Alle anderen EIB-Typen werden auf die Typen 1Byte..4Byte abgebildet - je nach Länge. Ein Element vom Typ 1Byte kann also eine 8 Bit Integer-Zahl sein (-128..+127), aber auch eine Prozentangabe (0..100).

Sämtliche Werte werden dabei 1:1 durchgereicht; wenn Sie sich also einen 32-Bit Fließkommawert nach IEEE-Norm (EIS9) vom EIB besorgen wollen – kein Problem.

Was dann aber die gelieferten "Roh"-Bytes bedeuten, ist Sache Ihrer Anwendungssoftware.

CTEIB4 ist ein reines Transportmedium, eine Interpretation der Daten findet nicht statt.

Der einzige Datentyp, der sich ohne "Vorbehandlung" in der AMX benutzen lässt, ist der Typ 8 Bit unsigned, also eine vorzeichenlose Ganzzahl im Bereich 0..255 - ein 1Byte-Typ. Praktischerweise, denn das ist genau der Typ, den die meisten Dimmer als Helligkeitswert zu Ansteuerung verwenden, bzw. melden.

Um beliebige andere EIB-Datentypen zu verwenden, müssen Sie sich also Methoden basteln, die jeweiligen Typen in Strings der entsprechenden Länge zu wandeln - möglichst in beide Richtungen, und möglichst mit Prüfung auf sinnvolle Werte...



## Gruppen, Adressen und Formate

Jede Komponente eines EIB-Systems, auf den von der AMX aus zugegriffen werden soll, ist EIB-seitig durch eine Adresse definiert, die Gruppenadresse. Über diese 15 Bit lange Adresse wird auf dem EIB signalisiert, welche(r) Akteur(en) angesprochen werden.

Das Gateway verwendet EIB-seitig nur die Gruppenadressen, eine vollständige und aktuelle Liste der relevanten Gruppenadressen ist also zur Einrichtung des Systems unbedingt notwendig.

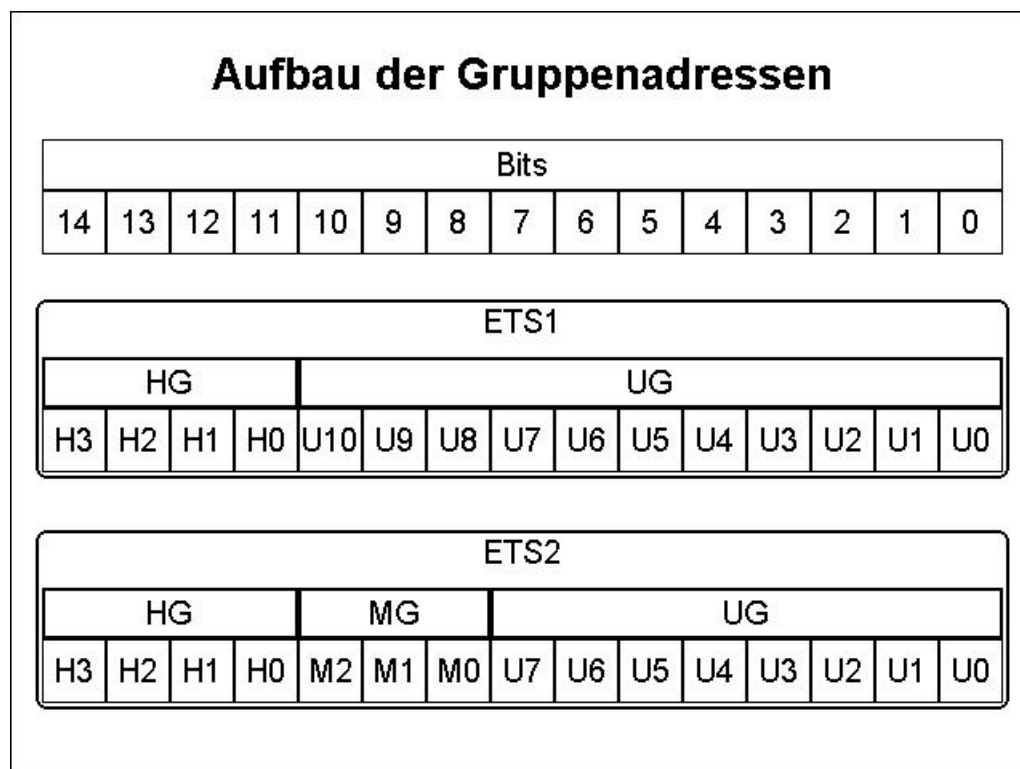
Das ETS1-Format stellt Gruppenadressen durch ein (dezimales) Zahlenpaar HG/UG dar, wobei die Hauptgruppe HG im Bereich 0..15 liegen muß, die Untergruppe UG im Bereich 0..2047.

Beispiel: 1/1027

Mit Einführung der ETS2-Software wurde ein neues Adressformat spezifiziert, das ein Zahlentripel HG/MG/UG zur Darstellung nutzt (MG: Mittelgruppe, HG: 0..15, MG: 0..7, UG: 0..255).

Beispiel: 1/4/3

**Ob die Gruppenadressen dabei in der alten ETS1-Notation (2-gruppig) oder in der neuen ETS2-Notation (3-gruppig) vorliegen, ist letztlich egal – das Modul verarbeitet beide Varianten. Der Unterschied liegt lediglich in der Notation der Adressen.**



Die beiden Formen können recht einfach in die jeweils andere Form umgerechnet werden:

Die Hauptgruppe (0..15) bleibt gleich:

$$HG_{ETS1} = HG_{ETS2}$$

Die 11 UG-Bits des ETS1-Formats verteilen sich im Verhältnis 3:8 auf MG und UG im ETS2-Format:

$$MG_{ETS2} = UG_{ETS1} \text{ div } 256$$

$$UG_{ETS2} = UG_{ETS1} \text{ mod } 256$$

bzw.

$$UG_{ETS1} = 256 \cdot MG_{ETS2} + UG_{ETS2}$$

**Bei der Angabe einer Adresse erkennt das Modul selbst das Format**, und schaltet die Formatierung der Ausgabe auf die zuletzt verwendete Form. Sollte eine Liste in der jeweils anderen Notation gewünscht werden, kann mit dem Befehl „ADR 2“ oder „ADR 3“ (s.u.) auf die entsprechende Darstellung umgeschaltet werden.

## Trigger

Es kann beim Umgang mit dem EIB gelegentlich vorkommen, daß Wertänderungen einzelner Aktoren nicht automatisch über den Bus mitgeteilt werden.

Ein typisches Beispiel sind sog. Kombiaktoren zur Lichtsteuerung, die typischerweise vier Gruppenadressen aufweisen:

- 1 Bit Schaltfunktion Ein/Aus
- 4 Bit Dimmfunktion
- 1 Byte Wertstellung
- 1 Byte Wertmeldung

Wohlgemerkt: alle vier Gruppenadressen beziehen sich auf ein und denselben Lichtkreis!

Wenn nun über eine der ersten drei Adressen auf den Dimmer zugegriffen wird, ändert sich die Helligkeit der angeschlossenen Lampe, und damit der Wert der vierten Gruppenadresse.

D.h., daß die 1Byte-Wertstellung keinesfalls immer den aktuellen Helligkeitswert enthalten muß, obwohl diese Adresse zum Dimmen mit Absolutwerten verwendet wird.

Die einzige Möglichkeit, an den aktuellen Helligkeitswert heranzukommen, besteht also darin, die vierte Adresse ausdrücklich nach ihrem aktuellen Wert zu fragen, also die Adresse zu pollen.

Wenn man nun aber möglichst immer den aktuellen Helligkeitswert der Lampe braucht - etwa um eine Bargraph-Anzeige auf einem TouchPanel anzusteuern - müßte bei jeder Änderung eines der ersten drei Werte der Wert der vierten Adresse abgefragt werden.

Das kann man natürlich "zu Fuß" programmieren, und bei jedem Feststellen einer Änderung per System\_Call 'EIB4\_NX Poll ...' diesen Wert anfragen. Das kann allerdings erheblichen Aufwand bedeuten:

für jede interessante Gruppenadresse muß ein alter Vergleichswert gespeichert, und dieser bei jedem Programmdurchlauf verglichen werden.

Wesentlich eleganter läßt sich das lösen, indem Trigger verwendet werden.

Dazu verwaltet das Modul für jede Gruppenadresse einen Verweis, welche Adresse bei einer Wertänderung gepollt werden soll. Bei der Definition der Adressen (oder auch später) wird also festgelegt, welche andere Adresse (in diesem Fall die zweite 1Byte-Adresse) abgefragt werden soll. Durch den Einbau dreier "WHEN...POLL"-Anweisungen (s.u.) ist also gewährleistet, daß immer der aktuelle Helligkeitswert zur Verfügung steht.

Natürlich kann unabhängig davon trotzdem eine Abfrage per System\_Call ausgelöst werden.

Derartige Konstruktionen, bei denen mehrere Aktoren auf eine "Summe" wirken, deren Wert aber ausdrücklich angefragt werden muß, scheint es in EIB-Systemen recht häufig zu geben, etwa in Szenen-Bausteinen.

Wenn Sie also damit zu kämpfen haben, daß manche Rückmeldungen einfach nicht stimmen oder "hängenbleiben", kann ein wohlplazierter Poll-Trigger manchmal Wunder wirken...

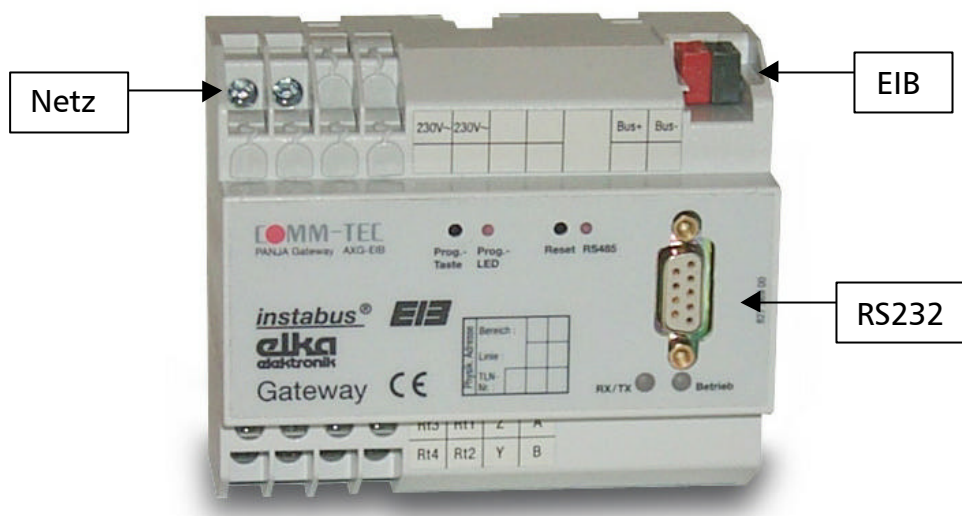
**Seien Sie aber bitte mit dem Einsatz aktiver Abfragen eher zurückhaltend** – die Bandbreite des EIB steht *allen* angeschlossenen Geräten *gemeinsam* zur Verfügung.

Gegen gelegentliches Abfragen einzelner Werte ist sicher nichts einzuwenden; alle erreichbaren Gruppen im Sekundentakt zu pollen ist aber Ressourcenverschwendung, und dürfte einigen Ärger nach sich ziehen...

# Installation und Inbetriebnahme

## Installation des Gateways

Das EIB-Gateway AXG-EIB wird auf einer Standard-Hutschiene im Schaltschrank montiert. Die rot-schwarzen Klemmen "Bus+" und "Bus-" werden mit dem EIB verbunden. Die Klemmen "230V~" werden mit der Netzversorgung verbunden.



Falls das Gateway EIB-seitig eine Hardware-Adresse zugewiesen bekommen soll, kann das über die "Prog-Taste" am Gateway und die ETS-Software geschehen. Bei Auslieferung des Gateways ist keine Adresse zugewiesen, das Gateway ist also am EIB nicht zu "sehen". In den meisten Fällen ist das auch nicht notwendig; bei Zugriff auf Aktoren, die "hinter" einem Linienkoppler sitzen, kann das aber notwendig sein. Diese Zuweisung sollte auf jeden Fall vom EIB-Installateur vorgenommen werden!

## Installation der RS232-Schnittstelle

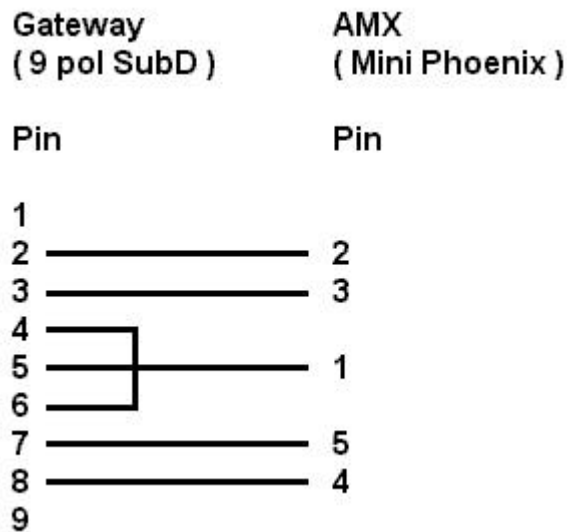
Der seriellen Schnittstelle wird eine eindeutige Adresse zugewiesen (Dip-Switches oder NetLinx Studio) und das Gateway über eine 5-adrige serielle Leitung mit einer maximalen Länge vom 20m angeschlossen. (Pin – Belegung s.u.)

Die notwendigen Einstellungen der seriellen Schnittstelle werden vom Modul erledigt. Es ist nicht notwendig, Baudrate, Parität, Handshake oder Sonstiges einzustellen.

## Verbindung Gateway-RS232 Schnittstelle

Zur Verbindung muß ein 5-adriges Kabel verwendet werden, da das Gateway zwingend mit Hardware-Handshake (RTS/CTS) arbeitet.

Anschlußbelegung (gültig für alle seriellen Schnittstellen der Netlinx Baureihe) :



## Einbinden des Moduls

Auf der mitgelieferten Diskette befindet sich das fertige Modul in der kompilierten Laufzeitversion, d.h. es ist nicht als NetLinx-Quellcode verfügbar. Diese Software muss nicht mehr wie in den Vorgängerversionen explizit eingespielt werden, sondern als Modul-Aufruf im Hauptprogramm nach der Sektion DEFINE\_START eingebunden werden. (siehe Beispiel im Anhang B)

## Gruppenadressen in das Modul eintragen

Sämtliche Gruppenadressen, auf die zugegriffen werden soll, müssen dem Program bekannt gemacht werden.

Nachdem geklärt ist, welche Gruppenadressen und Datentypen den einzelnen Funktionen zugeordnet sind, werden alle Funktionen auf die sechs unterstützten Datentypen "verteilt" und durchnummeriert.

Von jedem der sechs Datentypen verwaltet das Programm je 255 Stück, es kann also mit maximal 1530 Gruppenadressen Kontakt aufgenommen werden.

Das Modul generiert nun die Liste aller Adressen, Datentypen und Nummern, sendet sie ans Gateway und speichert sie. Alle Befehle von der AMX zum EIB und alle Meldungen vom EIB zur AMX laufen von nun an über die Angabe von Typ und Nummer.

Das hat den Vorteil, dass nicht bei jedem Zugriff umständlich die Adresse übermittelt werden muß

Die Übermittlung erfolgt aus einer direkt nach dem Modul eingebundenen Include – Datei durch Verwendung des System\_Calls 'EIB4\_NX Send Command'.

Nach Eintragen aller Adressen "programmiert" das Modul das Gateway automatisch; der früher notwendige Schritt, das Gateway über eine separate Windows-Software einzurichten, entfällt somit.

Ein Beispiel einer solchen Include-Datei zum Eintragen der Adressen findet sich im Anhang B.

## Versorgung des Moduls aus dem Hauptprogramm

Durch Einbinden der mitgelieferten System\_Calls in das Hauptprogramm werden Steuerfunktionen am EIB ausgelöst, und Meldungen über Zustandsänderungen in die entsprechenden Variablen verteilt.

Damit der NetLinx-Compiler die System\_Calls auch einbinden kann, müssen sie vor Benutzung in ein Verzeichnis auf der Festplatte kopiert werden. In *welches* Verzeichnis sie kopiert werden müssen, hängt davon ab, wie der Compiler installiert wurde.

Bei einer Standardinstallation der AMX-CD unter Windows98 oder NT liegen sämtliche System\_Calls im Verzeichnis „C:\Programme\Gemeinsame Dateien\AMXShare\SYCs“.

Kopieren Sie also bitte alle \*.LIB-Dateien aus dem Verzeichnis \syscall der mitgelieferten Diskette dorthin.

Sollten Sie bei der Installation ein anderes Unterverzeichnis zur Ablage der SYSTEM\_CALLS angegeben haben, stellen Sie bitte sicher, dass der Kompiler die System\_Calls auch finden kann.

## Kommunikation mit dem Modul

Das Modul verlangt nach zwei Adressangaben – zum Einen die Schnittstelle, an der das Gateway physikalisch angeschlossen ist, zum Anderen eine virtuelle Schnittstelle, über die die komplette Kommunikation abläuft. Zur Unterscheidung und Kennzeichnung werden folgende Begriffe verwendet :

- RS232 – Schnittstelle an der das Gateway physikalisch angeschlossen ist : **DEV**
- Virtuelle Schnittstelle zur Kommunikation : **vDEV**

Aus dem Programm wird nur die virtuelle Schnittstelle angesprochen, die physikalische Schnittstelle taucht nur in der Sektion DEFINE\_DEVICE und beim Aufruf des Moduls auf.

## System\_Calls

Dem Paket CTEIB4 liegen insgesamt 14 System\_Calls bei, die ins Hauptprogramm zur Ansteuerung eingebunden werden.

Zum Einlesen der häufigsten Datentypen wird kein eigener SYSTEM\_CALL mehr benötigt, die aktuellen Werte sind jederzeit in den Rückmelde Arrays verfügbar.

Für jeden der sechs Datentypen gibt es jeweils einen Call zum Setzen einer Adresse, z.B. System\_Call 'EIB4\_NX Set Switch', sowie einen zum aktiven Abfragen (pollen) einer Adresse, z.B. System\_Call 'EIB4\_NX Poll Switch'

Der System\_Call 'EIB4\_NX Send Command' - dient zum Absetzen von Befehlen an das Modul, etwa zur Definition der Gruppenadressen.

Der System\_Call 'EIB4\_NX EIS5 to String' rechnet den Datentyp EIS5 – ein 2-Byte Fließkommawert – in die entsprechende Dezimaldarstellung um.

Auf der Diskette liegen im Verzeichnis \SYSCALL folgende Dateien:

Dateiname	Klartextname	Bedeutung
CTEIB401_NX.LIB	'EIB4_NX Set Switch'	Switch-Wert setzen
CTEIB402_NX.LIB	'EIB4_NX Set Dim4'	Dim4-Wert setzen
CTEIB403_NX.LIB	'EIB4_NX Set 1Byte'	1Byte-Wert setzen
CTEIB404_NX.LIB	'EIB4_NX Set 2Byte'	2Byte-Wert setzen
CTEIB405_NX.LIB	'EIB4_NX Set 3Byte'	3Byte-Wert setzen
CTEIB406_NX.LIB	'EIB4_NX Set 4Byte'	4Byte-Wert setzen
CTEIB407_NX.LIB	'EIB4_NX Poll Switch'	Switch-Wert abfragen
CTEIB408_NX.LIB	'EIB4_NX Poll Dim4'	Dim4-Wert abfragen
CTEIB409_NX.LIB	'EIB4_NX Poll 1Byte'	1Byte-Wert abfragen
CTEIB410_NX.LIB	'EIB4_NX Poll 2Byte'	2Byte-Wert abfragen
CTEIB411_NX.LIB	'EIB4_NX Poll 3Byte'	3Byte-Wert abfragen
CTEIB412_NX.LIB	'EIB4_NX Poll 4Byte'	4Byte-Wert abfragen
CTEIB413_NX.LIB	'EIB4_NX Send Command'	Absetzen von Befehlen in das Modul
CTEIB414_NX.LIB	'EIB4_NX EIS5 to String'	Umrechnung EIS5 in String

## Setzen von Werten

Zum Setzen der Werte auf dem EIB dienen folgende sechs Calls:

```
System_Call 'EIB4_NX Set Switch' (vDEV, Wert, Nr)
System_Call 'EIB4_NX Set Dim4' (vDEV, Wert, Nr)
System_Call 'EIB4_NX Set 1Byte' (vDEV, Wert, Nr)
System_Call 'EIB4_NX Set 2Byte' (vDEV, Wert[2], Nr)
System_Call 'EIB4_NX Set 3Byte' (vDEV, Wert[3], Nr)
System_Call 'EIB4_NX Set 4Byte' (vDEV, Wert[4], Nr)
```

Parameter:

<i>vDEV</i>	Adresse der virtuellen Schnittstelle
<i>Wert</i>	Zu setzender Wert
<i>Nr</i>	Laufende Nummer des Aktors im Modul

Sie können an jeder Stelle im Programm eingesetzt werden.

## Rückmeldungen

Wertänderungen können im Hauptprogramm aus den jeweiligen Tabellen direkt gelesen werden. Das Einbinden eines SYSTEM\_CALLs ist hierzu nicht mehr notwendig. (früher SYSTEM\_CALL 'EIB Read Standard'(...))

Die Rückmeldungen stehen in den von Ihnen deklarierten Empfangstabellen z.B. :

SW[]	Tabelle mit den aktuellen Switch-Werten
D4[]	Tabelle mit den aktuellen Dim4-Werten
B1[]	Tabelle mit den aktuellen 1Byte-Werten
B2[][2]	Tabelle mit den aktuellen 2Byte-Werten
B3[][3]	Tabelle mit den aktuellen 3Byte-Werten
B4[][4]	Tabelle mit den aktuellen 4Byte-Werten

Zur Erinnerung : Diese Tabellen wurden als Integer-Arrays deklariert.

## Aktive Abfragen (Polling)

Aktive Abfragen können an jeder Stelle im Programm eingebaut werden.

```
System_Call 'EIB4_NX Poll Switch' (vDEV, Nr)
System_Call 'EIB4_NX Poll Dim4' (vDEV, Nr)
System_Call 'EIB4_NX Poll 1Byte' (vDEV, Nr)
System_Call 'EIB4_NX Poll 2Byte' (vDEV, Nr)
System_Call 'EIB4_NX Poll 3Byte' (vDEV, Nr)
System_Call 'EIB4_NX Poll 4Byte' (vDEV, Nr)
```

Parameter:

<i>vDEV</i>	Adresse der virtuellen Schnittstelle
<i>Nr</i>	Laufende Nummer des Aktors

Nach dem Absetzen einer Abfrage läuft das Programm weiter; beim Eintreffen der Wert-Meldung wandert diese über das Modul in die entsprechende Empfangstabelle.

## Befehle an das Modul

Zum Absetzen eines Befehls dient der folgende Call:

System\_Call 'EIB4\_NX Send Command' (vDEV, Befehl[50])

Parameter:

*vDEV* Adresse der virtuellen Schnittstelle  
*Befehl* Zeichenkette, die vom Modul interpretiert werden soll

Sämtliche Befehle lassen sich wahlweise per System\_Call 'EIB4\_NX Send Command' oder direkt als SEND\_COMMAND einbinden.

## Datenkonvertierung

2-Byte EIB-Typen werden recht häufig zur Darstellung von Analogwerten, wie z.B. Temperaturen verwendet. Dabei kommt der EIS-Datentyp EIS5 zum Einsatz, der in 16 Bit einen Fließkommawert codiert. Die 16 Bit teilen sich dabei wie folgt auf:

1 Bit	Vorzeichen	(S, 0..1)
4 Bit	Exponent	(E, 0..15)
11 Bit	Mantisse	(M, 0..2047)

Der eigentliche Wert ergibt sich aus  $-1^S * M * 0,01 * 2^E$

⇒ Wertebereich: +/- 670760,96

Zur Darstellung eines solchen Wertes (z.B. in einem Textfeld auf dem TouchPanel), bietet der System\_Call 'EIB4\_NX EIS5 to String' die Möglichkeit, einen 2-Byte-EIS5-Typ in die entsprechende Zifferndarstellung umzurechnen.

SYSTEM\_CALL 'EIB4\_NX EIS5 to String' (Bytes[2], String[10])

Parameter:

<i>Bytes[2]</i>	Binärer Wert, der umgerechnet werden soll
<i>String[10]</i>	Dezimaldarstellung, z.B. '-670760,96'

## “Terminal-Modus“

Zur Inbetriebnahme einer AMX-Installation mit EIB-Anbindung ist es manchmal unabdingbar, einzelne Befehle absetzen zu können, ohne das komplette Hauptprogramm des Masters neu einzuspielen und initialisieren zu müssen.

Das Modul bietet hier die Möglichkeit alle nachfolgend beschriebenen Befehle als SEND\_COMMAND anzunehmen und umzusetzen. Es ist nun also als Device mit eigenem Befehlssatz ansprechbar. Weiterhin sind einige dieser Befehle auf Kanäle des Moduls abgebildet, und können so mit einfachen ON / OFF / PULSE – Anweisungen ausgeführt werden.

## Befehlssatz des Moduls

Um nicht eine Vielzahl von System\_Calls zur Initialisierung des Moduls herumschleppen zu müssen, erfolgt die Befehlsgabe an das Modul und die Rückmeldung eventueller Fehler und Statusmeldungen im Klartext. Meldungen vom Modul werden automatisch in das Terminalfenster umgeleitet, (Text-) Befehle an das Modul werden mit dem 'EIB4\_NX Send Command' abgesetzt. Voraussetzungen zur Ausgabe dieser Meldungen in das Terminalfenster sind :

- im Terminal ist die Debug-Ausgabe freigegeben
- die EIB Rückmeldungen sind eingeschaltet
- **MSG ON** und
- **ON[vDEV,599]**

Diese EIB-Rückmeldungen dienen ausschliesslich zu Diagnosezwecken, und sollten zur Laufzeit abgeschaltet werden. Die ausgegebenen Meldungen werden durch die vorangestellte Zeichenkette “EIB:“ kenntlich gemacht

Bsp.: MSG ON → Debug-Ausgaben freigegeben  
ON[vDEV,599] → Klartextmeldungen EIB einschalten  
OFF[33003:1:0,599] → Klartextmeldungen EIB ausschalten (hier absolute Adressangabe)

Befehle bestehen aus einem oder mehreren Wörtern, die durch Leerzeichen (Space, \$20) getrennt werden, Groß- und Kleinschreibung ist dabei egal.

Es sind nur die Zeichen {'0'..'9','A'..'Z','a'..'z',' ','/','\$'} erlaubt.

Zur kompakten Bezeichnung der Datentypen gelten folgende Abkürzungen:

SW	Switch
D4	Dim4
1B	1Byte
2B	2Byte
3B	3Byte
4B	4Byte

Verwendung : hier wird angezeigt, wofür der betreffende Befehl üblicherweise verwendet wird.

Initialisierung	-	Include Datei zum Aufbau der Tabellen
Laufzeit	-	Hauptprogramm
Diagnose	-	Inbetriebnahme

Folgende Befehle stehen zur Verfügung

## ADD – Gruppenadresse eintragen

Verwendung : Initialisierung

Kanal : --

Der ADD-Befehl trägt eine Gruppenadresse (neu) ein, schafft also die Verbindung zwischen EIB-Gruppenadresse und dem Typ/Nummer-Paar in der AMX. Sind Gruppenadresse oder Typ/Nr bereits verwendet, wird der alte Eintrag ersetzt.

Syntax: SEND\_COMMAND vDEV,'ADD <Typ> <Nr> <Adresse>'

Bsp.: SEND\_COMMAND vDEV,'ADD SW 1 08/15  
Schalter Nr. 1 hat Adresse 08/15  
SEND\_COMMAND vDEV,'ADD 4B 255 15/1234  
4-Byte-Aktor Nr. 255 hat Adresse 15/1234



## ADR – Format der Adressdarstellung umschalten

Verwendung : Diagnose

Kanal : 593 – ON → ETS2 - Notation  
OFF → ETS1 - Notation

Mit ADR kann das Darstellungsformat für Gruppenadressen zwischen der alten ETS1-Notation (HG/UG) und der neuen ETS2-Notation (HG/MG/UG) umgeschaltet werden.  
Die Umschaltung wird nur für LIST-Ausgaben benötigt; bei der Adressangabe schaltet das Modul automatisch auf die zuletzt verwendete Notation um.

Syntax: SEND\_COMMAND vDEV,'ADR <Format>'

Bsp.: SEND\_COMMAND vDEV,'ADR 2      Ausgabe in ETS1-Notation HH/UUUU  
Oder  
ON[vDEV,593]  
  
SEND\_COMMAND vDEV,'ADR 3      Ausgabe in ETS2-Notation HH/M/UUU

## DELTA – Differentialübertragung Ein/Aus

Verwendung : Diagnose (Laufzeit)

Kanal : 594

Mit DELTA kann eingestellt werden, ob das Modul nur dann ein Paket an das Hauptprogramm sendet, wenn sich der Wert des Aktors geändert hat, oder ob jedesmal ein Paket gesendet wird, wenn überhaupt ein Paket empfangen wird – egal ob der Wert gleich bleibt oder nicht.  
Normalerweise werden nur Wertänderungen mitgeteilt, um möglichst wenig übertragen zu müssen.  
Es kann in Ausnahmefällen(!) aber sinnvoll sein, Meldungen über *alle* empfangenen Wert-Pakete an das Hauptprogramm zu übermitteln.  
Die RESEND-Logik (s.u.) bleibt davon unberührt.

Syntax: SEND\_COMMAND vDEV,'DELTA <Zustand>'

Bsp.: SEND\_COMMAND vDEV,'DELTA ON' (default)  
Nur Wertänderungen an das Hauptprogramm senden  
oder  
OFF[vDEV,594]  
  
SEND\_COMMAND vDEV,'DELTA OFF'  
Alle empfangenen Werte weiterleiten  
oder  
ON[vDVD,594]

## LIST – Anzeige der bestehenden Verknüpfungen

Verwendung : Diagnose

Kanal : 581 für SW  
582 für D4  
583 für 1B  
584 für 2B  
585 für 3B  
586 für 4B

LIST erzeugt eine Klartextliste der verwendeten Gruppenadressen und aktuellen Werte je Typ.

Syntax: SEND\_COMMAND vDEV,'LIST <Typ> [<StartNr>] [<EndNr>']  
Oder  
PULSE[vDEV,<Kanal>]

Die Angaben von Start und Ende sind optional, werden beide weggelassen (oder wird der betreffende Kanal verwendet) erfolgt die Ausgabe der Liste mit allen Einträgen.

Bsp.: SEND\_COMMAND vDEV,'LIST SW 4'      Wie ist Schalter Nr. 4 definiert?  
=> Antwort :    EIB : Switch #4: 1/0/4 Val: \$0  
  
PULSE[vDEV,586]      Zeige Definition aller 4-Byte-Typen  
=> Antwort :    EIB : Dim4 #1: 1/0/112 Val: \$0 => polls 1Byte #2  
                 EIB : Dim4 #2: 1/0/122 Val: \$0 => polls 1Byte #4

## NO POLL - Löschen eines Poll-Triggers

Verwendung: Diagnose (Laufzeit)

Kanal : --

Mit NOPOLL werden sämtliche automatischen Abfragen einer Adresse gelöscht.

Syntax: SEND\_COMMAND vDEV,'NOPOLL <Typ> <Nr>'

Bsp.:      SEND\_COMMAND vDEV,'NOPOLL SW 13'  
             SEND\_COMMAND vDEV,'NOPOLL 1B 217'

## POLL - Aktive Wertabfrage

Verwendung: Diagnose

Kanal : --

Mit dem POLL-Kommando wird ein Telegramm auf den EIB abgesetzt, um den aktuellen Wert eines Aktors zu erfragen (dessen Lese-Flag dazu natürlich gesetzt sein muss).

Dieser Befehl ist nur für den Terminal-Modus gedacht; für aktive Abfragen aus dem Hauptprogramm verwenden Sie bitte die entsprechenden System\_Calls.

Syntax: SEND\_COMMAND vDEV,'POLL <Typ> <Nr>'

Bsp.:	SEND_COMMAND vDEV,'POLL SW 2'	Wert von Schalter Nr.2 erfragen
	SEND_COMMAND vDEV,'POLL 2B 128'	Welchen Wert hat 2-Byte Nr. 128 ?

**RESEND - Alle Werte erneut schicken**

Verwendung: Diagnose

Kanal : 595

Nach einem RESEND-Befehl sendet das Modul die aktuellen Werte aller eingetragenen Gruppen an das Hauptprogramm, um den Speicherinhalt zu synchronisieren.

Syntax:        SEND\_COMMAND vDEV,'RESEND'  
                 Oder  
                 PULSE[vDEV,595]

## RESET - Gateway-Reset auslösen

Verwendung : Diagnose

Kanal : 598

Durch einen Gateway-Reset werden alle anstehenden Pakete im Gateway gelöscht, der Busankoppler initialisiert, und alle markierten Gruppenadressen werden abgefragt (siehe WHEN...POLL)

Syntax:        SEND\_COMMAND vDEV,'RESET'  
                 Oder  
                 PULSE[vDEV.598]

## SET - Gruppenadresse schreiben

Verwendung: Laufzeit, Diagnose

Kanal : --

Mit dem SET-Befehl kann im Terminal-Modus direkt auf einzelne Adressen zugegriffen werden, um die Funktion zu testen. Je nach angesprochenem Typ müssen bis zu vier (Byte-)Werte entweder als Dezimalzahlen oder – mit vorangestelltem \$-Zeichen – als Hexadezimalzahlen übergeben werden. Für den Schreibzugriff aus dem Hauptprogramm heraus verwenden Sie bitte die entsprechenden System Calls.

Syntax: `SEND COMMAND vDEV,'SET <Typ> <Nr> Wert [<Wert2> [<Wert3> [<Wert4>]]'`

Bsp.:	SEND_COMMAND vDEV,'SET SW 1 0'	Schalter Nr. 1 ausschalten
	SEND_COMMAND vDEV,'SET D4 3 \$E'	Dimmer 3 auf 14 setzen
	SEND_COMMAND vDEV,'SET 4B 217 1 2 3 4'	4-Byte-Wert Nr. 217 auf {1 2 3 4 } setzen

## STATUS – Statusinformation anzeigen

Verwendung : Diagnose

Kanal : 596

STATUS erzeugt eine Liste mit den wichtigsten Zustandsinformationen zum Betrieb des Gateways.

Syntax:        SEND\_COMMAND vDEV,'STATUS'  
                 Oder  
                 PULSE[vDEV,596]

=> Antwort:    EIB : Gateway Version: 3.2  
                 EIB : EEPROM Check: \$E922  
                 EIB : ADR mode: 3 groups (H/MM/UUU)  
                 EIB : DELTA mode: ON (report changes)  
                 EIB : No errors reported.

## UPLOAD - Gateway-Filtertabelle neu laden

Verwendung : Diagnose

Kanal : 597

Mit dem UPLOAD-Befehl kann die Filtertabelle im Gateway erneut geschrieben werden. Dies ist normalerweise nicht notwendig, da das Modul automatisch erkennt, ob sich Gruppenadressen geändert haben, oder ein neues (noch nicht programmiertes) Gateway angeschlossen wurde.

Syntax:        SEND\_COMMAND vDEV,'UPLOAD'  
                 Oder  
                 PULSE[vdveib,597]

## WATCH - Überwachung einer Adresse

Verwendung : Diagnose

Kanal : --

Mit WATCH kann der Zustand einer Gruppenadresse überwacht werden, um die Funktion eines Aktors oder Sensors während der Installation zu prüfen.

Wird ein Telegramm der überwachten Adresse empfangen, erzeugt das Modul eine Klartextmeldung mit dem alten und dem neuen Wert.

Syntax:        SEND\_COMMAND vDEV,'WATCH <Typ> <Nr>'  
                 bzw.  
                 SEND\_COMMAND vDEV,'WATCH OFF'

Bsp.:	SEND_COMMAND vDEV,'WATCH SW 4'	Überwache Schalter Nr. 4
	SEND_COMMAND vDEV,'WATCH 2B 23'	Überwache 2-Byte-Wert Nr. 23
	SEND_COMMAND vDEV,'WATCH OFF'	Überwachung abschalten

Ablauf

Eingabe :	SEND_COMMAND vDEV,'Watch SW 4'
Ausgabe :	EIB : OK, watching SW #4
	EIB : Change: SW #4: \$0 => \$1
	EIB : Change: SW #4: \$1 => \$0
	EIB : Change: SW #4: \$0 => \$1
	EIB : Change: SW #4: \$1 => \$0
Eingabe :	SEND_COMMAND vDEV,'Watch OFF'
	EIB : WATCH mode set off

## WHEN .. POLL - Setzen eines Poll-Triggers

Verwendung: Laufzeit, Diagnose

Kanal : --

Mit WHEN...POLL kann definiert werden, dass bei Wertänderungen eines Aktors automatisch ein anderer Aktor abgefragt („gepollt“) werden soll. Das ist z.B. bei Kombiaktoren hilfreich, bei denen der Schreibzugriff auf eine Adresse (Dimmen Auf/Ab) den Wert einer anderen Adresse ändert (Helligkeit), ohne dass diese ihren Wert automatisch kundtut.

Ein Sonderfall ist „WHEN START POLL <Typ> <Nr>“. Ist dieses Attribut für eine Adresse gesetzt, wird der aktuelle Wert automatisch nach einem Gateway-Reset eingelesen – unabhängig von sonstigen Verknüpfungen mit dieser Adresse.

Syntax:       SEND\_COMMAND vDEV, 'WHEN <Typ> <Nr> POLL <Typ> <Nr>'  
                  oder  
                  SEND\_COMMAND vDEV, 'WHEN START POLL <Typ> <Nr>'

Bsp.:	SEND_COMMAND vDEV,'WHEN SW 1 POLL 1B 4'	Bei Änderungen von Schalter Nr. 1 automatisch den Wert des 1-Byte-Aktors Nr. 4 erfragen
	SEND_COMMAND vDEV,'WHEN D4 17 POLL 4B 254'	
	SEND_COMMAND vDEV,'WHEN START POLL SW 123'	Nach Gateway-Reset Schalter 123 einlesen
	SEND_COMMAND vDEV,'WHEN START POLL 1B 27'	

## Levels

Alle 1Byte-Adressen stehen als Levels zur Verfügung (Levels 1..255); bei jeder Änderung des Wertes einer dieser Gruppenadressen wird der aktuelle Wert als Level an das Hauptprogramm übermittelt, um z.B. eine Bargraph-Anzeige anzusteuern.

Der umgekehrte Weg, mittels `SEND_LEVEL` auf die Aktoren zuzugreifen, ist nicht möglich.

## Anhang A

### Werte für Dim4-Aktoren

Der Call 'EIB Set Dim4' reicht die übergebenen Werte direkt an den Aktor weiter – die zurückgemeldeten Werte entsprechen nun den tatsächlichen Aktor-Werten (im Bereich 0..15), die auch gesendet wurden.

Die Bedeutung der 16 möglichen Werte ist dabei durch den EIB-Datentyp EIS2 festgelegt:

Bits 0..2 legen die Dimmgeschwindigkeit fest (0: Stop, 7: Max)

Bit 3 legt die Dimmrichtung fest (0: Abwärts, 1: Aufwärts)

Daraus ergeben sich folgende Bedeutungen:

0	Stop	8	Stop
1	Langsam dunkler	9	Langsam heller
2	...	10 (\$A)	...
3	...	11 (\$B)	...
4	dunkler	12 (\$C)	heller
5	...	13 (\$D)	...
6	...	14 (\$E)	...
7	Schnell dunkler	15 (\$F)	Schnell heller

# Anhang B

## Beispiel: Hauptprogramm

### PROGRAM\_NAME= 'HAUPTPROGRAMM

```
(*{PS_SOURCE_INFO(PROGRAM STATS) *)
(*****
(* FILE CREATED ON: 00/00/0000 AT: 00:00:00 *)
(*****
(* FILE_LAST_MODIFIED_ON: 00/00/0000 AT: 00:00:00 *)
(*****
(* ORPHAN_FILE_PLATFORM: 1 *)
(*****
(*!!FILE REVISION: *)
(* REVISION DATE: 00/00/0000 *)
(* *)
(* COMMENTS: *)
(* *)
(*****
(*}}PS_SOURCE_INFO *)
(*****

(*****
(* System Type : Netlinx *)
(*****
(* REV HISTORY: *)
(*****
```

### DEFINE\_DEVICE

```
dvEIB = 5001:1:0 // physikalische Schnittstelle
vdvEIB = 33003:1:0 // virtuelle Schnittstelle
dvTP = 128:1:0
```

### DEFINE\_CONSTANT

```
Sw_Licht1 = 1 // laufende Switch-Nummer Deckenlicht
Sw_Licht2 = 2 // laufende Switch-Nummer Pultlicht
Sw_Licht3 = 3 // laufende Switch-Nummer Pultlicht
Sw_Dimmer = 4 // Dimmer Ein/Aus

D4_Dimmer = 1 // relativ Dimmen

B1_Dim_Set = 1 // Dimmer setzen
```

### DEFINE\_VARIABLE

```
INTEGER nEIB_SWITCH[255] // Aktuelle Werte der EIB-Switches
INTEGER nEIB_Dim4[255] // Aktuelle Werte Dim4
INTEGER nEIB_1BYTE[255] // Aktuelle Werte 1Byte
INTEGER nEIB_2BYTE[255][2] // Aktuelle Werte 2Byte
INTEGER nEIB_3BYTE[255][3] // Aktuelle Werte 3Byte
INTEGER nEIB_4BYTE[255][4] // Aktuelle Werte 4Byte
```

### DEFINE\_START

```
DEFINE_MODULE 'CTEIB4_NXmod' GATEWAY_1 (vdvEIB,dvEIB,
nEIB_SWITCH,nEIB_DIM4, nEIB_1BYTE,
nEIB_2BYTE, nEIB_3BYTE,nEIB_4BYTE)

INCLUDE 'EIB_Table_NX.AXI' // → siehe Beispiel EIB_Table
```

## DEFINE\_EVENT

```
BUTTON_EVENT[dvTP,1]                                // Licht 1 EIN
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set Switch' (vdvEIB,1,Sw_Licht1)
    }
}
BUTTON_EVENT[dvTP,2]                                // Licht 1 AUS
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set Switch' (vdvEIB,0,Sw_Licht1)
    }
}
BUTTON_EVENT[dvTP,3]                                // Licht 3 umschalten
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set Switch' vdvEIB,NOT[vdvEIB,Sw_Licht2],Sw_Licht2)
    }
}
BUTTON_EVENT[dvTP,4]                                // Licht 4 10sec EIN
{
    PUSH :
    {
        CANCEL_WAIT 'LICHT3'
        SYSTEM_CALL 'EIB4_NX Set Switch' (vdvEIB,1,Sw_Licht3)
        WAIT 100 'Licht3'
        SYSTEM_CALL 'EIB4_NX Set Switch' (vdvEIB,0,Sw_Licht3)
    }
}
BUTTON_EVENT[dvTP,5]                                // Dimmer AUS
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set Switch' (vdvEIB,0,Sw_Dimmer)
    }
}
BUTTON_EVENT[dvTP,6]                                // Dimmer auf 25%
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set 1Byte' (vdvEIB,64,B1_Dim_Set)
    }
}
BUTTON_EVENT[dvTP,7]                                // Dimmer auf 50%
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set 1Byte' (vdvEIB,128,B1_Dim_Set)
    }
}
BUTTON_EVENT[dvTP,8]                                // Dimmer auf 75%
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set 1Byte' (vdvEIB,192,B1_Dim_Set)
    }
}
BUTTON_EVENT[dvTP,9]                                // Dimmer auf 100%
{
    PUSH :
    {
        SYSTEM_CALL 'EIB4_NX Set Switch' (vdvEIB,1,Sw_Dimmer)
    }
}
```

```

    }
    BUTTON_EVENT[dvTP,10] // heller
    BUTTON_EVENT[dvTP,11] // dunkler
    {
    PUSH :
    {
    IF (BUTTON.INPUT.CHANNEL = 10)
    {
    SYSTEM_CALL 'EIB4_NX Set Dim4' (vdvEIB,10,D4_Dimmer)
    }
    ELSE
    {
    SYSTEM_CALL 'EIB4_NX Set Dim4' (vdvEIB,2,D4_Dimmer)
    }
    }
    RELEASE : // stopp
    {
    SYSTEM_CALL 'EIB4_NX Set Dim4' (vdvEIB,0,D4_Dimmer)
    }
    }
}

```

## DEFINE\_PROGRAM

```

[dvTP,1] = nEIB_SWITCH[Sw_Licht1] // Rueckmeldung Licht 1 EIN
[dvTP,2] = NOT(nEIB_SWITCH[Sw_Licht1]) // Rueckmeldung Licht 1 AUS
[dvTP,3] = nEIB_SWITCH[Sw_Licht2] // Rueckmeldung LICHT 2 EIN
[dvTP,4] = nEIB_SWITCH[Sw_Licht3] // Rueckmeldung LICHT 3 EIN

[dvTP,10] = ((nEIB_Dim4[D4_Dimmer] >= 9) and (nEIB_Dim4[D4_Dimmer] <= 15))
[dvTP,11] = ((nEIB_Dim4[D4_Dimmer] >= 1) and (nEIB_Dim4[D4_Dimmer] <= 7))

```



## Beispiel : Include – Datei zum Eintragen der Gruppen

Die folgende Include-Datei zeigt die Übertragung der Gruppendefinitionen in das Modul.  
In einer Schleife wird alle 200ms jeweils ein Befehl mit dem System\_Call 'EIB4\_NX Send Command' an das Modul geschickt, um entweder eine Gruppenadresse mit einer Nummer und einem Typ zu verknüpfen, oder eine automatische Wertabfrage („Poll-Trigger“) zu definieren.

**PROGRAM\_NAME=EIB\_Table\_NX'**

### DEFINE\_VARIABLE

```
INTEGER    X                                // Zaehler
CHAR       Cmd[50]                          // Aktueller Befehl
```

### DEFINE\_START

```
X = 0                                // bei Reset anhalten
WAIT 50                               // 5 Sekunden warten
X = 1                                // Erste Gruppe eintragen
```

### DEFINE\_PROGRAM

```
WAIT 2                                // alle 200ms pruefen
{
  IF(X)                                // Etwas einzutragen ?
  {
    SWITCH (X)
    {
      CASE 1 : Cmd = 'ADD SW 1 1/1'      // Schalter 1: Adresse 1/1
      CASE 2 : Cmd = 'ADD SW 2 1/2'
      CASE 3 : Cmd = 'ADD SW 3 1/0/3'
      CASE 4 : Cmd = 'ADD SW 4 1/22'      // Dimmer Ein/Aus
      CASE 5 : Cmd = 'ADD D4 1 1/23'      // Relativ dimmen
      CASE 6 : Cmd = 'ADD 1B 1 1/24'      // Dimmer setzen
      CASE 7 : Cmd = 'ADD 1B 2 1/25'      // aktueller Dimmerwert

      CASE 8 : Cmd = 'WHEN START POLL 1B 2' // nach Reset lesen
      CASE 9 : Cmd = 'WHEN SW 4 POLL 1B 2' // bei Schalter 4 lesen
      CASE 10 : Cmd = 'WHEN D4 1 POLL 1B 2' // bei Dimmen lesen
      CASE 11 : Cmd = 'WHEN 1B 1 POLL 1B 2' // bei Setzen sowieso

      DEFAULT : Cmd = ""                 // Schluss
    }

    IF(Cmd <> "")      (* Etwas zu Senden ? *)
    {
      SYSTEM_CALL 'CTEIB4_NX Send Command' (vdvEIB,Cmd) // an das Modul
      X = X + 1                                           // naechster Eintrag
    }
  }
  ELSE                                // Ende erreicht
  {
    X = 0                                           // Zaehler aus
  }
}
```

# Anhang C

## Fehlermeldungen

In einigen Situationen erzeugt das Modul Meldungen, die ins Terminalfenster umgeleitet werden können. Voraussetzung zur Ausgabe dieser Fehlermeldungen sind :

- im Terminal ist die Debug-Ausgabe eingeschaltet
- die EIB-Rückmeldungen sind eingeschaltet
- **MSG ON** und **ON[vDEV,599]**

Diese Meldungen werden durch die vorangestellte Zeichenkette „EIB:“ kenntlich gemacht, und enthalten Klartextmeldungen, die im folgenden alphabetisch sortiert sind:

- **"Adress <Adresse> not used"** Das Gateway hat den Zustandswechsel einer Adresse gemeldet, die nicht verwendet wird. Abhilfe: Gateway neu laden („UPLOAD“)
- **"BA Busy: No connection to EIB"** Das Gateway hat keine Verbindung zum EIB.  
Mögliche Ursachen :
  - EIB nicht angeschlossen
  - Busleitung defekt
  - Busankoppler des Gateways wird vom EIB durch Telegrammflut lahmgelegt
- **"BA Error: Internal Error"** Interner Fehler des Gateway-Busankopplers
- **"BA-Layer: EIB problem"** Probleme beim Zugriff auf EIB
- **"Bad ADD address: <Adresse>"** Die im Befehl ADD angegebene Gruppenadresse ist ungültig
- **"Bad ADD number: <Nr>"** Die im Befehl ADD angegebene Nummer ist ungültig (0 oder >255)
- **"Bad ADD parm count"** Der ADD-Befehl ist unvollständig oder enthält zu viele Argumente
- **"Bad ADD type: <Typ>"** Im Befehl ADD angegebener Typ ist ungültig (SW, D4, 1B, 2B, 3B, 4B)
- **"Bad ADR parm count"** Der ADR-Befehl enthält keines oder mehr als ein Argument
- **"Bad LIST parm count"** Die Parameter-Anzahl des LIST-Befehls ist falsch
- **"Bad LIST start <Nr>"** Die LIST-Befehl enthält eine ungültige Start-Nummer (0 oder >255)
- **"Bad LIST type: <Typ>"** Im Befehl LIST angegebener Typ ist ungültig (SW, D4, 1B, 2B, 3B, 4B)
- **"Bad NOPOLL parm count"** Die Parameter-Anzahl des NOPOLL-Befehls ist falsch
- **"Bad POLL number: <Nr>"** Die im Befehl POLL angegebene Nummer ist ungültig (0 oder >255)
- **"Bad POLL parm count"** Die Parameter-Anzahl des POLL-Befehls ist falsch
- **"Bad POLL trigger - <Typ> <Nr> unused"** Der im WHEN...POLL-Befehl angegebene Trigger wird nicht verwendet
- **"Bad POLL type: <Typ>"** Im Befehl POLL angegebener Typ ist ungültig (SW, D4, 1B, 2B, 3B, 4B)
- **"Bad Poll <Typ> #<Nr> unused"** Das im Befehl POLL angegebene Element wird nicht verwendet
- **"Bad SET - <Typ> <Nr> unused"** Das im Befehl SET angegebene Element wird nicht verwendet
- **"Bad SET <Typ> - illegal value"** Die angegebenen Werte sind nicht zulässig
- **"Bad SET <Typ> - wrong parm len"** Die Parameter-Anzahl des SET-Befehls passt nicht zum angegebenen Typ
- **"Bad SET number <Nr>"** Die im Befehl SET angegebene Nummer ist ungültig (0 oder >255)
- **"Bad SET parm count"** Die Parameter-Anzahl des SET-Befehls ist falsch
- **"Bad SET type <Typ>"** Der im Befehl SET angegebene Typ ist ungültig (SW, D4, 1B, 2B, 3B, 4B)
- **"Bad WHEN - POLL expected"**
- **"Bad WHEN - START expected"** Der Aufbau des WHEN...POLL –Befehls ist falsch
- **"Bad WHEN dest number: <Nr>"**
- **"Bad WHEN number: <Nr>"** Die im Befehl WHEN...POLL angegebene Nummer ist ungültig (0 oder >255)
- **"Bad WHEN dest type: <Typ>"**
- **"Bad WHEN type: <Typ>"** Der im Befehl WHEN...POLL angegebene Typ ist ungültig (SW, D4, 1B, 2B, 3B, 4B)
- **"Bad WHEN parm count"** Die Parameter-Anzahl des WHEN...POLL -Befehls ist falsch
- **"Gateway Timeout"** Das Gateway hat mehr als zwei Sekunden nicht geantwortet (Verkabelung?)
- **"NAK: No EIB Device at <Adresse>"** Auf die Adresse reagiert kein Gerät am EIB
- **"Poll queue overflow"** Das Modul hat mehr Poll-Anfragen, als das Gateway bearbeiten kann
- **"Reconnect"** Das Modul versucht das Gateway neu zu initialisieren, Ursache z.B. zu viele NAKs vom EIB oder fehlgeschlagene Versuche die Kommunikation aufzubauen.
- **"Tx: Transmit buffer overflow"** Der Sendepuffer des Gateways ist voll
- **"Unknown: <Befehl>"** Das Modul kennt den angegebenen Befehl nicht