Handbuch Enertex® EibPC

Voraussetzung Enertex® EibPC: Patch 3.000 oder höher
Enertex® EibStudio: 3.000 oder höher

Hinweis

Der Inhalt dieses Dokuments darf ohne vorherige schriftliche Genehmigung durch die Enertex® Bayern GmbH in keiner Form, weder ganz noch teilweise, vervielfältigt, weitergegeben, verbreitet oder gespeichert werden.

Enertex® ist eine eingetragene Marke der Enertex® Bayern GmbH. Andere in diesem Handbuch erwähnte Produkt- und Firmennamen können Marken oder Handelsnamen ihrer jeweiligen Eigentümer sein.

Dieses Handbuch kann ohne Benachrichtigung oder Ankündigung geändert werden und erhebt keinen Anspruch auf Vollständigkeit oder Korrektheit.

Enertex® Bayern GmbH – Ebermannstädter Straße 8 - 91301 Forchheim - mail@enertex.de

Inhalt

Hinweise

Für den Betrieb des Enertex® EibPC benötigen Sie:

- Eine FT 1.2 kompatible KNX™-RS232 Schnittstelle **oder** eine Eibnet/IP Schnittstelle oder einen Router (siehe Seite 16 für kompatible Schnittstellen)
- Eine Stromversorgung DC, Leistung mind. 1,7 Watt Ausgangsleistung, 20 bis 30 VDC
- Für die Programmierung einen Windows® 7, Windows 8.1 , Mac OSX oder Linux® PC mit LAN Anschluss.

Bitte beachten Sie die Inbetriebnahmehinweise auf Seite 16 sowie die Reihenfolge beim Zuschalten der FT1.2 bzw. der IP Schnittstelle auf Seite16.

Hilfefunktion

Die Hilfefunktion wird mit Hilfe des Acrobat Readers realisiert. Dieses Dokument können Sie entweder direkt downloaden (www.eibpc.com) oder durch Drücken von F1 im Enertex® EibStudio öffnen.

Sie verfügen im Enertex® EibStudio über eine Kontexthilfe. Markieren Sie das Schlüsselwort, zu dem Sie Hilfe benötigen und drücken Sie F1 (Macintosh User: CMD-?). Sie werden dann direkt auf die entsprechende Stelle im Handbuch verwiesen. Enertex® EibStudio selbst verfügt über eine Autovervollständigung (STRG+Leertaste) und Syntaxhervorhebung, welche per Default aktiv ist.

Dieses pdf-Dokument nutzt die Gliederung in Teilabschnitte des Acrobat Readers, die auch als „Lesezeichen“ bezeichnet werden. Klicken Sie am linken Rand auf die Registerkarte Lesezeichen um diese anzuzeigen.

Dieses Dokument ist verlinkt. Wenn Sie eine Seitenangabe lesen, z.B. „mehr hierzu auf Seite 14“, klicken Sie einfach auf die Seitennummer und der Reader springt an diese Stelle.

Sicherheitshinweise

- Einbau und Montage elektrischer Geräte dürfen nur durch Elektrofachkräfte erfolgen.
- Beim Anschluss von KNX/EIB-Schnittstellen werden Fachkenntnisse durch KNX™-Schulungen vorausgesetzt.
- Bei Nichtbeachtung der Anleitung können Schäden am Gerät, Brand oder andere Gefahren auftreten.
- Diese Anleitung ist Bestandteil des Produkts und muss beim Endanwender verbleiben.
- Das Gerät darf nicht für Anwendungen mit Gefährdungspotential eingesetzt werden (Fehlfunktionen, möglicher Ausfall der Zeitschaltuhren, etc.).

Lizenzhinweise

- Mit dem Erwerb des Enertex® EibPC erwerben Sie eine Lizenz zur Nutzung von Enertex® EibStudio. Das Enertex® EibStudio und sämtliche auch eigenständig arbeitenden Komponenten dürfen nur zur Programmierung eines Enertex® EibPC genutzt werden.
- Der Hersteller haftet nicht für Kosten oder Schäden, die dem Benutzer oder Dritten durch den Einsatz diesen Gerätes, Missbrauch oder Störungen des Anschlusses, Störungen des Gerätes oder der Teilnehmergeräte entstehen.
- Eigenmächtige Veränderungen und Umbauten am Gerät führen zum Erlöschen der Gewährleistung!
- Für nicht bestimmungsgemäße Verwendung haftet der Hersteller ebenso nicht.

Support

Email

Der Standard-Support läuft über die E-Mail-Adresse: eibpc@enertex.de.
Unter http://knx-user-forum.de/eibpc ist ein eigener Bereich für den Support des Enertex® EibPC eingerichtet. Hier finden Sie auch direkten Rat von versierten Anwendern und Profis. Bitte senden Sie uns bei jeder Anfrage auch die folgenden Informationen:

- Anwenderprogramm (*.epc)
- Ggf. Importadressen (*.esf)
- Ggf. Log des Bustransfers des Enertex® EibStudio (*.log)

Durch die Option „Packe Informationen für Supportanfrage“ unter „?“ wird in der Menüleiste eine tar-Datei mit allen notwendigen Informationen erzeugt.

HOTLINE

Lehrvideos

Sie können zu den üblichen Öffnungszeiten auch den kostenlosen Telefonsupport der Hotline 09191/73395-0 nutzen.
Unter www.youtube.com finden Sie eine umfangreiche Lehrvideosammlung im Bereich „eibpc“

Updates

Auf der Seite www.eibpc.com finden Sie Updates zum Enertex® EibPC:

- Firmwareupdates (Eigentliches Programm mit Buszugriff)
- Patchupdates (Webserver, interne Dateistruktur)
- Enertex® EibStudio
- Auf S. 354 finden Sie die Neuerungen im Kurzüberblick.Enertex® EibPC - Übersicht

Abbildung 1: Der Enertex® EibPC

Zusammenfassung

Der Enertex® EibPC stellt eine Steuerung für die Hutschienenmontage (6 TE) für KNX™-Bussysteme dar. Er ermöglicht mit ca. 1,2 W Leistungsaufnahme stromsparend und umweltschonend vollständige Kontrolle über das KNX™-Bussystem.

Der Enertex® EibPC ist Szenenaktor, Berechnungscomputer, Logikzentrale, SPS, Schaltzeituhr, LAN-und Internet-Anbindung, Webserver und E-Mail-Client in einem Gerät.

Mit der Software Enertex® EibStudio ist eine Parametrierung des EibPCs und Zugriff auf Gruppenadressen ohne ETS möglich. Sie können diese Software unter www.enertex.de gratis downloaden.

Fertige Funktionsblöcke

Wenn Sie nur Grundfunktionen des Enertex® EibPC nutzen wollen, so reicht Ihnen die Verwendung von „Makros“ aus. Makros sind vordefinierte Funktionsblöcke, die den Anwendern in Form einer Bibliothek zur Verfügung stehen. Die frei erhältlichen Bibliotheken der Enertex® Bayern GmbH stellt Ihnen komplette Automatisierungslösungen für

- Lichtsteuerungen
- Rollo- und Jalousiesteuerungen (Beschattung)
- Schaltuhren
- Wintergartensteuerungen uvm.

zur Verfügung. Diese können Sie unter unter www.enertex.de gratis downloaden.

KNX-Funktionen

Alle Funktionen des KNX™-Bussystems sind ansprechbar

Dabei übernimmt der Enertex® EibPC im KNX™-Netz die Funktionalität von

- Szenenaktoren
- bedingten Anweisungen (wenn-dann)
- Schaltzeituhren
- Zeitgeber (Uhrzeit und Datum) (per LAN oder KNX™ oder Enertex® Eibstudio synchronisiert)
- genauesten Timern (im ms-Bereich)
- Reglern beliebiger Struktur
- Auswertungen mathematischer Ausdrücke
- Verzögerungsgliedern
- Verknüpfungen von KNX™-Objekten (Tor, Multiplexer, ...)
- Überwachung von Aktoren (z.B. zyklische Leseanforderungen).

Diese Funktionen können beliebig oft genutzt werden. Sie könnten also beispielsweise 65000 Szenenaktoren definieren. Der Enertex® EibPC verarbeitet die gesamte maximal mögliche Anzahl von Objekten einer KNX™-Vernetzung.

LAN-Funktionen

Der Enertex® EibPC besitzt eine LAN-Schnittstelle, so dass zusätzlich

- die Überwachung des Busverkehrs (ohne ETS [und PC])
- das Senden und Verarbeiten von beliebigen KNX™ -Telegrammen (ohne ETS)
- die Synchronisierung der Bus-Zeit via Internet (ohne ETS)
- das Senden, Empfangen und Verarbeiten von UDP-Telegrammen (Zusatz-Option NP) z.B. zur Steuerung von Multimediasystemen
- das Senden, Empfangen und Verarbeiten von TCP-Telegrammen als TCP Server und/oder Client (Zusatz-Option NP)
- das Senden von E-Mails (Zusatz-Option NP)
- ein integrierter Webserver (Zusatz-Option NP)
- eine VPN Verbindung über KNX Verbindung zu konfigurieren (Zusatz-Option NP)

möglich ist.

Datenlogger

Mitloggen von bis zu 500.000 Telegrammen möglich

Speicher - Der Enertex® EibPC speichert sämtliche Bustelegramme. Bis zu 500.000 Telegramme werden in einem Ringspeicher vorgehalten, wenn kein PC mit dem Enertex® EibPC verbunden ist. Bei einer durchschnittlichen Buslast von drei Telegrammen/Minute entspricht das allen Telegrammen der letzten 200 Tage.

Zeit - Mit Hilfe der Zeitstempel, die der Enertex® EibPC automatisch generiert, kann der Busverkehr jederzeit analysiert werden.

Online - Daneben besteht die Möglichkeit, die Daten online anzuzeigen und nach Absender und Gruppenadressen zu filtern.

Filter - Die Telegramme können bereits nach Geräteadressen und Gruppenadressen vorgefiltert werden.

Autolog - Das Enertex® EibStudio ermöglicht das zyklische Abspeichern von (ggf. gefilterten) Telegrammen in Dateien.

FTP - Der Enertex® EibPC kann Telegrammdaten an einen beliebigen FTP Server schicken. Das Enertex® EibStudio ermöglicht das Auswerten dieser Binärdaten und exportiert diese in lesbare CSV- Textdateien.

Software

Mit dem Enertex® EibStudio als Konfigurationsprogramm wird über die LAN-Schnittstelle des Enertex® EibPC zu einem Windows®- oder Linux®-PC eine Hausautomatisierung erstellt. Damit lässt sich der Enertex® EibPC ohne ETS auf einfache Weise programmieren.

Basic - Die Programmierung erfolgt in einer einfachen Basic-Syntax, für die keine aufwändigen Lehrgänge notwendig sind. Für die Grundfunktionalität ist noch nicht einmal dieses Basic zu erlernen: Der Anwender hat eine Auswahl an vorgefertigten Funktionsblöcken zur Verfügung, in die er lediglich Gruppenadressen o.ä. eintragen muss.

ETS - Das Enertex® EibStudio importiert die Adressen und Einstellungen der ETS. Es kann aber auch gänzlich ohne ETS-Import gearbeitet werden.

Inbetriebnahme

Programmierung

Sie benötigen für die Programmierung des Enertex® EibPC einen Windows®- oder Linux®-PC und eine LAN-Verbindung zum Enertex® EibPC. Der Enertex® EibPC selbst benötigt für seinen Betrieb keine LAN-Verbindung, wobei in diesem Fall einige Funktionen wie Synchronisierung der Systemzeit mit der Internet-Zeit etc. nicht genutzt werden können.

Wir empfehlen für Windows® Umgebungen Windows® 7 zu verwenden.

Stromversorgung

Die Leistungsaufnahme beträgt ca. 1,2 Watt bei 24 VDC

Der Enertex® EibPC benötigt eine externe Gleichstromversorgung im Bereich von 20V bis 30 V. Die Leistungsaufnahme beträgt im Normalfall etwa 1,2 W, bei angeschlossenem LAN ca. 1,7W.

Buszugriff

Um auf den KNX™-Bus zugreifen zu können, benötigt der Enertex® EibPC eine KNX™ Schnittstelle:

Hierzu ist entweder externe EIB-RS232 Schnittstelle für das FT 1.2 Protokoll oder eine IP Schnittstelle geeignet.

FT1.2 Schnittstelle

Bitte eine passende Schnittstelle aussuchen

Folgende Schnittstellen sind getestet:

- EIBMarkt IF-RS232 (Sie müssen umschalten auf FT 1.2)
- Siemens N148/04 (5WG1 148-1AB04) (Sie müssen umschalten auf FT 1.2)
- Gira RS232 UP 0504xx + BA 064500

Grundsätzlich FT1.2 fähig sind:

- Siemens UP 146 Z1 für FT 1.2 (5WG1 146-2AB11-Z1) + BA UP 114 (5WG1 114-2AB02)
- Berker RS-232 Datenschnittstelle FT 1.2 (750601xx) + BA Up 2.0 (75040002)
- Merten

Folgende Schnittstellen sind sicher ungeeignet:

- ABB EA/S 232.5
- Siemens N148/02 (5WG1 148-1AB02)
- Berker RS-232 Datenschnittstelle REG (7501 00 13)
- Berker RS-232 Datenschnittstelle UP (750600 xx)

IP Schnittstelle/Router

Grundsätzlich sind alle KNX™ zertifizierte IP Schnittstellen mit dem Enertex® EibPC kompatibel.

Besonders zu empfehlen ist der:

- Enertex-KNXnet/IP Router

Die IP Schnittstelle wird erst nach Überspielen und anschließendem Starten eines Anwendungsprogramm angesprochen und aktiviert. Der Router bietet viele Zusatz- und Diagnosefunktionen und kann den Enertex® EibPC nach einem Reset ohne Internetverbindung direkt mit der Uhrzeit synchronisieren. Das

- Enertex-KNXnet/IP Interface

ist die kostengünstige Variante für den Zugriff auf den Bus.

Enertex® KNXNet/IP Router

Abbildung 2: Der Enertex® KNXNet/IP Router

Mehrere Verbindungen

Der KNXnet/IP Router (3TE) unterstützt bis zu fünf KNXnet/IP-Tunnelverbindungen und kann als Linien- oder Bereichskoppler eingesetzt werden. Durch sein integriertes Display hat man jederzeit alle wichtigen Konfigurationsparameter im Blick: IP-Adresse, KNX-Geräteadresse und Anzahl der geöffneten Tunnelverbindungen. Das ermöglicht eine zügige und problemlose Inbetriebnahme.

Die IP-Adresse der Fast Ethernet Verbindung kann manuell über die ETS, oder automatisch über DHCP oder Zeroconf. konfiguriert werden. Über einen Telnet Zugang werden wichtige Betriebsparameter wie z.B. die IP Adressen des Tunellings zugänglich gemacht. Außerdem verfügt der KNXnet/IP Router über eine batteriegepufferte Echtzeituhr und stellt einen SNTP-Server im LAN zur Verfügung. Auch in dieser Hinsicht ist der Enertex® KNXNet/IP Router die ideale Ergänzung zum Enertex® EibPC, da er damit nach einem Stromausfall ohne Internetverbindung zeitsychronisiert werden kann.

Der Enertex® KNXNet/IP Router erlaubt zudem die Verbindung mit einem Telnet-Server, der weitere Informationen zum Busverkehr bereithält.

Der KNXnet/IP Router wird über Power over Ethernet oder durch eine externe 20-30V AC/DC Spannungsversorgung gespeist.

Der Enertex® EibPC kann sich die Stromversorgung mit dem Enertex® KNXNet/IP Router teilen, wenn sichergestellt ist, dass diese ca. 3 Watt zu Verfügung stellt.

Simulation

„Trockenübungen“

Da die RS232-Schnittstelle keine Rückmeldung über den Verbindungsstatus liefert, schickt der EibPC alle Telegramme raus, auch wenn keine Busverbindung aufgebaut ist. Dadurch lässt sich auch ohne Busverbindung jeder Code austesten.

Die IP Schnittstelle hat dagegen ein definiertes Handshake. Der EibPC wartet dann auf die Rückmeldung, bevor er Telegramme verschickt.

Installation

Hardware

Abbildung 3 zeigt den prinzipiellen Aufbau eines KNX™-Netzes, an welches der Enertex® EibPC angeschlossen wird. Um den Enertex® EibPC in das KNX™-System einzubinden, ist wie folgt vorzugehen:

Einfache Hutschienenmontage

1. Den Enertex® EibPC an eine 20V - 30V DC Versorgung anschließen. Dies kann entweder über ein separates Netzteil oder im KNX™-Bus geschehen. Will man den Enertex® EibPC an den KNX™-Bus anschließen, benötigt man hierzu eine Bus-Drossel zwischen Busleitung und Versorgung des Enertex® EibPCs.
2. Über den LAN-Anschluss (2) muss der Enertex® EibPC mit dem LAN verbunden werden.
3. Die RS232 Schnittstelle (3) wird mit einer RS232-KNX™-Schnittstelle (für FT 1.2 Protokoll) verbunden. Als Kabel nimmt man dafür ein Verlängerungskabel ("Male“ auf “Female").
4. EibPC neu starten (über EibStudio oder einfach Stromzufuhr unterbrechen).

Wichtig

Bitte beachten:

Die externe Sicherheitskleinspannung wird durch das Gerät mit dem Erdpotential des LAN verbunden. Damit besteht keine Isolation mehr zur Erde, wenn der LAN-Schirm geerdet wird. Um eine Trennung herzustellen, wird empfohlen eine externe Kleinspannungsversorgung nur für den Enertex® EibPC zu verwenden.

Die Info-LED erleichtert die Inbetriebnahme:

Wenn der EibPC startet, muss die RS232-Schnittstelle bereits mit dem betriebsbereiten KNX™ Bus verbunden sein. Um dies zu gewährleisten, gehen Sie wie folgt vor:

Einschalten:
grüne LED leuchtet

1. Verdrahtung wie oben beschrieben vornehmen.
2. Schnittstelle an den Bus anklemmen (Bus bereits in Betrieb).
3. Enertex® EibPC neustarten (über EibStudio oder einfach Stromzufuhr unterbrechen).

Bootvorgang beendet
grüne LED blinkt

4. **Wenn Sie mit FT1.2 RS232 Schnittstelle arbeiten:**
   Wenn der EibPC startet muss die FT1.2 RS232 in jedem Fall am eingeschalteten Bus hängen. **Also nicht:** Enertex® EibPC Einschalten und dann RS232 Verbindung herstellen.
   Die RS232 Schnittstelle kann mit dem Enertex® EibPC gleichzeitig eingeschalteten werden, nicht aber nach dem Bootvorgang des Enertex® EibPC.

Kurz danach:
Die RS232 wird initialisiert:
Ein paar kurze Impuls können an der LED beobachtet werden

5. **Wenn Sie mit IP Schnittstelle arbeiten:** Sie sollten auf diese Schnittstelle einmal einen Reset ausführen, da diese mitunter auch hängen können. Die Schnittstelle wird erst nach Überspielen und anschließendem Starten eines Anwendungsprogramm angesprochen und aktiviert. Der Enertex® EibPC nutzt das sog. Tunneling der IP Schnittstelle. Diese wird dadurch exklusiv belegt, d.h. Sie können die Schnittstelle in diesem Modus dann nicht von der ETS aus ansprechen. Um dennoch einen kurzfristigen Zugriff auf die Schnittstelle zu ermöglichen können Sie die Schnittstelle über den Menüpunkt VERBINDUNG TRENNEN und VERBINDUNG AUFBAUEN vom Enertex® EibPC trennen bzw. mit diesem erneut verbinden.

Enertex® EibPC erkennt Telegramm:
Schnelles, kurzes Blinken:

Jetzt haben Sie die Installation des Enertex® EibPC an Ihr KNX™ -System abgeschlossen.

Inbetriebnahme der RS232

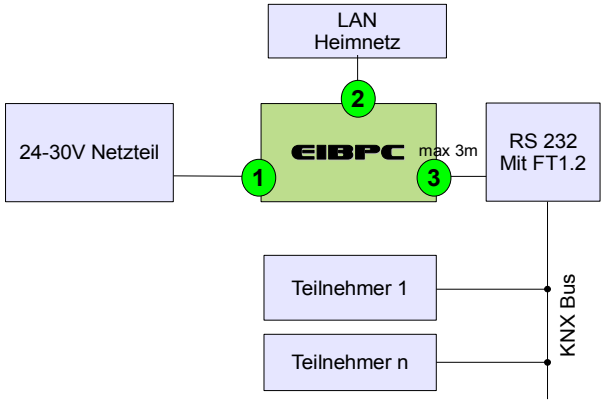


Abbildung 3: Anschluss des Enertex® EibPC an den KNX™-Bus über eine RS232 Schnittstelle.

Inbetriebnahme der IP Schnittstelle

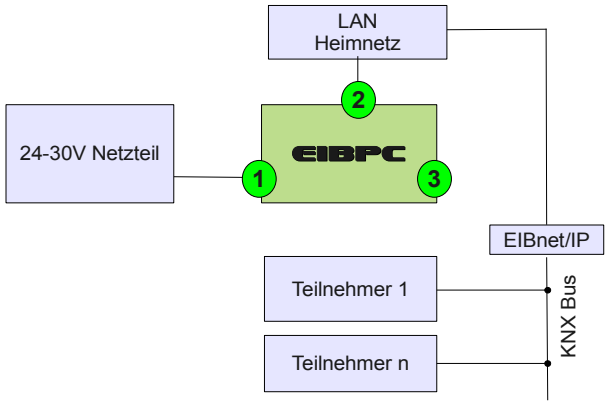


Abbildung 4: Anschluss des Enertex® EibPC an den KNX™-Bus über eine EIBnet/IP- Schnittstelle

In Abbildung 5 sind die Anschlüsse des Enertex® EibPC dargestellt.
Es sind folgende:

1. Spannungsversorgung 20-30V DC
2. LAN-Anschluss
3. RS232 Anschluss
4. Info-LED
5. Reset-Knopf

Abbildung 5: Enertex® EibPC Anschlussbelegung

Nach dem Einschalten oder Softwarereset (via EibStudio) des Enertex® EibPC können Sie Folgendes beobachten:

1. Zuerst ist die LED dauerleuchtend (Kontrolle der Betriebsspannung) und der Bootvorgang wird gestartet.
2. Ca. 4 Minuten nach dem Start blinkt die grüne LED im Sekundentakt mit gleichlanger Ein- und Ausschaltzeit. Nur während dieser Phase ist ein Reset möglich (s.u.).
3. Kurz danach wird die RS232 Schnittstelle initialisiert - Ein paar kurze Impulse können dabei an der LED beobachtet werden.
4. Wenn ein Telegramm am Bus ausgelöst wird (KNX-Schalter etc.), wird auch eine kurze Impulsfolge an der LED sichtbar (dazu muss kein Anwenderprogramm geladen sein oder ähnliches).

Die RS232 Schnittstelle kann mit dem Enertex® EibPC gleichzeitig eingeschalteten werden, wenn die Verbindung zum Enertex® EibPC bereits besteht. Sie darf nicht nach Bootvorgang des Enertex® EibPC angeschlossen werden, da Sie sonst vom Enertex® EibPC nicht ordnungsgemäß initialisiert werden kann.

Sie sollten auf diese Schnittstelle einmal einen Reset ausführen, da diese mitunter auch hängen können (bei einer nicht ordungsgemäß ausgeführten Trennung einer Verbindung z.B. mit der ETS oder einem Zugriff über einen anderen LAN Teilnehmer). Die Schnittstelle wird erst nach Überspielen und anschließendem Starten eines Anwendungsprogramm angesprochen und aktiviert. Der Enertex® EibPC nutzt das sog. Tunneling der IP Schnittstelle.

Reset - Werkseinstellungen

Während des Bootvorgangs leuchtet die grüne Info-LED durchgehend. Danach blinkt die Info-LED im Sekundentakt für 3 Sekunden. Nur während dieser Phase ist der Resetknopf aktiv.

Durch Betätigen des Resets wird Folgendes ausgelöst

1. Netzwerkeinstellungen auf Auslieferzustand zurücksetzen
2. Anwenderprogramm löschen
3. Daten zum Sonnenstand löschen.

Die Pinbelegung der RS232-Schnittstelle des Enertex® EibPC entspricht dem Standard EIA-232.

Verwenden Sie zur Kommunikation von Enertex® EibPC mit der KNX™ RS232-Schnittstelle das mitgelieferte Kabel. Wenn Sie Patch 1.101 installiert haben, benötigt der Bootvorgang ca. 40 Sekunden. Im anderen Fall beträgt die Bootzeit etwa 4 Minuten.

Ein Patch-Update benötigt auch ca. 40 Sekunden.

Kurzanleitung

Schnellstart - Enertex® EibStudio

Für die Inbetriebnahme des Enertex® EibPC, insbesondere dem Netzwerksetup, müssen Sie zuerst die Firewall des Routers und des PCs, von dem Sie auf den Enertex® EibPC zugreifen, so konfigurieren, dass Datenpakete des Enertex® EibPC nicht blockiert werden.

Der Enertex® EibPC muss über eine LAN-Verbindung über eine IP-Adresse erreichbar sein. Die Voreinstellung des Enertex® EibPC ist die Übertragung per DHCP Server, d.h. wenn ein DHCP fähiger Router aktiviert ist, so weist der DHCP Server dem Enertex® EibPC eine IP-Adresse zu. Diese Adresse muss nun auch dem Enertex® EibStudio bekannt gemacht werden, was in diesem Fall automatisch erfolgen kann, d.h. Enertex® EibStudio findet selbstständig den Enertex® EibPC im Netz. Wenn Sie mehr als einen Enertex® EibPC im Heimnetz nutzen wollen, sollten Sie mit festen IP-Adressen arbeiten, wie ab Seite 143 beschrieben.

Der Enertex® EibPC findet auch ohne DHCP Server eine eindeutige IP Adresse. Hier ist bei Peer-to-Peer ein Crossoverkabel notwendig!

Wenn Ihr Router keine DHCP Funktion aufweist, übernimmt der Enertex® EibPC selbst diese Funktion und gibt sich eigenständig eine IP-Adresse, die Sie dann vom Enertex® EibStudio aus abfragen können (dazu weiter unten mehr). Voraussetzung hierfür ist, dass eine eventuell eingestellte Firewall passend konfiguriert wird. Wenn Sie mit einer festen IP-Adresse arbeiten wollen, lesen Sie bitte ab Seite 143 weiter. Bei direkter Verbindung zu Ihrem PC (Peer-to-Peer) benötigen Sie in diesem Fall ein Crossoverkabel, das beim Systemstart schon angeschlossen sein muss.

1. Enertex® EibStudio starten

Die mitgelieferte Enertex® EibStudio Software wird einfach per Doppelklick gestartet. Es erfolgt keine Installation im herkömmlichen Sinn. Enertex® EibStudio nimmt z.B. keinerlei Einträge in der Windows®-Registry vor.

2. Firewall konfigurieren

Wichtig: Eine aktive Firewall verhindert u.U. die Kommunikation zwischen Enertex® EibStudio und Enertex® EibPC und muss entsprechend konfiguriert werden.

Wenn Sie das erste mal die IP-Adresse ihres EibPCs mit dem Button *Automatisch* aus dem Dialog nach Abbildung 8 erfragen, dann meldet sich die Windows-Firewall. Gestatten Sie in dieser Nachfrage dem Programm nconf, das für die LAN-Kommunikation von Enertex® EibStudio zuständig ist, den Zugriff auf das LAN. Damit richtet Windows® automatisch eine Firewall-Ausnahme für Eibstudio ein. Sie brauchen in diesem Fall keine weiteren Einstellungen an ihrer Windows® -Firewall vorzunehmen. Die in Abbildung 6 für Windows XP gezeigte Prozedur können Sie dann überspringen.

Windows 7 und Windows 8.1

Verschiedene Sicherheitssoftwarepakete verhindern u.U. das Laden eines Programms in den Enertex® EibPC. Besonders betroffen sind hier Windows 8.1 Nutzer. Meist sind neben den Firewall-Problemen auch Dateiberechtigungen. Die Sicherheitssoftware erlaubt mitunter nicht EibStudio nicht, dass im „Programme“-Ordner Unterverzeichnisse erstellt werden. Das geht auch dann nicht, wenn man entsprechende Einstellungen ändert. Es gibt zwei Möglichkeiten, dies zu umgehen:

1. Kopieren sie Eibstudio auf Ihren Desktop bzw. User-Verzeichnis und führen sie es dort aus.
2. Führen Sie EibStudio vom Programme-Verzeichnis mit dem Parameter "-D Verzeichnis" aus, wobei Verzeichnis hier ein Pfad in Ihrem Userverzeichnis sein muss, z.B. Download. Um ein Programm mit einem Parameter starten zu können, muss zunächst eine Verknüpfung auf diese Datei (hier also EibStudioVx_xxx.exe) erstellt werden und anschließend die Parameter in der Verknüpfung eingetragen werden.

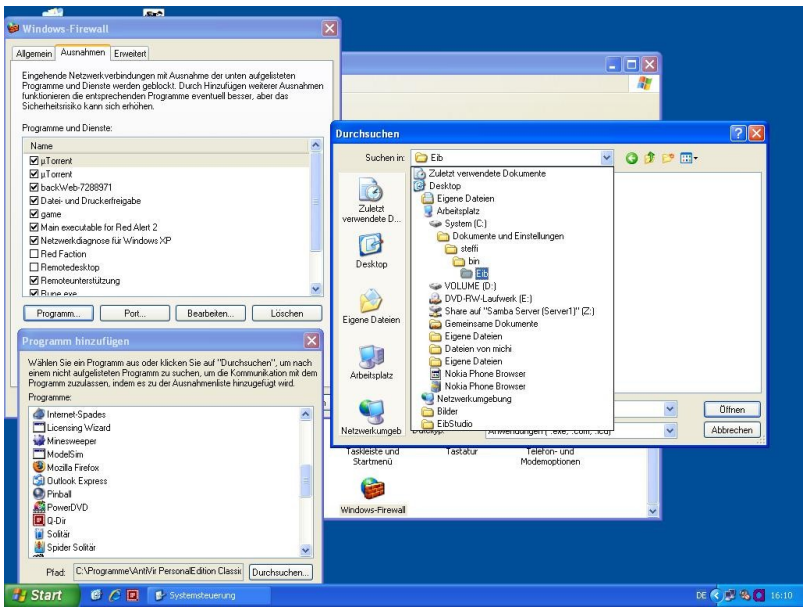


Abbildung 6: Konfiguration Windows®-Firewall

Bei einer Windows® -Firewall müssen Sie dem Programm nconf.exe (Teilprogramm von Enertex® EibStudio) eine Ausnahme einrichten. Enertex® EibStudio müssen Sie dazu mindestens einmal gestartet haben. Sie finden nconf.exe nach dem Aufruf von Enertex® EibStudio unter Ihrem Windows®-Nutzerverzeichnis wie in Abbildung 6 angegeben. Um dies einzurichten wählen Sie im Dialog der Firewall *Ausnahmen-Programm*, im Unterdialog *Programm hinzufügen,*dort *Durchsuchen* und geben Ihren Pfad entsprechend „C:\Dokumente und Einstellungen\IHR NUTZERNAME\bin\Eib\nconf.exe“ an. Alternativ schalten Sie den Kommunikationsport 4805 und 4806 für UDP-Telegramme in Ihrer Firewall frei. Wenn Sie UDP-Telegramme versenden wollen, müssen Sie zusätzlich, 4807 öffnen.

Beim erstmaligen Start werden Sie durch eine Setup-Routine geführt um die nötigen Einstellungen für die LAN-Verbindung zwischen Enertex® EibPC und Enertex® EibStudio vorzunehmen:

Im automatisch startenden Setup-Assistent (Abbildung 8) wählen Sie die Schaltfläche „Automatisch“. Falls Sie einen DHCP Server im Netzwerk haben, vergibt dieser dem Enertex® EibPC die IP Adresse. Findet ihr Enertex® EibPC keinen DHCP Server, so vergibt er sich nach dem Bootvorgang selbst eine IP Adresse. Dies kann nützlich sein, wenn der Enertex® EibPC direkt an den PC angeschlossen wird. In diesem Fall ist aber mindestens ein LAN Crossoverkabel zwischen EibPC und PC notwendig.

Wenn Sie nun die Schaltfläche „Automatisch“ klicken, erscheint die Adresse des Enertex® EibPC im Fensterabschnitt „Meldungen“ und wird in den Dialog von Abbildung 8 automatisch eingetragen. Sie können diesen Dialog auch über Optionen – Netzwerkeinstellungen aufrufen.

Anschließend speichern Sie über Optionen – Netzwerkeinstellungen Speichern ihre Netzwerkeinstellungen.

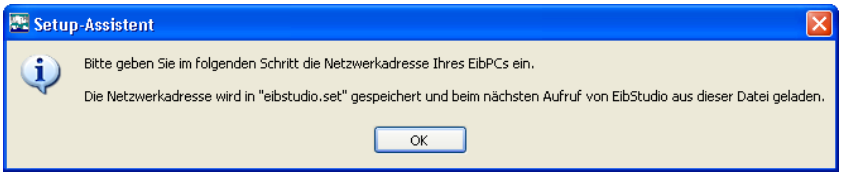


Abbildung 7: Setup-Assistent 1/3

Der Enertex® EibPC kann über die Schaltfläche Automatisch gesucht werden

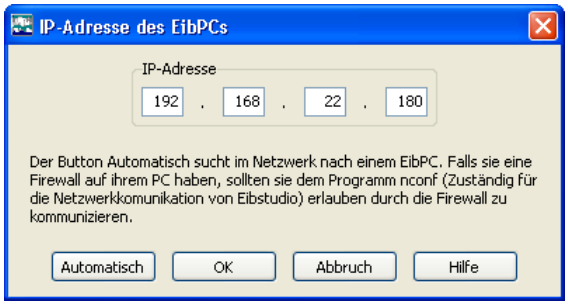


Abbildung 8: Setup-Assistent 2/3

Wenn Sie mit fest eingestellten IP Adressen arbeiten wollen, finden Sie ab Seite 143 weitere Informationen.

Sie werden nun aufgefordert ein neues Anwenderprogramm zu erstellen. Klicken Sie auf die entsprechende Schaltfläche (Abbildung 9), so können Sie im nächsten Schritt die ESF-Daten Ihres ets-Projekts in Enertex® EibStudio importieren. Wie Sie diese Daten aus der ETS zuvor exportieren (also die benötigte ESF-Datei erzeugen) ist auf Seite 158 angegeben.

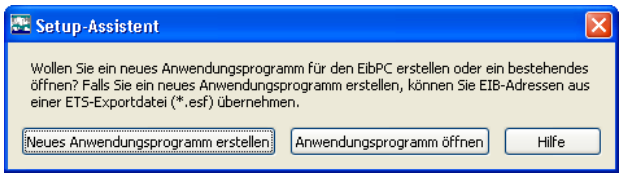


Abbildung 9: Setup-Assistent 3/3

Der Enertex® EibPC wurde gefunden.

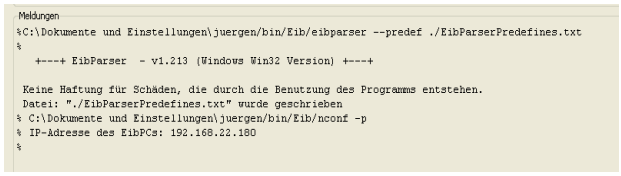


Abbildung 10: Meldungsfenster zeigt an, dass ein Enertex® EibPC gefunden wurde.

Schnittstelle konfigurieren

Über das Menü EIB-SCHNITTSTELLE kann die KNX™ Schnittstelle gewählt werden, über die der Enertex® EibPC auf den KNX™ Bus zugreift.

Abbildung 11: Schnittstelle einrichten

- Bei einer FT 1.2 müssen Sie die Inbetriebnahmehinweise auf 16ff. beachten.
  Sie können die FT1.2 Schnittstelle im Busmonitormodus oder im Gruppenmonitormodus betreiben. Der Gruppenmonitormodus ist grundsätzlich fehlertoleranter gegen Busfehler. Beachten Sie hierzu auch die Hinweise auf S.147. Falls die Schnittstelle vor dem Betrieb im Gruppenmonitormodus im Busmonitormodus eingesetzt wurde, muss sie vor dem Übertragen des Anwendungsprogramms durch ein kurzes Unterbrechen ihrer Stromversorgung zurückgesetzt werden.

- Bei einer IP - Schnittstelle müssen Sie die Inbetriebnahmehinweise auf 16ff. beachten und zudem im Dialog die entsprechenden Daten Ihrer Schnittstelle eingeben. Die Schnittstelle wird erst nach Überspielen und anschließendem Starten eines Anwendungsprogramm angesprochen und aktiviert.
  Der Enertex® EibPC nutzt das sog. Tunneling der IP Schnittstelle. Diese wird dadurch exklusiv belegt, d.h. Sie können die Schnittstelle in diesem Modus dann nicht von der ETS aus ansprechen. Um dennoch einen kurzfristigen Zugriff auf die Schnittstelle zu ermöglichen (z.B. Programmierung kleiner Änderungen der Gruppenadressen) können Sie die Schnittstelle über den Menüpunkt VERBINDUNG TRENNEN und VERBINDUNG AUFBAUEN vom Enertex® EibPC trennen bzw. mit diesem verbinden.

- Mit der Telegrammratenbegrenzung, die standardmäßig auf 7 Telegramme pro Sekunde steht, verhindern Sie ein „Überfahren“ des KNX™ Busses und gewährleisten stabilen Betieb Ihrer Installation.

FT1.2 Schnittstellenmodus

Hintergrund

Im Busmonitormodus berechnet der Enertex® EibPC bzw. die ETS Checksummen und greift über die Schnittstellen direkt auf den Bus ein. Jedes Telegramm muss sofort quittiert werden. Durch diese unmittelbare und zeitkritische Kommunikation zwischen Enertex® EibPC (bzw. ETS) und FT1.2 entsteht ein zusätzlicher Performancebedarf auf Seiten der FT1.2, weshalb hier bei hohem Busaufkommen Telegramme verloren gehen können. Im Gruppenmonitormodus quittiert die FT1.2 eintreffende Telegramme direkt. Eine Kommunikation mit dem Enertex® EibPC (bzw. der ETS) wird nicht notwendig. Daher ist diese Kommunikation etwas leistungsfähiger.

Der weitere Aufbau des Handbuchs

Wenn Sie den Netzwerksetup funktionsfähig eingerichtet haben, können Sie nun den Enertex® EibPC im KNX™ Netzwerk verwenden.

Sie können dieses Handbuch unterschiedlich nutzen: Die verschiedenen Abschnitte sind so aufgebaut, dass sie weitgehend in sich abgeschlossen sind. Wichtige Details können daher auch doppelt in den verschiedenen Abschnitten erklärt werden.

Funktionsblöcke

Ab Seite 26 zeigen wir die Anwendung von fertigen (mitgelieferten) Funktionsblöcken. Hierzu benötigen Sie keinerlei Programmierkenntnisse. Anhand eines Beispiels zeigen wir die einfache Verwendung dieser Funktionsblöcke. Damit können Sie bereits nach wenigen Minuten eine Automatisierung erstellen. Für die verfügbaren Funktionsblöcke existiert auch ein Funktionsblöcke-Handbuch, wo die verschiedenen Anwendungen erläutert werden.

Wenn Sie alle Möglichkeiten des Enertex® EibPC ausschöpfen wollen, empfehlen wir Ihnen den Abschnitt „Schritt für Schritt“ zu lesen.

Einführung Programmierung

Ab Seite 35 wird in Form einer „Schritt-für-Schritt“ Einführung die Programmierung mit dem Enertex® EibPC erläutert. Damit sind Sie in der Lage, die Möglichkeiten des Enertex® EibPC voll auszuschöpfen. Die Programmierung ist dabei denkbar einfach und konsequent. Das Handbuch setzt keinerlei Programmierkenntnisse voraus.

Vollständige Beschreibung des Enertex® EibPC

Ab Seite 133 wird der gesamte Umfang der Programmierumgebung systematisch erklärt.

Selbst definierte Funktionsblöcke

Auf Seite 323 wird gezeigt, wie Sie selbst Funktionsblöcke definieren können. Sie können so eigene Bibliotheken mit Funktionsblöcken generieren, die Sie dann mehrfach in verschiedenen Projekten nutzen können. Auf diese Weise können Sie wiederkehrende Aufgaben nahezu ohne Aufwand und in kurzer Zeit erledigen.

Mit dem Enertex® EibPC erwerben Sie sich eine Lizenz zur Nutzung des Enertex® EibStudio. Enertex® EibStudio und sämtliche Komponenten dürfen nur in Verbindung mit dem Enertex® EibPC genutzt werden.

Verwenden von vorgefertigten Funktionsblöcken

Makros/Funktionsblöcke

Wenn Sie nur Grundfunktionen des Enertex® EibPC nutzen wollen, so reicht Ihnen die Verwendung von „Makros“ aus. Makros sind vordefinierte Funktionsblöcke, die den Anwendern in Form einer sog. Bibliothek zur Verfügung stehen.

Standards

Die mitgelieferte Bibliothek der Enertex® Bayern GmbH stellt Ihnen komplette Automatisierungslösungen für die folgenden Aufgaben bereit:

Vordefinierte Bibliotheken

Bibliothek	***Anwendungen***
EnertexLicht.lib	Treppenhauslichter, Flurlichter mit Zeitautomatik
EnertexBeschattung.lib	Beschatten von Häusern
EnertexSchaltuhren.lib	Schaltuhren, Zeitsteuerung (veraltet, siehe EnertexSchaltuhrenV2.lib)
EnertexWiga.lib	Wintergartensteuerungen
Enertex.lib	Nützliches
EnertexWeb.lib	Makros zur einfacheren Anwendung des eingebauten Webservers (veraltet, siehe EnertexWebV3.lib).
EnertexWebV2.lib	Makros zur einfacheren Anwendung des eingebauten Webservers V2 (veraltet, siehe EnertexWebV3.lib).
EnertexWebV3.lib	Makros zur einfacheren Anwendung des eingebauten Webservers V2 (veraltet, siehe EnertexWebV3.lib).
EnertexLogik.lib	Einfache Logikverknüpfungen
EnertexOneWire.lib	Makros zum Einbinden von 1-Wire Sensors mit dem Zusatzadapter HA7E (see www.shop.enertex.de)
EnertexCommandFusion.lib	Anbindung iPhone/iPad/iPod
EnertexSchaltuhrenV2.lib	Schaltuhren, Zeitsteuerung
EnertexFlash.lib	Langzeitaufzeichnung im internen Flash samt Visualisierung
EnertexSzenen.lib	Erweiterte Szenensteuerung
EnertexPhyMonitor.lib	Überwachung von Aktoren
EnertexAnwesentheit.lib	Anwesenheitssimulation mit dem Enertex® EibPC
EnertexSqueezebox.lib	Ansteuerung einer Logitech-Sequeezebox mit dem Enertex® EibPC
EnertexRussound.lib	Ansteuerung einer Multiroom Audio Anlage „Russound“ mit dem Enertex® EibPC
EnertexCodingWettbewerb.lib	Verschiedene und nützliche User Makros
EnertexVPN.lib	VPN Funktionalität des EibPC

Zudem sind folgende User-Bibliotheken in unserer Makro-Bibliothek:

CHGLibrary_FritzBox.lib	Fitzbox Anbindung (getestete User-Library)
Sonos.lib	Sonos Musikplayer Anbindung (getestete User-Library)

Im Downloadbereich des Enertex® EibPC (www.eibpc.com) stehen Ihnen diese und weitere Bibliotheken zur Verfügung.

Für jedes Makro wird außerdem ein Anwendungsbeispiel angegeben.

Bevor Sie den Enertex® EibPC verwenden können, müssen Sie die Basisinstallation des Enertex® EibPC durchführen:

1. Enertex® EibPC installieren (s. Seite 18).
2. Enertex® EibPC und Enertex® EibStudio konfigurieren (Schnellstart auf Seite 21 bzw. erweiterter LAN-Setup, s. Seite 143).
3. Optional: Gruppenadressen Ihres ETS Projekts importieren (s.u.).

Anschließend können Sie nun:

4. Funktionsblöcke aufrufen und
5. das Anwenderprogramm in den Enertex® EibPC übertragen und starten.

Treppenhauslicht

An einem Beispiel wollen wir Ihnen den Enertex® EibPC erklären: Sie sollen Ihrem Kunden ein Treppenhauslicht parametrieren. Die Leuchtdauer betrage 300 Sekunden, die Installation ist wie in Abbildung 12 gegeben.

Beispielhafte Installation

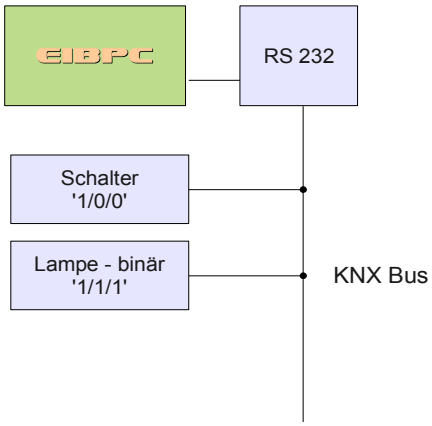


Abbildung 12: Blockschaltbild Treppenhauslicht

Zunächst starten Sie das Enertex® EibStudio. Wenn Sie Enertex® EibStudio zum ersten Mal aufrufen, werden Sie durch die Setup-Prozedur geleitet.

Anschließend sehen Sie dann ein Fenster, wie in Abbildung 13 dargestellt.

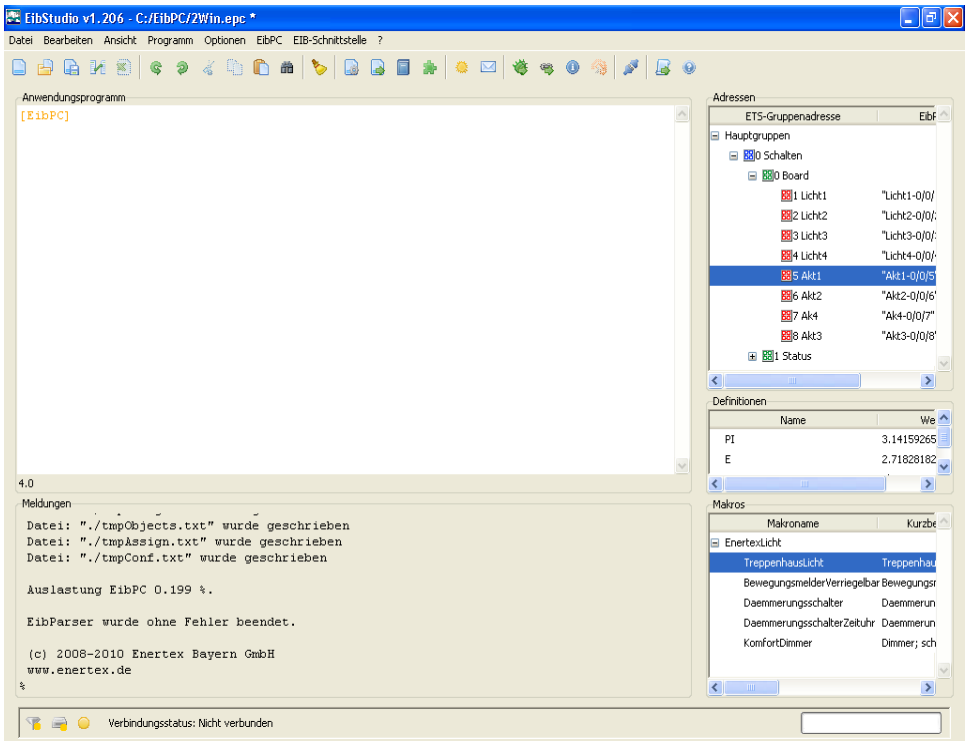


Abbildung 13: Enertex® EibStudio

Achten Sie auf den linken weißen und noch leeren Bereich „Anwenderprogramm“. Dieser Bereich des Enertex® EibStudio stellt einen integrierten Texteditor dar. Hier werden die notwendigen Informationen Ihres Programms gespeichert.

Wir importieren durch klicken auf die markierte Schaltfläche von Abbildung 13 die esf-Datei des Projekts. Sie erkennen den erfolgreichen Import daran, dass die Adressen im linken Fenster „Adressen“ von Abbildung 14 erscheinen. (Weitere Details, z.B. wie Sie in der ETS eine ESF-Datei erzeugen können, finden Sie auf Seite 152).

ESF-Import: Daten aus der ETS importieren

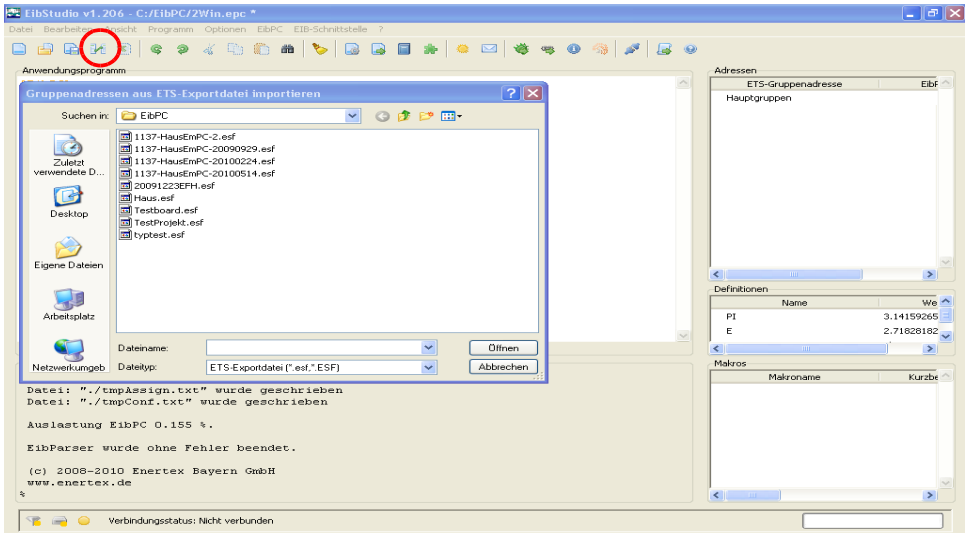


Abbildung 14: ESF-Import

Anschließend klicken Sie nun auf PROGRAMM / MAKRO-BIBLIOTHEKEN und dort auf den Menüpunkt HINZUFÜGEN. Wählen Sie die Bibliothek „EnertexLicht.lib“ aus und klicken Sie auf Hinzufügen.

Ihr Arbeitsbereich sollte nun ähnlich zu Abbildung 15 sein. Achten Sie auf den rechten Teilbereich „Adressen“. Hier sollten Ihre in der ETS parametrierten Gruppenadressen stehen, in unserem Beispiel die beiden „Schalter“ und „Lampe“, dargestellt mit der physikalischen Adresse als Zusatz im Namen.

Makrobibliotheken und ETS Daten sind eingebunden.

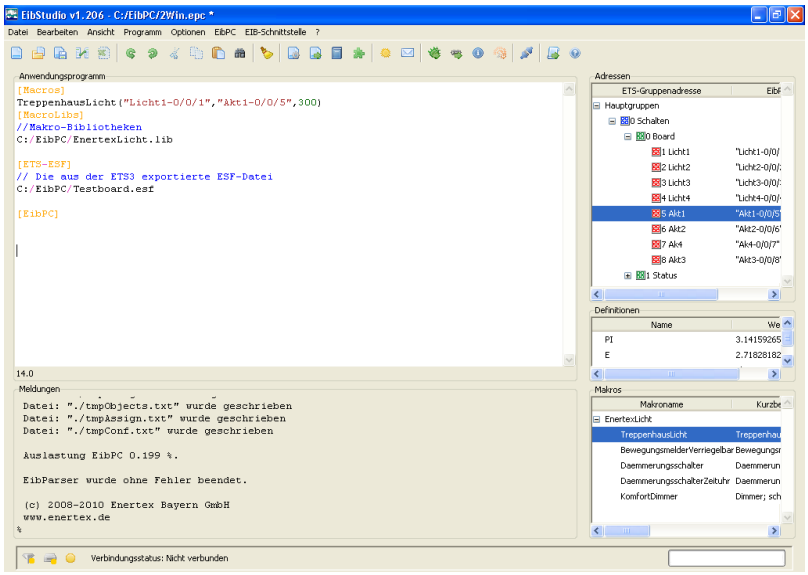


Abbildung 15: Arbeitsbereich Enertex® EibStudio

Nun wählen Sie PROGRAMM / MAKRO das Treppenhauslicht. Es erscheint ein Parameterdialog. Diesen können Sie nun entsprechend Ihren Vorgaben füllen.

Ihr Projekt sollte nun wie in Abbildung 16 dargestellt fertig sein. Klicken Sie auf die markierte Schaltfläche, um Ihr Programm an den Enertex® EibPC zu übertragen. Ihre Automatisierung ist damit bereits fertig.

Die Automatisierung ist fertig

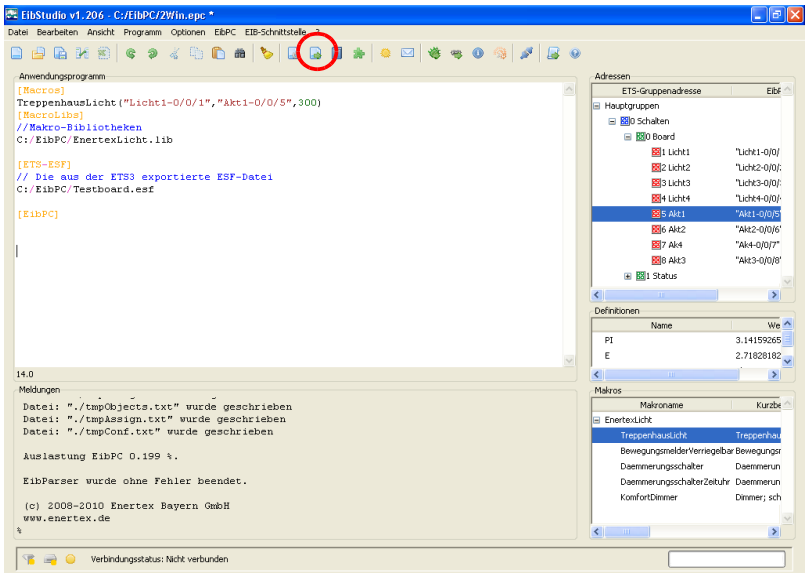


Abbildung 16: Ihre Automatisierung ist fertig.

Beschattung eines Hauses

Sie wollen eine Hausbeschattung vornehmen. Hierzu können Sie Makros aus der Bibliothek „EnertexBeschattung.lib“ verwenden.

Die Ausrichtung des Hauses ist wie in etwa in der Hauptrichtung der Himmelsrichtungen. Die Fenster sind mit einfachen Rollos ausgestattet. Der Bauherr möchte nun über ein Schaltelement auf der Gruppenadresse "FreigabeBeschattung-2/5/0" die Fenster beschatten. Damit es nicht völlig dunkel wird, soll der Rollo nicht ganz runter fahren, sondern leicht geöffnet bleiben.

Plan und ETS

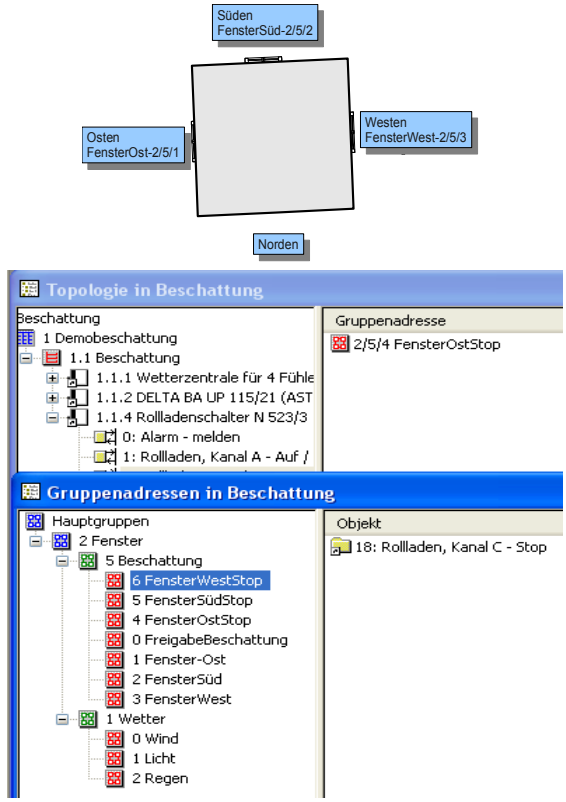


Abbildung 17: Ein Haus mit drei Fenstern – rechts die Vergabe der Gruppenadressen in der ets

Makroassistent aufrufen

Wenn Sie das Projekt wie in obiger Abbildung mit der ETS eingerichtet haben, exportieren Sie mit der ETS die Gruppenadressen mit Hilfe des OPC-Server (Datei-Datenaustausch). Dann starten Sie nun das Enertex® EibStudio. Wenn Sie Enertex® EibStudio zum ersten Mal aufrufen, werden Sie wie oben bereits gezeigt durch die Setup-Prozedur geleitet.

Der Assistent des Enertex® EibStudio importiert nun die Gruppenadresse, so dass nun - wie in Abbildung 18 gezeigt - im rechten Fenster die Gruppenadressen angezeigt werden.

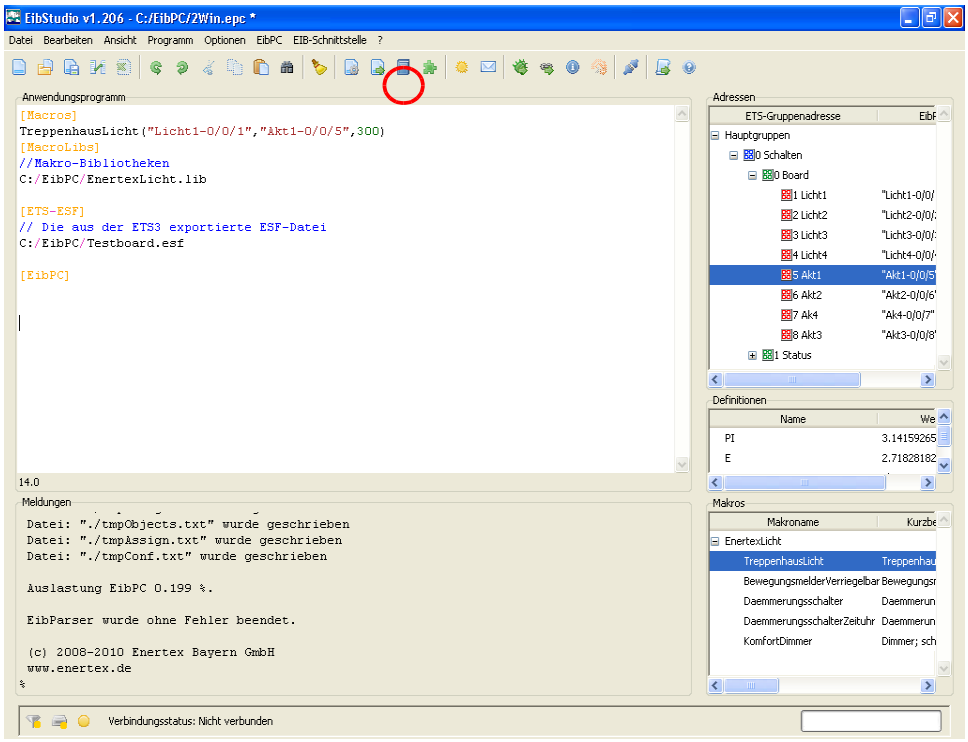


Abbildung 18: Import der Gruppenadressen ins Enertex® Eibstudio

Drücken Sie nun auf die markierte Schaltfläche von Abbildung 18. In diesem Fall geht ein Dialog auf und Sie können Bibliotheken hinzufügen. Wählen Sie „EnertexBeschattung.lib“ (im Downloadbereich von www.eibpc.com). Anschließend werden nun die Makros angezeigt, die in der Bibliothek abgespeichert sind (vgl. Abbildung 19).

Nun können Sie für das Ost-Fenster das Makro BeschattungRolloOstZeit klicken und anschließend parametrieren (vgl. Abbildung 22).

Makro auswählen

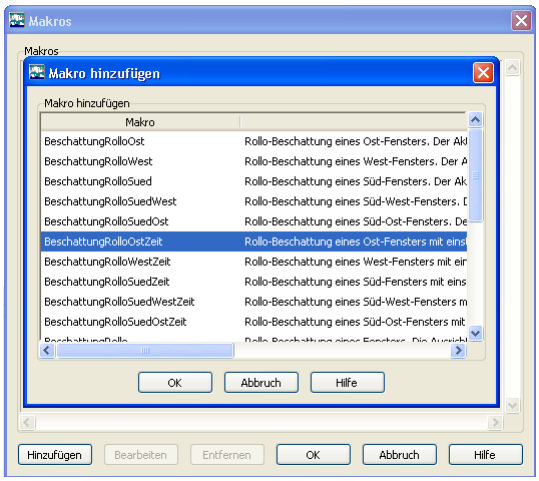


Abbildung 19: Die Makros der Bibliothek EnertexBeschattung.lib

Nun wird dieses Makro für alle drei Fenster parametriert. Die komplette Automatisierung für die drei Fenster ist in Abbildung 20 angegeben.

Hinweis 1:

Sie können anstelle mit den Assistenten zu arbeiten, auch direkt im Textfenster des Anwenderprogramms die Daten eingeben bzw. modifizieren.

Hinweis 2:

Makro bearbeiten

Der Sonnenstand ist abhängig vom Standort des Hauses. Die Standardvorgaben sind etwa in der Mitte Deutschlands vorgegeben. Sie können diese Vorgaben ändern bzw. genauer einstellen, indem Sie im Menü OPTIONEN – KOORDINATEN FÜR DIE SONNENSTANDSBERECHNUNG EINSTELLEN entsprechend Ihres Aufstellungsorts verändern (vgl. Seite 157).

Die Automatisierung ist fertig.

Anwendungsprogramm

```
[Macros]
// Makros
BeschattungRolloOstZeit("Freigabe-Beschattung-2/5/0","Fenster-Ost-2/5/1","Fenster-Ost-2/5/1",5000)
BeschattungRolloOstZeit("Freigabe-Beschattung-2/5/0","Fenster-Süd-2/5/2","Fenster-SüdStop-2/5/5",4500)
BeschattungRolloOstZeit("Freigabe-Beschattung-2/5/0","Fenster-West-2/5/3","Fenster-WestStop-2/5/6",4000)


[MacroLibs]
//Makro-Bibliotheken
C:/EibPC/EnertexBeschattung.lib

[ETS-ESF]
// Die aus der ETS3 exportierte ESF-Datei
C:/EibPC/Testboard.esf

[EibPC]
```

Abbildung 20: Die Beschattung der Fenster – Die komplette Automatisierung.

Visualisierung mit dem iPhone

Wenn Sie über einen Enertex® EibPC mit Software Option NP verfügen, können Sie auch Ihre Hausautomatisierung mit dem Enertex® EibPC über iPhones o.ä. visualisieren. Sie benötigen allerdings hier ein zusätzliches Tool, z.B. Command Fusion, um das iPhone für die Kommunikation zu ertüchtigen.

Mit Hilfe von Command Fusion können grafisch anspruchsvolle Visualisierungen für das Apple iPhone oder Ipod generiert werden. Unter www.knx-user-forum.de/ finden Sie einen Link zum freizugänglichen Design, das an die Design-Gestaltung der Enertex® Bayern GmbH angelehnt ist.

In der Bibliothek EnertexCommandFusion.lib werden hierzu entsprechende Makros vorgegeben, mit der die Anbindung erfolgen kann. Die Anwendung der Makros ist genauso, wie eben erklärt. Allerdings ist die Bedienung von Command Fusion und das dafür benötigte Hintergrundwissen nicht Bestandteil dieser Anleitung. Unter www.knx-user-forum.de/ finden Sie hierzu weitere Artikel. Unter www.enertex.de/downloads/d-eibpc/DokuCF-1.pdf finden Sie ein ausführliches Beispiel für die Implementierung einer Anbindung an das Command Fusion Tool.

EibPC und iPhone – ein Dreamteam...

Abbildung 21: Visualisierung mit dem iPhone

Erste Schritte bei der Programmierung

Im folgendem Abschnitt zeigen wir Ihnen, wie die Programmierung des Enertex® EibPC erfolgt. Hierzu unterweisen wir Sie Schritt für Schritt in die einfache und klein gehaltene Programmiersprache des Enertex® EibPC ein.

Fünf Schritte zum Erstellen eines Programms

Das Erstellen eines Programms kann in wenigen Schritten erfolgen:

1. Enertex® EibPC installieren (s. Seite 18).
2. Enertex® EibPC und Enertex® EibStudio konfigurieren (LAN-Setup s. Seite 18ff bzw. Seite 143).
3. Optional: Gruppenadressen Ihres ETS Projekts importieren.
4. Anwenderprogramm mit EibStudio erstellen.
5. Anwenderprogramm in den Enertex® EibPC übertragen und starten.

Für die nun folgende Einführung in die Programmierung des Enertex® EibPC müssen Sie zuerst die ersten beiden Schritte „Installation“ und „Konfiguration“ ausgeführt haben. Sie können es aber auch einfach lesen, um sich mit Ihrem Enertex® EibPC vertraut zu machen. Die weiteren Schritte erläutern wir in diesem Teil der Anleitung.

Aufbau der Schritt-für-Schritt Anleitung

Im Folgenden wollen wir Sie mit Beispielen Schritt für Schritt an die Bedienung des Enertex® EibStudios und den Möglichkeiten des Enertex® EibPC heranführen.

Enertex® EibStudio = Entwicklungsumgebung für den Enertex® EibPC

Enertex® EibStudio ist ein Applikationsprogramm für Windows®- und Linux®-Rechner. Es ist die Entwicklungsumgebung für die Programmierung des Enertex® EibPC.

Mit dem Enertex® EibPC erwerben Sie sich eine Lizenz zur Nutzung dieses Programms. Das Enertex® EibStudio und sämtliche Komponenten dürfen nur in Verbindung mit dem Enertex® EibPC genutzt werden.

Anwenderprogramme werden im Enertex® EibPC in nur 1 ms Zykluszeit abgearbeitet

Mit dem Enertex® EibStudio lassen sich Anwenderprogramme benutzerfreundlich erstellen, verändern und warten. Die Datei wird auf Knopfdruck mit dem integrierten Compiler, dem so genannten EibParser, übersetzt (kompiliert) und anschließend an den Enertex® EibPC gesendet. Der EibParser optimiert und kontrolliert den Code des Anwenders und stellt damit sicher, dass auch umfangreiche Programme innerhalb der Zykluszeit des Enertex® EibPC abgearbeitet werden können. Die Zykluszeit des Enertex® EibPC beträgt etwa 1 ms. (Weitere Einzelheiten finden Sie ab Seite 133.)

Ein einfaches Programm ohne Gruppenadressenimport

Beispiel 1: Ein Schalter, ein Dimmer und ein Schaltaktor

Wie in Abbildung 1 dargestellt ist der EibPC über eine RS232-Schnittstelle mit FT1.2-Protokoll an den KNX™-Bus angeschlossen. Zusätzlich gibt es noch einen Schalter, der auf die Adresse '0/0/1' ein Ein- bzw. Aussignal sendet, einen Schaltaktor für eine Lampe, die auf Adresse '1/1/1' empfängt und einen Dimmer, der auf Adresse '1/1/2' einen Prozentwert empfängt.

Die Adressen wurden in der ETS für die KNX™-Aktoren und Schalter entsprechend vergeben und programmiert. Der Dimmer ist dabei so parametriert, dass er den Prozentwert, der über die Adresse '1/1/2' gesendet wird, sofort einstellt. D.h. der Dimmer hat kein zusätzliches Schaltobjekt definiert sondern, eine angeschlossene Lampe leuchtet bei Werten > 0% mit der angegebenen Helligkeit.

Wir nehmen an, dass zunächst die Adressen nicht aus der ETS exportiert werden. Sie benötigen daher für das folgende Beispiel keine ets. Ihnen muss lediglich bekannt sein, welche Gruppenadressen vergeben wurden.

Ihr KNX™-Netz ist wie folgt parametriert:
Schalter sendet auf '1/0/0'
Aktor (Lampe) empfängt auf '1/1/1'
Dimmer empfängt auf '1/1/2'

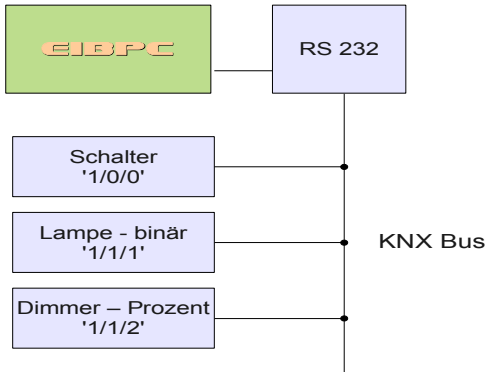


Abbildung 22: Blockschaltbild Beispiel 1

Die Aufgabe ...

Sie wollen nun folgende Funktionalität in Ihrem KNX™-Netz realisieren:

... und was ist das Problem?

Wird der Schalter „EIN“ gedrückt, soll die Lampe einschalten und der Dimmer auf 80% gehen.
Wenn er auf AUS geht, sollen die Lichter ausgehen.

Hintergrundwissen
Wie Sie wissen, können Sie dies nicht mit KNX™-Programmierung durch Vergabe von Gruppenadressen erreichen, denn der Schaltaktor erwartet einen Binärwert (EIN oder AUS) und der Dimmer einen Prozentwert. Dies bedeutet, der Schalter muss zwei verschiedene Telegramme auslösen. Über den Schalter können Sie jedoch bestenfalls nur ein Telegramm auslösen. Die meisten Schalter haben nicht einmal die Möglichkeit, einen Prozentwert auf den Bus zu schicken. So einfach die Aufgabe auch scheint: Ohne Automatisierung ist sie nicht lösbar. Vielleicht denken Sie bei diesem Beispiel auch an Szenenaktoren. Dazu kommen wir dann später noch.

Voraussetzungen:
- Der Enertex® EibPC ist im LAN installiert (s.Seite 18).
- Enertex® EibStudio ist konfiguriert (LAN-Setup von Seite143).

Mit dem Enertex® EibPC ist diese Aufgabe einfach zu lösen. Wir nehmen dazu an, Sie haben beispielsweise die Dokumentation des KNX™-Netzes vorliegen (Sie wissen also, welcher Aktor mit welcher Adresse parametriert ist), aber Sie haben keinen Zugriff auf die ETS oder auf die Orginal-Datensätze.

Zunächst starten Sie nun das Enertex® EibStudio. Sie sehen dann ein Fenster wie in Abbildung 23 dargestellt.

Der Bereich „Anwenderprogramm“ ist der integrierter Texteditor zum Erstellen von Programmen

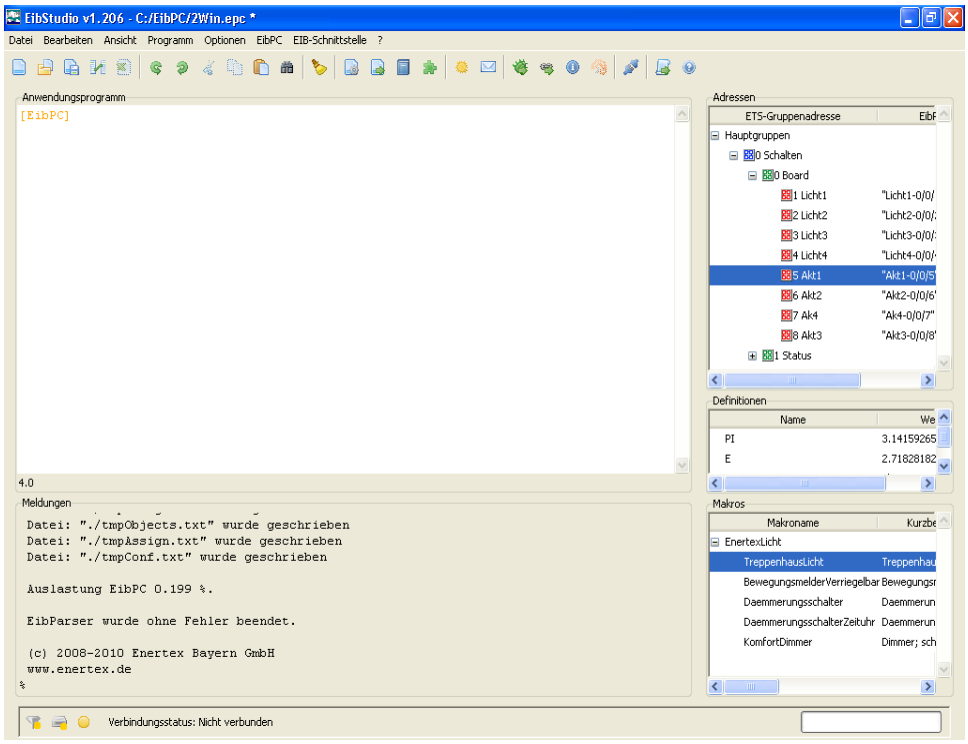


Abbildung 23: Enertex® EibStudio

Dieses „Programm“ müssen Sie in den Bereich Anwenderprogramm kopieren oder manuell eingeben

Unterscheiden Sie:
0 und O
„Null“ und Buchstabe „O“

Achten Sie auf den linken weißen und noch leeren Bereich „Anwenderprogramm“. Dieser Bereich des Enertex® EibStudio stellt einen integrierten Texteditor dar, da das Anwenderprogramm ein einfacher Text ist.

Zurück zum Problem: Ein Taster – zwei Telegramme. Lesen Sie das folgende Programm einfach durch, wir erklären es Schritt für Schritt. Aber Sie sollten bereits erkennen können: Einfacher geht es nicht.

```
[EibPC]
if ('1/0/0'b01==EIN) then write('1/1/1'b01,EIN); write('1/1/2'u08,80%) endif
if ('1/0/0'b01==AUS) then write('1/1/1'b01,AUS); write('1/1/2'u08,0%) endif
```

Kopieren Sie jetzt einfach den Text per Zwischenablage in den Bereich „Anwenderprogramm“.

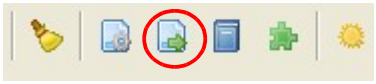


Abbildung 24: Kompilieren und an den Enertex® EibPC senden

Knopf oder Menü oder Shortcut

Dieses Programm muss nun in das Enertex® EibStudio eingegeben werden und anschließend in den Enertex® EibPC übertragen werden. Dazu drücken Sie den markierten Knopf (Abbildung 24) oder klicken im Menü „PROGRAMM“ auf die entsprechende Schaltfläche oder nutzen Sie den Shortcut „STRG-UMSCHALT-B“.

Siehe auch:
Syntaxhervorhebung (Seite 136)
Autovervollständigung (Seite 136)

Sie erkennen in Abbildung 25 die Syntaxhervorhebung: Enertex® EibStudio hilft Ihnen, Programme bereits beim Erstellen korrekt einzugeben. Zur Syntaxhervorhebung und Autovervollständigung von Schlüsselwörtern können Sie auf Seite 136 ff. Weiteres finden.

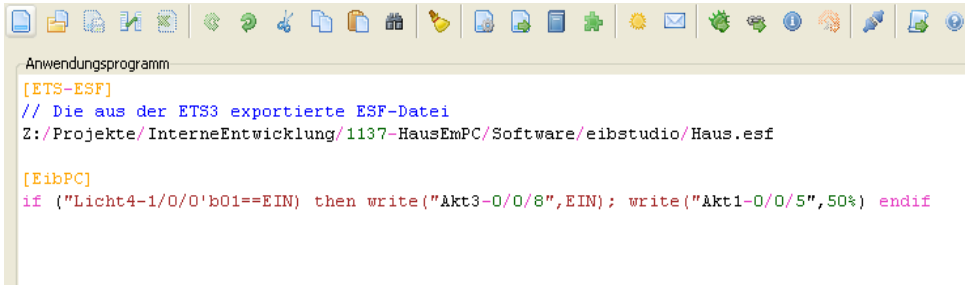


Abbildung 25: Syntaxhervorhebung im Enertex® EibStudio

if - Anweisung

Betrachten wir den Code etwas genauer. Unsere Aufgabe besteht aus einer bedingten Anweisung („*Wenn* der Schalter gedrückt ...“). Dazu gibt es bei der Programmierung die wichtige if-Anweisung.

```
if ('1/0/0'b01==EIN) then write ('1/1/1'b01,EIN); write ('1/1/2'u08,80%) endif
```

Die Syntax einer if-Anweisung ist wie folgt:

```
if (Abfragebedingung) then Anweisung endif
```

Der Anweisungsblock

Wenn die Abfragebedingung erfüllt ist, wird die Anweisung im then-Zweig einmal ausgeführt. Die if-Anweisung reagiert also nur auf Änderungen. Dazu im weiteren Verlauf dieser Einführung mehr.

Es können auch mehrere Anweisungen als Block ausgeführt werden. In diesem Fall sind die Anweisungen durch Strichpunkte zu trennen. **Nach der letzten Anweisung steht kein Strichpunkt**.

Eine Abfragebedingung wird dann als erfüllt angesehen, wenn Sie den Wert 1 annimmt. Nimmt Sie den Wert 0 an, so wird der then-Zweig nicht ausgeführt.

Manuelle Gruppenadresse

Was bedeutet '1/0/0'b01==EIN?

'1/0/0'b01 ist die Syntax für eine Gruppenadresse 1/0/0 mit einem binären Datentyp. Wir haben die Gruppenadresse und ihre Bezeichnung nicht aus der ETS importiert, sondern direkt verwendet. In diesem Handbuch reden wir bei dieser Schreibweise von „manuell“ eingegebenen Gruppenadressen. Das “Bildungsgesetz“ hierfür ist einfach die Gruppenadresse eingerahmt von zwei '-Zeichen (weitere Informationen sind auf Seite 160 zu finden). Somit ist das „Bildungsgesetz“ für die Syntax von manuell vergebenen Adressen:

Der Datentyp muss bei der manuellen Gruppenadresse und bei Werten mit angegeben werden!

Manuelle Adresse: 'Gruppenadresse' Datentyp

Gruppenadressen tragen unterschiedliche Informationen, abhängig von der Anzahl der Bits im dazugehörigem KNX™-Telegramm und der Deutung dieser Informationen. Wir sprechen hier von Datentypen. Wenn wir die manuelle Gruppenadresse und Wert verwenden, müssen wir den Datentyp kennen und mit angeben.

In unserem Beispiel verwenden wir zwei Datentypen (Weiteres auf Seite 160):

- den Binärwert (Werte 0 und 1), gekennzeichnet durch den Zusatz 'b01' und1
- den Prozentwert (Werte 0.0 bis 100.0), gekennzeichnet durch den Zusatz '%'.

„Eins“ oder „Ein Prozent“?

Der Wert 1 kann also dem binären Datentyp oder dem prozentualem Datentyp zugeordnet werden. Wenn wir 1b01 schreiben, beziehen wir uns auf den binären, schreiben wir 1% meinen wir den prozentualen Datentyp. Gleichermaßen geben wir bei der Schreibweise der Gruppenadresse als Zusatz den Datentyp mit an: '1/0/0'b01. Dies bedeutet, wir wollen auf die Adresse 1/0/0 eine Information der Länge ein Bit übertragen.

Etwas Hilfe: Vordefinierte Werte

Enertex® EibStudio kennt vordefinierte Konstanten, um das Programm lesbarer zu machen. Die Konstante AUS ist gleichbedeutend mit 0b01 und EIN mit 1b01.

Da 1b01 und 1% unterschiedliche Datentypen sind, können Sie diese nicht direkt miteinander verknüpfen, also ist etwa die Zuweisung zu einer Variablen der Form *Fehler=1b01 + 1%* nicht möglich und wird als Fehler beim Kompilieren des Programms gemeldet. Wir benötigen für die Verknüpfung unterschiedlicher Datentypen die convert-Funktion. Wir kommen darauf später zurück. (Auf Seite 57 bzw. Seite 217 finden Sie hierzu mehr).

Eine Übersicht der vordefinierten Variablen ist auf Seite 348 aufgelistet. Dabei besteht ein Datentyp immer aus einem Buchstaben und zwei Ziffern. Ausnahme hiervon ist nur der Prozentwert. Hier reicht das Prozentzeichen. Weiteres zu Datentypen finden Sie auf Seite 160 bzw. im weiteren Verlauf dieser Schritt-für-Schritt-Anleitung.

Der Datentyp „%“ ist kompatibel zum Datentyp „u08“ und wird durch Skalierung intern angepasst (wobei der Datentyp in Zehntel Prozent Genauigkeit angegeben werden kann). 100% entspricht intern einem Wert von 255.

Und wieder zurück zum Beispiel

Damit kommen wir wieder zurück zu Ausdruck: '1/0/0'b01==EIN

Ein Vergleich wird mit „==“ programmiert.

Die beiden Gleichheitszeichen == sind für den Vergleich zuständig. Das Ergebnis eines Vergleichs von zwei Ausdrücken (Variablen, Gruppenadressen, etc.) kann nur die folgenden Werte annehmen:

0b01, wenn die beiden Werte nicht gleich sind,

1b01, wenn die beiden Werte gleich sind.

Weitere Vergleichs-Verknüpfungen finden Sie auf Seite 178.

Wenn nun auf den Bus die Adresse '1/0/0' der Wert 1b01 (EIN) gesendet wird, wird der Vergleich den Wert EIN annehmen. Damit wird die Bedingung der if-Anweisung wahr (entspricht dem Wert 1b01 bzw. EIN) und der then-Zweig wird ausgeführt. Mit der write-Funktion (s.Seite 169) wird ein gültiges EIB-Telegramm an die angegebene Adresse mit dem angegebenen Wert auf den Bus geschrieben.

Die Anweisung

```
write('1/1/1'b01,EIN)
```

schreibt auf die Gruppenadresse 1/1/1 den Wert 1 (Datentypen 1b01).

Nun dürften Sie

```
if ('1/0/0'b01==EIN) then write('1/1/1'b01,EIN); write('1/1/2'u08,80%) endif
```

restlos verstanden haben. Wenn nicht, lesen dieses Beispiel nochmals durch, bevor Sie das nächste Beispiel bearbeiten.

Ein Programm mit Gruppenadressenimport

Importdatei einbinden

Beispiel 2: Alles wie gehabt – aber nun mit Importadressen

Der Aufbau bleibt wie in Abbildung 1 beschrieben. Nun wollen wir in diesem Beispiel mit den aus der ETS exportierten Adressen arbeiten, dazu können Sie die Adressen wie auf Seite 158 beschrieben, exportieren. Um diese Adressen nutzen zu können, müssen sie in das Enertex® EibStudio importiert werden.

Sehen Sie dazu den Dialog Datei in Abbildung 26, klicken Sie Gruppenadressen aus ETS-Exportdatei importieren. (Näheres hierzu finden Sie auch auf Seite 152.)

[Sektionen]

Damit neben dem Programm auch verschiedene Daten des Projekts wie z.B. Aufstellungsorts, E-Mail-Konfiguration etc. verarbeitet werden können, gibt es innerhalb des Anwenderprogramms sogenannte Sektionen. Diese sind durch einen Namen in eckigen Klammern gekennzeichnet. Eine Sektion beginnt mit ihrem Namen und endet mit dem Anfang der nächsten Sektion oder dem Ende der Datei. Zu Sektionen finden Sie auch auf Seite 133 weitere Informationen.

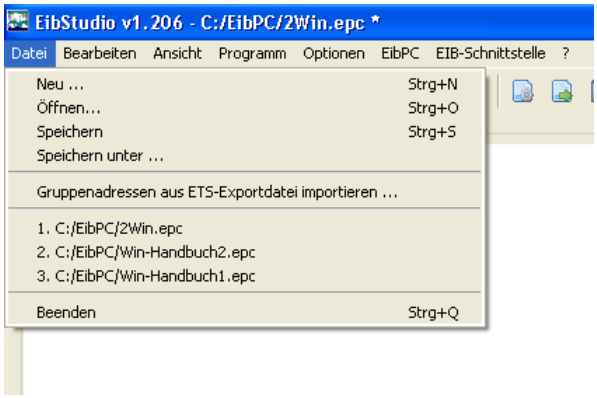


Abbildung 26: ETS Exportdatei laden

Der Pfad zur Datei wird in der Sektion [ETS-ESF] angezeigt. Jetzt „kennt“ Enertex® EibStudio und damit der Enertex® EibPC die Gruppenadressen aus Ihrer ETS Programmierung. Im rechten oberen „Adressen“ benannten Bereich tauchen nun alle Adressen der ETS auf. Per Drag&Drop oder über STRG+C bzw. STRG+V können Sie nun die Adressen im Anwendungsprogramm nutzen (zur Darstellung der Gruppenadresse siehe nächste Seite)

Importierte Adressen

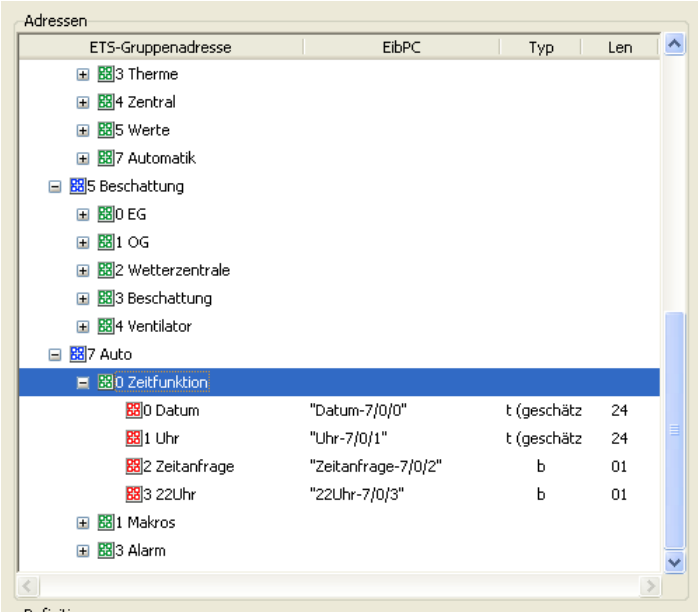


Abbildung 27: Importierte Adressen

Zurück zum Problem: Ein Taster – zwei Telegramme, jetzt mit importierten Gruppenadressen. Unsere Umsetzung lautet nun:

```
[EibPC]
if ("Schalter-1/0/0"==EIN) then write("Lampe-1/1/1",EIN); write("Dimmer-1/1/2",80%) endif
if ("Schalter-1/0/0"==AUS) then write("Lampe-1/1/1",AUS); write("Dimmer-1/1/2",0%) endif
```

Das meiste dürfte nun bekannt sein. Aber wir beleuchten nochmals die Gruppenadresse.

Was bedeutet "Schalter-1/0/0“?
Eine Gruppenadresse besteht in der ETS aus drei Namensteilen:

Nur der Untergruppenname wird verwendet.

1. Name der Hauptgruppe
2. Name der Mittelgruppe
3. Name der Untergruppe

Nur in der Kombination der drei Namen ist die Gruppenadresse eindeutig. Um den Namen nicht vollständig zu importieren und somit den Code unleserlich zu gestalten, verwenden wir importierte Gruppenadressen mit Hilfe des Untergruppennamens. Da dies aber nicht eindeutig ist, schreibt die Importfunktion durch ein „-“ Zeichen automatisch die physikalische Gruppenadresse dazu. Das Ganze wird dann von zwei Anführungszeichen eingerahmt. Also lautet das „Bildungsgesetz“ für importierte Gruppenadressen:

Datentyp ist optional

Importierte Adresse: "Untergruppenname-Gruppenadresse“{Datentyp}

Datentypen der Gruppenadressen werden indirekt aus der Verwendung ermittelt und müssen vom Anwender nicht angegeben werden.

Der Export auf Seiten der ETS-Software ist nicht vollständig implementiert. Insbesondere bei Datentypen mit Bitlängen von 8 oder mehr Bits werden nur die Bitlängen und oftmals nicht deren Deutung exportiert. Daher kann man im obigen Beispiel write("Dimmer-1/1/2"u08,80%) den Datentyp des Prozentwerts als Anhang der Gruppenadresse (u08) mit angeben. Allerdings überprüft der Compiler sowieso die Datentypen auf Ihre Verknüpfbarkeit: Im Falle der write Funktion bedeutet dies, dass deren 1. Argument, also die Gruppenadresse, vom selben Datentyp sein **muss** wie das zweite, im Beispiel der Prozentwert. Dadurch ist der Datentyp der Gruppenadresse indirekt durch die Verwendung in der write-Funktion festgelegt. Wir nennen dieses Vorgehen implizite Konvertierung (siehe auch Seite 160).

Hintergrundwissen:
Datentypen konsequent umgesetzt

Der Compiler kennt die Datentypen, Rückgabewerte und Vorgaben der Funktionen wie z.B. Datentyp des 1. Arguments muss gleich dem 2. Arguments von write sein. Wenn daher nur das 2. Argument einen eindeutigen Datentypen hat, so liegt das 1. Argument eindeutig fest. Wenn die Datentypen nicht zueinander passen, gibt der Compiler eine Fehlermeldung aus.

Ein weiteres Beispiel: Die Funkion sin (Sinus, Seite 215) berechnet den Sinus einer 32-Bit Fließkommazahl im Bogenmaß. Hier weiß der Compiler: sin gibt eine Fließkommazahl zurück und erwartet als Argument ebenso eine Fließkommazahl. Daher wird in einem Programmauszug „a=sin(b)“ eindeutig festgelegt, dass die Variable a und die Variable b vom Datentypen Fließkommazahl, 32 Bit, sein müssen.

Aus diesem Grund können Sie auch nicht a=2u16+4u32 programmieren. Bei Summen legt der Compiler fest: Alle Summanden einer Summe müssen vom gleichen Datentyp sein. Sollten Sie Zahlen unterschiedlicher Datentypen summieren wollen, müssen Sie auf die convert-Funktion zurückgreifen.

Der Vorteil dieser „strengen“ Datentypregelung ist, dass Sie sich niemals um die Anpassung der importierten Gruppenadressen bemühen müssen. Ihr Programm wird auch keine „überraschenden“ Ergebnisse liefern, wie sie es bei der ungewollten Umwandlung von z.B. 32 Bit Zahlen in 16 Bit Zahlen durch das „Abschneiden“ der höherwertigen Bits entstehen werden.

Die Angabe eines Datentyp ist immer optional, auch wenn der Import der Gruppenadressen nicht eindeutig ist. Sollte einmal dadurch ein Problem entstehen, weist Sie aber der integrierte Compiler des Enertex® EibStudio darauf hin.

Zurück zum Anwenderprogramm: Wir können das Ganze noch vereinfachen, indem wir anstelle der zweiten if-Funktion die erste mit einem else-Zweig ausstatten. Dabei gilt es zu berücksichtigen, dass eine Anweisung nicht durch einen Zeilenumbruch unterbrochen werden darf. Der else-Zweig wird ausgeführt, wenn die Abfragebedingung nicht erfüllt ist. Damit lautet die Syntax:

if (Abfragebedingung) then Anweisung{sblock}1 else Anweisung{sblock}2 endif

Die Umsetzung im Anwenderprogramm lautet damit

```
[EibPC]
if ("Schalter-1/0/0"==EIN) then write("Lampe-1/1/1",EIN); write("Dimmer-1/1/2"u08,80%) else write("Lampe-1/1/1",AUS); write("Dimmer-1/1/2"u08,0%)
endif
```

„Warum ist das so klein geschrieben?“, mögen Sie sich fragen. Die Antwort ist:
Wir wollen andeuten: Eine Anweisung darf nicht einfach mit einem Zeilenumbruch („RETURN“) unterbrochen werden. Sonst würde das Enertex® EibStudio beim Kompilieren einen Fehler anzeigen. Das Enertex® EibStudio verarbeitet Anweisungen immer zeilenweise.

Daher nochmal eindringlich:

Zeilenumbruch

Eine Anweisung darf nicht durch einen Zeilenumbruch unterbrochen werden.

Um über mehrere Zeilen größere Anweisungen übersichtlich zu gestalten, verwenden sie als Umbruch zwei Backslash (\\)-Zeichen.

Alles wird gut leserlich
*Der Umbruch mit *

Wir können unser Programm damit etwas übersichtlicher schreiben. Außerdem vereinfachen wir noch mit Hilfe des else-Zweiges:

```
[EibPC]
if ("Schalter-1/0/0"==EIN) then                    \\
                    write("Lampe-1/1/1",EIN);      \\
                    write("Dimmer-1/1/2"u08,80%)   \\
                else                               \\
                    write("Lampe-1/1/1",AUS);      \\
                    write("Dimmer-1/1/2"u08,0%)    \\
                endif
```

Noch schöner?

Wir empfehlen, das Anwenderprogramm wie eben gezeigt, zu formatieren. Dies erhöht die Lesbarkeit für Sie enorm und vermeidet somit Fehler.

Auch hier nochmal zur Erinnerung: Sollen mehrere Anweisungen als Block aufgefasst werden, ist zwischen Anweisungen ein Strichpunkt einzufügen.

Bei If-Anweisungen können Sie alternativ mit geschweiften Klammern { } arbeiten.

```
[EibPC]
if ("Schalter-1/0/0"==EIN) then {
                    write("Lampe-1/1/1",EIN);
                    write("Dimmer-1/1/2"u08,80%)
                } else {
                    write("Lampe-1/1/1",AUS);
                    write("Dimmer-1/1/2"u08,0%)
                } endif
```

Nun können wir unser Programm, wie in Abbildung 25 gezeigt, starten.

Nochmal if

Eine Anmerkung zur if-Anweisung:

Die Abfragebedingung der if-Anweisung muss vom Typ b01 sein, also binär, 1 Bit Länge. Der integrierte Compiler von Enertex® EibStudio überprüft dies. Unsere Variable "Schalter-1/0/0" ist von Typ b01. Daher ist die Abfrage

```
if ("Schalter-1/0/0"==EIN) then          \\
```

überflüssig (aber nicht falsch). Sie hätten in diesem Fall

```
if ("Schalter-1/0/0") then          \\
```

programmieren können. Falsch wäre allerdings

```
if ("Dimmer-1/1/2") then          \\
```

da "Dimmer-1/1/2“ eine vorzeichenlose ganze Zahl darstellt. Der Compiler bemerkt dies und gibt einen Fehler aus.

Kommentare?

Sie können ihre Programmierung kommentieren. Dazu gibt es zwei Möglichkeiten:

1. Kommentare beginnen mit „//“ und stehen am Anfang einer Zeile.
2. **Anstelle** einer Anweisung können Kommentare an beliebiger Stelle stehen (Strichpunkt beachten), die durch /* */ eingerahmt werden. Z.B.

```
/* Dies ist ein Kommentar */
// Das ist noch ein Kommentar
u=5;/* Und dies ist noch Kommentar */; u4=5
```

Gratulation.

Sie können bereits die wichtigsten Grundlagen und haben verstanden, wie einfach die Programmierung des Enertex® EibPC erfolgt.

Inbetriebnahme: Der Systemstart

Ein Wechseltaster

Die Vorbelegung ist immer null.

Schauen Sie nochmals das Beispiel von eben an: Wenn auf der Gruppenadresse "Schalter-1/0/0" (oder bei der manuellen Eingabe '1/0'0'b01) ein EIN gesendet wird, soll das Licht angehen. Die Standard-Vorbelegung von Variablen (dazu später mehr) und Gruppenadressen usw. ist null (AUS, 0, 0.0 ...). Alle Objekte Ihres Anwenderprogramms haben die Eigenschaft „gültig“ (valid).

Was ist aber, wenn der Lichtschalter bereits gedrückt wurde, also eigentlich auf EIN steht, bevor der Enertex® EibPC gestartet wurde? Wie kann optimalerweise das Programm gestaltet werden? „Manchmal kann man damit leben, dass der Anwender in solchen Fällen EIN und AUS und wieder EIN drücken muss, bis etwas passiert“ könnten Sie sagen.

Aber der Enertex® EibPC kann dies viel besser.

Wir haben hierzu die Funktion:

systemstart()	Rückgabe 1b01 beim Systemstart

Die Funktion erwartet kein Argument. Nachdem der Enertex® EibPC das Anwenderprogramm gestartet hat, ist der binäre Rückgabewert 1b01 (weitere Dokumentation der Funktion finden Sie auf Seite 221).

Bevor wir die Anwendung dieser Funktion zeigen, benötigen wir noch:

read(*Gruppenadresse*)	Leseanforderung für die Gruppenadresse

Mit read sind wir in der Lage den Aktor, der auf der angegebenen Gruppenadresse sendet, zu „befragen“, welchen Wert er gerade hat. (Weiteres siehe auch auf Seite 170).

Wichtig: Damit der Aktor auch tatsächlich antwortet, muss das Lesen-Flag in der ETS gesetzt werden.

Nun zum eigentlichen Programm:

Bei manuellen Gruppenadressen:
read('1/0/0'b01)

```
[ETS-ESF]
Haus.esf
[EibPC]
if (systemstart()) then read("Schalter-1/0/0") endif
if ("Schalter-1/0/0"==EIN) then                          \\
                    write("Lampe-1/1/1",EIN);        \\
                    write("Dimmer-1/1/2"u08,80%)  \\
                else
                    write("Lampe-1/1/1",AUS);        \\
                    write("Dimmer-1/1/2"u08,0%)   \\
                endif
```

Ein wichtiges Detail

Beim Systemstart wird nun auf den Bus die Leseanforderung geschickt. Der Aktor sendet seinen Wert. Anschließend wird im Anwenderprogramm die if-Anweisung abgearbeitet: Wenn der Schalter auf EIN steht und dies meldet, gehen die Lichter ein.

Wenn der Schalter nach Leseanforderung durch if systemstart() ein AUS auf 1/0/0 schickt, tut der Enertex® EibPC nichts. „Warum?“ Man könnte doch denken, dass in diesem Fall nach Eintreffen der Antwort vom Aktor der else-Zweig ausgeführt wird. Dies ist nicht der Fall, da ja beim Start alle Objekte und Anweisungen als gültig angenommen werden. Objekte werden dann ungültig, wenn diese sich ändern (Ausnahmen sind hier nur wenige Sonderfunktionen). Und nur wenn Sie sich ändern, lösen sie ihrerseits Aktionen aus. Der Enertex® EibPC „merkt sich“ immer die Zustände seiner Objekte.

Der else-Zweig ist bereits gültig, da ja alle Objekte mit Null vorbelegt werden. Die Anweisungen im else-Zweig werden beim Start des Anwenderprogramm aber nicht ausgeführt. Wenn nun die Antwort auf die Leseanforderung mit AUS (0b01) eintrifft, wird der else-Zweig nicht ausgeführt, da ja bereits das Objekt "Schalter-1/0/0" mit AUS initialisiert wird. Ein erneutes Eintreffen des Telegramm führt daher nicht zu einer Zustandsänderung und der Enertex® EibPC löst im else-Zweig keine Telegramme aus.

Wir können das Programm nun noch erweitern, so dass in jedem Fall zuerst die Aktoren ausgeschaltet werden. Es könnte ja sein, dass der Schaltaktor noch auf EIN steht.

Dann soll eine Leseanforderung den Zustand des Schalters abfragen und entsprechend das Licht geschaltet werden.

```
[ETS-ESF]
/EibPc/Haus.esf
[EibPC]
if (systemstart()) then                                  \\
                       read("Schalter-1/0/0") ;          \\
                       write("Lampe-1/1/1",AUS);         \\
                       write("Dimmer-1/1/2"u08,0%)       \\
                    endif

if ("Schalter-1/0/0"==EIN) then                                  \\
                                write("Lampe-1/1/1",EIN);        \\
                                write("Dimmer-1/1/2"u08,80%)     \\
                             else
                                write("Lampe-1/1/1",AUS);        \\
                                write("Dimmer-1/1/2"u08,0%)      \\
                             endif
```

Auch dieses Programm muss auf den EibPC überspielt werden.

Wenn Sie mehr als eine Gruppenadresse lesen wollen, können Sie die [InitGA] Sektion, wie auf Seite 171 beschrieben, nutzen. In diesem Fall, werden die Leseanforderungen automatisch vor der eigentlichen Verarbeitung Ihres Programms auf den Bus gegeben.

Initialisierung von mehreren Gruppenadressen

```
[ETS-ESF]
/EibPc/Haus.esf
[InitGA]
"Schalter-1/0/0"
[EibPC]
if ("Schalter-1/0/0"==EIN) then {
                                write("Lampe-1/1/1",EIN);
                                write("Dimmer-1/1/2"u08,80%)
                             } else {
                                write("Lampe-1/1/1",AUS);
                                write("Dimmer-1/1/2"u08,0%)
} endif
```

Prozessabbild: Nur auf Änderungen reagieren

Der Enertex® EibPC bildet die Aktivitäten am Bus in seinem Zwischenspeicher ab (Prozessabbild). Typischerweise wird alle ms einmal das Programm durchlaufen. Der Enertex® EibPC prüft dabei, an welcher Stelle diese Zwischenspeicher sich geändert haben und verarbeitet entsprechende Zeilen neu. Dies bedeutet im obigen Beispiel, wenn der Schalter einmal auf 1/0/0 ein EIN sendet (davor sei dies AUS), wird die Lampe EIN geschaltet. Im nächsten Programmablauf (nach 1 ms) steht ja der Zwischenspeicher von 1/0/0 bereits auf EIN und daher wird erwartungsgemäß nicht erneut ein EIN gesendet.

Getrennte Ein- und Ausschalter

Angenommen wir haben eine Installation wie in Abbildung 28.

Wird der EinSchalter „EIN“ gedrückt, soll die Lampe einschalten und der Dimmer auf 80% gehen. Wenn der Aus-Schalter auf AUS geht, sollen die Lichter ausgehen.

Ein Telegramm für EIN und und ein separates Telegramm für AUS

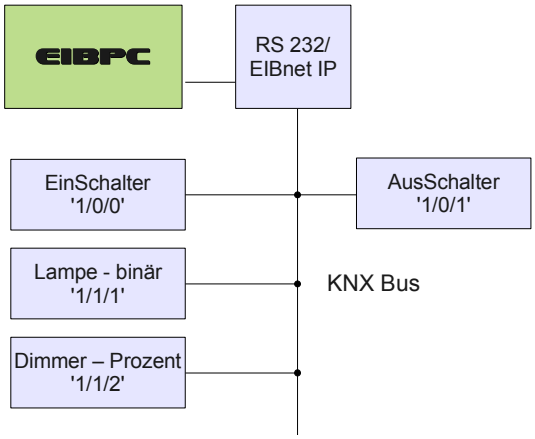


Abbildung 28: Blockschaltbild Ein- und Ausschalter

Unser Ein-Schalter sendet bei Betätigung immer nur EIN auf 1/0/0, der Aus-Schalter immer nur AUS auf 1/0/1. Aufgrund des Sachverhaltes mit dem Prozessabbild, reagiert der Enertex® EibPC nur auf Änderung. Wenn man dies vergisst, könnte man meinen, das folgende Programm macht das Richtige – was aber nicht der Fall ist - und wir erklären gleich warum.

Das geht nicht wie gewollt

```
// ACHTUNG: GEHT NICHT WIE GEWOLLT
if ("EinSchalter-1/0/0"==EIN) then                              \\
                              write("Lampe-1/1/1",EIN);       \\
                              write("Dimmer-1/1/2"u08,80%)  \\
                            endif
if ("AusSchalter-1/0/1"==AUS) then                              \\
                              write("Lampe-1/1/1",AUS);       \\
                              write("Dimmer-1/1/2"u08,0%)   \\
                            endif
```

Beim ersten Durchlauf (nach der Inbetriebnahme) wird ggf. das Programm noch einmal die Lampen ein- bzw. ausschalten. Aber wenn die erste bzw. zweite if-Anweisung einmal ausgeführt wurde und erneut ein EIN auf der Adresse 1/0/0 gesendet wird, passiert nichts. Schließlich ist ja das Abbild der Gruppenadresse bereits richtig ausgewertet und steht schon auf EIN. Wir müssen also wissen, ob eine Gruppenadresse erneut gesendet wurde.

Für solche Aufgabenstellungen gibt es die Funktion event:

```
event(Gruppenadresse)
```

Das geht.
*And-Funktion: Wenn ein Telegramm eintrifft **und** der Schalter auf EIN geht, soll auch nur eingeschaltet werden.*

Sie ist eine Funktion, die anzeigt, wenn auf dem Bus eine Nachricht mit der angegeben Gruppenadresse eingetroffen ist. Sie überprüft nicht, ob sich die Nachricht ändert, welchen Inhalt sie hat oder welchen Typ. Sobald eine Nachricht eintrifft, geht sie für einen Verarbeitungszyklus des Enertex® EibPC auf EIN. In Abbildung 29 ist dies schematisiert dargestellt. Weitere Event-Funktionen finden Sie auf S. 174ff.

```
// Alles Paletti
if event("EinSchalter-1/0/0") and  ("EinSchalter-1/0/0"==EIN) then     \\
                              write("Lampe-1/1/1",EIN);                \\
                              write("Dimmer-1/1/2"u08,80%)             \\
                            endif
if event("AusSchalter-1/0/1") and ("AusSchalter-1/0/1"==AUS) then     \\
                              write("Lampe-1/1/1",AUS);                \\
                              write("Dimmer-1/1/2"u08,0%)              \\
                            endif
```

So funktioniert event(): Man erkennt, dass auf beide Flanken reagiert wird.

Wir nutzen die and-Funktion (UND). Sie verknüpft die beiden Bedingungen der if-Anweisung. Mehr zu logischen Verknüpfungen finden Sie auf Seite 176 bzw. im weiterem Verlauf dieser Einführung. Dieses „Ver-Unden“ wird notwendig, wenn z.B. auf die Gruppenadresse 1/0/0 auch ein AUS gesendet werden kann, und hier andere Aktoren geschaltet werden sollen. Die Event-Funktion geht auch in diesem Fall auf EIN (1b01), da diese nur ein Ereignis anzeigt (siehe Abbildung 29).

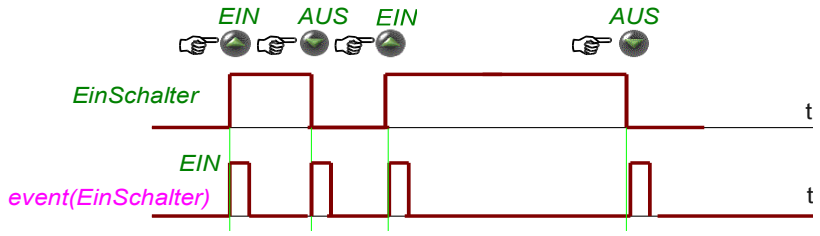


Abbildung 29: Funktionsweise event

Etwas Hintergrundwissen

Eine Besonderheit der event-Funktionen ist, dass diese nicht bei if-Anweisungen mit else-Zweig stehen dürfen. Aus folgendem Grund: Die Abfragebedingung der if-Anweisung

```
if event("EinSchalter-1/0/0") and ("EinSchalter-1/0/0"==EIN) then \\
        write("Lampe-1/1/1",EIN) else write("Lampe-1/1/1",AUS) endif
```

würde beim Eintreffen einer EIN Meldung des Schalters auf 1/0/0 für einen Verarbeitungszyklus auf EIN gehen und daher die if-Anweisung (then-Zweig) ausführen und die Lampe einschalten. Im nächsten Zyklus geht event wieder auf AUS (siehe Abbildung 29) und nun wird der else-Zweig ausgeführt. Die Lampe geht sofort wieder aus. Um dem Anwender hier die Sache zu vereinfachen, gibt der Compiler einen Fehler aus.

Eine Flurlichtsteuerung

Bewegungsmelder und Schalter

Beispiel 3: Ein zusätzlicher Bewegungsmelder

Der Aufbau wird nun um einen Bewegungsmelder mit der logischen Adresse “Bewegungsmelder-1/2/0“ erweitert. Der Dimmer soll wahlweise über einen Schalter oder mit einem Bewegungsmelder geschaltet werden.

Der Dimmer ist so parametriert, dass er über ein Schaltobjekt “Dimmer-1/1/1“ (binär, d.h. Datentyp b01) ein- und ausgeschaltet wird. Die Helligkeit des Dimmers wird mit der Gruppenadresse “DimmerWert-1/1/2“ übertragen. Abbildung 30 zeigt den prinzipiellen Aufbau.

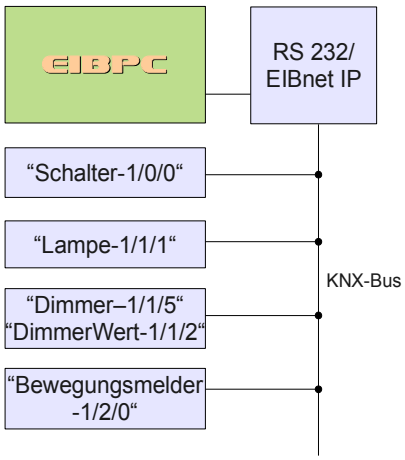


Abbildung 30: Blockschaltbild 3

Hier unsere Aufgabe

> Wird der Schalter „EIN“ gedrückt, soll die Lampe einschalten und der Dimmer auf 80% gehen. Wenn er auf AUS geht, sollen die Lichter aus gehen. Wenn der Schalter aktiv ist, soll der Bewegungsmelder deaktiviert werden.
>
> Sendet der Bewegungsmelder ein EIN-Telegramm, soll der Dimmer auf 50% seiner Leuchtkraft gehen.

Variablen

Namensbildung:
Buchstabe + beliebige Kombination von Buchstaben und Zahlen.
Keine Umlaute und Sonderzeichen!

Bevor wir uns an die Umsetzung machen, zeigen wir ein wichtiges Element der Programmierung: Variablen. Variablen sind Platzhalter für Zahlen, Werte und Telegramme. Im Enertex® EibStudio definieren Sie eine Variable auf verschiedenen Wegen (vgl. Seite 164). Dabei ist wieder eine Definition pro Zeile zu verwenden.

1. Variable=Wert
2. Variable=Gruppenadresse
3. Variable=Funktion()

Der Datentyp der Variable wird nicht wie bei Gruppenadressen am Ende angehängt. Der Datentyp wird ja entweder bei der Zahl oder bei der Zuweisung an eine Gruppenadresse direkt deutlich. Enertex® EibStudio überprüft beim Erstellen des Programms, ob die Zuweisung zu einem bestimmten Datentyp im Programm immer gleich ist. Wenn Sie einen Fehler machen, erhalten Sie vom Enertex® EibStudio eine Fehlermeldung.

Demnach sind

```
Dimmer=80%
BewegungsMelder=“Bewegungsmelder-1/2/0“
```

gültige Definitionen. Die erste definiert die Variable *Dimmer* auf 80% (=204u08). Die zweite weist der Variablen *Bewegungsmelder* den **Inhalt** des Telegramms zu, welches auf die Gruppenadresse “Bewegungsmelder-1/2/0“ gesendet wird.

Vom Anfang bis zum Ende und wieder von vorne...

Wie schon angemerkt, durchläuft der Enertex® EibPC das Anwendungsprogramm fortlaufend. Anschaulich gesprochen (und auch nicht ganz korrekt beschrieben), fängt der Enertex® EibPC wenn er an das Ende des Programms mit der Verarbeitung gekommen ist, von vorne wieder an.

Valdierung...

Die Verarbeitung erfolgt dabei immer nur auf Veränderung: Es werden die Zustände der eingegangen Telegramme gespeichert und nur bei Veränderung der Telegramme oder der Variable wird die Verarbeitung (Ausführen von Anweisungen) angestoßen. Wir bezeichnen dies als Validierungsschema.

Im Beispiel ist daher wichtig zu verstehen, dass die Anweisung „Setzte *Dimmer* auf 80%“ nur einmal beim ersten Durchlauf des Programms ausgeführt wird, da sich 80% ja nicht mehr ändert. *BewegungsMelder* hingegen hängt von der Gruppenadresse "Bewegungsmelder-1/2/0" bzw. eben den Telegrammen mit dieser Gruppenadresse am KNX™ Bus ab.

Ein kleiner Exkurs:

Setzen von Variablen in der if-Anweisung

*Im folgenden Anwendungsprogramm wird beim ersten Durchlauf die Variable DimmVar auf 80% gesetzt. Wenn der Bewegungsmelder auf der Gruppenadresse 1/2/0 ein EIN (1b01) sendet, so ist die if-Anweisung erfüllt und DimmVar wird auf 30% gesetzt. Beim erneuten Durchlauf wird DimmVar **nicht** auf 80% gesetzt, weil sich 80% ja nicht geändert hat. Man kann also eine Variable bzw. deren Inhalt in der if-Anweisung „überschreiben“:*

```
DimmVar=80%
Melder="Bewegungsmelder-1/2/0"
if Melder then DimmVar=30% endif
```

Setzen von Variablen in der if-Anweisung und durch Zuweisung einer Gruppenadresse

Im folgenden, etwas geänderten, Anwendungsprogramm wird beim ersten Durchlauf die Variable DimmVar auf den Wert der Grupennaddresse 5/8/1 gesetzt. Wenn der Bewegungsmelder auf der Gruppenadresse 1/2/0 ein EIN (1b01) sendet, so wird DimmVar auf 30% gesetzt. Alsbald beim erneuten Durchlauf vom Bus auf die Gruppenadresse 5/8/1 ein Telegramm eintrifft (und der Wert anders ist, als der zuletzt auf dieser Gruppenadresse übertragene) wird DimmVar auf diesen neuen Wert gesetzt.

```
DimmVar="Dimmer-5/8/1"
Melder="Bewegungsmelder-1/2/0"
if Melder then DimmVar=30% endif
```

Exakt eine Zuweisung in der „Hauptebene“ notwendig: Definition der Variable

Man kann die Zuweisung in der if-Anweisung als Zuweisung in der „2.-Ebene“ verstehen. Die „Hauptebene“ ist die Definition der Variablen (z.B. DimmVar=80%) ohne vorangestelltes „If“. In der Hauptebene darf eine Variable immer nur eine einzige Zuweisung erfahren, sonst compiliert der Code nicht. Dies bedeutet anschaulich, eine Variable darf im Anwendungsprogramm nur ein einziges mal „links stehen“. Daher ist folgendes Programm nicht lauffähig:

```
//ACHTUNG ungültiger Code
DimmVar="Dimmer-5/8/1"
DimmVar=30%
```

*Die Zuweisung in der Hauptebene nennen wir also die Definition der Variablen. Eine Variable **muss immer in der Hauptebene** definiert werden, bevor Sie im Programm verwendet werden kann. Daher wird im folgenden Code LogikB mit AUS in der Hauptebene definiert.*

Durch das Setzen der Variablen in der if-Anweisung entsteht die Möglichkeit, beliebig Verknüpfungen mit Hilfe von Variablen auszuarbeiten. Gleichermaßen ist dies aber auch schon bei der Definition der Variable möglich:

```
// Variante A
LogikA="Bewegungsmelder-1/2/0" and "Dimmer-5/8/1">50%
// Variante B (identisch zu A).
LogikB=AUS
if "Bewegungsmelder-1/2/0" and "Dimmer-5/8/1">50% then LogikB=EIN else LogikB=AUS endif
```

Die beiden Variablen LogikA und LogikB sind identisch – viele Wege führen nach Rom...

Unsere Aufgabe lässt sich nun mit Hilfe der Variablen wie folgt schreiben:

```
[ETS-ESF]
// Die aus der ETS exportierte ESF-Datei
EibPC/Haus.esf
[EibPC]

// Systemstart
if (systemstart()) then                              \\
                        read("Schalter-1/0/0") ;     \\
                        write("Lampe-1/1/1",AUS);    \\
                        write("DimmerWert-1/1/2"u08,0%)  \\
                    endif
if systemstart() then BewegungsMelder=AUS endif

// Variablen
Schalter="Schalter-1/0/0"
BewegungsMelder="Bewegungsmelder-1/2/0"
Dimmer=80%

// Der Schalter
if (Schalter==EIN) or (BewegungsMelder==EIN) then    \\
                                write("Lampe-1/1/1",EIN);    \\
                                write("Dimmer-1/1/5",EIN)    \\
                                endif

if (Schalter==EIN) then write("DimmerWert-1/1/2"u08,80%) endif

//Bewegungsmelder
if (BewegungsMelder==EIN) and (Schalter==AUS)  then \\
                                 write("DimmerWert-1/1/2"u08,50%) \\
                                 endif

if (Schalter==AUS) or (BewegungsMelder==AUS) then    \\
                                write("Dimmer-1/1/5",AUS); \\
                                write("Lampe-1/1/1",AUS)   \\
                                endif
```

Einschalten: Initialisierung des Programms

systemstart() kann man beliebig oft verwenden

Oder-Funktion

And-Funktion: Der Bewegungsmelder soll nur bei Schalter==AUS den-Dimmer anschalten.

In diesem Beispiel kommt die or-Funktion zum Einsatz. Sie erlaubt es, mit verschiedenen Aktoren eine logische Oder-Verknüpfung zu erstellen. D.h. wenn die Variable *Schalter* EIN ist **oder** die Variable *BewegungsMelder* EIN ist oder beide EIN sind, wird die if-Anweisung wahr und somit der then-Zweig ausgeführt. Die and-Funktion „ver-undet" die beiden Bedingungen. Sie können Ausdrücke vergleichen (größer, kleiner) und invertieren. Mehr zu logischen Verknüpfungen finden Sie auf Seite 176.

Fehlersuche leicht gemacht...

Bei der Programmierung macht auch der beste Profi Fehler. Um Ihr Programm zu „debuggen", d.h. nach Fehlern zu suchen, können Sie den Wert, der aktuell einer Variablen zugewiesen ist, abfragen. Abbildung 31 zeigt den Debugger-Knopf (Alternativ im Menu „EibPC-Wert eines EIB-Objekts abfragen"). Bei Betätigen erscheint ein Fenster mit Variablen, die Sie nun klicken können. Abbildung 32 zeigt z.B. die Abfrage der Konstanten π. Achten Sie hier auf das Meldungsfenster unten im Enertex® EibStudio.

Der integrierte Debugger

Abbildung 31: Der Debugger-Knopf

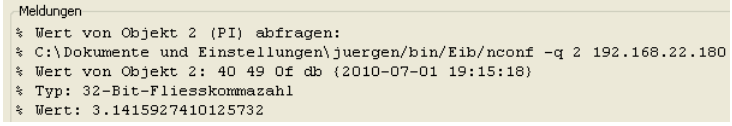


Abbildung 32: Wertabfrage

Ein Flurlicht mit Zeitsteuerung

Eine Zeitschaltuhr

Beispiel 4: Ein Bewegungsmelder, Schalter und Helligkeit je nach Uhrzeit

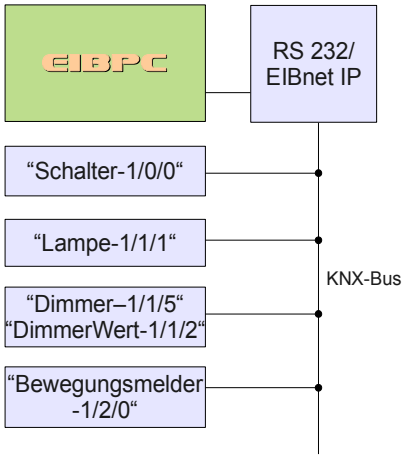


Abbildung 33: Blockschaltbild 3

Wir benutzen das selbe Beispiel wie in Abbildung 30 (Abb.33). Der Dimmer soll hier zeitgesteuert seine Intensität verändern.

Dazu unsere selbst gestellte Aufgabe:

Wird der Schalter „EIN“ gedrückt, soll die Lampe einschalten und der Dimmer auf 100% gehen. Wenn er auf „AUS“ geht, sollen die Lichter ausgehen. Wenn der Schalter aktiv ist, soll der Bewegungsmelder deaktiviert werden.

Sendet der Bewegungsmelder ein EIN-Telegramm, soll der Dimmer auf

Inbetriebnahme

- 50% seiner Leuchtkraft gehen, wenn es nach 20:00 Uhr ist
- 30% seiner Leuchtkraft gehen, wenn es nach 23:00 Uhr ist
- 10% seiner Leuchtkraft gehen, wenn es nach 3:00 Uhr ist
- 100% seiner Leuchtkraft gehen, wenn es nach 7:30 Uhr ist

Mit unseren Überlegungen können Sie das fast schon ohne Hilfe, Sie müssten aber noch wissen, wie man eine Schaltuhr programmiert. Dazu gibt es die Zeitfunktion htime.

Schalter

htime(Stunde, Minute, Sekunde)

Die Funktion liefert logisch *EIN* (1b01), wenn die Tageszeit **exakt** erreicht ist. Im anderen Fall ist sie *AUS* (0b01). Mit Hilfe einer if-Anweisung kann nun zu beliebigen Zeitpunkten umgeschaltet werden.

Unser Anwenderprogramm lautet damit:

Schaltzeituhr

Bewegungsmelder

```
[ETS-ESF]
// Die aus der ETS exportierte ESF-Datei
EibPC/Haus.esf
[EibPC]
if (systemstart()) then                                          \\
                        BewegungsMelder=AUS;       \\
                        read("Schalter-1/0/0") ;           \\
                        write("Lampe-1/1/1",AUS);       \\
                        write("Dimmer-1/1/2"u08,0%)   \\
                endif
// Variablen
Schalter="Schalter-1/0/0"
BewegungsMelder="Bewegungsmelder-1/2/0"
Dimmer=100%
// Der Schalter
if (Schalter==EIN)  then                              \\
        write("Lampe-1/1/1",EIN);            \\
        write("Dimmer-1/1/5",EIN);           \\
        write("DimmerWert-1/1/2",100) \\
        endif
if (Schalter==AUS) then                           \\
        write("Lampe-1/1/1",AUS);       \\
        write("Dimmer-1/1/2"u08,0%)   \\
        endif
//Bewegungsmelder
if (htime(20,00,00)) then Dimmer=50% endif
if (htime(23,00,00)) then Dimmer=30% endif
if (htime(03,00,00)) then Dimmer=10% endif
if (htime(07,30,00)) then Dimmer=100% endif

if  (BewegungsMelder==EIN) and (Schalter==AUS) then write("Dimmer-1/1/5",EIN); write("DimmerWert-1/1/2",Dimmer) endif
if  (BewegungsMelder==AUS) and (Schalter==AUS) then write("Dimmer-1/1/5",AUS) endif
```

Durch die Zeitfunktion htime lassen sich Zeitsteuerungen schnell und einfach realisieren. Weitere Zeitfunktionen finden Sie auf Seite 193 für Minuten- , Sekunden- , Tages-, Monats und Jahreszeitschaltuhren.

Nicht nur eine Zeitschaltuhr

Sie können quasi beliebig viele (65.000) Schaltzeituhren nutzen, indem Sie die Funktion entsprechend oft anwenden. Diese sind dabei weder miteinander verbunden, noch begrenzen sich gegenseitig. Lesen Sie dazu mehr auf Seite 195 ff.

Beispielsweise können Sie mit:

```
if (stime(43)) then
```

jede Minute, wenn exakt 43 Sekunden erreicht sind, eine Anweisung ausführen.

Wichtige Sonderfunktionen für zeitgesteuerte Ausgaben sind außerdem

- delay - Funktion,
- after - Funktion und
- cycle - Funktion.

Weiteres finden Sie im nächsten Schritt dieser Einführung und auf Seite 202.

Zeitschaltuhr und Vergleichszeitschaltuhren

Beim Einschalten steht die Variable Dimmer auf 100%. Wenn der Bewegungsmelder in diesem Fall anspricht, wird der Dimmaktor diesen Wert annehmen. Es könnte aber sein, das die Inbetriebnahme (Überspielen des Anwenderprogramms) z.B. um 20:00:01 erfolgt. Da htime() einen exakten Zeitvergleich durchführt und somit nur dann die entsprechende if-Anweisung ausgeführt wird, wenn die angegebene Uhrzeit erreicht ist, wird Dimmer nicht geändert. Fragen Sie mit Hilfe des Debuggers (Abbildung 32) den Wert der Variablen Dimmer ab. Sie würde bei unseren eben gemachten Annahmen noch auf 100% stehen.

Dennoch könnten Sie dieses Problem lösen, indem Sie bei der Anweisung if systemstart()... entsprechende Angaben machen, oder Sie nutzen die Vergleichszeitschaltuhren. Lesen Sie dazu mehr auf Seite 195 ff.

Systemuhr einstellen

Wenn Sie von den Schaltzeituhren Gebrauch machen wollen, müssen Sie den Enertex® EibPC mit der Uhrzeit bekannt machen. Am einfachsten geht dies, wenn der Enertex® EibPC über Ihren Router eine Internetverbindung aufbauen kann. Alternativ synchronisiert das KNXTM Gerät mit Funkempfänger o.ä. regelmäßig die Systemzeit auf dem Bus. Weiteres hierzu finden Sie auf 156.

Ein Freigabe-Schalter und das Validierungsschema

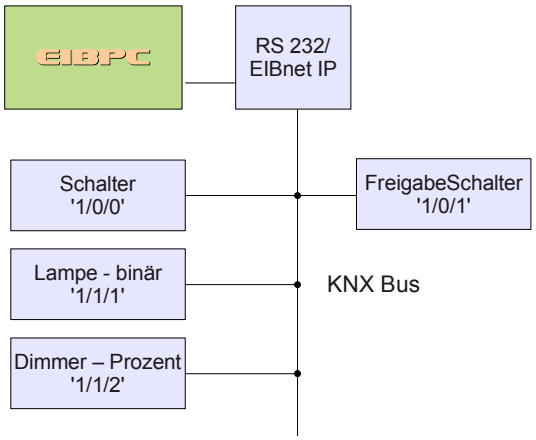


Abbildung 34: Blockschaltbild Schalter und FreigabeSchalter

Wir haben einen Aufbau wie in Abbildung 34 dargestellt.

Wird der Schalter „EIN“ gedrückt, soll die Lampe einschalten und der Dimmer auf 80% gehen. Wenn er auf „AUS“ geht, sollen die Lichter ausgehen. Wenn der Freigabe-Schalter (Gruppenadresse 1/0/1) auf „AUS“ steht, soll der Schalter „EIN“ ignoriert werden.

Eine einfache Aufgabe: Mit ihrem bereits erlernten Wissen finden Sie schnell die richtige Lösung:

Richtig gelöst – so wird's gemacht

```
[ETS-ESF]
EibPC/Haus.esf
[EibPC]
if (systemstart()) then read("Schalter-1/0/0"); read("FreigabeSchalter-1/0/1") endif
if ("Schalter-1/0/0"==EIN) and ("FreigabeSchalter-1/0/1"==EIN) then      \\
                                    write("Lampe-1/1/1",EIN);       \\
                                    write("Dimmer-1/1/2"u08,80%)  \\
                                else
                                    write("Lampe-1/1/1",AUS);       \\
                                    write("Dimmer-1/1/2"u08,0%)   \\
                                endif
```

Geht ausgezeichnet.

if-Anweisungen verschachteln

if-Anweisungen können verschachtelt werden. Sie könnten daher der Meinung sein, dass Sie wie folgt vorgehen:

Ist zwar gültige Syntax (keine Fehler durch den Compiler) ...

... macht aber was anderes, als gewollt

```
[ETS-ESF]
/EibPc/Haus.esf
[EibPC]
if (systemstart()) then read("Schalter-1/0/0"); read("FreigabeSchalter-1/0/1") endif
if  ("FreigabeSchalter-1/0/1"==EIN)  then                \\
            if ("Schalter-1/0/0"==EIN) then              \\
                    write("Lampe-1/1/1",EIN);       \\
                    write("Dimmer-1/1/2"u08,80%)  \\
                else                                     \\
                    write("Lampe-1/1/1",AUS);       \\
                    write("Dimmer-1/1/2"u08,0%)   \\
                endif                                    \\
        endif
```

Der Enertex® EibPC ist ein Zustandsspeicher. Alle Telegramme werden intern gespeichert und so ein Prozessabbild erzeugt.

Dieses Programm ist an sich nicht fehlerhaft, aber es wird anders funktionieren, als Sie sich ggf. gedacht haben. Der Enertex® EibPC verarbeitet nur Ausdrücke, welche sich ändern und merkt sich daher den letzten Zustand, der über eine Gruppenadresse gesendet wurde. Wenn auf die Gruppenadresse "Freigabeschalter-1/0/1" ein EIN gesendet wurde, nachdem ein AUS vorlag, wird die innere if-Anweisung genau einmal ausgewertet. Also wird in diesem Fall die Lampe bzw. der Dimmer ein- oder ausgeschaltet, je nachdem welcher Wert über die Gruppenadresse "Schalter-1/0/0" zuletzt geschickt wurde. Wenn sich nun - "Freigabeschalter-1/0/1" ist immer noch EIN - die Gruppenadresse "Schalter-1/0/0" ändert, wird kein Telegramm mehr ausgelöst, da die Bedingung der äußeren if-Anweisung sich nicht ändert (vergleichen Sie auch die Anmerkungen auf Seite 46).

Das Abbilden der Zustände der eingegangen Telegramme bezeichnen wir als Validierungsschema. Es macht die Anwendung und Programmierung einfach und intuitiv, solange man Verschachtelungen von if-Anweisungen vermeidet. Wie im Beispiel gezeigt, ist diese Vermeidung immer auf einfache Weise möglich.

Wir empfehlen Anfängern die Vermeidung von verschachtelten if-Anweisungen.

Ein Treppenhauslicht

Beispiel 5: Ein Treppenhauslicht

Abbildung 35 sei die Installation, für die ein Treppenhauslicht (3 min Schaltzeit) realisiert werden soll. Der Schalter liefert dabei immer nur einen EIN-Impuls (es wird also nicht mehr ausgeschaltet).

Sie möchten zudem auf einem KNX™-Anzeigeelement anzeigen, wie oft der Schalter gedrückt wurde. Dazu haben Sie die Gruppenadresse 1/2/2 in der ETS für das Anzeigeelement als Zeichenkette mit max. 14 Zeichen definiert.

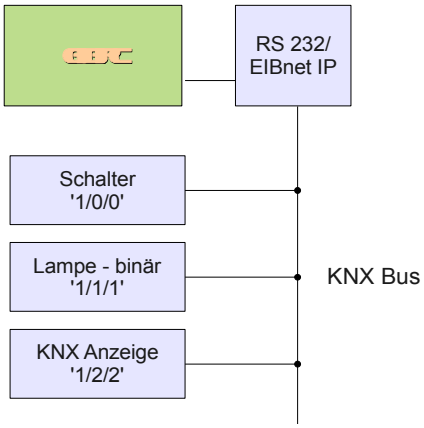


Abbildung 35: Treppenhauslicht

Beim Systemstart soll das Licht ausgehen. Der Schalter liefert abwechselnd EIN und AUS Telegramme. Nach Schalterdruck („Schalterstellung“ sei beliebig) soll die Lampe angehen und automatisch nach 3 Minuten wieder ausgehen. Die bereits erfolgten Einschaltvorgänge sollen am KNX™-Anzeigeelement dargestellt werden.

Zwei Varianten: Nachtriggerbar oder nicht?

Variante 1: Bei erneutem Schalterdruck während der 3 Minuten Einschaltzeit soll der Timer nicht erneut starten

Variante 2: Bei erneutem Schalterdruck während der 3 Minuten Einschaltzeit soll der Timer erneut starten

Variante 1: Nicht Nachtriggern

Wir geben zunächst die Realisierung für ***Variante 1*** an. Diesmal ohne die Verwendung von importierten Gruppenadressen.

In dieser Realisierung soll der Timer nicht erneut gestartet werden, wenn dieser bereits läuft (kein „Nachtriggern“).

Zunächst das Programm:

Definition einer KNX™ Zeichenkette
$ Text$c14

```
[EibPC]
if systemstart() then write('1/1/1'b01,AUS) endif
SchaltVorgang=AUS
Zaehler=0u32
LeerString=$ Noch nichts! $c14
if event('1/0/0'b01) then  \\
                              SchaltVorgang=EIN;  \\
                              write('1/1/1'b01,EIN);  \\
                              Zaehler=Zaehler+1u32\\
                           endif
if (after(event('1/0/0'b01), 180000u64)) then                              \\
                                    write('1/1/1'b01,AUS);                 \\
                                    SchaltVorgang=AUS;                     \\
                                    write('1/2/2'c14,convert(Zaehler,LeerString)) \\
                            endif
```

Neu
event(Gruppenadresse)
after(Ausdruck, Zeit in ms)
convert(Quelle, Ziel)

Zur Erinnerung:

event(Gruppenadresse)

zeigt an, wenn auf dem Bus eine Nachricht mit der angegeben Gruppenadresse eingetroffen ist. Sie überprüft nicht, ob sich die Nachricht ändert, welchen Inhalt sie hat oder welchen Typ. Sobald eine Nachricht eintrifft, geht sie für einen Verarbeitungsszyklus des Enertex® EibPC auf EIN. Damit wird die Bedingung der if-Anweisung wahr und die Anweisung ausgeführt.

Sie haben bereits alle Sonderfunktionen, die mit der Kommunikation auf dem KNX Bus zu tun haben, kennen gelernt. Nur mit Hilfe dieser Funktionen können Sie auf den Bus Einfluss nehmen.

Alle Möglichkeiten mit dem Bus zu kommunizieren im Überblick.

Zur Wiederholung:

- *write(GruppenAdresse, Wert) schreibt einen Wert auf die Gruppenadresse am Bus*
- *read(GruppenAdresse) schreibt eine Leseanforderung auf die Gruppenadresse am Bus*
- *event(GruppenAdresse) überprüft, ob eine Nachricht auf der Gruppenadresse am Bus gesendet wurde*
- *Variable=Gruppenadresse weist einer Variablen den Wert, welcher mit einem Telegramm auf die angegebene Gruppenadresse gesendet wird, zu.*

Datentyp u64
64 Bit, ohne Vorzeichen, ganze Zahl

Wir nutzen außerdem die after-Funktion. Die Verzögerungsfunktion after, die wir auch Präzisionstimer nennen, arbeitet mit einer Auflösung von 1 ms:

after(Variable {Typ b01}, Zeit in ms {Typ u64})

Der Präzsionstimer erwartet als erstes Argument eine Variable oder einen Ausdruck vom binären Typ (Datentyp b01, zu Datentypen siehe auch Seite 38 und 158 ff). Die Funktion after verzögert den Eingang (EIN) um die Zeit, die im zweiten Argument angeben wird. Der Rückgabewert ist also EIN – Impuls. Grafisch lässt sich das mit Abbildung 36 recht anschaulich darstellen. Das zweite Argument ist vom Typ ganzzahlig, vorzeichenlos mit 64 Bit. Wir benötigen daher den Datentyp u64. Dieser Wert gibt die Verzögerungszeit in ms an.

Die Verzögerungsfunktion

Sie können damit Verzögerungen über Jahrzehnte hinweg einstellen. Wenn die Funktion after einmal gestartet wird, verarbeitet Sie nur einen Impuls ihres Eingangs. Dadurch entsteht die Abbildung 36 gezeigte Totzeit, die gleich der Verzögerungszeit ist. Im Beispiel nutzen wir eine Verzögerung von

180.000ms = 3*60*1000ms = 3*60s = 3min.

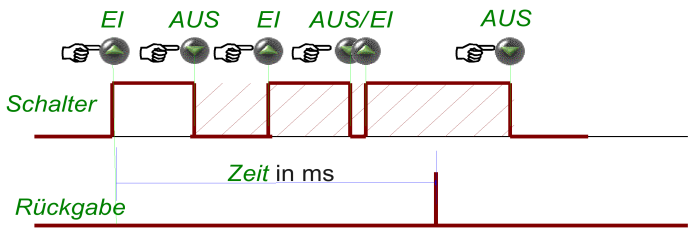


Abbildung 36: After-Funktion

Die Funktion after ist nicht nachtriggerbar, wie in Abbildung 36 durch die „Totzeit“ dargestellt wird. In unserem Fall (Variante 1) ist das ja so gewünscht. Das heißt, wenn after einmal gestartet wurde, werden sämtliche weitere Änderungen des Eingangs ignoriert (siehe Schraffur in der Abbildung).

Argumente von Funktionen sind beliebige Ausdrücke, Konstanten oder Variablen

In unserem Beispiel schreiben wir aber:

```
if (after(event('1/0/0'b01), 180000u64)) then
```

Das Argument ist in diesem Fall keine Variable, sondern eine Rückgabe. Denkbar wäre auch ein Vergleich, wie Sie bereits auf Seite 38 gelernt haben, ist der Rückgabewert eines Vergleichs ein binärer Datentyp.

Grundsätzlich kann also anstelle einer Variablen auch eine beliebige andere Funktion stehen („Verschachteln“), wichtig ist lediglich, dass diese Funktion oder der Ausdruck den passenden Datentyp als Rückgabewert aufweist. Im obigen Beispiel ist bereits SchaltVorgang ein binärer Datentyp, daher ist der Vergleich nicht notwendig. Sie können aber nun after() in diesem Sinne nutzen, um Schwellwerte (z.B. if after(Licht>200u32,ZeitWert)) zu verzögern. Die Verzögerungszeit muss nicht konstant sein. Sie könnte z.B. über ein Telegramm oder eine Variable eingegeben werden.

So oft wie sie wollen

An dieser Stelle noch ein weiterer Hinweis: Die Anzahl der im Programm verwendeten Präzisionstimer ist nicht begrenzt. Sie können jederzeit die Funktion mehrfach benutzen. Für jede aufgerufene after-Funktion wird ein neues Objekt angelegt. Die after-Funktionen begrenzen sich daher nicht gegenseitig in der Verwendung. Dies gilt immer und auch für alle weiteren Funktionen der Programmierung mit Enertex® EibStudio.

Zyklus gesteuert

Eine wichtige Sonderfunktion für zeitgesteuerte Ausgaben ist die cycle - Funktion. Mit Hilfe von diesen Funktionen können in einem bestimmten Zeitzyklus Aufgaben abgearbeitet werden. Weiteres finden Sie hierzu auf Seite 202.

Es fehlt noch die Konvertierungsfunktion:

convert(Variable1 {Typ Quelle}, Variable2 {Typ Ziel})

Diese hat als Rückgabewert den einen Wert, der in Variable1 gespeichert wird, aber in den Typ der in Variable2 gewandelt wird. Der Wert an sich der Variable2 wird ignoriert, es wird nur gebraucht, um den neuen Datentyp zu ermitteln. Mit convert können Sie beispielsweise einen binären Wert in eine 8 Bit vorzeichenlose Zahl umwandeln:

```
a='1/2/2'b01
b=convert(a,1u08)
```

Im Beispiel erhöhen wir die Variable Zaehler im then-Zweig bei jedem Schaltzyklus der Lampe

```
Zaehler=Zaehler+1u32
```

Datentyp u32
32Bit, ohne Vorzeichen, ganze Zahl

Dabei ist Zaehler vom Typ ganzzahlig, vorzeichenlos und 32 Bit. Sie können damit bis 2^{32}-1=4294967295 zählen. Sollte das Licht tatsächlich so oft betätigt worden sein, würde die Variable beim erneuten Schalterbetätigen einen Überlauf bekommen und wieder bei 0 anfangen zu zählen.

KNX-Zeichenketten

Zeichenketten aus bis zu 14 Buchstaben werden über Gruppenadressen vom Datentyp c14 übertragen. Die Definition einer c14-Zeichenkette erfolgt eingerahmt von zwei $-Zeichen

$ Text $c14 entspricht Zeichenkette „ Text “ (Mit Leerzeichen)

Diesen Datentyp können Sie direkt auf KNX™ Anzeigeelemente ausgeben. Daneben gibt es bei der Programmierung des Enertex® EibPC auch Zeichenketten mit einer Länge von bis zu 1400 Zeichen. Diesen werden wir für die Verarbeitung von LAN Telegrammen weiter hinten in dieser Einführung noch benötigen.

Mit

```
LeerString=$ Noch nichts! $c14
```

wird eine Zeichenkettenvariable LeerString mit Inhalt „ Noch nichts! “ (inklusive Leerzeichen) definiert.

Damit wird in der Anweisung

```
write('1/2/2'c14,convert(Zaehler,LeerString))
```

schließlich der Zaehler als eine Zeichenkette gewandelt auf ein KNX™-Anzeigeelement gegeben.

LeerString wird für die convert-Funktion nicht ausgewertet. Es dient lediglich dazu, den Datentyp für die Konvertierung zu definieren. Daher wäre an dieser Stelle auch write('1/2/2'c14,convert(Zaehler,$ $)) eine denkbare Lösung.

Eine Besonderheit im Programm mag Ihnen noch unverständlich erscheinen:

Wie in jeder anderen Programmiersprache: Zaehler=Zaehler+1

```
if event('1/0/0'b01) and (SchaltVorgang==AUS) then  \\
                                    SchaltVorgang=EIN;  \\
                                    write('1/1/1'b01,EIN);  \\
                                    Zaehler=Zaehler+1u32\\
                                endif
```

Warum ist hier die Event-Funktion in der if-Anweisung mit der Variablen SchaltVorgang verknüpft? Wäre dies nicht der Fall, würde bei jedem event('1/0/0'b01), d.h. Schalterdruck, ein Telegramm auf den Bus geschrieben und der Zaehler erhöht. In unserer Aufgabenstellung wollen wir jedoch nicht die Anzahl der Schaltbetätigungen sondern die Anzahl der Zyklen mit eingeschalteter Lampe ermitteln.

Daher haben wir in der if-Anweisung die event-Funktion „verriegelt".

Zur Ergänzung: Astrofunktion Tag oder Nacht?

Eine weitere nützliche Funktion ist die sun-Funktion: Sie hat den Rückgabewert EIN (1b01), wenn die Sonne auf - bzw. AUS (0b01) wenn sie untergegangen ist (siehe auch Seiten 157 und 190).

Variante 2: Nochmal Drücken

Nun zur ***Variante 2***. Es soll der Timer für die Lichtschaltung immer wieder neu gestartet werden, wenn erneut der Lichtschalter betätigt wird.

Hierfür benötigen wir

delay(Signal {Typ b01}, Zeit in ms {Typ u64})

delay erwartet als erstes Argument eine Variable oder einen Ausdruck vom binären Typ (Datentyp b01, zu Datentypen siehe auch Seite 38 und 158 ff). Die Funktion delay arbeitet wie in Abbildung 37 und gibt den Wechsel des Signals von AUS auf EIN verzögert als Impuls wieder angeben. delay wird bei erneutem Aufruf den Timer neu starten („Re-Trigger"):

Der Rückgabewert ist also EIN oder AUS. Das zweite Argument ist vom Typ ganzzahlig, vorzeichenlos mit 64 Bit, wir benötigen damit den Datentyp u64.

Eine EIN-Flanke startet den delay Timer.

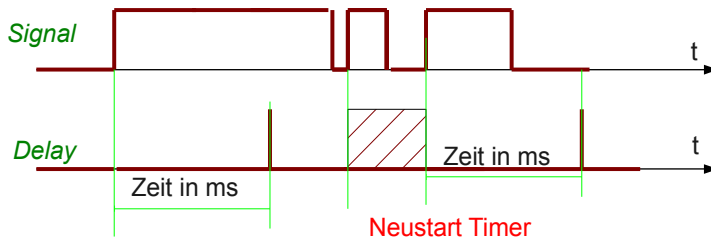


Abbildung 37: delay-Funktion

Damit ist unser Programm nur an einer Stelle zu ändern und wir müssen lediglich after durch delay ersetzten.

```
[EibPC]
if systemstart() then write('1/1/1'b01,AUS) endif
SchaltVorgang=AUS
Zaehler=0u32
LeerString=$ Noch nichts! $c14
if event('1/0/0'b01) and (SchaltVorgang==AUS) then  \\
                                    SchaltVorgang=EIN;  \\
                                    write('1/1/1'b01,EIN);  \\
                                    Zaehler=Zaehler+1u32\\
                                endif
if (delay(event('1/0/0'b01), 180000u64)) then                                   \\
                                        write('1/1/1'b01,AUS);                  \\
                                        SchaltVorgang=AUS;                      \\
                                        write('1/2/2'c14,convert(Zaehler,LeerString)) \\
                                 endif
```

Variante 2:
Ersetze after durch delay

An dieser Stelle noch einmal der Hinweis: Die Anzahl der im Programm verwendeten delay-Timer ist nicht begrenzt. Sie können diese Funktion mehrfach benutzen. Für jede aufgerufene delay-Funktion wird ein neues Objekt angelegt. Die delay-Funktionen begrenzen sich daher nicht gegenseitig in der Verwendung. Dies gilt immer und auch für alle weiteren Funktionen der Programmierung mit Enertex® EibStudio.

Alternativ kann der Code vereinfacht werden und ohne die Variable *SchaltVorgang* auskommen. Es bleibt dem Leser überlassen, dies nachzuvollziehen.

```
[EibPC]
if systemstart() then write('1/1/1'b01,AUS) endif
Zaehler=0u32
LeerString=$ Noch nichts! $c14
if event('1/0/0'b01) and ('1/1/1'b01==AUS) then {
                write('1/1/1'b01,EIN);
                Zaehler=Zaehler+1u32
} endif
if (delay(event('1/0/0'b01), 180000u64)) then {
               write('1/1/1'b01,AUS);
              write('1/2/2'c14,convert(Zaehler,LeerString))
} endif
```

Datumsteuerung

Angenommen, Sie möchten mit einem KNX™ Anzeigelement, das über die Gruppenadresse 1/2/3 eine KNX™ Zeichenkette erwartet, einen Geburtstagskalender realisieren:

Geburtstage anzeigen

Wenn ein Geburtstag vorliegt, soll dies am Anzeigeelement angezeigt werden.

In allen anderen Fällen soll die Zeit angezeigt werden, wie lange der EibPC online ist.

Wir nutzen das Makro „Online“

Um diese Aufgabe zu stemmen, nutzen wir das Makro Online() aus der Bibliothek Enertex.lib. Wir müssen die Bibliothek in unser Programm einbinden. Klicken Sie dazu auf Programm / Makro-bibliotheken und dort auf den Menüpunkt Hinzufügen. Wählen Sie die Bibliothek „Enertex.lib“ aus und klicken sie auf Hinzufügen.

Im zweiten Schritt wählen Sie in Programm / Makro das Makro Online(). Es erscheint ein Parameterdialog, den Sie einfach mit OK bestätigen.

Das Makro Online() definiert die Variable „AnzeigeOnline“, eine KNX™ Zeichenkette (Datentyp c14), die die Zeit vom letzten Start des Anwenderprogramms in eine Zeichenkette schreibt. Das Format ist ddd.hh:mm (ddd: Tage von 0 bis 999, hh: Stunden, mm: Minuten).

Hier nun das Programm:

Makros Einbinden

```
[Macros]
//Makros
Online()

[MacroLibs]
//Makro-Bibliotheken
Enertex.lib

[EibPC]
Geburtstag=AUS
AnzeigeGeb=$            $c14
// Hier beliebige Geburtstage eintragen
if month(10,01) then AnzeigeGeb=$    Alex      $c14;Geburtstag=EIN endif
if month(15,01) then AnzeigeGeb=$    Martin    $c14;Geburtstag=EIN endif
if month(21,02) then AnzeigeGeb=$    Jürgen    $c14;Geburtstag=EIN endif
// Rücksetzen kurz vom Umschalten
if htime(23,59,59) then Geburtstag=AUS endif
```

! bedeutet Invertierung

```
//Hier nun die Ausgabe der Zeit
if change(AnzeigeOnline) and !Geburtstag then write("FlurText-0/2/1"c14,AnzeigeOnline) endif
if systemstart() then write("FlurText-0/2/1"c14,AnzeigeOnline) endif

// Geburtstagsanzeige soll blinken
AnzeigeBlink=0
if Geburtstag and cycle(0,3)      then write("FlurText-0/2/1"c14,AnzeigeGeb);AnzeigeBlink=1 endif
if after(AnzeigeBlink==1,1700u64) then write("FlurText-0/2/1"c14,$Geb.-tag$c14);AnzeigeBlink=2 endif
if after(AnzeigeBlink==2,700u64)  then write("FlurText-0/2/1"c14,$    !!+++!!    $c14);AnzeigeBlink=0 endif
```

Wir haben die Funktion month genutzt:

month(*Tag*,*Monat*)

Die Funktion liefert logisch EIN (1b01), wenn das Datum eines beliebigen Jahres erreicht ist. Im anderen Fall ist sie AUS (0b01).Der Schaltpunkt ist dabei exakt 00:00:00 Uhr.

Jeden Tag um 23:59:59 (Anweisung mit htime) wird der *Geburtstag* wieder auf AUS geschaltet.

Sommer per Datumsdefinition

Sie können mit month auch Zeiträume definieren: Die folgenden Variable Sommer ist im Zeitraum vom 1. Mai bis 30. September auf EIN, sonst auf AUS.

```
Sommer=month(01,05) and !month(30,09)
```

Damit können Sie den Sommer als Abfragebedingung realisieren:

```
if Sommer then ....
```

Beschattung

Wir greifen das Beispiel von Seite 31 nochmals auf.

Nun möchten wir die Automatisierung dahingehend ändern, dass die Beschattung über das Freigabeobjekt des Schalters erfolgt, aber nur wenn es draußen um 9:00 Uhr morgens bereits heller als 5000 Lux ist. Abends sollen alle Rollos auf die Nachtstellung verfahren (Rollos unten).

Ihr Aufstellungsort sei Nürnberg (49°27', 11° 5'). Sie können diese Daten mit Hilfe des Menüs Optionen – Koordinaten für die Sonnenstandsberechnung einstellen verändern (vgl. Seite 157).

Und hier gleich das Ergebnis der Programmierung. Sie sollten bereits alle Elemente der Programmierung kennen. Das Beispiel soll lediglich die „Vermischung“ bei der Anwendung von Makros und einer Programmierung verdeutlichen.

Variablen können auch Argumente der Makros sein.

```
[Location]
// Länge und Breite des Aufstellungsorts
11.083333333333334
49.45

[Macros]
//Makros
BeschattungRolloOstZeit(SchattenFreigabe,"Fenster-Ost-2/5/1","Fenster-Ost-2/5/1",5000)
BeschattungRolloSuedZeit(SchattenFreigabe,"FensterSüd-2/5/2","FensterSüdStop-2/5/5",4500)
BeschattungRolloWestZeit(SchattenFreigabe,"FensterWest-2/5/3","FensterWestStop-2/5/6",4000)

[MacroLibs]
//Makro-Bibliotheken
EnertexBeschattung.lib

[ETS-ESF]
// Die aus der ETS exportierte ESF-Datei
Beschattung.esf

[EibPC]
SchattenFreigabe=AUS
if ("FreigabeBeschattung-2/5/0"==EIN) and ("Licht-2/1/1">5000f16) and chtime(9,00,00) then \\
        SchattenFreigabe=EIN endif

// Wenn die Sonne untergeht...
if sun()==AUS then \\
write("Fenster-Ost-2/5/1",EIN); write("FensterSüd-2/5/2",EIN); write("FensterWest-2/5/3",EIN) endif
```

Ob Argumente von Makros Variablen oder Gruppenadressen oder beides sein können, hängt von der Implementierung des Makros ab. Bei den vorgegebenen Makrobibliotheken wird dies immer im Makroassistenten mit angegeben.

Taupunkttemperaturberechnung

Im folgenden Beispiel geht es weniger um Automatisierungsaufgaben. Wir wollen Ihnen an dieser Stelle lediglich zeigen, wie flexibel Sie mit Variablen rechnen können.

Ein Taschenrechner

In einem Wintergarten soll die Taupunkttemperatur (TD) abhängig von der Luftfeuchte (r) und der Raumtemperatur(T) errechnet werden.

Hierzu die notwendigen Berechnungen:

```
Bezeichnungen:
r = relative Luftfeuchte
T = Temperatur in °C
TD = Taupunkttemperatur in °C
DD = Dampfdruck in hPa
SDD = Sättigungsdampfdruck in hPa
Parameter:
a = 7.5, b = 237.3 für Temperatur >= 0
Formeln:
TD(r,T) = b*v / (a-v) mit
v(r,T) = log10(DD(r,T)/ 6.1078
DD(r,T) = r/100 * SDD(T)
SDD(T) = 6.1078 * 10^((a*T) / (b+T))
```

Die Umsetzung lautet dann:

Datentyp f32:
32 Bit Fließkommazahl

log(a,b): $Log_a(b)$
pow(x,y): x^y

```
a=7.5f32
b= 237.3f32
r="SensorLuftfeuchte-1/0/2"
T="SensorTemperatur-1/0/3"
SDDT=6.1078f32 * pow(10f32,(a*T)/(b+T))
DDrT= r/100f32*SDDT
v=log(10f32,(DDrT/6.1078f32))
TD=b*v/(a-v)
```

Sie können Ihren Enertex® EibPC nutzen wie einen Taschenrechner - weitere mathematische Funktionen finden Sie auf Seite 208.

Überwachen des Busverkehrs (Monitor)

Zwischenspeicher 500.000 Telegramme

Der EibPC speichert bis zu 500.000 Telegramme in einem Ringspeicher. Diese können abgeholt, überwacht und in einer Datei gespeichert werden, um diese z.B. mit Microsoft® Excel oder OpenOffice Calc Daten auszuwerten.

Der PC muss beim Aufzeichnen **nicht** mit dem Enertex® EibPC verbunden sein.

Telegramme abspeichern

Telegramme abholen

Um die im Enertex® EibPC aufgelaufenen Telegramme abzuholen, klicken Sie im Menü EibPC auf EIBTelegramme Abholen. In der Statuszeile links unten wird Ihnen der Fortschritt beim Abholen der Telegramme angezeigt. Die Geschwindigkeit, mit der die Telegramme übertragen werden, hängt in erster Linie von der Performance ihres PCs ab.

CSV: Durch Kommas getrennte Textdatei mit den Daten Ihrer Buskommunikation

Wenn alle Telegramme abgeholt sind bzw. Sie die Übertragung unterbrechen, erscheint ein Dialog, mit dem Sie diese Telegramme in eine CSV-Datei abspeichern können. Diese CSV Datei stellt eine Textdatei dar, in der die Daten in Tabellenform abgelegt werden. Dabei sind die einzelnen Daten durch Kommas getrennt. Sie können die Daten dann in Microsoft® Excel oder OpenOffice Calc oder einem Tabellenverarbeitungsprogramm Ihrer Wahl importieren.

Abbildung 38: Import der Aufzeichnungsdaten in OpenOffice

Das „Loggen" des Bustransfers erleichert die Diagnose von Fehlern und weitreichende Optimierungen

In Microsoft® Excel wählen Sie im Dialog Daten den Unterpunkt Importieren. In den dann folgenden Dialogen wählen Sie das Format am besten als „Text" und die Textkennzeichnung als {keine}. Gleichermaßen gehen Sie bei OpenOffice Calc vor.

Speichern in Datei

Sie können zusätzlich auch die momentane Buskommunikation sichtbar machen. Dazu aktivieren Sie die Option "EIB-Telegramme abholen / Verbindungsstatus" oder klicken Sie auf die Schaltfläche  . Wenn der Enertex® EibPC mit dem Enertex® EibStudio verbunden ist, wechselt die Schaltfläche zu  .

Online-Telegramme

Im Fenster "Meldungen" werden nun alle Telegramme ausgegeben, die gerade auf den Bus geschrieben wurden. Abbildung 39 zeigt einen typischen Auszug aus dem Meldungsfenster, der an sich selbsterklärend ist.

Meldungen

```
% 2034-01-01 16:40:47 | Sender: EibPC | GA: "Akt1-0/0/5" | Wert: EIN | Typ: Binärwert |  Schreiben
% 2034-01-01 16:40:48 | Sender: 1.1.7 | GA: "Heizaktor-1/0/5" | Wert: 0 | Typ: positive Ganzzahl |  Schreiben
% 2034-01-01 16:40:48 | Sender: 1.1.7 | GA: "HeizungAn-1/0/6" | Wert: AUS | Typ: Binärwert |  Schreiben
% 2034-01-01 16:40:52 | Sender: EibPC | GA: "Akt1-0/0/5" | Wert: EIN | Typ: Binärwert |  Schreiben
% 2034-01-01 16:40:57 | Sender: EibPC | GA: "Akt1-0/0/5" | Wert: EIN | Typ: Binärwert |  Schreiben
% 2034-01-01 16:41:02 | Sender: EibPC | GA: "Akt1-0/0/5" | Wert: EIN | Typ: Binärwert |  Schreiben
% 2034-01-01 16:41:07 | Sender: EibPC | GA: "Akt1-0/0/5" | Wert: EIN | Typ: Binärwert |  Schreiben
```

Abbildung 39: EIB-Telegramme abrufen

„Trockenübungen"

Da die RS232-Schnittstelle keine Rückmeldung über den Verbindungsstatus liefert, schickt der EibPC alle Telegramme raus, auch wenn keine Busverbindung aufgebaut ist. Dadurch lässt sich auch ohne Busverbindung jeder Code austesten.

Die IP Schnittstelle hat dagegen ein definiertes Handshake. Der EibPC wartet dann auf die Rückmeldung, bevor er Telegramme verschickt.

Telegramme filtern

Sie können Filter einrichten, so dass nur Telegramme von bestimmten Gruppenadressen oder von bestimmten Geräteadressen (Absendern) angezeigt oder gespeichert werden. Dazu müssen Sie zunächst den Filter einrichten (siehe Abbildung 40), was Sie über den Menüpunkt EibPC – Filter einrichten einstellen können.

Abbildung 40: EIB-Telegramme filtern

Wildcards „?“ : Beliebige Ziffer, „“ Beliebige Zahl*

Mit den Einstellungen in Abbildung 40 werden nur Telegramme angezeigt, die vom Absender mit der physikalischen Adresse 1/1/2 kommen und an die Ziel- Gruppenadresse 0/1/5 geschickt werden.

Möchten Sie alle Telegramme eines bestimmen Absenders beobachten, unabhängig vom Ziel, müssen Sie die Option Ziel deaktivieren (siehe Abbildung 41).

.

Abbildung 41: alle EIB-Telegramme eines Absenders filtern

So sehen Sie, ob der Filter aktiviert wurde.

Wenn der Filter aktiviert wird (EibPC – aktuelle EIBTelegramme Filtern),wechselt die Statusanzeige wie in Abbildung 42 andeutet. Der Filter wirkt sowohl auf Telegramme im Meldungsfenster, wie auch beim Abspeichern in eine CSV-Datei.

Abbildung 42: Filteranzeige rechts unten in der Statusanzeige, rechtes Bild: Filter aktiviert

Telegramme zyklisch Speichern

Dabei haben Sie die Möglichkeit, Wildcards zu benutzen: Schreibt man für das Ziel, letztes Eingabefeld „1/1/2**?**3“, so werden alle Gruppenadresse 1/1/2**0**3, 1/1/2**1**3 ... 1/1/2**9**3 herausgefiltert. Sie können beliebig viele Wildcards benutzen.

Telegramme auf FTP Server

Sie können das Enertex® EibStudio „von sich aus“ regelmäßig Telegramme abholen und auf ihrem PC abspeichern lassen. Hierzu steht der „Autolog“-Dialog zur Verfügung. Auch hier sortiert ein aktiver Filter Telegramme aus. Siehe S. 148.

Alternativ können Sie den Enertex® EibPC „von sich aus“ regelmäßig Telegramme auf einen beliebigen FTP-Server speichern lassen. Weiteres finden Sie auf S. 150.

Szenen

Szenenaktor

Beispiel 6: Ein Szenenaktor

Ein Szenebaustein wird gewünscht

Abbildung 43 sei die Installation, für die ein Szenenaktor realisiert werden soll. Der Schalter ist in der ETS so parametriert, dass er als Szenenschalter funktioniert: Das Aufrufen der Szene wird beim einfachen Drücken initiiert und das Speichern einer eingestellten Szene mit langen Tastendruck. Die ETS-Programmierung ist so, dass der Schalter auf der Adresse 1/0/0 eine Szene mit Nummer 1 aufruft.

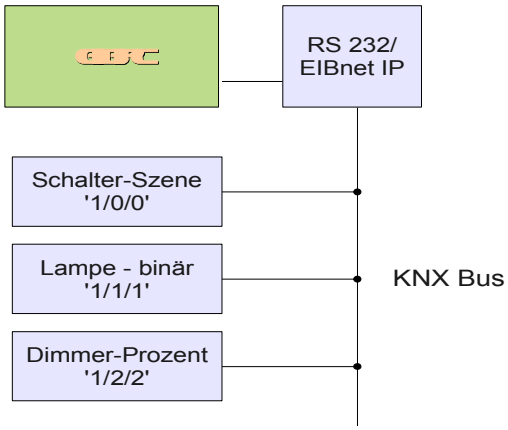


Abbildung 43: Szenen

Unsere Aufgabe ist demnach:

> Der Enertex® EibPC soll sich im KNX Netz wie ein Szenenaktor, der auf die Adresse 1/0/0 parametriert wurden, verhalten.

Eine Szene

Um einen Szenenaktor zu definieren, verwenden Sie die Funktion

scene(*GruppenadresseSzenebaustein*, *GruppenadresseAktor1, GruppenadresseAktor2, , ... GruppenadresseAktorN)*

Sie können bis zu 65.000 Aktoren an diesen einem Szenenaktor „anschließen“. Die Anzahl der Argumente, d.h. Aktoren die einem Szenenaktor bzw. Szenenfunktion zugeordnet sind, ist beliebig.

Der Szenenaktor kann 64 verschiedene Szenarien abspeichern (Werte 0 bis 63). Über die *GruppenadresseSzenebaustein (8 Bit, vorzeichenlos)* werden diese Szenarien-Nummern (0u08 bis 63u08) über Ihre geeigneten Aktoren, gespeichert und abgerufen. Die Aktoren und Schalter müssen Sie wie gewöhnlich in der ETS programmieren.

Die Realisierung

Mit der scene-Funktion ist diese Aufgabe wie folgt recht einfach gelöst:

```
[EibPC]
scene('1/0/0'u08, "Lampe-1/1/1“, "'Dimmer-1/2/2")
```

Mehr ist nicht zu tun. Nun ist im Enertex® EibPC ein Szenenaktor mit 64 ansprechbaren Szenen angelegt. Sie können dabei natürlich Szenen mit manueller Gruppenadresse ansprechen.

Stromausfall?

Szenen neu konfigurieren

Gespeicherte Szenen bleiben auch bei einer vorübergehenden Trennung des Enertex® EibPC von der Stromversorgung und Änderung des Anwenderprogramms (erneutes und verändertes Einspielen) erhalten. Wenn Sie die Szenen komplett vom Festspeicher des Enertex® EibPC löschen wollen, wählen Sie im Menu EibPC den entsprechenden Punkt. Wenn Sie im Anwenderprogramm den im Enertex® EibPC definierten Szenenaktoren veränderte Gruppenadressen vergeben (also im Beispiel "Szene-1/0/0", mit weiteren Gruppenadressen), sollten Sie dies in jedem Falle tun.

Sie können die scene-Funktion nahezu beliebig oft (max. 65000) einsetzen, z.B. für zwei Szenenaktoren:

```
[EibPC]
MyVar=34
scene('1/0/0'u08, "Lampe-1/1/1“, "'Dimmer-1/2/2",MyVar)
scene('2/0/0'u08, "Lampe-1/1/1“, '1/3/2'u08, '5/8/8'f32, '4/4/56's16)
```

Der Enertex® EibPC selbst kann aber auch Szenen ansprechen und deren Speicherung initiieren. Dazu gibt es die beiden Funktionen

Szenen speichern
Szenen aufrufen

storescene(*GruppenadresseSzenebaustein*, *SzenenNummer{u08}*)
callscene(*GruppenadresseSzenebaustein*, *SzenenNummer{u08}*)

Die erste Funktion weist einen Szenebaustein (KNX-Gerät oder scene-Funktion des Enertex® EibPC) an, die Szene mit der *SzenenNummer* zu speichern. Zweite Funktion initiiert das Auslesen, d.h. die Szene wird aufgerufen. Die Szenen werden über die Gruppenadresse angesprochen.

Anstelle von Gruppenadressen können Sie auch Variablen nutzen, wie im obigem Beispiel MyVar. Die Gruppenadresse des Szenenaktors kann sich, sowohl auf einen Szenenaktor im Enertex® EibPC (d.h. Scene-Funktion), als auch auf einen externen Szenebaustein beziehen.

Szenen vorbelegen

Mit Hilfe von presetscene kann eine Szene bei der Inbetriebnahme vorbelegt werden (s. S.225).

Szenen ohne Szenentaster

Zum Schluss noch ein kleines „Schmanckerl“:

Angenommen der Schalter Ihres KNX™-Netzes (Abbildung 43) kann nur Binärtelegramme liefern: Sie haben Ihr System so programmiert, dass das AUS Telegramm bereits alle Lichter ausschaltet. Wenn Sie nun auf EIN drücken, soll die Szene mit Nummer 1 aufgerufen werden, d.h. auf Gruppenadresse 1/0/0 wird nun nur noch ein Bit gesendet.

Aber Sie wollen noch mehr:

Drücken Sie innerhalb von 1 Sekunde ein zweites Mal EIN, soll die Szene gespeichert werden. Sie realisieren den „Doppelklick“ wie Sie ihn von Ihrem Umgang mit dem PC und Maus gewohnt sind.

Kein Problem:

Kurz für (siehe Seite 160)
Tastendruck=0u08
Szene=1u08

Nach einer Sekunde auswerten, ob ein oder zweimal geschaltet wird.

```
[EibPC]
// Der Szenenaktor
scene('1/2/3'u08, "Lampe-1/1/1", "Dimmer-1/2/2"u08)
// Variablen
Tastendruck=0
Szene =1

// Zählen des Tastendrucks
if (("Schalter-1/0/0") and event("Schalter-1/0/0")) then Tastendruck= Tastendruck+1 endif

// Rücksetzten des Zählers
if after(event("Schalter-1/0/0"),1000u64) then Tastendruck=0 endif

// Speichern oder Aufrufen
if after(event('1/0/0'b01),1000u64) and (Tastendruck ==1) then callscene('1/2/3'u08,Szene) endif
if after(event('1/0/0'b01),1000u64) and (Tastendruck ==2) then storescene('1/2/3'u08,Szene) endif
```

Damit nicht genug:

In der Bibliothek Enertex.lib finden Sie einen vorgefertigten Funktionsblock „Doppelklick()“ für die Doppelbelegung von Tastern.

Erweiterte LAN Funktionen (Nur Option NP)

Multimediasteuerung

Neben den Funktionen zum Zugriff auf den KNX™ Bus kann der Enertex® EibPC auch LAN Telegramme empfangen, senden und auswerten.

Wir stellen uns wie üblich eine kleine Aufgabe, um das Vorgehen und die Anwendung der LAN Funktionen darzustellen.

Für die Namensauflösung muss der Enertex® EibPC konfiguriert werden.

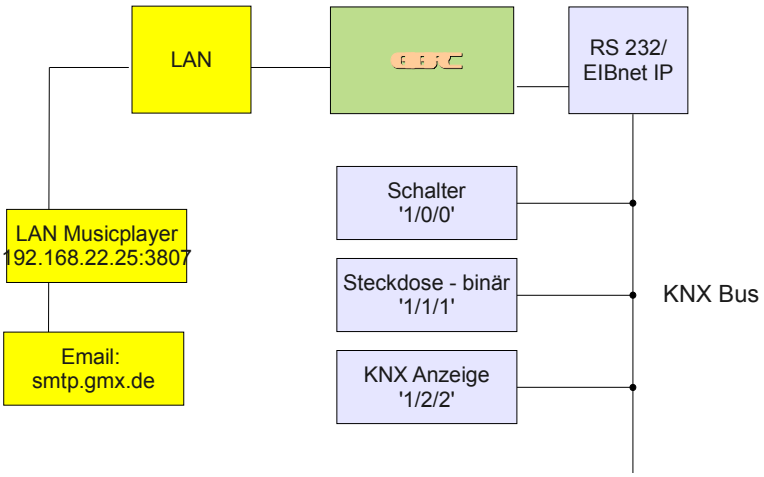


Abbildung 44: Treppenhauslicht

Die Aufgabe: Eine Multimediaapplikation

- Wenn der Schalter 1/0/0 gedrückt wird, soll der Enertex® EibPC die Steckdose, die über Gruppenadresse 1/0/0 gesteuert wird, anschalten (an dieser hängt der Musikplayer).
- Nach 4 Sekunden soll der Enertex® EibPC dem Musicplayer den String „Start Music“ per UDP Telegramm an Port 3807 schicken.
- Der Musikplayer schickt dann auf Port 4806 einen String „Play Music: „Title“ wobei Title der aktuell gespielte Titel sei. Der Enertex® EibPC soll den Titel auf die KNX Anzeige schreiben.
- Nach dem Abspielen von 10 Titeln soll der Enertex® EibPC eine mail an EibPC@enertex.de schicken, wobei er die 10 gespielten Titel aneinander gereiht in die Mail schreiben soll (Trennung durch das Leerzeichen). Header der Mail sei „Ein Hallöchen vom EIBPC“.
- Anschließend soll er dem Musicplayer den String „Stop Music“ per UDP Telegramm an Port 3807 schicken und nach 5 Sekunden die Steckdose wieder ausschalten.

Ein erstes Problem: Der Emailzugang muss eingerichtet werden. Hierzu benötigt der Enertex® EibPC einen DNS Server, sodass dieser die Namensauflösung beherrscht (siehe Menü EibPC → DNS Server):

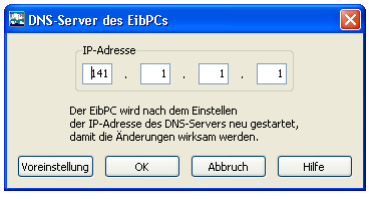


Abbildung 45: DNS Server einrichten

Als zweites müssen wir nun unser E-Mail-Konto einrichten.

Hierzu klicken Sie auf Optionen → E-mail-Einstellungen und tragen ihre Einstellungen für den smtp-Server ein (siehe hierzu auch den Dialog Abbildung 47).

Sie müssen ein E-mail Konto bei einem Provider haben.

Einrichten per Dialog ...

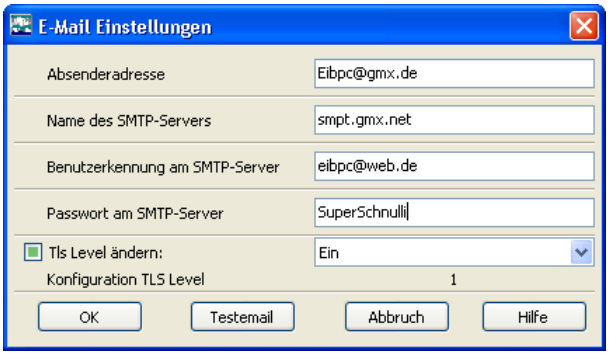


Abbildung 46: E-Mail einrichten

Die eingegeben Daten erscheinen nach dem „OK“ im Anwenderprogramm wie in Abbildung 47 angedeutet und können dort auch direkt editiert werden.

... oder von Hand

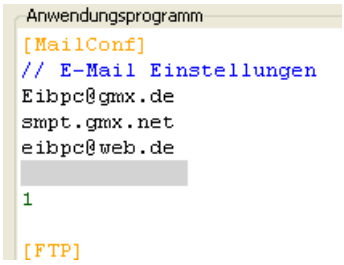


Abbildung 47: E-Mail einrichten

Der Rest ist nun recht schnell erledigt – Sie kennen sich ja bereits mit der Syntax des Enertex® EibPC sehr gut aus. Was Sie nun nur noch wissen müssen: Welche Funktionen gibt es zum UDP- und Mailversand? Wie können Sie größere Zeichenketten (mehr als 14 Zeichen) verarbeiten und definieren?

Zunächst das Programm, dann die Erklärungen:

Mailconfiguration

$ $: Zeichenkette bis zu 1400 Zeichen

sendudp

readudp
split: Auftrennen von 1400er Strings
find: Suchen in 1400er Strings
size: 1400er Stringgröße

sendmail

```
[MailConf]
// E-Mail-Einstellungen
Eibpc@gmx.de
smtp.gmx.de
eibpc@web.de
SuperSchnulli
1

[EibPC]
// Init: Die Titel abspeichern
Titel=$ $
if delay(systemstart(),10000u64) then write ('1/0/0'b01,EIN) endif
if ('1/0/0'b01) then write('1/0/1'b01, EIN); Titel=$ $;TitleNr=0 endif
// Senden von Start Music
if delay('1/0/1'b01,4000u64)  then sendudp(3807u16,192.168.22.25,$ Start Music$) endif
// Strings für die Verarbeitung
StringPlayer= $ $
KNXTitle=$ $c14
TitleNr=0
Port=0u16;IP=0u32
Pattern=$Play Music: $
// Empfangen der Titel
if  event(readudp(Port,IP,StringPlayer)) then                                    \\
        KNXTitle= convert(split(StringPlayer,find(StringPlayer,Pattern,0u16)+size(Pattern) ,END),KNXTitle);   \\
                write('1/1/1'c14,KNXTitle);                                      \\
                Titel=Titel+split(StringPlayer,find(StringPlayer,$Play Music: $,0u16),END);   \\
                TitleNr= TitleNr+1 endif
if TitleNr==10 then sendmail($EibPC@enertex.de$, $Ein Hallöchen vom EIBPC$,Titel);   \\
                  sendudp(3807u16,192.168.22.21,$ Stop Music$);                  \\
                TitleNr=11 endif
if delay(TitleNr==11, 5000u64) then write('1/0/1'b01, AUS) endif
```

$ $: Zeichenkette bis zu 1400 Zeichen

Zuerst haben wir den String *Titel* definiert. Im Gegensatz zum Datentypen c14, ist bei diesem Typ die Zeichenkette bis zu 1400 Zeichen lang. Dieser Datentyp wird als Definition eingerahmt von zwei $-Zeichen und wir bezeichnen ihn als c1400.

$ Text $ entspricht Zeichenkette „ Text “ (Mit Leerzeichen)

Diesen Datentyp können Sie **nicht** auf KNX™ Anzeigeelemente ausgeben bzw. nur mit Hilfe der convert Funktion in einen KNX™ Typ wandeln.

Eine c1400 Zeichenkette wird am Ende mit einem \0 Zeichen angelegt, um deren Länge festzulegen. Dies muss bei den LAN Telegrammen beachtet werden.

Als nächstes machen wir Gebrauch von sendudp:

sendudp: Beliebige Argumente, nicht nur Strings!

sendudp(Port {Typ u16},IP Adresse, Beliebige Anzahl und Typen von Argumente)

sendudp erwartet als erstes Argument eine Variable oder einen Ausdruck vom Datentyp u16. Dieses Argument gibt die Portnummer an, auf der ein UDP Telegramm gesendet werden soll. Die IP Adresse (zweites Argument) ist in der üblichen Schreibweise einzugeben und ist intern kompatibel zum Datentyp u32. Die eigentlichen Daten beginnen ab dem 3. Argument. Diese sind beliebig in Anzahl und Art.

Der Enertex® EibPC sendet selbst immer mit Port 4807. Dieser Port kann nicht verändert werden. Beim Versenden gibt man den Port an, auf welchem der Empfänger das Telegramm entgegen nimmt.

Wenn Ihr LAN Gerät über einen DNS-Namen ansprechbar ist, können Sie auch die Funktion

resolve(String{c1400})

nutzen. Sie können in diesen Fall z.B. einen Ausdruck der Form

sendudp mit Namensauflösung

```
sendudp(3807u16,resolve($www.enertex.HomeMusicPlayer.de$),$ Start Music$)
```

verwenden.
sendudp kann beliebige Datenformate verarbeiten. Sie sind also nicht an die Verwendung von c1400-Zeichenketten (wie im vorliegenden Beispiel) gebunden. Dazu jedoch später mehr.

Das Gegenstück zum Versenden eines UDP Telegramms ist dessen Empfang. Hierzu steht die Funktion readudp zur Verfügung.

readudp: Port, IP und beliebige Argumente, nicht nur Strings!

readudp(Port {Typ u16},IP Adresse{u32}, Beliebige Argumente)

Auch hier gilt: Das erste Argument ist eine Variable oder einen Ausdruck vom Datentyp u16 und stellt den Port der Übertragung dar, auf dem ein UDP Telegramm empfangen wurde. Die IP Adresse (zweites Argument) ist vom Datentyp u32. Die eigentlichen „Nutzdaten“ beginnen ab dem 3. Argument. Diese sind beliebig in Anzahl und Art. Wenn Ihr LAN Gerät über DNS einen Namen ansprechbar ist, können Sie auch hier die Funktion resolve nutzen.

Der Enertex® EibPC empfängt immer auf Port 4806. Dieser Port kann nicht verändert werden und ist vom Sender von UDP Telegrammen zu berücksichtigen.

readudp überprüft im Eingangspuffer des Enertex® EibPC, ob ein UDP Telegramm vorliegt. Wenn dies der Fall ist, so werden die Argumente der Funktion, in jedem Fall *Port* und *IP*, soweit mit Daten „gefüllt“, bis die empfangene Datenmenge (eingetroffenes UDP Telegramm) zur Datenlänge der Argumente der readudp Funktion passen.

Readudp und Event: Ist eine Nachricht eingetroffen?

Eine Besonderheit ist die Verwendung der event-Funktion: Sie zeigt an, ob ein Telegramm eingelesen wurde. Im vorliegenden Beispiel haben wir uns darauf verlassen, dass die Daten vom richtigen Port und Absender kommen. Wir hätten aber auch folgendes Schreiben können, um uns zu versichern...

```
if  event(readudp(Port,IP,StringPlayer)) and (Port==808u16) and IP=192.168.22.21) then \\
```

Wenn wir demnach eine u32 Variable Integer bzw. f32 Variable Float definiert hätten, würde die Anweisungen die empfangenen Daten unterschiedlich interpretieren.

```
readudp(Port,IP,StringPlayer)
readudp(Port,IP,Integer, Float)
```

Nun zur Stringverarbeitung: split

Auseinanderpflücken von Strings

split(String{c1400}, PositionStart{u16}, PositionEnde{u16})

gibt den Teil des Ausgangsstrings zwischen den beiden Positionen (in Zahlen) an. Die größte Position ist hierbei 1399u16, für die auch die „eingebaute“ Konstante END genutzt werden kann.

Sie können Zeichenketten vom Typ c14 und c1400 einfach durch „+“ verketten:

Verketten von Strings

```
Titel=split($ Hallo $,1u16,3u16)+split($ Hallo $,1u16,3u16)
```

würde daher bedeuten, dass *Titel* mit dem String „HaHa“ vorbelegt wird.

Bleibt uns noch die Suche nach einer Zeichenfolge in einer Zeichenkette mit find

Substrings in Zeichenketten finden

find(Quelle{c1400}, SuchString{c1400}, Auftreten{u16})

Mit dieser Funktion kann in der Zeichenkette *Quelle* nach der Postion dem *SuchString* gesucht werden. Dabei kann über die Variable *Auftreten* die davor liegenden gleichen Zeichenketten übersprungen:

```
Pos=find($ Hallo Welt Hallo $,$Hallo$,1u16)
```

In diesem Beispiel ist dann *Pos* mit dem Wert 12u16 belegt. Wenn find nichts findet, so gibt Sie den Wert 65535u16 zurück. Für diesen Wert kann auch die „eingebaute“ Konstante *EOS* (end of string) genutzt werden.

Zurück zum Beispiel:
Jetzt können wir die Zeichenketten des UDP Telegramms verarbeiten:

```
KNXTitle= convert(split(StringPlayer,find(StringPlayer,$Play Music: $,0u16),EOS),KNXTitle)\\
```

Sie erkennen, dass wir am Ende den Datentyp der Zeichenkette in einen KNX c14-Datentyp wandeln müssen. Dies geschieht mit Hilfe der convert Funktion. Die Umwandlung erfolgt durch Herausnehmen der ersten 14 Zeichen aus der Zeichenkette *StringPlayer*.

... und zum Schluss schick'mer eine e-mail

Es bleibt noch die Funktion sendmail

sendmail(ID {Typ c1400}, Header {c1400}, Body{c1400})

Die sendmail-Funktion schreibt eine Mail auf den eingestellten E-mail Server (siehe Abbildung 48). Zu beachten ist, dass bei den angegeben Zeichenketten aus internen Systemgründen nur 255 Zeichen Berücksichtigung finden. Insbesondere wird die E-mail, die der Enertex® EibPC versendet nicht mehr als die ersten 255 Zeichen von *Body* übertragen.

Binärtelegramme

sendudp und readudp können auch zum erarbeiten von binären Daten genutzt werden. In diesem Fall werden also nicht beliebige Zeichenketten ausgewertet, sondern Telegramme, deren Aufbau und Datenanordnung genau spezifiziert sind.

Dazu stellen wir uns nun die folgende Aufgabe, wobei berücksichtigt werden muss, dass der Enertex® EibPC selbst immer nur auf Port 4806 empfängt und seine Absendeport immer 4807 ist.

Verschiedene Telegrammdefinitonen

Der Enertex® EibPC empfängt auf Port 4806 (Port des Senders 5220, Sender 192.168.22.22) Telegramme der Form:

Typ1: Command 1000 bis 1200: Anordung der Daten im Telegramm

```
Command (32 Bit)
Zahl1 (Vorzeichen, 32 Bit)
Fließkommazahl (32 Bit)
Zahl1 (Ohne Vorzeichen 8Bit)
```

Typ2: Command 2000 bis 2200 : Anordung der Daten im Telegramm

```
Command (32 Bit)
Fließkommazahl (32 Bit)
Zahl (Ohne Vorzeichen 8 Bit)
```

– Wenn Command den Wert 1000 hat, soll auf dem gleichen Kommunikationsweg das Ergebnis der Summe der beiden Zahlen gesendet werden. Das Antworttelegramm ist dabei gleich aufgebaut und das Ergebnis ist an der Stelle der Zahl1 zu senden. Command der Antwort betrage 1100.

– Wenn Command den Wert 1001 hat, soll auf dem gleichen Kommunikationsweg das Ergebnis der Differenz der beiden Zahlen als Fließkommazahl gesendet werden. Das Antworttelegramm ist dabei gleich aufgebaut und das Ergebnis ist an der Stelle der Fließkommazahl zu senden. Command der Antwort betrage 1101.

– Wenn Command den Wert 2000 hat, soll die Wurzel der Fließkommazahl berechnet und mit Command 2001 das Ergebnis zurückgeschickt werden. Ist dabei die Fließkommazahl kleiner Null, soll die Ganzzahl des Telegramms auf 1 gesetzt werden, ansonsten auf 0.

Das Programm ist nun einfach geschrieben: Sie werden bemerken, dass wir die Verschachtelung von if-Anweisungen auch hier vermeiden (siehe Seite 54).

Die beiden Variablen Command1 und Command2 sorgen zum einen für die Definition der Einlesefunktionen mit dem readudp (Verarbeitung der Argumente von Typ1 und Typ2).

event(readudp): Wird bei jedem UDP-Telegramm erneut ausgewertet

Command1: Es ist ein Telegramm eingetroffen, IP und Port sind ok.
Command2: dito

```
[EibPC]
// Variablen
Command=0u32;Zahl1=0u32;Zahl2=0u32;Fliesskomma=0f32
IP=0u32;Port=0u16
// Ist das Telegramm auch vom richtigen Port?
CheckSender=(IP==192.168.22.22) and (Port==5220u16)
// Die beiden Telegrammtypen
Command1=event(readudp(Port,IP,Command, Zahl1, Fliesskomma, Zahl2 )) and CheckSender
Command2=event(readudp(Port,IP,Command, Fliesskomma, Zahl1)) and CheckSender

// Summe zurück
if Command==1000u32 and Command1 then                                    \\
        sendudp(Port,IP, 1100u16,Zahl1+Zahl2, Fliesskomma, Zahl2 ) \\
        endif
// Differenz zurück
if Command==1001u32 and Command1 then                                                                    \\
        sendudp(Port,IP, 1100u16,Zahl1, convert(Zahl1,Fliesskomma)-convert(Zahl2,Fliesskomma), Zahl2 ) \\
        endif

// Wurzel
if Command==2001u32 and Command2 and Fliesskomma>0f32 then \\
        sendudp(Port,IP, 2001u16,sqrt(Fliesskomma), 0u32 )      \\
        endif
// Wurzel geht nicht, weil Argument kleiner null
if Command==2001u32 and Command2  and Fliesskomma<0f32 then  \\
        sendudp(Port,IP, 2001u16,Fliesskomma, 1u32 )            \\
        endif
```

event(readudp(...)) wechselt bei **jedem beliebigen, eintreffenden** UDP-Telegramm von Port 5220 kurz von 0b01 auf 1b01.

Command1 und *Command2* reagieren dann (Verknüpfung mit *CheckSender*) bei jedem Telegramm, das 192.168.22.22 von seinem Port 5220 (Zielport beim Enertex® EibPC ist 4806) sendet.

Durch die Verknüpfung mit der Variablen *Command* wird dann die gewünschte Telegrammauswertung vorgenommen. Demnach sind *Command1* und *Command2* vom Verhalten absolut identisch, insbesondere ist *Command1* auch auf EIN wenn *Command2* auf EIN geht.

Alternativ kann man daher auch *Command1* und *Command2* in einem Telegramm zusammenfassen, wie im Folgenden mit der Variablen *Telegramm* realisiert wurde.

```
[EibPC]
// Variablen
Command=0u32;Zahl1=0u32;Zahl2=0u32;Fliesskomma=0f32
IP=0u32;Port=0u16
// Die beiden Telegrammtypen
CheckSender=(IP==192.168.22.22) and (Port==5220u16)
Telegramm=(event(readudp(Port,IP,Command, Zahl1, Fliesskomma, Zahl2 )) or event(readudp(Port,IP,Command,
Fliesskomma, Zahl1))) and CheckSender

// Summe zurück
if Command==1000u32 and Telegramm then                                    \\
        sendudp(Port,IP, 1100u16,Zahl1+Zahl2, Fliesskomma, Zahl2 ) \\
        endif

// Differenz zurück
if Command==1001u32 and Telegramm then                                                    \\
        sendudp(Port,IP, 1100u16,Zahl1, convert(Zahl1,Fliesskomma)-convert(Zahl2,Fliesskomma), Zahl2 ) \\
        endif

// Wurzel
if Command==2001u32 and Telegramm and Fliesskomma>0f32 then  \\
        sendudp(Port,IP, 2001u16,sqrt(Fliesskomma), 0u32 ) \\
        endif
// Wurzel
if Command==2001u32 and Telegramm and Fliesskomma<0f32 then \\
        sendudp(Port,IP, 2001u16,Fliesskomma, 1u32 ) \\
        endif
```

TCP/IP Server und Client

Wenn Ihr Enertex® EibPC mit Software-Option NP ausgestattet ist, arbeitet dieser grundsätzlich als TCP/IP Server auf Port 4809. Gleichermaßen stellt der Enertex® EibPC auch einen TCP/IP Client zur Verfügung.

Die Syntax für die Implementierung einer TCP/IP Kommunikation ist exakt die Gleiche wie bei der UDP-Kommunikation. Daher können wir an dieser Stelle einfach auf die vorherigen Seiten verweisen. Der Vorteil liegt für den Anwender darin, dass mit gleichem Vorgehen nunmehr einfach eine TCP Verbindung aufgebaut werden kann. Daher sind entsprechend Codezeilen einfach wiederzuverwenden.

Zum Überblick in Kurzform:

Gleiche Syntax, nur „tcp" statt „udp"

```
sendtcp(Port {Typ u16},IP Adresse, Beliebige Anzahl und Typen von Argumente)
```

sendtcp erwartet als erstes Argument eine Variable oder einen Ausdruck vom Datentyp u16. Dieses Argument gibt die Portnummer an, auf der ein TCP/IP-Telegramm gesendet werden soll. Die IP Adresse (zweites Argument) ist in der üblichen Schreibweise einzugeben und ist intern kompatibel zum Datentyp u32. Die eigentlichen Daten beginnen ab dem 3. Argument. Diese sind beliebig in Anzahl und Art.

Der Enertex® EibPC sendet selbst immer mit Port 4809. Dieser Port kann nicht verändert werden. Beim Versenden gibt man den Port an, auf welchem der Empfänger das Telegramm entgegen nimmt.

Zum Empfangen von TCP Telegrammen steht die Funktion readtcp zur Verfügung.

```
readtcp(Port {Typ u16},IP Adresse{u32}, Beliebige Argumente)
```

Auch hier gilt: Das erste Argument ist eine Variable oder ein Ausdruck vom Datentyp u16 und stellt den Port der Übertragung dar, auf dem ein TCP-Telegramm empfangen wurde. Die IP Adresse (zweites Argument) ist vom Datentyp u32. Die eigentlichen „Nutzdaten" beginnen ab dem 3 Argument. Diese sind beliebig in Anzahl und Art. Wieder gilt: Wenn Ihr LAN Gerät über DNS mit einem Namen ansprechbar ist, können Sie auch hier die Funktion resolve nutzen.

readtcp füllt seine Argumente mit Daten

readtcp überprüft im Eingangspuffer des Enertex® EibPC, ob ein TCP-Telegramm vorliegt. Wenn dies der Fall ist, so werden die Argumente der Funktion, in jedem Fall *Port* und *IP*, soweit mit Daten „gefüllt", bis die empfangene Datenmenge (eingetroffenes TCP-Telegramm) zur Datenlänge der Argumente der readtcp Funktion passen.

Um zu überprüfen, ob eine Nachricht eingetroffen ist, können Sie wie bei den UDP-Telegrammen auf event(readtcp()) zurückgreifen.

Ereignis aufgetreten?

```
if event(readtcp(Port,IP,StringPlayer)) and (Port==808u16) and (IP==192.168.22.21) then \\
```

Client und Server

Der Enertex® EibPC baut sowohl Client als auch Server Verbindungen mit sendtcp und readtcp auf.

Wenn Sie den Enertex® EibPC als Server betreiben, muss sich ein Client am Enertex® EibPC anmelden. Das geschieht ohne Ihr Zutun vom Betriebssystem automatisch, da der Enertex® EibPC immer empfangsbereit ist.

Wenn der Client eine Nachricht an den Enertex® EibPC schickt, so wird event(readtcp()) aktiv. Im Programm muss man nun „nachschauen", ob die IP gerade die von einem gewünschten Teilnehmer kommt und dann entsprechend verarbeiten. Nach 30 Sekunden ohne Aktivität auf einer bestehenden Verbindung trennt der Enertex® EibPC diese automatisch.

Soll der Enertex® EibPC als Client arbeiten, so muss er sich am Server mittels

```
connecttcp(Port {Typ u16},IP Adresse{u32})
```

anmelden.

Um ordnungsgemäß die Verbindung zu trennen, benötigt man:

```
closetcp(Port {Typ u16},IP Adresse{u32})
```

Mehr Details

Unter www.enertex.de/downloads/d-eibpc/DokuCF-1.pdf finden sie ein ausführliches Beispiel für die Implementierung einer komplexeren Kommandoverarbeitung, wie Sie für Command Fusion realisiert wurde.

Webserver

Der Enertex® EibPC kann bei der Ausstattung mit Option NP auch eine Visualisierung und Steuerung Ihrer KNX™-Installation vornehmen. Die Visualisierung ist mit Hilfe eines Internet-Browsers möglich, wobei bei der Enertex® Bayern GmbH mit dem Mozilla Firefox getestet wird.

Der Webserver im EibPC wird über die Eingabe der IP in der Adresszeile des Browsers aufgerufen.

Visualisierungsassistent

Die Oberfläche kann bequem mit dem Enertex Visualisierungsassistenten erstellt werden.

In nur sechs Schritten zur eigenen Visu:

Schritt 1 - Öffnen des Assistenten

Wunderwerk „VisuAssi“

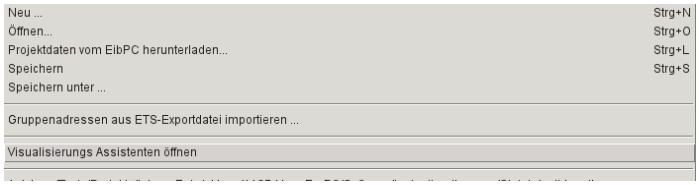


Abbildung 49: Aufrufen des Visualisierungsassistenen

Schritt 2 – Stockwerke und Räumlichkeiten anlegen

Jeder Raum wird in der Visualisierung eine eigene Seite.

Räumlichkeiten definieren

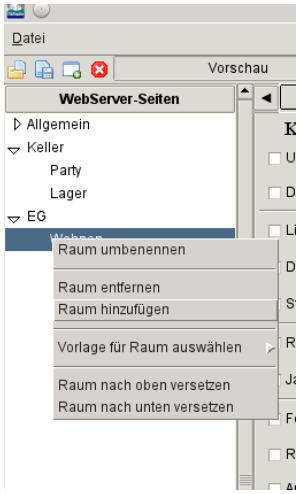


Abbildung 50: Seiten anlegen

Schritt 3 – Räume individuell gestalten

Jedem Raum kann einzeln und schnell parametriert werden.

Zur Auswahl stehen div. Standartelemente wie Lichter, Dimmer, Rollo, Raumkontroller.

Anzahl der gewünschten Elemente vorgeben. Die Vorschau zeigt symbolisch die im Entstehen begriffene Seite.

Jede Seite kann individuell gestaltet werden

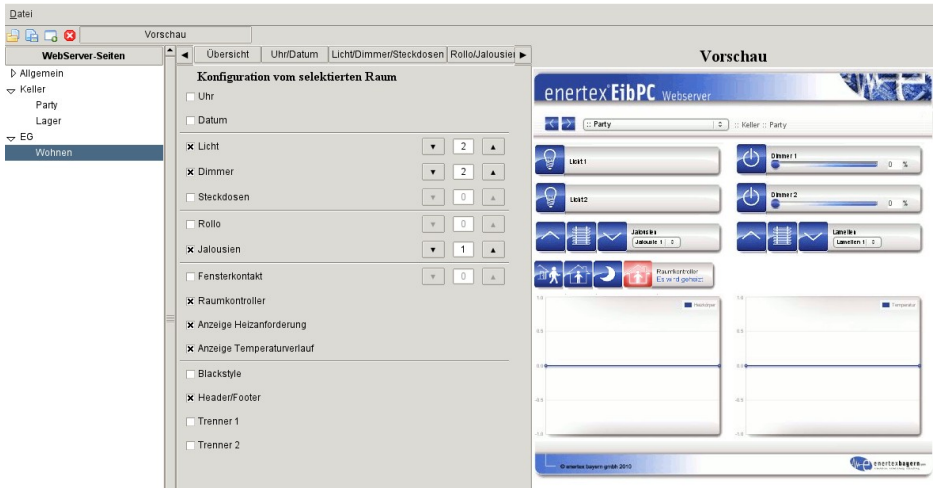


Abbildung 51: Seiten anlegen

Schritt 4 – Gruppenadressen mit Elementen verbinden

In der Menüleiste entsprechend die Reiter auswählen, Anzeigetexte eingeben und per Drag and Drop die entsprechenden Gruppenadressen dem Element zuordnen.

Gruppenadressen und Elemente verbinden

Abbildung 52: Gruppenadressen mit den Elementen verbinden

Schritt 5 – Visualisierungsprojekt speichern und in das EPC-Programm laden

Beim Speichern prüft der Asisstent die vorhandenen Verknüpfungen und warnt bei Unstimmigkeiten oder fehlenden Gruppenadressen.

Nachdem alle Elemente mit einer Gruppenadresse verknüpft sind muss das Projekt in das EibPC-Programm geladen werden.

Ein Dialog führt durch die nötigen Schritte.

Speichern, laden, fertig

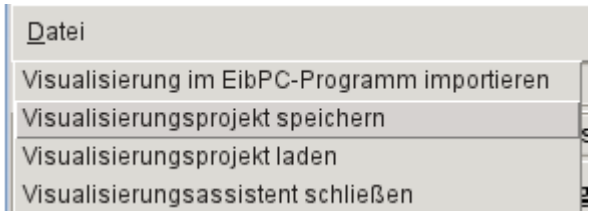


Abbildung 53: Visualisierungsprojekt speichern und importieren

Es werden drei Dateien angelegt und per *#include* in das epc-Programm eingebunden. Dies macht der Assistent völlig autonom.

Am Ende muss die aktuelle EnertexWebV*x*.lib in das epc-Programm geladen werden.

Wie Makro-Bibliotheken eingebunden werden, ist auf Seite 230 erklärt.

das modifizierte epc-Programm

```
[MacroLibs]
//Makro-Bibliotheken
EnertexWebV2.lib

[WebServer]
#include VA_EibPC_Webserver.epc
[Macros]
#include VA_EibPC_Macros.epc

[ETS-ESF]
// Die aus der ETS3 exportierte ESF-Datei
1148-Gebaeude-6.esf

[EibPC]
#include VA_EibPC.epc
```

Schritt 6 – Programm übertragen und Browser öffnen

Das Programm muss an den EibPC übertragen werden (F3).

Ggf. kann es nach der Übertragung zu Verzögerungen kommen, da alle Status für den Webserver abgefragt werden.

Der Webserver im EibPC wir über die Eingabe der IP in der Adresszeile des Browsers aufgerufen.

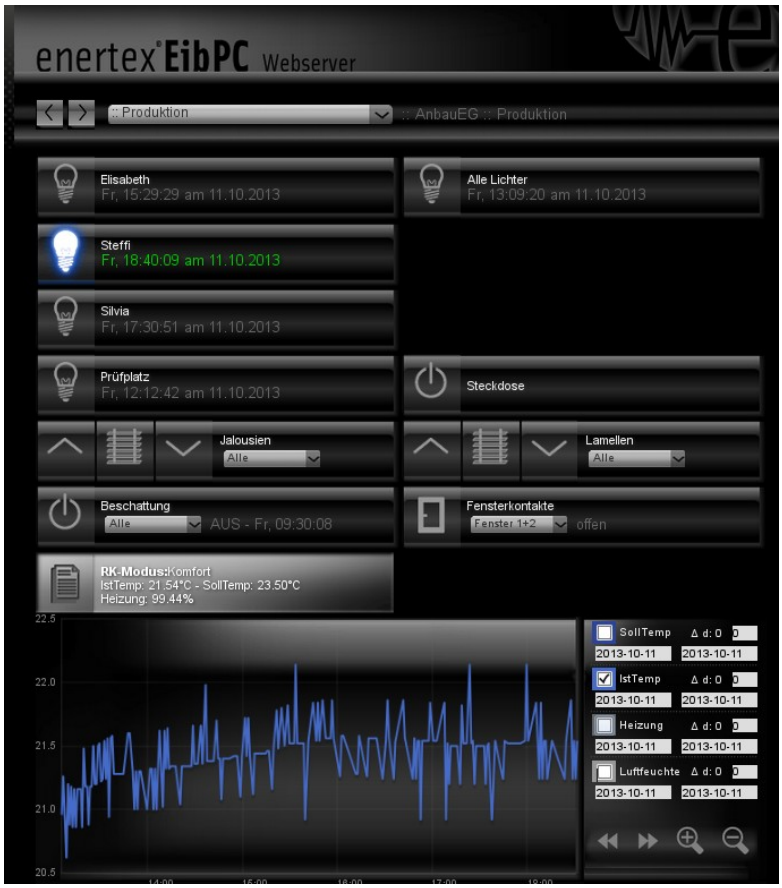


Abbildung 54: Webserver Beispiel

Sollten Sie tiefer in die Oberflächenprogrammierung einsteigen wollen, so lesen Sie bitte folgendes Kapitel aufmerksam durch.

Einseiten-Version

Für unser Beispiel nehmen wir an, dass wir

- eine Schaltuhr steuern,
- einen Temperaturverlauf des Wohnzimmers darstellen,
- die Werte der Wetterstation anzeigen
- und einen Lichtschalter realisieren wollen.

Am Ende wird der Webserver schließlich wie in Abbildung 55 aussehen. Wir zeigen nun, wie das Ganze funktioniert. Dabei nutzen wir zunächst nur die Einseiten-Version des Webservers. Wenn auf dem Browser in der Adressleiste die IP des Enertex® EibPC eingegeben wird, dann wird die Ansicht des Webservers sichtbar. Diese kann wie folgt aussehen:

So wird unser Webserver aussehen

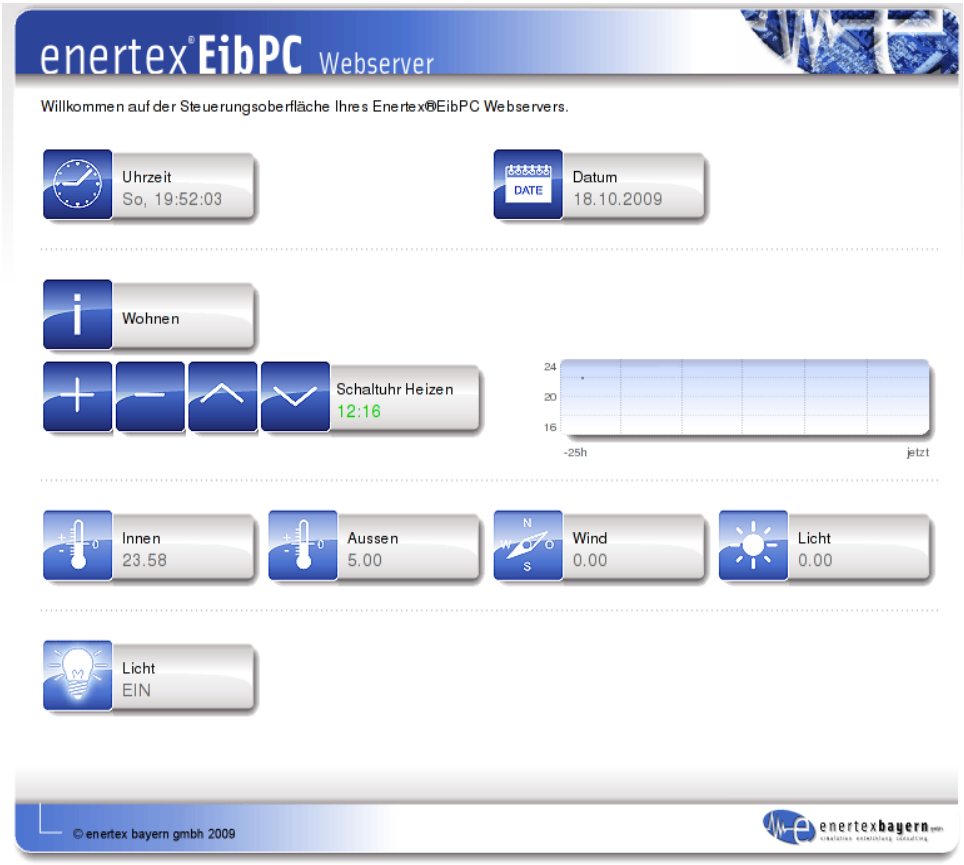


Abbildung 55: Der Webserver

Das minimale Design: 4 Spalten, 5 Zeilen

Der Webserver ordnet wie in einem Schachbrettmuster seine Elemente an, die entweder eine vorgegebene einfache oder doppelte Länge aufweisen. Es gibt dabei grundsätzlich die Möglichkeit, auch Felder freizulassen sowie Trennlinien einzufügen.

Die Anordnung und die Konfiguration der Webelemente wird in der Sektion [WebServer] vorgenommen. Durch die feste Vorgabe von eingebauten Icons, Längen und Variablen kann die Konfiguration ohne grafischen Aufwand rein textbasiert erfolgen. Dadurch kann auf sehr einfache Weise ein professionelles Web-Interface erstellt werden. Die Icons (Übersicht s. Seite 300) sind in einem einheitlichem Design erstellt.

Das minimale Design sieht vier Spalten und fünf Zeilen vor. Die Kopf- und Fußzeile können abgeschaltet werden, so dass auf Touchpanels mit 800x600 Zeilen Auflösung durch die Gestaltung der Bedienelemente eine sehr ergonomische Oberfläche entsteht. Wenn die Zeilen nicht in der Sektion [WebServer] definiert werden, so bleiben die Flächen unbesetzt. Im einfachsten Fall besteht dann der Webserver aus einer leeren Seite.

Zunächst müssen die Elemente konfiguriert werden. Dies erfolgt im Anwenderprogramm wie folgt:

```
[EibPC]

[WebServer]
button(1)[CLOCK]$Uhrzeit$ none button(2)[DATE]$Datum$
line
button(3)[INFO]$Wohnen$
shifter(4)[PLUS,MINUS,UP,DOWN]$Schaltuhr Heizen$  chart(5)[$16$,$20$,$24$]
line
button(6)[TEMPERATURE]$Innen$  button(7)[TEMPERATURE]$Aussen$   \\
                                    button(8)[WIND]$Wind$  button(9)[WEATHER]$Licht$
line
button(10)[LIGHT]$Licht$
```

Was man alles konfigurieren kann

Es gibt die Elemente

- *button*: 1-fache Länge, ein grafisches Symbol (Icon), eine fest vorzugebende und eine variable Textzeile, Symbol dynamisch veränderbar
- *shifter*: 2-fache Länge, ein bis vier grafische Symbole (Icon), eine fest vorzugebende und eine variable Textzeile, ein Symbol dynamisch veränderbar
- *chart*: 2-fache Länge, dient zur Anzeige von Verläufen (XY-Diagrammen)
- *none*: Leerelement
- *line*: Dünne Linie zur Abtrennung
- In der Übersicht auf Seite 300 findet man die möglichen Symbole, die entweder über Indizes (Zahlen) oder die vordefinierten Konstanten angesprochen werden können.

Schauen wir noch einmal auf das Bildungsgesetz zur Erzeugung des Webservers:

Grundregeln zum Bauen des Webservers

1. Jedes Element hat eine ID (maximal 39 Elemente).
2. Jede Zeile der Konfiguration entspricht einer Zeile der Visualisierung.
3. Elemente werden durch ein oder mehrere Leerzeichen voneinander getrennt.
4. Leere Elemente (*none*) werden automatisch generiert (z. B. die letzte Zeile des Webservers).
5. Der Webserver besteht aus maximal zehn Spalten. Jede Zeile der Konfiguration entspricht einer Zeile der Visualisierung.

Damit sind wir schon in der Lage, den Webserver zu „designen“.

Nun aber stellt sich die Frage: Wie können wir im Anwenderprogramm auf die Eingaben des Anwenders reagieren?

Dem aufmerksamen Leser werden schon in den Definitionen der Elemente die Zahlen in runden Klammern aufgefallen sein. Diese dienen der Zuordnung im Anwenderprogramm, wobei sich die Elemente *button* und *shifter* mit Hilfe der Funktionen webbutton und webdisplay bzw. *chart* mit Hilfe der Funktion webchart ansprechen lassen.

Knopfdruck abfragen

Knöpfe und Texte verändern

Die Funktion

webbutton*(ID{u08})*

Fließkommazahlen werden von der Funktion direkt in Zeichenketten umgewandelt. Datumsangaben, wie z.B. die Rückgabe von settime() etc. ebenso.

gibt bei Betätigen bzw. Drücken der Schaltfläche eines Webbuttons (*button* oder *shifter*) mit der *ID* (Zahl zwischen 0 und 255) für einen Verarbeitungszyklus einen Wert ungleich Null zurück. Bei einem *button*-Element wechselt bei Betätigung die Rückgabe von 0 auf 1 (u08) und wieder zurück auf 0. Bei einem *shifter*-Element wechselt bei Betätigung die Rückgabe auf 1,2,3 oder 4 (u08), je nachdem, welches Schaltelement des *shifters* betätigt wird. Dabei kann der *shifter* variabel einen, zwei, drei oder vier Icons anzeigen. Die Nummern beziehen sich dabei auf die Anordnung der Schaltelemente von links nach rechts. Wenn *button* oder *shifter* die gleiche *ID* aufweisen, so reagiert die Funktion webbutton auch in beiden Fällen der Betätigung zweimal. *ID* kann wahllos vergeben und muss nicht fortlaufend nummeriert werden. Allerdings sind nur Zahlen zwischen 0 und 255 erlaubt.

Abbildung 55 zeigt, dass *button* bzw. *shifter* jeweils aus Grafiken und zwei Textzeilen bestehen. Die erste Textzeile wurde bereits in der obigen Konfiguration am Ende des Elements angegeben und kann während der Laufzeit nicht mehr verändert werden. Die zweite Zeile wird mit der Funktion

webdisplay(*ID{u08},Text{beliebiger Datentyp},Icon{u08},Zustand{u08},TextStil{u08})*

verändert. Bei Aufruf der Funktion wird zudem das Icon des Webelements mit der *ID* (Zahl zwischen 0 und 255) auf das Symbol mit der Nummer *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt. Außerdem erleichtern vordefinierte Konstanten die Auswahl, die in Tabelle 4 (Seite 301) aufgelistet sind. Das Argument *Text* bezeichnet eine beliebige Variable, deren in einen String gewandelter Wert, auf die variable Textzeile des Webelements ausgegeben wird. Jedes Icon hat mindestens die Zustände ACTIVE (==1), INACTIVE (==2), DARKRED (==0) und BRIGHTRED (==9).

Einer dieser Zustände kann im Argument *State* übergeben werden. Eine Übersicht über die möglichen Zustände gibt Tabelle 5 (Seite 301). Der auszugebende Text kann in den Stilen GREY (==0), GREEN (==1), BLINKRED(==2) und BLINKBLUE (==3) dargestellt werden.

Für die Anzeige von Werten nutzen wir den Zustand DISPLAY (==3) des betreffenden Icons. Dieser Zustand existiert nicht für alle Icons, siehe auch Seite 300.

Schließlich fehlt uns noch die Möglichkeit, das XY-Diagramm mit Werten zu befüllen. Dazu gibt es die Funktion

webchart(*ID, Var{u08}, X1{c14}, X2{c14}*)

Die Funktion spricht das XY-Diagramm *chart* an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser *ID* angesprochen. Allerdings stören sich die *IDs* eines *charts* und eines *buttons* bzw. *shifters* nicht. Bei Aufruf der Funktion wird die XY-Darstellung des Wertes *Var* aktiviert. Es können Werte aus dem Bereich 1...30 dargestellt werden. 0 bedeutet keine Darstellung, Werte größer als 30 sind unzulässig und werden wie 0 interpretiert. Bei jedem Aufruf der Funktion werden zunächst die Werte von links beginnend dargestellt. Wenn nach 47 Aufrufen das Ende erreicht ist werden die Werte nach links geschoben. Die X-Achsenbeschriftung wird mit den Argumenten *X1, X2* (Datentyp c14) vorgegeben und kann während der Laufzeit angepasst werden.

Damit wird nun unser Anwenderprogramm wie folgt ausschauen:

Der Graph wird bei jedem Aufruf der Funktion weitergeschoben.

```
[EibPC]
// Webserver 1
if stime(0) then webdisplay(21,settime(),CLOCK,INACTIVE,GREY) endif
if stime(0) then webdisplay(22,setdate(),CLOCK,INACTIVE,GREY) endif
// Zyklisch die Wohnzimmertemperatur anzeigen
if cycle(30,0) then webchart(10,convert(3.75*("RkWohnzimmerTemp-3/1/28"-15.0),0), $-25h$c14,$jetzt$c14)
endif
// Schaltuhr einstellen
SchaltuhrH=0
SchaltuhrM=0
if webbutton(6)==1 then SchaltuhrH=SchaltuhrH+1 endif
if webbutton(6)==2 then SchaltuhrH=SchaltuhrH-1 endif
if webbutton(6)==3 then SchaltuhrM=SchaltuhrM+1 endif
if webbutton(6)==4 then SchaltuhrM=SchaltuhrM-1 endif
if SchaltuhrH>23 then SchaltuhrH=0 endif
if SchaltuhrM>59 then SchaltuhrM=0 endif
if change(SchaltuhrM) or change(SchaltuhrH) then \\
        webdisplay(0,convert(SchaltuhrH,$$c14)+$:$c14+convert(SchaltuhrM,\\
        $$c14),DOWN,INACTIVE,GREEN) endif

// Wetterstation
if stime(0) then webdisplay(2,"RkWohnzimmerTemp-3/1/28",TEMPERATURE,DISPLAY,GREY) endif
if stime(0) then webdisplay(3,"ThermeAußentemperatur-3/3/13",TEMPERATURE,DISPLAY,GREY) endif
if stime(0) then webdisplay(4,"Wind-3/5/1",WIND,DISPLAY,GREY) endif
if stime(0) then webdisplay(5,"Licht-3/5/2",WEATHER,DISPLAY,GREY) endif

// Lichtschalter
if webbutton(30)==1 then if "ArbeitszimmerLicht-1/1/16" then write("ArbeitszimmerLicht-1/1/16",AUS) endif; \\
                         if !"ArbeitszimmerLicht-1/1/16"  then write("ArbeitszimmerLicht-1/1/16",EIN) endif \\
                         endif
if "ArbeitszimmerLicht-1/1/16" then webdisplay(30,$EIN$c14,LIGHT,ACTIVE,GREY) else \\
webdisplay(30,$AUS$c14,LIGHT,INACTIVE,GREY) endif
```

Skala im chart

Das webchart kann 47 Ganzzahlwerte auf der X-Achse und 30 Ganzzahlwerte (Einteilungen) auf der Y-Achse darstellen. Die X-Achse entspricht in unserem Beispiel dem Zeitraum und die Y-Achse dem Messwert. Da wir Raumtemperaturen darstellen, ist eine Skala von 16 bis 24 (falls man keine Frostüberwachung macht) sinnvoll.

Über cycle stellen wir das Zeitintervall (X-Achse) der Anzeige ein. Der gesamte, im webchart dargestellte Zeitraum, berechnet sich aus 47 mal Zeitintervall, also im Beispiel 30x47 min= 23.5 Std.

Die Errechnung der Skalierung (Y-Achse) gestaltet sich wie folgt: 8 Grad (16 bis 24) verteilen sich auf 30 Stufen. Daher wählen wir 30/8 = 3,75 als Skalierung für die Temperaturwerte.

Zugriff auf Variablendefinitionen für die IDs der Konfiguration

Da wir im Anwendungsprogramm die Webelemente mit den gleichen IDs wie bei der Konfiguration des Webservers verwenden, können wir mit u08-Variablendefinitionen in der Sektion [EibPC] Konstanten für den Zugriff generieren:

```
[WebServer]
button(ClockWebID)[CLOCK]$Uhrzeit$ none button(DateWebID)[DATE]$Datum$
[EibPC]
ClockWebID=22
DateWebID=21
// Webserver
if stime(0) then webdisplay(ClockWebID,settime(),CLOCK,INACTIVE,GREY) endif
if stime(0) then webdisplay(DateWebID,setdate(),CLOCK,INACTIVE,GREY) endif
```

Beachten Sie, dass der Webserver dabei die Variable statisch auswertet. Wenn sich also die Variable *ClockWebID* während der Laufzeit ändert, bezieht sich der Index immer noch auf den Anfangswert 22.

Das Bedienkonzept des Webeservers

Was passiert beim Klicken auf die Buttons?

Der integrierte Webserver ist so aufgebaut, dass er bei Tastendruck auf die Schaltflächen im Webbrowser sofort reagiert und an die Verarbeitungsschleife eine entsprechende Information schickt. Beim Tastendruck wechselt zusätzlich das Icon einer Symbolgruppe immer sofort auf den Zustand ACTIVE, der durch einen Beleuchtungseffekt gekennzeichnet ist. Damit soll dem Anwender die Erkennung der Betätigung erleichtert werden.

Wenn das Anwenderprogramm auf den Tastendruck reagiert, indem z. B. mit webdisplay oder webchart der Zustand der Anzeigen geändert wird, wird die HTML-Datei des Webservers modifiziert. Etwa 1,5 Sekunden nach Betätigung schickt der Webserver an den Browser einen Updatebefehl, der dazu führt, dass die neue HTML-Datei aufgerufen wird. Diese Reaktionszeit hängt von der Auslastung des Enertex® EibPCs ab und kann etwas ansteigen. Zudem erzeugt der Webserver diesen Update-Befehl alle 30 Sekunden auch ohne Benutzereingaben am Webbrowser, so dass das Webinterface einen aktuellen Zustand darstellt.

Mehr-Seiten Version

Mehr-Seiten-Webserver im „Enertexblau“-Design

Abbildung 56: Der Webserver – Mehr-Seiten Version im Blue-Design

Mehr-Seiten-Webserver im „Black“-Design

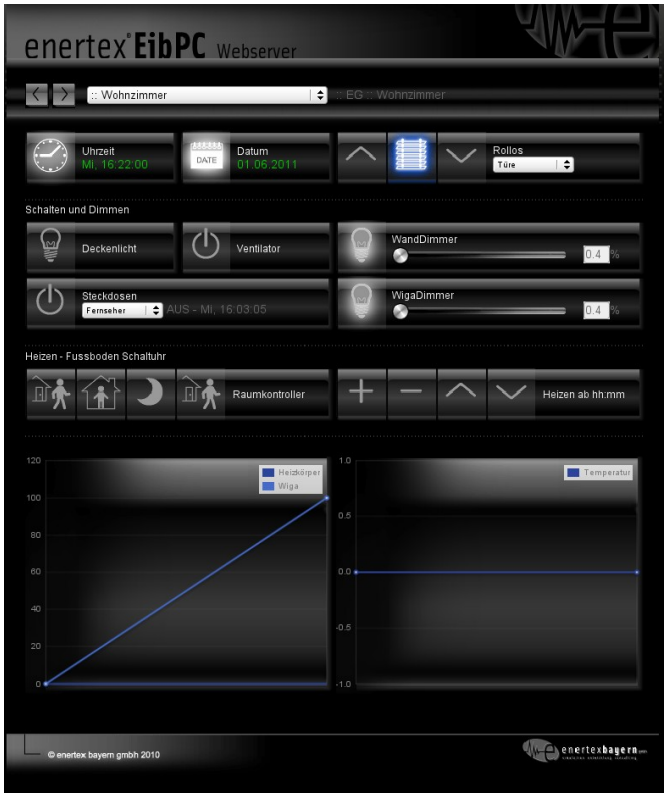


Abbildung 57: Der Webserver - Mehr-SeitenbVersion im Black-Design

In den Abbildungen 56 und 57 sind die beiden aktuellen Desgin-Variationen dargestellt. Standardmäßig wird die Blue-Version verwendet.

Im Folgenden zeigen wir nun die Anwendung mit mehreren Seiten.

Bei der Verwendung von mehreren Seiten, ist es sinnvoll, diese Seiten thematisch in Gruppen aufzuteilen, z.B. Keller. Jede Seite wiederum kann ihre eigene Überschrift tragen, also z.B. Heizraum. Der Webserver zeigt über die Listbox der Seitennavigation (vgl. Abbildung 58) diese Zuordnung an.

Abbildung 58: Seitennavigation

Zum Generieren einer seitenbasierten Visualisierung benötigen wir zunächst das page-Element in der Sektion [WebServer].

Eine Seite definieren

```
[WebServer]
page(HeizRaumPageID)[$Keller$,$Heizung$]
[EibPC]
HeizRaumPageID=1
```

Mit IDs arbeiten, dann wird's später einfacher

Eine Seite wird also mit

page(ID)[$GruppenName$,$SeitenName$]

erzeugt. Dieses Kommando generiert die Seitennavigation automatisch. Wenn eine Seite mit diesem Kommando einer Gruppe, im Beispiel „Keller“ zugeordnet wird, erscheint diese dann so geordnet in der Auswahlbox. Die Seite selbst trägt den Titel „Heizung“. Die Schnellauswahl (Vor- bzw. Zurückknopf in Abbildung 58) ist durch die Reihenfolge der Definition gegeben.

Mit *header* und *footer* können Sie die interne Seitenkopf bzw. Fußzeile ausschalten. Sie haben zudem die Möglichkeit, eine Grafik einer externen Quelle einzubinden. Die Gestaltung der Kopf- und Fußzeile ist für alle Seiten gleich.

lokales Element:
Das Element ist an die Seite gebunden.
globales Element:
Das Element wird auf allen Seiten gleich dargestellt.

Für die Anzeige gibt es globale oder lokale Elemente. Global bedeutet, dass das Element auf verschiedenen Seiten verwendet werden kann, aber nur ein einziges mal erzeugt wurde. Ein Zugriff bzw. eine Veränderung des Elements durch das Anwendungsprogramm ist daher für alle Seiten gleich. Hingegen kann ein lokales Element nur in einer Seite verändert werden. Vom Design sind lokale und globale Elemente identisch, wobei erstere durch den Zusatz „p“ (page) gekennzeichnet sind. Die Seitennummerierung beginnt bei 1. Wir erklären das am Beispiel und erweitern obigen Code:

Gobale IDs dürfen mit lokalen übereinstimmen. Es werden aber getrennte Objekte angesprochen.

```
[WebServer]
// Seite 1
page(HeizRaumPageID)[$Keller$,$Heizung$]
button(LichtID)[LIGHT]$Alle Lichter$ none pbutton(SteckerID)[LIGHT]$Steckdosen$
// Seite 2
page(SchlafenPageID)[$OG$,$Schlafen$]
button(LichtID)[LIGHT]$Alle Lichter$ none pbutton(SteckerID)[LIGHT]$Steckdosen$

[EibPC]
KellerID=0
HeizRaumPageID=1
SchlafenPageID=2
LichtID=1
SteckerID=1
if '1/2/4'b01 then display(LichtID,$EIN$,LIGHT,ACTIVE,BLINKBLUE)  else \\
                        display(LichtID,$AUS$,LIGHT,INACTIVE,GREY) endif
if '1/2/5'b01 then display(SteckerID,$EIN$,LIGHT,ACTIVE,BLINKBLUE,HeizRaumPageID)  else \\
                        display(SteckerID,$AUS$,LIGHT,INACTIVE,GREY,HeizRaumPageID) endif
if button(LichtID)==1 then write('1/2/4'b01, AUS) endif
if pbutton(SteckerID,HeizRaumPageID)==1 then write('1/2/5'b01,AUS) endif
```

Der *button* mit der *LichtID* ist global, er ist auf jeder Seite gleich. Wenn man also dessen Status-Icon bzw. Statustextzeile mit display (bzw. webdisplay) verändert, so ändert dieser sich auf allen Seiten. Anders ist es bei dem lokalem Element *pbutton* mit der *SteckerID*. Bei Änderung der Gruppenadresse ('1/2/5'b01) wird nur das Element mit der *SteckerID* und der SeitenID *HeizRaumPageID* (d.h. auf das Element der *SteckerID* auf dieser Seite) verändert. Um eine Änderung der Grafik bzw. des Textes zu erzielen, muss man hier die Funktion pdisplay nutzen. Um es nochmals klar zu machen, zeigen wir folgende Abbildung:

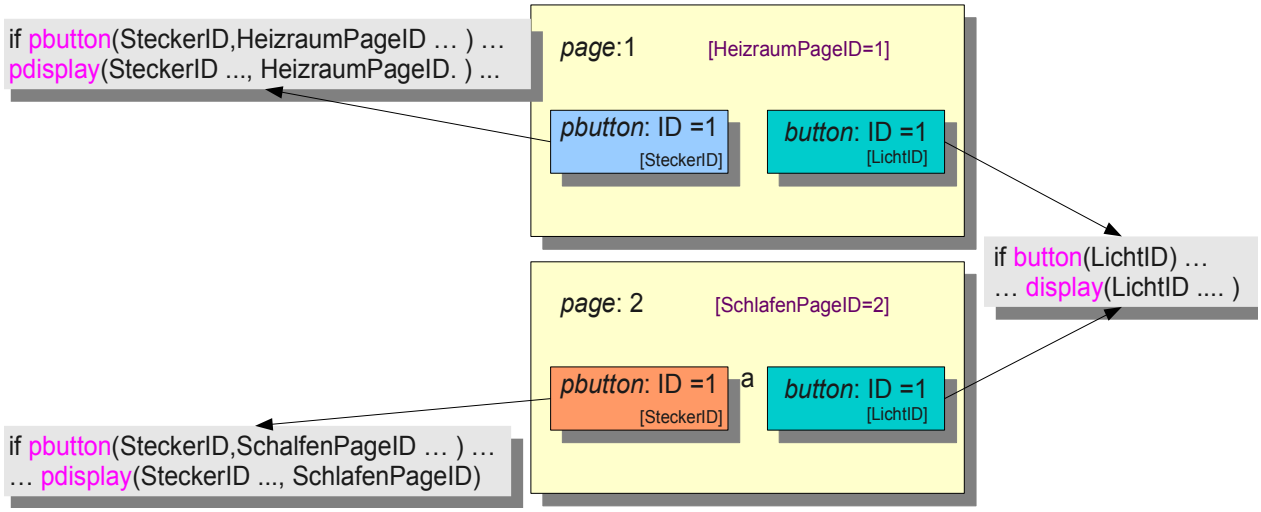


Abbildung 59:Globale und lokale (seitengebundene) Elemente

Auch bei der der Abfrage des Status mittels den Funktionen button bzw. pbutton muss zwischen den globalen und seitenbezogenen Elementen (Buchstabe „p“) unterschieden werden. Die erstere Funktion bezieht sich auf den Zustand eines globalen Knopfes, die zweite auf einen seitenbezogenen (=lokalen).

Mit globalen Elementen können Alarmmeldungen oder Zeitanzeigen für jede Seite realisiert werden.

Der fundamentale Vorteil eines globalen Elements ist die Eigenschaft, dass es auf allen Seiten gleich dargestellt wird. Auf diese Weise lassen sich Alarmmeldungen und Zustandsanzeigen (z.B. Zeit) erzeugen, die für alle Seiten gleichermaßen zu nutzen sind. Rein grafisch sind dabei globale Elemente nicht von den lokalen zu unterscheiden.

In gleicher Weise gilt das für sämtliche weitere Elemente, die sich durch den Zusatz *p* in der Namensgebung unterscheiden.

Ein interessantes Element stellt der *mbutton* dar (Abbildung 60).
Wir betrachten diesen zunächst in der „globalen“ Variante. Dieses Element hat zwei Bedienebenen.

Bedienelement und Auswahlbox

1. Links den Betätigungsknopf samt Symboldarstellung
2. Eine konfigurierbare Auswahlbox mit bis zu 255 Elementen

Abbildung 60:mbutton

Für den Bediener der Visualisierung stellt sich das Konzept wie folgt dar: Durch Drücken auf die Auswahlbox kann das Schaltelement konfiguriert bzw. ausgewählt werden. Wenn die Webseite z. B. einen Raum mit vier schaltbaren Steckdosen visualisiert, kann hier ausgewählt werden, welche Steckdose geschaltet wird, wenn der linke Button gedrückt wird. Durch Drücken des Betätigungsknopfes wird dann die gewünschte Aktion ausgeführt.

Für den Programmierer stellt sich daher folgende Aufgabe:

Das ist zu tun:

A) Erkennen, welches Element der Auswahlbox betätigt wird
Dies geschieht mit Hilfe von mbutton() (s.u.)
B) Umschalten des Bedienelements mit Hilfe von display()
C) Erkennen, ob der Betätigungsknopf bedient wurde.
Dies geschieht wieder mit Hilfe von mbutton() (s.u.)
D) Aktualisieren der Anzeige mit Hilfe von display()

Die Funktion

mbutton(*ID{u08},Auswahl{u08}*)

ist dabei entscheidend für die Auswertung der Auswahlbox.

Wenn die Listbox so verändert wird, dass das Element mit dem Index *Auswahl* verändert wird, gibt die Funktion den Wert 255 zurück. Wird der Betätigungsknopf gedrückt, so wird im Gegensatz dazu der Wertindex der *Auswahl* zurückgegeben. Wir wollen dies am Beispiel erklären. Zunächst müssen wir unserem Webserver das *mbutton*-Element (siehe dessen Syntax auf S. 286) hinzufügen.

Hier zu ergänzen wir unsere Konfiguration:

Ein globaler Mbutton mit 3 Elementen

```
[WebServer]
// Seite 1
page(HeizRaumPageID)[$Keller$,$Heizung$]
button(LichtID)[LIGHT]$Alle Lichter$ pbutton(SteckerID)[LIGHT]$Steckdosen$
mbutton(mSteckerID)[$1.Stecker$,$2.Steckdose$,$3.Steckdose$][SWITCH]$Multi-Schalter$
// Seite 2
page(2)[$OG$,$Schlafen$]
button(LichtID)[LIGHT]$Alle Lichter$ pbutton(SteckerID)[LIGHT]$Steckdosen$
```

Auf diese Weise haben wir das globale Webelement *mbutton* hinzugefügt. In gleicher Weise hätten wir ein lokales Element (*mpbutton*) definieren können.

Diese Variablen machen die Programmierung einfacher

Wie sprechen wir das Element an? Dazu ergänzen wir zunächst im Programm die folgenden Zeilen:

```
mSteckerID=1
mStecker1=1
mStecker2=2
mStecker3=3
mText=$AUS$
mICON=SWITCH
mState=ACTIVE
mDeko=GREY
mChoice=1
// Abfragen der Listbox
if mbutton(mSteckerID,mStecker1)==255 then {
        mChoice=1
} endif
if mbutton(mSteckerID,mStecker2)==255 then {
        mChoice=2
} endif
if mbutton(mSteckerID,mStecker3)==255 then {
        mChoice=3
} endif
// Einstellen der Anzeige
if change(mText) or change(mICON) or change(mState) or change(mDeko) or change(mChoice) then {
        display(mSteckerID,mText,mICON,mState,mDeko,mChoice)
} endif
```

Zustand 0 von mbutton() bedeutet: User hat Listbox betätigt

Einstellen der Anzeige: Falls eine Information sich ändert, display()-aufrufen. Ein Aufruf für alle Änderungen

Nun kann der Anwender bereits die Listbox betätigen und der Enertex® EibPC „merkt“ sich diese Einstellung. Was jetzt noch fehlt sind die beiden Aufgaben C) und D) der obigen Erläuterungen.

Auffällig ist, dass display einen Text mit übergibt, obwohl es doch auf S. 276 heißt:

> *„In der zweiten Zeile kann kein Text angezeigt werden.“*

Im vorliegenden Fall wird zwar kein Text angezeigt, aber das Element „stört sich“ auch nicht daran, dass display() ein weiteres Argument liefert. Ein mshifter-Element, das grundsätzlich die gleiche Optik aufweist und trotzdem doppelt so breit ist, könnte einen dynamischen Text in der 2. Zeile anzeigen. Eine Anpassung an dieses Element ist im Anwendungsprogramm also nicht weiter notwendig.

Zum Erkennen des Betätigungsfeldes müssen wir wieder die Funktion mbutton bedienen und wir ergänzen unser Programm wie folgt:

1. Steckdose liegt auf '1/5/1'b01
2. Steckdose liegt auf '1/5/2'b01
3. Steckdose liegt auf '1/5/3'b01

```
if mbutton(mSteckerID,mStecker1)==1 then {
        write('1/5/1'b01,!'1/5/1'b01)
} endif
if mbutton(mSteckerID,mStecker2)==1 then {
        write('1/5/2'b01,!'1/5/2'b01)
} endif
if mbutton(mSteckerID,mStecker3)==1 then {
        write('1/5/3'b01,!'1/5/3'b01)
} endif
```

Was jetzt noch fehlt, ist die Darstellung des Betätigungsknopfes. Diese soll ja den Zustand des Schaltaktors darstellen. Das machen wir nun wie folgt:

```
if '1/5/1'b01 and mChoice==mStecker1 then {
        mText=$EIN$;
        mDeko=GREEN;
        mState=ACTIVE
} endif
if !'1/5/1'b01 and mChoice==mStecker1 then {
        mText=$AUS$;
        mDeko=GREY;
        mState=INACTIVE
} endif
if '1/5/2'b01 and mChoice==mStecker2 then {
        mText=$EIN$;
        mDeko=GREEN;
        mState=DISPLAY
} endif
if !'1/5/2'b01 and mChoice==mStecker2 then {
        mText=$AUS$;
        mDeko=GREY;
        mState=INACTIVE
} endif
if '1/5/3'b01 and mChoice==mStecker3 then {
        mText=$EIN$;
        mDeko=GREEN;
        mState=DISPLAY
} endif
if !'1/5/3'b01 and mChoice==mStecker3 then {
        mText=$AUS$;
        mDeko=GREY;
        mState=INACTIVE
} endif
```

Damit wird klar, weswegen wir anfangs mit Variablen gearbeitet hatten. Diese machen es nun möglich das Programm einfach zu gestalten. Wird von außen geschaltet oder die Variable *mChoice* verändert, so wird die Information für die Darstellung mit display() entsprechend modifiziert, und diese Veränderung stößt seinerseits die Ausgabe auf das Webelement an.

In analoger Weise können Sie mit den Elementen *mshifter* bzw. den seitenbezogenen Varianten *mpshifter*, *mpbutton* arbeiten.

Für den Anwender kompakt

Der Anwender hat mit dem *mbutton* ein sehr kompaktes Bedienelement zur Verfügung, mit dem mehrfache Aufgaben bzw. Bedienung einer mehrfachen Auswahl erledigt werden können. Für den Programmierer stellt sich die Aufgabe diese Möglichkeiten auch entsprechend zu verarbeiten.

Der angegebene Code lässt sich aber einfach in ein Makro verpacken. So finden Sie in der Bibliothek EnertexWeb.lib die entsprechenden Mbuttons-Makros. Unser Code kann daher einfach durch den Aufruf

```
[Macros]
Mbutton3(Stecker,mSteckerID,'1/5/1'b01,'1/5/2'b01,'1/5/2'b01)
[MacroLibs]
EnertexWeb.lib
```

realisiert werden. Für die Varianten der seitenbezogenen Elemente, sowie für die Elemente mit zwei-, drei- und vierfach Auswahl sind in der Bibliothek EnertexWeb.lib entsprechende Makros zu finden.

Ein weiteres Element für die Webgestaltung ist der *pslider.* Dies ist der lokal seitenbezogenen Schieberegler bzw. der *slider* ist die globale Variante hiervon.

Mit getslider wird der Aktuelle Zustand abgefragt und mit setslider gesetzt

```
[WebServer]
// Seite 3
page(SliderPageID)[$OG$,$Wohnen$]
slider(SliderID)[LIGHT]$Alle Dimmer$ pslider(pSliderID)[LIGHT]$TV Beleuchtung$
[EibPC]
SliderPageID=3
SliderID=4
pSliderID=5
MySlider=getslider(SliderID)
if change(MySlider) then setslider(SliderID,MySlider,LIGHT,INACTIVE) endif
MypSlider=getpslider(pSliderID,SliderPageID)
if change(MypSlider) then {
        setpslider(pSliderID,MypSlider,LIGHT,INACTIVE,SliderPageID)
} endif
```

Auch hier muss der Anwender die Programmierung des s*liders* bzw. dessen Stellung vornehmen. Beim Loslassen des s*lider*-Knopfes sendet der Webserver über die Funktion

Global ....

```
getslider(ID{u08})
```

bzw.

.... Lokal

```
getpslider(ID{u08},SeitenID{u08})
```

den Zustand des Sliders als Rückgabewert vom Typ u08. Damit der *slider* diese Stellung behält muss dieser auf den gewünschten Wert mit Hilfe von

```
setslider(ID{u08},Wert{u08},Icon{u08}, Stil{u08})
```

bzw.

```
setpslider(ID{u08},Wert{u08},Icon{u08}, Stil{u08}, SeitenID{u08})
```

gesetzt werden.

Wenn Sie auf andere Seiten springen möchten oder andere Webseiten in Ihren Webseiten einbinden möchten, arbeiten Sie mit den Elementen *plink*, *link*, *frame* und *dframe,* die wie folgt zu nutzen sind - **denken Sie dabei an das Sicherheitsrisiko bei der Verwendung externer Links für Ihren Client***:*

Verlinken

```
[Webserver]
page(5)[$Diverses$,$Links$]
plink(1)[MAIL][1]$Gehe zu Seite 1$       link(2)[MAIL][$http://www.shop.enertex.de$]$Einkaufen bei Enertex$
frame [$http://www.enertex.de$]
dframe [$http://www.enertex.de$]
```

Die Konfiguration der Elemente *plink* bzw. *link* erfolgt wie bei *button* Elementen. Mit Hilfe der Funktion pdisplay() kann das Icon, der Text und der Link selbst verändert werden.

mchart und picture

Damit kommen wir zu zwei weiteren wichtigen Webelementen für die Visualisierung mit dem Enertex® EibPC, *mchart* bzw. *picture*. Das *picture* Element kann zum Einbinden externer Kamerabilder oder Wettervorhersagen genutzt werden. Das *mchart* (oder *mpchart*) ist für die Darstellung von bis zu vier verschiedenen Graphen zur Visualisierung von beliebigen Verläufen konzipiert. Mit Hilfe der Funktion pdisplay() kann der Link von *picture* zur Laufzeit verändert werden.

Das Einbinden externer Quellen wird wie folgt erreicht:

Einbinden externer Grafiken

```
page(WetterID)[$Diverses$,$Wetter$]
picture(0)[DOUBLE,CENTERGRAF]($Das Wetter$, \\
   $http://de.weather.yahoo.com/images/eur_germany_outlook_DE_DE_440_dmy_y.jpg$) \\
picture(1)[DOUBLE,CENTERGRAF]($Satellitenbild$ ,\\
  $http://de.weather.yahoo.com/images/eur_satintl_440_dmy_y.jpg$)
```

Die genaue Syntax zum *mchart* bzw. *mpchart*, insbesondere den Darstellungsoptionen, finden Sie auf S. 289. Die Darstellung von XY-Diagrammen ist dabei wie folgt zu konfigurieren:

```
mpchart(VorlaufID)[DOUBLE,XY]($Temperatur$,LINE,$Vorlauf$,LINEDOTS,\\
$AußenTemp$,COLUMN)  \\
mpchart(AussenID)[DOUBLE,XY]($Temperatur$,LINE)
```

Die erste Anweisung definiert ein *chart*, das bis zu drei Graphenverläufe aufnimmt, die zweite Anweisung hingegen nur einen einzigen Verlauf.

Um diese im Anwendungsprogramm alle 15 Minuten mit Daten zu füllen, gehen wir wie folgt vor:

Darstellung XY-Diagramme

```
[EibPC]
VorlaufID=0
AussenID=1

//Zeitanzeige als X-Achse im1/4 Stunden-Raster
Time=convert(hour(),0.0)+convert(minute(),0.0)/60.0
if mtime(15,0) then mpchart(VorlaufID,Time,'2/3/5'f16,WetterID) endif
if mtime(30,0) then mpchart(VorlaufID,Time,'2/3/4'f16,WetterID) endif
if mtime(45,0) then mpchart(AussenID,Time,'2/3/4'f16,WetterID) endif
if mtime(00,0) then mpchart(AussenID,Time,'2/3/4'f16,WetterID) endif
```

Gratulation. Sie sind am Ende dieser umfassenden Einführung angekommen und sollten nun selbst in der Lage sein, mit dem Enertex® EibPC zu arbeiten.

Visualisierungsassistent

Ab EibStudio® V2.200 können Sie auch mit dem Visualisierungsassistenten Webseiten erstellen.

Ausführliches Webserver-Beispiel

Im folgenden soll eine vollständige Visualisierung eines Gebäudes erstellt werden. Diese Visualisierung ist im Foyer des Enertex-Gebäudes auf einem 27-Zoll HP – Android Panel aktiv.

Die vorliegende Visualisierung ist im Rahmen eines Praktikums entstanden und erklärt die wesentlichen Punkte bei der Programmierung und Erstellung der Webseiten.

Im Eingangsbereich bei der Enertex® Bayern GmbH

Abbildung 61: HP Android basierte Visualisierung

Ziel ist es,Statusanzeige mit einer zentralen Schaltmöglichkeit der Haussteuerung, sowie Wetterinformationen auf mehreren Seiten zu visualisieren.

Beginnen wir mit der Hauptseite, welche am Ende wie folgt dargestellt wird:

Hauptseite der Visualisierung

Abbildung 62: Hauptseite der Visualisierung

Zu Beginn wollen wir unseren Aufbau des im EibStudio einteilen, diese Einteilung ist für die Funktionalität der Anweisungen notwendig. Grundsätzlich gibt es verschiedene Sektionen die mit „[ ]“ gekennzeichnet sind. In diesen Sektionen werden wir Unterpunkte einführen, bei denen Schritt für Schritt mit auskommentierten (//) Beschreibungen, Variablen definiert, Webserver-Seiten hinzugefügt oder Anweisungen vergeben werden.

```
[MacroLibs]
/Bibliotheken/InternEnertexWebV3.lib
[ETS-ESF]
//
[WebServer]
//
[Macros]
//
[EibPC]
//// Deklarationen
//// Funktionen
```

Um nun unsere erste Seite im *EibPC* anzulegen, wird eine *SeitenID* benötigt. Diese sollte möglichst die Darstellungskomponente der Seite Spiegeln und wird in die Sektion [EibPC] geschrieben.

```
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
```

Anfänglich muss man sich neben Namen (*Wetterseite*) und Kategorie (*Allgemein*) der Hauptseite, ebenso im Klaren sein, in welche Designrichtung der Webserver gehen soll. Wir entschieden uns für die Designvariante *black* und als optische Abrundung wurde eine Hintergrundgrafik implementiert, welche das Firmenlogo darstellt (vgl. Abbildung 62). **Achtung**: „ Hintergrundgrafiken werden nicht skaliert“. Bei einer Anordnung von 6 x 8 Elementen beträgt die perfekte Skalierung 1025x801 Pixel. Die Grafik muss dabei so angeordnet werden, dass die am Ende in der Visu freibleibende Fläche in der Mitte der Seite mit dem gewünschten Firmenlogo steht.

Ist das Hintergrundbild soweit bearbeitet, kann man es in die Sektion [WebServer] einbinden:

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]

[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
```

Damit der EibPC die Hintergrundgrafik auch unter dem angegebenen Pfad findet, muss diese exportiert werden. Dies geschieht mit Hilfe des *EibStudio* unter dem Reiter *EibPC* und dem Menüpunkt *Dateitransfer Dialog.*

Bilder hochladen

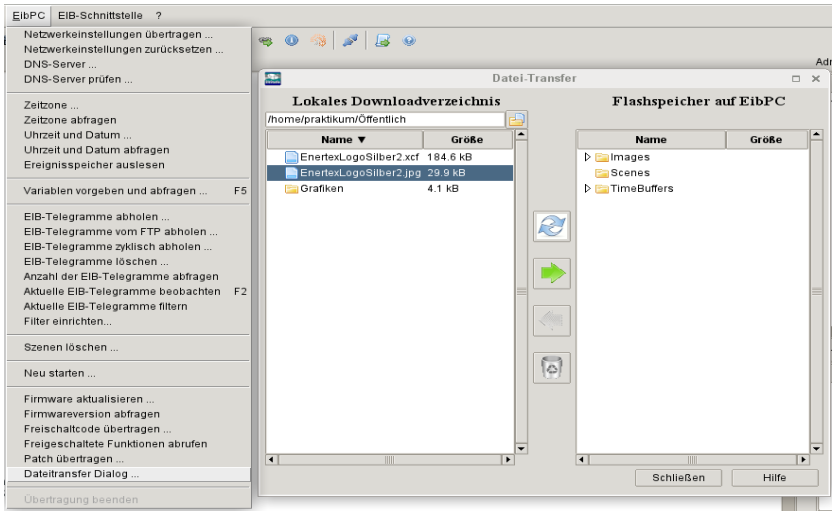


Abbildung 63: Dateitransferdialog

Mit der Möglichkeit, eigene Bilder in den Enertex EibPC zu laden, können Sie diese auch zur Gestaltung einer eigenen Anzeige für die Header und Footer Grafik nutzen. Zunächst müssen Sie die Grafiken wie beschrieben vorab in den Enertex® EibPC hochladen.

In unserem Beispiel setzten wir header und footer jeweils auf (0) und verzichteten somit auf Kopf- und Fußzeile, sowie der Möglichkeit, eigene Bilder dafür zu verwenden.

Nachdem die Designarbeiten soweit abgeschlossen sind, geht es an die Implementierung der Anzeigebutton. Diese können je nach belieben mit Gruppenadressen und Icons (Übersicht s. Seite 300) versehen werden. Viele der Button arbeiten mit vorgefertigten Makros, welche in den Enertex Bibliotheken vorhanden sind.

Grundsätzlich unterscheidet man **lokale** und **globale** Elemente bei der Anzeige. Global bedeutet, dass das Element auf verschiedenen Seiten verwendet werden kann, aber nur ein einziges mal erzeugt wurde. Ein Zugriff bzw. eine Veränderung des Elements durch das Anwendungsprogramm ist daher für alle Seiten gleich. Hingegen kann ein lokales Element nur in einer Seite verändert werden. Vom Design sind lokale und globale Elemente identisch, wobei erstere durch den Zusatz „p“ (page) gekennzeichnet sind.

Einige der Elemente (*z.B **shifter**(ClockID)[CLOCK] $Uhrzeit$* und ***shifter**(DateID)[DATE]$Datum$)* wurden auf unserer Hauptseite als Global initialisiert, da diese als durchgängige Informationen auf allen Seiten dienen sollen. Für diese Button müssen Webelement IDs initialisiert werden, welche ebenso in die Sektion [EibPC] geschrieben werden.

```
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
//Initialisierung der Webelement IDs
WindID = 1
WolkenID = 2
ClockID = 3
DateID = 4
```

Um im späteren Verlauf schneller arbeiten zu können empfiehlt es sich, funktionsorientierte ID Bezeichnungen zu wählen.

Die lokalen Elemente werden direkt in der Webserver-Konfiguration mit ID´s von (0 bis 39) erstellt, sind aber ebenso wie die globalen Elemente vorerst ohne Funktionalität.

Der Unterschied zwischen Button und Shifter ist rein die Größe der Elemente. Während ein Button einfache Breite besitzt, wird der Shifter mit doppelter Breite dargestellt.

Konfiguriert werden diese Elemente wie folgt:

shifter(ID)[Grafik1,Grafik2,Grafik3,Grafik4]$Text$	button(ID)[Grafik] $Text$
pshifter(ID)[Grafik1,Grafik2,Grafik3,Grafik4]$Text$	**p**button(ID)[Grafik] $Text$

Grafik 2 bis Grafik 4 sind optional

Nun wissen wir, wie die Elemente in die Sektion [WebServer] auf unserer Seite eingebunden werden. Anordnen lassen sie sich auf verschiedene Art und Weise als Schachbrettmuster. Durch die feste Vorgabe von eingebauten Icons, Längen und Variablen kann die Konfiguration ohne grafischen Aufwand rein textbasiert erfolgen.

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]
header(0)
footer(0)
// Erste Zeile
button(WINDID)[WIND]$Wind in km/h$ \\
  pbutton(31)[TEMPERATURE]$Außentemperatur$\\
  pshifter(17)[TEMPERATURE,TEMPERATURE]$Aussentemp.: Min und Max in °C$\\
  pbutton(5)[OKCIRCLE]$Status HG$
// Zweite Zeile
pbutton(15)[WEATHER]$Licht in Lux$ \\
  button(WolkenID)[WEATHER]$Aktuelles Wetter$\\
  pshifter(14)[WEATHER,NIGHT]$Sonnenauf- und Sonnenuntergang$\\
  pbutton(6)[OKCIRCLE]$Status NG$
// Dritte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$\\
  pshifter(13)[INFO]$Online$\\
  shifter(DateID)[DATE]$Datum$
```

Kompilieren wir unser bisheriges Programm, so ergibt sich folgende Darstellung im Webserver.

Abbildung 64: Elemente im Schachbrettmuster

Die Anordnung der Elemente wird so dargestellt wie gewollt. Allerdings ist unser Logo auf der Hintergrundgrafik nicht zu erkennen, da es teilweise von den Elementen verdeckt wird, wir es aber auch für eine Darstellung von 6x8 Feldern mittig skaliert haben.

Um noch etwas fürs Auge zu schaffen, aber auch um mehr Informationen zu bekommen, binden wir zwei grafische Wettervorhersagen mit ein. Diese sollen unsere Hintergrundgrafik seitlich einrahmen und über der geschaffenen globalen Fußzeile aus Uhrzeit und Datum stehen. Zur Einbindung verwenden wir folgende Funktion:

picture(ID)[Höhe,Typ]($Beschriftung$,$www-LINK$)

Bei den ausgewählten Links ist möglichst darauf zu achten, dass diese unabhängig von Zeit und Datum sind. So aktualisieren die Seiten sich selbständig und es benötigt keinen weiteren Code, um einen refresh zu erzeugen.

Beispiele:

Gut:

$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-tt_webL.jpg$

Schlecht:

$http://www.wetteronline.de/?
daytime=day&diagram=true&fcdatstr=**20140916**&iid=DL&pid=p_city_local&sid=Pictogram$

Nach Auswahl der beiden Links, müssen wir unsere picture Funktionen ebenfalls in der Sektion [WebServer] einbinden.Um die seitliche Einrahmung des Logos zu erreichen, müssen zwischen den picture Funktionen zwei **none** Elemente eingesetzt werden. Diese werden im Webserver als Platzhalter verwendet und in keiner Weiße dargestellt.

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]
header(0)
footer(0)
// Erste Zeile
button(WINDID)[WIND]$Wind in km/h$ \\
  pbutton(1)[TEMPERATURE]$Außentemperatur$\\
  pshifter(2)[TEMPERATURE,TEMPERATURE]$Aussentemp.: Min und Max in °C$\\
  pbutton(3)[OKCIRCLE]$Status HG$
// Zweite Zeile
pbutton(4)[WEATHER]$Licht in Lux$ \\
  button(WolkenID)[WEATHER]$Aktuelles Wetter$\\
  pshifter(5)[WEATHER,NIGHT]$Sonnenauf- und Sonnenuntergang$\\
  pbutton(6)[OKCIRCLE]$Status NG$
// Dritte Zeile
picture(7)[DOUBLE,CENTERGRAF]($Durchschnittliche Tagestemperaturen$,
$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-tt_webL.jpg$) \\
  none \\
  none \\
  picture(8)[DOUBLE,CENTERGRAF]($3 – Tagesvorhersage$,
$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-3t_webL.jpg$)
// Vierte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$\\
  pshifter(9)[INFO]$Online$\\
  shifter(DateID)[DATE]$Datum$
```

Abbildung 65: Die Konfiguration der Buttons

Die Anordnung unserer Anzeige Button ist somit vorerst abgeschlossen und wir können uns der Funktionalität widmen. Dazu müssen wir im Code in der Sektion [Macros] abreiten. Hier lässt sich mit Hilfe der **Makro-Bibliothek(*InternEnertexWebV3*)** für jeden Button/Shifter die passende Funktionalität finden.

Beispiel Datumsanzeige global:

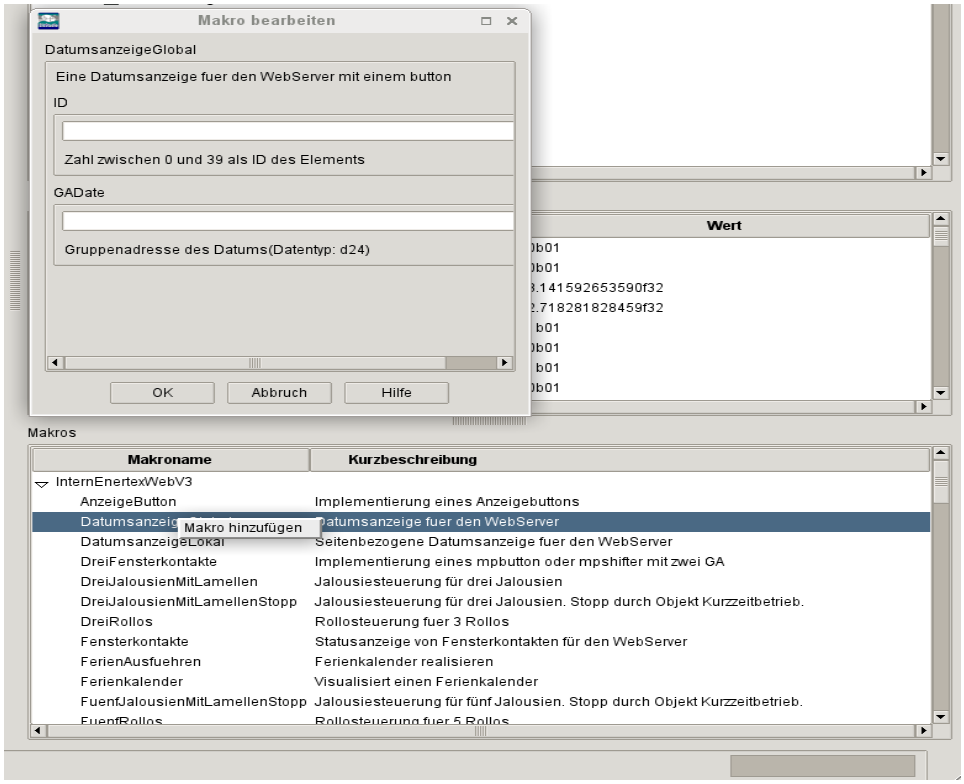


Abbildung 66: Makroauswahl

Wird für jeden Button/Shifter das richtige Macro angewendet und die jeweiligen Felder entsprechend ausgefüllt, so entstehen in der Sektion [Macros] folgende Zeilen

```
[Macros]
// Makros für die Wetterseite in der Kategorie Allgemein:
DatumsanzeigeGlobal(DateID,"Datum-7/0/0")
UhrzeitanzeigeGlobal(ClockID,"Uhr-7/0/1")
WindAnzeigeButtonGlobal(WINDID,"Wind-14/6/0")
MinMaxTemperaturAnzeigeButtonLokal(2,AllgemeinWetterseiteID,"Außentemperatur-14/6/3")
LichtAnzeigeButtonLokal(4,AllgemeinWetterseiteID,"Licht-14/6/4")
SonnenAufUntergangAnzeigeShifterLokal(5, AllgemeinWetterseiteID)
OnlineAnzeigeButtonLokal(9,AllgemeinWetterseiteID)
```

Nun ist allen Anzeige Button eine Funktion zugeteilt, bis auf

```
pbutton(1)[TEMPERATURE]$Außentemperatur$
pbutton(3)[OKCIRCLE]$Status HG$
pbutton(6)[OKCIRCLE]$Status NG$
```

Die ID´s (3) und (6) werden erst im späteren Verlauf belegt. Der *pbutton(1)* jedoch, soll ein dynamisches Icon erhalten, das sich je nach Außentemperatur einfärbt und diese auch anzeigt. So soll blau für kalt, grau für mäßig, dunkel rot für angenehm warm und rot für heiß stehen.

Dies realisieren wir durch die Funktion:

```
pdisplay(ID, Text, Icon, State, TextStil, PageID, [Mbutton])
```

Hiermit werden *pbutton* oder *pshifter* angesprochen

Zuerst muss die Unterteilung der verschiedenen Zustände angelegt werden. Wir entschieden uns für:

```
>= 27 °C = heiß
< 27 °C und >= 18 °C = angenehm warm
< 18 °C und >= 5 °C = mäßig
< 5 °C = kalt
```

Je nach Außentemperatursensor kann der Rückgabewert des Sensors variieren. Unsere Wetterstation liefert den Datenwert **f16** zurück. Somit gilt: 0f16 = 0°C

Die Zustandsanweisungen müssen nun noch in if-Anweisungen im Sektor [EibPC] verpackt werden und mit pdisplay übergeben werden.

```
[EibPC]
//// Funktionen
/// Temperaturanzeige dynamisch
if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3">=27f16 then
pdisplay(1,"Außentemperatur-14/6/3",TEMPERATURE,BRIGHTRED,BLINKRED,AllgemeinWetterseiteID)
endif


if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3">=18f16 and "Außentemperatur-14/6/3"
<27f16  then pdisplay(1,"Außentemperatur-
14/6/3",TEMPERATURE,DARKRED,GREEN,AllgemeinWetterseiteID) endif


if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3">=5f16 and "Außentemperatur-14/6/3"
<18f16  then pdisplay(1,"Außentemperatur-
14/6/3",TEMPERATURE,DISPLAY,GREY,AllgemeinWetterseiteID) endif


if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3" < 5f16  then
pdisplay(1,"Außentemperatur-14/6/3",TEMPERATURE,ACTIVE,BLINKBLUE,AllgemeinWetterseiteID) endif
```

Nun „weiß“ der *pbutton(1)* wie er sein Icon bei der entsprechenden Außentemperatur darstellen soll.

Um auch der letzten Anzeige eine Funktionalität zu geben fehlt uns folgender Button:

```
button(WolkenID)[WEATHER]$Aktuelles Wetter$
```

Hier wollen wir eine Sonneneinstrahlungsabhängige (*Luxwert*) Anzeige der aktuellen Bewölkung implementieren. Es wird unterschieden in Sommerzeit (Monat Mai bis Monat September) und Winterzeit (Monat Oktober bis Monat März), da der Sonnen-Höhenwinkel (Elevation) und somit die Intensität der Einstrahlung variiert. Sollte es regnen, wird eine Regenmeldung ausgegeben.

Nun muss auch hier eine Einteilung der verschiedenen Zustände angelegt werden. Wir entschieden uns für folgende Einteilung:

Sommerzeit:
Einstrahlung <= 20000 Lux und dies länger als 1 Minute, dann bedeckt
20000 Lux < Einstrahlung <= 45000 Lux und dies länger als 1 Minute, dann wolkig
45000 Lux < Einstrahlung <= 70000 Lux und dies länger als 1 Minute, dann sonnig
70000 Lux < Einstrahlung und dies länger als 1 Minute, dann starker Sonnenschein
Winterzeit:
Einstrahlung <= 3500 Lux und dies länger als 1 Minute, dann bedeckt
3500 Lux < Einstrahlung <= 9000 Lux und dies länger als 1 Minute, dann wolkig
9000 Lux < Einstrahlung <= 15000 Lux und dies länger als 1 Minute, dann sonnig
15000 Lux < Einstrahlung und dies länger als 1 Minute, dann starker Sonnenschein

Achtung:
Sommerzeit und Winterzeit ergeben zwei Bedingungen in der if-Anweisung. Somit muss Sommerzeit und Winterzeit jeweils als Variable definiert werden. Da die Variablen entweder wahr oder falsch bzw. EIN oder AUS sein können definieren wir einen Zeitraum mit Hilfe der month(dd,mm) Funktion. Dies geschieht ebenfalls in der Sektion [EibPC] unterhalb der Webelement ID´s.

```
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
//Initialisierung der Webelement IDs
WINDID = 1
WolkenID = 2
ClockID = 3
DateID = 4
//Variablen
Sommerzeit=month(01,05) and !month(30,09)
Winterzeit =month(01,10) and !month(30,04)
```

Nun realisieren wir die Anzeige mit den Funktionen:

webdisplay(ID,Text,Icon,Zustand,TextStil)
delay(Signal, Zeit)

Tragen wir nun unsere Bedingungen als if-Anweisung in die Sektion [EibPC] ein, ergibt sich folgender Code:

```
[EibPC]
//// Funktionen
/// Wetter
//Sommerzeit
if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==EIN) then webdisplay(WolkenID, $Es
regnet$,WEATHER,STATE6,GREY) endif

if( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS and delay("Licht-14/6/4" <= 20000f16,60000u64))
then webdisplay(WolkenID,$bedeckt$,WEATHER,STATE5,GREY) endif

if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 20000f16 and delay("Licht-
14/6/4" <= 45000f16,60000u64)) then webdisplay(WolkenID, $wolkig$,WEATHER,STATE4,GREY) endif

if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 45000f16 and delay("Licht-
14/6/4" <= 70000f16,60000u64)) then webdisplay(WolkenID, $sonnig$,WEATHER,DISPLAY,GREY) endif

if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS  and  delay("Licht-14/6/4" > 70000f16,60000u64))
then webdisplay(WolkenID, $starker Sonnenschein$,WEATHER,BRIGHTRED,BLINKRED) endif
//Winterzeit
if (Winterzeit == EIN and "Außentemperatur-14/6/3" < 0.0 and  "Regenmeldung-14/6/1"==EIN) then
webdisplay(WolkenID, $Es schneit$,ICE,ACTIVE,BLINKBLUE) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and delay("Licht-14/6/4" <= 3500f16,60000u64))
then webdisplay(WolkenID,$bedeckt$,WEATHER,STATE5,GREY) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 3500f16 and delay("Licht-
14/6/4" <= 9000f16,60000u64)) then webdisplay(WolkenID, $wolkig$,WEATHER,STATE4,GREY) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 9000f16 and delay("Licht-
14/6/4" <= 15000f16,60000u64)) then webdisplay(WolkenID, $sonnig$,WEATHER,DISPLAY,GREY) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and delay("Licht-14/6/4" >15000f16,60000u64)) then
webdisplay(WolkenID, $starker Sonnenschein$,WEATHER,BRIGHTRED,BLINKRED) endif
```

Um die Seite zu vollenden, benötigen wir erst unsere zwei weiteren Seiten von Status Hauptgebäude und Status Nebengebäude.Diese Seiten sollen als zentrale Schaltmöglichkeiten dienen und anzeigen,ob und wenn ja in welcher Etage noch Fenster offen sind.Zudem werden hier Weiterleitungen in die entsprechende Etage eingebaut werden.

Diese Weiterleitungen fehlen uns ebenso auf der Hauptseite. Im Falle eines offenen Fensters oder einer leuchtenden Lampe, in einem beliebigen Raum im Gebäude, soll der jeweilige Status Button (*Status HG*) auf rot wechseln und aus dem √ ein **X** machen.

Wir binden also zwei Verlinkungen in Form von plink(ID)[Grafik] [PageSprungIndex] $Text$ in den Webserver mit ein, die auf (*Status HG)* und (*Status NG*) weiterleiten sollen.

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]
header(0)
footer(0)
// Erste Zeile
button(WINDID)[WIND]$Wind in km/h$ \\
  pbutton(1)[TEMPERATURE]$Außentemperatur$\\
  pshifter(2)[TEMPERATURE,TEMPERATURE]$Aussentemp.: Min und Max in °C$\\
  plink(10)[RIGHT][StatusHGID]$Zur Statusseite Hauptgebäude$\\
  pbutton(3)[OKCIRCLE]$Status HG$
// Zweite Zeile
pbutton(4)[WEATHER]$Licht in Lux$ \\
  button(WolkenID)[WEATHER]$Aktuelles Wetter$\\
  pshifter(5)[WEATHER,NIGHT]$Sonnenauf- und Sonnenuntergang$\\
  plink(11)[RIGHT][StatusNGID]$Zur Statusseite Nebengebäude$\\
  pbutton(6)[OKCIRCLE]$Status NG$
// Dritte Zeile
picture(7)[DOUBLE,CENTERGRAF]($Durchschnittliche Tagestemperaturen$,
$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-tt_webL.jpg$) \\
  none \\
  none \\
  picture(8)[DOUBLE,CENTERGRAF]($3 – Tagesvorhersage$,$http://wetter.tagesschau.de/import/wetter-
cms/vorhersagen/img/de-vs-3t_webL.jpg$)
// Vierte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$\\
  pshifter(9)[INFO]$Online$\\
  shifter(DateID)[DATE]$Datum$

page(StatusHGID)[$Allgemein$,$Statusseite Hauptgebäude$]
design $black$
header(0)
footer(0)

page(StatusNGID)[$Allgemein$,$Statusseite Nebengebäude$]
design $black$
header(0)
footer(0)

[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
StatusHGID = 2
StatusNGID = 3
```

Die Verlinkung der neuen Seiten funktioniert nun bereits. Um diese optisch noch etwas zu verbessern verwenden wir in der [EibPC] Sektion die Funktion:

plink(ID, Text, Icon, Iconzustand, PageID, PageSprungIndex)

Damit lässt sich der Status des Icon von plink verändern.

```
[EibPC]

////Funktionen
// plinks Hauptseite
plink(10,$Zur Statusseite Hauptgebäude$,RIGHT,ACTIVE,AllgemeinWetterseiteID,StatusHGID)
plink(11,$Zur Statusseite Nebengebäude$,RIGHT,ACTIVE,AllgemeinWetterseiteID,StatusNGID)
```

Nachdem alles in die verschiedenen Sektionen geschrieben wurde und wir den Code erfolgreich kompiliert haben, ist unsere Webserver-Hauptseite wie in Abbildung 62 gestaltet.

Die Statusseite des Hauptgebäudes wird nun erstellt. Sie soll Ausschaltmöglichkeiten für alle Lichter in einer Etage sowie des ganzen Hauptgebäudes beinhalten und anzeigen, ob ein Fenster in der jeweiligen Etage geöffnet ist.

Statusseite des Hauptgebäudes:

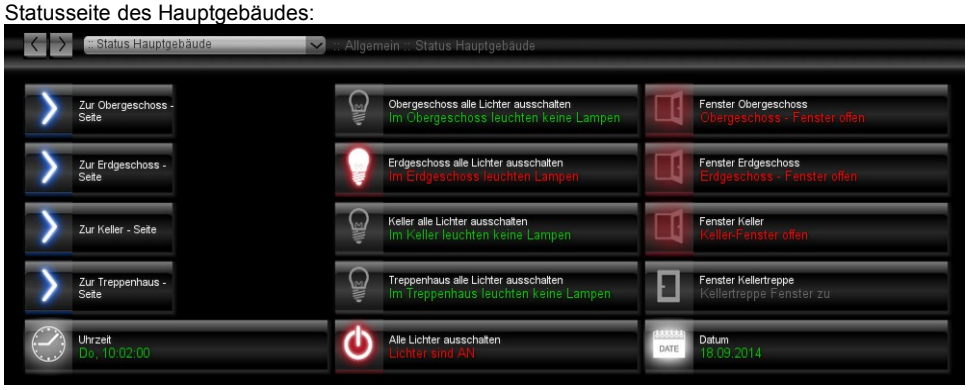


Abbildung 67: Statusanzeige mit Sprungzielen

Um wieder schneller in die Einzelnen Etagen zu gelangen, binden wir wieder plinks ein.Dort werden nun die einzelnen Räume aufgeführt und sind somit separat ansteuerbar. Sobald nun in einem Raum eine Lampe an ist, wird auf der Statusseite des Hauptgebäudes ein Lampensymbol leuchtend rot und der passende Text erscheint in der Anzeige. Betätigt man nun den Shifter, so werden alle Lampen der Etage ausgeschaltet. Will man dies bei mehreren oder allen Etagen mit nur einem Klick durchführen, so lässt sich das durch die Betätigung des Hauptschalters in der untersten Zeile, zwischen Uhrzeit- und Datumsanzeige, realisieren.

Die Seite wird nun wieder Schritt für Schritt aufgebaut und beginnt wie unsere Hauptseite mit der der Einbindung der Shifter und plinks. Darauf erhalten diese ihre Funktionalität und Designakzente.Um die Funktionalität jedoch zu gewährleisten, sollten Sie **davor** Ihre verschiedenen **Etagenseiten** mit allen Lichtern und Fenstern aufbauen. Dies führt zu einer besseren Übersicht der Gruppenadressen und erleichtert die Fehlersuche, falls bei den if-Anweisungen etwas schief läuft.

Beispiel Etagenseite:

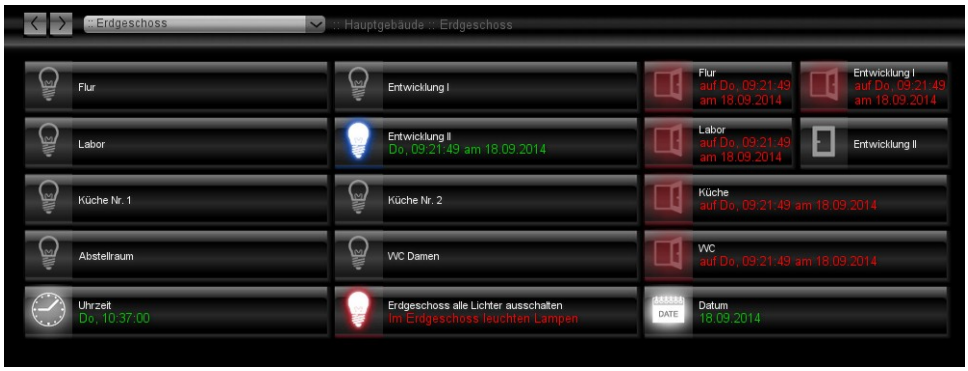


Abbildung 68: Detaillierte Anzeigen

Ist dies abgeschlossen können Sie sich wieder der Statusseite zuwenden.

Wie schon erwähnt erstellen wir zuerst unser Grundgerüst mit Button und Shifter. Die Etagenschalter haben wir als Global implementiert, da wir diese auf die Statusseite und Etagenseite einbinden. Das Beispiel behandelt nur den Hauptschalter des Erdgeschosses, die anderen können eins zu eins assoziiert werden.

```
[Webserver]
//Webelemente für Status Hauptgebäude
page(StatusID)[$Allgemein$,$Status Hauptgebäude$]
design $black$
header(0)
footer(0)
// Erste Zeile
plink(5)[RIGHT][ObergeschossID]$Zur Obergeschoss – Seite$
  none
  shifter(HauptschalterObergeschoss)[LIGHT]$Obergeschoss alle Lichter ausschalten$
  pshifter(11)[WINDOW]$Fenster Obergeschoss$
// Zweite Zeile
plink(6)[RIGHT][ErdgeschossID]$Zur Erdgeschoss-Seite$
  none
  shifter(HauptschalterErdgeschoss)[LIGHT]$Erdgeschoss alle Lichter ausschalten$
  pshifter(10)[WINDOW]$Fenster Erdgeschoss$
// Dritte Zeile
plink(7)[RIGHT][KellerID]$Zur Keller – Seite$
  none shifter(HauptschalterKellergeschoss)[LIGHT]$Keller alle Lichter ausschalten$ pshifter(9)[WINDOW]
$Fenster Keller$
// Vierte Zeile
plink(15)[RIGHT][TreppenhausID]$Zur Treppenhaus-Seite$
  none
  shifter(HauptschalterTreppenhaus)[SWITCH]$Treppenhaus alle Lichter ausschalten$
  pshifter(31)[WINDOW]$Fenster Kellertreppe$
//Fünfte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$
  pshifter(20)[SWITCH]$Alle Lichter ausschalten$
  shifter(DateID)[DATE]$Datum$
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
//Initialisierung der Webelement IDs
WINDID = 1
WolkenID = 2
ClockID = 3
DateID = 4
HauptschalterErdgeschoss = 5
```

Zur Funktion der Etagenschalter:

Prinzipiell besteht jeder Etagenschalter aus einer Variable die *wahr* oder *falsch* sein kann (Typ b01). Die Variable wird durch die Status-Gruppenadressen der Lampen geschrieben und ist wahr, sobald in einem Raum ein Licht an ist. Beispiel :

> *Wenn (Status_Raum1 == AUS oder Status_Raum2 == AUS oder Status_Raum 3 == AUS) dann soll die Variable Licht_Erdgeschoss auf EIN gesetzt werden, ansonsten auf EIN stehen.*

Im EibStudio wird dies dann für alle gewünschten Etagen wie folgt umgesetzt:

```
[EibPC]
//// Deklarationen
//Variablen
EGAlleLichter = 0b01


//// Funktionen
//Erdgeschoss Alle Lichter
if("Labor_1_Status-0/1/18" == AUS and "Büro_2_Status-0/1/19" == AUS and "Konstruktion_3_Status-0/1/20"
== AUS and "Konstruktion_4_Status-0/1/21" == AUS and "Flur_5_Status-0/1/22" == AUS and
"Küche_6_Status-0/1/23" == AUS and "Küche_7_Status-0/1/24" == AUS and "Abstellraum 8_Status-0/1/25"
== AUS and "WCDamen_9_Status-0/1/26" == AUS) then EGAlleLichter = AUS else EGAlleLichter = EIN
endif
```

Damit in unserem Webserver die Anzeige auch zu dem Wert der Variable passt, passen wir unseren Shifter über die Funktion webdisplay(ID, Text, Icon, State, TextStil,[Mbutton]) an.

```
[EibPC]
////Funktionen
/// Webdisplay Lichter globale Variablen
// EG
if EGAlleLichter == EIN then webdisplay (HauptschalterErdgeschoss,$Im Erdgeschoss leuchten
Lampen$,LIGHT,BRIGHTRED,BLINKRED) else webdisplay(HauptschalterErdgeschoss,$Im Erdgeschoss
leuchten keine Lampen$,LIGHT,INACTIVE,GREEN) endif
```

Damit die Ausführung des Shifters nun auch zum gewünschten Effekt führt, dem Ausschalten aller Lichter der Etage, muss unser Shifter noch mit einem fertigen Macro zugewiesen werden. Das Macro findet sich unter folgender Bezeichnung:

UmschaltShifterAUSGlobal

```
[Macros]
UmschaltShifterAUSGlobal(HauptschalterErdgeschoss,"EGalleLichter-
0/5/1",EGAlleLichter,LIGHT)
```

Wurde das Macro erfolgeich eingebunden, schalten sich nun alle Lichter der gewünschten Etage aus und die Icondarstellung des Shifters verändert sich von rot auf grau mit grüner Statusschrift.

Diese Prozedur führt man mit neuen Variablen ebenso für die Fensterkontakte aus. Der Unterschied zu den Lichtern besteht in der Ausführung als lokal und in der Funktionalität der Shifter. Wenn Sie keine Fenster haben, die über einen Stellmotor geschlossen werden können, entfällt die Einbindung eines Macros am Ende. Der Shifter dient also so nur als Anzeige.

```
[EibPC]
//// Deklarationen
//Variablen
EG_FensterOffen = 0b01
////Funktionen
//Erdgeschoss Fenster
if "Büro2+3Fensterkontakt-4/1/0" or"Konstruktion4+5Fensterkontakt-4/1/2" or "Küche7+8Fensterkontakt-4/1/4"
or "WCDamen10Fensterkontakt-4/1/6" or "Labor1Fensterkontakt-4/1/7" or "Flur6+9Fensterkontakt-4/1/8" then
EG_FensterOffen = 1b01 endif
/// pdisplay Fenster lokale Variablen
//EG
if EG_FensterOffen then pdisplay(10,$Erdgeschoss - Fenster
offen$,WINDOW,BRIGHTRED,BLINKRED,StatusID) else pdisplay(10,$Fenster
zu$,WINDOW,INACTIVE,GREY,StatusID) endif
```

Um alle Etagen bzw. das komplette Hauptgebäude schalten zu können, muss in der ETS eine Gruppenadresse angelegt werden, die alle gewünschten Lampen schaltet. Diese bindet man dann in das Macro *UmschaltShifterAUSLokal* ein. Nun kann man mit einem Klick das Gebäude schalten. Um das Icon anzupassen verwenden wir *pdisplay* und stellen es je nach Zustand dar.

```
[EibPC]
//// Deklarationen
//Variablen
AlleLichterHG = 0b01
////Funktionen
/// pdisplay Lichter lokale Variablen
//Hauptschalter
if AlleLichterHG == EIN  then pdisplay(20,$Lichter sind
AN$,SWITCH,BRIGHTRED,BLINKRED,StatusID) else pdisplay(20, $Lichter sind
AUS$,SWITCH,INACTIVE,GREEN,StatusID)endif
```

Damit auf unserer Hauptseite erkennbar ist, ob der Status von Lampen und Fenstern in Ordnung, beziehungsweise aus und geschlossen sind, verbinden wir die Variablen der Einzelnen Etagen (*EgAlleLichter,EG_FensterOffen,...*) mit einer neuen Status Variable *StatusHG*.

```
[EibPC]
//Variablen
StatusHG = 0b01
// if-Status
if ( Treppe == EIN or OGAlleLichter== EIN or EGAlleLichter == EIN or KGAlleLichter == EIN or
EG_FensterOffen == EIN or KG_FensterOffen ==EIN or OG_FensterOffen == EIN ) then StatusHG = EIN
else StatusHG = AUS endif


if StatusHG == EIN then pdisplay(5,$Status
kontrollieren$,CROSSCIRCLE,BRIGHTRED,BLINKRED,AllgemeinWetterseiteID) endif
if StatusHG == AUS then pdisplay(3,$Status OK$,OKCIRCLE,ACTIVE,GREEN,AllgemeinWetterseiteID) endif
```

Wird dies auch für das Nebengebäude erledigt, so erhalten wir bei jeweils einer leuchtenden Lampe der beiden Gebäude folgende Darstellung auf unserer Hauptseite:

Somit wäre unsere Zentrale Steuerseite fertig. Der Status unserer Lichter wird auf der Hauptseite angezeigt und wir haben die Möglichkeit mit nur einem Klick in auf die zentrale Schaltseite zu gelangen. Mit einem Blick erhalten wir Informationen über Außentemperatur, Bewölkung, 3-Tages Wettertrend, Windgeschwindigkeit, Datum, Uhrzeit sowie Sonnenauf- und Sonnenuntergang.

Was jetzt noch fehlt ist ein richtiger Blick in die Zukunft. Wir kreieren eine zusätzliche Wetterseite, die uns alles über die kommenden Stunden und Tage verraten wird.

```
[Webserver]
page(WetterseiteID)[$Allgemein$,$Wetterseite 2$]
design $black$
header(0)
footer(0)
```

Dazu benötigen wir eine große Datenbank von Wetterdaten.Um an diese zu gelangen ist ein Benutzerkonto auf Weather Wunderground erforderlich.

Haben Sie dies erfolgreich abgeschlossen, müssen Sie sich unter dem Reiter Key Settings einen Key generieren. Hier genügt die kostenfreie Version für Developer. Jedoch müssen Sie berücksichtigen, dass die kostenfreie Version eine begrenzte Anzahl an Anfragen hat. Diese beträgt am Tag 500 und in der Minute maximal 10 Anfragen.

Haben Sie ihren Key generiert, so müssen Sie ihn in unser Macro Wettervorhersage einbauen.

```
[Macros]
Wettervorhersage(Key,Wetterstation,Germany,after(systemstart(),1300u64) or cycle(10,00))
```

Für die Wetterstation müssen Sie ihre Stadt eintragen, oder suchen Sie nach einer passenden Station für Ihre Gegend.

Sind die Daten alle eingegeben, sehen sie nach Systemstart in der Variablenabfrage (F5) eine Liste von neuen Variablen, welche Weather Underground zur Verfügung stellt. Überprüfen Sie ob Ihre angegebene Wetterstation auch die Daten liefert, welche Sie implementieren möchten. Wenn alles passt, lassen sich die Daten nun mit Hilfe eines Button oder Shifter anzeigen.

Beispiel:

```
[Webserver]
page(WetterseiteID)[$Allgemein$,$Wetterseite 2$]
design $black$
header(0)
footer(0)


button(WINDID)[WIND]$Wind in km/h$ pbutton(14)[WEATHER]$Wetter$ pbutton(25)[WEATHER]
$Regenwahrscheinlichkeit$ pbutton(24)[RAIN]$Feuchtigkeit$  pbutton(12)[TEMPERATURE]$Aussentemp.:
Min$ pbutton(13)[TEMPERATURE]$Aussentemp.:Max$


[EibPC]
///Wounderworld Wetter
//Wettervorhersage Heute
if after(systemstart(),3000u64) or  change(Wettervorhersage_Heute_Wind_Max_Richtung) then
pdisplay(11,$aus$ + convert(Wettervorhersage_Heute_Wind_Max_Richtung,$
$),WIND,ACTIVE,BLINKBLUE,WetterseiteID) endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Temperatur_Min) then
pdisplay(12,convert(Wettervorhersage_Heute_Temperatur_Min,$$)+$
°C$,TEMPERATURE,ACTIVE,BLINKBLUE,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Temperatur_Max) then
pdisplay(13,convert(Wettervorhersage_Heute_Temperatur_Max,$$)+$
°C$,TEMPERATURE,BRIGHTRED,BLINKRED,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Wetter) then
pdisplay(14,convert(Wettervorhersage_Heute_Wetter,$$),WEATHER,STATE4,GREY,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Regenwahrscheinlichkeit) then
pdisplay(25,convert(Wettervorhersage_Heute_Regenwahrscheinlichkeit,$
$),WEATHER,STATE6,BLINKBLUE,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Feuchtigkeit) then
pdisplay(24,convert(Wettervorhersage_Heute_Feuchtigkeit,$
$),RAIN,ACTIVE,BLINKBLUE,WetterseiteID)endif
```

Mit der picture Funktion, fügen wir noch eine Windanzeige ein, welche die Windrichtungen in Deutschland anzeigt.

```
[Webserver]
picture(9)[DOUBLE,ZOOMGRAF]($Windrichtungen$,$http://wwwdyn.zdf.de/ext/weather/wind-brd-0.jpg$)
```

Daneben platzieren wir mit der Funktion timechart einen Temperaturverlauf der letzten 24 Stunden.

```
[Webserver]
mtimechart(8)[QUAD,NOAUTOSCALE,48,-10,45]( $Aussentemperaturverlauf$,LEFT, ChartBuffer1)
```

Damit der timechart weiß, welche Werte er darstellen soll, müssen wir folgende Konfigurationen vornehmen:

timebufferconfig(ChartBufferID, MemTyp, Laenge, DataTyp)

MemTyp gibt an, ob der Speicher im Ring (0) oder linear (1) beschrieben wird. Die Länge der max. Aufzeichnung der Zeitreihe wird mit Laenge (0u16 bis 65565u16) angegeben. DataTyp stellt eine repräsentative Zahl der Zeitreihe dar, z.B. 0f16 für 16-Bitzahlen oder 3% für u08 Werte.

Nun müssen die Zeitreihen noch mit Daten „befüllt“ werden. Die Funktion

timebufferadd(ChartBufferID, Daten)

erledigt diese Aufgabe.

Nachdem die Zeitreihe über einige Zeit im Enertex® EibPC aufgenommen wurde, muss sichergestellt werden, dass diese auch beim Neueinspielen des Programms oder Neustart die Werte nicht verloren gehen. Die Funktionen

timebufferstore(ChartBufferID)

timebufferread(ChartBufferID)

sind für diese Aufgabe geschaffen. timebufferstore legt die Werte des Timebuffers mit der ChartBufferID permanent in den Flashspeicherdes Enertex® EibPC ab, timebufferread liest einen abgespeicherten Puffer zurück.

```
button(ELEVATION)[WEATHER]$Höhenwinkel in °$ button(Sonnenstand)[UP]$Sonnenstand$
button(AZIMUTH)[WEATHER]$AZIMUTH$ pbutton(23)[CLOCK]$Sonnenhöchstand:$
[EibPC]
//mtimechart
Len=35040u16
Datatyp=3.3f16
MemTyp=0
timebufferconfig(ChartBuffer1, MemTyp, Len,"Außentemperatur-14/6/3" )

if mtime(0,0) or mtime(15,0) or mtime(30,0) or mtime(45,0) then
{timebufferadd(ChartBuffer1,"Außentemperatur-14/6/3")
} endif

if chtime(01,00,00) then {
timebufferstore(ChartBuffer1)
} endif

if systemstart() then {
timebufferread(ChartBuffer1)
} endif
```

Somit wäre der Temperaturverlauf korrekt angezeigt.Damit wir den timechart noch besser ausnutzen, fügen wir noch den Sonnenstandsverlauf hinzu. Dafür benötigen wir den Azimutwinkel und den Höhenwinkel der Sonne. Diese beiden Daten bekommen wir durch die Funktionen

azimuth()

Höhenwinkel : elevation()

Der Enertex® EibPC muss die geographische Länge und Breite des betreffenden Ortes kennen. Diese können im Enertex® EibStudio eingegeben werden(s.Seite.157). Um an Ihre Koordinaten und weitere Informationen zu gelangen, könnte Ihnen diese Seite hier weiterhelfen.

Den timechart und die Konfiguration müssen Sie nun wie folgt erweitern:

```
[Webserver]
mtimechart(8)[QUAD,NOAUTOSCALE,48,-10,45,-10,65]( $Aussentemperaturverlauf$,LEFT,
ChartBuffer1,$Sonnenstand$,RIGHT,ChartBuffer2
[EibPC]
Len=35040u16
Datatyp=3.3f16
MemTyp=0
timebufferconfig(ChartBuffer1, MemTyp, Len,"Außentemperatur-14/6/3" )
timebufferconfig(ChartBuffer2, MemTyp, Len,elevation() )
if mtime(0,0) or mtime(15,0) or mtime(30,0) or mtime(45,0) then
{timebufferadd(ChartBuffer1,"Außentemperatur-14/6/3");
timebufferadd(ChartBuffer2,elevation() )
} endif
```

Die beiden Winkelanzeigen, sowie einen Statusbutton des Sonnenverlaufes, der ausgibt, ob der Sonnenstand steigt oder fällt, können Sie nun auch einfügen. Dafür benötigen Sie den Azimutwinkel.

Der Sonnenhöchststand ist immer bei Azimutwinkel = 180 Grad, davor steigt der Sonnenstand. Um zu wissen wann der Sonnenhöchststand war, lassen wir uns bei Azimutwinkel 180 ° die Uhrzeit ausgeben. Somit ergibt sich folgender Code.

```
[Webserver]
button(ELEVATION)[WEATHER]$Höhenwinkel in °$ button(Sonnenstand)[UP]$Sonnenstand$
button(AZIMUTH)[WEATHER]$AZIMUTH$ pbutton(23)[CLOCK]$Sonnenhöchstand:$
[EibPC]
if change(azimuth()) then webdisplay(AZIMUTH,(convert(convert(azimuth(), 0.0 ), $$c14 ) + $ Grad $c14),
WIND, ACTIVE, GREEN) endif
if change(elevation()) then webdisplay(ELEVATION,(convert(convert(elevation(), 0.0 ), $$c14) + $ Grad$c14),
WIND, ACTIVE, GREEN) endif


if (change(elevation()) and azimuth()<175f32 and elevation()>0f32 ) then webdisplay(Sonnenstand, $steigt$,
UP, BRIGHTRED, BLINKRED) endif


if (change(elevation()) and azimuth()>175f32 and elevation()<185f32 )then webdisplay(Sonnenstand,$Sonne
im Zenit$, WEATHER, BRIGHTRED, BLINKRED) endif


if(change(elevation()) and azimuth()>185f32 and elevation()>0f32 )then webdisplay(Sonnenstand,$fällt$ ,
DOWN, ACTIVE, BLINKBLUE) endif


if (change(elevation()) and azimuth()>185f32 and elevation()<0f32 )then webdisplay(Sonnenstand,$Nacht$ ,
NIGHT, DISPLAY, GREY) endif


if azimuth()>=185f32  then pdisplay(23,settime(),CLOCK,ACTIVE,GREEN,WetterseiteID) endif
```

Als nächstes fügen wir ein Regenradar als picture ein. Wir entschieden uns für diese Version, welche für alle Bundesländer verfügbar ist. Als erstes muss man das Radar als Grafik anzeigen lassen, womit man diese Ansicht erhält. Alle fünf Minuten wird ein neues Bild aufgenommen, welches wir als eine Variable abspeichern und dann im Webserver angezeigen. Insgesamt benötigen wir also 12 Variablen. Die Konfiguration schaut somit wie folgt aus:

```
[Webserver]
 picture(4)[DOUBLE,CENTERGRAF]($RegenRadar$,$www.google.de$)
[EibPC]
URL1 = $$
URL2 = $$
URL3 = $$
URL4 = $$
URL5 = $$
URL6 = $$
URL7 = $$
URL8 = $$
URL9 = $$
URL10 = $$
URL11 = $$
URL12 = $$
if mtime(01,05) then picture(4,$Regenradar$,WetterseiteID,URL1) endif
if mtime(06,05) then picture(4,$Regenradar$,WetterseiteID,URL2) endif
if mtime(11,05) then picture(4,$Regenradar$,WetterseiteID,URL3) endif
if mtime(16,05) then picture(4,$Regenradar$,WetterseiteID,URL4) endif
if mtime(21,05) then picture(4,$Regenradar$,WetterseiteID,URL5) endif
if mtime(26,05) then picture(4,$Regenradar$,WetterseiteID,URL6) endif
if mtime(31,05) then picture(4,$Regenradar$,WetterseiteID,URL7) endif
```

```
if mtime(36,05) then picture(4,$Regenradar$,WetterseiteID,URL8) endif
if mtime(41,05) then picture(4,$Regenradar$,WetterseiteID,URL9) endif
if mtime(46,05) then picture(4,$Regenradar$,WetterseiteID,URL10) endif
if mtime(51,05) then picture(4,$Regenradar$,WetterseiteID,URL11) endif
if mtime(56,05) then picture(4,$Regenradar$,WetterseiteID,URL12) endif
```

Mit diesem Code erneuert sich alle fünf Minuten das Radarbild, allerdings fehlt noch die Belegung der einzelnen URL´s. Dazu benötigen wir die URL der Grafik, welche um 09.45 wie folgt zurückgegeben wird:

Die Fett markierten Zahlen sind für das Datum zuständig, kursiv und unterstrichen gibt die Uhrzeit an.

```
http://www.wetteronline.de/?
pid=p_radar_map&ireq=true&src=radar/vermarktung/p_radar_map/wom/2014/09/30/Intensity/
BAY/grey_flat/201409300745_BAY_Intensity.gif#1412074528673
```

Programmierung für Experten

Im folgenden Abschnitt sollen Programmcode und Beispiele erläutert werden, die tiefer in die Programmierung des Enertex® EibPCs einsteigen. Dabei werden interessante Codefragmente aus den Makrobibliotheken, sowie der Hintergrund der Vorgehensweise erläutert.

Performance

Zykluszeit

Hintergrund

Eine häufig gestellte Frage der Anwender ist: *Wie lange braucht der Enertex® EibPC bei der Verarbeitung tatsächlich*? Grundsätzlich hängt das natürlich von der Programmgröße bzw. der Art der Programmierung und den auftretenden Ereignissen ab. Durch die „Validierung“ (s. S. 125) des Programms werden pro Zyklus nur diejenigen Teile des Programms aktiv, die sich auch wirklich ändern. Daher erfolgt im Normalfall die Verarbeitung in weniger als 1 ms, bei komplexeren Programmen in wenigen ms. Die Zykluszeit wird - abhängig vom Programm - durchaus schwanken. Es ist somit auch die minimale und die maximale Verarbeitungszeit während des Betriebs des Enertex® EibPCs von Interesse.

Über das Enertex® EibStudio kann nach jedem Zyklus dem Betriebssystem des EibPCs eine Pausenzeit von bis zu 50 ms eingeräumt werden (s. S. 135), um z.B. Emails zu versenden, Webserveranfragen abzuarbeiten etc.. Die minimalen Zeitscheiben des Linux-Systems sind dabei ca. 1 ms.

Um die Verarbeitungszeit des Enertex® EibPCs zu berechnen, kann die Funktion afterc genutzt werden:

Ein Countdown – Zähler: Restzeit wird von Max an runtergezählt.

```
afterc(Variable {Typ b01}, Max{Typ u64}, Restzeit {Typ u64})
```

Diese Funktion wird wie die after-Funktion bei einem Wechsel der *Variable* (1. Argument) von AUS auf EIN getriggert: Der Rückgabewert ist nach Ablauf der vorgegebenen Zeit *Max* (2. Argument in ms) für einen Verarbeitungszyklus auf EIN. In jedem Zyklus von Beginn des Triggerimpulses von *Variable* wird dabei die *Restzeit* (3. Argument) wie ein Countdown-Zähler aktualisiert. Der Startwert von *Variable* ist *Max*. Die Änderung von *Restzeit* erfolgt immer exakt zu dem Zeitpunkt, an dem die Verarbeitung in einem Zyklus aktiv wird. Die Änderung von *Restzeit* ist also die Summe aus der eben genannten Pausenzeit plus der Verarbeitungszeit des vorangegangenen Zyklus. Damit lässt sich die Zyklusdauer berechnen, indem man mit systemstart einen afterc -Timer triggert und damit den Countdown von *Restzeit* startet, also z.B.

```
Max=10000000000000000u64
if afterc(systemstart(), Max, Restzeit) then { ..... } endif
```

Max wird dabei möglichst groß gewählt, um zu gewährleisten, dass das Ende des Countdowns möglichst nicht erreicht wird.

Mit dem Code

Berechnung der Zykluszeiten

```
MaxZyklusZeit=max(StoppZeit-Restzeit-PerformanceZeit,MaxZyklusZeit);
MinZyklusZeit=min(StoppZeit-Restzeit -PerformanceZeit,MinZyklusZeit);
```

kann damit die minimale und maximale Zykluszeit mit einer Genauigkeit von ca. ± 1ms (Zeitscheibe Linuxsystemtimer) berechnet werden.

Einen Sonderfall gibt es noch zu berücksichtigen: Bei der Initialisierung des allerersten Programmdurchlaufs müssen alle Programmteile durchlaufen werden, die dann aufgrund der Validierung später „nur bei Bedarf“ ausgewertet werden. Daher kann die erste Verarbeitungsschleife durchaus mehrere Hundert ms benötigten, wenn das Programm eine Speicherauslastung von ca. 30% erreicht. Der Start des Countdownzählers muss daher verzögert werden, wenn man die Initialisierung des Programms als Sonderfall in der Messung der Zykluszeiten nicht berücksichtigen will.

Daher verzögert man den Impuls von systemstart beim Programmstart mit einem weiteren after Timer durch eine Verschachtelung:

```
if afterc(after(systemstart(),10000u64), Max, Restzeit) then { ... } endif
```

Insgesamt schaut damit die Berechnung der Zyklusdauer wie folgt aus:

Und der gesamte Code: MaxZyklusZeit ist die maximale Dauer einer Verarbeitung, MinZyklusZeit die minimale Dauer.

```
// Berechnet die minimale und maximale Zyklusdauer
// der Verarbeitung. Dabei ist die Performance-Angabe im EibStudio immer
// als Offset dabei.

Max=1000000000000000u64
Restzeit=0u64
StoppZeit=Max
MaxZyklusZeit=0u64
MinZyklusZeit=Max
// Im EibStudio ggf. geändert, Defaultwert ist 20ms
PerformanceZeit=20u64

// Die erste Zyklus kann etwas länger dauern ...
if afterc(after(systemstart(),10000u64), Max, Restzeit) then {
      StoppZeit=0u64;
} endif
if change(Restzeit) then {
            MaxZyklusZeit=max(StoppZeit-Restzeit-PerformanceZeit,MaxZyklusZeit);
            MinZyklusZeit=min(StoppZeit-Restzeit -PerformanceZeit,MinZyklusZeit);
            StoppZeit=Restzeit;
} endif
```

Timerabbruch

Der Timer afterc nutzt das Argument Restzeit (s.o.) zur Speicherung der bereits abgelaufenen Timerzeit. Der Anwender muss daher darauf achten, dass verschiedene afterc Timer unterschiedliche Variablen zu dieser Speicherung nutzen:

Jeder Timer benötigt seine „eigene“ Restzeitvariable

```
// Zähler 1
RestZeit1=0u64
RestZeit2=0u64

if afterc(systemstart(),10000u64, Restzeit1) then {
      write('1/2/3'c14,$Timer1$c14)
} endif
if afterc(systemstart(),13000u64, Restzeit2) then {
      write('1/2/3'c14,$Timer2$c14)
} endif
```

Gleiches gilt für die Funktion

delayc(TriggerVariable {Typ b01}, Max{Typ u64}, Restzeit {Typ u64})

deren Timer – genau wie delay – durch jeden Wechsel der TriggerVariable (1. Argument) von AUS auf EIN erneut getriggert wird. Auch hier gilt, dass für Restzeit jeweils eine eigene Variable genutzt werden muss, da sich sonst die Timer gegenseitig stören.

Nach Ablauf des Timers, steht der Wert des 3. Arguments (Restzeit) auf 0u64, beim Triggern des Timers wird er auf den Wert von Max gesetzt. Wenn die Restzeit während einer aktiven Phase vom Anwender verändert, so wird damit die Ablaufzeit des Timers verändert.

Die Restzeitvariable kann vom Programm verändert werden und beeinflusst somit den Ablauf des Timers

```
RestZeit1=0u64
 if afterc(systemstart(),10000u64, RestZeit1) then {
      write('1/2/3'c14,$Timer1$c14)
} endif
if RestZeit1>1000u64 then RestZeit1=500u64 endif
RestZeit2=0u64
if delayc(systemstart(),13000u64, RestZeit2) then {
      write('1/2/3'c14,$Timer2$c14)
} endif
```

Im obigen Beispiel wird nur der afterc Timer verändert, die Restzeitvariable des delayc Timers ist unverändert.

Damit kann nun auch ein Timer angehalten werden, wenn beispielsweise das Ende des Ablaufs und die damit verbundene Aktion der if-Anweisung nicht mehr benötigt wird:

Timer

```
MyTrigger=AUS
RestZeit1=0u64
 if afterc(MyTrigger,10000u64, RestZeit1) then {
      write('1/2/3'c14,$Timer1$c14)
} endif
```

mit Abbruchbedingung =>

```
if MyTrigger== AUS then RestZeit1=0u64 endif
```

Wenn Im Beispiel *MyTrigger* auf EIN wechselt, wird der Timer gestartet, wenn *MyTrigger* vor Ablauf der Zeit auf AUS wechselt, so wird der Timer durch das Setzen von *RestZeit1=0u64* angehalten. Der then-Zweig wird dann nicht ausgeführt.

Will man den Timer vorzeitig beenden, aber den then-Zweig ausführen, so muss *RestZeit1=1u64* gesetzt werden. In diesem Fall erfolgt die Ausführung im nächsten Verarbeitungszyklus.

Warteschlange

Die ereignisbasierte Verarbeitung im EibPC erfordert die Programmierung von sogenannten „Zustandsautomaten“. Das (abstrakte) Grundprinzip eines Zustandsautomaten ist, dass die Programmierung nicht sequentiell erfolgt, sondern dass abhängig von Ereignissen die Software einen bestimmten Zustand einnimmt.

Beim Datenaustausch mit einem anderen Gerät z.B. über TCP/IP Telegrammen kann man z.B. folgende Zustände definieren:

1. Daten des anderen Teilnehmers entgegennehmen
2. Daten an den anderen Teilnehmer verschicken
3. Daten des anderen Teilnehmers zwischenspeichern
4. Daten des anderen Teilnehmers auswerten
5. Verschiedene KNX Aktionen auf dem Bus ausführen

Jeder dieser Zustände ist, zumindest grundsätzlich, von den anderen unabhängig, d.h. der EibPC muss Daten entgegennehmen, während z.B. KNX Telegramme eintreffen. Darüber hinaus können verschiedene Zustände jeweils andere „antriggern“ bzw. kann das Eintreffen eines KNX Telegramms die Datenverarbeitung anregen.

Command Fusion

Auf http://knx-user-forum.de/special/Enertex EibPC und Command Fusion findet sich eine umfangreiche Beschreibung einer Zustandsmaschine, die eine Warteschlange verarbeitet.

Anwesentheits -Zustandsmaschine

Im Supportforum entstand folgende Anforderung. Die Anwender möchten mit dem Makro `Bei_Sonnenuntergang_Gedeckelt_mitFreigabe` beim Sonnenuntergang bzw. spätestens zu einer im Makro bestimmten Zeit eine Gruppenadresse schalten.

In gleicher Weise soll das Makro `Bei_Sonnenaufgang_Gedeckelt_mitFreigabe` beim Sonnenunteraufgang genutzt werden.

```
Bei_Sonnenuntergang_Gedeckelt_mitFreigabe(Sued,FreigabeVar,"Licht Wohnen-2/2/3",AUS,22060000,22,31,00)
Bei_Sonnenaufgang_Gedeckelt_mitFreigabe(Sonnenaufgang1,FreigabeVar,"Rolläden Ost-5/2/0",RAUF,7200000,07,28,00)
```

Die Makros werden mit der Freigabe-Variable *FreigabeVar* parametriert.

Die Freigabe wird hierzu in folgende Betrachtungszeiträume unterteilt:

- Tag-Modus: Sonnenaufgang bis Sonnenuntergang
- Früh-Modus: Zeitraum nach 00:00h und noch vor Sonnenaufgang
- Spätmodus: Nach Sonnenuntergang und nicht nach 0:00 Uhr

Der Anwender betätigt eine Gruppenadresse "Anwesenheit-8/1/1" (Typ b01, EIN==anwesend).

Die Freigabe-Variable *FreigabeVar* soll abhängig von den folgenden Zuständen geschaltet werden.

Die Zustände der Realisierung

Zustand 1:

Beschreibung:

Frühmodus

Ziel:

Es soll kein Makro durchlaufen werden, unabhängig davon, ob "Anwesenheit-8/1/1" auf EIN oder AUS steht.

Damit muss *FreigabeVar* auf AUS gesetzt werden bzw. in diesem (AUS-) Zustand verharren.

Zustand 2:

Beschreibung:

Zeitraum nach Sonnenaufgang bis Sonnenuntergang

Ziel:

Wenn "Anwesenheit-8/1/1" auf EIN steht, soll *FreigabeVar* auf EIN gesetzt werden, die Makros werden also aktiv, wenn "Anwesenheit-8/1/1" auf AUS steht, soll *FreigabeVar* auf AUS gesetzt werden, die Makros werden deaktiviert.

Wenn die Gruppenadresse "Anwesenheit-8/1/1" verändert wird (Bustelegramm/User), soll der *FreigabeVar* unmittelbar deren Wert annehmen.

Zustand 3:

Beschreibung:

Spätmodus

Ziel:

Wenn "Anwesenheit-8/1/1" auf EIN steht, soll *FreigabeVar* auf EIN gesetzt werden, die Makros werden also aktiv, wenn "Anwesenheit-8/1/1" auf AUS steht, soll *FreigabeVar* auf AUS gesetzt werden, die Makros werden deaktiviert.

Dies kann nun direkt in ein Programm umgesetzt werden:

Direkte Abbildung der Zustandsmaschine im EibPC

```
FreigabeVar=AUS
TFrueh=chtime(00,00,01) and !chtime(12,00,00)
// Zustand 1: Frühmodus
if TFrueh and !sun() then FreigabeVar=AUS endif
// Zustand 2: TagModus
if sun() and change("Anwesenheit-8/1/1") then FreigabeVar="Anwesenheit-8/1/1" endif
// Zustand3   Spätmodus
if !TFrueh and !sun() then FreigabeVar="Anwesenheit-8/1/1" endif
```

TFrueh : Vormittags (ab 0:00 Uhr)

Besonders ist hier die Verwendung der Variable *TFrueh*. Diese ist über eine Verknüpfung von einer Schaltuhr um Mitternacht und einer nach Mittag realisiert. Es wird damit sichergestellt, dass *TFrueh* um 0:00 Uhr auf EIN und ab Nachmittag auf AUS steht.

Anwesenheitssimulation

In der Makrosammlung befinden sich Makros zur Anwesenheitssimulation. Das grundsätzliche Konzept dieser Makros soll im folgenden erläutert werden.

Bei einer Anwesenheitsimulation können zwei Zustände unterschieden werden:

1. Aufzeichnen

Während dieser Phase werden vorher ausgewählte Gruppenadressen aufgezeichnet. Meist sind das vom Bewohner ausgelöste Gruppentelegramme z.B. beim Betätigen von Schaltern. Die Aufzeichnung wird meistens über ein 2-Wochenintervall ausgeführt, wobei die Aufzeichnung kontinuierlich alte Werte überschreibt.

2. Abspielen

Wenn der Bewohner einer Liegenschaft z.B. in den Urlaub geht, sollen nun die Gruppentelegramme vom Enertex® EibPC ausgelöst werden, sodass Außenstehenden den Eindruck einer Anwesenheit der Bewohner vermittelt wird. Dabei soll das Abspielen tages- und uhrzeitgleich erfolgen, sodass z.B. die Aufzeichnung von einem Samstag auch wieder am Samstag abgespielt wird.

Als Konsequenz wird benötigt:

- Ermittlung der Rohdaten der Telegramme
- Ermittlung der sendenden Gruppenadresse
- Ermittlung der Zeitpunkt des Eintreffens der Telegramm
- Aufzeichnen der Daten
- Versenden von Rohdaten zeit-versetzt auf den Bus

Ermittlung der sendenden Gruppenadresse

Rohinfomationen vom Bus

Für diese Aufgabe benötigt man die Funktion readrawknx:

readrawknx(*Sim_Control {u08}, Sim_Sender{u08}, Sim_GA{u08}, Sim_IsGA{b01}, Sim_RoutingCnt {u08}, Sim_Len{u08}, Sim_Data{c1400}*)

Wenn ein beliebiges KNX-Telegramm am Bus beobachtet wird, aktualisiert die Funktion readrawknx ihre Argumente. In diesem Fall, werden die Argumente der Funktion mit Daten „gefüllt“. Die empfangenen Nutzdaten werden dann in das Argument *Sim_Data* kopiert, die Datenmenge (Bitlänge) kann mit der Variable *Sim_Len* abgefragt werden.

Bei Eintreffen eines Telegramms wird das Argument *Sim_IsGA* entsprechend gesetzt, d.h. handelt es sich um ein gewöhnliches Gruppentelegramm, so wird dieses Argument von readrawknx auf EIN gesetzt und *Sim_GA* enthält die Adresse selbst. Die Funktion readrawknx kann mit event verknüpft werden, um ein Eintreffen eines Telegramms verarbeiten zu können.

Mit den gewählten Definitionen

```
Sim_GA=0u16
Sim_IsGa=OFF
Sim_RoutingCnt=0
Sim_Len=0
Sim_Data=$$c4000
Recorder=$$c4000
Timestamp=$$c4000
```

kann man nun das Eintreffen eines Telegramms wie folgt verarbeiten:

```
if event(readrawknx(Sim_Kontroll,Sim_Sender,Sim_GA,Sim_IsGa,Sim_RoutingCnt,Sim_Len,Sim_Data))  then ....
```

Gruppenadressen identifizieren

Dabei ist zu beachten, dass hier die Gruppenadresse *Sim_GA* als 16-Bit Wert berechnet wird. Um diese Adresse mit der gewöhnlichen Schreibweise vergleichen zu können, steht die Funktion getaddress zur Verfügung: Im folgenden Beispiel

```
MeinGA=getaddress("Licht-1/2/3")
```

ist nun MeinGA der 16-Bit Wert, welcher die Gruppenadresse repräsentiert und wie diese auch in *Sim_GA* kopiert wird. Damit lässt sich nun feststellen, auf welcher Gruppenadresse das eingetroffene Telegramm gesendet wurde.

Mit Hilfe der Variablen

```
Sim_GA=OFF
```

soll im folgenden das Aufzeichnen einer eingetroffenen Nachricht getriggert werden. Dazu werden für jede aufzuzeichnende Gruppenadresse im Code if-Abfragen hinterlegt. *Sim_GA* wird wie oben beschrieben von readrawknx ermittelt.

Für jede Gruppenadresse, die aufgezeichnet werden soll, eine if-Abfrage

Code-Teil 1

```
if Sim_GA==getaddress("Heizvorlauf-0/0/1") then Sim_MyGA=ON else Sim_MyGA=OFF endif
if Sim_GA==getaddress("Temperatur-3/5/0") then Sim_MyGA=ON else Sim_MyGA=OFF endif
if Sim_GA==getaddress("Licht-1/0/29"u16) then Sim_MyGA=ON else Sim_MyGA=OFF endif
```

Die beiden Betriebsarten Aufzeichnen/Abspielen werden über

```
Sim_Play=OFF
```

realisiert. Dabei soll bei *Sim_Play* = ON die (bestehende) Aufzeichnung abgespielt werden und bei OFF die Aufzeichnung starten.

Ermittlung der Rohdaten der Telegramme

Nun ist zu klären, wie die Rohdaten der Telegramme am Bus ermittelt werden können. Dazu

Code-Teil 2

```
if event(readrawknx(Sim_Kontroll,Sim_Sender,Sim_GA,Sim_IsGa,Sim_RoutingCnt,Sim_Len,Sim_Data)) and
Sim_Len!=0 and Sim_IsGa and !Sim_Play then {
  if !Sim_MyGA then Sim_Next=OFF endif;
  if Sim_MyGA then {
     if Sim_Len==1 then Sim_RawData=convert(stringcast(Sim_Data,0u08,1u16) and 0x7F,0u32) endif;
     if Sim_Len==2 then Sim_RawData=convert(stringcast(Sim_Data,0u08,2u16),0u32) endif;
     // Byte Order has to be considered
      if Sim_Len==3 then Sim_RawData=convert(stringcast(Sim_Data,0u08,2u16),0u32)*256u32
+convert(stringcast(Sim_Data,0u08,3u16),0u32) endif;
     if Sim_Len==5 then Sim_RawData=convert(stringcast(Sim_Data,0u08,2u16),0u32)*16777216u32
+convert(stringcast(Sim_Data,0u08,3u16),0u32)*65536u32+convert(stringcast(Sim_Data,0u08,4u16),0u32)*2
56u32+convert(stringcast(Sim_Data,0u08,5u16),0u32)  endif;
     Sim_Next=ON;
  } endif;
}endif
```

Sim_RawData sind die Rohdaten im u32 Format. Wenn nur 1 Bit gesendet wurde, so sind 31 Bits „ungenutzt“. Die eintreffenden Daten werden von readrawknx in die String-Variable *Sim_Data* geschrieben. Diese sind grundsätzlich als Rohdaten zu betrachten und sollen dann in den u32 Bit Wert gewandelt werden. Die Anordnung der Daten in 4 Bytes (32 Bit) vereinheitlicht die Speicherung der Telegrammdaten und vereinfacht dadurch das Procedere (wie noch zu zeigen ist).

Zum Verarbeiten dieser Rohdaten im String *Sim_RawData* müssen nun die einzelnen Bytes als 1-Byte Integerwerte interpretiert werden. Dies geschieht mit Hilfe der Funktion stringcast.

X=stringcast(*src{cxxxx}, dest, Pos{u16}*)

Diese Funktion betrachtet die Bytes im String *src* ab der Byte-Postion *Pos*. *dest* gibt dabei den Zieldatentyp Umwandlung an, der somit die Anzahl der Bytes vorgibt, welche für die Umwandlung in das Ergebnis *X* festlegt. Anhand von Abbildung 69 soll das erläutert werden: Die Grafik zeige den String als Byteanordnung. An Position 3{u16} steht der Wert hexadezimal 0x74.

String als Bytearray

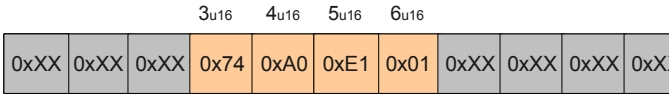


Abbildung 69: String src als Arrayfeld.

Eine Anweisung Z1=stringcast(*src,0,3u16*) wird eine Variable Z1 vom Datentyp u08 (Argument „0“) definieren. Der Wert wird aus *src* (Abbildung 69) an der Position 3{u16} gewonnen und ist damit im vorliegenden Fall 0x74 (dezimal 116). Eine Anweisung Z2=stringcast(*src,10u32,3u16*) definiert hingegen die Zahl 0x74a0e101 (dezimal 1956700417). Diese Anzahl der Bytes, die aus dem String entnommen werden, wird durch das Argument 10u32 gewonnen: Der Datentyp u32 ist 32 Bit lang und besteht aus 4 Bytes. Der Wert 10 von „10u32“ an sich, wird hier ignoriert. Die Reihenfolge der Bytes bleibt bei der stringcast Funktion unverändert.

Zurück zum Beispiel: *Sim_RawData* enthält die Daten des eintreffenden Telegramms in den ersten 4 Bytes. Die Reihenfolge der Bytes am Bus ist dabei unterschiedlich zur Byte-Reihenfolge des Linuxsystems des Enertex® EibPCs. Um diese Daten verwenden zu können, muss die Bytereihenfolge umgedreht werden, d.h. das letzte Bit muss an die erste Stelle usw.. Diese Umordnung wird mit Hilfe der Multiplikationen mit 256, 65536 und 16777216 realisiert.

Die vorliegende Verarbeitung von Rohdaten beschränkt sich demnach auf max. 32 Bit breite Telegramme. Längere Datentelegramme können nicht aufgezeichnet werden, anderseits werden für 1 Bit Objekte sicherlich Bytes „verschwendet“, da alle Telegramme gleich behandelt werden. Dennoch stellt dieses Vorgehen gewissermaßen einen optimalen Kompromiss dar, da die Verarbeitung später einfacher wird.

Mit dem Code-Teil 2 sind nun die Daten in der u32 – Variablen *Sim_RawData* berechnet.

Ermittlung des Zeitpunkt des Eintreffens der Telegramme

Die Sendezeitpunkte der Telegramme muss relativ bestimmt werden, da eine zuvor aufgezeichnete Simulation relativ (zeitversetzt) zum Startpunkt der Simulation erfolgen soll:

Code-Teil 3

```
// Die Uhr wird gestartet (Countdowntimer)
if Sim_Start then {
        Position=0u16;
        Sim_MyGA=OFF;
        if !Sim_Play then {
            stringset(TimeStamp,convert(Interval,0u32),Position);
        } endif;
} endif

// Die Uhr wird gestoppt nach dem Intervall
if afterc(Sim_Start,Interval,Timer) then {
        Position=0u16;
} endif
```

Der relative Zeitstempel ...

... wird nur 32 Bit breit gespeichert

Beim Wechsel vom *Sim_Start* auf EIN initialisiert die erste if-Anweisung den String TimeStamp. Zudem wird ein afterc-Timer (s.o.) gestartet. Dabei legt *Interval* fest, wie lange die Aufzeichung erfolgen soll, z.B. 1 Tag = 86400000ms. Diese Funktion aktualisiert bei jedem Schleifendurchlauf wie ein Countdown-Zähler die Variable *Timer*. Diese zählt also relativ zum Startzeitpunkt verstrichene Zeit in ms runter. Im String *Timestamp* wird der Start auf Position 0 geschrieben, wobei hier aus Gründen der Einfachheit die maximale Aufzeichnungsdauer auf 32 Bit (49 Tage) begrenzt ist.

Aufzeichnen der Daten

Wenn nun mit Code-Teil 1 festgelegt wird, dass die eintreffende GA aufgezeichnet werden soll (*Sim_MyGA* auf EIN), so werden die Daten in den String *Data* und die Gruppenadresse in den String *Recoder* gespeichert. Da die Gruppenadressen nur 16 Bit breit sind, kann die Bitlänge dann gleich in diesem Array mit abgelegt werden. Für die Speicherung von Rohdaten in einem String verwendet man stringset.

Abpeichern
- *Data*
- *Recoder*
- *Timestamp*

stringset(*dest{cxxxx}, src, pos{u16}*)

Diese Funktion schreibt in den Zielstring *dest* an dessen Speicherpostion *Pos* den (binären) Inhalt von *src*.

Code-Teil 4

```
if  !Sim_Play and Sim_Next then {
        stringset(TimeStamp,convert(Timer,0u32),Postion);
        //ggf. alten Zeitstempel löschen
        stringset(TimeStamp,convert(Timer,0u32),Postion+4u16);
        // GA abspeichern
        stringset(Recorder,Sim_GA,Postion);
        // Die Länge speichern
        stringset(Recorder,Sim_GA,Postion+2u16);
        // Den Wert speichern
        stringset(Data,Sim_RawData,Postion);
        Sim_MyGA=OFF;
        Sim_Next=OFF;
        Sim_GA=65365u16;
        Postion=Postion+4u16;
        // Überlauf?
        if Postion>capacity(TimeStamp) then Sim_Start=OFF endif;
} endif
```

Dadurch das Zeitstempel, Daten und Gruppenadressen 32 Bit breit abgespeichert werden, ist die Position eines Telegramms in diesen Strings gleich, was die Verarbeitung vereinfacht. In einen c1400 String können damit 350 Telegramme aufgezeichnet werden. Mit Hilfe von 65k Strings können 16341 Telegramme aufgezeichnet werden, im vorliegenden Fall wurde mit c4000 demnach der Telegrammspeicher auf 1000 festgelegt. Die Funktion capacity zeigt an, wie viel Bytes der String maximal abspeichern kann.

Nach Ablauf der vorgegeben Zeit wird in Code-Teil 3 die Aufzeichnung neu gestartet. Dabei werden die erst gespeicherten Werte neu überschrieben, die alten bleiben erhalten, was ggf. stören kann. Daher wird im obigen Code-Teil 4 ein eventuell aus einer vergangenen Aufzeichnung bestehenden Zeitstempel gelöscht.

Abspielen einer Aufzeichnung

Das Abspielen der Aufzeichnung ist relativ einfach. Es werden hierzu lediglich die Gruppenadresse und die Rohdaten auf den Speicher „geladen“ (Strings) und diese auf den Bus geschrieben. Dabei muss zunächst der Timer aus Code-Teil 3 neu gestartet werden. Die aktuelle Countdown-Zeit in *Timer* wird mit dem Zeitstempel in *Timestamp* verglichen und bei Unterschreiten der Zeit das Schreiben veranlasst:

Code-Teil 5

Abspielen

```
if Sim_Play and Timer<convert(stringcast(TimeStamp,1u32,Position),0u64) then {
      SimGA_Out=stringcast(Recorder,0u16,Position);
      SimGA_Len=stringcast(Recorder,0,Position+2u16);
      SimGA_Val=stringcast(Data,0u32,Position);
      if SimGA_Len==1 then write(address(SimGA_Out),convert(SimGA_Val,EIN)) endif;
      if SimGA_Len==2 then write(address(SimGA_Out),convert(SimGA_Val,0)) endif;
      if SimGA_Len==3 then write(address(SimGA_Out),convert(SimGA_Val,0u16)) endif;
      if SimGA_Len==4 then write(address(SimGA_Out),SimGA_Val) endif;
      Position=Position+4u16;
} endif
```

Dabei müssen die Datentypen aufgrund der Verwendung der Rohdaten nicht beachtet werden. Lediglich die Länge der Telegramme ist auszuwerten, so dass sie denen der Aufzeichnung entsprechen.

Die Makrobibliothek EnertexAnwesenheit.lib ist auf diese Weise realisiert.

In der Bibliothek wird die Aufzeichnung in kleinere Tagesintervalle zerlegt und beim Abspielen später wieder zusammengesetzt. Die Aufzeichnung beginnt dann jeweils zum nächsten Tagesintervall.

Nützliches

Encoding bei C14

Der KNX™ Standard sieht vor, dass Geräte bei 14-Byte Textmeldungen („c14“-Typen) nur den ASCII Code umsetzen müssen. Dabei wird in einer Erweiterung optional ISO8859-1 erlaubt, die ihrerseits nur aus 1-Byte Zeichen bestehen (vgl. http://de.wikipedia.org/wiki/ISO_8859-1).

Die Vorgabe des Strings im Enertex® EibStudio folgt der Codierung des Betriebssystems, so dass z.B. unter Windows XP, Deutsche Version, das ° Zeichen (MASCULINE ORDINAL INDICATOR) als Ein-Byte 0xBA dem Parser übergeben wird.

Wird unter OSX, 10.6.8, mit dem Enertex® EibStudio gearbeitet, so ist die Systemcodierung UTF-8. In diesem Fall ist das ° Zeichen ein 2-Byte Zeichen (0xC2BA) und der EibParser wirft eine Fehlermeldung, dass dies nicht erlaubt ist, aus.

Demnach kompiliert in der WinXP Version von Enertex® EibStudio

```
st=$10 °C$c14
```

problemlos, nicht aber unter OSX. Im ersten Fall handelt es sich beim °-Zeichen um ein zulässiges 1-Byte Zeichen, im zweiten Fall um ein unzulässiges 2-Byte Zeichen. Der Versatz 0xC2 ist keineswegs konstant: z.B. ist "ä" unter 8859-15 der 8-Bit Wert 0xE4, hingegen unter UTF-8 der 16-Bit Wert 0xc3a4.

Windows, OSX oder Linux

Generell kann man festhalten: Windows-XP und Win7 User (in der deutschen Version) können das °-Zeichen und Umlaute einfach nutzen, OSX oder Linux-User nicht bzw. ist hier die Systemcodierung für die Anwendung entscheidend.

Will man dennoch einen erweiterten ASCII Satz des IS8859-15 nutzen, muss für Utf-8 Systeme mit folgendem Konstrukt gearbeitet werden, wobei encode genutzt wird (vgl. S 235):

```
string=convert(encode($Hällo °C$,$utf-8$c14,$iso8859-15$c14),$$c14)
```

Das Problem der Codierung entsteht nur bei der Verwendung von Sonderzeichen im c14 String, da dieser als 1-Byte ASCII dargestellt werden muss. Alle anderen Strings (cX mit X von 1 bis 65534) werden von Enertex® EibStudio automatisch in UTF-8 konvertiert und plattformunabhängig codiert.

Daher ist die Befehlszeit nicht plattformunabhängig und würde beispielsweise unter Windows XP sicher zu einem ungültigen Zeichen am Bus führen.

Um einen Code für c14 Strings plattformunabhängig zu erstellen, muss daher auf die Compilerdirektive #ifdef und die vordefinierten Konstanten OSX, WIN und LINUX zurückgegriffen werden:

```
#ifdef OSX
string=convert(encode($Hällo °C$,$utf-8$c14,$iso8859-15$c14),$$c14)
#endif
#ifdef WIN
string=$Hällo °C$c14
#endif
```

Nun kann EibStudio das Telegramm z.B. unter OSX nicht mehr richtig darstellen, da ja ein "ä" unter UTF-8 ein anderer Zeichencode wäre als am KNX™ Bus. Am Bus sendet aber der Enertex® EibPC den richtige Code, was sich leicht anhand der Rohdaten zeigen lässt.

Stringverkettung mit unterschiedlicher Länge

Bei der Stringverarbeitung wird häufig auf die Verkettung zurückgegriffen, d.h. das „Aneinanderhängen“ von Zeichenketten.

So wird z.B. im Code

Verkettung gleichgroßer Strings

```
s1=$Hallo $c1000
s2=$Welt$c1000
s3=s1+s2
```

der String *s3* den Inhalt *Hallo Welt* haben. Die Datentypkontrolle im EibParser sorgt dafür, dass *s3* vom Typ c1000 ist. Der EibParser sorgt dafür, dass die Verkettung die Größe des längsten Strings aufnehmen kann, im vorliegenden Fall sind das für *s1+s2* 1000 Bytes. *s3* werden als Ergebnis der Verkettung *s1+s2* 1000 Bytes zugewiesen.

Wenn in *s2* bereits 950 Bytes an Daten vorliegen und *s1* seinerseits 90 Bytes belegt, so werden 40 Bytes bei der Verkettung „verloren“ gehen, da *s3* nur max. 1000 Bytes aufnehmen kann.

Im folgenden Code ist das ebenso zu beachten:

Verkettung zweier Strings und Zuweisung auf einen größeren String

```
s1=$Hallo $c1000
s2=$Welt$c1000
s3=$$c2000
if htime(10,00,00) then s3=s1+s2 endif
```

Auch hier wird die Verkettung *s1+s2* die Länge von 1000 Bytes, da sie sich aus aus zwei 1000 Byte-Strings zusammensetzt. Die Zuweisung auf den 2000 Bytes langen *s3* erfolgt erst nach der Verkettung. Da aber schon die Verkettungsoperation die Länge auf 1000 Bytes begrenzt hat, können hier Bytes „verloren“ gehen.

Im folgenden Code ist das anders:

Verkettung zweier Strings und Zuweisung auf einen kleineren String

```
s1=$Hallo $c1000
s2=$Welt$c1000
s3=$$c200
if htime(10,00,00) then s3=s1+s2 endif
```

Auch hier wird die Verkettung *s1+s2* die Länge von 1000 Bytes, da sie sich aus aus zwei 1000 Byte-Strings zusammensetzt. Die Zuweisung auf den 200 Bytes langen *s3* erfolgt erst als Ergebnis der Verkettung: Zunächst begrenzt die Verkettungsoperation *s1+s2* die Länge auf 1000 Bytes, die Zuweisung auf *s3* begrenzt dessen Länge auf 200 Bytes, also gehen ggf. 800 Bytes an Daten „verloren“.

Datenverlust vermeiden:

Soll die Verkettung *s1+s2* in keinem Fall Daten verlieren, muss eine Dummy-Variable eingeführt werden:

```
s1=$Hallo $c1000
s2=$Welt$c1000
s3=$$c2000
dummy=$$c2000
if htime(10,00,00) then s3=s1+s2+dummy endif
```

Hier ist gewährleistet, dass *s1+s2+dummy* 2000 Bytes als Ergebnis aufnehmen kann. Daher wird auch die Verkettung 2000 Bytes an *s3* als Ergebnis liefern.

FTP Datenströme

Mit Hilfe von konfigurierbaren FTP Transfers können beliebige ASCII („Klartext“) Dateien auf einen externen FTP Server geschrieben werden. Die maximale Dateigröße beträgt 64 kB.

Vier Datenströme

Dazu können **vier** verschiedene Handles (=ID-Nummer des Transfers) angelegt werden, die - selbst per Warteschlange gepuffert - diese Dateien auf dem Server anlegen. Die Dateien werden per Timeout auch früher (und dann ggf. weniger Bytes) oder per flushftp() vom User initiiert geschrieben. Die Dateinamen werden von der Firmware automatisch nach Datum und Uhrzeit vergeben.

Im folgenden soll das Vorgehen beim Anlegen und Anwenden dieser FTP Auslagerung detailliert beschreiben werden.

Zunächst muss im Programm der Stream und sein Handle definiert werden. Hierzu wird die Funktion

ftpconfig(server,user,password,path,timeout)

benötigt (S. 257). Ein Handle bezeichnet eine eindeutige Zahl (ID) für einen Transfer und ist etwa gleichbedeutend mit einer Namensgebung.

Die ersten 3 Argumente dienen der Konfiguration des Tranfers: IP Adresse, Username und Passwort, dann folgt das Zielverzeichnis auf dem Server und ein Timeout Parameter. Mit dieser Anweisung reserviert man einen 64 kByte großen Puffer im Enertex® EibPC. Die Übertragung des Puffers erfolgt dann, wenn entweder der Puffer vollends befüllt wurde (dazu unten mehr) oder die Anzahl *timeout* Sekunden seit dem letzten Transfer verstrichen sind.

Konfiguration des Transfers

```
// ServerDaten
server=$ftp.enertex.de$
user=$enertex$
password=$enertex$
path=$KNX/Telegramme$

// Timeout in Sekunden
timeout=900u32

// FTP Queue anlegen
// Wenn Handle ungleich Null, dann ist das fehlerfrei gelungen
Handle=ftpConfig(server,user,password,path,timeout)
```

Während des Betriebs müssen nun die Daten in den Puffer geschrieben werden. Dazu wird

Mehrere Strings werden in einer Textzeile zusammengefasst

sendftp(handle,data1,[data2],[...])

benötigt. Die Funktion erlaubt beliebige Strings als Argumente, da die Zieldatei ebenso nur eine Textdatei darstellt. Etwaige Daten in Form von Zahlwerten müssen daher mit Hilfe der convert-Funktion umgewandelt werden. Dabei wird am Ende der Datenübermittlung von sendftp ein LF-CR (Zeilenumbruch für Windows geeignet) eingefügt. Ein Aufruf von sendftp kann mehrere Teilstrings übergeben, aber nicht mehr als 1400 Bytes insgesamt übernehmen. Demnach ist die maximale Zeichenlänge 1400 Bytes:

```
// Daten in die Queue schreiben
Data1=$Daten Nr. $
Data2=$ des internen Zählers - $
Nr=0u16
status=3
// minütlich werden die Daten Data1 in den internen Buffer geschrieben
// nach 15 Minuten (timeout) werden die Daten am FTP-Server ausgelagert
if stime(00) then {
  status=sendftp(Handle, Data1,convert(Nr,$$),Data2,convert(settime(),$$));
  Nr=Nr+1u16;
} endif
```

Wenn die Variable *status* auf 1 steht, war das Schreiben auf den Puffer des Transfers erfolgreich. Dies hat aber noch nichts damit zu tun, dass die Daten schon auf dem FTP Server angekommen sind.

Dazu muss der Status des FTP Datenstroms abgefragt werden.

Es steht hierfür

```
ftpstate(handle)
```

zur Verfügung.

Mit

```
ftpstatus=ftpstate(Handle)
if ftpstatus==5 then write('1/2/3'c14,$FTP Overflow$c14) endif
```

kann nun folgender Status erfragt werden:

- Konfiguriert / fehlerfrei = 0
- die letzte Übertragung war fehlerfrei = 1
- der FTP Server war nicht erreichbar = 2
- das Passwort/User nicht erlaubt = 3
- Das Zielverzeichnis nicht existent und konnte nicht angelegt werden = 4
- Die Warteschlange hat einen Überlauf (=5); dies kann nur entstehen, wenn vorher die Übertragung nicht erfolgreich war.
- Handle nicht definiert = 6

Wenn es für die Verarbeitung von Bedeutung ist, den Füllstand des Datenstrompuffers zu ermitteln, kann dies mit Hilfe von

```
ftpbuffer(handle)
ftptimeout(handle)
```

in Erfahrung gebracht werden. Die erste Funktion gibt die Anzahl der genutzten Bytes im Puffer, die zweite die verstrichene Zeit seit dem letzten Transfer an.

```
if mtime(0,0) then {
        //Füllstand des FTP Buffers
        buffer=ftpbuffer(Handle)+1u16
        //Bereits verstrichene Zeit seit dem letzten Transfer in Sekunden.
        timeout=ftptimeout(Handle)
} endif
```

Neben dem automatischen Schreiben der Daten auf den FTP Server, kann der Puffer auch mit Hilfe der Funktion

```
flushftp(handle)
```

durch Hochladen der Daten zum FTP-Server „manuell“ geleert werden:

```
// Daten "manuell" flushen (nur dann wird die Übertragung aktiv)
// täglich um 00:00:00 Uhr
if htime(00,00,00) then {
  status=flushftp(Handle);
} endif
```

Wenn nicht manuell geleert bzw. geschrieben wird, initiiert der Enertex® EibPC den Transfer selbstständig. Der Transfer erfolgt, wenn der Puffer voll ist oder der konfigurierte Timeout (in Sekunden) seit dem letzten Transfer verstrichen ist.

Ändern des Zielpfads

Sie können auch den Zielpfad zur Laufzeit umkonfigurieren. Dazu sollten sie bereits offen Streams vorher rausschreiben („flushen)

```
changepath=OFF
newpath=$ftp/Telegramme2$
if changepath then {
        status=flushftp(Handle);
} endif
if changepath and status==0 then {
        Handle=ftpConfig(server,user,password,newpath,timeout)
}endif
```

Nutzung von eigenen Html-Code und Grafiken auf dem Webserver

Weboutput

Mit den webouput Feld des Webservers kann der Anwender eigenen Html-Code auf der Visu anzeigen lassen (vgl. S. 18 / 294). Im Ausgabefeld kann ein einfacher Text, aber auch ein komplexer HTML- Code dynamisch dargestellt werden.

Fehlerhafter bzw. ungültiger Html-Code im Weboutput kann den Seitenaufbau stören. Derartige Fehler werden nicht vom kostenlosen Support behoben. Arbeiten Sie hier mit Tools wie etwa http://www.quackit.com/html/online-html-editor/, um den HTML-Code zu testen.

Dazu muss zunächst im Webserver das Ausgabefeld definiert werden:

Defintion von zwei Outputfeldern

```
[WebServer]
page (2) [$Haus$,$Energie$]
weboutput(Out1)[QUAD,ICON] weboutput(Out2)[QUAD,NOICON]
[EibPC]
Out1=2
Out2=3
```

Weboutputs sind immer global

Man beachte, dass das weboutput-Feld nur global definiert werden kann. Das Element kann mit oder ohne Icon dargestellt werden (ICON bzw. NOICON). Die Breite ist festgelegt auf 2 Einheitsbreiten, die Höhe kann einfach (SINGLE), doppelt (DOUBLE) oder vierfach (QUAD) sein. Alternativ kann ab Version 3.100 die Breite und Höhe in ganzen Zahlen der Breite-Höhe Einheitsgrößen vorgegeben werden, also z.B. 2x5 oder 4x1 etc.

Die Einschränkung auf globale Elemente ergibt sich aus der Möglichkeit, dass das Weboutput-Feld 65 kByte an Daten aufnehmen kann. Bei 40 globalen Elementen sind dies 2 MB, die im RAM für diese Elemente vorgehalten werden müssen.

Maximal 65000 Zeichen

Mit der Funktion

```
weboutput(ID,Data)
```

werden die Daten auf das Feld geschrieben. Data ist hier ein String mit der maximalen Länge von 65534 Bytes (Typ c65534). Eine Besonderheit ist, dass dieser String auch ein gültiger html-Code sein kann. Damit ist es möglich, eigene Formatierungen und Anzeigen zu dynamisieren.

Wir wollen nun die beiden eingangs definierten Felder so beschreiben, dass eine Webseite wie in Abbildung 70 entsteht:

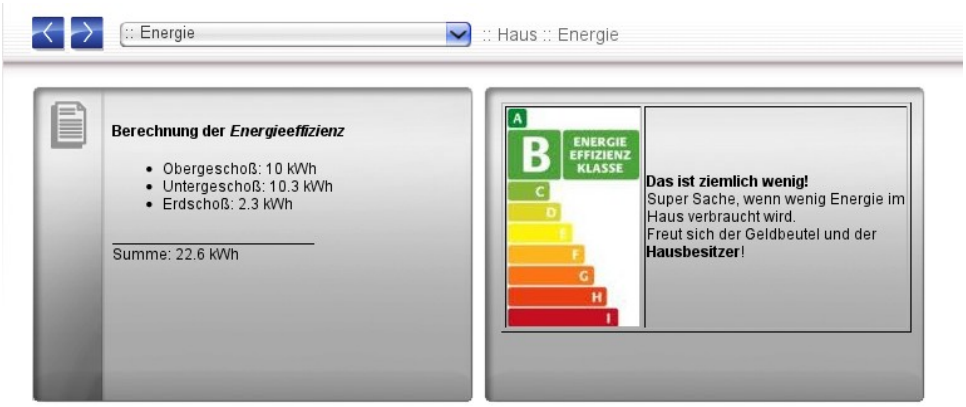


Abbildung 70: Gewünschte Ausgabe

HTML Code...

Für die Erstellung des eigentlichen Html-Codes sei auf http://de.selfhtml.org verwiesen. Der HTML-Code für kann mit Hilfe der Webseite wie folgt vorgegeben werden:

kann auch dynamisch angepasst werden

```
if systemstart() then {
        weboutput(Out1,$<h4>Berechnung der <i>Energieeffizienz</i></h4> <ul style="list-style-type:disc"> <li>Obergescho&szlig;:  10 kWh </li> <li> Untergescho&szlig;: 10.3 kWh </li> <li> Erdscho&szlig;:  2.3 kWh </li> </ul> _____ <br>  Summe: 22.6 kWh  $c10000)
} endif
```

Man beachte, dass der Code innerhalb der $-Zeichen nicht umgebrochen werden kann. Bei der Entwicklung empfiehlt es sich u.U., zunächst den HTML Code getrennt zu erstellen und zu testen.

Mit Hilfe einer anderen Abhängigkeit als if systemstart() kann der Text und die Formatierung jederzeit auch während der Laufzeit des Programms verändert werden.

Das zweite weboutput-Feld soll zudem mit einer eigenen Grafik ausgestattet werden. Zunächst ist eine PNG oder JPG Datei in den Enertex® EibPC hochzuladen. Dies können Sie mit Enertex® EibStudio wie auf S. 207 beschrieben vorab erledigen. Der Pfad der Grafik für den weboutput ist dann `/upload/` + Dateiname. Damit wird kann die Grafik samt etwas Text und HTML Formatierung mit folgender Anweisung initialisiert werden:

```
if systemstart() then {
   weboutput(Out2,$  <table border="1"><tr> <td class="oben"> <img src="/upload/effb.jpg"
alt="Bild fehl"></td> <td class="mittig"><b>Das ist ziemlich wenig! </b><br> Super Sache,
wenn wenig Energie im <br> Haus verbraucht wird. <br>Freut sich der Geldbeutel und der
<br> <b> Hausbesitzer</b>!</td></tr></table>$)
} endif
```

Die Ausgabe kann auch von aktuellen Werten, z.B. Zählerständen eines KNX Geräts abhängig gemacht werden, was im folgenden gezeigt wird.

Ein Energiezähler misst auf der GA "Energie-2/3/5“,"Energie-2/3/6“ "Energie-2/3/7“ vom Typ u32 den Verbrauch in Wh. Wir definieren zunächst 3 Variablen in kWh als String (c1400)

Den Verbrauch in kWh umrechnen

```
VerbrauchOG_kWh=convert(convert("Energie-2/3/5",1f32)/1000f32,$$)
VerbrauchEG_kWh=convert(convert("Energie-2/3/6"1f32)/1000f32,$$)
VerbrauchUG_kWh=convert(convert("Energie-2/3/7",1f32)/1000f32,$$)
Summe_kWh= convert(convert("Energie-2/3/7“+"Energie-2/3/6"+"Energie-
2/3/5“,1f32)/1000f32,$$)
```

Um zwölf Uhr soll täglich der Wert angezeigt werden:

und auf den Webserver als String - Verknüpfung (man beachte das +-Zeichen)

```
if htime(12,0,0)  then {
            weboutput(Out1,$<h4>Berechnung der <i>Energieeffizienz</i></h4> <ul style="list-
style-type:disc"> <li>Obergescho&szlig;: $+ VerbrauchOG_kWh +$ kWh</li> <li>
Untergescho&szlig;: $+VerbrauchUG_kWh+$ kWh </li> <li> Erdscho&szlig;: $
+VerbrauchEG_kWh+$ kWh </li> </ul> _____ <br>       Summe:$
+Summe_kWh+$ kWh  $c10000)
} endif
```

Je nach den tatsächlich übertragenen Werten, wird am Webserver etwa wie in Abbildung 71 dargestellt die Anzeige sein:

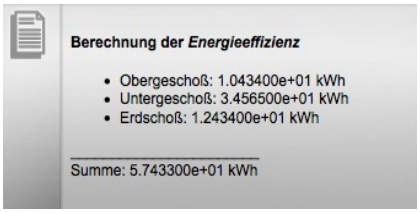


Abbildung 71: Dynamische Ausgabe

Im obigen Codeabschnitt wurde der HTML-String aus Teilstrings durch Verkettung („+“-Zeichen) gewonnen. Es ist wichtig sicherzustellen, dass die Verkettung die passende Stringlänge erzeugt. Die Funktion weboutput kann bis zu 65564 Bytes an das weboutput-Element übergeben. Die obige Verkettung besteht aber „nur“ aus $$ (= c1400) und einem c10000 String. Die Stringverkettung reserviert für das Ergebnis die Anzahl der Bytes, wie es das „längste“ String-Argument vorgibt. In diesem Fall sind dies 10.000 Bytes, gegeben durch den einen c10000 String im Code (vgl. oben).

Es sei an dieser Stelle nochmals angemerkt, dass Sonderzeichen aus mehreren Bytes bestehen können, wie schon auf S. 112 ausführlich beschrieben. Die Verkettung könnte theoretisch mehr als 10.000 Bytes als Ergebnis bringen, wenn die Strings jeweils die volle Länge ihrer Definition ausschöpfen. In diesem Fall würden die „überstehenden“ Zeichen nicht berücksichtigt bzw. schneidet die Verkettungsfunktion diese Zeichen vom String vor dem Kopieren in das Ergebnis ab. Es bleibt dem Anwender hier selbst überlassen, dies zu berücksichtigen vgl. S. 113.

Aus Gründen der Übersichtlichkeit kann der HTML-Code auch in eine eigene Datei ausgelagert werden. Hierzu arbeitet man mit der #include-Direktive (vgl. S. 136).

Zurück zum Beispiel:

Den meisten Anwendern wird hier die Ausgabedarstellung der exponentiellen Fließkommadarstellung nicht gefallen. Daher sollten die Darstellung der Werte mit der Funktion stringformat noch lesbarer gemacht werden. Diese Funktion wandelt eine Zahl in einen String, wobei man führende Nullen, die angezeigte Genaugigkeit und Fließkommadarstellung parametrieren kann.

Argumente:
1. Wert (hier f32)
2. Umwandlung von F32 in Fließ-kommadarstellung: 4
3. Darstellung mit führenden Nullen: 4
4. Max. Länge: 8
5. Genauigkeit: 1 Stelle

```
VerbrauchOG_kWh=stringformat(convert("Energie-2/3/5",1f32)/1000f32,4,4,8,1)
VerbrauchEG_kWh=stringformat(convert("Energie-2/3/6",1f32)/1000f32,4,4,8,1)
VerbrauchUG_kWh=stringformat(convert("Energie-2/3/7" ,1f32)/1000f32, 4,4,8,1)
Summe_kWh=stringformat(convert("Energie-2/3/5"+"Energie-2/3/5"+"Energie-2/3/5",1f32)/1000f32,4,4,8,1)
```

Abbildung 72: Formatierte Ausgabe

Picture, Header und Footer

Mit der Möglichkeit, eigene Bilder in den Enertex® EibPC zu laden, können Sie diese auch zur Gestaltung einer eigenen Anzeige für die Header und Footer- und picture Grafik nutzen. Zunächst müssen Sie die Grafiken wie auf S. 207 beschrieben vorab in den Enertex® EibPC hochladen. Im Folgenden sei dies mit *untergang.png*, *aufgang.png* und *sun.png* bereits geschehen.

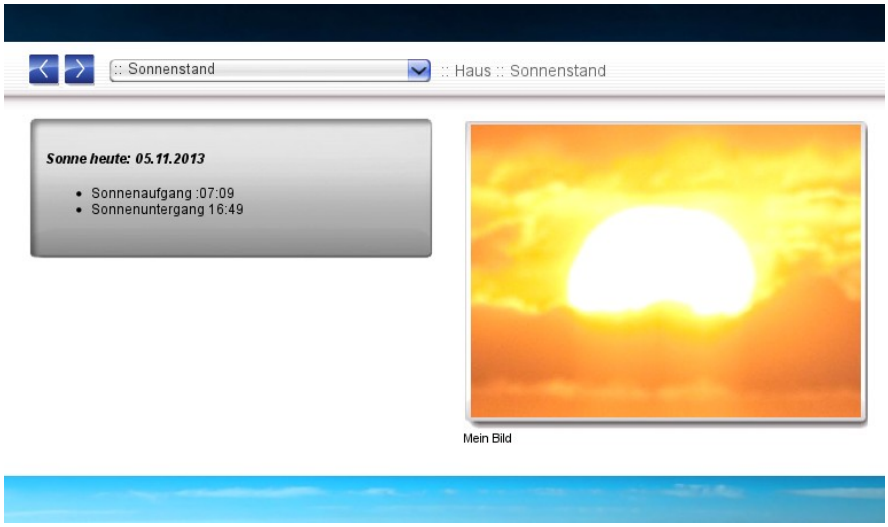


Abbildung 73: Verwendung eigener Grafiken

Anstelle der externen Links bei header und footer (vgl. S 296) können Sie nun einen Verweis auf die Grafiken, die im Enertex® EibPC gespeichert wurden, verlinken. Anstelle eines externen Links geben Sie die abgelegte Grafik als Pfad `/upload/` + Dateiname in der Konfiguration an:

Sonnenaufgang und -Untergang passend formatiert als String

```
[WebServer]
page (2) [$Haus$,$Sonnenstand$]
header(2) $/upload/untergang.png$
weboutput(Out1)[DOUBLE,NOICON] picture(0)[DOUBLE,CENTERGRAF] ($Mein Bild$, $/upload/sun.png$)
footer(2) $/upload/aufgang.png$

[EibPC]
Out1=2
Sonnenaufgang=stringformat(sunrisehour(),0,3,2,2)+$:$+stringformat(sunriseminute(),0,3,2,2)
Sonnenuntergang=stringformat(sunsethour(),0,3,2,2)+$:$+stringformat(sunsetminute(),0,3,2,2)

if systemstart() or htime(0,0,0) then weboutput(Out1,$<h4><i>Sonne heute: $ +convert(setdate(),$$)+$ </i></h4> <ul style="list-style-type:disc"> <li> Sonnenaufgang :$ +Sonnenaufgang+$</li> <li>Sonnenuntergang $+Sonnenuntergang+$</li>$ ) endif
```

Damit können Sie nun ihre Webseite etwa wie in obiger Abbildung 73 gestalten.

Visualisieren von Zeitreihen

Erweitertes Mtimechart (EXT)

Mit dem EibPC (ab V3.000) können auf einfache Weise Zeitreihen aufgenommen, permanent gespeichert und visualisiert werden. Dazu steht im Webserver das (globale) Diagramm-Element mtimechart (S. 290) zur Verfügung.

Wie in der Definition S. 290 angegeben, können Diagramme mit zwei Achsen beschriftet und die Skalierung automatisch oder per Wertebereichsvorgabe gewählt werden. Es gibt Diagramme in den Größen DOUBLE (2,2), TRIPLE (3,2), QUAD (4,2) und LONG (4,4) mit den Einheitshöhen und -breiten wie in den Klammern angegebenen.

Zusätzlich gibt es die EXT-Diagramme EXTDOUBLE, EXTTRIPLE, EXTLONG, die die erweiterten Funktionalitäten bei einfacherer Parametrierung ermöglichen. Mit diesen wollen wir beginnen. Wir definieren zunächst ein Webelement mtimechart mit sechsfacher Breite, wo beide Achsen angezeigt und nicht automatisch skaliert werden.

```
[WebServer]
page(1)[$Log$,$Room1$]
mtimechart(R1_ID)[EXTLONG,NOAUTOSCALE,192,17,30,0,100] ( $Temp$,LEFT, ChartBuffer0,
$Control$,RIGHT,ChartBuffer1 )
```

In der Definition wird neben der Angabe, dass kein automatisches Skalieren der Achsen gewünscht wird (NOAUTOSCALE), festgelegt, dass das Diagramm maximal 192 Wertepaare (Wert/Zeit) darstellt. Maximal möglich sind hier 256. Die erste Y-Achse stellt Werte im Bereich 17 bis 30 dar, die zweite 0 bis 100. Der erste Puffer (Buffer0) wird der linken und der zweite Puffer (Buffer1) der rechten Achse zugeordnet.

Ein mtimechart-Diagramm mit erweiterten Anwendungsmöglichkeiten zum Verschieben, Ausblenden, Zoomen von Graphen

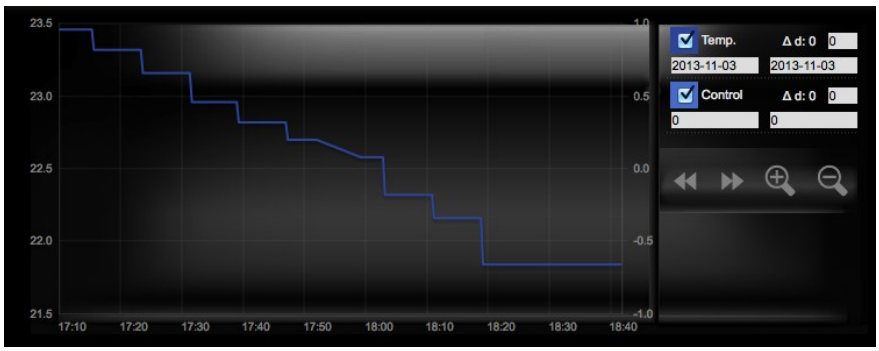


Abbildung 74: Timechart Webelement EXTLONG

Zeitreihen vergleichen: Feld Δd

Abbildung 74 zeigt dieses mtimechart-Diagramm, welches zwei von insgesamt vier möglichen Zeitreihen darstellt. Der User kann in der Zeitreihe links und rechts scrollen (Buttons << bzw. >>), sowie zoomen. Beide Operationen werden auf alle Graphen eines mtimecharts angewandt. Mit dem Feld für Δd kann ein Versatz in Tagen für die angezeigten Zeitreihen einzeln eingestellt werden, um so z.B. Verbrauchsdaten von zwei Zeitreihen im Jahresverlauf zu vergleichen. Diese Bedienfunktionen sind Teil des EXT-Diagramms selbst, sodass bei der Webelementprogrammierung kein weiterer Aufwand entsteht. Lediglich die Zeitreihen (timebuffer) selbst müssen noch konfiguriert und aufgenommen werden.

Dazu betrachte man die folgende Definition (vgl. 269):

timebufferconfig(*ChartBufferID, MemTyp, Laenge, DataTyp*)

Den Speichergröße an die Bedürfnisse anpassen

Mit dieser Funktion lassen bis zu 256 (ID 0 bis 255) verschiedene Puffer für die Aufzeichnung von Zeitreihen anlegen. *MemTyp* gibt an, ob der Speicher im Ring (0) oder linear (1) beschrieben wird (dazu unten mehr). Die Länge der max. Aufzeichnung der Zeitreihe wird mit *Laenge* (0u16 bis 65565u16) angegeben. Pro abgespeicherten Wert (s.u.) benötigt eine Zeitreihe 12 Bytes unabhängig vom gespeicherten *DataTyp*. Daher empfiehlt es sich, diese Größe der Speicher den tatsächlichen Bedürfnissen anzupassen: Eine Zeitreihe mit der maximalen Länge belegt 780 kB RAM.

DataTyp stellt eine repräsentative Zahl der Zeitreihe dar, z.B. 0f16 für 16-Bitzahlen oder 3% für u08 Werte. Die Zahl selbst wird nicht weiter verarbeitet und dient dem Compiler nur dazu, die Typinformation zu gewinnen. Wir wollen den timebuffer mit ID 0 für die Aufzeichnung der Temperatur-Gruppenaddresse 1/2/3 (vom Typ f16) nutzen, die ID 1 für die Stellgröße des Heizreglers 1/2/4 (u08).

```
[EibPC]
R1_ID=1
// Timebuffer IDs vergeben:
ChartBuffer1=1
ChartBuffer1=2
// timebufferconfig: Einen Zeitbuffer konfigurieren
 MemTyp=0
Len=35040u16
Datatyp=3.3f16
timebufferconfig(ChartBuffer1, MemTyp, Len, "Temperature-1/2/3")
timebufferconfig(ChartBuffer2, MemTyp, Len, "Controll-1/2/4")
```

Die Lesebarkeit des Codes wird erhöht, wenn wir im obigen Beispiel als letztes Argument die zu speichernde Variable oder Gruppenadresse angeben. Dies ist nicht zwingend notwendig, z.B. würde auch timebufferconfig(ChartBuffer1, MemTyp, Len, 2.2f16) oder timebufferconfig(ChartBuffer2, MemTyp, Len,2) den timebuffer richtig konfigurieren.

Mit der Konfiguration des timebuffers wird dem Webelement mtimechart der Speicher der Zeitreihe (timebuffer) zur Darstellung übermittelt, indem man deren ID (=Handle, Zugriffsnummer) konfiguriert. Dabei greift sich das Webelement immer die letzten gültigen Daten im Speicher heraus.

Nun müssen die Zeitreihen noch mit Daten „befüllt“ werden. Die Funktion

Einen Wert und den Zeitstempel in den Speicher der Zeitreihe schreiben

timebufferadd(*ChartBufferID, Daten*)

erledigt diese Aufgabe. Die Funktion schreibt den aktuellen Wert der Variable oder Gruppenadresse (*Daten*) sowie den Zeitstempel, der sich aus der Systemzeit des Enertex® EibPC herleitet, in den Speicher der ausgewählten Zeitreihe. Damit besteht eine Zeitreihe genau genommen aus der Kombination Wert-Zeitstempel. Werte können bis zu 4 Byte lang sein, Zeitstempel benötigen intern 8 Bytes.

1.23 (4 Byte)	2.23 (4 Byte)	45.23 (4 Byte)	1.23 (4 Byte)	2.23 (4 Byte)	45.23 (4 Byte)
2013-11-08 8:00:00.223	2013-11-08 8:00:00.823	2013-11-08 8:03:00.223	2013-11-08 8:04:00.000	2013-11-09 8:00:00.700	2013-11-09 8:03:00.675

Abbildung 75:Aufbau der Zeitreihe (timebuffer)

Das Schreiben einen neuen Wertes aktualisiert automatisch auch das verknüpfte Webelement

Wie Abbildung 75 nahelegen soll, ist es nicht notwendigerweise so, dass die Werte im timebuffer im gleichen zeitlichen Abstand aufgenommen werden müssen, wenngleich dies beim Loggen von Energiedaten meist der Fall sein kann. Das Webelement mtimechart wertet den Zeitstempel automatisch richtig aus.

Wenn das Argument *MemTyp* von timebufferconfig als Ring[speicher] definiert wurde, so wird beim Erreichen des letzten Wertes der Speicher wieder von vorne befüllt, d.h. der älteste Wert wird gegen den neuesten getauscht. Ist *MemTyp* als Linear[speicher] definiert, so stoppt die Aufzeichnung, wenn der Speicher gefüllt ist.

Mit einer Zeitreihe verknüpfte Diagramme werden in der Visu automatisch aktualisiert, d.h. es können grundsätzlich die gleichen Zeitreihen in verschiedenen Diagrammen dargestellt werden. Um beispielsweise alle 15 Minuten einen Wert in den Puffer zu schreiben und die jeweils letzten 240 Werte in unserem Diagramm anzuzeigen, genügt folgender Code:

```
// Werte in den Buffer schreiben
if mtime(0,0) or mtime(15,0) or mtime(30,0) or mtime(45,0) then {
        timebufferadd(ChartBuffer0,"Temperature-1/2/3");
        timebufferadd(ChartBuffer1,"Controll-1/2/4");
} endif
```

Mit

timebuffersize(*ChartBufferID*)

kann der Füllstand eines Buffers jederzeit abgefragt werden.

Das mtimechart Webelement zeigt nun 192 Werte an, was einen Zeitraum von 2 Tagen gleichkommt. Unser Puffer hat Platz für 35040 Werte, was bei ¼ Stundenwerten einem Jahr Aufzeichnungsdauer entspricht. Abbildung 76 zeigt die Möglichkeiten für den Anwender, die vergangenen Werte darzustellen: Es kann ein Start- und Enddatum vorgegeben werden. Wenn in der Zeitreihe mehr als die im Webelement konfigurierte Anzahl von Werten in diesem Zeitraum abgespeichert werden, so passt das Diagramm die Anzeige so an, dass es Zwischenwerte ausblendet.

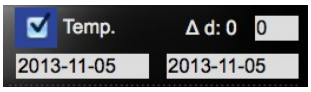


Abbildung 76: Timechart Verschieben

Werte aus der Vergangenheit anzeigen ...

Beispiel: Der Anwender stellt Zeitraum 4 Tage ein (z.B. 2013-07-11 bis 2013-09-13). Bei der hier gegebene Konfiguration sind im timebuffer (ID 0 und 1) 384 Werte abgelegt. Das Diagramm kann aber nur 192 Werte darstellen und lässt daher bei der Darstellung jeden zweiten Wert weg, so dass effektiv ½ Stundenwerte über 4 Tage angezeigt werden. Werteschwankungen, die nur im ¼ Stundenraster vorhanden sind, werden nicht mehr angezeigt. Die Zeitachse wird an die eingegebene Zeitdauer angepasst bzw. skaliert. Wenn der Anwender die Datumsfelder auf unterschiedliche Zeitintervalle parametriert, so wird die Achse so skaliert, dass die gespeicherten Werte vom ältesten zum neuesten Datum dargestellt werden.

... oder mit der Vergangenheit vergleichen.

Mit dem Feld für Δd wird ein Versatz in Tagen für das Darstellungsintervall des timebuffer eingestellt. Dabei wird die Zeitachse nicht skaliert, sondern bleibt auf dem eingestellten Datumsbereich stehen. Die Werte der um diesen Tage zurückversetzen Zeitreihe werden aus dem Speicher ausgelesen. Auf diese Weise können Kurven unterschiedlicher Zeitreihen übereinander gelegt und verglichen werden.

Wichtig ist hier anzumerken: Alsbald der User ein Diagramm verschiebt oder skaliert, hängt er dieses vom Echtzeitwebserver ab, d.h. weitere Wertänderungen, die in die Zeitreihe (timebuffer) geschrieben werden, sind nicht mehr am Webserver sichtbar, bis ein Seitenrefresh (meist Taste F5) des Browser ausgeführt wird. Dies betrifft aber nicht die anderen Elemente der Webseite.

Nachdem die Zeitreihe über einige Zeit im Enertex® EibPC aufgenommen wurde, muss sichergestellt werden, dass diese auch beim Neueinspielen des Programms oder Neustart die Werte nicht verloren gehen. Die Funktionen

timebufferstore(*ChartBufferID*)

timebufferread(*ChartBufferID*)

sind für diese Aufgabe geschaffen (vgl. S. 270).

timebufferstore legt die Werte des Timebuffers mit der *ChartBufferID* permanent in den Flashspeicher des Enertex® EibPC ab, timebufferread liest einen abgespeicherten Puffer zurück. Zudem können die Werte mit Enertex® EibStudio wie auf S. 207 beschrieben auf einem externen Gerät zur Sicherung der Daten herunter- und hochgeladen werden.

Somit speichern wir unsere Puffer alle 24 h auf folgende Weise:

Speichern im Flash

```
// Wert im Flash speichern
if chtime(01,00,00) then {
        timebufferstore(ChartBuffer0);
        timebufferstore(ChartBuffer1);
} endif
```

Die Werte sichern wir beim Programmstart wie folgt zurück:

Laden aus dem Flash

```
if systemstart() then {
        timebufferread(ChartBuffer0);
        timebufferread(ChartBuffer1);
} endif
```

Wird ein mtimechart als EXTDOUBLE deklariert, so entfallen die Felder der Abbildung 76, insbesondere ist das Verschieben und Übereinanderlegen verschiedener timebuffer nicht mehr möglich.

Einfaches mtimechart

Weniger „Bedienkomfort“, speziell bei der Anwendung mit einem Touchpanel, aber dafür mehr Platz für die Darstellung bieten die „einfachen“, d.h. nicht EXT-Formattypen der mtimecharts (vgl. Abbildung 77). In dieser Ausprägung erinnert das Diagramm an die mcharts bzw- mpcharts (vgl. S. 263 und S. 264), wobei auch hier die Zeitachse automatisch aus dem timebuffer geholt und skaliert wird.

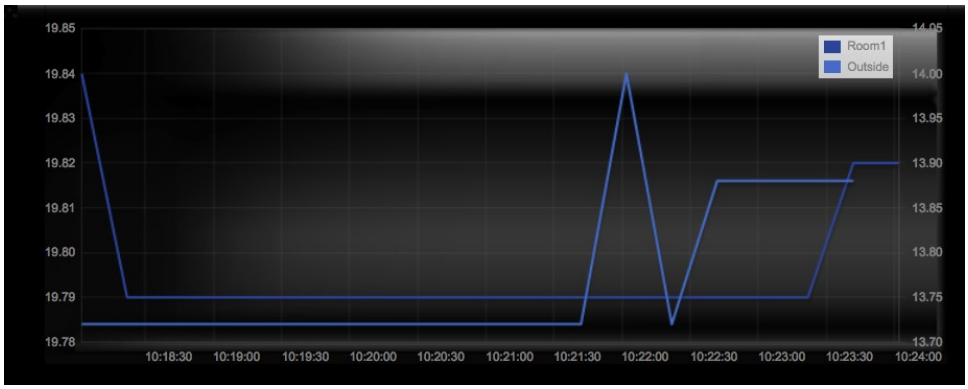


Abbildung 77: Standard-art Webelement

Die Definition des Elements ist im Webserver identisch zur Definition des EXT-Types, nur dass hier anstelle des EXTLONG als Format LONG seht.

```
[WebServer]
page(1)[$Log$,$Room1$]
mtimechart(R1_ID)[EXTLONG,NOAUTOSCALE,192,17,30,0,100] ( $Temp$,LEFT, ChartBuffer0,
$Control$,RIGHT,ChartBuffer1 )
```

Auch hier gilt: Alsbald der User ein Diagramm verschiebt oder skaliert, hängt er es vom Echtzeitwebserver ab, d.h. weitere Wertänderungen, die in die Zeitreihe (timebuffer) geschrieben werden, sind nicht mehr am Webserver sichtbar, bis ein Seitenrefresh (meist Taste F5) des Browser ausgeführt wird. Dies betrifft nicht die anderen Elemente der Webseite.

Die Zeitreihe wird grundsätzlich wie in den Codeabschnitten auf S. 119 und S. 120 identisch definiert.

Mit der Funktion

mtimechartpos(TimeChartID,ChartIdx,ChartBuffer,StartPos,EndPos)

mtimechart(TimeChartID,ChartIdx,ChartBuffer,StartZeit,EndZeit)

(vgl. S.) kann auch vom Anwendungsprogramm die Anzeige ähnlich zur direkten Werteingabe in Abbildung 76 der Darstellungsbereich der mtimechart-Webelemente verändert werden.

Auswählen der Werte nach Position im timebuffer mit der Funktion mtimechartpos() oder der Zeit mit mitimechart()

mtimechartpos benötigt neben der ID und dem Graphenindex im mtimechart die Position des Wertebereichs der Daten im Puffer, an welcher der Wert steht. Wie in Abbildung 78 angedeutet, „nummeriert“ der Enertex® EibPC jeden Speicherplatz von 0 bis n-1 durch. Dabei ist n die konfigurierte Pufferlänge, in Abbildung 78 sieht man einen Puffer mit der Länge 4000, der Startposition 0 und der Endposition 3999. Mit Hilfe von mtimechartpos greift man auf die entsprechende Stelle im Timebuffer zurück, dabei ist Position 0 immer der älteste Wert im Puffer und Position n-1 (im Beispiel also die 3999) der aktuellste Wert im Puffer.

0_{u16}	1_{u16}	2_{u16}	3_{u16}	4_{u16}	5_{u16}		3998_{u16}	3999_{u16}
1.23 (4 Byte)	2.23 (4 Byte)	45.23 (4 Byte)	1.23 (4 Byte)	2.23 (4 Byte)	45.23 (4 Byte)	...	1.23 (4 Byte)	2.23 (4 Byte)
2013-11-08 8:00:00.223	2013-11-08 8:00:00.823	2013-11-08 8:03:00.223	2013-11-08 8:04:00.000	2013-11-09 8:00:00.700	2013-11-09 8:03:00.675		2013-11-18 14:30:00.223	2013-11-18 21:00:00.000

Abbildung 78:Aufbau der Zeitreihe (timebuffer) mit Index und Länge 4000

mtimechart wertet nicht den Index des Graphen aus, sondern den Wert des Zeitstempels selbst. Dabei sind die Zeitangaben StartZeit,EndZeit im Argument als utc-Millisekundenformat vorzugeben. Um dies für den Anwender einfach zu gestalten, kann auf die Funktion

utc(*Zeit*)

zurückgegriffen werden (vgl. 186). Diese wandelt eine Stringangabe der Form $2013-01-30 14:00:00$ in das utc-Millisekundenformat.

```
if systemstart() {
    mtimechart(1,0,ChartBuffer0,utc($2013-01-30 14:00:00$),utc($2013-01-30 14:00:00$))
} enduf
```

Wechsel der angezeigten Puffer eines mtimechart

Interessant ist die Möglichkeit, die im Webelement vorkonfigurierte Verknüpfung von Zeitreihe zum Graphen zu „lösen“ und im Graphen einen anderen Puffer zu darzustellen.

Dazu ein weiteres Beispiel: Wie in Abbildung 79 soll über ein mpshifter-Webelement eine Auswahl getroffen werden, die im aufgenommenen Timebuffer dargestellt wird.

Das selbe Diagramm für unterschiedliche timebuffer verwenden: Hierzu wird mit der Auswahlbox unten links das Jahr gewählt. Das Anwendungsprogramm baut die Verbindung des Diagrammgrafen zum jeweiligen timebuffer auf.

Abbildung 79:Verändern der Darstellung während der Laufzeit

Im Webserver werden dazu die drei abgebildeten Elemente definiert, wobei der pshifter lediglich der Anzeige der aktuellen Uhrzeit dient. Beim Start des Anwendungsprogramms ist das Webelement auf den Timebuffer mit der ID Chartbuffer3 verknüpft.

```
[WebServer]
page(PageID)[$Log$,$Room5$]
design $black$
mtimechart(TimeChartID)[LONG,2,255,30,17,256,0] ( $Room1$,LEFTGRAF, ChartBuffer3)
mpshifter(SelectID)[$2011$,$2012$,$2013$][DATE]$Room1$ pshifter(ClockID)[CLOCK]
$Aktuelle Uhrzeit$
```

Wir definieren drei Zeitreihen (Timebuffer),

```
MemTyp=1
Len=30640u16
Datatyp=3.3f16
timebufferconfig(ChartBuffer0, MemTyp, Len, "RkWohnzimmerTemp-3/1/28")
timebufferconfig(ChartBuffer1, MemTyp, Len, "RkWohnzimmerTemp-3/1/28")
timebufferconfig(ChartBuffer2, MemTyp, Len, "RkWohnzimmerTemp-3/1/28")
```

Für jedes Jahr einen timebuffer, vgl. S. 206 für das Sichern der Daten auf einen externen PC.

die nun für jeweils 1 Jahr im ¼ Takt die Daten aufzeichnen:

Das „Sampeln“ (Abspeichern) der Werte in den Pufffer.

```
Y2011=date(1,1,11) and !date(1,1,12)
Y2012=date(1,1,12) and !date(1,1,13)
Y2013=date(1,1,13) and !date(1,1,15)
if (mtime(45,00) or mtime(45,00) or mtime(15,00) or mtime(00,00) ) and Y2011 then {
        timebufferadd(ChartBuffer0,"RkWohnzimmerTemp-3/1/28");
} endif
if (mtime(45,00) or mtime(45,00) or mtime(15,00) or mtime(00,00) ) and Y2012 then {
        timebufferadd(ChartBuffer1,"RkWohnzimmerTemp-3/1/28");
} endif
if (mtime(45,00) or mtime(45,00) or mtime(15,00) or mtime(00,00) ) and Y2013 then {
        timebufferadd(ChartBuffer2,"RkWohnzimmerTemp-3/1/28");
} endif
```

Wenn nun der Anwender die Auswahlbox verändert, so soll der entsprechende Timebuffer dargestellt werden:

Auswahlbox auswerten

```
if mpbutton(SelectID,1,PageID)==255 then {
        mtimechartpos(TimeChartID,0,ChartBuffer0,0u16,30639u16);
        pdisplay(SelectID,$Es wird 2011 dargestellt$,DATE,DISPLAY,GREY,PageID,1)
} endif
if mpbutton(SelectID,2,PageID)==255 then {
        mtimechartpos(TimeChartID,0,ChartBuffer1,0u16,30639u16);
        pdisplay(SelectID,$Es wird 2012 dargestellt$,DATE,DISPLAY,GREY,PageID,2)
} endif
if mpbutton(SelectID,3,PageID)==255 then {
        mtimechartpos(TimeChartID,0,ChartBuffer2,0u16,30639u16);
        pdisplay(SelectID,$Es wird 2013 dargestellt$,DATE,DISPLAY,GREY,PageID,3)
} endif
```

Man erkennt, wie der Graph mit Index 0 des mtimechart auf die verschiedenen Timerbuffer über deren ID „umgelenkt“ wird. Dazu greifen wir auf die Funktion mtimechartpos zurück, die den Jahres--Chartbuffer jeweils mit dem Graphen 0 verknüpft.

Noch eine kleine Ergänzung für die Anzeige der Uhrzeit: Diese wird nun in der Visu sekundengenau angezeigt, da der Echtzeitwebserver diese bei jeder Änderung des „Sekundenzeigers“ entsprechend anpasst.

Das Validierungsschema

Hintergrund

Für fortgeschrittene Programmierer soll hier einiges zum Hintergrund des Validierungsschemas erläutert werden.

Die Aufgabe der Automatisierung (Steuerung) ist es, das Anwenderprogramm so zu durchlaufen, dass sowohl ereignisorientierte Telegrammverarbeitung, beispielsweise das Eintreffen von KNX™-Telegrammen, wie auch zyklusorientierte Verarbeitung, wie z.B. Timer, Schaltuhren, verarbeitet werden können. Im Enertex® EibPC soll immer alles gleichwertig verarbeitet werden: Kein Telegramm soll in einer Warteschlange hängen bleiben auf die Verarbeitung warten müssen. Diese kann als harte Echtzeitanforderung verstanden werden.

Damit wird das erste Designkriterium für die Firmware festgelegt:

- *Versteht man jede Anweisung im Anwenderprogramm als Prozess, so werden innerhalb einer Zykluszeit determiniert alle Prozesse gleichzeitig bzw. parallel abgearbeitet.*

Es ergibt sich in der Folge:

- *Grundsätzlich muss das Anwendungsprogramm zyklisch durchlaufen werden.*
  Das Programm wird gestartet, liest die „Eingänge“ (Telegramme) ab, wertet Timer, Schaltuhren etc. aus, arbeitet das weitere Programm ab, es werden die „Ausgänge“ geschaltet und schließlich fängt das Ganze von vorne an. Dies ist also in gewisser Weise anders, als im „normalen“ Programm. Ein solches wird nur einmal gestartet und entsprechend der Eingaben des Anwenders wird ein Unterprogramm aufgerufen und solange abgearbeitet, bis die Verarbeitung bzw. der Anwender andere Angaben macht.

Viele KNX Geräte, die z.B. neben einer Touchpanelbedienung auch einfache Logik verarbeiten können, wählen zum einen eine Zykluszeit von minimal 100ms. Für die reine KNX Lösung mit vielleicht 10 UND und ODER Verknüpfungen wäre das ausreichend und die Zykluszeit von 100ms entschärft die Performanceproblematik zudem. Allerdings ist die Logik „überschaubar“. Der Enertex® EibPC soll diesbezüglich nicht beschränkt sein. Außerdem ist auch die Möglichkeit gegeben, mit TCP, UDP und beliebigen RS232 Telegrammen zu arbeiten und auch umfangreiche Berechnungen durchzuführen. Daher reicht eine einfache Erhöhung der Zykluszeit nicht aus.

- *Der Anwender soll von diesen Problemen möglichst entlastet werden. Eine Programmierung soll im Stile*
  if Taster==EIN then Write("ZimmerLicht-1/3/5“,EIN) endif
  erfolgen.
  Es ist zu notieren, dass das Anwenderprogramm zyklisch durchlaufen werden muss. Wenn das Programm im Sinne einer „klassischen“ Programmierung verarbeitet werden würde, so würde pro Zyklus festgestellt werden, dass z.B. die Variable auf EIN steht und das Zimmerlicht einzuschalten ist. Dies ist aber nicht so gewünscht.

Konzept

Aus diesen Anforderungen wurde das Validierungskonzept geboren. Jede Variable, Funktion bzw. Ausdruck weiß für sich genommen,

- *ob es sich geändert hat,*
- *ob es konstant geblieben ist,*
- *ob ein Ereignis aufgetreten ist,*
- *ob von ihm abhängige Ausdrücke sich verändern.*

Wenn eine Veränderung in diesem Sinne aufgetreten ist, so wird das Objekt auf den Status „ungültig“ gesetzt und die Auswertung des Objekts und seiner Abhängigkeiten neu initiiert. Wenn das Objekt dann verarbeitet ist, so wird es gültig (=valid). Eine „Auswertung“ einer Funktion „write“ beispielsweise ist gleichbedeutend mit dem Schreiben auf dem Bus. ***Das gesamte Programm wird compiliert, in dem es die Objekte in diese Abhängigkeitsstruktur umsetzt.***

Objektbaum

Variablen

```
[EibPC]
x=2
y="SaunaDimmer-1/0/1"+3%+x
z='1/2/3'b01 or '1/2/4'b01
```

Mit dieser Definition werden Variablen definiert und initialisiert. Der EibParser bringt diesen Codezeilen in eine Art Abhängkeitsbaum, wie in Abbildung 80 dargestellt. Wie in der Grafik gezeigt, „kennt“ jede Variable ihre Abhängigkeiten.

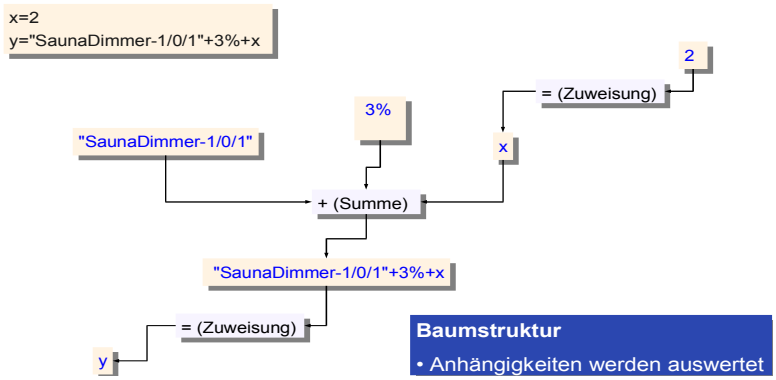


Abbildung 80: Baumstruktur für y="SaunaDimmer-1/0/1"+3%+x und x=2

Die Verarbeitung erfolgt in folgenden Schritten in jedem Zyklus, solange bis kein Objekt mehr invalid ist.

- *Invalidierung*
  Wenn eine Variable invalid ist, heißt das zunächst nur, dass sie neu berechnet werden muss. Damit eine Variable, Gruppenadresse überhaupt invalid wird, muss entweder die Initialisierung (erster Verarbeitungszyklus) vorliegen oder in allen anderen Zyklen ein Ereignis eingetreten sein. Trifft z.B. ein Telegramm vom Bus ein, schaut „weiß" die Firmware welches Objekt der Verarbeitung dadurch direkt invalid werden muss. Invalidiert werden können nur Objekte, die vom Systemtimer, TCP/UDP,RS232 oder einer if-Anweisung abhängig sind (vgl. Abbildung 81). Hierzu noch weiter unten mehr.
- *Berechnung: Wenn nun eine Variable (oder ein Ausdruck, Funktion etc.) invalidiert wurde, wird sie neu berechnet und führt nur bei ihrer Änderung den nächsten Schritt aus.*
- *Bedingte Invalidierung: Eine (veränderte) Variable schickt allen Variablen in der Abhängigkeitsliste ein Invalid-Signal.*

Die Verarbeitung erfolgt genau genommen eben nicht Zeilenweise, sondern ausgelöst durch ein Ereignis in einer Abhängigkeitsliste.

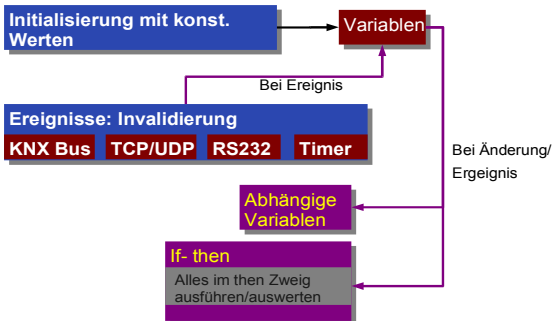


Abbildung 81: Baumstruktur

Damit zurück zum Beispiel: Nach dem ersten Durchlauf steht x auf 2 und y auf dem Wert der Gruppenadresse plus 3% plus x. Beim nächsten und allen folgenden Zyklen ändert sich in diesem Programm für x nichts mehr, da 2 konstant ist. Anders schaut es für y aus: Wenn vom Bus die Gruppenadresse eintrifft und sich deren Wert verändert hat, so wird y neu berechnet. Y ist von einem Ausdruck abhängig, der sich geändert hat und bekommt dies „mitgeteilt". Damit „weiß" y, dass es sich neu berechnen muss. Es wurde von der Gruppenadresse „invalidiert". Dabei wird y im übrigen nur dann ungültig, wenn sich der Wert der Gruppenadresse tatsächlich geändert hat. Gleichermaßen würde sich y neu berechnen, wenn sich x ändern würde. Die Invalidierung vererbt sich in der Auswertereihenfolge eines Ausdrucks („von unten nach oben" in der Abhängigkeit) bis zu der Ebene, bei der die veränderten Abhängigkeiten keine Veränderung mehr zur Folge haben.
Die Variable z ist indirekt von KNX™ Telegrammen abhängig: Wenn zunächst 1/2/3 auf EIN geht, ist der ODER Ausdruck auf EIN und dieser teilt das z mit, wenn er zuvor auf AUS stand. Wenn nun 1/2/4 auf EIN geht, wird wieder ODER invalidiert und berechnet sich neu. Da aber ODER bereits auf EIN steht, wird z nicht invalidiert und muss daher nicht neu berechnet werden.

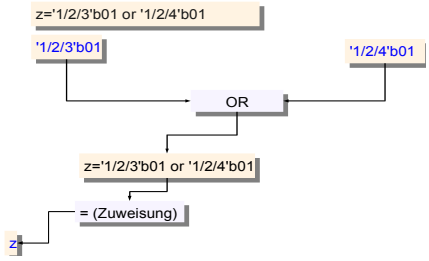


Abbildung 82: Baumstruktur für z='1/2/3'b01 or '1/2/4'b01

Eine optimale Performance ist somit a priori gewährleistet, da am KNX Bus pro 40ms max. 1 Telegramm eintreffen kann. Dieses wird sicher nicht alle Objekte invalidieren, da nur ein bestimmter (in einer umfangreichen Programmierung verschwindender Teil) von diesem Telegramm abhängig sein kann. Daher erreicht der EibPC eine Zyklusverarbeitung im Mikrosekundenbereich.

Funktionen zur Verarbeitung

```
[EibPC]
x=sin(3.14f32)
tan(2.0f32)
y=cos("Temperatur-1/0/1")
z=event("Temperatur-1/0/1")
```

Funktionen, die die Ausdrücke verarbeiten, werden abhängig von ihren Ausdrücken ausgewertet: Ändert sich das Argument (oder die Argumente), so werden die Funktionen bzw. deren „Ausgänge“ ungültig und berechnen sich neu. Im Beispiel wird also während der gesamten Laufzeit des Programms nur einmal der Sinus und der Tangens berechnet - der Cosinus nur, wenn sich die Temperatur ändert, genauso wie z bei Eintreffen eines Telegramms auf 1/0/1.

Funktionen zur Ausgabe

```
[EibPC]
write("Temperatur-1/2/1",22.3)
write("Schalter-1/2/10",!"Schalter-1/2/10")
read("Temperatur-1/2/1")
```

Funktionen, die Ausgaben auf den Bus oder TCP/UDP/RS232 liefern, werden nie von Ihren Argumenten ungültig. Daher führt obiges Programm nie etwas aus. Die zweite Anweisung würde sonst im Zyklustakt des Enertex® EibPC den Bus beschreiben.

Funktionen zur Zeitsteuerung

```
[EibPC]
o=stime(19)
```

Funktionen zur zeitlichen Steuerung werden ebenso nicht von Ihren Argumenten invalidiert. Diese sind vom Zeitgeber (interne Uhr) des Enertex® EibPC abhängig und werden von diesem entsprechend zur eingestellten Zeit invalidiert. O wird nur minütig bei 19 Sekunden nach der letzten vollen Minute für einen Verarbeitungszyklus auf EIN gehen.

Verschachteln von if-Anweisungen

Die If-Anweisung verhält sich wie eine Funktion, deren Argument die Abfragebedingung darstellt. Wenn die Abfragebedingung ungültig wird, so wird if neu ausgewertet. Diese Auswertung ist unabhängig davon, ob sich die Abfragebedingung vom Wert her ändert. Da die Abfragebedingung aber nur invalid werden kann, wenn sich deren Wert ändert, ist de facto eine ungültige Abfragebedingung auch eine veränderte Abfragebedingung für solche If-Anweisungen, die nicht in Verschachtelungen stehen.

```
[EibPC]
a=stime(33)
if stime(33) then write("Temperatur-1/2/1",22.3) endif
if '1/2/3'b01 then write("Temperatur-1/2/1",22.3) endif
```

Im Beispiel wird stime und abhängig davon „a“ jeweils 33 Sekunden nach der letzten vollen Minute invalid. Die Abfragebedingung der if-Anweisung wird daher ungültig und diese wird neu ausgewertet. Bei Abfragebedingung==EIN invalidiert die If-Funktion jede Anweisung im then-Zweig. Nun bekommt die write-Funktion ein „invalid“ und wird ausgewertet. Auswerten heißt in diesem Fall, das gewünschte Telegramm auf den Bus zu schreiben. '1/2/3'b01 wird zwar bei jedem Telegramm „invalid“, reicht aber dieses Signal nicht an die if-Anweisung weiter, wenn das eintreffende Telegramm (auf dieser Gruppenadresse) den gleichen Wert hat wie das letzte.

```
[EibPC]
a=1
if '1/2/3'b01 then a=3 endif
```

Nach der Initialisierung steht a auf dem Wert 1. Wenn nun '1/2/3'b01 von AUS auf EIN wechselt (Telegramm trifft ein), steht a auf 3. a bleibt auf diesem Wert und wird nicht im nächsten Zyklus auf 1 zurückgesetzt, da sich ja „1“ als Konstante in der ersten Zuweisung nicht mehr ändert.

Verschachteln von If

```
[EibPC]
a=1
b='1/2/4'b01
z=0
if  '1/2/3'b01 then {
     if b==EIN then  a=3 endif;
     z=cos(1);
    write('1/3/4'b01,AUS)
} endif
```

Würde die innere if-Anweisung nur durch ihre Abfragebedingung (b==EIN) invalidiert, würde die äußere Abfrage nicht berücksichtigt werden. Daher baut der Compiler das Konstrukt wie folgt um: Die innere if-Anweisung wird durch die äußere invalidiert (und nicht durch ihre Abfragebedingung). Die Auswertung der inneren Abfrage wird durch das Invalidieren angestoßen und damit in jedem Fall der then Zweig, unabhängig, ob die Abfragebedingung b==EIN seit der letzten Verarbeitung verändert wurde. Die äußere If-Anweisung invalidiert alle Objekte ihres then Zweiges in der Reihenfolge wie im Programm aufgeführt. Anschließend werden die ungültigen Ausdrücke ausgewertet. Dazu folgendes Beispiel:

```
a=AUS
b=1
if change(a) then {
   if b==1 then write('1/2/3'b01,AUS);b=2 endif;
   if b==2 then write('1/2/3'b01,EIN) endif
} endif
```

In diesem Beispiel wird bei Veränderung von a die äußere If-Abfrage aktiv und macht alle Anweisungen im then Zweig ungültig. Dann beginnt die Verarbeitung: Die erste innere If-Abfrage wird aktiv, setzt b auf 2 und schreibt auf 1/2/3 ein AUS. Die zweite If-Abfrage hatte nun bereits von der äußeren If-Abfrage zuvor ein „ungültig"-Signal bekommen und wird aktiv, da inzwischen b auf 2 gesetzt wurde. Daher wird nun auf 1/2/3 ein EIN-Telegramm geschrieben. Bei allen weiteren Veränderungen von a, wird die äußere If-Abfrage aktiv und daraufhin nur noch die zweite innere if-Abfrage.

Ändert man das Beispiel wie folgt:

```
a=AUS
b=1
if change(a) then {
   if b==2 then write('1/2/3'b01,EIN) endif;
   if b==1 then write('1/2/3'b01,AUS);b=2 endif
} endif
```

so wird nun bei der ersten Änderung von a die erste innere if-Anweisung ungültig, aber die Abfragebedingung steht auf 0b01, da ja die Variable b noch auf 1 steht. Die zweite innere if-Anweisung wird aber ausgeführt. Beim nächsten (und allen weiteren) wird nur noch die erste if-Anweisung ausgeführt.
Wie beim Eingangsbeispiel mit y wird die folgende If-Anweisung nur den then-Zweig invalidieren, wenn die ODER-Verknüpfung selbst „invalid" wird:

```
if '1/2/3'b01 or '1/2/4'b01 then { ... } endif
```

Zeitsteuerung im then-Zweig

Auch Zeitfunktionen werden in einer verschachtelten If-Anweisung eines then-Zweigs nur dann ausgewertet, wenn die if-Bedingung ein „invalid" an ihren then-Zweig sendet.

Dazu folgendes Beispiel:

```
Taste='1/2/3'b01
a=AUS
if Taste then {
   if htime(12,00,00) then {
       a=EIN
   } endif
} endif
```

a steht hier nur auf EIN, wenn Taste exakt um 12:00:00 auf EIN geht (htime generiert nur zu diesem Zeitpunkt einen EIN-Impuls). Besser wäre in diesem Fall daher die Verwendung von chtime. In diesem Fall wird a auf EIN gesetzt , wenn Taste nach 12:00 und vor Mitternacht auf EIN geht.

Der „Else-Zweig"

Der else-Zweig einer if-Funktion ist wie eine zweite eigenständige if-Funktion mit negierter Abfragebedingung zu verstehen.

```
[EibPC]
a=AUS
b=1
c='1/2/3'b01
if systemstart() then {
   if b==1 then {
                  write('1/2/3'b01,AUS);b=2
         }  else {
                  write('1/2/3'b01,EIN)
         } endif
} endif
// identisch zu
//    if systemstart() then {
//        if b==1 then write('1/2/3'b01,AUS);b=2 endif
//        if  !(b==1) then write('1/2/3'b01,EIN) endif
//    } endif
```

Schreiben auf Warteschlangen

Wenn die Verarbeitung eines „Programmdurchlaufs“ abgeschlossen sind, werden die systeminternen Warteschlangen für die Ausgabe beschrieben. Folgendes Beispiel:

Schreiben auf den BUS

```
[EibPC]
c='1/2/3'u08
b=1
if systemstart() then {
   if b==1 then {
                  write('1/2/3'u08,b);
                  b=2;
         }  else {
                  write('1/2/3'u08,b)
         } endif
} endif
```

Es wird (vgl. Beispiel oben) zweimal ein write angeregt. Die Daten werden erst in die Warteschlange gestellt, nachdem alle Objekte gültig sind. Wie oben bereits dargestellt, bedeutet dies, dass b auf 2 steht. Daher wird auf 1/2/3 zweimal der Wert 2 gesendet.

Gleichermaßen gilt diese Aussage für die Warteschlange für IP und RS232 Telegramme:

Schreiben auf den IP Sockets und RS232

```
[EibPC]
b=1
s=$Hallo$
if systemstart() then {
   if b==1 then {
            sendudp(4809u16,192.168.22.1,s);
            s=$Welt$;
            b=2
   }  else {
            sendudp(4809u16,192.168.22.1,s)
    } endif
} endif
```

*Es wird (vgl. Beispiel oben) zweimal ein string **Welt** geschickt. Auch hier wird erst auf die IP-Warteschlange geschrieben, nachdem alle Objekte gültig sind.*

Gleichermaßen gilt diese Aussage für die Warteschlange für das Schreiben und Lesen vom Flash, sowie das Schreiben auf strings mit stringset.

Schreiben auf das Flash

```
[EibPC]
b=1
s=$$
if systemstart() then {
  stringset(s,b,0u16);
  writeflash(b,0u16);
  b=b+1
} endif
```

Hier wird b zuerst um eins erhöht, und dann der Wert in das Array bzw. Flash geschrieben.

Asynchroner Zugriff

Bei einigen Funktionsaufrufen (wie etwa das Anlegen einer TCP-Verbindung) kann nicht gewährleistet werden, dass diese Ihren Rückgabewert innerhalb einer Verarbeitungsschleife des EibPC aktualisieren. Daher wurden diese Funktionen als eigenständige Threads innerhalb der Firmware realisiert. Diese Threads aktualisieren ihren Rückgabewert entsprechend ihrer eigenen Verarbeitung „asynchron“ zur Hauptverarbeitungsschleife. Dazu folgendes Beispiel:

```
[EibPC]
// TCP off == 5
TCP=5
if after(systemstart(),2000u64) then {
  TCP=connecttcp(233u16,192.168.2.100)
} endif
```

2 Sekunden nach Systemstart wird die Connectcp-Funktion aufgerufen. Da die Verarbeitung nun den entsprechenden Thread anstößt, ist der Rückgabewert zunächst 0 (Verbindung wird aufgebaut) und die if-Schleife verlassen. Wenn nach einiger Zeit die Verbindung besteht, aktualisiert nun connecttcp selbstständig die Variable TCP - unabhängig von der if-Abfrage - auf den Wert 1 (Verbindung besteht).
In analoger Weise sind die Rückgabewerte Funktionen resolve, readflash, sendmail, ping und writeflash als asynchron zu betrachten.

Arbeiten mit Makros

Makros sind immer im globalen Kontext wie eine Art Codeersetzung zu verstehen. Ab der Version 2.000 von Enertex® EibStudio können nun Makros auch im Code in der Sektion [EibPC] genutzt werden, was sich auch auf das Valdierungskonzept auswirkt.

Man betrachte das folgende Makro:

ACHTUNG:
Dieses Makro macht nie etwas

```
:begin MyFunction( Message )
    write( '9/2/0'c14, $Display $c14);
    write( '9/2/0'c14, $Message:$c14);
    write( '9/2/0'c14, convert(Message,$$c14))
:return AUS
:end
```

Der Eibparser verarbeitet diese Makros so, dass nur diejenigen Teile des Macrocodes nach der :return Anweisung eine Verknüpfung mit dem Validierungskonzept aufweisen. Daher wird folgender Code

```
[EibPC]
if sun()  then MyFunction($Licht$) endif
```

bei Sonnenaufgang nichts ausgeben. Er wird nämlich von Eibparser „expandiert" in die Form:

Alle write's sind global!

```
[EibPC]
write( '9/2/0'c14, $Display $c14);
write( '9/2/0'c14, $Message:$c14);
write( '9/2/0'c14, convert($Licht$,$$c14))
if sun() then AUS endif
```

Die write-Funktionen sind also nicht verknüpft mit der Funktion sun(). Verändern wir das Makro, indem wir nun die drei Ausaben in die :return Anweisung schreiben, so wird die Valdierung auf diese Objekte „weitergeleitet".

„Umleiten der Valdierung"

```
:begin MyOutputFunction( Message )
:return {
    write( '9/2/0'c14, $Display $c14);
    write( '9/2/0'c14, $Message:$c14);
    write( '9/2/0'c14, convert(Message,$$c14))
}
:end
```

Somit wird nun

```
[EibPC]
if sun()  then MyOutputFunction($Licht$) endif
```

auf die GA die drei Nachrichten „Display ", „Message:" und „Licht" ausgegeben.

Mit der :return Anweisung werden die Abhängigkeiten von einer if-Abfrage „weitergeleitet". Mit Hilfe dieser Anweisung lässt sich nun das Validierungskonzept komplett auch mit eigenen Funktionen vererben. Mit Hilfe der :return Anweisung kann aber auch ein ganzer Codeblock oder nur gezielte Teile des Makros vom aufrufenden Code abhängig gemacht werden.

Nochmals Schritt für Schritt

Nur globale Definitionen

```
:begin Akt_3(Aktor,Now)
   Variable=3
    if Now then write(Aktor,Variable) endif
:return AUS
:endif
```

Wenn das Makro wie folgt genutzt wird

```
[EibPC]
if sun() then Akt_3('1/2/3'u08,chtime(5,00,00)) endif
```

gilt zu beachten, dass der Eibparser nur die Objekte in die Abhängigkeit der aufrufenden if-Funktion bringt, die bei der :return Anweisung steht. In diesem Fall wird daher nur das Objekt (Konstante) AUS in die Abhängigkeit der if sun() ... Abfrage gebracht.

:return liegt also nicht nur den Rückgabewert fest, sondern auch welche Teile des Makros mit der Validierung des aufrufenden Kontext verknüpft werden.

In expandierter Form ist der Code demnach:

```
[EibPC]
Variable=3
if chtime(5,00,00) then write('1/2/3'u08,Variable) endif
if sun()  then AUS endif
```

Mit der Definition

```
:begin Akt(Aktor,Now)
:return Variable=3; if Now then write(Aktor,Variable) endif
:endif
```

wird nun

```
[EibPC]
Variable=0
if sun()  then Akt('1/2/3'u08,chtime(5,00,00)) endif
```

wenn bei Sonnenaufgang bereits 5:00 Uhr verstrichen ist, Variable auf 3 gesetzt und dies auf den Aktor geschrieben. Expandiert wird dies zu:

```
[EibPC]
Variable=0
if sun()  then Variable=3; if chtime(5,00,00)) then write('1/2/3'u08,Variable) endif
```

Wichtig ist hier, dass die Variable durch das „Einbinden“ in den then Zweig gar nicht mehr im globalen Kontext definiert wird, und daher die Definition Variable=0 in jedem Fall notwendig wird.

Rekursion

```
a=AUS
if a==EIN then a=!a else a=!a endif
```

Dieses Programm verursacht eine Rekursion, wie man anhand der Grafik sieht: Beim Initialisieren ist der else-Zweig aktiv und die Objekte in dessen Abhängigkeitsliste werden verarbeitet. Diese invertieren a und weisen a neu zu. Dabei wird a invalid und nach Neuberechnung ist es verändert. Dies bedeutet, dass a nun auf EIN steht und seinerseits seine Abhängigkeiten invalidiert. Daher wird nun die if-Abfrage neu ausgewertet und der then Zweig aktiv. Die Objekte in dessen Abhängigkeitsliste werden verarbeitet. Dabei wird a invertiert. Nun wird a invalid und nach Neuberechnung ist es verändert usw. Es bildet sich also innerhalb einer Verarbeitung eine Endlosschleife aus.
Diese wird von der Firmware mit einem Event (PROC_REPITIONS) abgefangen. Wenn dieses Ereignis auftritt, wird die aktuelle Verarbeitung des Objekts und damit die Endlosschleife unterbrochen.

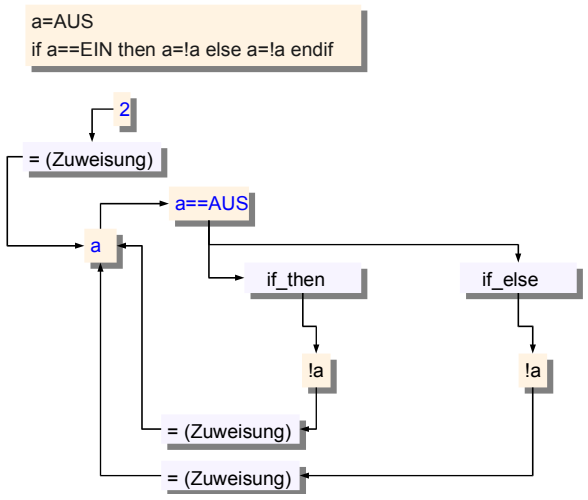


Abbildung 83: Baumstruktur für z='1/2/3'b01 or '1/2/4'b01

Insgesamt gewährleistet das Validierungskonzept, dass

- *der Enertex® EibPC auch bei umfangreichen Programmen extrem performant ist (Mikrosekunden),*
- *die Anwender Standardaufgaben (Wenn Taste, dann Licht) einfach coden können,*
- *alle Anweisungen parallel in einem Zyklus abgearbeitet werden.*

Enertex® EibStudio

Grundsätzliches

Einfache, komfortable Bedienoberfläche

Enertex® EibStudio ist ein Applikationsprogramm für Windows®- und Linux®-Rechner. Es ist die Entwicklungsumgebung für die Programmierung des Enertex® EibPC. Mit dem Erwerb des Enertex® EibPC erwerben Sie sich eine Lizenz zur Nutzung dieses Programms. Das Enertex® EibStudio darf nur in Verbindung mit dem Enertex® EibPC genutzt werden. Mit dem Enertex® EibStudio lassen sich Anwenderprogramme benutzerfreundlich erstellen, verändern und warten.

Installation

Kein Setup notwendig!

Das Enertex® EibStudio ist eine ausführbare Datei. Sämtliche notwendigen Programmteile sind integrierter Bestandteil dieser Datei. Es bedarf daher keiner Installation. Sie können dieses Programm von jedem Speicherort aus direkt ausführen. Insbesondere wird Ihre Windows®-Registry oder Ihr System vom Enertex® EibStudio nicht modifiziert.

Die Programmiersprache

Der Enertex® EibPC wird in einer einfachen, eigens entwickelten und anwenderfreundlichen Sprache programmiert. Das Programm muss für den Enertex® EibPC übersetzt (kompiliert) werden, da auf diese Weise der Code in optimaler Weise für den Enertex® EibPC zugeschnitten wird und keinerlei Performance-Probleme auftreten können Der notwendige Compiler, EibParser genannt, ist integrierter Bestandteil des Enertex® EibStudio. Auch ein Editor ist Bestandteil des Enertex® EibStudio. Er kennt alle Schlüsselwörter des Befehlssatzes der Programmiersprache und verfügt über Syntaxhervorhebung und Autovervollständigung.

Compiler

Das Anwenderprogramm wird in Form einer Textdatei eingegeben. Damit unterschiedliche Daten im Programm auch verschieden verarbeitet werden, gibt es innerhalb des Anwenderprogramms so genannte Sektionen. Diese sind durch einen Namen in eckigen Klammern gekennzeichnet. Eine Sektion beginnt mit ihrem Namen und endet mit dem Anfang der nächsten Sektion oder dem Ende der Datei.

Sektionen

Folgende Sektionen untergliedern die Textdatei

● [EibPC]	Ihre Programmierung
● [ETS-ESF]	Import von ETS Daten (Seite 158)
● [Location]	Geographische Daten des Aufstellungsorts (S. 157 und 190)
● [MacroLibs]	Vordefinierte Funktionsbibliotheken, welche eingebunden werden (S. 26 ff. und S. 323)
● [Macros]	Anwendung von Funktionsbibliotheken
● [RS232]	Konfiguration der RS232 Schnittstelle, wenn diese nicht von FT1.2 belegt (S. 237).
● [FTP]	Auslagern von Telegrammen auf FTP Server (S. 150).
● [Performance]	Einstellungen zur Performance des EibPCs (S. 135).
● [MailConf]	Konfiguration des Mailversand (Option NP) (S. 145)
● [WebServer]	Konfiguration des Webservers (Option NP) (S. 298)
● [VPN]	VPN Konfiguration (S. 254)
● [InitGA]	Gruppenadressen initialisieren (S. 172)

Anwenderprogramm

Im Anwenderprogramm (Sektion [EibPC]) steht Ihr Programm zur Hausautomatisierung. Sollten Sie nicht weiter in die Programmierung einsteigen wollen, steht Ihnen eine umfangreiche Bibliothek zur Verfügung, die Sie ohne Programmierkenntnisse bedienen können. Näheres hierzu finden Sie auf Seite 26

Das Anwenderprogramm (Sektion [EibPC]) besteht aus mehreren Anweisungen (s.u.). Die Verarbeitung erfolgt immer zeilenweise, d.h. eine Anweisung darf nicht durch einen Zeilenumbruch („RETURN“) unterbrochen werden. Wenn man einen Zeilenumbruch für bessere Lesbarkeit wünscht, so muss man die Zeile mit einem Doppel-Backslash „\\“ Zeichen abschließen und kann anschließend in der nächsten Zeile die Anweisung fortsetzen.

Kommentare

Kommentare beginnen mit „//“ und stehen am Anfang einer Zeile. Daneben gibt es noch Kommentare, die anstelle einer Anweisung stehen können, die durch /* */ eingerahmt werden. Z.B. `/* Dies ist ein Kommentar */`. Mit Hilfe des Menüpunkt Bearbeiten-Selektion kommentieren kann auch eine ganze Selektion kommentiert werden.

Mögliche Kommentare:

```
[EibPC]
/* Super, dieser Kommentar  */ c=$$
z= 1 // Eine Variable wurde definiert

/* Super, dieser Kommentar mit Strichpunkt  */; c2=$$

// Noch ein Kommentar

/* Meine Beste if-Anweisung */ if /* Achtung Bedingung ist 1b01 */ EIN  /* und wir haben nun
den Then-Zweig */ then {
    c=elog()
   /* Das ist ein Kommentar ohne Strichpunkt */
   /* auch kein Strichpunkt */
 }  endif

if  AUS then {
    //  Oder lieber so
    z=4;
    // Diese Anweisung eben war ziemlich dumm
    // weil wir lieber z auf 7 setzen sollten
    z=7;
    // Ende gut alles gut
} endif
```

Anweisungen

Anweisungen sind:

- Variablendefinitionen, siehe Seite 164
- Funktionsaufrufe, siehe Seite 176
- if-Anweisungen, siehe Seite 167
- Kommentare, eingefasst von /* und */ Zeichen (C-Syntax)

Anweisungsblock

Wenn zwei oder mehr Anweisungen als Block ausgeführt werden, können diese zu einem Anweisungsblock zusammengefasst werden. Dabei sind die Anweisungen durch Strichpunkt zu trennen.

Nach der letzten Anweisung steht kein Strichpunkt.

Zeilenumbruch

Auch für einen Anweisungsblock gilt:

Er darf nicht durch einen Zeilenumbruch („RETURN“) unterbrochen werden. Wenn man einen Zeilenumbruch für bessere Lesbarkeit wünscht, so muss man die Zeile mit einem Doppel-Backslash „\\“ Zeichen abschließen und kann anschließend in der nächsten Zeile den Anweisungsblock fortsetzen. Bei if-Abfragen kann alternativ mit geschweiften Klammern gearbeitet werden (Siehe Beispiel S. 41).

Performance

Das Anwenderprogramm wird nach einem so genannten „Validierungsschema“ abgearbeitet. Dies bedeutet, eine Anweisung wird immer nur ausgewertet, wenn eine Veränderung ihrer Abhängigkeit eintritt. Weitere Erläuterungen hierzu finden Sie auf Seite 168. Die Generierung des Telegrammpuffers und dessen Vorverarbeitung wird ständig durchlaufen, d.h. ankommende Telegramme werden im 1-ms-Takt abgearbeitet und in eine separate Warteschlange geschrieben. Gleiches gilt für RS232 und LAN Telegramme.

Anschließend wird die Hauptverarbeitung im Anwendungsprogramm ausgeführt. Da die Warteschlangen in Echtzeit abgearbeitet werden, ist sichergestellt, dass kein Telegrammverlust auftritt. Eine Telegrammübertragung am KNX™ Bus ist allerdings bestenfalls alle 50 ms theoretisch möglich, sodass der Anwender eine parametrierbare Pausenzeit nach der Verarbeitung 1 und 50 ms einstellen kann.

Über das Menü Optionen – Performance können Sie die Einstellungen verändern. Damit wird im Anwendungsprogramm folgende Sektion eingefügt, die auch direkt manipuliert werden kann:

Hinweis: Die „Busantwortzeit“ ist mit der Einführung des Echtzeitwebservers nicht mehr einzustellen. Aus Kompatiblitätsgründen ist diese Einstellung jedoch noch vorhanden, wird aber ignoriert.

```
[Performance]
// Performance-Einstellungen: Zykluszeit, Busantwortzeit, Autorefresh
15
1500
10
```

Die Webserverrefreshzeit bezieht sich ausschließlich auf das Nachladen externer Bildquellen.

Autovervollständigung

Mit der Tastenkombination STRG+Leertaste (Space) bietet die automatische Vervollständigung von Ausdrücken schon nach dem Eintippen der ersten Buchstaben eine Liste der passenden Funktionen und Definitionen wie z.B. in Abbildung 84 gezeigt.

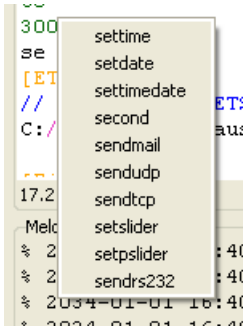


Abbildung 84: Autovervollständigung nach Eingabe „s“ und STRG-Leertaste („Space)

Syntaxhervorhebung

Daneben kann der integrierte Editor die Syntax der Schreibweise hervorheben. Beim Speichern des Programms geschieht dies immer, sofern die Funktion nicht im Menu ANSICHT deaktiviert wurde.

Während der Eingabe eines neuen Anwenderprogramms kann die Syntaxhervorhebung einfach durch STRG+UMSCHALT+D initiiert werden.

Alternativ können Sie im Menü BEARBEITEN auch die Syntaxhervorhebung nach Eingabe eines Zeichens automatisch erfolgen.

Direktiven

#include

Direktive

- #include *Datei*

Wirkung

- Bindet eine Datei (Pfad und Dateiname) an diese Stelle im Programm ein
- Eine eingebundene Datei darf ihrerseits auch mit #include weitere Dateien rekursiv einbinden
- #include bewirkt ein Einbinden der externen Datei an der Stelle im Code, wo die #include steht. Die Sektionen werden dabei nicht gesondert berücksichtigt, d.h. es wird die Datei direkt eingebunden.Wenn durch das Einbinden von mehreren Dateien eine Sektion doppelt oder mehrfach im Code generiert wird, wird nur die erst genannte Sektion ausgewertert

#break_if_older_version

Direktive

- #break_if_older_version *Nummer*

Wirkung

- Der Compiler (EibParser) stoppt das Compileren, wenn die *Nummer* kleiner als dessen Version ist

#addto

Direktive

- #addto [EibPC]
- #addto [Macros]
- #addto [WebServer]

Wirkung

- Der Compiler (EibParser) fügt die folgenden Zeilen der Sektion, deren Ende entweder durch eine neue Sektion oder eine weitere #addto Direktive vorgegeben ist, ein.

Beispiel 1

```
[EibPC]
a=1
[Macros]
mymacro(a)
#addto [EibPC]
b=a+1
```

Beispiel 2

```
[WebServer]
page(1) [$EG$,$Küche$]
button(2) [LIGHT] $Text$
[EibPC]
a=1
#addto [WebServer]
page(3) [$EG$,$Keller$]
button(2) [LIGHT] $Text$
```

#define

Direktive

- #define *String*

Wirkung

- Ein Symbol (String) für den Präprozessor definieren

Hinweis:

- Der Compiler kennt dieses Symbol nur in Verbindung mit #ifdef Direktiven
- Der Compiler unterscheidet div. Betriebssysteme

```
[ETS-ESF]
// Windows
#ifdef WIN
z:/diskstation/public/knx/test.lib
#endif
//Linux
#ifdef LIN
/share/diskstation/public/knx/test.lib
#endif
//OSX
#ifdef OSX
smb://diskstation/public/KNX/test.lib
#endif
```

#undef

Direktive

- #undef *String*

Wirkung

- Ein Symbol (String) für den Präprozessor löschen

#ifdef

Direktive

- #ifdef *String*

Wirkung

- Der nachfolgende Codeabschnitt wird bis zur nächsten im Code enthaltenen *#endif* Direktive nur compiliert, wenn der *String* definiert ist.

#ifndef

Direktive

- #ifndef *String*

Wirkung

- Der nachfolgende Codeabschnitt wird bis zur nächsten im Code enthaltenen *#endif* Direktive nur compiliert, wenn der *String* nicht definiert ist.

#endif

Direktive

- #endif

Wirkung

- beendet die #ifdef- Direktive

Oberfläche

Auf einen Blick

– übersichtlich und strukturiert

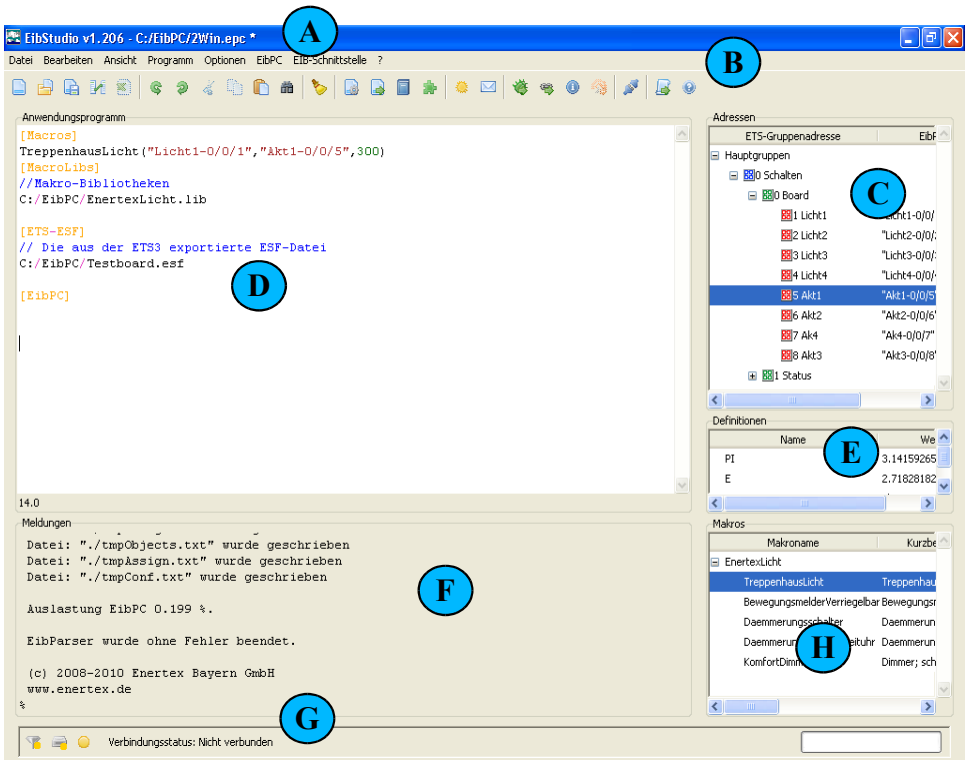


Abbildung 85: Oberfläche Enertex® EibStudio unter Windows XP®

Legende zu Abbildung 84:

- A) Menüleiste - Zugriff auf alle Funktionen
- B) Toolbar – schneller Zugriff auf wichtige Funktionen
- C) Fensterbereich für Importadressen - diese sind per Copy & Paste bzw. Drag&Drop in das eigene Projekt einzufügen
- D) Editor - mit Syntaxhervorhebung und Autovervollständigung
- E) Fensterbereich für vordefinierten Konstanten --diese sind per Copy & Paste in das eigene Projekt einzufügen
- F) Fensterbereich für Meldungen die vom EibParser während des Kompilierens erzeugt werden
- G) aktueller Verbindungsstatus
- H) Makros aus den geladenen Makrobibliotheken

Menüleiste

Schaltfläche	Short-Cut (Win/Lin)	Kurzbeschreibung
Datei -		
Neu ...	Strg+N	Erstellt ein neues Anwenderprogramm und lädt die Adressen der ausgewählten ETS-Exportdatei
Öffnen ...	Strg+O	Öffnet ein bestehendes Anwenderprogramm
Speichern	Strg+S	Überschreibt die vorher geöffnete oder zuletzt gespeicherte Datei
Speichern unter ...		Speichert das Anwenderprogramm in eine anzugebende Datei
Gruppenadressen aus ETS-Exportdatei importieren ...		Importiert die Adressen der ausgewählten ETS-Exportdatei
Visualisierungsassistenten öffnen		Öffnet den Visualisierungsassistent
Beenden	Strg+Q	Beendet die Software
Bearbeiten -		
Rückgängig	Strg+Z	Macht den letzten Arbeitsschritt rückgängig
Wiederherstellen	Strg+Umschalt+Z	Stellt den letzten Arbeitsschritt wieder her
Ausschneiden	Strg+X	Schneidet eine markierte Textstelle aus
Kopieren	Strg+C	Kopiert eine markierte Textstelle
Einfügen	Strg+V	Fügt eine markierte Textstelle ein
Selektion kommentieren	Strg+d	Kommentiert die aktuelle Selektion im Anwendungsprogramm
Selektion entkommentieren	Strg+Umschalt+d	Entfernt Kommentarzeichen die aktuelle Selektion im Anwendungsprogramm
Adressen kopieren		Kopiert die im Adressfenster markierte Adresse
Definition kopieren		Kopiert die im Definitionsbereich markierte Variable
Makro hinzufügen		Fügt ein Makro ein
Suchen	Strg+F	Suchfunktion
Ersetzen	Strg+G	Ersetzen Funktion
Ansicht -		
Meldungsfenster leeren		Entfernt den Text im Meldungsfenster
Begriffe automatisch hervorheben		Begriffe werden während des Schreibens hervorgehoben – syntax highlighting
Begriffe hervorheben		Bei deaktivierter automatischer Hervorhebung gibt es die Möglichkeit, die Syntax-Hervorhebung für den schon bestehenden Text zu aktivieren
Schriftgröße 8		Ändert die Schriftgröße im Editor auf 8
Schriftgröße 9		Ändert die Schriftgröße im Editor auf 9
Schriftgröße 10		Ändert die Schriftgröße im Editor auf 10

Schriftgröße 11		Ändert die Schriftgröße im Editor auf 11
Schriftgröße 12		Ändert die Schriftgröße im Editor auf 12
Schriftgröße 13		Ändert die Schriftgröße im Editor auf 13
Schriftgröße 14		Ändert die Schriftgröße im Editor auf 14
Schriftgröße 15		Ändert die Schriftgröße im Editor auf 15
Schriftgröße 16		Ändert die Schriftgröße im Editor auf 16
Programm -		
Programm kompilieren	F7	Das Anwenderprogramm wird mit dem EibParser kompiliert; im Meldungsfenster erscheint ein Statusbericht
Programm kompilieren, an EibPc senden und starten	F3	Das Anwenderprogramm wird mit dem EibParser kompiliert, an den EibPc gesendet und gestartet; im Meldungsfenster erscheint ein Statusbericht
Programm im EibPC zurücksetzen		Das vorhanden Programm wird gelöscht und durch ein leeres Programm ersetzt
Makro-Bibliotheken ...		Makro-Bibliothek hinzufügen
Makros...		Makro hinzufügen
Optionen -		
IP Adresse EibPC einstellen ...		Die bestehende Netzwerkkonfiguration kann bearbeitet werden
EibStudio® Einstellungen laden ...		Läd eine *.set Datei, in der die Netzwerkkonfigurationen gespeichert sind
EibStudio® Einstellungen speichern		Überschreibt die vorher geöffnete oder zuletzt gespeicherte Datei
EibStudio® Einstellungen speichern unter ...		Speichert die aktuelle Netzwerkkonfiguration in eine anzugebende Datei
Koordinaten für Sonnenstandberechnung einstellen		Koordinaten für Sonnenstandberechnung einstellen
English		Deutsche Sprache aktivieren
Deutsch		Englishe Sprache aktivieren
Email Einstellungen ...		Daten für den E-Mail Versand eingeben
VPN Konfiguration		Daten für VPN eingeben
WebServer Passwortschutz Konfiguration		Daten für HTTPS eingeben
Performance Einstellungen ...		Daten für den Performance Einstellungen eingeben
RS232 Einstellungen ...		Konfiguration der RS232 Schnittstelle, wenn nicht als EIB interface gebraucht
FTP Einstellungen ...		Konfiguration für Auslagerung vaon Telegrammdaten auf einem FTP
NTP-Zeitsynchronisation		Konfiguration für NTP
Auf Updates prüfen ...		Verbindet mit dem Internet und sucht automatisch nach neuen Updates

Automatisch auf Updates prüfen		Option, die bei jedem Neustart von Eibstudio auf neue Updates prüft
Enertex® EibPC -		
Netzwerkeinstellungen übertragen		Die Netzwerkeinstellungen werden an den EibPC gesendet
Netzwerkeinstellungen zurücksetzen		Netzwerkeinstellungen werden in den Auslieferzustand zurückgesetzt
DNS-Server ...		Angabe eines DNS-Servers im Ipv4 Format
DNS-Server prüfen ...		Der angegeben DNS-Server wird überprüft und eine entsprechende Meldung ausgegeben
Zeitzone ...		Zeitzone für den Enertex® EibPC kann verändert werden
Zeitzone abfragen		Aktuell eingestellte Zeitzone des EibPCs wird im Meldungsfenster ausgegeben
Uhrzeit und Datum ...		Uhrzeit und Datum für den Enertex® EibPC® kann verändert werden
Uhrzeit und Datum abfragen		Aktuell eingestellte Uhrzeit und Datum wird im Meldungsfenster ausgegeben
Ereignisspeicher auslesen		Den Ereignisspeicher auslesen
Variablen vorgeben und abfragen ...		Der aktuelle Wert eines Eib-Objekts oder einer Variable wird abgefragt und im Meldungsfenster angezeigt Der Wert eines Objektes oder einer Variable kann verändert und auf den KNX™ Bus geschrieben werden
EIB-Telegramme abholen ...		EIB-Telegramme werden vom EibPC abgeholt und in einer CSV-Datei abgespeichert
EIB-Telegramme vom FTP abholen ...		Binärdateien mit gespeicherten Telegrammen werden vom FTP-Server abgeholt und als CSV-Datei exportiert
EIB-Telegramme zyklisch abholen ...		EIB-Telegramme werden zyklisch abgeholt
EIB-Telegramme löschen ...		Alle im EibPC gespeicherten EIB-Telegramme werden gelöscht
Anzahl der EIB-Telegramme abfragen		Fragt die Anzahl der im Enertex® EibPC gespeicherten Telegramme ab
Aktuelle EIB-Telegramme beobachten	F2	Gibt die aktuellen Telegramm im Meldungsfenster aus
Aktuelle EIB-Telegramme filtern		Aktuelle EIB-Telegramme können vor der Ausgabe gefiltert werden
Filter einrichten ...		Filter für EIB-Telegramme einrichten
Szenen löschen ...		Alle im Enertex® EibPC gespeicherten Szenen werden gelöscht
Neu starten ...		Der Enertex® EibPC wird neu gestartet
Firmware aktualisieren ...		Ein Firmwareupdate wird an den Enertex® EibPC gesendet
Firmwareversion abfragen		Die aktuelle Firmwareversion, Seriennummer und eingespielte Patches werden ausgegeben
Freischaltcode übertragen		Ein Freischaltcode wird an den Enertex® EibPC übertragen
Freigeschaltete Funktionen abrufen		Übersicht über freigeschaltete Funktionen
Patch übertragen ...		Ein Patchupdate wird an den Enertex® EibPC gesendet
Übertragung beenden		Die Übertragung an den Enertex® EibPC wird manuell beendet
EIB-Schnittstelle -		
Auswählen		Dialog zur Auswahl und Konfiguration der Eibschnittstelle (FT1.2 oder EibNet/IP)

Verbindung aufbauen		Baut eine unterbrochene Verbindung zur EibNet/IP Schnittstelle auf
Verbindung trennen		Unterbricht die Verbindung einer EibNet/IP Schnittstelle mit dem EibPC
Verbindungsstatus abfragen		Erfragt den Status der Verbindung zur EibNet/IP Schnittstelle
? -		
Hilfe	F1	Öffnet das Handbuch
Packe Informationen für Supportanfrage		Erstellt eine komprimierte Datei mit allen für einen effizienten Support nötigen Firm- und Softwaredaten
Info		Versionsinformation des Enertex® EibStudio, Enertex® EibPC

Netzwerkeinstellungen

Einschalten

Nach Anschluss der Stromversorgung und des Netzwerkkabels ist der Enertex® EibPC nach ca. 2-3 Minuten betriebsbereit und antwortet auf Netzwerkanfragen. Zur Installation und Betriebsanzeige (eingebaute LED) siehe auch Seite 18 ff.

DHCP

Sie haben die Möglichkeit dem Enertex® EibPC eine feste IP-Adresse in ihrem Heimnetz zuzuweisen oder per DHCP Server zu arbeiten. Im Auslieferungszustand ist der Enertex® EibPC auf DHCP eingestellt. Wenn er keinen DHCP Server findet, gibt er sich selbst eine freie IP-Adresse, dazu muss der Enertex® EibPC über ein CrossOver Kabel mit einem PC verbunden sein.

Wenn die Firewall passend eingerichtet ist, können Sie die IP-Adresse im Menü Optionen → Netzwerkeinstellungen wie in Abbildung 87 abfragen, indem Sie auf die Schaltfläche „Automatisch“ klicken. Im Fenster „Meldungen“ sehen Sie nun die Antwort des Enertex® EibPC. Sie können anschließend den Enertex® EibPC wie in aber auch mit fester IP Adresse konfigurieren (s.u.)

Firewall freischalten

Wichtig: Eine aktive Firewall verhindert u.U. die Kommunikation zwischen Enertex® EibStudio und Enertex® EibPC und muss entsprechend konfiguriert werden. Details hierzu finden Sie in Ihrer Anleitung der Firewall. Schalten Sie dazu den Kommunikationsport 4805 und 4806 für UDP-Telegramme in Ihrer Firewall frei. Wenn Sie UDP-Telegramme versenden wollen (siehe Seite 245), müssen sie zusätzlich 4807 öffnen.

Alternativ: Bei einer Windows®-Firewall können Sie auch direkt dem Programm nconf.exe (Teilprogramm von Enertex® EibStudio) beispielsweise eine Ausnahme einrichten. Enertex® EibStudio müssen Sie dazu mindestens einmal gestartet haben. Sie finden nconf.exe nach dem Aufruf von Enertex® EibStudio unter Ihren Windows®-Nutzerverzeichnis wie in Abbildung 86 angegeben. Um dies einzurichten wählen Sie im Dialog der

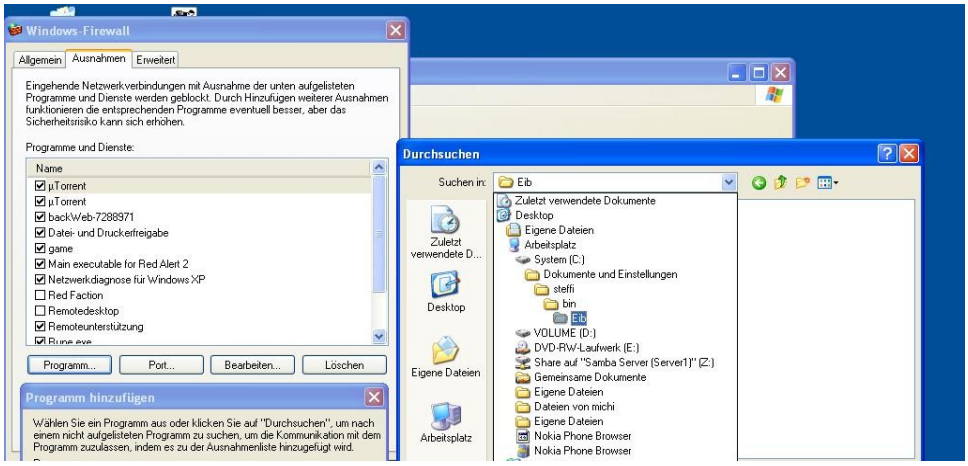


Abbildung 86: Konfiguration Windows®-Firewall

Firewall *Ausnahmen-Programm*, im Unterdialog *Programm hinzufügen,*dort *Durchsuchen* und geben ihren Pfad entsprechend „C:\Dokumente und Einstellungen\IHR NUTZERNAME\bin\Eib\nconf.exe“ an.

Probleme mit Firewalls

Kaspersky unter Windows 7

- Wenn die Kaspersky Firewall aktiviert und die Windows Firewall deaktiviert ist, kann das Eibstudio nicht senden
- Wenn die Kaspersky Firewall und die Windows Firewall deaktiviert ist, kann das Eibstudio nicht senden
- Wenn die Kaspersky Firewall deaktiviert und die Windows Firewall aktiviert ist, kann das Eibstudio senden

Das selbe Verhalten wurde mit Windows Vista und Norton 360 geschildert.

Eingebauter DHCP-Ersatz

Wenn Sie keinen DHCP-Server im Netzwerk haben, sucht sich der Enertex® EibPC nach dem Bootvorgang selbst eine freie Netzwerkadresse. Sie können diese im Menü Optionen → Netzwerkeinstellungen wie in Abbildung 87 abfragen, in dem Sie auf die Schaltfläche „Automatisch“ klicken. Im Fenster „Meldungen“ sehen Sie nun die Antwort des Enertex® EibPC, welche direkt in den Dialog Abbildung 87 eingetragen werden. Sollten Sie im Meldungsfenster eine Fehlermeldung lesen, so liegt das an ihrer Firewall.

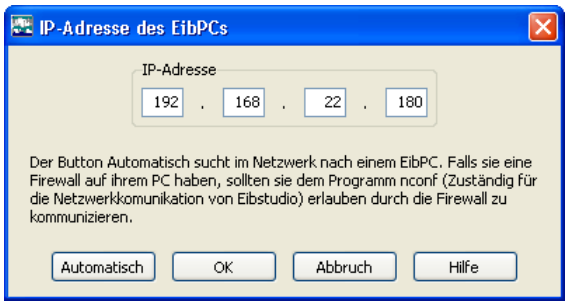


Abbildung 87: Netzwerkeinstellungen eingeben

Feste IP

Wenn Sie den Enertex® EibPC auf eine feste IP Adresse konfigurieren wollen, müssen Sie zunächst entweder mit DHCP oder eingebautem DHCP-Ersatz (z.B. bei Direktverbindung zu Ihrem PC) den Enertex® EibPC ansprechen (siehe Optionen → Netzwerkeinstellungen, Dialog Abbildung 87 und Schaltfläche „Automatisch“). Dann, wenn sich bereits mit dem Enertex® EibPC verbunden haben, können Sie auch die Netzwerkeinstellungen des Enertex® EibPC problemlos ändern. Klicken auf im Menü EibPC → Netzwerkeinstellungen des EibPCs ändern wie in Abbildung 88 gezeigt. Es sollte der rechte Dialog mit der Standardeinstellung aufgehen. Hier tragen Sie nun ihre gewünschten Daten ein.

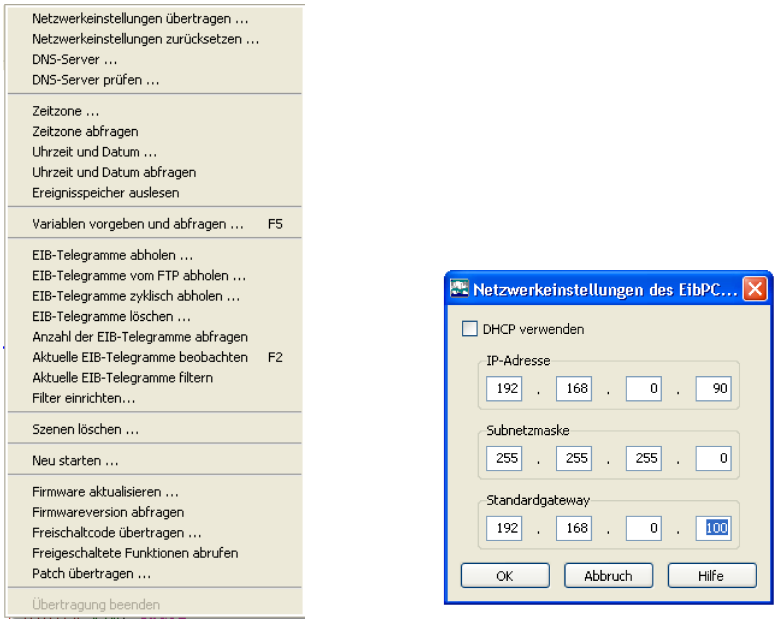


Abbildung 88: Netzwerkeinstellungen eingeben

Um die Adresse auch für den Enertex® EibPC zu übernehmen, müssen die Daten an den Enertex® EibPC übermittelt werden. Danach startet der Enertex® EibPC neu (2-3 min Bootvorgang). Wenn Sie dem Enertex® EibPC eine neue Adresse zugewiesen haben, müssen diese anschließend wieder im Enertex® EibStudio bekannt machen. Hier tragen Sie nun unter Optionen → Netzwerkeinstellungen, Dialog Abbildung 87, die neuen Daten ein.

IP-Adresse = Adresse des Enertex® EibPC, muss eine Adresse des neuen Heimnetzes sein
Netzwerkmaske = Ihre Netzmaske, meist 255.255.255.0
Standard-Gateway = Netzwerkadresse des Routers. Falls kein Router vorhanden ist, dann muss hier die IP-Adresse des EibPCs eingetragen werden.

Wenn der Enertex® EibPC die Daten übernommen hat, muss dieser neu gebootet werden. Dies dauert ca. 2 Minuten.

Netzwerkeinstellungen speichern und öffnen

Eine neue Netzwerkkonfiguration sollte für das Enertex® EibStudio in einer Datei mit Endung „set“ gespeichert werden. Dazu steht ihnen der Dialog Optionen → Netzwerkeinstellungen Speichern zur Verfügung.

Eine bestehende Netzwerkkonfiguration öffnen Sie mit dem entsprechenden Dialog im Datei-Menü. Enertex® EibStudio speichert diese Datei per default mit Endung „.set“, wie in Abbildung 89 ersichtlich.

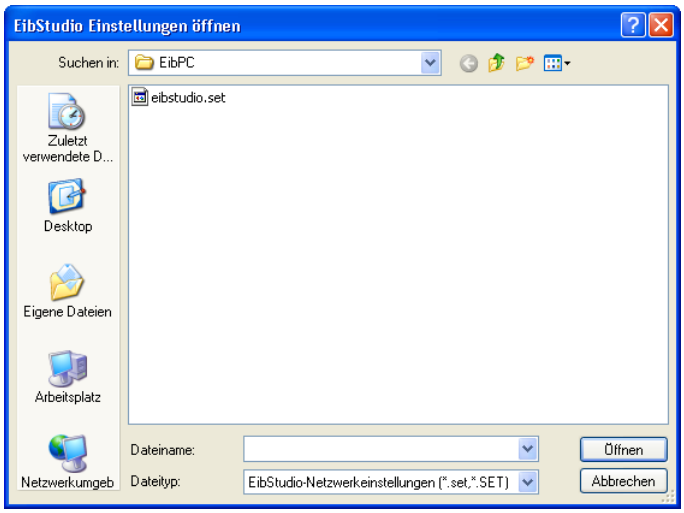


Abbildung 89: Netzwerkeinstellungen öffnen

DNS-Server

Wenn der Enertex® EibPC eine Internetverbindung aufbauen muss, so verbindet sich der Enertex® EibPC zu einem DNS-Server zum Zwecke der Namensauflösung. Sie können die Defaultverbindung ändern. Hierzu steht der Dialog Abbildung 90 zur Verfügung.

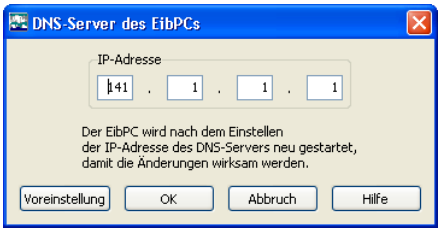


Abbildung 90: DNS Server einstellen

Wenn Sie überprüfen wollen, ob der Enertex® EibPC die Daten übernommen hat, und ob der DNS Server erreichbar ist, so klicken Sie EibPC → Dns Server Überprüfen und achten Sie auf das Fenster „Meldungen“.

Werkseinstellungen

Die Werkseinstellungen lassen sich mit Hilfe des Resetknopfes (zur Benutzung siehe Seite 20) zurücksetzen. In diesem Fall werden auch eventuell gespeicherte Szenen und das Anwenderprogramm gelöscht.

Mit der Funktion EibPC → Netzwerkeinstellungen zurücksetzen lassen sich ebenso die Netzwerkeinstellungen wieder herstellen. Bei dieser Funktion muss der Enertex® EibPC durch Unterbrechen der Stromversorgung neu gestartet werden. Beachten Sie dazu die entsprechende Dialogmeldung.

E-Mail-Einstellungen (Option NP)

Wenn Sie im Besitz der Zusatzoption NP sind, können Sie auch mit dem Enertex® EibPC Mails versenden. Hierzu müssen Sie die Freischaltcodes in den Enertex® EibPC einspielen (siehe Seite 245) und einen SMPT-Zugang einrichten, von dem der Enertex® EibPC aus mailen kann. Zudem sollten Sie den DNS Server (s.o.) eingerichtet haben und der Enertex® EibPC über eine Online-Verbindung zum Internet verfügen.

Die E-Mail Einstellungen können über den Dialog Optionen → E-Mail Einstellungen wie in Abbildung 91 gezeigt konfiguriert werden.

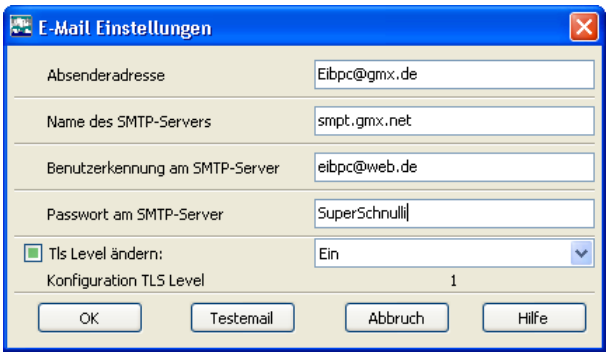


Abbildung 91: E-Mail einrichten

Die eingegeben Daten erscheinen nach dem Klicken der Schaltfläche „OK“ im Anwenderprogramm, wie in der Abbildung 92 angedeutet und können dort auch direkt editiert werden.

Abbildung 92: E-Mail einrichten

KNX™ Schnittstelle konfigurieren

Über das Menü EIB-Schnittstelle kann die KNX™ Schnittstelle gewählt werden, über die der Enertex® EibPC auf den KNX™ Bus zugreift.

Abbildung 93: Schnittstelle einrichten

- Bei einer FT 1.2 müssen Sie die Inbetriebnahmehinweise auf 16ff. beachten. Sie können die FT1.2 Schnittstelle im Busmonitormodus oder im Gruppenmonitormodus betreiben. Im letzteren quittiert die Schnittstelle sämtliche KNXTM-Telegramme. Der Gruppenmonitormodus ist grundsätzlich fehlertoleranter gegen Busfehler. Beachten Sie hierzu auch die Hinweise auf S.147. Falls die Schnittstelle vor dem Betrieb im Gruppenmonitormodus im Busmonitormodus eingesetzt wurde, muss sie vor dem Übertragen des Anwendungsprogramms durch ein kurzes Unterbrechen ihrer Stromversorgung zurückgesetzt werden.

- Bei einer IP - Schnittstelle müssen Sie die Inbetriebnahmehinweise auf 16ff. beachten und zudem im Dialog die entsprechenden Daten Ihrer Schnittstelle eingeben. Die Schnittstelle wird erst nach Überspielen und anschließendem Starten eines Anwendungsprogramm angesprochen und aktiviert.
  Der Enertex® EibPC nutzt das sog. Tunneling der IP Schnittstelle. Diese wird dadurch exklusiv belegt, d.h. Sie können die Schnittstelle in diesem Modus dann nicht von der ETS aus ansprechen. Um dennoch einen kurzfristigen Zugriff auf die Schnittstelle zu ermöglichen (z.B. Programmierung kleiner Änderungen der Gruppenadressen) können Sie die Schnittstelle über den Menüpunkt Verbindung trennen und Verbindung aufbauen vom Enertex® EibPC trennen bzw. mit diesem verbinden.

- Mit der Telegrammratenbegrenzung, die standardmäßig auf 7 Telegramme pro Sekunde steht, verhindern Sie ein „Überfahren“ des KNX™ Busses und gewährleisten stabilen Betieb Ihrer Installation.

KNX™ Busfehler

Grundsätzlich kann der Anwender mit dem Busmonitor der ETS und bei Einsatz einer FT1.2 Schnittstelle überprüfen, ob diese ungestört funktioniert. Wenn dies der Fall ist, kann davon ausgegangen werden, dass ein fehlerfreier Betrieb des Enertex® EibPC mit FT1.2 Schnittstelle gewährleistet wird.

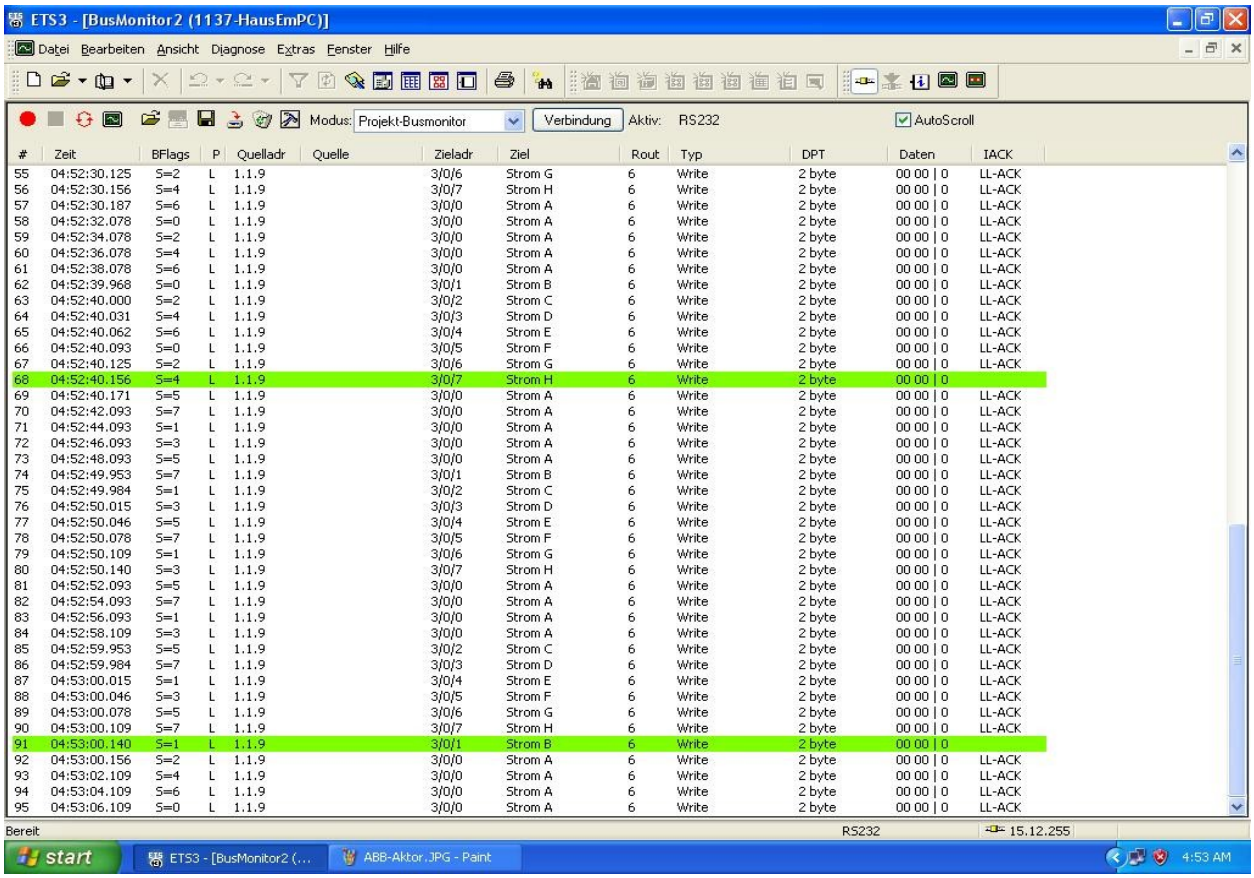


Abbildung 94: Probleme bei zu hohem Aufkommen an Telegrammen mit FT1.2 Schnittstelle

Um zu testen, ob ihr Projekt so viele Telegramme generiert, dass auf der Hardwareebene bereits Telegramme verloren gehen, gehen Sie wie folgt vor:

1. Schließen Sie zunächst die FT1.2 Schnittstelle die ETS an.
2. Setzten Sie das Projekt mit der ETS in den **Busmonitormodus** und beobachten diesen für einige Minuten. **Achten Sie darauf, dass Sie nicht den Gruppenmonitor in der ETS nutzen**
3. Wenn der Busmonitor der ETS Telegramme zeigt, welche in grün gefärbte Telegramme sind (Abbildung 94), ist die Installation nicht unproblematisch. Reduzieren Sie in diesem Fall das Aufkommen der Telegramme, indem der entsprechende Aktor mit Telegrammratenbegrenzung parametriert wird.
4. Wenn nach einiger Zeit derartige Telegramme nicht auftreten, haben Sie bezogen auf das Datenaufkommen auf dem KNX™-Bus eine fehlerfreie Installation und können nun die FT1.2 Schnittstelle mit dem Enertex® EibPC verbinden.

Der Busmonitor des Enertex® EibPC

Verbindungsanzeige und Statusbericht

Mit dem Menü EibPC → EIB-Telegramme abholen können Sie die Netzwerkverbindung überprüfen. Ist diese Option gesetzt, wird der aktuelle Verbindungsstatus im linken, unteren Fensterbereich angezeigt. Alternativ genügt ein Klick auf die Schaltfläche . Wenn der Enertex® EibPC mit dem Enertex® EibStudio verbunden ist, wechselt die Schaltfläche zu  und in der Statusanzeige wechselt der Status von „?“ zu „Verbunden mit 192.168.22.133“ (wenn Letzteres die eingestellte Netzwerkadresse des Enertex® EibPC ist).

Verbindungsstatus ganz unten links

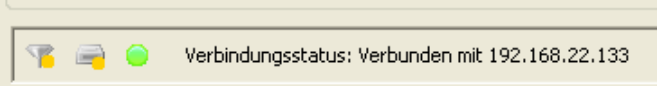


Abbildung 95: Verbindungsstatus

Zwei Busmonitore

Der Enertex® EibPC verfügt über zwei Ringspeicher. Der größere von beiden speichert bis zu 500.000 Telegramme und ist für Langzeituntersuchungen gedacht. Dabei muss kein weiterer PC beim Aufzeichnen mit dem Enertex® EibPC verbunden sein. Der zweite Ringspeicher für 10 Telegramme ist für die Beobachtung des momentanen Busverkehrs gedacht. Hier haben Sie die Möglichkeit, nach bestimmten Absenderadressen (Geräten) und Gruppenadresse zu filtern und ggf. nur für Sie relevante Daten abzufragen.

Wird ein neues Anwenderprogramm überspielt, so werden beide Ringspeicher gelöscht.

Um die im Enertex® EibPC aufgelaufenen max. 500.000 Telegramme abzuholen, klicken Sie im Menü EibPC auf EIBTelegramme Abholen. In der Statuszeile links unten wird Ihnen der Fortschritt beim Abholen der Telegramme angezeigt. Die Geschwindigkeit, mit der die Telegramme übertragen werden, hängt zum größten Teil von der Performance ihres PCs ab.

Busdaten speichern

Wenn alle Telegramme abgeholt sind bzw. Sie die Übertragung unterbrechen, erscheint ein Dialog, mit welchem Sie diese Telegramme in eine CSV-Datei abspeichern können. Diese CSV Datei stellt eine Textdatei dar, im welcher die Daten in Tabellenform abgelegt werden. Dabei sind die einzelnen Daten durch Kommas getrennt. Sie können die Daten dann in Microsoft® Excel oder OpenOffice® Calc oder einem Tabellverarbeitungsprogramm Ihrer Wahl importieren. Für die einfache Darstellung eignet sich auch das Windows Programm „CSView“.

Typische Busdaten

	A	B	C	D	E	F	
1	Datum/Uhrzeit	Absender	Empfaenger	Datentyp	Wert	Telegrammtyp	Telegrammdaten
2	2009-07-17-13:43:18	EibPC	"BewässerungAnzeige-2/0/11"	Text	Leer	Schreiben	bc 00 00 10 0b ef 00 80 32
3	2009-07-17-13:43:18	EibPC	"BewässerungAnzeige-2/0/11"	Text	2 Tagen	Schreiben	bc 00 00 10 0b ef 00 80 32
4	2009-07-17-13:43:19	EibPC	"BewässerungAnzeige-2/0/11"	Text	2 Tagen	Schreiben	bc 00 00 10 0b ef 00 80 32
5	2009-07-17-13:43:19	EibPC	"Wasser123Auto-2/0/10"	Binärwert	EIN	Schreiben	bc 00 00 10 0a e1 00 81 00
6	2009-07-17-13:43:24	EibPC	"RegenMeldung-5/2/2"			Lesen	bc 00 00 2a 02 e1 00 00 00
7	2009-07-17-13:43:24	1/1/49	"RegenMeldung-5/2/2"	Binärwert	AUS	Antworten	bc 11 31 2a 02 e1 00 40 ea

Abbildung 96: Import der Aufzeichnungsdaten in OpenOffice

In Microsoft® Excel wählen Sie im Dialog Daten den Unterpunkt Importieren. In den dann folgenden Dialogen wählen Sie das Format am besten als „Text“ und die Textkennzeichnung als {keine}. Gleichermaßen gehen Sie bei OpenOffice® Calc vor. Auf diese Weise erhalten Sie Daten im Format von Abbildung 96.

Momentane Buskommunikation

Sie können zusätzlich auch die momentane Buskommunikation sichtbar zu machen. Dazu aktivieren Sie die Option “EIB-Telegramme abholen / Verbindungsstatus“ oder klicken Sie auf die Schaltfläche  . Wenn der Enertex® EibPC mit dem Enertex® EibStudio verbunden ist, wechselt die Schaltfläche zu  .

Im Fenster “Meldungen“ werden nun alle Telegramme ausgegeben, die gerade auf den Bus geschrieben wurden. Abbildung 97 zeigt einen typischen Auszug aus dem Meldungsfenster, der an sich selbsterklärend ist.

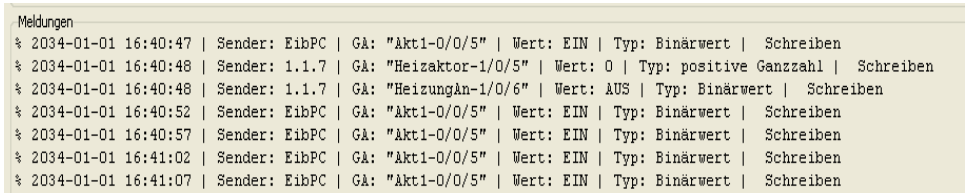


Abbildung 97: EIB-Telegramme abrufen

Wenn das Enertex® EibStudio gestartet wurde, kann auch der Busverkehr zyklisch in einer Datei gespeichert werden. Die Zykluszeit kann maximale 99 Tage betragen, d.h. es werden alle 99 Tage die Daten in eine csv-Datei abgespeichert. Der Dateiname ist in diesem Fall fest vorgegeben.

Autolog

Er ist nach dem Muster „autolog-2009-11-20-13-53-45.csv“ aufgebaut, also im Beispiel: Autolog, vom 20.11.2009, 13:53:45 Uhr.

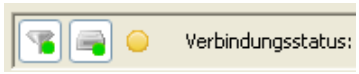


Abbildung 98: Autolog (2. Symbol) mit Filter aktiviert.

Sie können Filter einrichten, so dass nur Telegramme von bestimmten Gruppenadressen oder von bestimmten Geräteadressen (Absendern) angezeigt oder gespeichert werden. Dazu müssen Sie zunächst den Filter einrichten (siehe Abbildung 99), was Sie über den Menüpunkt EibPC – Filter einrichten einstellen können.

Abbildung 99: EIB-Telegramme filtern

Mit den Einstellungen in Abbildung 99 werden nur Telegramme angezeigt, die vom Absender mit der physikalischen Adresse 1/1/2 kommen und an die Ziel- Gruppenadresse 0/1/5 geschickt werden.

Möchten Sie alle Telegramme eines bestimmen Absenders beobachten, unabhängig vom Ziel, müssen Sie die Option Ziel deaktivieren (siehe Abbildung 100).

Abbildung 100: alle EIB-Telegramme eines Absenders filtern

Wenn der Filter aktiviert wird (EibPC – aktuelle EIBTelegramme Filtern), wechselt die Statusanzeige wie in Abbildung 101 andeutet. Der Filter wirkt sowohl auf Telegramme im Meldungsfenster, wie auch beim Abspeichern in eine CSV-Datei, wie auch in der Autolog-Funktion.

Abbildung 101: Filteranzeige rechts unten in der Statusanzeige, rechtes Bild: Filter aktiviert

Dabei haben Sie die Möglichkeit, Wildcards zu benutzen:
Das Fragezeichen „?“ steht für eine beliebige Ziffer, der Stern „*“ für eine beliebige Zahl.

Filter mit Wildcards

Schreibt man für das Ziel, letztes Eingabefeld „1/1/2**?**3“, so werden nur Telegramme mit Gruppenadresse 1/1/2**0**3, 1/1/2**1**3 ... 1/1/2**9**3 verarbeitet. Sie können beliebig viele Wildcards benutzen. Schreiben Sie 1/*/203 so werden alle Telegramme verarbeitet, deren Hauptgruppe 1 und Untergruppe 203 aufweist (beliebige Mittelgruppen).

Der Enertex® EibPC kann die Telegrammdaten auch an einem FTP Server hochladen. In diesem Fall werden nur Binärdaten an diesen verschickt. Die Konfiguration dieses FTP Dienstes können Sie das Menü Optionen – FTP-Transfer vornehmen.

Direktes Auslagern auf FTP

Alternativ können Sie die Einstellungen auch direkt im Anwendungsprogramm vornehmen, in dem Sie wie folgt eine Sektion [FTP] erstellen:

```
[FTP]
// FTP Server
192.179.169.67
//Benutzer
roman
//Passwort
praktikum
//Remote FTP-Verzeichnis
telegrams/MyLog
//Timeout
120
//Max. Puffer
3000
```

Analog dazu können Sie die Daten auch über einen Dialog unter Optionen – FTP Einstellungen (Abb. 102) eingeben. Die Daten werden nun zyklisch ausgelagert. Der Intervall wird entweder durch die Zeit oder dem Pufferfüllstand vorgegeben.

Im Dialog (Abb. 102) wird entweder in einem Zeitintervall von 60 Sekunden oder bei einem Pufferfüllstand von 3000 Telegrammen ausgelagert. Je nachdem welcher Wert eher eintrifft.

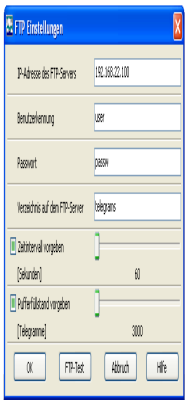


Abbildung 102: Eingabedialog FTP Einstellungen

Die Binärdaten können Sie anschließend im Enertex® EibStudio auswerten lassen, um so einen CSV-Export wie oben beschrieben zu generierten. Hierzu steht das Menü EibPC – EIB-TELEGRAMME VOM FTP ABHOLEN zur Verfügung.

Hinweis

Für das Umsetzen in das lesbare Format muss im Enertex® EibStudio das Anwendungsprogramm, welches die auf dem Enertex® EibPC die Telegramme generiert hatte, geladen werden.

Auslagern von FTP im Klartext

Auf S. 114 finden Sie ein ausführliches Beispiel, wie Sie auch Klartextinformationen („ASCII“) auf einem externen FTP-Server abgelegen können.

Anwenderprogramme kompilieren und überspielen

Ein Anwenderprogramm muss kompiliert, an den Enertex® EibPC übertragen und gestartet werden. Sie können diese Schritte wie in Abbildung 103 gezeigt im Menü PROGRAMM einzeln durchführen, die Tastenkombination STRG+UMSCHALT+B oder den Knopf drücken. Das Programm wird überspielt und gestartet. In der Statusanzeige (vgl. Abbildung 95) wird der Übertragungsvorgang dargestellt. Das Übertragen und Starten dauert je nach Programmgröße zwischen 1 und 5 Sekunden.

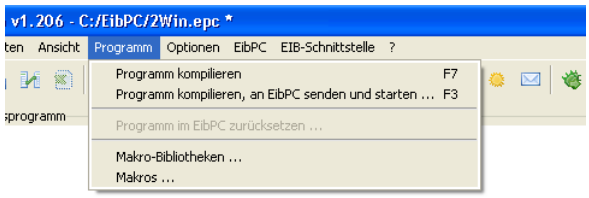


Abbildung 103: Anwenderprogramm kompilieren, an den EibPC senden und starten

ETS Adressen

Enertex® EibStudio kann Adressen aus Ihrem ets-Projekt direkt importieren. Näheres hierzu finden Sie auf Seite 158 beschrieben. Importierte Adressen werden im Fenster Adressen angezeigt, wie Abbildung 104 gezeigt. Per Drag&Drop oder Markierung Kopieren (STRG C) können Sie die Adressen auch über die Tastatur kopieren und anschließend im Anwendungsprogramm eingefügen.

Exportierte ets-Adressen laden

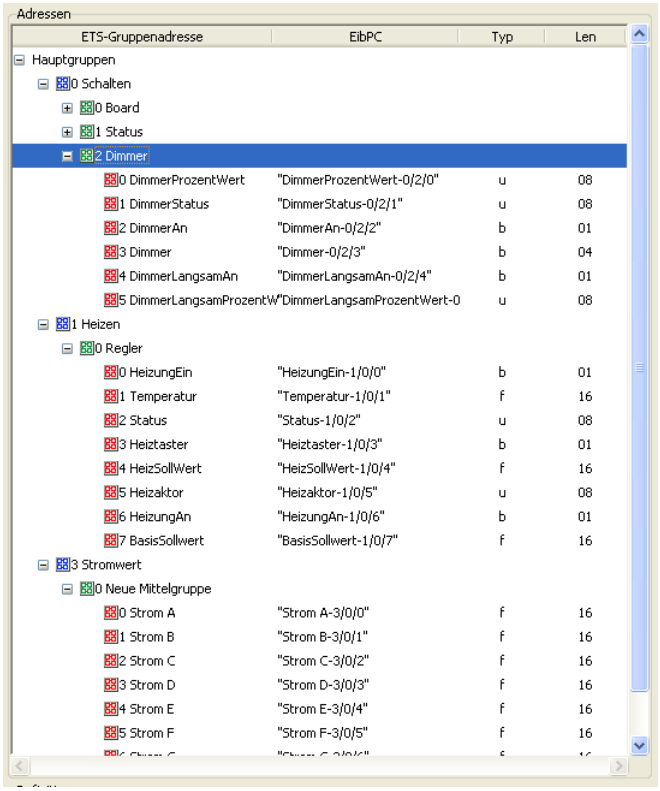


Abbildung 104: Adressen

Debugger

Um Ihr Programm zu „debuggen“, d.h. nach Fehlern zu suchen, können Sie

- Variablen
- Konstanten
- Manuelle Gruppenadressen
- Importierte Gruppenadressen

abfragen, Werte auf den Bus schreiben und den Wert von Variablen ändern.

Wichtig

Bevor Sie ein Programm oder Werte des Enertex® EibPC debuggen können, müssen Sie dieses an den Enertex® EibPC übertragen haben.

Zum Debuggen drücken Sie  oder alternativ im Menü EibPC – VARIABLEN VORGEBEN UND ABFRAGEN. Es erscheint ein Fenster (Abbildung 105) mit Variablen und importierten oder manuellen Gruppenadressen.

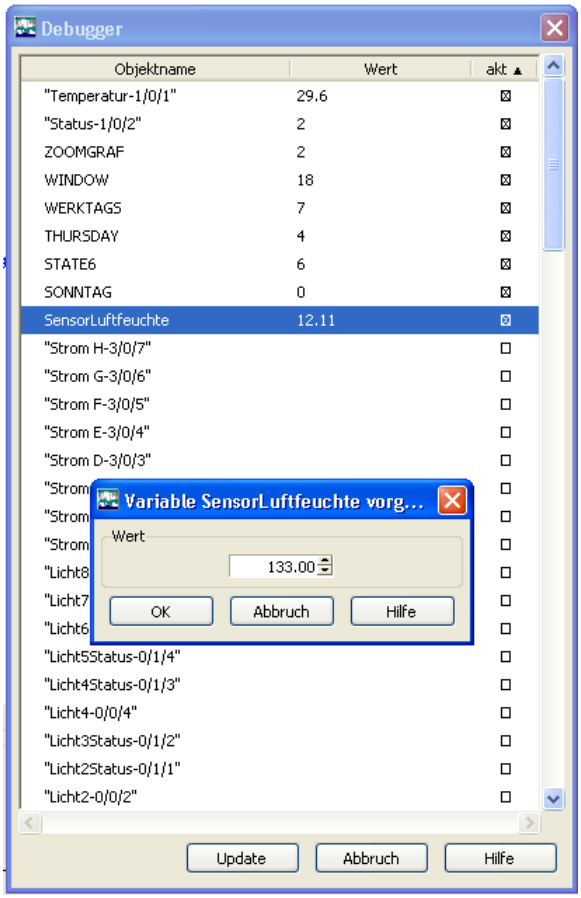


Abbildung 105: Debugger

Hier haben Sie die Möglichkeit Werte abzufragen und zu ändern.

Wenn Sie Gruppenadressen debuggen, wird der Wert angezeigt, welchen der Enertex® EibPC für die Gruppenadresse gespeichert hat (also keine Leseanforderung auf den Bus).

Im Meldungsfenster wird der abgerufene oder geschriebene Wert angezeigt.

Für die schnelle Fehlersuche

Wenn sie ein Objekt, d.h. eine Variable, Konstante oder Gruppenadresse, mit der rechten Maustaste anklicken, stehen ihnen drei Optionen zur Verfügung:

- Wert schreiben
- Wert vom Bus lesen (nur wenn eine Gruppenaddresse ausgewählt wurde)
- Wert aus EibPC lesen

Bei importierten Gruppenadressen können Sie direkt Werte auf den EIB schreiben oder lesen.

Dabei erkennt das Enertex® EibStudio selbstständig um welchen Datentyp es sich handelt und bietet Ihnen entsprechende Eingabemöglichkeiten an.

Variablen setzen und abfragen

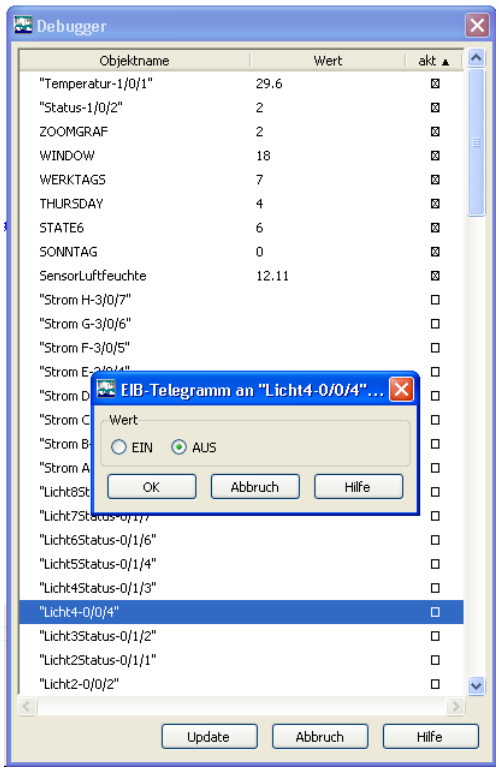


Abbildung 106: Debugger

Möchten Sie zum Beispiel, wie in Abbildung 106 dargestellt, für die Adresse “Licht1-0/0/1“ einen Wert auf den EIB schreiben, haben Sie die Wahl zwischen EIN und AUS, da es sich hier um eine Adresse mit binärem Datentyp handelt.

Mit der Option „Wert vom Bus lesen“ wird eine Leseanforderung auf den Bus geschrieben. Daraufhin antwortet der Busteilnehmer, der die entsprechende Gruppenadresse hat, mit seinem aktuellen Wert. Eibstudio wertet dieses Antwort-Telegramm aus und trägt den Wert in die Spalte *Wert* des Debuggers ein. Wenn der Busteilnehmer mit der entsprechenden Gruppenadresse nicht antwortet, dann wird in der Spalte Wert *nicht lesbar* eingetragen.

Mehrere Variablen abfragen

Mit dem Button *Update* ist es möglich mehrere Werte auf einmal vom Bus abzufragen. Dazu müssen sie z.B. wie in Abbildung 106 gezeigt, bei denjenigen Variablen Kreuze in der Spalte *akt*. setzen, die sie beobachten wollen. Drücken sie danach auf den Button *Update*, dann werden die Werte aller Variablen, die sie mit dem Kreuz markiert haben, automatisch aktualisiert.

Patch-Updates einspielen

Über das Menu EibPC - Patch übertragen lässt sich einfach per Knopfdruck ein neues Patchupdate für den Enertex® EibPC einspielen. Nach einem Update startet sich der Enertex® EibPC automatisch neu. Sobald die grüne Anzeige-LED wieder blinkt, ist der Enertex® EibPC betriebsbereit.

Updates werden ausschließlich von der Enertex® Bayern GmbH herausgegeben.

Firmware-Updates einspielen

Wie in Abbildung 107 sichtbar, lässt über das Menu EibPC sich einfach per Knopfdruck eine neues Firmwareupdate für den Enertex® EibPC einspielen. Nach einem Update startet sich der Enertex® EibPC automatisch neu. Sobald die grüne Anzeige-LED wieder blinkt, ist der Enertex® EibPC betriebsbereit.

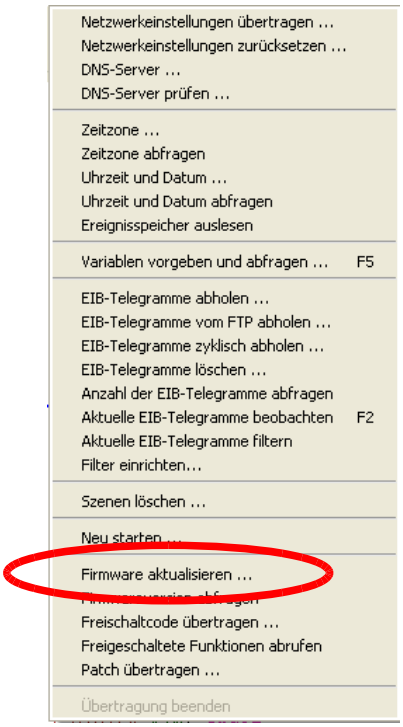


Abbildung 107: Firmwareupdate

Firmware-Version abfragen

Updates werden ausschließlich von der Enertex® Bayern GmbH herausgegeben. Sie können die Firmwareversion im Menu EibPC jederzeit abfragen. Die Version wird dann im Meldungsfenster angezeigt (siehe Abbildung 107).

Absturz

Sie können das Anwenderprogramm im EibPC zurücksetzen. Dadurch wird das vorhandene Programm gelöscht und der Enertex® EibPC in den Leerlauf gesetzt. Dieser Ablauf erfordert einen Hardwareneustart des Enertex® EibPC, weshalb die Stromversorgung zu unterbrechen ist.

Gehen Sie dabei wie folgt vor:

Wenn mal nichts mehr geht

1. Wählen Sie im Menü Optionen den Punkt „Anwenderprogramm zurücksetzen“ (vgl. Abbildung 103)
2. Beim folgenden Dialog drücken Sie auf „Weiter“
3. Unterbrechen Sie kurz die Stromversorgung des Enertex® EibPC
4. Warten sie ca. 5 min bis der Enertex® EibPC wieder betriebsbereit ist
5. Nun können Sie den Enertex® EibPC wieder mit einem neuen Anwenderprogramm bespielen (Seite 152)
6. Sollten Sie einen vom Standard abweichenden Aufstellungsort eingeben, wird zusätzlich beim nächsten Programmstart einmalig 2 min. Rechenzeit zum Erstellen der neuen Sonnendaten benötigt.

Resetknopf

Wenn auch dieses Vorgehen nicht mehr hilft, können Sie den Resetknopf (zur Benutzung siehe Seite 20) betätigen. In diesem Fall werden

- das Anwenderprogramm,
- Daten zum Aufstellungsort,
- Netzwerkeinstellungen

gelöscht bzw. auf den Auslieferungszustand zurückgesetzt. **Beachten Sie hierbei auch obigen Punkt 6.**

Meldungen

Das Meldungsfenster dient auch zur Anzeige von

- Daten des Busmonitor (Siehe Seite 148)
- Daten des Debuggers von Variablen (Siehe Seite 152)
- Meldungen des integrierten Compilers „EibParser“. Wenn Fehler beim Kompilieren auftreten, werden diese auch in diesem Fensterbereich angezeigt.

Busmonitor
Compiler
Datenübertragung

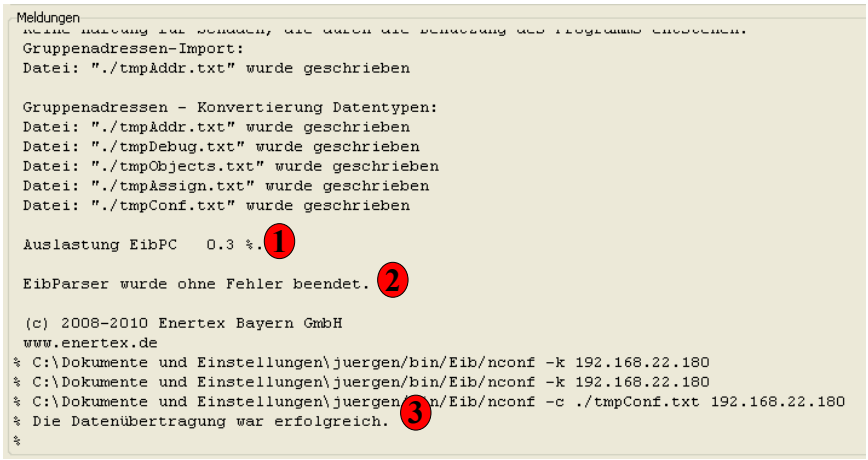


Abbildung 108: Meldungen

Statusberichte lassen sich im Meldungsfenster ablesen.

1) zeigt an, dass der Enertex® EibPC zu 0.3% vom Speicher ausgelastet ist.
2) zeigt an, dass das Anwenderprogramm ohne Fehler kompiliert wurde.
3) zeigt an, dass die Datenübertragung erfolgreich war.

Zeitzone

Enertex® EibPC = Zeitmaster
oder
KNX-BUS = Zeitmaster

Sobald der Enertex® EibPC eine Internetverbindung über Ihr Netzwerk herstellen kann, ermittelt er über das NTP-Protokoll die aktuelle Tageszeit und synchronisiert diese Zeit. Sie haben die Möglichkeit bis zu neun eigene NTP-Server anzugeben.

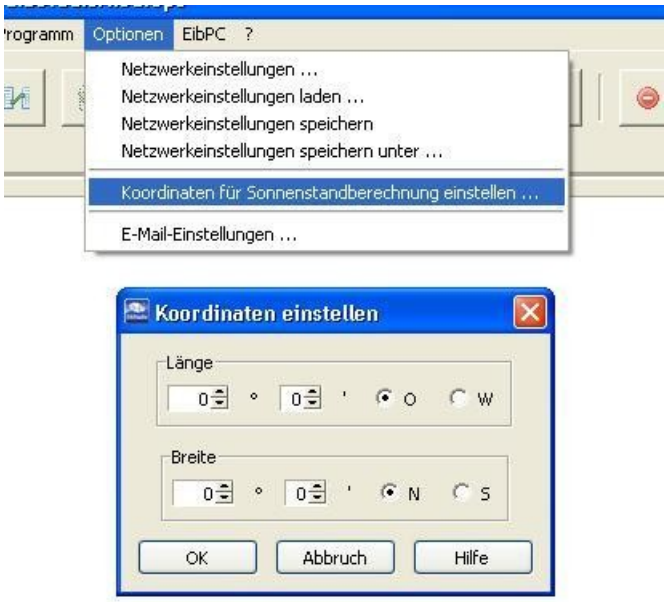
Die Zeitsynchronisation kann im Menü unter Optionen → NTP-Zeitsynchronisation auch unterbunden werden. Deshalb muss angegeben werden, ob das Anwendungsprogramm erst nach der Zeitsynchronisation gestartet werden soll oder sofort nach Betriebsstart. Standardmäßig startet das Anwendungsprogramm sobald der EibPC betriebsbereit ist.

Sie können die Zeitzone, in der Sie sich befinden im Menü EibPC mit dem Punkt *„Zeitzone des EibPC einstellen“* modifizieren.

Außerdem können Sie den Enertex® EibPC zum „Zeitmaster“ Ihrer KNX™-Installation machen, d.h. Sie schicken in regelmäßigen Zeitabständen die aktuelle Systemzeit und das Datum als KNX™-Telegramm an alle Busteilnehmer.

Wenn Sie über keine Internetverbindung verfügen, können Sie den Enertex® EibPC mit KNX™-Zeittelegrammen synchronisieren oder die Zeit manuell eingeben. Weiteres hierzu finden Sie auf Seite 180.

Sonnenstand



Neuberechnung der Sonnendaten dauert einige Minuten

Abbildung 109:

Wie in Abbildung 109 gezeigt, können Sie über das Menü Optionen Ihren Längen- und Breitengrad eingeben, um so mit Hilfe der Funktion sun(), wie in Seite 190 angegeben, den Sonnenstand passend zu Ihrer geographischen Lage zu berechnen. Auch für die Funktionen azimuth und elavation (Seite 190) sind diese Koordinaten für die optimale Arbeitsweise anzugeben.

Nach dem Aufruf legt Enertex® EibStudio die Sektion [Location] im Anwenderprogramm an und schreibt die angegebenen Werte. Wenn diese Sektion nicht existiert, nimmt der Enertex® EibPC die Defaultwerte als Koordinaten 11° 3′ 29″ E, 49° 43′ 11″N; Dezimal: 11.07 und 49.68 an.

Beispiel:

```
[Location]
// Länge und Breite des Aufstellungsorts
11.06888888888889
49.0722222222222
```

Wurden die Daten verändert und das Anwenderprogramm neu gestartet, berechnet der Enertex® EibPC den Sonnenauf- und Untergang sowie den aktuellen Sonnenstand neu. Dieser Vorgang kann einige Minuten in Anspruch nehmen und wird Ihnen angezeigt, wenn Sie sich mit dem Enertex® EibPC verbinden.

Bitte beachten Sie, dass der Enertex® EibPC während dieser Zeit blockiert ist. Wenn Sie sich über das Menü EibPC → EIB-Telegramme abholen mit dem Enertex® EibPC verbinden, wird Ihnen in solchen Fällen eine Statusmeldung angezeigt.

Hinweis: Sie können die Sektion [Location] im Anwenderprogramm auch direkt modifizieren.

Sie finden die Daten für Ihren Standort z.B. im Internet unter www.geodatenzentrum.de.

Variablen und KNX™-Gruppenadressen

Manuelle oder importierte Gruppenadressen

Die Gruppenadressen, die in einem ets-Projekt angelegt wurden, können auf einfache Weise importiert werden. Das Vorgehen hierzu wird auf Seite 158 gezeigt.

Daneben besteht die Möglichkeit, auch direkt ohne diesen Import und damit ohne das ets-Projekt auf Gruppenadressen zu schreiben. Seite 167 zeigt die notwendige Vorgehensweise.

Innerhalb der Programmierung des Enertex® EibPCs im Abschnitt [EibPC] können Sie dann, mit Hilfe der Syntax von Seite 160, auf die Variablen und Gruppenadressen zugreifen.

ets Projektdaten

ets Export von Gruppenadressen

Wenn Sie Ihre ets-Daten mit dem Enertex® EibStudio nutzen wollen, gehen Sie wie folgt vor:

- Gruppenadressen aus der ETS exportieren als sog. ESF-Export.
- Gruppenadressen in das Enertex® EibStudio importieren.

Zum Export aus der ETS geht man wie folgt vor: Man öffnet in der ETS die Export-Funktion „Datenaustausch“ im Menü Datei (siehe Abbildung 110).

Abbildung 110: Export in der ets

Anschließend geht der Dialog nach Abbildung 111 auf. Hier wählt man den zweiten Knopf „Export zum OPC-Server“ aus. Im anschließenden Dateidialog gibt man dann eine Datei an, in der der Export erfolgt. Wir nennen diese (Text-) Datei im Folgenden ets-ESF-Export-Datei.

Abbildung 111: Export in der ETS zum „OPC-Server“

Die ets-ESF-Export-Datei stellt man am besten in das Projektverzeichnis.

Import von Gruppenadressen in das Enertex® EibStudio

Um den Export der Gruppenadressen der ESF-Daten des ets-Projekts zu nutzen, muss diese Datei in der Sektion [ETS-ESF] des Anwenderprogramms mit Pfad und Namen geschrieben werden. Das Enertex® EibStudio erledigt diesen Arbeitsschritt für Sie, wenn Sie wie nun beschrieben vorgehen:

Um die ESF-Datei zu laden klicken Sie:

BEARBEITEN → ADRESSEN AUS ETS-EXPORTDATEI (*.ESF) LADEN

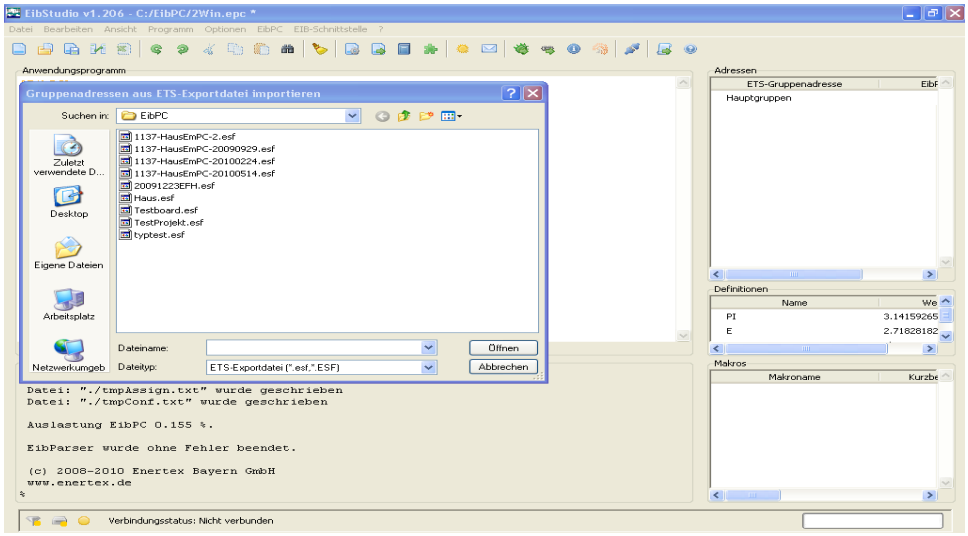


Abbildung 112: ETS-Exportdatei laden

Nun generiert Enertex® EibStudio automatisch die Sektion [ETS-ESF] im Anwenderprogramm wie in Abbildung 113 dargestellt.

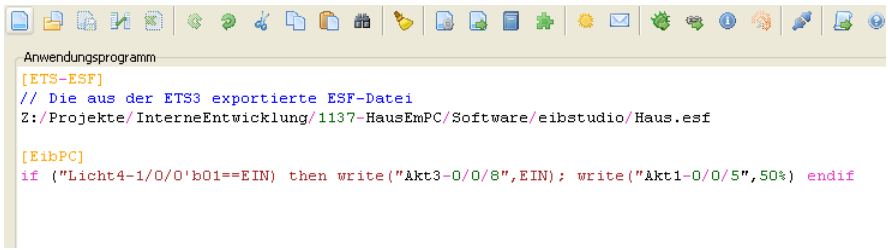


Abbildung 113: ETS-ESF-Sektion

In der ETS-ESF-Sektion wird nun der absolute Pfad zur exportierten ESF-Datei angezeigt. Im Adressbereich sind die aus dieser Datei importieren Adressen sichtbar und können per Copy & Paste genutzt werden. Vergleichen Sie hierzu Seite 152 und Abbildung 104.

Verwendung von Variablen und Gruppenadressen im Anwenderprogramm des Enertex® EibPC

Telegrammaufbau

Schematischer Aufbau eines Telegramms

Zieladresse			Bitlänge (der Nutzdaten)	Nutzdaten
Hauptgr.	Mittelgr.	Untergr.	Zahl	Werte
1	0	2	1...112	verschiedene Datentypen, z.B. 10°C, 10%, -2, „Alarm“

Abbildung 114: Schematischer Telegrammaufbau

Nützliches Hintergrundwissen

Mit “Telegramm“ wird ein kompletter Datenblock zur Übertragung einer Information bezeichnet. Das Telegramm enthält außer den zu transportierenden Daten selbst sämtliche zusätzlich benötigten Informationen in bestimmten Feldern, wie in Abbildung 114 angedeutet.

Datentypen

Bei den transportierten Daten muss sowohl die Bitlänge als auch die „Deutung“ der Daten bekannt sein, damit eine sinnvolle Programmierung überhaupt möglich ist.

So gibt es z.B. numerische Typen in unterschiedlicher Bitlänge: Der Wert „1“ kann sowohl mit 1 Bit, 4 Bit, 8 Bit usw. zu übertragen werden. Aufgrund dieser Mehrdeutigkeit muss bei der Programmierung der sogenannte „Datentyp“ genau spezifiziert werden.

Der Begriff „Datentyp“ fasst den numerischen Typ („Deutung“) und Bitlänge in einer Einheit zusammen.

Im Enertex® EibStudio wurde dazu folgende Syntax gewählt:

Der Datentyp wird jeweils an eine Zahl oder Adresse angehängt.
Der Datentyp besteht dabei meist aus einem Buchstaben und einer zweistelligen Zahl.

Mögliche Typen sind (angelehnt an übliche Programmiersprachen):

- vorzeichenlose (pos.) ganze Zahlen Buchstabe u („unsigned“)
- vorzeichenbehaftete ganze Zahlen Buchstabe s („signed“)
- Fließkommazahlen Buchstabe f („float“)
- Zeichenketten Buchstabe c („char“)
- Datum und Zeit Buchstabe t bzw. d bzw. y („time“, „day“, „year“)

Folgende Bitlängen sind möglich

- 1 Bit Ziffern 01
- 4 Bit Ziffern 04
- 8 Bit Ziffern 08
- 16 Bit Ziffern 16
- 24 Bit Ziffern 24
- 32 Bit Ziffern 32
- 64 Bit Ziffern 64
- 112 Bit (14 Zeichen) Ziffern 14
- 11200 Bit (1400 Zeichen) keine Ziffern
- 65534 Bit (65534 Zeichen) Ziffern 65534

Demnach ist u08 eine Datentyp der Länge 8 Bit und stellt eine vorzeichenlose (positive) ganze Zahl dar.

Zahlen (Konstanten)

Mit Hilfe des Datentyps können Zahlen und Konstanten im Enertex® EibStudio deklariert werden.

Bei Zahlen wird die Zahl dem Datentyp vorangestellt, also z.B.

- 2u08 Unsigned 8-Bit-Integer: 2
- 2.0f16 Fließkommazahl 2.0
- -6s32 Ganze Zahl mit Vorzeichen -6
- 33.2% Prozentwert 33.2 (entspricht 84)
- 35 8Bit

Ungültige Syntax erkennt der EibParser (integrierter Compiler im Enertex® EibStudio) und erzeugt eine Fehlermeldung.

Bei ganzen positiven Zahlen der Länge 8 Bit und Fließkommazahlen der Länge 16 Bit kann auf die Angabe des Datentypen verzichtet werden, d.h. Werte in der Form

- 0 ... 255 sind vom Datentyp u08.
- 2.0 (Dezimalpunkt in der Zahl) sind vom Datentyp f16.

Bei diese beiden Zahlentypen ist die Angabe von Datentypen optional.

Sondertyp: % (Prozentwert)

In der ETS Programmierung werden Prozentwerte „%“ verwendet. Diese sind kompatibel zum Datentyp „u08“ und werden von den KNX™-Aktoren durch Skalierung intern angepasst. Um dem Anwender hier die Programmierung zu vereinfachen, haben wir den Prozentwert für Konstanten definiert. Dabei kann der Prozentwert mit einer Kommastelle genau angegeben werden, also z.B. 2.3%. Aufgrund der Skalierung entspricht 100% einem Wert von 255u08 oder allgemein lautet die Umrechnung einer Größe Y% wie folgt:

$$X\,[\mathrm{u08}] = \frac{Y\,[\%]}{100} \cdot 255 \qquad \text{bei Abschneiden der Kommastellen}$$

Der integrierte Compiler im Enertex® EibStudio nimmt diese Anpassung für Sie vor, so dass Sie hier Aktoren wie gewohnt ansprechen können.

Wenn Sie unterschiedliche Datentypen in ihrem Anwenderprogramm miteinander verknüpfen, z.B. als die Summe von 2u08 und 2u32, so wird vom integrierten Compiler im Enertex® EibStudio ein Fehler gemeldet, so dass nicht unbeabsichtigte Überläufe, numerische Probleme etc. auftreten können. Um dennoch solche Zahlen in einander umzuwandeln und damit verarbeiten zu können, benutzen Sie die

convert-Funktion. Damit sind auch Umwandlungen von Zahlen in Zeichenketten möglich. Weiteres hierzu finden Sie auf Seite 217.

Hexadezimaldarstellung

Vorzeichenlose Ganzzahlen (Typ „u“) können auch in Hexadezimaldarstellung mit Präfix „0x“ angegeben werden. Der Compiler wandelt diese Darstellung in die entsprechende Zahl um.

- Datentyp u08: Es müssen 2 Stellen angegebenen werden: 0xF1 (= 241) oder 0xF1u08 (= 241)
- Datentyp u16: Es müssen mindestens zwei Stellen und der Datentyp „u16“ angegebenen werden: 0xF1A3u16 (= 61859u16)
- Datentyp u24: Es müssen mindestens zwei Stellen und der Datentyp „u24“ angegebenen 0x03A186u24 hexadezimale Schreibweise (=237958 bzw. Byte 1 0x03, Byte 2 0xA1 Byte 3 0x86)
- Datentyp u32: Es müssen mindestens zwei Stellen und der Datentyp „u32“ angegebenen werden: 0xF1A3u32 (= 61859u32)
- Datentyp u64: Es müssen mindestens zwei Stellen und der Datentyp „u64“ angegebenen werden: 0xF1A3u64 (= 61859u64)

Sondertyp: Zeichenkette

Zeichenketten werden in der Form

- $Zeichenkette$c14 angeben. Dabei steht Zeichenkette für einen beliebigen Text, der aber nicht mehr als aus 14 Zeichen bestehen darf. Dieser Typ ist kompatibel zur KNX Zeichenkette (z.B: Anzeigeelemente).
- $Zeichenkette$ (ohne Zusatz c14) ist der zweite eingebaute Datentyp. Hier steht Zeichenkette für einen beliebigen Text, der aber nun aus 1400 Zeichen bestehen darf.

Also man unterscheide:

$ Hallo $c14: Zeichenkette aus maximal 14 Zeichen

$ Hallo $: Zeichenkette aus maximal 1400 Zeichen

Die beiden Zeichenkette können mit Hilfe der convert-Funktion ineinander umgewandelt werden (siehe Seite 217).

Sondertyp: IP Adresse

IP Adressen (Zusatzpaket Option NP) können Sie in der üblichen Syntax eingeben

- 192.168.22.100 kennzeichnet die gleichlautende Adresse. Intern ist diese Adresse identisch zum Datentyp u32.

Sondertyp: Pyhsikalische Adresse

Physikalische Adressen können Sie in der folgenden Syntax eingeben

- 1.12.230 kennzeichnet die gleichlautende physikalische Adresse eines KNX™ Busteilnehmers. Diese Adresse identisch zum Datentyp u16.

Datentypen in der Übersicht

Typ	Datentyp	Beispiel für Konstante	Verwendung	Zahlen-Bereich	EIS Datentyp
Binär	b01	1b01	Schaltaktor, Jalousieaktor	0, 1	EIS1/EIS7
2 Bit	b02	2b02	Sperrobjekte	0,1 ... 3	EIS8
4 Bit	b04	10b04	Dimmen	0,1 ... 15	EIS2
Prozentwert	%	85.3%	Heizregler, Stellglieder	0,0.1 ... 100.0	EIS6/EIS14.001
8 Bit Integer ohne Vorzeichen	u08	255	Einfache Zahlen, Raumtemperaturregler etc.	0, ... 255	EIS6/EIS14.001
8 Bit Integer ohne Vorzeichen	u08	255u8	Optional mit Datentypen		EIS6/EIS14.001
8 Bit Integer mit Vorzeichen	s08	-45s08	Temperatursensor	-128... 127	EIS14.000
16 Bit Integer ohne Vorzeichen	u16	45u16		0 ... 65535	EIS10.000
16 Bit Integer mit Vorzeichen	s16	-450s16		-32768 ... 32767	EIS10.001
24 Bit Integer ohne Vorzeichen	u24	92235u24		0 .. 16777215	n.V.
32 Bit Integer ohne Vorzeichen	u32	92235u32		0 .. 4294967295	EIS11.000
32 Bit Integer mit Vorzeichen	s32	-9999s32		-2147483648 .. 2147483647	EIS11.001
64 Bit Integer ohne Vorzeichen	u64	92235u64		0 .. 18446744073709551615	n.V.
64 Bit Integer mit Vorzeichen	s64	-9999s64		-9223372036854775808 .. 9223372036854775807	n.V.
Short Float	f16	4.0	Windsensor	-671088.64 .. 670760.96	EIS5
Short Float	f16	4.0f16		-671088.64 .. 670760.96	EIS5
Float 32 Bit	f32	4.0e01f32		-3.40282e+38 .. 3.40282e+38	EIS9
Zeichenkette	c14	$HalloWelt$c14	Anzeigepanels	14 Zeichen	EIS15
Zeichenkette	(c1400)	$HalloWelt$	LAN Telegramme	1400 Zeichen	n.v.
Zeichenketten können von1 bis 65534 definiert werden	c65534	$HalloWelt$c65534		65534 Zeichen	n.v.
IP Adresse	(u32)	192.168.22.100	Feste IP Adressen bei sendudp etc.		EIS11.000

Tabelle 1: Datentypen

Hinweis:

Die Datentypen d24, t24, y64 sind KNX DTP Typen, die über ihre Definition im Enertex® EibPC korrekt verarbeitet werden. Eine Eingabe als Konstante ist nicht notwendig und daher nicht möglich. Diese Datentypen werden nur in Verbindung mit den Funktionen getdate und gettime benötigt. Weiteres hierzu finden siehe ab Seite 180.

Variablen

Variablen beginnen mit Buchstaben, gefolgt von einer beliebigen Anzahl und Kombination von Buchstaben oder Zahlen und dem „_“ Zeichen. Variablen werden mit einem Wert oder einer Funktion initialisiert. Groß- und Kleinschreibung wird im Gegensatz zu Schlüsselwörtern und Funktionsnamen beachtet.

Daher sind z.B. adresse und Adresse unterschiedliche Variablen.

Der Compiler „EibParser“ überprüft bei Zuweisung einer Variable und deren Weiterverarbeitung immer die Datentypen und unterbindet durch eine Fehlermeldung beim Generieren des Anwendungsprogamms unzulässige Verknüpfungen von nicht kompatiblen Datentypen, so dass nicht unbeabsichtigte Überläufe, numerische Probleme etc. auftreten können.

Wollen Sie Variablen mit unterschiedlichen Datentypen verknüpfen, benutzen Sie die convert-Funktion (siehe Seite 217).

Jeder Variable muss ein einziges Mal initialisiert werden. Die Deklaration von Variablen muss demnach eindeutig sein.

Ein paar Beispiele

```
a=123
A1=1b01
adresse=A1 or 0b01
Adresse=4%+5%+23u08
Wert=4e4*0.2
w=4e16f32
```

Variablen dürfen nicht von sich selbst abhängig definiert werden („Rekursion“). Daher ist folgender Ausdruck als Definition ungültig:

Nicht zulässig, aber...

```
a=a+1
```

Hingegen ist es zulässig, mit Hilfe von Variablen auf diese Weise einen Zähler zu programmieren:

... hier schon

```
//Deklaration
a=0
//Zählen
if (sun()) then a=a+1 endif
```

Keine Sonderzeichen in Variablennamen

Umlaute sind bei Variablennamen nicht erlaubt. Daher ist folgender Ausdruck **ungültig**

```
KücheLichtEin=1b01
```

Konvertierung

Es dürfen nur Variablen mit gültigem Datentyp verknüpft werden. Funktionen erwarten zum Teil unterschiedliche oder gleiche Datentypen als Argumente. Um beliebige Datentypen miteinander zu verknüpfen, muss die convert-Funktion (siehe Seite 217) genutzt werden.

Beispiel: Ein 8Bit unsigned Wert soll mit einem 16Bit signed Wert addiert werden:

```
Var1=10u08
Var2=300s16
// Var3=310
Var3=convert(Var1,Var2)+Var2
```

Implizite Konvertierung

Sowohl Var2, als auch die convert(Var1,Var2) sind mit Datentyp s16 definiert. Dies erkennt der integrierte Compiler des Enertex® EibStudio und legt damit den Datentyp Var3 selbstständig fest. Wir nennen dies eine **implizite Konvertierung**, da der Anwender den Datentyp von Var3 nicht explizit (ausdrücklich) vorgibt. In jedem Fall überprüft der Compiler die Datentypen von Variablen und deren Verknüpfungen.

Vordefinierte Variablen

Im Enertex® EibStudio sind Variablen vordefiniert, um das Erstellen eines Anwenderprogramms möglichst einfach zu gestalten. Diese Variablen sind im Enertex® EibStudio im Fenster Definitionen aufgeführt. Vordefinierte Variablen können nicht überschrieben werden. Eine Übersicht der vorhandenen Variablen ist in Seite 348 aufgelistet.

Importierte KNX™ -Gruppenadressen

Angenommen, Sie haben die in Tabelle 1 dargestellte Gruppenadresse im ETS Projekt definiert und anschließend wie oben beschrieben aus der ETS exportiert. Schließlich haben Sie die Gruppenadresse, wie in Kapitel 152 beschrieben, in das Enertex® EibStudio importiert und möchten diese Gruppenadresse in Ihrem Anwenderprogramm nutzen.

	Name	Adresse
Hauptgruppe	Beleuchtung Steckdosen	1
Mittelgruppe	Keller	0
Untergruppe	KellerWC	0

Tabelle 2: ETS Gruppenadresse

Hierzu müssen Sie wissen, wie eine derartige Variable importiert bzw. im Anwenderprogramm ansprechbar wird. Die Gruppenadresse wird nach dem Bildungsgesetz

Bildungsgesetz

Importierte Adresse: "Untergruppenname-Gruppenadresse"

automatisch intern durch die Angabe der Exportdatei in der Sektion [ETS-ESF] geladen. Wenn also die EIB-Objekte von der ETS exportiert wurden, kann man auf diese immer zugreifen.

Beispiel: ETS Projekt und Gruppenadressen

Angenommen, Sie haben Gruppenadressen in der ETS definiert, wie in der Abbildung 115 als Screenshot auszugsweise dargestellt.

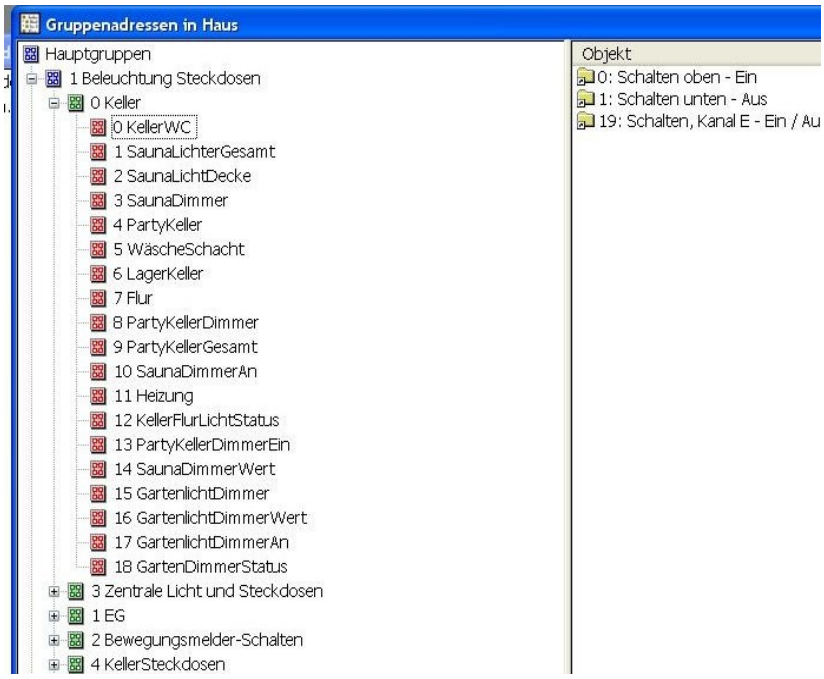


Abbildung 115: Gruppenadressen eines ETS Projekts

Mit Hilfe der Importfunktion werden dann die Adressen

- "KellerWC-1/0/0"
- "SaunaLichterGesamt-1/0/1"
- "SaunaDimmer-1/0/3"

Copy&Paste von imporierten Adressen

usw. im Enertex® EibStudio im Adressen-Fenster zur Verfügung gestellt (vgl. Abbildung 115). Wenn Sie auf diese Adressen mit der linken Maustaste drücken, können Sie die Adresse per Copy&Paste verwenden.

Import der ESF-Datei

Wenn Sie diese Gruppenadressen und deren Definition im Enertex® EibStudio nutzen wollen, so müssen Sie die Gruppenadressen zunächst wie auf Seite 158 erläutert exportieren und anschließend die Export-Datei in die Sektion [ETS-ESF] mit Pfad und Namen eintragen (siehe Beispiel dazu auf Seite 158).

Innerhalb der Programmierung des Enertex® EibPCs im Sektion [EibPC] können Sie dann mit Hilfe von folgender Syntax darauf zugreifen:

```
[EibPC]
a="KellerWC-1/0/0"
```

Sie können dazu einfach die Adressen im Fensterbereich des Enertex® EibStudio kopieren und anschließend im Textfenster Anwenderprogramm einfügen.

Mängel bei ETS

Die ETS exportiert leider nicht für alle Datentypen den passenden nummerischen Typ (wobei die Bitlänge immer richtig angegeben wird).

Enertex® EibStudio erkennt beim Kompilieren, welche Adressen seitens der ETS nicht vollständig definiert und daher unvollständig sind. Aufgrund der Datentypenvorgaben kann jedoch der Compiler auf den Datentyp zurück schließen. Daher erkennt der Compiler im folgenden Beispiel, dass die Variable *a* vom Datentyp u08 (Prozentwert) sein muss.

```
[EibPC]
b=5%
a="Dimmer-5/1/0"+b
```

Erklärung:

Implizite Konvertierung

Eine Summe kann nur mit gleichen Datentypen gebildet werden. Da b eindeutig vom Typ Prozentwert ist, müssen beide Teile der Summe vom Typ Prozentwert sein. Daher ist "Dimmer-5/1/0" eindeutig als Prozentwert zu deuten und damit liegt der Datentyp für die Gruppenadresse indirekt (implizit) vor. Der Anwender muss daher nicht weiter den Datentyp definieren. Sollte allerdings der ets-Export hier einen Datentyp s08 vorgegeben haben, wirft der Compiler natürlich eine Fehlermeldung aus. Gleiches gilt, wenn die Bitlänge nicht übereinstimmt - diese wird ja in jedem Fall von der ets-Exportfunktion korrekt dargestellt.

Nicht benötigte Gruppenadressen

Möchten Sie Ihrem Kunden nur bestimmte Adressen zur Verfügung stellen, bearbeiten Sie einfach dazu die ESF-Datei direkt mit einem Texteditor und löschen Sie Zeilen mit den nicht benötigten Gruppenadressen. Sie können hier auch die Namen der Gruppenadressen direkt bearbeiten, wenn Sie dies für notwendig erachten oder nicht Zugriff auf die Orginal-ETS Datensätze haben. Beachten Sie aber, dass Sie bei diesem Vorgehen nicht unbeabsichtigt Fehler generieren.

„Manuelle“ Gruppenadressen

Neben der Möglichkeit Gruppenadressen mit Hilfe der ETS Projektdaten zu nutzen, können Sie Gruppenadressen beliebig selbst definieren, ohne dabei auf die ETS zurückgreifen zu müssen. Dazu müssen Sie lediglich die nun folgende Schreibweise nutzen:

Manuelle Adresse: 'Gruppenadresse'Datentyp

Gruppenadressen ohne die Verwendung der ETS beginnen mit einem einfachen Anführungszeichen, gefolgt von der Hauptgruppe/Mittelgruppe/Untergruppe (im numerischen Format), gefolgt von einem einfachen Anführungszeichen und dem Datentyp, wie er in Tabelle 1 dargestellt wurde.

Beispiele

'1/0/0'u08
'1/0/1'b01
'5/0/81's16

Im obigem Beispiel ist die erste Gruppenadresse 1/0/0 vom Typ einer vorzeichenlosen ganzen Zahl mit 8 Bit Länge, die Adresse 1/0/1 ist ein binärer Typ und 5/0/81 ist vom Typ eines vorzeichenbehafteten ganzen Zahl 16 Bit Länge. Die gleichzeitige Verwendung von importierten und manuellen Adressen ist jederzeit möglich.

Die if-Anweisung

Wie Sie schon der Schritt-für-Schritt Einführung (Seite 35) entnehmen konnten, ist die if-Anweisung, also die bedingte Anweisung zentraler Punkt einer Automatisierung. Die Syntax einer if-Anweisung lautet wie folgt:

```
if (Abfragebedingung)  then Anweisung endif
if (Abfragebedingung)  then Anweisung1 else Anweisung2 endif
```

Die Abfragebedingung muss vom Datentyp b01 sein. Wenn die Abfragebedingung nicht von diesem Datentyp ist, gibt der integrierte Compiler des Enertex® EibStudio einen Fehler aus.

Es können auch mehrere Anweisungen als Block ausgeführt werden. In diesem Fall sind die Anweisungen durch Strichpunkt zu trennen. **Nach der letzten Anweisung steht kein Strichpunkt**.

Eine Abfragebedingung wird dann als erfüllt angesehen, wenn sie den Wert EIN (1b01) annimmt. In diesem Fall wird der then-Zweig ausgeführt. Hat die if-Abfrage einen else-Zweig, so wird dieser bei Nicht-Erfüllung, d.h. der Wert der Abfragebedingung AUS (0b01) angesehen.

Beispiel

```
if (a==23u08) then write('1/2/3's16,-234s16) else write('1/2/3's16,0s16) endif
```

Für die Abfragebedingung der if-Anweisung gilt das Valdierungschema, wie es auf der folgenden Seite 168 für Variablen beschrieben ist. Dies bedeutet, dass eine if-Anweisung nur (einmal) ausgewertet wird, wenn die Abfragebedingung sich ändert. Wenn a den Wert 23u08 annimmt, so wird einmal der then-Zweig ausgeführt. Im darauf folgende Zyklus, wenn a sich nicht ändert, wird auch die if-Funktion nicht erneut ausgeführt. Wenn sich a ändert, aber die Abfragebedingung nicht erfüllt ist, wird bei jeder dieser Änderungen einmal der else-Zweig ausgeführt. Weiteres hierzu finden Sie auf Seite 168 oder in der Schritt-für-Schritt Einführung.

Lesen und Schreiben auf dem KNX™-Bus

Zuweisen von Variablen und Gruppenadressen - Lesen vom KNX™-Bus

Sie können mittels einfacher Zuweisung die Information, die ein Sensor an eine Gruppenadresse sendet, in einer Variablen oder Funktion nutzen. D.h. mit der Anweisung *Variable = Gruppenadresse* wird die Information, die an die *Gruppenadresse* gesendet wird, der *Variablen* im Enertex® EibPC zugewiesen. Dabei kann die *Gruppenadresse* wie eben beschrieben importiert oder „manuell“ im Enertex® EibPC definiert werden.

Die Verwendung von Variablen und Gruppenadressen (importierte oder manuelle) ist daher kompatibel, d.h. Sie können an Stelle von Variablen auch Gruppenadressen als Argumente von Funktionen verwenden.

Beispiel “Manuelle“ KNX™ Gruppenadressen zuweisen

```
a='1/0/0'u08
b='1/0/0'u08 + '1/0/1'u08
c= a+ '1/0/1'u08
```

Im obigem Beispiel wird der Inhalt der Gruppenadresse 1/0/0 (genauer gesagt. Den Inhalt der Nutzdaten des Telegramms mit dieser Gruppenadresse), welche vom Typ eines vorzeichenlosen ganzen Zahl ist, der Variablen a zugewiesen. Die Variable b ist die Summe aus den beiden Werten, die über die Gruppenadresse 1/0/0 und Gruppenadresse 1/0/1 gesendet werden. Die Variable c ist inhaltlich identisch zur Variablen b, man kann also Gruppenadressen wie Variablen nutzen.

Der Enertex® EibPC speichert immer den aktuellen Zustand des Inhalts der Gruppenadresse. Wenn der Enertex® EibPC neu gestartet wird, muss dies berücksichtigt werden. In diesem Fall kennt der Enertex® EibPC bereits gesendete Telegramme nicht und initialisiert den Zustand mit 0 (0.0 etc. je nach Datentypen).

Vorbelegung und systemstart()

Mit der Funktion systemstart(), siehe Seite 221 können Variablen, die von Gruppenadressen abhängig definiert werden, beim Systemstart mit bestimmten Werten definiert vorbelegt werden.

Eine Variable, die in irgendeiner Weise von einem Bustelegramm abhängig ist, wird nur ausgewertet, wenn sich der Wert, der über das Bustelegramm eintrifft, ändert.

Beispiel

```
c= 1u08+ '1/0/1'u08
if systemstart() then c=2u08 endif
if (c==2u08) then write('3/3/3'b01,EIN) endif
if change(c) then write('3/3/2'b01,EIN) endif
```

Der Enertex® EibPC initialisiert die Auswertung des Telegramms mit der Gruppenadresse 1/0/1 zu 0, demnach ist c mit 1u08 vorbelegt. Durch die Anweisung if systemstart() wird c auf 2u08 gesetzt. Wenn über den Bus ein neues Telegramm mit der Gruppenadresse 1/0/1 eintrifft, so wird c neu ausgewertet.

Dann überprüft der Enertex® EibPC, ob aufgrund der Änderung der Variablen c, weitere Variablen und Anweisungen, die von c abhängig sind, auch neu ausgewertet werden müssen. Im Beispiel ist das der Fall. Wenn auf der Gruppenadresse 1/0/1 der Wert 1u08 gesendet wird, liefert der Vergleich in der if-Anweisung 1b01 und somit schreibt der Enertex® EibPC ein Telegramm auf Gruppenadresse 3/3/3 mit dem Wert 1b01 auf den Bus. Die change-Funktion (Seite 217) überprüft, ob sich eine Variable im letzten Verarbeitungszyklus geändert hat. In diesem Fall ist ihr Rückgabewert EIN (1b01). Daher wird die letzte if-Anweisung immer ausgeführt, wenn sich die Variable ändert und somit ein Telegramm auf Gruppenadresse 3/3/2 mit Wert EIN (1b01) geschrieben.

Das Validierungschema

Der Enertex® EibPC verarbeitet nur sich ändernde Ausdrücke und merkt sich dabei den letzten Zustand, der über eine Gruppenadresse gesendet wurde. Das Abbilden der Zustände der eingegangen Telegramme und Variablen bezeichnen wir als Validierungsschema. Es macht die Anwendung und Programmierung einfach und intuitiv, solange man Verschachtelungen von if-Anweisungen vermeidet. Wie im Beispiel gezeigt, ist diese Vermeidung immer auf einfache Weise möglich.

Wir empfehlen Anfängern die Vermeidung von verschachtelten if-Anweisungen.

Vergleichen Sie hierzu auch das Beispiel auf Seite 170 zur read-Funktion.

Variablendefinition

Beispiel: Importierte Gruppenadressen zuweisen

```
a="KellerWC-1/0/0"
b="SaunaLicht-1/0/1"
c=max("HeizungKeller1-1/0/2","HeizungKeller2-1/0/3","HeizungKeller3-1/0/4")
d=min(c,"HeizungKeller3-1/0/4")
```

a und *b* sind Zuweisungen von importierten Gruppenadressen, die vom Typ des in der ETS definierten Datentyps sind.

Um unterschiedliche nummerische Datentypen zu verarbeiten, benötigt man die convert Funktion (s.-Seite 217). Die Variable c nutzt im Beispiel die max Funktion, welche das Maximum einer beliebigen Anzahl von Variablen oder Gruppenadressen ist.

Dabei müssen die Datentypen der Gruppenadressen (Argumente der max-Funktion) gleich sein. Dabei wird die Variable c vom Typ der Argumente der max-Funktion sein und daher durch die Argumente der max()-Funktion im Datentyp festgelegt (siehe hierzu auch Seite 164).

Die Variable d ist das Minimum der Variablen c und der importierten Gruppenadresse "HeizungKeller3-1/0/4".

Schreiben auf den KNX™-Bus: write()

Das Schreiben einer Information auf den KNX™ Bus wird mit der Hilfe der write-Funktion realisiert.

Definition

- write(*Gruppenadresse, Wert*)

Argumente

- Zwei Argumente vom gleichen Datentyp. Datentypen aber sonst beliebig.
- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse
- *Wert*: Der Wert, welcher auf die KNX™ Gruppenadresse (auf den KNX™-Bus) geschrieben werden soll.

Wirkung

- Auf dem Bus wird ein gültiges KNX™ Telegramm generiert, welches den *Wert* auf die *Gruppenadresse* schreibt.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel

```
write("KellerWC-1/0/0",1b01)
write('1/0/1'u08,10%) endif
```

Man beachte: Die Typen „u08“ und „%“ sind gleichwertig und kompatibel (siehe auch Seite 160).

Leseanforderung einer Gruppenadresse: read()

Die Leseanforderung des Wertes eines Aktors mit Hilfe der entsprechenden Gruppenadresse vom KNX™-Bus wird mit der Hilfe der read-Funktion realisiert.

Hinweis:

Damit der Aktor im KNX™ Netzwerk auch antwortet, muss das Lese-Flag bei dessen ETS Programmierung auch gesetzt werden.

Definition

- read(*Gruppenadresse*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse

Wirkung

- Auf dem Bus wird ein gültiges KNX™ Telegramm generiert, das über das „Lese-Flag“ alle auf die *Gruppenadresse* parametrierte Aktoren abfragt.
- *Gruppenadresse* kann optional mit dem !-Zeichen negiert werden (ohne dass diese ein Auswirkung auf die Wirkung der Funktion hat).

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: Aktuellen Temperaturwert vom Bus „erfragen“

Auf der Adresse 2/3/4 kann von einem Temperatursensor eine Temperatur auf den Bus im Fließkommaformat f16 (16 Bit) gesendet werden. Das Bit “Leseanforderung“ ist in der ETS gesetzt, d.h. die Temperatur kann per Leseanforderung abgefragt werden.
Jeden Tag, 18:30 Uhr, 20 Sekunden, soll die Variable Temperatur von Sensor erfragt werden.

Umsetzung

```
Temperatur='2/3/4'f16
if chtime(18,30,20) then read('2/3/4'f16) endif
```

Mit der Anweisung *Variable = Gruppenadresse* wird die Information, die, angestoßen durch die read-Funktion, an die *Gruppenadresse* gesendet wird, einer Variablen zugewiesen.
Insgesamt kann der Ablauf des Beispiels mit Abbildung 116 dargestellt werden.

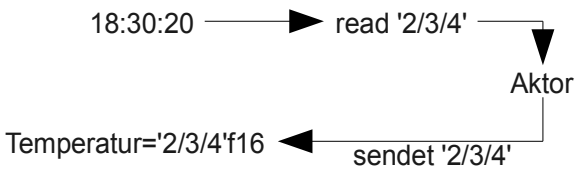


Abbildung 116: Funktionsweise von read

Sobald die Uhrzeit 18:30:20 erreicht wurde, geht die chtime auf EIN, die Bedingung der if-Anweisung ist erfüllt und die read sendet die Leseanforderung. Der Aktor antwortet nun und schickt den Wert auf die Gruppenadresse '2/3/4'f16. Der Enertex® EibPC schreibt das Ergebnis in die Variable Temperatur.

Initialisieren von Gruppenadressen

Bevor der Enertex ® EibPC mit der Verarbeitung des Anwenderprogramms beginnt, kann der Benutzer den Speicher der Gruppeadressen initialisieren. Die Enertex ® EibPC speichert immer den aktuellen Stand des Inhaltes der Gruppenadresse als ein Abbild im Speicher. (siehe auch gaimage() auf Seite 240). Wenn der Enertex® EibPC ein neues Programm hochgeladen hat, so sind diese Abbilder im Speicher auf 0 gesetzt. Aber da der KNX Bus bereits vor dem EibPC läuft, werden die Abbilder im Speicher nicht mit den Zuständen der Gruppenadressen übereinstimmen (außer sie sind tatsächlich Null, was höchst unwahrscheinlich der Fall ist).

Die Sektion [InitGA]

Um sich mit dem KNX™ Bus vor der eigentlichen Verarbeitung zu synchronisieren, kann der Anwender Gruppenadressen mit einem Lesetelegramm vom Enertex® EibPC abfragen. Dazu definiert man die Sektion [InitGA] wie folgt:

Beispiel

```
[InitGA]
// Temperatur als Manuelle Gruppenadresse
'2/3/4'f16
“Heating-2/3/4”
“Lights-2/3/2”
[EibPC]
if “Lights-2/3/2” and '2/3/4'f16<10.0 then write(“Heating-2/3/4”,100%) endif
```

Wichtige Erklärungen

- Sie können das Kontextmenü (rechte Maustaste) über den Gruppenadressen nutzen, um Enertex® EibStudio den passenden Eintrag in die Sektion machen zu lassen.
- Sie können jede beliebige Gruppenaddrese einfügen. Wenn Sie mit manuellen Gruppenadressen arbeiten (siehe oben '2/3/4'f16), muss die Gruppenadresse tatsächlich auch einmal im Anwenderprogramm verwendet werden.
- Initialisierung einer Gruppenadresse bedeutet, dass der Enertex® EibPC vor dem eigentlichen Beginn der Verarbeitung Leseanforderungen auf den Bus ausgibt. Nach jeder Leseanforderung auf eine Gruppenadresse wird auf die Antwort gewartet, jedoch nicht länger als 1500 ms.
- Nachdem für die letzte Gruppenadresse auf diese Weise eine Leseanforderung auf den Bus gesandt wurde (und auf das Eintreffen der Antwort maximal 1500 ms gewartet wurde), wird mit der Verarbeitung begonnen.
- Sie können alternativ auch die Funktion initga nutzen.
- Alle Anweisungen die als Bestandteil einer der in der Sektion [InitGA] aufgeführten Gruppenadressen beinhalten, werden ungültig und die Verarbeitung im ersten Durchlauf ausgeführt.
- Wenn eine Gruppenadresse nicht auf initGA antwortet, wird ein Event generiert.

Beispiel 2

```
[InitGA]
“Lights-2/3/2”
[EibPC]
if “Lights-2/3/2” write(“Heating-2/3/4”,100%) endif
if !“Lights-2/3/2” write(“Heating-2/3/4”,0%) endif
```

Im Normalfall wird jede Anweisung beim Systemstart auf gültig gesetzt, also wird weder

```
if “Lights-2/3/2” write(“Heating-2/3/4”,100%) endif
```

noch

```
if !“Lights-2/3/2” write(“Heating-2/3/4”,0%) endif
```

ausgeführt.

Durch die Verwendung von [InitGA] werden alle in dieser Sektion aufgeführten Gruppenadressen auf ungültig gesetzt und die Verarbeitung im ersten Durchlauf ausgeführt.

Im obigen Beispiel wird auf die GA „Lights-2/3/2“ eine Leseanforderung geschrieben. Abhängig von der Antwort wird entweder die 1. oder 2. If-Anweisung ausgeführt.

Initga

Definition

- initga(*Gruppenadresse*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse
- Die Gruppenadresse kann optional invertiert werden (Verwendung des !-Vorzeichens).

Wirkung

- Die Funktion bewirkt exakt das selbe Verhalten, als wäre *Gruppenadresse* in einer Sektion [InitGA] definiert.
- Die Funktion kann nur „toplevel“ genutzt werden. Sie kann nicht im then- oder else Zweig einer if-Abfrage stehen.
- Die Funktion kann auch in Verbindung mit der Funktion comobject (S. 222) genutzt werden

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel

Folgendes Beispiel zeigt die Verwendung alternativ zu obiger [InitGA] Sektion

```
[EibPC]
// Temperatur als Manuelle Gruppenadresse
initGA('2/3/4'f16)
initGA(“Heating-2/3/4”)
initGA(“Lights-2/3/2”)
if “Lights-2/3/2” and '2/3/4'f16<10.0 then write(“Heating-2/3/4”,100%) endif
```

Beispiel 2 - comobject

Folgendes Beispiel zeigt die Verwendung in Kombination mit der Funktion comobject.

```
[EibPC]
initga(!"Licht KG Treppe-0/0/2")
initga(comobject("Licht EG -Decke Flur-0/0/14","Licht EG Speis-0/0/18"))
```

Sowohl die Verwendung von Negationen als auch die Funktion comobject sind in Verbindung mit der Funktion initga möglich. Dies bringt ins Besondere bei der Programmierung von Makros erhebliche Vorteile mit sich.

Bus-Aktivität

Event

Diese Funktion reagiert immer, wenn ein Telegramm für die überwachte Adresse auf den Bus geschrieben wird. Sie reagiert nicht auf Variablen.

Im Zusammenhang mit UDP, TPC/IP oder RS232 Telegrammen reagiert sie identisch auf das Eintreffen von UDP Nachrichten (siehe Seite 245 bzw. Seite 67 in der Schritt-für-Schritt Einführung).

Eine Event-Funktion ist wie folgt definiert:

Definition

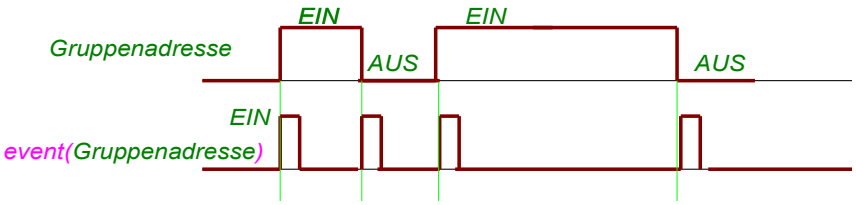
- Funktion event(*Gruppenadresse*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse
- Bei UDP, TPC/IP oder RS232 Telegrammen (siehe Seite 143) kann die event-Funktion auf die Funktion readudp angewendet werden.
- *Gruppenadresse* kann optional mit dem !-Zeichen negiert werden (ohne dass diese eine Auswirkung auf die Funktion hat).

Wirkung

- Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus ein Telegramm mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.



Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Eine Besonderheit der event-Funktionen ist, dass diese nicht bei if-Anweisungen mit else-Zweig stehen dürfen. Die event-Funktion geht immer nur für einen Verarbeitungszyklus auf EIN gehen und wird bei Eintreffen eines Telegramms auf die Gruppenadresse bei der if-Anweisung den then-Zweig ausführen. Im nächsten Zyklus geht event wieder auf AUS und nun wird der else-Zweig ausgeführt. Um den Anwender die Programmierung zu vereinfachen, wurde die Verwendung der event-Funktion an dieser Stelle durch den Compiler beschränkt (vgl. Seite 46).

Ein Beispiel für die Verwendung der event-Funktion.

Immer wenn die Adresse “BewegungsMelder-3/2/3“ oder !“BewegungsMelder-3/2/4“ ein Ereignis bekommt, soll die Variable Licht auf EIN gesetzt werden.

Nach 3 Minuten soll die Variable Licht wieder auf AUS gesetzt werden.

Die Umsetzung lautet dann:

```
if (event(“BewegungsMelder-3/2/3“)) or (event(“BewegungsMelder-3/2/4“)) then Licht=EIN endif
if(after(Licht,180000u64)==EIN) then Licht=AUS endif
```

Das Überwachen der Busaktivität auf eine Gruppenadresse wird mit der Hilfe der event-Funktion realisiert.

Mit den nun folgenden Event-Funktionen kann analysiert werden, ob das am Bus geschriebene Telegramm

1. ein „normales“ Schreibtelegramm,
2. eine Leseanforderung oder
3. eine Antwort auf eine Leseanforderung

darstellt.

Eventread

Definition

- Funktion eventread(*Gruppenadresse*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse
- *Gruppenadresse* kann optional mit dem !-Zeichen negiert werden (ohne, dass diese eine Auswirkung auf die Funktion hat).

Wirkung

- Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus eine Leseanforderung mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Eventresponse

Definition

- Funktion eventresponse(*Gruppenadresse*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse
- *Gruppenadresse* kann optional mit dem !-Zeichen negiert werden (ohne dass diese eine Auswirkung auf die Funktion hat).

Wirkung

- Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus eine Antwort auf eine Leseanforderung mit der Gruppenadresse geschickt wird. Der Inhalt spielt dabei keine Rolle.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Eventwrite

Definition

- Funktion eventwrite(*Gruppenadresse*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse
- *Gruppenadresse* kann optional mit dem !-Zeichen negiert werden (ohne dass diese eine Auswirkung auf die Funktion hat).

Wirkung

- Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus auf die Gruppenadresse geschrieben wird, unabhängig von deren Inhalt.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Writeresponse

Definition

- Funktion writeresponse(*Gruppenadresse,Wert*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX™ Gruppenadresse
- *Wert*: Der Wert, der auf die KNX™ Gruppenadresse (auf den KNX™-Bus) geschrieben werden soll

Wirkung

- Antwortet auf eine Leseanforderung, indem ein gültiges KNX™ Telegramm generiert wird, das den *Wert* auf die *Gruppenadresse* schreibt. Das Antwortflag wird im Telegramm gesetzt.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel writeresponse

```
Auf eine Leseanforderung soll geantwortet werden.

if eventread('GA'b01) then writeresponse('GA'b01,a) endif
```

Szenenaktoren

Szenenaktoren sind ebenso spezielle KNX™-Funktionen. Weiteres finden Sie in der Schritt-für-Schritt Anleitung auf Seite 65 und auf Seite 225.

Befehle und Funktionen

Hinweis:

Für alle Argumente von Funktionen können anstelle von Variablen auch direkt Gruppenadressen verwendet werden.

Logische Verknüpfungen

Und - Verknüpfung

Zum Erstellen von Und-Verknüpfungen ist die and Anweisung vorgesehen. Diese Anweisung ist wie folgt aufgebaut:

Definition

- *A* and *B* [and *C* ... usw.]

Argumente

- Alle Argumente (*A, B, C* ...) vom gleichen Datentyp. Datentypen aber sonst beliebig.
- Beliebig viele Verknüpfungen

Wirkung

- Die Variable *A* wird bitweise mit der Variablen *B* (und der Variablen *C* usw.) „verundet“: Das Ergebnis der Operation and ist null (alle Bits), wenn eine der Variablen null (alle Bits) ist. Im anderen Fall ist das Ergebnis eine bitweise Verundung, d.h. das n-te Bit des Ergebnisses ist null, alsbald eines der Bits der Eingangsvariablen Null ist. Ansonsten ist das n-te Bit des Ergebnisses 1, d.h. die jeweils n-ten Bits der zwei (oder mehr) Eingangsvariablen sind 1.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Und-Verknüpfung

LichtAktorEin ist das Ergebnis der Und-Verknüpfung von Variable TasterEin und Variable LichtFreigabe

Die Umsetzung im Anwenderprogramm lautet dann:

```
LichtAktorEin = TasterEin and LichtFreigabe
```

Wenn *TasterEin* 1b01 ist und *LichtFreigabe* 1b01 ist, dann ist *LichtAktorEin* auf 1b01, sonst 0b01.

Beispiel: Und-Verknüpfung bei unterschiedlichen Variablen

Wenn die Variable TasterEin auf '1' und die Variable Windgeschwindigkeit genau 2.9 m/s ist, soll die Variable LichtAktorEin auf '1' gesetzt werden.

Für die Umsetzung benötigen wir die „if“ - Anweisung und den Vergleich „==“. (Dabei ist die gesamte if-Abfrage in Klammern zu setzen). Die Umsetzung lautet dann:

```
if ((TasterEin==1u08) and (Windgeschwindigkeit==2.9f16)) then LichtAktorEin=1u08 endif
```

Oder - Verknüpfungen

Zum Erstellen von Oder-Verknüpfungen ist die or-Anweisung vorgesehen. Dabei ist die Anweisung wie folgt aufgebaut:

Definition

- *A* or *B* [or *C* ... usw.]

Argumente

- Alle Argumente (*A, B, C* ...) vom gleichen Datentyp. Datentypen aber sonst beliebig.
- Beliebig viele Verknüpfungen

Wirkung

- Die Variable *A* wird bitweise der Variablen *B* (und der Variablen *C* usw.) „verodert“, was heißt: Das Ergebnis der Operation or ist null, wenn beide der Variablen null sind. Im anderen Fall ist das Ergebnis eine bitweise Veroderung , d.h. das n-te Bit des Ergebnisses ist eins, alsbald eines der Bits der Eingangsvariablen eins ist.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Oder-Verknüpfung

LichtAktorEin ist das Ergebnis der Oder-Verknüpfung von Variable TasterEin und Variable LichtFreigabe

Die Umsetzung lautet dann:

```
LichtAktorEin = TasterEin or LichtFreigabe
```

Wenn *TasterEin* 1b01 ist oder *LichtFreigabe* 1b01 oder beide 1b01 sind, dann ist *LichtAktorEin* auf 1b01, sonst 0b01.

Beispiel: Oder-Verknüpfung bei unterschiedlichen Variablentypen

Wenn die Variable TasterEin auf '1' oder die Variable Windgeschwindigkeit genau 2.9 m/s ist, soll die Variable LichtAktorEin auf '1' gesetzt werden.

Für die Umsetzung benötigen wir die „if“ - Anweisung und den Vergleich „==“. Dabei ist die gesamte if-Abfrage in Klammern zu setzen. Die Umsetzung in der EibParserdatei lautet dann:

```
if ((TasterEin==1u08) or (Windgeschwindigkeit==2.9f16)) then LichtAktorEin=1u08 endif
```

Exklusiv-Oder

Zum Erstellen von Exklusiv-Oder-Verknüpfungen („Entweder oder“) ist die xor-Anweisung vorgesehen. Dabei ist die Anweisung wie folgt aufgebaut:

Definition

- *A* xor *B* [xor *C* ... usw.]

Argumente

- Alle Argumente (*A, B, C* ...) vom gleichen Datentyp. Datentypen aber sonst beliebig.
- Beliebig viele Verknüpfungen.

Wirkung

- Die Variable *A* wird bitweise der Variablen *B* (und der Variablen *C* usw.) „ver-x-odert“, was heißt: Das Ergebnis der Operation xor ist null (bitweise), wenn beide Variablen null oder eins sind. Im anderen Fall ist das n-te Bit des Ergebnisses eins, alsbald **nur** eines der Bits der Eingangsvariablen eins ist.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Xoder-Verknüpfung

Wenn entweder Taste1 (Typ b01) oder Taste2 (Typ b01) gedrückt ist, soll LichtAktorEin auf 1b01 gehen. Wenn beide auf 0b01 oder 1b01 stehen, soll LichtAktorEin auf 0b01 gehen

Die Umsetzung lautet dann:

```
LichtAktorEin = Taste1 xor Taste2
```

Vergleichsoperatoren

Zum Erstellen von Vergleichs-Verknüpfungen sind folgende Anweisungen vorgesehen.

Definition

- *A > B* größer
- *A < B* kleiner
- *A == B* gleich
- *A >= B* größer gleich
- *A =< B* kleiner gleich
- *A !=B* nicht gleich

Argumente

- Zwei Argumente (*A, B*) vom gleichen Datentyp.
- Datentypen: uXX,sXX,fXX, mit XX beliebigen, auf Seite 161 definierten Bitlängen.

Wirkung

- Die Variable *A* wird mit der Variablen *B* – je nach Operator – verglichen:

  Das Ergebnis der Operation > ist 1b01, wenn die Variable A größer als die Variable B ist.

  Das Ergebnis der Operation < ist 1b01, wenn die Variable A kleiner als die Variable B ist.

  Das Ergebnis der Operation == ist 1b01, wenn die Variable A den selben Wert hat wie die Variable B.

  Das Ergebnis der Operation >= ist 1b01, wenn die Variable A größer gleich der Variable B ist.

  Das Ergebnis der Operation =< ist 1b01, wenn die Variable A kleiner gleich der Variable B ist.

  Das Ergebnis der Operation != ist 1b01, wenn die Variable A nicht den selben Wert wie die Variabel B hat.

  In allen anderen Fällen ist das Ergebnis 0b01.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Hysterese-Vergleich

Definition

- Funktion hysteresis(*Var*,*UnteresLimit*,*OberesLimit*)

Argumente

- 3 Argumente (*Var*,*UnteresLimit*,*OberesLimit*) vom gleichen Datentyp.
- Datentypen: uXX,sXX,fXX, mit XX beliebigen, auf Seite 161 definierten Bitlängen.

Wirkung

- Das Argument *Var* wird mit den Variablen *UnteresLimit* und *OberesLimit* einer Hysterese-funktion verglichen.
- Hat der letzte Vergleich zu einem Ergebnis 0b01 geführt und gilt (*Var*≥*OberesLimit*), so nimmt die Funktion den Wert 1b01 an.
- Hat der letzte Vergleich zu einem Ergebnis 1b01 geführt und gilt (*Var*≥*UnteresLimit*), so nimmt die Funktion den Wert 0b01 an.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: Temperaturgeregelte Beschattung

Wenn ein Temperaturaktor wärmer als 25°C (Gruppenadresse 1/3/4, Datentyp f16) meldet soll die Beschattung auf Gruppenadresse 4/5/77 auf EIN fahren.
Erst wenn die Temperatur wieder unter 23°C fällt, soll die Beschattung wieder hochfahren.

Umsetzung im Anwenderprogramm:

```
if hysteresis('1/3/4'f16,23f16,25f16) then write('4/5/77'b01,EIN) \\
                                      else write('4/5/77'b01,AUS) endif
```

Invertierung

Zum Invertieren von binären Werten (Datentyp b01) steht Folgende Syntax zur Verfügung

Definition

- *!A*

Argumente

- Argument *A* vom Datentyp b01

Wirkung

- Die Variable *A* wird invertiert.
  Das Ergebnis der Operation ist 1b01, wenn die Variable A 0b01 ist
  Das Ergebnis der Operation ist 0b01, wenn die Variable A 1b01 ist

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: Invertierter Taster

```
LichtAktorEin (b01) soll sich invers zu Taste1 (b01) verhalten.
```

Die Umsetzung lautet dann:

```
LichtAktorEin = !Taste1
```

Wenn *Taste1* auf 1b01 steht, so ist *LichtAktorEin* auf 0b01. Wenn *Taste1* auf 0b01 steht, so ist *LichtAktorEin* auf 1b01.

Shift

Zum Shiften von numerischen Datentypen steht folgende Funktion zur Verfügung:

Definition

- shift*(Operand, Zahl)*

Argumente

- Argument *Operand* von einem beliebigen numerischen Datentyp
- Argument *Zahl* vom Datentyp s08

Wirkung

- arithmetische Verschiebung des Operanden um *Zahl*. Bei positiver Zahl Shift nach links, bei negativer Zahl nach rechts. Geshiftet werden die Anzahl Bits der Zahl des Eingangs.

Datentyp Ergebnis (Rückgabe)

- wie *Operand*

Systemzeit

Die Zeit und das Datum des Enertex® EibPC manuell setzen

Wenn Sie über keine Internetverbindung verfügen, können Sie am Enertex® EibPC die Zeit und das Datum manuell eingeben. Über EibPC – Uhrzeit und Datum gelangen Sie in den in Abbildung 117 gezeigten Dialog. Sie können in dem Dialog das aktuelle Datum und die aktuelle Uhrzeit eingeben und dann *OK* klicken. Alternativ dazu können Sie aber auch den Button *Systemzeit übernehmen* klicken. Daraufhin wird die Systemzeit des EibPCs auf die aktuelle Systemzeit des PCs, auf dem Eibstudio läuft, gesetzt.

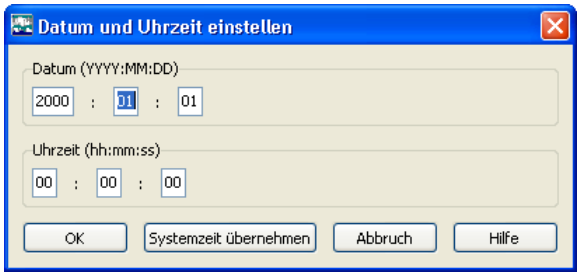


Abbildung 117: Datum und Uhrzeit manuell eingeben

Synchronisierung mit einem Internet-Zeitserver (NTP)

Das Network Time Protocol (NTP) ist ein Standard zur Synchronisierung von Uhren in Computersystemen über das Internet. Die lokale Uhr wird mit Hilfe von externen Zeitsignalen, die direkt von einer Atomuhr von einem NTP-Server gesendet werden, synchronisiert. Die Synchronisierung erfolgt als bald eine Internetverbindung besteht über die Adresse pool.ntp.org.

Synchronisierung mit dem KNX™-Bus

Die Systemuhr des Enertex® EibPC kann von außen gesteuert und gelesen werden. Dadurch lassen sich Aktoren, ect. im KNX Bus mit dem Enertex® EibPC synchronisieren oder die Systemuhr des Enertex® EibPC angepasst werden.

Es gibt sowohl die Möglichkeit, Daten vom Enertex® EibPC an den KNX™-Bus zu senden, als auch Daten vom KNX™-Bus an den Enertex® EibPC zu übermitteln.

Synchronisierung mit dem Anwendungsprogramm.

Zusätzlich gibt es die Möglichkeit, im Anwendungsprogramm die Systemzeit mit Hilfe von Variablen auszulesen und zu verändern.

Beachten Sie, dass durch das Setzen bzw. Verändern der Systemzeit die Timerfunktionen in ihrem Ablauf gestört werden können.

Priorität bei der Synchronisierung hat dabei immer der NTP Server, d.h. wird dieser angeschlossen, so wird die Zeit automatisch neu geschrieben.

Die Zeitsynchronisation kann im Menü unter Optionen → NTP-Zeitsynchronisation (Abb. 118) auch unterbunden werden. Deshalb muss angegeben werden, ob das Anwendungsprogramm erst nach der Zeitsynchronisation gestartet werden soll oder sofort nach Betriebsstart. Wenn nach fünf Minuten kein NTP-Server erreicht werden konnte, beginnt das Anwendungsprogramm trotzdem.

Standardmäßig startet das Anwendungsprogramm sobald der EibPC betriebsbereit ist.

Der NTP-Server synchronisiert sich alle 10 Minuten.

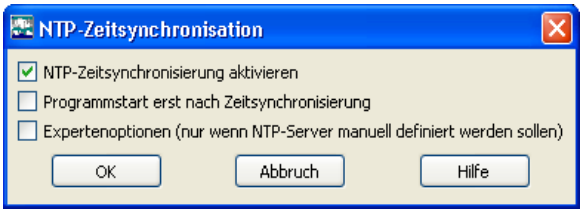


Abbildung 118: NTP Einstellungen konfigurieren

Die Zeit des Enertex® EibPC neu setzen

Definition

- Funktion gettime(*Adresse*) mit:

Argumente

- Ein Argument vom Datentyp t24

Wirkung

- Die Systemuhr des Enertex® EibPC wird mit der in *Adresse* gespeicherten Zeit überschrieben und somit neu gesetzt.

Datentyp Ergebnis (Rückgabe)

- keine

Hinweise:

1. Es ist keine Zuweisung der Form *a*=gettime(*b*) möglich (Fehlermeldung).
2. Die Funktion wird nur ausgeführt, wenn sie im then oder else Zweig einer if-Anweisung steht.

Beispiel: gettime

Wöchentlich am Sonntag um 00:00 Uhr soll die Systemuhr mit einer im KNX™-Bus vorhandenen Funkuhr abgeglichen und neu gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if(cwtime(0,0,0,0)) then read(“Funkuhr-1/2/1“) endif
if event(“Funkuhr-1/2/1“) then gettime(“Funkuhr-1/2/1“) endif
```

Durch die read-Funktion wird eine Leseanforderung an die Gruppenadresse generiert. Die Information, die daraufhin auf den KNX Bus gesendet wird, wird durch die gettime-Funktion in die Systemuhr des Enertex® EibPC geschrieben.

Die Zeit des Enertex® EibPC auf den KNX™-Bus schreiben

Definition

- Funktion settime()

Argumente

- keines

Wirkung

- Die Systemzeit wird aus dem Enertex® EibPC gelesen und einer Variablen als Wert zugewiesen. Rückgabewert ist die aktuelle Uhrzeit im DPT-Format.

Datentyp Ergebnis (Rückgabe)

- Datentyp t24

Beispiel 1: settime

Am 1. jeden Monats soll die Gruppenadresse "Wanduhr-4/3/5" und die Variable Uhrzeit mit der Systemuhr abgeglichen (und damit neu gesetzt) werden.

Umsetzung im Anwenderprogramm:

```
if (day(1)) then write(„Wanduhr-4/3/5“t24,settime()) endif
if (day(1)) then Uhrzeit=settime() endif
```

Das Datum des Enertex® EibPC neu setzen

Definition

- Funktion getdate(*Adresse*) mit:

Argumente

- Ein Argument vom Datentyp d24.

Wirkung

- Die Systemuhr des Enertex® EibPC wird mit der in *Adresse* gespeicherten Zeit überschrieben und somit neu gesetzt.

Datentyp Ergebnis (Rückgabe)

- keiner

Hinweise:

1. Es ist keine Zuweisung der Form *a*=gettime(*b*) möglich (Fehlermeldung).
2. Die Funktion wird nur ausgeführt, wenn sie im then oder else Zweig einer if-Anweisung steht.

Beispiel: GetDate

Alle sechs Monate soll das Systemdatum mit einer im KNX Bus vorhandenen Funkuhr abgeglichen und neu gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if (month(1,1) or month(1,7)) then read(“Funkuhr-1/2/2“) endif
if event(“Funkuhr-1/2/2“) then getdate(“Funkuhr-1/2/2“) endif
```

Das Datum des Enertex® EibPC auf den KNX™-Bus schreiben

Definition

- Funktion setdate()

Argumente

- keines

Wirkung

- Das Systemdatum wird aus dem Enertex® EibPC gelesen. Der Rückgabewert ist die Zeit im Format des Typ d24.

Datentyp Ergebnis (Rückgabe)

- Datentyp d24

Beispiel: SetDate

Am 1.1. jeden Jahres soll die Adresse “Datum-3/5/3“ mit dem Datum des Enertex® EibPC abgeglichen und neu gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if (month(1,1)) then write(“Datum-3/5/3“d24, setdate()) endif
```

GetTimeDate - Die Zeit und das Datum des Enertex® EibPC neu setzen

Definition

- Funktion gettimedate(*Adresse*) mit:

Argumente

- Ein Argument vom Datentyp y64

Wirkung

- Die Systemuhr und das Systemdatum des Enertex® EibPC werden mit der in *Adresse* gespeicherten Zeit und dem Datum überschrieben und somit neu gesetzt.

Datentyp Ergebnis (Rückgabe)

- keine

Hinweise:

1. Es ist keine Zuweisung der Form *a*=gettimedate(*b*) möglich (Fehlermeldung)
2. Die Funktion wird nur ausgeführt, wenn sie im then oder else Zweig einer if-Anweisung steht.

Beispiel: GetTimeDate

Alle sechs Monate sollen die Systemzeit und das Systemdatum mit einer im KNX Bus vorhandenen Funkuhr abgeglichen und neu gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if (month(1,1) or month(1,7)) then read(“Funkuhr-1/2/3“) endif
if event(“Funkuhr-1/2/3“) then gettimedate(“Funkuhr-1/2/3“) endif
```

SetTimeDate - Die Zeit und das Datum des Enertex® EibPC auf den KNX™-Bus schreiben

Definition

- Funktion settimedate()

Argumente

- keines

Wirkung

- Die Systemzeit und das Systemdatum werden aus dem Enertex® EibPC gelesen und einer Variable als Wert zugewiesen.

Datentyp Ergebnis (Rückgabe)

- Datentyp y64

Beispiel: SetDate

Am 1.1. jeden Jahres soll die Adresse “Funkuhr-1/2/1“ mit der Systemzeit und dem Systemdatum des Enertex® EibPC abgeglichen und neu gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if (month(1,1)) then write(“Funkuhr-1/2/1“y64, settimedate()) endif
```

Hour

Definition

- Funktion hour()

Argumente

- keines

Wirkung

- Die Systemzeit (Stunde) wird in eine Variable gespeichert

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Beispiel:

Stoppuhr siehe S. 184

Minute

Definition

- Funktion minute()

Argumente

- keines

Wirkung

- Die Systemzeit (Minute) wird in eine Variable gespeichert

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Beispiel:

Stoppuhr siehe S. 184

Second

Definition

- Funktion second()

Argumente

- keines

Wirkung

- Die Systemzeit (Sekunde) wird in eine Variable gespeichert

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Beispiel:Stoppuhr

Stoppen der Zeit in Sekunden, an welcher die Variable Stopper_Go auf ein steht. Es soll ein c1400 String angegeben werden, der die Zeit Formatiert in 000d:000h:000m:000s (Tage, Stunden, Minuten, Sekunden) ausgibt.

Hier die Umsetzung, wobei die Sekunden in der Variablen *Stopper_time* und die formatierte Ausgabe in *Stopper* zu finden sind. Vgl. Sie hierzu auch das Beispiel Stoppuhr V2 auf S. 233.

Stringformat für die formatierte Ausgabe/Umwandlung

```
[EibPC]
Stopper=$$
Stopper_start=0s32
Stopper_time=1s32
Stopper_Go=AUS

// Starte Stoppuhr (Offset berechnen)
if (Stopper_Go) then {
        Stopper_start=-convert(hour(),0s32)*3600s32-convert(minute(),0s32)*60s32-
convert(second(),0s32)
} endif
if change(dayofweek()) then Stopper_start=Stopper_start+86400s32 endif

// Ende Stoppzeit
if !Stopper_Go then {

Stopper_time=convert(hour(),0s32)*3600s32+convert(minute(),0s32)*60s32+convert(second(),
0s32)+Stopper_start;
        Stopper=stringformat(Stopper_start/86400s32,0,3,3,3)+$d:$+\\
            stringformat(mod(Stopper_start,86400s32)/3600s32,0,3,3,3)+$h:$+\\
            stringformat(mod(Stopper_start,3600s32)/60s32,0,3,3,3)+$m:$+\\
            stringformat(mod(Stopper_start,60s32),0,3,3,3)+$s$
} endif
```

Changehour

Definition

- Funktion changehour(*Arg*)

Argumente

- *Arg*, Datentyp u08

Wirkung

- Die Systemzeit (Stunde) wird auf den Wert von *Arg* gesetzt.
- Beachten Sie, dass durch das Setzen bzw. Verändern der Systemzeit die Timerfunktionen in ihrem Ablauf gestört werden können.
- Sollte ihr Enertex® EibPC eine ntp-Verbindung aufbauen können, so wird diese die Zeit wieder zurücksetzen (vgl. Hinweis auf S. 180).

Datentyp Ergebnis (Rückgabe)

- keine

Changeminute

Definition

- Funktion changeminute(*Arg*)

Argumente

- *Arg*, Datentyp u08

Wirkung

- Die Systemzeit (Minute) wird auf den Wert von *Arg* gesetzt.
- Beachten Sie, dass durch das Setzen bzw. Verändern der Systemzeit die Timerfunktionen in ihrem Ablauf gestört werden können.
- Sollte ihr Enertex® EibPC eine ntp-Verbindung aufbauen können, so wird diese die Zeit wieder zurücksetzen (vgl. Hinweis auf S. 180).

Datentyp Ergebnis (Rückgabe)

- keine

Changesecond

Definition

- Funktion changesecond(*Arg*)

Argumente

- *Arg*, Datentyp u08

Wirkung

- Die Systemzeit (Sekunde) wird auf den Wert von *Arg* gesetzt.
- Beachten Sie, dass durch das Setzen bzw. Verändern der Systemzeit die Timerfunktionen in ihrem Ablauf gestört werden können.
- Sollte ihr Enertex® EibPC eine ntp-Verbindung aufbauen können, so wird diese die Zeit wieder zurücksetzen (vgl. Hinweis auf S. 180).

Datentyp Ergebnis (Rückgabe)

- keine

Utc

Definition

- Funktion utc(*Zeit*)

Argumente

- *Zeit* als String im Forma $*YYYY-MM-DD HH:MM:SS*$, Datentyp c1400

Wirkung

- Wandelt eine Zeitangabe im YYYY-MM-DD HH:MM:SS Format zurück in einen UTC-Wert. Dieser Wert ist kompatibel zum Unix-Zeitstempel, wird allerdings im Gegensatz zu diesen statt in Sekunden in Millisekunden angegeben.

Datentyp Ergebnis (Rückgabe)

- u64

Utctime

Definition

- Funktion utctime()

Argumente

- keine

Wirkung

- Gibt die Anzahl der verstrichenen Millisekunden seit Jan 1, 1970 00:00 an. Die Funktion ist kompatibel zum Unix-Zeitstempel.

Datentyp Ergebnis (Rückgabe)

- u64

Utcconvert

Definition

- Funktion utcconvert(*utc*)

Argumente

- *utc* Zeit in ms, Datentyp u64

Wirkung

- Wandelt eine Zeitangabe im utc-Format in ein c1400 String-Format
  YYYY-MM-DD HH:MM:SS
  um.

Datentyp Ergebnis (Rückgabe)

- String

Beispiel: UTC Umwandlung

Umsetzung im Anwenderprogramm:

```
// Gibt aktuelle Zeitangabe im UTC-Format zurück
utcZeit=utctime()

// Convertiert UTC-Format in YYYY-MM-DD HH:MM:SS
DateTime=utcconvert(1364826122000u64)

// Wandelt  2012-09-03 20:00.17 in UTC-Format um
utcZ=utc($2012-09-03 20:00:17$)
```

Datumssteuerung

Date- Datumsvergleich

Ein Datumsvergleich ist wie folgt definiert:

Definition

- Funktion date(*dd,mm,yyy*) mit:
  dd: Tag (1..31)
  mm: Monat (1=Januar, 12=Dezember)
  yyy: Jahresdifferenz (0..255) vom Jahr 2000 an

Argumente

- Alle vom Datentyp u08

Wirkung

- Der Ausgang ist 1b01, wenn das Datum erreicht oder bereits verstrichen ist. Liegt das Datum vor dem eingestellten Wert, geht der Ausgang auf 0.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: Datumsvergleichszeitschaltuhr

Am 01. Oktober 2009 soll die Variable EinzugEin auf 1u08 gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if date(10,1,09) then EinzugEin=1 endif
```

Month - Monatsvergleich

Ein Monatsvergleich ist wie folgt definiert:

Definition

- Funktion month(*dd,mm*) mit:
  dd: Tag (1..31)
  mm: Monat (1=Januar, 12=Dezember)

Argumente

- Zwei Argumente vom Datentyp u08

Wirkung

- Der Ausgang ist 1b01, wenn das Datum erreicht oder bereits verstrichen ist. Liegt das Datum vor dem eingestellten Wert, geht der Ausgang auf 0b01. Mit dem Beginn eines neuen Jahres (Januar, 1) geht der Ausgang auf 0b01, bis der Monat und Tag den eingestellten Wert erreichen.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: Monatsvergleichszeitschaltuhr

Jedes Jahr am 01. Dezember soll die Variable WeihnachtenBeleuchtungEin auf 1 gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if month(1,12) then WeihnachtenBeleuchtungEin=1 endif
```

Beispiel: Variablendefinition „Sommer“

Es solle eine Variable Sommer definiert werden, die vom 1.5. bis 30.9. eines jeden Jahres auf 1b01 (EIN) steht.

Umsetzung im Anwenderprogramm:

```
Sommer=month(01,05) and !month(30,09)
```

Day - Tagesvergleich

Ein Tagesvergleich ist wie folgt definiert:

Definition

- Funktion day(*dd*) mit:
  dd: Tag (1..31)

Argumente

- Argument Datentyp u08

Wirkung

- Der Ausgang ist 1b01, wenn der Tag erreicht oder bereits verstrichen ist. Liegt der Tag vor dem eingestellten Wert, geht der Ausgang auf 0b01. Mit dem Beginn eines neuen Monats geht der Ausgang auf 0b01, bis der Tag den eingestellten Wert erreicht.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: Tagesvergleichszeitschaltuhr

Jeden 6. im Monat soll die Variable RasensprengerEin auf 1 gesetzt werden.

Die Umsetzung in der EibParserdatei lautet dann:

```
if day(6) then RasensprengerEin=1 endif
```

DayOfWeek – Wochentag

Um den aktuellen Tag der internen Uhr des Enertex® EibPC abzufragen nutzen Sie die Funktion

Definition

- Funktion dayofweek() mit:

Argumente

- keine

Wirkung

- Der Ausgang gibt den aktuellen Wochentag [0{Sonntag}..6{Samstag} zurück.

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Beispiel: Tagesvergleichszeitschaltuhr

Abfrage, ob heute Sonntag ist und wenn ja, dann die Variable RasensprengerEin auf EIN setzten

Die Umsetzung in der EibParserdatei lautet dann:

```
if dayofweek()==SONNTAG then RasensprengerEin=1 endif
```

Easterday

Definition

- Funktion easterday(*Offset*)

Argumente

- Argument *Offset* Datentyp s16

Wirkung

- Tag des Ostersonntags berechnen. Dabei wird ein *Offset* für das Berechnungsergebnis angegeben, z.B. Ostersonntag +40 Tage, Ostersonntag – 30 Tage.

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Eastermonth

Definition

- Funktion eastermonth(*Offset*)

Argumente

- Argument *Offset* Datentyp s16

Wirkung

- Monat des Ostersonntags berechnen. Dabei wird ein *Offset* für das Berechnungsergebnis angegeben, z.B. Ostersonntag +40 Tage, Ostersonntag – 30 Tage.

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Beispiel: Berechnung des Aschermittwochs; (Aschermittwoch ist 46 Tage vor Ostersonntag:)

```
uAschermittwochTag=easterday(-46s16)
uAschermittwochMonat=eastermonth(-46s16)
```

Beschattung und Sonnenstand

Sun - Sonnenstand

Die Sonnenstand-Funktion gibt aus, ob es Tag oder Nacht ist. Das Programm muss dazu die geographische Länge und Breite des betreffenden Ortes kennen. Diese können wie auf S. 157 Beschrieben, im Enertex® EibStudio eingegeben werden.

Definition

- Funktion sun()

Argumente

- keine

Wirkung

- Rückgabewert: Der Rückgabewert ist 1 binär, wenn es Tag ist und 0 binär, wenn es Nacht ist.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel 2: Sonnenstand

Wenn es Tag ist soll die Variable JalousieEin auf 0 gesetzt werden.

Die Umsetzung im Anwenderprogramm lautet dann:

```
if (sun()==1b01) then JalousieEin=0 endif
if (sun()==HELL) then JalousieEin=0 endif
```

“HELL“ ist eine vordefinierte Variable (s. Seite 348) mit dem Binärwert 1b01 und kann deshalb als Vergleichsoperator an Stelle von 1b01 angegeben werden.

Azimuth

Definition

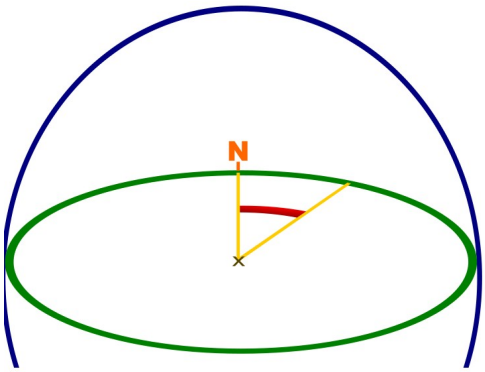
- Funktion azimuth()

Argumente

- keine. Der Enertex® EibPC muss die geographische Länge und Breite des betreffenden Ortes kennen. Diese können im Enertex® EibStudio eingegeben werden (s. Seite 157).

Wirkung

- Diese Funktion berechnet zyklisch (Zeitraster: 5 Minuten) den Azimuthwinkel der Sonne in Grad, Nord über Ost.



(Quelle: Wikipedia)

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel 3: azimuth berechnen

Berechnen Sie alle 5 Minuten Azimutwinkel der Sonne am Aufstellungsort des Enertex® EibPC

Die Umsetzung in der EibParserdatei lautet dann:

```
AWinkel=azimuth()
```

Hinweis:

Diese Funktion wird bei Hausbeschattungen benötigt. In der Bibliothek EnertexBeschattung.lib finden Sie ausführliche Beispiele.

Elevation

Definition

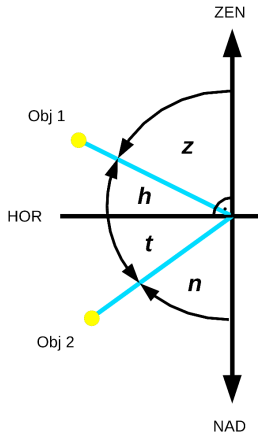
- Funktion elevation()

Argumente

- keine. Der Enertex® EibPC muss jedoch die geographische Länge und Breite des betreffenden Ortes kennen. Diese können im Enertex® EibStudio eingegeben werden (s. Seit 157).

Wirkung

- Diese Funktion berechnet zyklisch (Zeitraster: 5 Minuten) den Höhenwinkel der Sonne in Grad.



(Quelle: Wikipedia)

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel 4: elevation

Berechnen Sie um 5:00 den Höhenwinkel der Sonne am Aufstellungsort des Enertex® EibPC

Die Umsetzung in lautet:

```
HWinkel=0f32
if htime(5,00,00) then HWinkel=elevation() endif
```

Hinweis:

Diese Funktion wird bei Hausbeschattungen benötigt. In der Bibliothek EnertexBeschattung.lib finden Sie ausführliche Beispiele.

Presun

Definition

- Funktion presun(*hh,mm*)
  hh: Stunden (0... 23)
  mm: Minuten (0... 59)

Argumente

- zwei Argument vom Datentyp u08

Wirkung

- Gibt in der angegeben Zeit vor dem Wechsel auf Tag eine 1 aus, bzw. vor dem Wechsel auf Nacht eine 0 aus. Das Programm muss dazu die geographische Länge und Breite des betreffenden Ortes kennen.

Datentyp Ergebnis (Rückgabe)

- Sonnenstand, 1= Tag, 0 = Nacht vom Datentyp u08

```
s=$$
if presun(1,30) then s=$Eine Stunde vor Sonnenaufgang$ endif
if !presun(0,20) then s=$20 Minuten vor Sonnenuntergang$ endif
```

Sunrisehour - Stunde des Sonnenaufgangs

Definition
- Funktion sunrisehour()

Argumente
- keine

Wirkung
- Rückgabewert: Der Rückgabewert ist die Stunde (0 bis 23), zu der die Sonne aufgeht.

Datentyp Ergebnis (Rückgabe)
- Datentyp u08

Sunriseminute - Minute des Sonnenaufgangs

Definition
- Funktion sunriseminute()

Argumente
- keine

Wirkung
- Rückgabewert: Der Rückgabewert ist die Minute (0 bis 59), zu der die Sonne aufgeht.

Datentyp Ergebnis (Rückgabe)
- Datentyp u08

Beispiel: Sonnenaufgang visualisieren

Schreiben Sie auf die Gruppenadresse 1/4/4 (Typ c14) die Zeit für den Sonnenaufgang.

Dies setzen Sie wie folgt um:

```
if htime(sunrisehour(),sunriseminute(),0) then \\
  write('1/4/4'c14, convert(sunrisehour(),$$c14)+$:$c14+convert(sunriseminute(),$$c14))  \\
endif
```

Sunsethour - Stunde des Sonnenuntergang

Definition
- Funktion sunsethour()

Argumente
- keine

Wirkung
- Rückgabewert: Der Rückgabewert ist die Stunde (0 bis 23), zu der die Sonne am aktuellen Tag untergeht.

Datentyp Ergebnis (Rückgabe)
- Datentyp u08

Sunsetminute - Minute des Sonnenuntergangs

Definition
- Funktion sunsetminute()

Argumente
- keine

Wirkung
- Rückgabewert: Der Rückgabewert ist die Minute (0 bis 59), zu welcher die Sonne am aktuellem Tag untergeht.

Datentyp Ergebnis (Rückgabe)
- Datentyp u08

Beispiel: siehe obiges Beispiel „Sonnenaufgang visualisieren“

```
if htime(sunsethour(),sunsetminute(),0) then \\
  write('1/4/4'c14, convert(sunsethour(),$$c14)+$:$c14+convert(sunsetminute(),$$c14))  endif
```

Zeitschaltuhren

Tageszeitschaltuhr

Zeitschaltuhren sind Funktionen, die beim Eintreten der angegebenen Tageszeit für einen Verarbeitungszyklus des Enertex® EibPC ihren Rückgabewert von AUS auf EIN und dann wieder auf AUS wechseln. Zeitschaltuhren sind Objekte, die regelmäßige Aktionen anstoßen, z.B. jede Nacht um 1:00 Uhr wird die Garagenbeleuchtung ausgeschaltet etc..

Um die Anwendung einfach zu gestalten, unterscheiden wir vier Typen von Zeitschaltuhren:

- Die Wochenzeitschaltuhr, die eine Aktion pro Woche anstößt,
- die Tageszeitschaltuhr, die pro Tag eine Aktion ausführt,
- die Stundenzeitschaltuhr, die pro Stunde aktiv wird und schließlich
- die Minutenzeitschaltuhr, die pro Minute eine Aktion anstößt.

Um die Aktion auszuführen, muss die Zeitschaltuhr exakt den Zeitpunkt durchlaufen. Dies ist bei der Programmierung zu berücksichtigen. Als Referenzzeit für alle Zeitschaltuhren wird die Systemzeit des Enertex® EibPC verwendet (siehe auch Seite 156), die entweder über das Internet oder per KNX™-Systemgerät dem Enertex® EibPC vorgegeben wird.

Definition

- wtime(*hh,mm,ss,dd*) mit:
  hh: Stunden (0..23)
  mm: Minuten (0..59)
  ss: Sekunden (0..59)
  dd: Tag (0=Sonntag, 6=Samstag,7=Werktags, 8=Wochenende)

Argumente

- 4 Argumente vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b01, wenn die aktuelle Zeit und der Tag der Enertex® EibPC-Systemuhr nicht gleich *hh*:*mm*:*ss* und Tag *dd* sind. Wenn der Zeitpunkt eintrifft (und exakt übereinstimmt), geht der Ausgabewert auf 1b01. (Wenn der Zeitpunkt überschritten wird, wieder auf 0b01)

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel Wochenzeitschaltuhr

Jeden Dienstag, 01:00 Uhr, 30 Sekunden, soll die Variable LichtAktorEin auf 0b01 gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if wtime(01,00,30,DIENSTAG) then LichtAktorEin=0b01 endif
```

Hinweis:

Für die Tage und Wochenende und Werktags sind Konstanten (geschrieben in Großbuchstaben) definiert (MONTAG, DIENSTAG, WERKTAGS, WOCHENENDE etc.)

Definition

- htime(*hh,mm,ss*) mit:
  hh: Stunden (0..23)
  mm: Minuten (0..59)
  ss: Sekunden (0..59)

Argumente

- 3 Argumente vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b01, wenn die aktuelle Zeit der Enertex® EibPC-Systemuhr nicht gleich *hh*:*mm*:*ss* ist. Wenn der Zeitpunkt eintrifft (und exakt übereinstimmt), geht der Ausgabewert auf 1b01. (Wenn der Zeitpunkt überschritten wird, wieder auf 0b01)

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel Tageszeitschaltuhr

Jeden Tag, 22:04 Uhr, 7 Sekunden, soll die Variable LichtAktorEin auf '0' gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if htime(22,04,07) then LichtAktorEin=0b01 endif
```

Stundenzeitschaltuhr

Eine Stundenzeitschaltuhr ist wie folgt definiert:

Definition

- mtime(*mm,ss*) mit:
  mm: Minuten (0..59)
  ss: Sekunden (0..59)

Argumente

- Zwei Argumente vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b01, wenn die aktuelle Minuten-Sekunden-Zeit der Enertex® EibPC-Systemuhr nicht gleich *mm*:*ss* ist (Stunde beliebig). Wenn der Zeitpunkt eintrifft (und exakt übereinstimmt), geht der Ausgabewert auf 1b01. (Wenn der Zeitpunkt überschritten wird, wieder auf 0b01).

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel Stundenzeitschaltuhr

Stündlich, immer nach 22 Minuten, 7 Sekunden nach einer vollen Stunde, soll die Variable LichtAktorEin auf '0' gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if mtime(22,07) then LichtAktorEin=0b01 endif
```

Minutenzeitschaltuhr

Eine Minutenzeitschaltuhr ist wie folgt definiert:

Definition

- stime(*ss*) mit:
  ss: Sekunden (0..59)

Argumente

- Ein Argumente vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b01, wenn die aktuelle Sekunden-Zeit der Enertex® EibPC-Systemuhr nicht gleich *ss* ist (Stunde, Minute beliebig). Wenn der Zeitpunkt eintrifft (und exakt übereinstimmt), geht der Ausgabewert auf 1b01. (Wenn der Zeitpunkt überschritten wird, wieder auf 0b01).

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel Minutenzeitschaltuhr

Immer nach 34 Sekunden nach einer vollen Minute soll die Variable Fensterkontakte auf '0' gesetzt werden.

Immer nach 5 Sekunden nach einer vollen Minute, soll die Variable auf '1' gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if stime(34) then Fensterkontakte=0 endif
if stime(5) then Fensterkontakte=1 endif
```

Vergleichszeitschaltuhren

Grundsätzliches

Vergleichszeitschaltuhren sind Objekte, die einen Zeitvergleich ermöglichen. Abhängig von dem Ergebnis des Vergleichs lässt sich dann ein Bustelegramm anstoßen, z.B. jede Nacht von 1:00 Uhr bis 6:00 Uhr wird die Garagenbeleuchtung ausgeschaltet. Wenn der eingestellte Zeitpunkt erreicht ist, sind sie bis zum nächsten Tag auf 1b01, im Gegensatz zur den Schaltzeituhren, die nur zum Zeitpunkt selbst auf 1b01 springen und danach sofort wieder auf 0b01. Damit sind Vergleichszeitschaltuhren sehr ähnlich zu den üblicheren Zeitschaltuhren, haben aber den Vorteil, dass der Zeitpunkt nicht exakt eingetroffen sein muss (z.B. Stromausfall, Neustart).

Als Referenzzeit für alle Vergleichszeitschaltuhren wird die Systemzeit des Enertex® EibPC verwendet (siehe auch Seite 156), die entweder über das Internet oder per KNX™-Systemgerät dem Enertex® EibPC vorgegeben wird.

Um die Anwendung einfach zu gestalten, unterscheiden wir vier Typen von Vergleichszeitschaltuhren:

- Die Wochenvergleichszeitschaltuhr, die eine Aktion pro Woche anstößt,
- die Tagesvergleichszeitschaltuhr, die pro Tag eine Aktion ausführt,
- die Stundenvergleichszeitschaltuhr, welche pro Stunde aktiv wird und schließlich
- die Minutenvergleichszeitschaltuhr, welche pro Minute eine Aktion anstößt.

Wochenvergleichszeitschaltuhr

Eine Wochenvergleichszeitschaltuhr ist wie folgt definiert:

Definition

- cwtime(*hh,mm,ss,dd*) mit:
  ss: Sekunden (0..59)
  mm: Minuten (0..59)
  hh: Stunden (0..23)
  dd: Tag (0=Sonntag, 6=Samstag)

Argumente

- 4 Argumente vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b00, wenn die aktuelle Zeit und der Tag der Enertex® EibPC-Systemuhr vor *hh*:*mm*:*ss* und Tag *dd* liegen. Nach diesem Zeitpunkt geht der Ausgabewert auf 1b01 und bleibt auf diesem Wert bis zum nächsten Sonntag, 00:00:00 Uhr.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: Wochenvergleichszeitschaltuhr

Jede Woche ab Dienstag, 01:00 Uhr, 30 Sekunden, soll die Variable LichtAktorEin auf '0' gesetzt werden. Mit Beginn einer neuen Woche soll die Variable wieder auf '1' gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if cwtime(01,00,30,DIENSTAG) then LichtAktorEin=0 else LichtAktorEin=1 endif
```

Hinweis:

1. Für alle Tage einzeln, das Wochenende und Werktags sind Konstanten (geschrieben in Großbuchstaben) definiert, z.B.

   ```
   if cwtime(01,00,30,WERKTAGS) then LichtAktorEin=0 else LichtAktorEin=1 endif
   ```

2. cwtime geht am Schalttag auf 1b01 und bleibt dies bis Sonntag 24:00. Wenn daher MONTAG oder WERKTAGS angegeben wird, ist cwtime immer auf 1b01.

Tagesvergleichszeitschaltuhr

Eine Tagesvergleichszeitschaltuhr ist wie folgt definiert:

Definition

- chtime(*hh,mm,ss*) mit:
  ss: Sekunden (0..59)
  mm: Minuten (0..59)
  hh: Stunden (0..23)

Argumente

- 3 Argumente vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b01, wenn die aktuelle Zeit der Enertex® EibPC-Systemuhr nicht gleich *hh*:*mm*:*ss* ist. Wenn der Zeitpunkt eintrifft, geht der Ausgabewert auf 1b01 und bleibt auf diesem Wert bis zum nächsten Tag (d.h. 00:00:00 Uhr).

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel Tagesvergleichszeitschaltuhr

Jeden Tag ab 22:04 Uhr, 7 Sekunden, soll die Variable LichtAktorEin auf '0' gesetzt werden. Mit dem Beginn eines neuen Tages wird die Variable wieder auf '1' gesetzt.

Umsetzung im Anwenderprogramm:

```
if chtime(22,04,07) then LichtAktorEin=0 else LichtAktorEin=1 endif
```

Stundenvergleichszeitschaltuhr

Eine Stundenvergleichszeitschaltuhr ist wie folgt definiert:

Definition

- cmtime(*mm,ss*) mit:
  ss: Sekunden (0..59)
  mm: Minuten (0..59)

Argumente

- Zwei Argumente vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b01, wenn die aktuelle Minuten-Sekunden-Zeit der Enertex® EibPC-Systemuhr nicht gleich *mm*:*ss* ist. Wenn der Zeitpunkt eintrifft, geht der Ausgabewert auf 1b01 und bleibt auf diesem Wert bis zur nächsten Stunde.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel Stundenvergleichszeitschaltuhr

Stündlich, immer nach 22 Minuten, 7 Sekunden, soll die Variable LichtAktorEin auf '0' gesetzt werden. Zur vollen Stunde soll die Variable wieder auf '1' gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if cmtime(22,07) then LichtAktorEin=0 else LichtAktorEin=1 endif
```

Minutenvergleichszeitschaltuhr

Eine Minutenvergleichszeitschaltuhr ist wie folgt definiert:

Definition

- cstime(*ss*) mit:
  ss: Sekunden (0..59)

Argumente

- Ein Argument vom Datentyp u08

Wirkung

- Der Rückgabewert ist 0b01, wenn die aktuelle Sekunden-Zeit der Enertex® EibPC Systemuhr nicht gleich *ss* ist. Wenn der Zeitpunkt eintrifft, geht der Ausgabewert auf 1b01 und bleibt auf diesem Wert, bis zur nächsten Minute.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel Sekundenvergleichszeitschaltuhr

Immer nach 34 Sekunden nach einer vollen Minute soll die Variable Fensterkontakte auf '0' gesetzt werden. Zu Beginn einer neuen Minute bis zum Erreichen der eingestellten Zeit soll die Variable auf '1' gesetzt werden.

Umsetzung im Anwenderprogramm:

```
if cstime(34) then Fensterkontakte=0 else Fensterkontakte=1 endif
```

Spezielle Zeitfunktionen

Präzisionstimer - programmierbare Verzögerungszeit

Mit Hilfe von delay und after können sehr kurze Zeitkonstanten generiert werden, wie sie z.B. bei der Steuerung von Bewegungsmeldern (Leuchtdauer, Entprellen gegen Wiedereinschalten) oder bestimmten Regelalgorithmen notwendig werden. Der Enertex® EibPC selbst reagiert im Mikrosekundenbereich.

Die minimale Verzögerungszeit beträgt 1 ms, die maximal einstellbare Verzögerungszeit beträgt ca. 30 Jahre.

Delay

Definition

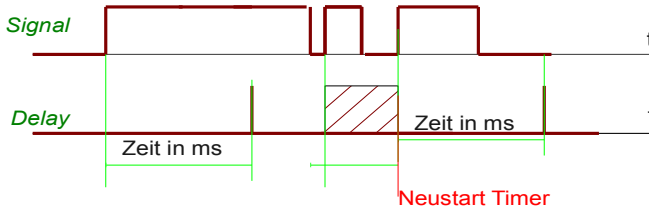
- Funktion delay(*Signal*, *Zeit*)

Argumente

- Argument *Signal* vom Datentyp b01
- Argument *Zeit* vom Datentyp u64

Wirkung

- Die Funktion startet beim Übergang der Variablen *Signal* von AUS auf EIN einen Timer. Nach Ablauf der *Zeit* in ms erzeugt die Delay-Funktion einen EIN-Impuls.



- Wenn ein neuer AUS-EIN Impuls auftritt, während der interne Timer läuft, wird der Timer neu gestartet.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Hinweis:

- delay darf nicht im then oder else Zweig einer if-Anweisung stehen.
- Wenn der Enertex® EibPC mit delay (mit Hilfe einer if-Anweisung und write) ein Telegramm auf den Bus schreibt, kann je nach Buslast und Busgeschwindigkeit eine zusätzliche (variable) Verzögerungszeit von wenigen ms entstehen, bis dieses dort erscheint. Verzögerungszeiten von wenigen ms entstehen, bis dieses dort erscheint.

Beispiel: verzögerte Variablenzuweisung

Wenn die Variable LichtAktor (Datetyp f16) kleiner als 1000f16 ist, soll nach 10s die Variable Licht (Datentyp b01) auf EIN gehen

Umsetzung im Anwenderprogramm:

```
Licht=AUS
if delay(LichtAktor<1000f16,10000u64) then Licht=EIN endif
```

Beispiel: Einschaltverzögerung

Wenn LichtTaster (Typ b01) EIN ist, soll nach 1300 ms die Variable LichtAktor (Typ b01) auf EIN gehen.

Umsetzung im Anwenderprogramm:

```
if delay(LichtTaster,1300u64) then LichtAktor=1b01 endif
```

Alternative 1

```
if delay(LichtTaster==1b01,1300u64) then LichtAktor=1b01 endif
```

Alternative 2

```
if (delay(LichtTaster,1300u64)==1b01) then LichtAktor=1b01 endif
```

Man beachte, dass “LichtAktor“ nur gesetzt, nicht aber gelöscht wird. Siehe dazu auch das folgende Beispiel.

Beispiel: Ausschaltverzögerung

Wenn LichtTaster (Datentyp b01) *AUS* ist, soll nach 4000 ms die Variable LichtAktor auf *AUS* gehen.

Die Umsetzung in der EibParserdatei lautet dann:

```
if (delay(LichtTaster==AUS,4000u64)) then LichtAktor=0b01 endif
```

Beispiel: Verschiedene Ein- und Ausschaltverzögerung

Wenn LichtTaster (Datentyp b01) *EIN* ist soll nach 1300 ms die Variable LichtAktor (Datentyp b01) auf *EIN* ist, wenn LichtTaster (Datentyp b01) *AUS* ist, soll nach 4000 ms die Variable LichtAktor auf *AUS* gehen.

Umsetzung im Anwenderprogramm:

```
if (delay(LichtTaster==EIN,1300u64)) then LichtAktor=EIN endif
if (after(LichtTaster==AUS,4000u64)) then LichtAktor=AUS endif
```

Delayc

Definition

- Funktion delayc(*Signal*, *Zeit,xT*)

Argumente

- Argument *Signal* vom Datentyp b01
- Argument *Zeit* vom Datentyp u64
- Argument *xT* vom Datentyp u64

Wirkung

- Wie delay (S. 198).
- Über die Variable *xT* kann die verbleibende Zeit ausgefragt werden.
  ACHTUNG: Wenn Sie dieselbe Variable *xT* für verschiedene Funktionen delayc im Programmcode nutzen, führt dies zu einem undefiniertem Verhalten.
- Die Initialisierung der Variable *xT* wird bei Aktivierung des Timers überschrieben.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Hinweis:

- delayc darf nicht im then oder else Zweig einer if-Anweisung stehen.
- Wenn der Enertex® EibPC mit delayc (mit Hilfe einer if-Anweisung und write) ein Telegramm auf den Bus schreibt, kann je nach Buslast und Busgeschwindigkeit eine zusätzliche (variable) Verzögerungszeit von wenigen ms entstehen, bis dieses dort erscheint.

Beispiel Verbleibende Zeit ausfragen:

Wenn LichtTaster (Typ b01) EIN ist, soll nach 4000 ms die Variable LichtAktor (Typ b01) auf EIN gehen. Die noch verbleibende Zeit - gemessen ab LichtTaster == EIN bis Ende der Verzögerung 4000 ms - soll alle 200ms auf die Adresse '2/2/2' geschrieben werden.

```
xT=0u64
debug='2/2/2'u64
// Der Timer kann nun über xT ausgefragt werden
if (delayc(LichtTaster==EIN,4000u64),xT) then LichtAktor=1b01 endif
if (change(xT/200u64)) then write('2/2/2'u64, xT) endif
```

After

Definition

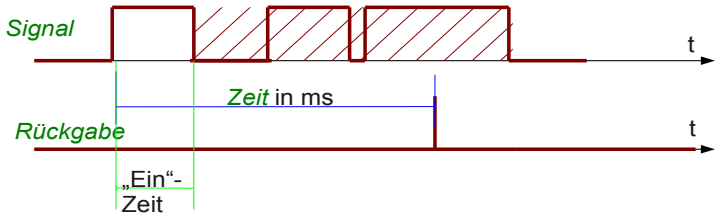
- Funktion after(*Signal*, *Zeit*)

Argumente

- Argument *Signal* vom Datentyp b01
- Argument *Zeit* vom Datentyp u64

Wirkung

- Die Funktion startet beim Übergang der Variablen *Signal* von AUS auf EIN einen Timer. Nach Ablauf der *Zeit* in ms erzeugt die After-Funktion einen EIN-Impuls.



- Während des Zeitintervalls Totzeit ist die Funktion gesperrt, d.h. neu ankommende Impulse werden ignoriert

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Hinweis:

- Wenn der Enertex® EibPC mit after (mit Hilfe einer if-Anweisung und write) ein Telegramm auf den Bus schreibt, kann je nach Buslast und Busgeschwindigkeit eine zusätzliche (variable) Verzögerungszeit von wenigen ms entstehen, bis dieses dort erscheint.

Beispiel: Ein- und Auschaltverzögerung

Die Variable LichtAktor (Datetyp b01) soll nach 1000 ms der Variablen LichtTaster (Datentyp b01) folgen.

Umsetzung im Anwenderprogramm:

```
LichtAktor =AUS
if after(LichtTaster,1000u64) then LichtAktor =EIN endif
```

Beispiel: Einschaltverzögerung

Wenn LichtTaster (Typ b01) EIN ist, soll nach 1300 ms die Variable LichtAktor (Typ b01) auf EIN gehen.

Umsetzung im Anwenderprogramm:

```
if (after(LichtTaster,1300u64)==1b01) then LichtAktor=1b01 endif
```

Alternative 1

```
if after(LichtTaster==1b01,1300u64) then LichtAktor=1b01 endif
```

Alternative 2

```
if after(LichtTaster,1300u64) then LichtAktor=1b01 endif
```

Man beachte, dass “LichtAktor“ nur gesetzt, nicht aber gelöscht wird. Siehe dazu auch das folgende Beispiel.

Beispiel: Ausschaltverzögerung

Wenn LichtTaster (Datentyp b01) *AUS* ist, soll nach 4000 ms die Variable LichtAktor auf *AUS* gehen.

Die Umsetzung in der EibParserdatei lautet dann:

```
if (after(LichtTaster==AUS,4000u64)) then LichtAktor=0b01 endif
```

Beispiel: Verschiedene Ein- und Ausschaltverzögerung

Wenn LichtTaster (Datentyp b01) *EIN* ist soll nach 1300 ms die Variable LichtAktor (Datentyp b01) *EIN* ist gehen, wenn LichtTaster (Datentyp b01) *AUS* ist, soll nach 4000 ms die Variable LichtAktor auf *AUS* gehen.

Umsetzung im Anwenderprogramm:

```
if (after(LichtTaster==EIN,1300u64)) then LichtAktor=EIN endif
if (after(LichtTaster==AUS,4000u64)) then LichtAktor=AUS endif
```

Beispiel: Doppelte Belegung einer Taste:

Siehe Seite 66.

Afterc

Definition

- Funktion afterc(*Signal*, *Zeit, xT*)

Argumente

- Argument *Signal* vom Datentyp b01
- Argument *Zeit* vom Datentyp u64
- Argument *xT* in ms vom Datentyp u64

Wirkung

- Wie after S. 199.
- Über die Variable *xT* kann die verbleibende Zeit ausgefragt werden.
  ACHTUNG: Wenn Sie dieselbe Variable *xT* für verschiedene Funktionen afterc im Programmcode nutzen, führt dies zu einem undefiniertem Verhalten.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Hinweis:

- Wenn der Enertex® EibPC mit afterc (mit Hilfe einer if-Anweisung und write) ein Telegramm auf den Bus schreibt, kann je nach Buslast und Busgeschwindigkeit eine zusätzliche (variable) Verzögerungszeit von wenigen ms entstehen, bis dieses dort erscheint.

Beispiel: Verbleibende Zeit anzeigen – Zeit Messen

Die Variable LichtAktor (Datetyp b01) soll nach 1000 ms der Variablen LichtTaster (Datentyp b01) folgen. Die noch verbleibende Zeit - gemessen ab LichtTaster == EIN bis Ende der Verzögerung 4000 ms - soll alle 300ms auf die Adresse '2/2/2' geschrieben werden.

Umsetzung im Anwenderprogramm:

```
LichtAktor =AUS
xT=0u64
if afterc(LichtTaster,1000u64,xT) then LichtAktor =EIN endif
if (change(xt/300u64)) then write('2/2/2'u64, xC) endif
```

Zyklustimer - cycle

Definition

- Funktion cycle(*mm,ss*) mit:
  mm: Minuten (0...255)
  ss: Sekunden (0..59)

Argumente

- Zwei Argumente *mm,ss* vom Datentyp u08

Wirkung

- Der Rückgabewert geht für einen Verarbeitungzyklus wiederkehrend auf 1b01, sonst ist er 0b01. Die Wiederholungszeit ist in mm:ss (Minuten:Sekunden) definiert.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: Cycle

Immer nach 1 Minute und 5 Sekunden soll eine Leseanforderung an die Adresse “Licht1-0/0/1“ gesendet werden.

Umsetzung im Anwenderprogramm:

```
if cycle(01,05) then read("Licht1-0/0/1") endif
```

Remanentspeicher

Sie können den Flash-Speicher des Enertex® EibPCs nutzen, um Variablen im Speicher abzulegen. Hierzu sind 1000 Speicherplätze vorgesehen, welche Variablen mit beliebigen Datentypen aufnehmen können. Dieser Speicher wird weder beim Firmwareupdate, noch bei der Ausführung eines Resets, noch beim Einspielen eines Patches, noch bei der Änderung des Programms verändert.

Beim Ablegen der Daten einer Variablen in einem Speicherplatz werden nur binäre Daten gespeichert. Wenn Daten aus dem Flash in eine Variable zurückgeschrieben werden, muss darauf geachtet werden, dass die Datentypen der zuvor abgelegten Daten gleich sind. Jeder Speicherplatz kann bis zu 1400 Bytes an Daten aufnehmen. Die Anzahl der im Speicherplatz „genutzten" Daten hängt dabei vom Datentyp der Variablen bzw. dessen Bitlänge ab (vgl. Seite 161).

Beachten Sie folgenden HINWEIS:

Das Schreiben auf den Flash kann bei fehlerhaften Programmen, Netzunterbrechungen etc. zu Dateisystemfehlern führen. Außerdem verkürzt das Schreiben auf den Flash dessen Lebensdauer.

Um Dateisystemfehler zu vermeiden, hat Enertex® Bayern GmbH ein Patch 1.100 bei neu ausgelieferten Enertex® EibPC eingeführt, welches eine Art Checkdisk auf das Flash beim Neustart automatisch ausführt. Dieses Patch kann nur im Werk aufgespielt werden, da hierzu eine Neuinitalisierung des integrierten Flashes notwendig wird. Kontaktieren Sie Enertex® Bayern GmbH für weitere Details

Ohne Patch 1.100 übernehmen wir keine Garantie auf Dateisystemfehler, wenn ein Speichern von Variablen auf das Flash erfolgte, was mit Hilfe eines Logs im Werk nachvollzogen werden kann.

Readflash

Definition

- Funktion readflash(*Variable*, *Speicherplatz)*

Argumente

- *Variable* beliebiger Datentyp
- *Speicherplatz* vom Datentyp u16. Gültige Werte sind 0u16 bis 999u16

Wirkung

- Es werden vom *Speicherplatz* (Nummer 0u16 bis 999u16) im eingebauten Flash soviele Binärdaten auf den Speicher der *Variable* zurückgeschrieben, wie diese aufnehmen kann (vgl. Bitlänge Seite 161). Der Rückgabewert ist 0b01 wenn, das Lesen erfolgreich war, sonst wird 1b01 zurückgegeben.
- **Es wird empfohlen, bei neuen Programmen die Funktionen readflashvar und writeflashvar zu nutzen (vgl. S. 205).**

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

  (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Writeflash

Definition

- Funktion writeflash(*Variable*, *Speicherplatz)*

Argumente

- *Variable* beliebiger Datentyp
- *Speicherplatz* vom Datentyp u16. Gültige Werte sind 0u16 bis 999u16

Wirkung

- Es werden auf den *Speicherplatz* (Nummer 0u16 bis 999u16) im eingebauten Flash die Binärdaten des Speicherinhalts (vgl. Bitlänge Seite 161) der *Variable* abgelegt. Der Rückgabewert ist 0b01 wenn, das Schreiben erfolgreich war, sonst wird 1b01 zurückgegeben.
- **Es wird empfohlen, bei neuen Programmen die Funktionen readflashvar und writeflashvar zu nutzen (vgl. S. 205).**

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

  (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Beispiel:

Beim Systemstart sollen zehn 1400-Bytes Strings (c1400) in das Flash auf die ersten 10 Speicherplätze geschrieben werden und anschließend wieder ausgelesen werden. Sollte Schreiben oder Auslesen Probleme bereiten, so soll auf eine Gruppenadresse '8/5/2'c14 eine Textmeldung ausgeben werden.
Das Ergebnis des Auslesens soll ebenso auf diese Gruppenadresse geschrieben werden.

```
[EibPC]
a=$: Zahl$
nr=0u16
read_nok=OFF
write_nok=OFF
new_r=ON
new_w=ON
TestGA='8/5/2'c14

if cycle(0,1) and nr<10u16 then write_nok=writeflash(convert(nr,$$)+a,nr);
nr=nr+1u16;new_w=!new_w endif
if cycle(0,1) and nr>9u16 then {
        read_nok=readflash(a,nr-10u16);
        nr=nr+1u16;
        if (nr<20u16) then new_r=!new_r endif
} endif

if write_nok then write('8/5/2'c14,$W-Err: $c14+convert(nr,$$c14)) endif
if change(new_w) then write('8/5/2'c14,convert(convert(nr,$$)+a,$$c14)) endif

if read_nok then write('8/5/2'c14,$R-Err: $c14+convert(nr-10u16,$$c14)) endif
if change(new_r) then write('8/5/2'c14,convert(a,$$c14)) endif
```

Beispiel 2:

Der letzte Wert, der auf den Bus gesendet wird, soll im Flash gespeichert werden und nach einem Neustart automatisch auf den Bus gesendet werden.

```
Value=0u08
if change("Wohnküche RTR Modus-5/1/7") then {
        writeflash("Wohnküche RTR Modus-5/1/7",0u16)
} endif
if systemstart() then readflash(Value, 0u16) endif
if after(systemstart(),1000u64) then write("Wohnküche RTR Modus-5/1/7",Value) endif
```

Readflashvar

Definition

- Funktion readflashvar(*Variable)*

Argumente

- *Variable* beliebiger Datentyp

Wirkung

- Im eingebauten Flash werden die Binärdaten auf den Speicher der *Variable* zurückgeschrieben, wie diese aufnehmen kann (vgl. Bitlänge Seite 161). Der Rückgabewert ist 0b01 wenn, das Lesen erfolgreich war, sonst wird 1b01 zurückgegeben.
- Das Lesen bzw. die De-Referenzierung erfolgt über den Variablennamen. Wenn der Anwender ein neues Programm einspielt, so wird unabhängig von den Programmänderungen die Variable mit zuletzt im Flash gespeicherten Wert überschrieben.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

  (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Writeflashvar

Definition

- Funktion writeflashvar(*Variable)*

Argumente

- *Variable* beliebiger Datentyp

Wirkung

- Es werden im eingebauten Flash die Binärdaten des Speicherinhalts (vgl. Bitlänge Seite 161) der *Variable* abgelegt. Der Rückgabewert ist 0b01 wenn, das Schreiben erfolgreich war, sonst wird 1b01 zurückgegeben.
- Das Schreiben bzw. die Referenzierung erfolgt ausschließlich über den Variablennamen..

Datentyp Ergebnis (Rückgabe)

- Datentyp b01
  (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Beispiel :

Der letzte Wert einer Variable soll um Mitternacht oder vor dem Einspielen einer neuen Benutzerprogrammierung im Flash gespeichert werden und nach einem Neustart automatisch in die Variable geladen werden.
Hinweis: Die vordefinierte Variable SHUTDOWN wird vom Enertex® EibStudio vor Einspielen einer neuen Anwenderprogrammierung automatisch auf EIN gesetzt, sodass die Anwendung genügend Zeit bekommt, z.B. Werte auf das Flash zu speichern (vgl.S 221)

```
ValuePowerK1="K1-Wirkenergiezähler (Verbrauch)-14/2/76"
if htime(0,0,0) or SHUTDOWN then {
        writeflashvar(ValuePowerK1)
} endif
if systemstart() then readflashvar(ValuePowerK1) endif
```

Datensicherung

Neben den Werten im Remanentspeicher, die über die Funktionen readflash writeflash, readflashvar und writeflashvar angesprochen werden können, sind zudem

- Szenen
- Zeitreihen von Diagrammen („Timebuffer“)
- Bilder und Daten für den internen Webserver.
- und (je nach Anwenderwahl) das gesamte Projekt

auf dem Flash des Enertex® EibPC abgelegt.

Diese Daten können über das Enertex® EibStudio vom Enertex® EibPC geladen und separat gesichert werden.

Projektdaten

Wenn beim Übertragen eines Programms die Option „Projektdaten als Klartext auf den EibPC hochladen“ angekreuzt wird, so werden sämtliche Daten, einschließlich verwendeter Makrodaten, ESF etc. vom EibStudio auf ein separates Verzeichnis im Flash abgelegt.

Zu einem späteren Zeitpunkt können diese Daten unter dem Menü Datei/Projektdaten als Klartext auf den EibPC herunterladen wiederhergestellt werden.

Bilder, Szenen, Zeitreihen

Im Menü EibPC/Dateitransfer Dialog befindet sich ein Up- und Downloadmanager.

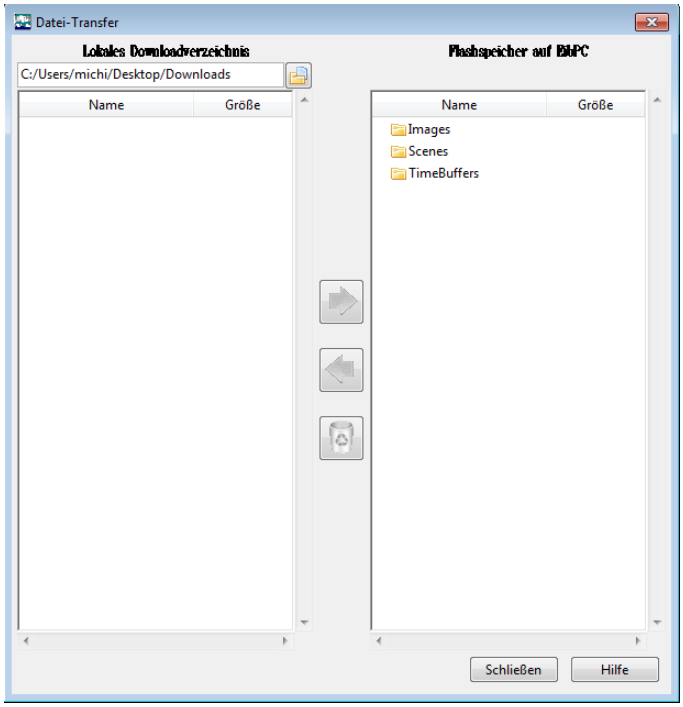


Abbildung 119: Up- und Downloads von Dateien auf den internen Flash

Damit können Sie z.B. Bilder auf den Enertex® EibPC laden, die Sie im Webserver nutzen können. Daneben kann ein Backup der auf dem Enertex® EibPC gespeicherten Zeitreihen auf einen lokalen PC sichern.

Der Up/Downloadmanager übernimmt je nach Dateiendung die richtige Zuordnung für die Verarbeitung im Enertex® EibPC. Bilddateien müssen die Endung „jpg“ oder „png“ tragen, Szenen dem internen Schema des Enertex® EibPCs folgen, Zeitreihen werden anhand des Dateinamen erkannt und zugeordnet.

D.h. je nach Dateiname erkennt Enertex® EibStudio das Zielverzeichnis automatisch.

Wenn auf die Grafiken von den Weblementen zugegriffen werden sollen, ist der Pfad der Grafik dann `/upload/` + Dateiname, vgl. S. 116.

Arithmetische Verknüpfungen („Berechnungen“)

Grundsätzliches

Mit dem Enertex® EibPC können nicht nur (logische und zeitliche) Abläufe programmiert, sondern auch mathematische Ausdrücke ausgewertet und daraus entsprechende Reaktionen auf die KNX–Vernetzung, z.B. durch Senden von entsprechenden Adressen, hervorgerufen werden.

Für alle Argumente von Funktionen können anstelle von Variablen auch direkt Gruppenadresse verwendet werden.

Abs- Absolutwert

Definition

- Funktion abs(*Variable*)

Argumente

- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge

Wirkung

- Rückgabewert: Absolutwert der *Variable*

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argument

Beispiel Absolutwert:

Man berechne von a (=2.5f23) den Absolutwert und speichere diesen in b.

Die Umsetzung in der EibParserdatei lautet dann:

```
a=-2.5f32
b=abs(a)
```

Addition

Definition

- *Variable1* + *Variable2 [...]*

Argumente

- alle Argumente vom gleichen Datentyp
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge

Wirkung

- Die Werte der Variablen werden addiert. Es können nur Werte vom gleichen Datentyp addiert werden. Wollen Sie dennoch beispielsweise einen vorzeichenlosen 8-Bit Wert mit einem vorzeichenbehafteten 16-Bit Wert addieren, benutzen Sie die convert-Funktion (siehe Seite 217)

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Hinweis:

Mit gleicher Syntax können Sie Zeichenketten verketten (siehe Seite 229).

Acos - Arkuscosinus

Definition

- Funktion acos(*Variable*)

Argumente

- Ein Argument *Variable* von Datentyp f32

Wirkung

- Berechnung des Arkuscosinus der *Variable,* Ergebnis angegeben in RAD
- Wenn das Argument größer 1f32 oder kleiner -1.0f32 ist erfolgt keine Berechnung

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel Arkuscosinus:

In Variable b steht das Ergebnis des Arkuscosinus von Variable a.

Die Umsetzung in der EibParserdatei lautet dann:

```
a=5f32
b=acos(a)
```

Asin - Arkussinus

Definition

- Funktion asin(*Variable*)

Argumente

- Ein Argument *Variable* von Datentyp f32

Wirkung

- Berechnung des Arkussinus der *Variable,* Ergebnis angegeben in RAD
- Wenn das Argument größer 1f32 oder kleiner -1.0f32 ist erfolgt keine Berechnung

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel Arkussinus:

```
In Variable b steht das Ergebnis des Arkussinus von Variable a.
```

Umsetzung im Anwenderprogramm

```
a=5f32
b=asin(a)
```

Atan - Arkustangens

Definition

- Funktion atan(*Variable1*)

Argumente

- Ein Argument *Variable* von Datentyp f32

Wirkung

- Berechnung des Arkustangens der *Variable,* Ergebnis angegeben in RAD

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel Arkustangens:

```
In Variable b steht das Ergebnis des Argustangens von Variable a.
```

Umsetzung im Anwenderprogramm:

```
a=5f32
b=atan(a)
```

Cos - Cosinus

Definition

- Funktion cos(*Variable*)

Argumente

- Ein Argument *Variable* von Datentyp f32

Wirkung

- Berechnung des cosinus der *Variable,* Ergebnis angegeben in RAD

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel Cosinus:

```
In Variable b steht der Cosinus von Variable a.
```

Umsetzung im Anwenderprogramm

```
a=5f32
b=cos(a)
```

Ceil - kleinste ganze Zahl

Definition

- Funktion ceil(*Variable*)

Argumente

- Ein Argument *Variable* von Datentyp f32 oder f16

Wirkung

- Kleinste Ganzzahl ≥ *Variable* berechnen

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Division

Definition

- *Variable1 / Variable2 [...]*

Argumente

- alle Argumente vom gleichen Datentyp.
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge

Wirkung

- Variable1 wird durch Variable2 dividiert.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel

Der Heizungsvorlauf soll abhängig von der Außentemparatur eingestellt werden. Ist die Außentemperatur unter 0°C geht der Vorlauf auf 55°C. Bei 30°C Außentemparatur ist der Vorlauf auf 30°C eingestellt.
Außentemperatur = 15°C
V = 30 + 25/30 * (30 - Außentemperatur)

Umsetzung im Anwenderprogramm

```
V = 30f16 + 25f16 / 30f16 * (30f16 – "Außentemperatur-3/5/0"f16)
```

Average - Durchschnitt

Definition

- Funktion average(*Variable1*, *Variable2, [...])*

Argumente

- alle Argumente vom gleichen Datentyp.
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge

Wirkung

- Rückgabewert: Der Durchschnittswert der angegebenen Variablen, die alle vom selben Datentyp sein müssen. (Anstelle von Variablen, können auch manuelle oder ETS Gruppe-nadressen angegeben werden.) Die Genauigkeit der Berechnung ergibt sich durch den verwendeten Datentypen.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Durchschnitt berechnen

Es soll der durchschnittliche Wert der Heizaktoren ermittelt werden.

Umsetzung im Anwenderprogramm:

```
c=average("HeizungKeller1-1/0/2","HeizungKeller2-1/0/3","HeizungKeller3-1/0/4" /
          "HeizungKeller4-1/0/5","HeizungKeller5-1/0/6")
```

Exp - Exponentialfunktion

Definition

- Funktion exp(*Variable*)

Argumente

- Ein Argument *Variable* von Datentyp f32

Wirkung

- Berechnung des Exponentialfunktion der *Variable*

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel E-Funktion:

```
In Variable b steht das Ergebnis der E-Funktion von Variable a.
```

Die Umsetzung in der EibParserdatei lautet dann:

```
a=5f32
b=exp(a)
```

Floor - Größte ganze Zahl

Definition

- Funktion floor(*Variable)*

Argumente

- Ein Argument *Variable* von Datentyp f32 oder f16

Wirkung

- Rückgabwert: Größte Ganzzahl ≤ *Variable* ausgeben.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Log- Logarithmus

Definition

- Funktion log(*Variable1, Variable2*)

Argumente

- Zwei Argumente vom Datentyp f32
- *Variable1*: Numerus
- *Variable2*: Basisl

Wirkung

- Rückgabwert: Das Ergebnis der Logarithmusrechnung.
- Falls ein nicht-positives Argument oder eine nicht-positive Basis vorgegeben wird, erfolgt keine Berechnung.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Max - Maximum

Eine Maximum-Funktion ist wie folgt definiert:

Definition

- Funktion max(*Variable1*, *Variable2, [...]*)

Argumente

- alle Argumente vom gleichen Datentyp.
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge.

Wirkung

- Rückgabewert: Das Maximum der angegebenen Variablen, die alle vom selben Datentyp sein müssen.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Maximum von 5 Prozentwerten

Es soll der maximale Wert der Heizaktoren ermittelt werden.

Umsetzung im Anwenderprogramm:

```
c=max("HeizungKeller1-1/0/2","HeizungKeller2-1/0/3","HeizungKeller3-1/0/4" /
        "HeizungKeller4-1/0/5","HeizungKeller5-1/0/6")
```

Min - Minimum

Das Minimum von beliebig vielen Variablen wird wie folgt berechnet:

Definition

- Funktion min(*Variable1*, *Variable2, [...]*)

Argumente

- alle Argumente vom gleichen Datentyp.
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge

Wirkung

- Rückgabewert: Das Minimum der angegebenen Argumente, die alle vom selben Datentyp sein müssen.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Minimum von 5 Prozentwerten

Es soll der minimale Wert der Heizaktoren ermittelt werden.

Umsetzung im Anwenderprogramm:

```
c=min("HeizungKeller1-1/0/2","HeizungKeller2-1/0/3","HeizungKeller3-1/0/4" /
        "HeizungKeller4-1/0/5","HeizungKeller5-1/0/6")
```

Mod - Modulo

Definition

- Funktion mod*(Variable1, Variable2)*

Argumente

- alle Argumente vom gleichen Datentyp.
- Datentypen: uXX und sXX mit XX beliebige auf Seite 161 definierte Bitlänge

Wirkung

- Funktion führt Modulo-Berechnung Rückgaberwert = *Variable1* modulo *Variable2*

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Mutliplikation

Definition

- *Variable1 * Variable2 [...]*

Argumente

- Alle Argumente vom gleichen Datentyp.
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge

Wirkung

- Die Werte der Variablen werden multipliziert.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Pow - Potenz

Definition

- Funktion pow(*Variable1, Variable2*)

Argumente

- Zwei Argumente vom Datentyp f32
- *Variable1*: Basiszahl
- *Variable2*: Exponent

Wirkung

- Rückgabewert: Das Ergebnis der Potenzrechnung.
- Beide Variablen müssen als Fließkomma 32Bit Wert angegeben werden. Siehe hierzu Tabelle 1 und die dort gemachten Erläuterungen.
- Aktualisierung der Abhängigkeiten bei Wertänderung. Falls eine negative Basis vorgegeben wird, erfolgt keine Berechnung.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel: *Siehe Taupunkttemperaturberechnung, Seite 62*

Sqrt - Quadratwurzel

Definition

- Funktion sqrt(*Variable*)

Argumente

- Ein Argument vom Datentyp f32

Wirkung

- Quadratwurzel der *Variable* berechnen. Die Variable muss als Fließkomma 32Bit Wert angegeben werden. Siehe hierzu Tabelle 1 und die dort gemachten Erläuterungen.
- Falls *Variable* kleiner 0 ist, erfolgt keine Berechnung.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel Quadratwurzel:

```
In Variable b steht das Ergebnis der Quadratwurzel-Funktion von Variable a.
```

Die Umsetzung in der EibParserdatei lautet dann:

```
a=5f32
b=sqrt(a)
```

Sin - Sinus

Definition

- Funktion sin(*Variable*)

Argumente

- Ein Argument vom Datentyp f32

Wirkung

- Rückgabewert: Sinus der *Variable*, Ergebnis angegeben in RAD.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel Sinus:

```
In Variable b steht der Sinus von Variable a.
```

Die Umsetzung in der EibParserdatei lautet dann:

```
a=4f32
b=sin(a)
```

Subtraktion

Definition

- *Variable1 - Variable2 [...]*

Argumente

- Alle Argumente vom gleichen Datentyp.
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge.

Wirkung

- Die Werte der Variablen werden subtrahiert.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Tan - Tangens

Definition

- Funktion tan(*Variable*)

Argumente

- Ein Argument vom Datentyp f32

Wirkung

- Tangens der *Variable*, Ergebnis angegeben in RAD.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Beispiel Tangens:

```
In Variable b steht das Ergebnis des Tangens von Variable a.
```

Die Umsetzung in der EibParserdatei lautet dann:

```
a=5f32
b=tan(a)
```

Sonderfunktionen

Change

Definition

- Funktion change(*Variable*)

Argumente

- Ein Argument von beliebigem Datentyp

Wirkung

- Rückgabewert: EIN bei Änderung der Überwachten Adresse bzw. Variable im vorhergehenden Verarbeitungszyklus des Enertex® EibPC.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Eine Besonderheit der change-Funktionen ist, dass diese nicht bei if-Anweisungen mit else-Zweig stehen dürfen. Ähnlich zur event-Funktion (siehe Seite 173) geht die change-Funktion immer nur für einen Verarbeitungszyklus auf EIN und wird dann bei der if-Anweisung den then-Zweig ausführen. Im nächsten Zyklus geht change wieder auf AUS und nun würde der else-Zweig ausgeführt. Um den Anwender die Programmierung zu vereinfachen, wurde die Verwendung der change-Funktion an dieser Stelle durch den Compiler beschränkt.

*Eine Änderung des Arguments aktiviert change in der **darauffolgenden** Verarbeitungsschleife.*

Beispiel: Change

Wenn sich die maximale Heizleistung ändert, dann soll der Heizungsvorlauf nach geregelt werden.

Umsetzung im Anwenderprogramm

```
if change(HeizungMax) then write("Heizvorlauf-0/0/1",HeizungBedarf) endif
```

Convert - Konvertieren von Datentypen

Definition

- Funktion convert(*Variable1*, *Variable2*)

Argumente

- Zwei Argumente von beliebigem Datentyp

Wirkung

- Konvertiert den Datentyp von *Variable1* in den von *Variable2*.
- Es sind alle Umwandlungen ineinander erlaubt mit der Ausnahme von b01 bzw. nach f16 oder f32.
- Wird vom Datentyp f16 in c14 bzw. c1400 umgewandelt, so wird für die Darstellung der Zeichenkette eine Festkommadarstellung mit zwei Nachkommastellen gewählt.
- Wird vom Datentyp f32 in c14 bzw. c1400 umgewandelt, so wird für die Darstellung der Zeichenkette Exponentialdarstellung gewählt wird.
- Der Wert von *Variable2* wird immer ignoriert. Dieses Argument dient nur zu Angabe des Datentyps.

Datentyp Ergebnis (Rückgabe)

- Das Ergebnis der Konvertierung der *Variable1* in den Datentyp von *Variable2*.

Hinweis:

Durch die Umwandlung von Datentypen können Informationen z.B. durch „Abschneiden“ von Bits verloren gehen.

Beispiel: Convert-Funktion

Ein vorzeichenloser 8-Bit Wert soll mit einem vorzeichenbehafteten 16-Bit Wert addiert werden.

Die Umsetzung in der EibParserdatei lautet dann:

```
Var1=10u08
Var2=300s16
Var3=convert(Var1,Var2)+Var2
```

Devicenr

Definition

- Funktion devicenr()

Argumente

- keine

Wirkung

- Seriennummer des EibPC abfragen

Datentyp Ergebnis (Rückgabe)

- Datentyp u32

Beispiel: Devicenr

Die Seriennummer soll der Variable SNR zugewiesen werden.

Die Umsetzung in der EibParserdatei lautet dann:

```
SNR= devicenr()
```

Elog

Definition

- Funktion elog()

Argumente

- keine

Wirkung

- Auslesen des ältesten Eventspeicher Eintrags.
- Nach dem Auslesen des Logeintrags wird dieser gelöscht.
- Die Funktion gibt im Klartext die aufgetretenen Ereignisse zurück. Dabei wird folgendes Format in den String geschrieben: Objekttext.:EventID@Zeitstempel. Der Objekttext stellt den Programmcode dar, welcher das Ereignisse auslöst. Der Programmcode ist auf 32 Zeichen begrenzt.

Datentyp Ergebnis (Rückgabe)

- Datentyp c1400 String

Beispiel: siehe elognum S. 218

Elognum

Definition

- Funktion elognum()

Argumente

- keine

Wirkung

- Gibt die Anzahl der Einträge im Fehlerspeicher zurück.

Datentyp Ergebnis (Rückgabe)

- Datentyp u16

Beispiel: elog

Die letzte EventNummer auslesen und den Speicher um eins zurücksetzen.

Die Umsetzung in der EibParserdatei lautet dann:

```
EventInfo=$$
EventNr=elognum()
if change(EventNr) then EventInfo=elog() endif
```

Eval

Definition

- Funktion eval(*Arg*)

Argumente

- Ein Argument, Datentyp beliebig

Wirkung

- Unabhängig vom Validierungsschema wird die Auswertung des Ausdrucks vorgenommen. Dies ist insbesondere bei der if-Abfrage von Bedeutung, wenn Verschachtelungen in der üblichen Syntax von C-Programmen realisiert werden sollen.

Datentyp Ergebnis (Rückgabe)

- gleicher Datentyp wie Argument

Beispiel: Zähler

Sie möchten einen Zähler programmieren, der mit dem Verarbeitungszyklus des EibPCs bis 100 hochzählt.

Umsetzung:

```
Counter=0
if eval(Counter<100) then Counter=Counter+1 endif
```

Hinweise:

Die Programmierung mit Hilfe des Validierungsschemas ist ein Garant für eine stabile und optimierte, ereignisorientierte Verarbeitung der Telegramme: Nur bei Änderung wird ein Ausdruck/Variable/Funktion ungültig, so dass der Enertex® EibPC **nur** die davon abhängigen Ausdrücke neu verarbeitet. Die eval-Funktion unterbricht das Valdierungsschema bei der Verarbeitung und erzeugt somit zum Einen eine höhere Systemlast bei der Verarbeitung.

Wenn Sie anstelle von

```
if '1/0/0'b01 then write('1/0/1'b01,AUS) endif
```

versehentlich mit *if eval('1/0/0'b01)* arbeiten würden, können Sie ihre KNX™ Installation zum Absturz bringen. Wir empfehlen die Verwendung der eval-Funktion nur erfahrenen Programmieren, da das Validierungsschema optimal auf die Verwendung im Enertex® EibPC und desses Programmierung abgestimmt ist.

Eine Anweisung

```
if Counter<100 then Counter=Counter+1 endif
```

würde im Normalfall beim Systemstart oder Neusetzen der Variablen *Counter* von z.B. 102 auf 10 nur ein einziges mal ausgeführt, da ja dann *Counter*<100 gültig ist und eine weitere Auswertung nicht mehr vorgesehen ist.

Wir empfehlen, bei Verschachtelungen anstelle der Eval-Funktion soweit möglich mit and zu arbeiten.

Processingtime

Definition

- Funktion processingtime()

Argumente

- keines.

Wirkung

- Der EibPC benötigt für die Verarbeitung seines Programms pro Zyklus eine gewisse Zeit. Diese Verarbeitungszeit wird mit dieser Funktion in ms zurückgegeben.

Datentyp Ergebnis (Rückgabe)

- Verarbeitungszeit in ms als Datentyp u16.

Beispiel:

Es soll die max. Dauer der Verarbeitung pro Sekunde in einem Diagramm visualisiert werden. Die Maximalwert über alle Zyklen soll zudem angegeben werden.

```
[WebServer]
page(1) [$Test$,$Processingtime$]
mtimechart(1)[EXTLONG,AUTOSCALE,256,0,10,0,1]($Time in ms $,LEFT, Buffer0)
[EibPC]
Buffer0=0
timebufferconfig(Buffer0, 0, 3600u16, t)

// per Second
t=0u16
if t < processingtime() then t=processingtime() endif
// Maximum
m=0u16
if m < processingtime() then m=processingtime() endif

// write to chart
if cycle(0,1) then {
        timebufferadd(Buffer0,t);
        t=0u16;
} endif

// Generate some load
y=0f32
if cycle(0,10) then
y=cos(34f32)+sqrt(234f32)+tan(34f32)*7f32+cos(34f32)+sqrt(234f32)+tan(34f32)*7f32+cos(34f
32)+sqrt(234f32)+tan(34f32)*7f32+cos(34f32)+sqrt(234f32)+tan(34f32)*7f32+cos(34f32)+sqrt(
234f32)+tan(34f32)*7f32+cos(34f32)+sqrt(234f32)+tan(34f32)*7f32+cos(34f32)+sqrt(234f32)+t
an(34f32)*7f32 endif
```

Systemstart

Definition

- Funktion systemstart()

Argumente

- keine

Wirkung

- Nach dem Einspielen eines neuen Anwenderprogramm oder einem Neustart des Enertex® EibPC wechselt diese Funktion während des ersten Verarbeitungszyklus von AUS auf EIN.

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

Beispiel: systemstart

Beim Systemstart sollen die Variablen LichterAus und RolloRauf einmalig auf 0b01 gesetzt werden.

Umsetzung

```
if systemstart() then LichterAus=AUS; RolloRauf=RUNTER endif
```

Programmende

Ein eigentliches Programmende gibt es beim Enertex® EibPC nicht. Ein Enertex® EibPC Programm wird dadurch beendet, dass entweder die Stromversorgung getrennt wird, oder der Anwender ein neues Programm einspielt. Im letzteren Fall setzt Enertex® EibStudio die eingebaute Variable SHUTDOWN auf EIN, sodass im Anwenderprogramm entsprechende Aktionen davon abhängig ausgeführt werden können. Enertex® EibStudio wartet dann weitere 5 Sekunden, bevor das Anwendungsprogramm gestoppt wird. Laufende Schreibaktionen auf das Flash werden noch ordentlich ausgeführt.

Beispiel siehe S. 205.

Random - Zufallszahl

Definition

- Funktion random(Max)

Argumente

- Ein Argument Max vom Datentyp u32

Wirkung

- Gibt eine Zufallszahl im Bereich von 0 bis Max zurück.

Datentyp Ergebnis (Rückgabe)

- Datentyp u32

Beispiell Schaltimpuls mit Zufallszeit

Jeden Abend um 22:00 Uhr plus einer Zufallszeit von 3 Min soll die Variable RolloRunter auf EIN gesetzt werden.

Umsetzung

```
// Zufallszahl von 0 bis 180 (32 Bit vorzeichnlos)
Zufall=convert(random(180u32),0u08)
// Umwanden in Minuten und Sekunden
Min=Zufall/60
Sek=Zufall-Min*60
if htime(22, Min, Sek) then RolloRunter=AUS endif
```

Comobject - Kommunikationsobjekt

Definition

- Funktion comobject(*Variable1*, *Variable2, [...])*

Argumente

- alle Argumente vom gleichen Datentyp.
- Datentypen: uXX, sXX und fXX, mit XX beliebige auf Seite 161 definierte Bitlänge.

Wirkung

- Rückgabewert: Der Wert der Variablen, der sich als letzter verändert hat.

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Ein Aktor mit mehreren Variablen – Status ermittlen

Sie wollen den Status eines Aktors (1 Bit) ermitteln. Der Aktor wird über drei Gruppenadressen "GA_a-1/2/3","GA_b-1/2/4" und "GA_c-1/2/5" angesprochen.
Wenn der Aktor 3 Minuten eingeschaltet war, und nicht bereits manuell wieder ausgeschaltet wurde, soll er auf AUS geschaltet werden.

Umsetzung

```
StatusAktor=comobject("GA_a-1/2/3","GA_b-1/2/4","GA_c-1/2/5")
if delay(StatusAktor==EIN,180000u64) and StatusAktor==EIN then write("GA_a-1/2/3", AUS)
endif
```

Sleep - Passivmodus

Definition

- Funktion sleep(Status)

Argumente

- Ein Argument Status vom Datentyp b01.

Wirkung

- Falls der Eingang auf AUS geht, übergibt der EibPC ausgehende EIB-Telegramme und UDP-Pakete an die entsprechende Sendewarteschlange. Liegt am Eingang der Wert EIN an, so werden ausgehende EIB-Telegramme und UDP-Pakete verworfen, d. h. nicht mehr an die Sendewarteschlangen übergeben. Bereits in den Sendewarteschlangen befindliche Daten werden nicht gelöscht und bei Verfügbarkeit der jeweiligen Schnittstelle auf den Bus bzw. das Ethernet geschrieben.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: EibPC in den Passivmodus setzen.

Sie möchten einen Enertex® EibPC in über die Gruppenadresse 2/5/6 (b01) in den Passivmodus setzten

Umsetzung

```
if '2/5/6'b01 then sleep(EIN) else sleep(AUS) endif
```

Hinweis:

Diese Funktion ist hilfreich, wenn in einem bestehenden System ein Programm getestet werden soll ohne tatsächlich auf den Bus zu schreiben. Ohne die Anwender oder etwa ein Programm eines anderen Enertex® EibPC zu stören, können Sie neue Progamme austesten (der Webserver lässt sich dabei normal bedienen). Wenn der Enertex® EibPC im Passivmodus läuft, arbeitet er intern normal weiter, d.h. es werden Variablen berechnet, Zustände geändert und der Webserver angepasst etc.

Eibtelegramm

Mit dieser Funktion können KNX™-Telegramme auf der untersten Ebene erstellen werden. In diesem Fall können Geräte z. B. auch über ihre physikalische Adresse angesprochen werden, wie es bei der Programmierung von Anwendungsdaten der Fall ist. Der EibPC arbeitet intern im Gruppentelegrammmodus und daher auch nur Gruppentelegramme mitloggt und aufzeichnet, die an eine Gruppenadresse gesendet werden.

Sollen auch andere Telegramme (z.B. an eine physikalische Adresse gesendete) beobachtet werden, so muss die ETS im Busmonitor und mit einer Busmonitor fähigen Schnittstelle arbeiten. Geeignete Schnittstellen für den ETS Busmonitor zur Beobachtung von vom Enertex® EibPC gesendeten Telegrammen mit physikalischer Adressierung sind:

- EIBMarkt IF-RS232 (Sie müssen umschalten auf FT 1.2 und Busmonitormodus)
- EIB-IP-Router N146 (Sie müssen in der ETS mit dem Busmonitor arbeiten und die Schnittstelle im Routing-Modus betreiben)
- EIB KNX IP Schnittstelle PoE (Sie müssen in der ETS mit dem Busmonitor arbeiten)

Definition

- Funktion eibtelegramm(Kontrollfeld, Zieladresse, Telegramminformation, Nutzinformation1 .. Nutzinformation18)

Argumente

- Kontrollfeld vom Datentyp u08

  Das Kontrollfeld ist das erste Zeichen eines jeden Telegramms.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	1	0	W	1	P1	P0	0	0
	1	0	1	1	1	1	0	0
	1*128 + 0*64 + 1*32 + 1*16 + 1*8 + 1*4 + 0*2 + 0*1							
u08 Datentyp	188							

Abbildung 120: Kontrollfeld eines KNX™-Telegramms

  Bit W: Wiederholung; hat den Wert 1

  P1 und P0 geben die Prioritätsstufe an. Diese ist im Normalfall niedrig; d.h. P1=P0=1

  Somit ergibt sich für das Kontrollfeld die Struktur: 10111100b = 188u08

- Zieladresse (physikalische Adresse oder Gruppenadresse) vom Datentyp u16

Bit:	16 .. 12	11 .. 8	7 .. 0
Physikalische Adresse	Bereichs-adresse	Linien-adresse	Teilnehmeradresse
z.B.	1	3	5
Binär:	0001	0011	0101
	1*4096 +	1*512+1*256	+ 0*8+1*4+0*2+1*1
u16-Datentyp	4869		

  Abbildung 121: Umwandlung der physikalischen Adresse in einen u16-Datentyp

- Telegramminformation vom Datentyp u08 enthält:

  a) die Kennzeichnung des Typs der Adresse des Empfängers in Bit 7 (MSB)

  Wert = 0 → physikalische Adresse

  Wert = 1 → Gruppenadresse

  b) den Routing-Zähler in den Bits 4..6

  Zählerstand 7: Ein Telegramm mit diesem Zählerstand wird unverändert und beliebig oft über alle Koppler weiter vermittelt

  Zählerstand 6..1: Ein Telegramm mit einem solchen Zählerstand wird weitervermittelt, sein Routing-Zähler wird beim Durchlaufen eines Kopplers jeweils um eins vermindert

  Zählerstand 0: Ein Telegramm mit diesem Zählerstand wird nicht weitervermittelt

  c) die Länge der Nutzdaten in den Bits 0..3, welche aber aus der Anzahl der Eingänge berechnet wird und deshalb nicht angegeben werden muss

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	0	1	1	1	0	0	0	0
	0*128 + 1*64 + 1*32 + 0*16 + 0*8 + 0*4 + 0*2 + 0*1							
u16 Datentyp	112							

Abbildung 122: Telegramminformationen zum Senden mit physikalischer Adresse

- Nutzinformation1... Nutzinformation18 vom Datentyp u08

  Die ersten beiden Bytes des Nutzdatenbefehls enthalten den auszuführenden Befehl und in den meisten Fällen finden auch die zu übertragenden Informationen darin Platz.

  Eine genau Codierung entnehmen Sie bitte entsprechender Fachliteratur.

Wirkung

Die Zustandsspeicher der Eingangsobjekte werden an die entsprechenden Stellen eines allgemeinen KNX™ Telegramm Objekts kopiert. Die individuelle Adresse des Senders kann nicht angegeben werden, sie wird durch die Adresse der Bus Access Unit (= am Enertex® EibPC angeschlossene Schnittstelle) ersetzt.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: physikalische Adresse eines Teilnehmers ansprechen

Alle 10 Minuten soll an den Aktor mit der physikalischen Adresse 1/3/5 eine Leseanforderung gesendet werden.

Umsetzung

```
if cycle(10,0) then eibtelegramm(188u08,4869u16,112u08,0u08) endif
//Äquivalent dazu können die Daten auch als Hex-Werte angegeben werden:
if cycle(10,0) then eibtelegramm(0xbc,0x1105u16,0x70,0x00) endif
```

Lichtszenen

Scene - Szenenaktor

Es können pro Szenenfunktion („Szenenaktor“) bis zu 64 Szenen gespeichert und aufgerufen werden. Die Anzahl der Szenenfunktionen (Szenenaktoren) ist dabei nicht begrenzt – nur durch die Anzahl der maximal möglichen Gruppenadressen in der ets.

Gespeicherte Szenen bleiben auch bei einer vorübergehenden Trennung des Enertex® EibPC von der Stromversorgung oder Änderungen im Anwenderprogramm bestehen. Nur bei Änderung der Gruppenadressen, die für die Szenen von Bedeutung sind, sollten die Szenen über das Menü EibPC → Szenen löschen zurückgesetzt werden.

Definition

- Funktion scene(*GruppenadresseSzenebaustein*, *Akt1, Akt2, ..., AktorN*)

Argumente

- *GruppenadresseSzenebaustein* vom Datentyp u08, sonst weitere beliebige (Datentypen und Anzahl) Gruppenadressen
- *AktorXX* (XX 0 bis max. 65000)
  Im Gegensatz zu herkömmlichen KNX-Szenenaktoren, können hier neben Gruppenadressen auch Variablen genutzt werden (s. Beispiel presetscene S. 226)

Wirkung

- Es wird ein KNX™ Szenenaktor mit der Gruppenadresse *GruppenadresseSzenebaustein* erstellt. Dieser kann entweder mit Hilfe von KNX™ Schaltern und passender ETS Parametrierung oder über unten aufgeführten Funktionen storescene oder callscene angesprochen werden.
- Sie können beliebig viele Szenebausteine definieren.
- Vorgabewerte können mit presetscene (s.u.) vorgegeben werden.

Datentyp Ergebnis (Rückgabe)

- keine

Anmerkung

1. Ein inaktiv Schalten von Eingängen ist mit der Funktion el presetscene S. 226 möglich.
2. Sie können (wie für alle anderen Funktionen auch) beliebig viele „Szenenbausteine“ (also scene-Funktionen) definieren.
3. Jeder so definierte Szenenbaustein verwaltet 64 Szenen (Nummer 0 bis 64)
4. Auf Seite 65 finden Sie ausführlich die Anwendung beschrieben.

Beispiel: Lichtszene

Sie möchten einen Szenenaktor für einen Dimmer und eine Lampe erstellen.

Umsetzung im Anwenderprogramm

```
scene(“SceneBaustein-1/4/3“u08, “Dimmer-1/1/2“, “DimmerWert-1/1/3“, “Lampe-1/1/1“)
```

Presetscene – Vorbelegung für Szenenaktor

Definition

- Funktion presetscene(*GruppenadresseSzenebaustein*, *SzenenNummer, OptionOverwrite, WertVar1,KonfVar1,[WertVar2,KonfVar2,...WertVarN, KonfVarN*)

Argumente

- *GruppenadresseSzenebaustein* und *Nummer* vom Datentyp u08
- *OptionOverwrite* vom Datentyp b01
- *WertVarXX* und *WertVarXX;* XX das XX-te Argument passend zur entsprechenden *Variable bzw. GruppenadresseAktor* die mit Funktion scene definiert wurde (XX 0 bis max. 65000)
- *KonfVarXX* vom Datentyp b01

Wirkung

- Vorbelegung für den Szenenaktor mit der Gruppenadresse *GruppenadresseSzenebaustein* und entsprechender *SzenenNummer* erstellen.
- Ist die *OptionOverwrite* mit 1b01 vorbelegt, wird ein bereits bestehender Datensatz bei Neustart überschrieben. Die Funktion wird dann in jedem Fall die Daten auf dem Flash aktualisieren.

  Durch eine Vorbelegung mit 0b01 wird eine bereits gespeicherte Szene nicht überschrieben. Die Funktion wird keinesfalls die Daten auf dem Flash aktualisieren, falls diese bereits angelegt sind. Ist die Szene mit der angegeben *SzenenNummer* noch nicht angelegt, wird diese dennoch auf dem Flash wie parametriert geschrieben.

- *KonfVarXX* dient zu Vorgabe, ob das entsprechende Eingangsobjekt in dieser Szenennummer aktiv ist. Inaktiv bei 0b01, aktiv bei 1b01. Wenn aktiv, ist *WertVarXX* der entsprechende Wert der Vorbelegung.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: Lichtszene mit presetscene

Sie möchten einen Szenenaktor für einen Dimmer und eine Lampe erstellen. Zusätzlichen sollen die Variablen Var1 und Var2 verändert werden.

Szeneaktor “SceneBaustein-1/4/3“u08, Nummer 13 soll wie folgt vorbelegt werden:
- Bereits gespeicherte Szenen sollen überschrieben werden
- Der Dimmer sollen inaktiv sein (müssen aber angegeben werden, z.B. 0b01 und 49%)
- Die Lampe sowie die beiden Variablen Var1 und Var2 sollen aktiv sein (wobei auf die Gruppenadresse “Lampe-1/1/1“ ein EIN Signal gesendet, die Var1 auf -20 und Var2 auf „scene on“ gesetzt werden sollen)

Umsetzung im Anwenderprogramm

```
Var1=123s32
Var2=$scene off$c14

scene(“SceneBaustein-1/4/3“u08, “Dimmer-1/1/2“, “DimmerWert-1/1/3“, “Lampe-1/1/1“, Var1, Var2)

presetscene(“SceneBaustein-1/4/3“u08, 13, EIN, 0b01, AUS, 49%, AUS, “Lampe-1/1/1“, EIN, -20s32, EIN, $scene on$, EIN)
```

Hinweis:

Die Szenenfunktionen scene und presetscene stehen „toplevel“, also nicht in Abhängigkeit einer if-Anweisung. Für die einfache Verwendung von Szenen steht eine Makrobibliothek EnertexSzene.lib zur Verfügung.

Storescene - Szene speichern

Definition

- Funktion storescene(*GruppenadresseSzenebaustein*, *Nummer*)

Argumente

- Zwei Argumente: *GruppenadresseSzenebaustein* und *Nummer* vom Datentyp u08

Wirkung

- Diese Funktion setzt voraus, dass ein Szenenaktor auf diese Gruppenadresse parametriert wurde (entweder über KNX™ Szenenaktoren oder scene-Funktionen).
- Die Funktion löst ein Telegramm auf *GruppenadresseSzenebaustein* und damit ein Speichern der Szene mit *Nummer* aus.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: storescene

Sie möchten eine definierte Szene des obigen Beispiels auf der Nummer 1 speichern.

Umsetzung im Anwenderprogramm

```
if TasterSceneSpeichern==EIN then storescene(“SceneBaustein-1/4/3“u08,1) endif
```

Callscene - Szene aufrufen

Definition

- Funktion callscene(*GruppenadresseSzenebaustein*, *Nummer*)

Argumente

- Zwei Argumente: *GruppenadresseSzenebaustein* und *Nummer* vom Datentyp u08

Wirkung

- Diese Funktion setzt voraus, dass ein Szenenaktor auf die *GruppenadresseSzenebaustein* parametriert wurde (entweder über KNX™ Szenenaktoren oder scene-Funktionen).
- Die Funktion löst ein Telegramm auf *GruppenadresseSzenebaustein* und damit beim Szenenaktor ein Abrufen der Szene mit *Nummer* aus.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: Callscene

Sie möchten, die in Beispiel zur scene-Funktion definierte Szene, mit der Nummer 1 abrufen.

Umsetzung im Anwenderprogramm

```
if TasterSceneAbrufen==EIN then callscene(“SceneBaustein-1/4/3“u08,1) endif
```

Stringfunktionen

Strings können variabel von 1 bis 65534 Byte definiert werden. Dazu ist hinter der Zeichenkette der entsprechende Endpunkt anzugeben. z.B. ein String mit einer Länge von 55 Byte wird wie folgt definiert: String= $$c55.

Der Datentyp c14 wird vom Compiler separat behandelt, da er mit dem KNX Datentyp EIS15 kompatibel ist und im Gegensatz zu allen anderen Strings keine Nullterminierung am Ende hat, sowie keine Sonderzeichen erlaubt. Vgl. auch die Anmerkungen zum Encoding auf S. 112.

Verketten

Definition

- *String1* + *String2* [+ *String3* ... *StringN*]

Argumente

- Beliebig viele Argumente, aber entweder alle vom Datentyp c14 oder alle vom Datentyp c1400 bzw. einer selbst definierten Stringlänge.

Wirkung

- Die Strings werden aneinander gefügt. Übersteigt die resultierende Länge die maximale Länge des größten Strings, so werden diese Zeichen „abgeschnitten“ (vgl. auch Anmerkungen auf S. 117)

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Beispiel: Addition von Strings

Die Zeichenketten string1 und string2 sollen „addiert“ bzw. aneinander gehängt werden.

Die Zeichenketten string3 und 4 sind selbst definiert und sollen ebenfalls addiert werden.

Umsetzung im Anwenderprogramm

```
string1=$Zeichen$
string2=$kette$
string3=$55 Byte$c55
string4=$65534 Byte$c65534
// Ergebnis: „Zeichenkette“
result=string1+string2
result2=string3+string4
```

Vergleichen Sie auch Anmerkungen zur Verkettung auf S. 113.

Find

Definition

- Funktion find(*String1, String2, Pos1*)

Argumente

- 3 Argumente, *String1, String2* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge, *Pos1* vom Datentyp u16

Wirkung

- *String1*: Zeichenkette, in der eine (Teil-)Zeichenkette gesucht werden soll
- *String2*: zu findende Zeichenkette
- *Pos1*: Die ersten *Pos1* - Vorkommen der zu findenden Zeichenkette ignorieren
- Die Funktion berechnet die Position des ersten Zeichens der Zeichenkette (0..1399u16).

  Sie gibt 65535u16 zurück, falls die Zeichenkette nicht gefunden wurde.
- Für 65534u16 ist die Konstante END definiert.

Datentyp Ergebnis (Rückgabe)

- Datentyp u16

Beispiel: Suche Zeichenfolge

In der Variable string=$Zeichenkette$ soll die Zeichenkette „kette“ gesucht werden. Es sollen keine (0) Vorkommen ignoriert werden.

Wenn „kette“ nicht gefunden wird, soll die Variable Error auf 1 gesetzt werden.

Umsetzung im Anwenderprogramm

```
Error
String=$Zeichenkette$
Suche=$kette$
Result=find(String,Suche,0u16)
if Result==65535u16 then Error=EIN endif
```

Stringcast

Definition

- Funktion stringcast(*String, Data, Pos*)

Argumente

- 3 Argumente, *String* vom Datentyp c1400,*Data* vom beliebigen Datentyp, *Pos* vom Datentyp u16

Wirkung

- *String* Zeichenkette (1400 Bytes bzw. einer selbst definierten Stringlänge), von der eine bestimmte Anzahl von Bytes in einen anderen Datentyp kopiert werden sollen. Die Anzahl der Bytes wird durch den Datentyp von *Data* festgelegt. Dabei werden nur die Rohdaten kopiert (cast) und keine Umwandlung der Datentypen vorgenommen.
- *Pos*: Die Position des ersten Zeichens im String, das in den Zieltyp kopiert werden sollen.

Datentyp Ergebnis (Rückgabe)

- n Bits (n = Bytelänge des Datentyps *Data*) aus dem *String,* d.h. die Rohdaten werden zurückgegeben

Beispiel: Konvertierung eines Strings in eine Fließkommazahl

In der Variable a=98 sollen die ersten zwei Byte in eine Fließkommazahl geschrieben werden

Umsetzung im Anwenderprogramm

```
a=$98$
z=stringcast(a,0.0,0u16)
// z interpretiert 0x39 0x38 (ASCII „98“) als „72.9600000“
```

Hinweis:

In Verbindung mit stringset und stringcast können c1400 Strings zur Verwaltung von Datenarrays genutzt werden. Siehe hierzu Beispiel zu stringset auf S. 230

Stringset

Definition

- Funktion stringset(*String, Data, Pos*)

Argumente

- 3 Argumente, *String* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge, *Data* vom beliebigen Datentyp, *Pos* vom Datentyp u16

Wirkung

- *String* Zeichenkette, in der ein oder mehrere Bytes verändert werden sollen.
- *Data*: Diese Bytes (=Zeichen) ersetzen einen Teil von *String*. Es werden also soviele Bytes in den String kopiert, wie *Data* selbst enthält.
- *Pos*: Die Startposition der Bytes im String, die ausgetauscht werden sollen. Die Anzahl der Bytes ergibt sich aus dem Datentyp von *Data.*

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: Ersetze Zeichenfolge

In der Variable a=$ nnette$ soll das 1.te Zeichen auf den Wert 65 =('A') gesetzt werden

Umsetzung im Anwenderprogramm

```
a=$ nnette$
if systemstart() then stringset(a,65,0u16) endif
```

Beispiel: Datenarray - Erstellen und Auslesen

In einem Array sollen alle 15 Min-Werte der Temperatur von der Gruppenadresse '1/1/1'f16 abgelegt werden. Gleichzeitig soll die Temperaturdifferenz der letzten Änderung aus diesem Array entnommen werden.

Die Umsetzung ist wie folgt. Dabei gilt es zu berücksichtigen, dass bei Stringset der Anwender die Bytegröße angeben muss.

Mit dem Debugger (S. 152) können Sie auch die „Rohdaten“ im Array anschauen. Allerdings ist dies wohl nur für Ganzzahltypen sinnvoll.

Es können bis zu 65534 Bytes in Strings genutzt werden.

```
[EibPC]
array=$$
Var='1/1/1'f16
ReadVar=0.0
// Bytessize of f16 == 2
ByteSize=2u16
Pos=0u16

if cycle(15,0) then {
        Pos=Pos+ByteSize;
        stringset(array,Var,Pos);
        if Pos==END then Pos=0u16 endif
} endif
if cycle(15,0) then {
        if (Pos>2u16) then {
                ReadVar=stringcast(array,Var,Pos-ByteSize)-stringcast(array,Var,Pos)
        } endif
} endif
```

Stringformat

Definition

- Funktion stringformat(*Data, Umwandlungtyp, Format, Feldbreite, Präzision*)

Argumente

- Argument *Data* vom Datentyp uXX, sXX, fXX, mit XX beliebigen, auf Seite 161 definierten Bitlängen.
- Argumente *Format, Feldbreite, Präzision, Umwandlungtyp* vom Datentyp u08

Wirkung

- *Umwandlungstyp*
  - 0: uXX / sXX → Dezimaldarstellung
  - 1: uXX / sXX → Oktaldarstellung
  - 2: uXX / sXX → Hexadezimaldarstellung ('x')
  - 3: uXX / sXX → Hexadezimaldarstellung ('X')
  - 4: fXX → Fließkommadarstellung
  - 5: fXX → Exponentialdarstellung ('e')
  - 6: fXX → Exponentialdarstellung ('E')
- Mit Angabe von *Format* wird wie folgt formatiert:
  - 0: (entfällt)
  - 1: Leerzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp sXX oder fXX und keine Umwandlung ins Oktal- oder Hexadezimalformat)
  - 2: Vorzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp sXX oder fXX und keine Umwandlung ins Oktal- oder Hexadezimalformat)
  - 3: Auffüllen mit Nullen (wird ignoriert, falls *Data* vom Datentyp uXX oder sXX bzw. falls bei den Fließkommazahlen die *Präzision* anderweitige Ausdrücke generiert).
  - 4: Auffüllen mit Nullen und Leerzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp sXX oder fXX und keine Umwandlung ins Oktal- oder Hexadezimalformat; wird ignoriert, falls *Data* vom Datentyp uXX oder sXX bzw. falls bei den Fließkommazahlen die *Präzision* anderweitige Ausdrücke generiert.
  - 5: Auffüllen mit Nullen und Vorzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp sXX oder fXX und keine Umwandlung ins Oktal- oder Hexadezimalformat; wird ignoriert, falls *Data* vom Datentyp uXX oder sXX bzw. falls bei den Fließkommazahlen die *Präzision* anderweitige Ausdrücke generiert.)
  - 6: Linksbündig
  - 7: Linksbündig und Leerzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp sXX oder fXX und keine Umwandlung ins Oktal- oder Hexadezimalformat)
  - 8: Linksbündig und Vorzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Berechnungstyp sXX oder fXX und keine Umwandlung ins Oktal- oder Hexadezimalformat)
  - 9: Alternative Formatierung (s.u) , nur zulässig, falls keine Umwandlung ins Dezimalformat:
  - 10: Alternative Formatierung (s.u) und Leerzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Berechnungstyp fXX)
  - 11: Alternative Formatierung (s.u) und Vorzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Berechnungstyp fXX)
  - 12: Alternative Formatierung (s.u) und Auffüllen mit Nullen (nur zulässig, falls keine Umwandlung ins Dezimalformat; wird ignoriert, falls *Data* vom Datentyp uXX oder sXX und *Präzision* angegeben wurde)
  - 13: Alternative Formatierung (s.u), Auffüllen mit Nullen und Leerzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp fXX)
  - 14: Alternative Formatierung (s.u), Auffüllen mit Nullen und Vorzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp fXX)

- 15: Alternative Formatierung (s.u) und linksbündig (nur zulässig, falls keine Umwandlung ins Dezimalformat)
- 16: Alternative Formatierung (s.u), linksbündig und Leerzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp fXX)
- 17: Alternative Formatierung (s.u), linksbündig und Vorzeichen bei positiver Zahl (nur zulässig, falls *Data* vom Datentyp fXX)
- 18: 0x-Präfix auch bei 0 und Auffüllen mit Nullen (nur zulässig bei Umwandlung ins Hexadezimalformat 'x'. Wird ignoriert, wenn *Präzision* angegeben wurde)
- 19: 0x-Präfix auch bei 0 und linksbündig (nur zulässig bei Umwandlung ins Hexadezimalformat 'x')
- 20: 0X-Präfix auch bei 0 und Auffüllen mit Nullen (nur zulässig bei Umwandlung ins Hexadezimalformat 'X'. Wird ignoriert, wenn *Präzision* angegeben wurde)
- 21: 0X-Präfix auch bei 0 und linksbündig (nur zulässig bei Umwandlung ins Hexadezimalformat 'X')

- *Feldbreite:* Angabe der minimalen Feldbreite
- *Präzision* Angabe der Genauigkeit
- Alternative Formatierung:
  Bei einer Umwandlung ins Oktalformat wird das Zeichen "0" vorangestellt, bei einer Umwandlung ins Hexadezimalformat die Zeichenkette "0x" bzw. "0X".
  Bei einer Umwandlung in Fließkomma- oder Exponentialdarstellung enthält die Ausgabe ein Dezimalzeichen, auch falls diesem keine Ziffern folgen.

  Bei Umwandlungen in andere Formate ist das Ergebnis undefiniert.

Datentyp Ergebnis (Rückgabe)

- Datentyp c1400

Beispiel: Stoppuhr V2 (Vgl. Beispiel:Stoppuhr, S. 184).

Stoppen der Zeit in Sekunden, an der die Variable Stopper_Go auf ein steht. Es soll ein c1400 String angegeben werden, der die Zeit Formatiert in 000d:000h:000m:000s (Tage, Stunden, Minuten, Sekunden) ausgibt.

Hier die Umsetzung, wobei die Sekunden in der Variablen *Stopper_time* und die formatierte Ausgabe in *Stopper* zu finden sind. Im Unterschied zu Beispiel:Stoppuhr (S. 184) wird die Zeitspanne mittels after gezählt.

```
[EibPC]
Stopper=$$
Stopper_time=0s32
Stopper_Go=AUS
if (Stopper_Go) then {
        Stopper_time=1s32;
        write(address(85u16),$Starte$c14)
} endif
if after(change(Stopper_time),1000u64) then Stopper_time=Stopper_time+1s32 endif

// Ende Stoppzeit
if !Stopper_Go then {
        Stopper=stringformat(Stopper_time/86400s32,0,3,3,3)+$d:$+\\
            stringformat(mod(Stopper_time,86400s32)/3600s32,0,3,3,3)+$h:$+\\
            stringformat(mod(Stopper_time,3600s32)/60s32,0,3,3,3)+$m:$+\\
            stringformat(mod(Stopper_time,60s32),0,3,3,3)+$s$
} endif
```

Split

Definition

- Funktion split(*String*, *Pos1, Pos2*)

Argumente

- 3 Argumente, *String* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge, *Pos1* und *Pos2* vom Datentyp u16

Wirkung

- *String*: Zeichenkette, aus der eine Zeichenkette entnommen werden soll
- *Pos1*: erstes Zeichen der abzutrennenden Zeichenkette (0...1399u16)
- *Pos2*: letztes Zeichen der abzutrennenden Zeichenkette (0...1399u16). Wenn *Pos2* gleich 65534u16 (vordefinierte Konstante END) ist, so wird die Zeichenkette bis zu deren Ende abgetrennt.
- Die Variable *String* muss vom Datentyp c1400 sein.
- Rückgabewert: die von Variable String abgetrennte Zeichenkette

Datentyp Ergebnis (Rückgabe)

- Eine Zeichenkette vom Datentyp c1400

Beispiel: split

Aus der Variable string=$Zeichenkette$ soll die Zeichenkette „kette“ entnommen werden. Das erste abzutrennende Zeichen ist an Position 7 (Zählvorgang beginnt bei 0), das letzte Zeichen ist an Position 11.

Die Umsetzung in der EibParserdatei lautet dann:

```
string=$Zeichenkette$
result=split(string, 7u16,11u16)
```

Beispiel: Suche Zeichenfolge (2)

In der Variable string=$Zeichenkette: Hallo$ soll die Zeichenkette „Hallo“ abgetrennt werden.

Umsetzung:

```
String=$Zeichenkette: Hallo$
TeilString=split(String,find(String,$:$,0u16),1399u16)
```

Size

Definition

- Funktion size(*String*)

Argumente

- Ein Argumente, *String* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge

Wirkung

- Von der Zeichenkette *String* soll die Länge bestimmt werden. Die Länge ist durch das Terminierungszeichen „\0“ am Ende von Zeichenketten gegeben.

Datentyp Ergebnis (Rückgabe)

- Datentyp u16

Beispiel: size

Die Länge des string=$Zeichenkette$ soll bestimmt werden.

Umsetzung im Anwenderprogramm

```
string=$Zeichenkette$
result=size(string)
```

Capacity

Definition

- Funktion capacity(*String*)

Argumente

- Ein Argument, *String* vom Datentyp c1400 bzw. mit einer selbst definierten Stringlänge

Wirkung

- Von der Zeichenkette *String* soll die maximal verfügbare Länge bestimmt werden.

Datentyp Ergebnis (Rückgabe)

- Datentyp u16

Beispiel: capacity

Die maximal verfügbare Länge des strings=$Zeichenkette$ soll bestimmt werden.

Umsetzung im Anwenderprogramm

```
string=$Zeichenkette$
// result ist 1400, da der String c1400 max. 1400 Zeichen aufnehmen kann.
result=capacity(string)
```

Tostring

Definition

- Funktion tostring(*char1[,char2, ... charN]*)

Argumente

- Mindestens ein Argument, *char1* vom Datentyp u08 als Charactercode der UTF-8 Codierung (vgl. http://de.wikipedia.org/wiki/UTF-8)

Wirkung

- Es wird Zeichenkette aus den einzelnen Bytes gebildet, die terminierende Null wird automatisch angehängt.

Datentyp Ergebnis (Rückgabe)

- Datentyp c1400

Beispiel: capacity

Die maximal verfügbare Länge des strings=$Zeichenkette$ soll bestimmt werden.

Umsetzung im Anwenderprogramm

```
Eurozeichen=tostring(0xE2,0x82,0xAC)
```

Encode

Definition

- Funktion encode(*String,Quellcodierung, Zielcodierung*)

Argumente

- *String* vom Datentyp c1400 bzw. mit einer selbst definierten Stringlänge
- *Quellcodierung* mit den üblichen Bezeichnungen, z.B. „UTF-8“ als c14 String.
- *Zielcodierung* mit den üblichen Bezeichnungen, z.B. „UTF-8“ als c14 String.

Wirkung

- Eine Zeichenkette *String*, die in der Quellcodierung vorliegt, wird in die Zielcodierung übertragen.

Datentyp Ergebnis (Rückgabe)

- Datentyp String-Format

Beispiel: encode

Einen String umkodieren von UTF-8 nach ISO-8859

Umsetzung im Anwenderprogramm

```
// String
s1=$Hallöchen$c4000

// String von UTF nach Windows (Deutsch) kodieren;
sDE=encode(s1,$UTF-8$c14,$ISO-8859-15$c14)
```

Einen String umkodieren von ISO-8859 nach UTF-8 codieren

```
// String von UTF nach Windows (Europa) kodieren:
sEU=encode(s1,$UTF-8$c14,$ISO-8859-1$c14)
sUTF=encode(sDE,$ISO-8859-1$c14,$UTF-8$c14)
```

Beachten Sie auch die Hinweise auf S. 112.

Urldecode

Definition

- Funktion urldecode(*String,Quellcodierung, Zielcodierung*)

Argumente

- *String* vom Datentyp c1400 bzw. mit einer selbst definierten Stringlänge
- *Quellcodierung* mit den üblichen Bezeichnungen, z.B. „UTF-8“
- *Zielcodierung* mit den üblichen Bezeichnungen, z.B. „UTF-8“

Wirkung

- Eine Zeichenkette *String*, die in der Quellcodierung vorliegt, wird in die Zielcodierung übertragen, wobei die URL-Codierung verwendet wird.

Datentyp Ergebnis (Rückgabe)

- Datentyp String-Format

Beispiel: encode

```
Einen String $ÜberMich.de$,umkodieren
```

Umsetzung im Anwenderprogramm

```
// String:org: $Hallöchen auf http:\\enertex.de$
org=urldecode($Hall%c3%b6chen%20auf%20http%3a%5c%5cenertex.de$,$utf-8$c14,$utf-8$c14)
```

Beachten Sie auch die Hinweise auf S. 112.

Urlencode

Definition

- Funktion urlencode(*String,Quellcodierung, Zielcodierung*)

Argumente

- *String* vom Datentyp c1400 bzw. mit einer selbst definierten Stringlänge
- *Quellcodierung* mit den üblichen Bezeichnungen, z.B. „UTF-8“
- *Zielcodierung* mit den üblichen Bezeichnungen, z.B. „UTF-8“

Wirkung

- Eine Zeichenkette *String*, die in der Quellcodierung vorliegt, wird in die Zielcodierung übertragen, wobei die URL-Codierung verwendet wird.

Datentyp Ergebnis (Rückgabe)

- Datentyp String-Format

Beispiel: encode

```
Einen String $ÜberMich.de$,umkodieren
```

Umsetzung im Anwenderprogramm

```
// String ulr=$Hall%c3%b6chen%20auf%20http%3a%5c%5cenertex.de$
url=urlencode($Hallöchen auf http:\\enertex.de$,$utf-8$c14,$utf-8$c14)
```

Beachten Sie auch die Hinweise auf S. 112.

RS232 Schnittstelle

Wenn Sie ihre KNX™ Verbindung mit einer IP Schnittstelle herstellen, so können Sie die RS232 Schnittstelle frei programmieren, über die RS232 auf andere Geräte zuzugreifen. Dabei ist die Syntax exakt identisch zu den Netzwerkfunktionen für Lesen und Schreiben auf UDP bzw. TCP . Vergleichen Sie hier die Ausführungen auf S.67 und S.74.

Konfiguration

Um die Schnittstelle nutzen zu können, müssen Sie diese konfigurieren. Dazu fügen Sie im Anwendungsprogramm die Sektion [RS232] ein bzw. nutzen Sie das Menü Optionen – RS232-Einstellungen.

Die möglichen Parameter entnehmen Sie den vorausgehenden Kommentaren.

```
[RS232]
// Baudrate für die RS-232-Anwenderschnittstelle: Dezimaldarstellung.
// Zulässige Werte: 0 , 50 , 75 , 110 , 134 , 150 , 200 , 300 , 600 , 1200 , 1800 , 2400 , 4800
// 9600 , 19200, 38400 , 57600 , 115200 , 230400
9600
//Datenbits für die RS-232-Anwenderschnittstelle: Dezimaldarstellung. Gültige Werte: 5, 6, 7, 8
8
//Stoppbits für die RS-232-Anwenderschnittstelle: Dezimaldarstellung. Gültige Werte: 1, 2
1
//Parität für die RS-232-Anwenderschnittstelle: Dezimaldarstellung. aus = 0 / gerade = 1 /
ungerade = 2
0
//Flußsteuerung für die RS-232-Anwenderschnittstelle: Dezimaldarstellung.
//aus = 0 / RTS/CTS = 1 / Xon/Xoff = 2
0
```

Readrs232

Definition

- Funktion readrs232(*RawData,Len*)

Argumente

- *RawData* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge
- *Len* vom Datentyp u16

Wirkung

- Von der asynchronen seriellen Schnittstelle empfangene Daten (maximale Länge: 1400B) werden in das Zielobjekt *RawData* geschrieben, beginnend an der in *Len* definierten Position. Abschließend wird die Position um die Anzahl der angefügten Bytes entsprechend erhöht. Das Zielobjekt ist i. a. nicht terminiert.
- Wenn ein RS232 Telegramm an den Enertex® EibPC geschickt wird, aktualisiert jede Funktion readrs232 ihr Argumente in der angegebenen Weise.
- Um zu prüfen, ob ein Telegramm eingetroffen ist, kann die Funktion event auf readrs232 angewendet werden.

Datentyp Ergebnis (Rückgabe)

- keine

Hinweis:

Je nach Konfiguration der RS232-Schnittstelle (Baudrate) können pro Verarbeitungsschleife mehr als ein Zeichen in den Puffer des Enertex® EibPC auflaufen. Die Länge des aktuellen Puffer kann mit dem 2. Argument erfragt werden. Wenn die Funktion im Anwenderprogramm verarbeitet wird (s.u.), so muss dieses Argument zurückgesetzt werden.

Beispiel: Einlesen von RS232 Daten

Es sollen von der Schnittstelle bei Eintreffen von neuen Binärdaten diese in einem Pufferstring geschrieben werden.

```
[EibPC]
rawdata=$$
len=0u16
Buffer=$$
if event(readrs232(rawdata,len)) then {
  Buffer=Buffer + split(rawdata,0u16,len);
  len=0u16
} endif
```

Beispiel: Einlesen von 10 Bytes mit der RS232

Es sollen von der RS232 Schnittstelle 10 Bytes gelesen und in einen Puffer geschrieben werden.

```
[EibPC]
rawdata=$$
len=0u16
Buffer=$$
if event(readrs232(rawdata,len)) and len>9u16 then {
  Buffer=Buffer + split(rawdata,0u16,9u16);
  len=len-10u16;
  rawdata=split(rawdata,10u16,EOS)
} endif
```

Resetrs232

Definition

- Funktion resetrs232()

Argumente

- keine

Wirkung

- Führt einen Reset auf die RS232 aus.

Datentyp Ergebnis (Rückgabe)

- keine

Sendrs232

Definition

- Funktion sendrs232(*arg 1*[, *arg2, ... argN*])

Argumente

- *arg2* bis *argN* beliebig

Wirkung

- Zu übertragene „Nutzdaten“ sind beliebig in Anzahl und Datentyp.
- Wenn *arg2* bis *argN* vom Datentyp c1400 sind, werden die terminierenden Nullzeichen der Strings nicht mit übermittelt.

Datentyp Ergebnis (Rückgabe)

- keine

KNX-Telegramm-Routing

Mit den beiden Funktionen address und readknx kann der Enertex® EibPC wie ein frei programmierbarer Router für KNX™ Telegramme eingesetzt werden: Wenn Sie beispielsweise per TCP/IP Client die Gruppenadresse (als Zahl) an den Enertex® EibPC übermitteln, können mit address direkt diese Gruppenadressen schreiben, ohne eine weitere Programmierung vornehmen zu müssen. Gleichermaßen kann ein eintreffendes Telegramm mit readknx an den Client zurückgemeldet werden. Im Opensource Projekt „EibPC-Homecontrol“ ist diese Vorgehensweise genutzt. address kann in den Funktionen für den Buszugriff, wie z.B. event, write, scene etc., anstelle einer Gruppenadresse verwendet werden.

Address

Definition

- Funktion address(*Variable*)

Argumente

- Ein Argument vom Datentyp u16

Wirkung

- Rückgabewert: eine Gruppenadresse, wie sie von write, read etc. verwendet werden kann.

Datentyp Ergebnis (Rückgabe)

- Datentyp Gruppenadresse

Eine Besonderheit der Funktionen für den Buszugriff ist, dass diese als Argumente Gruppenadressen erwarten. So muss z.B. das 1. Argument in write('5/3/11'b01, EIN) eine Gruppenadresse sein. Die Funktion address konvertiert eine 16-Bit Zahl in eine Gruppenadresse. Diese Zahl berechnet sich aus Adresse=HG x 2048+MG x 256 + UG, wobei im Beispiel '5/3/11' HG=5, MG=3 und UG=11 ist. Sie müssen diese Zahl selbst berechnen oder die Funktion getaddress benutzen.

Beispiel: address

Sie möchten auf die Gruppenadresse *'5/3/11'b01* den EIN schreiben, wenn das System gestartet wurde.

Umsetzung im Anwenderprogramm

```
if systemstart() then write(address(11019u16),EIN) endif
```

Getaddress

Definition

- Funktion getaddress(*Gruppenadresse*)

Argumente

- *Gruppenadresse* Manuelle oder importierte Gruppenadresse

Wirkung

- Die Funktion gibt den 16 Bit Wert (vorzeichenlos) zurück, welche die Gruppenadresse repräsentiert.

Datentyp Ergebnis (Rückgabe)

- u16

Um 12:00 soll die Gruppenadresse 1/1/27 gelesen und um 12:30 soll der Wert 10% auf diese geschrieben werden.

```
[EibPC]
a=getaddress("Dimmer-1/1/27")
if htime(12,00,00) then read(address(a)) endif
if htime(12,30,00) then write(address(a),16) endif
```

Hinweis

Im Normalfall benötigen Sie diese Funktion nicht, da Sie direkt read("Dimmer-1/1/27") etc. coden können. Diese Funktion ist nur für vereinfachtes Makrocoding gedacht.

Gaimage

Definition

- Funktion gaimage(*Nummer)*

Argumente

- *Nummer* vom Datentyp u16.

Wirkung

- Die Funktion gibt den Inhalt des internen Abbildspeichers im Enertex® EibPC der betreffenden Gruppenadresse zurück. Dabei werden die gespeicherten Binärdaten des Telegramms in einen String kopiert (vgl. convert) und das Ergebnis als Rückgabewert weitergegeben.
- Die *Nummer* stellt die Gruppenadresse eines ETS-Exports dar.
- Die Funktion dient zur Übermittlung von im Enertex® EibPC gespeicherten Zuständen von Gruppenadressen.

Datentyp Ergebnis (Rückgabe)

- c1400

Eine Besonderheit der Funktion ist, dass sie als Argument die Gruppenadressen als 16 Bit Adresse erwartet. Die Nummer stellt eine 16-Bit Zahl dar. Sie berechnet sich aus Adresse=HG x 2048+MG x 256 + UG, wobei zum Beispiel '5/3/11' HG=5, MG=3 und UG=11 ist. Sie müssen diese Zahl selbst berechnen oder die Funktion getaddress benutzen.

Der aktuelle Wert einer Gruppenadresse '1/2/1'f16 soll um 12:00 Uhr als Text in eine Variable gespeichert werden.

```
MyVar=$$
if htime(12,0,0) then {
        MyVar=convert(gaimage(getaddress('1/2/1'f16)),$$)
}endif
```

Getganame

Definition

- Funktion getganame(*Gruppenadresse, Codierung)*

Argumente

- Gruppenadresse Eine importierte Gruppenadresse
- Codierung mit der üblichen Bezeichnung, z.B. $UTF-8$c14 als c14 String, dient der direkten Umwandlung der GA in eine beliebige Systemcodierung.

Wirkung

- Die Funktion gibt den Namen der Gruppenadresse im Enertex® EibPC-Format zurück, wenn diese Gruppenadresse im Anwendungsprogramm importiert wurde (ESF Import)

Datentyp Ergebnis (Rückgabe)

- c1400

Der Name einer Gruppenadresse soll als Text in der Standard-Windowscodierung (iso8859-15) in einer Variable gespeichert werden.

```
// MyVar=$"LüftenArbeiten-0/0/2"$
MyVar=getganame("LüftenArbeiten-0/0/2",$utf-8$c14)
```

ReadKnx

Definition

- Funktion readknx(*Nummer*, *Ausgabe*)

Argumente

- *Nummer* vom Datentyp u16.
- *Ausgabe* vom Datentyp c1400.

Wirkung

- Bei Eintreffen eines Telegramms schreibt die Funktion auf das Argument *Nummer* die Gruppenadresse des Telegramms. Die Binärdaten des Telegramms werden in den String *Ausgabe* kopiert. Dabei nimmt nun *Ausgabe* den Typ des zuletzt gesendeten Datentyps an. Um diesen in Text zu wandeln, kann convert die Daten in lesbare Form bringen.

Datentyp Ergebnis (Rückgabe)

- Das Ergebnis der Konvertierung

Hinweis:

Auf die Funktion readknx kann die Event-Funktion angewendet werden.

Beispiel: Verschicken der aktuell eintreffenden Telegramme per UDP

Sie möchten die jeweils, aktuelle am KNX™ Bus geschriebene, Telegramminformation per UDP an den Empfänger 192.168.22.199 schicken. Dabei soll die Gruppenadresse im Format u16 und die Information im Format GA:XXXXX INF:YYYYYYY übermittelt werden.

```
adr=0u16
info=$$
if event(readknx(adr,info)) then {
        sendudp (5000u16, 192.168.22.199,$GA:$+convert(adr,$$)+$INF:$+info)
}endif
```

Readrawknx

Definition

- Funktion readrawknx(*Kontrollfeld, PhyAdresse, Zieladresse, IsGroubAdress, RoutingZähler, Bits, Nutzdaten*)

Argumente

- *Kontrollfeld* vom Datentyp u08.
- *PhyAdresse* des Absenders vom Datentyp u16 (in der üblichen Schreibweise z.B. 2.1.14).
- *Zieladresse* vom Datentyp u16.
- *IsGroubAdress* vom Datentyp b01.
- *RoutingZähler* vom Datentyp u08.
- *BitLänge* vom Datentyp u08.
- *Nutzdaten* vom Datentyp $$.

weiter Informationen zum Telegrammaufbau finden Sie auf S. 223

Wirkung

- Wenn ein beliebiges KNX-Telegramm am Bus beobachtet wird, aktualisiert die Funktion readrawknx ihre Argumente.
- Wenn dies der Fall ist, so werden die Argumente der Funktion soweit mit Daten „gefüllt“, bis die empfangene Datenmenge (eingetroffenes KNX Telegramm) zur Datenlänge der Argumente der readrawknx Funktion passt. In jedem Fall werden die Variablen *PhysAdresse* und *Gruppenadresse* von der Funktion readrawknx bei jedem neu eintreffenden Telegramm mit den aktuellen Daten des Versenders überschrieben.
- Die *PhyAdresse* wird in der üblichen Schreibweise (2.1.4) angegeben.
- Wenn *IsGroubAdress* auf EIN steht, wurde ein Telegramm an eine Gruppenadresse geschickt, im anderen Fall an einen Aktor.
- *BitLänge* ist die Bitlänge des Telegramms (Datentyp) in Byte+1.
- Um zu prüfen, ob ein Telegramm eingetroffen ist, kann die Funktion event auf readrawknx angewendet werden. Dies wird notwendig, wenn Telegramme mit identischen Inhalt ausgewertet werden müssen (s. Funktion readupd S. 245).

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: KNX™ Telegramme auswerten

Es soll gezählt werden, wie viele Telegramme von der physikalischen Adresse 1.3.14 auf die Gruppenadresse 2/4/44 gesendet wurden. Dabei muss unterschieden werden, ob das Telegramm an eine Gruppenadresse oder an eine physikalische Adresse geschrieben wurde.

Umsetzung im Anwenderprogramm:

```
Raw_Kontroll=0
Raw_Sender=10.2.1
Raw_GA=0u16
Raw_IsGa=AUS
Raw_RoutingCnt=0
Raw_Len=0
Raw_Data=$$
count=0u08
if event(readrawknx(Raw_Kontroll,Raw_Sender,Raw_GA,Raw_IsGa,Raw_RoutingCnt,
Raw_Len,Raw_Data)) and Raw_Sender==1.3.14 and Raw_GA==getaddress('2/4/44'b01) and
Raw_IsGa  then {
        count=count+1
} endif
```

Beispiel: Aktorüberwachung

Es wird überprüft, ob vom einem KNX Gerät mindestens alle 120 Minuten ein Telegramm eintrifft. Außerdem ein paar Statistiken zum Bustransfer.

Umsetzung im Anwenderprogramm:

```
// ---------------------------------------
// Physikalische Geräteadresse
// ---------------------------------------
Raw_Dev=1.1.60

// Auswertung
// ---------------------------------------
// Maximale Zeit zwischen zwei Telegrammen vom selben Gerät seit Beginn der Aufzeichnung
Raw_MaxTime=0u16
// Minimale Zeit zwischen zwei Telegrammen vom selben Gerät seit Beginn der Aufzeichnung
Raw_MinTime=65365u16
// Letzte ermittlete Zeit
Raw_CalcTime=0u16
// Durchschnittswert über alle Telegramme vom selben Gerät
Raw_AvgTime=0u64

// Fehlerzeit: Wann soll ein Fehler erkannt werden
Raw_TimeWatch=120u64*60000u64


// Argumente von readrawknx:
Raw_Kontroll=0
Raw_Sender=0.0.0
Raw_GA=0u16
Raw_IsGa=AUS
Raw_RoutingCnt=0
Raw_Len=0
Raw_Data=$$


// ---------------------------------------
// Hilfsvariablen
Raw_AvgTrigger=0u64
Raw_Error=AUS
Raw_AvgTimeSum=0u64
// In welcher Zeitskala wird gerechnet:  1000 entspricht in Sekundengenauigkeit
//                                       60000 entspricht in Minutengenauigkeit
Raw_TimeScale=1000u64

Raw_Time=Raw_TimeWatch


// Nur bei Gruppentelegrammen auf den EibPC reagieren und auch nur, wenn die SenderAdresse die richtige ist

if event(readrawknx(Raw_Kontroll,Raw_Sender,Raw_GA,Raw_IsGa,Raw_RoutingCnt,Raw_Len,Raw_Data)) and  Raw_Sender==Raw_Dev and Raw_IsGa then {
                // Zeit seit letzem Diagramm in Sekunden wandeln und Min und Max berechnen
                // Raw_Time wird über die Delayc-Funktion berechnet und kann damit hier ausgewertet werden
                Raw_CalcTime=convert((Raw_TimeWatch-Raw_Time)/Raw_TimeScale,0u16);
                if Raw_MaxTime<Raw_CalcTime then Raw_MaxTime=Raw_CalcTime endif;
                if Raw_MinTime>Raw_CalcTime then Raw_MinTime=Raw_CalcTime endif;
                // Durchschnittswert=Raw_AvgTime/Raw_Trigger
                Raw_AvgTimeSum=Raw_AvgTimeSum+convert(Raw_CalcTime,0u64);
                Raw_AvgTrigger=Raw_AvgTrigger+1u64;
                Raw_AvgTime=Raw_AvgTimeSum/Raw_AvgTrigger;
} endif
```

```
// Alle Raw_TimeWatch wird ein Telgramm erwartet: Dann wird delay immer wieder neu getriggert
// sonst Fehlerbedingung!
if delayc(change(Raw_AvgTrigger),Raw_TimeWatch,Raw_Time) then {
        Raw_Error=EIN
} endif
```

Hinweis:

Die event-Funktion bzw. in der if-Anweisung die Verknüpfung mit *Telegramm* stellt sicher, dass in jedem Fall der then-Zweig aufgerufen wird. Damit wird zudem gewährleistet, dass die Daten auf den Bus geschrieben werden, wenn identische UDP-Telegramme mehrfach auftreten (und aufgrund des Validierungsschemas, siehe auch Seiten 53 und 168).

Netzwerkfunktionen

Freischaltcodes

Um die Netzwerkfunktionen dieses Kapitels nutzen zu können, müssen Sie die Option NP im Enertex® EibPC freischalten. Sie müssen dazu die Seriennummer ihres Enertex® EibPC kennen und ein entsprechendes Zusatzpaket kaufen. Der Freischaltcode ist immer an die Seriennummer des Gerät gebunden und ist nicht übertragbar auf andere Geräte.

Um einen gültigen Freischaltcode an den Enertex® EibPC zu übertragen, wählen Sie im Menü EibPC → Freischaltcode Übertragen und folgen Sie den Anweisungen des Dialogs.

Namensauflösung

Für einige der Netzwerkfunktionen (sendmail, resolve) benötigt der Enertex® EibPC einen Zugang zu einem DNS Server. Auf Seite 145 finden Sie hierzu die Konfiguration bzw. wie Sie eine erfolgreiche Konfiguration testen können.

Standard-Ports

Die Ports, über welche der Enertex® EibPC kommuniziert, können über das Menü Optionen/Ports verändert werden. Dieses Menü generiert eine Sektion [Ports], die dann die folgenden Einträge erghält

```
[Ports]
///UDP InPort
12600
// UDP OutPort
3245
// TCP Port
23456
//Modbus Port
5901
```

UDP Telegramme

UDP Ports

Der Enertex® EibPC selbst verschickt die Daten beim UDP Transfer immer über seinen Port 4807. Der Zielport des Empfängers kann beliebig gewählt werden.

Der Enertex® EibPC empfängt Daten beim UDP Transfer standardmäßig über seinen Port 4806. Der Sender muss daher diesen Zielport angegeben. Der Port, über welchen der Senders Telegramme verschickt, kann vom Enertex® EibPC ermittelt werden.

Readudp

Definition

- Funktion readudp(*Port, IP, arg 1*[*, arg2, ... argN*])

Argumente

- Argument *Port* vom Datentyp u16 (und ist der Port, über den der Absender Daten verschickt. Zielport des Senders muss immer Port 4806 sein).
- Argument *IP* vom Datentyp u32 (die Adresse des Absenders in der üblichen Schreibweise z.B. 192.168.22.100)
- *arg2* bis *argN* beliebig

Wirkung

- Übertragene „Nutzdaten“ beginnen ab dem 3 Argument. Diese sind beliebig in Anzahl und Datentyp.
- Wenn ein beliebiges UDP Telegramm an den Enertex® EibPC geschickt wird, aktualisiert jede Funktion readudp ihre Argumente. Wenn dies der Fall ist, so werden die Argumente der Funktion soweit mit Daten „gefüllt“, bis die empfangene Datenmenge (eingetroffenes UDP Telegramm) zur Datenlänge der Argumente der readudp Funktion passt. In jedem Fall werden die Variablen *Port* und *IP* von der Funktion readudp bei jedem neu eintreffenden Telegramm mit den aktuellen Daten des Versenders überschrieben.
- Die IP Adresse (Variable *IP*) wird in der üblichen Schreibweise (xxx.xxx.xxx.xxx mit xxx: Zahl zwischen 0 und 255) abgegeben.
- Wenn Ihr LAN Gerät über DNS einen Namen ansprechbar ist, kann die Funktion resolve anstelle einer IP Adresse stehen.
- Um zu prüfen, ob ein Telegramm eingetroffen ist, kann die Funktion event auf readudp angewendet werden. Dies wird notwendig, wenn Telegramme mit identischen Inhalt ausgewertet werden müssen (s.u.).
- Der Enertex® EibPC empfängt immer auf Port 4806. Dieser Port kann nicht verändert werden und ist vom Sender von UDP Telegrammen zu berücksichtigen.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: Daten von UDP Telegramme auf den KNX™ Bus schreiben

Ein UDP Telegramm wird vom Absender 122.32.22.1 an den Enertex® EibPC über den Port 2243u16 des Absenders geschickt. Die Nutzdaten sind drei u08 Daten und sollen immer, wenn ein Telegramm eingetroffen ist, an die Gruppenadressen 3/4/0,3/4/1,3/4/2 geschickt werden.

Umsetzung im Anwenderprogramm:

```
Port=0u16
IP=0u32
Data1=0;Data2=0;Data3=0
Telegramm=event(readudp(Port, IP,Data1,Data2,Data3))
if (Port==2243u16) and (IP==122.32.22.1) and Telegramm then \\
                    write('3/4/0'u08,Data1);                \\
                    write('3/4/1'u08,Data2);                \\
                    write('3/4/2'u08,Data3)                 \\
                    endif
```

Hinweis:

Die event-Funktion bzw. in der if-Anweisung die Verknüpfung mit *Telegramm* stellt sicher, in jedem Fall der then-Zweig aufgerufen wird und die Daten auf den Bus geschrieben werden, wenn identische UDP-Telegramme mehrfach geschickt werden (und aufgrund des Validierungsschemas, siehe auch Seiten 53 und 168).

Sendudp

Definition

- Funktion sendudp(*Port, IP, arg 1*[*, arg2, ... argN*])

Argumente

- Argument *Port* vom Datentyp u16
- Argument *IP* vom Datentyp u32 (Adresse des Empfängers in der üblichen Schreibweise z.B. 192.168.22.100)
- *arg2* bis *argN* beliebig

Wirkung

- Argument *Port* ist der Port, an dem der Enertex® EibPC Daten verschickt.
- Der Enertex® EibPC selbst verschickt die Daten über seinen Port 4807.
- Übertragene „Nutzdaten“ beginnen ab dem 3 Argument. Diese sind beliebig in Anzahl und Datentyp.
- Die IP Adresse (Variable *IP*) wird in der üblichen Schreibweise (xxx.xxx.xxx.xxx mit xxx: Zahl zwischen 0 und 255) abgegeben.
- Wenn Ihr LAN Gerät über DNS einen Namen ansprechbar ist, kann die Funktion resolve anstelle einer IP Adresse stehen.
- Wenn *arg2* bis *argN* vom Datentyp c1400 sind, werden die terminierenden Nullzeichen der Strings mit übermittelt.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: UDP Telegramme versenden

Alle 2 Minuten soll ein UDP Telegramm an den Empfänger www.enertex.de an den Port 5555u16 vom Enertex® EibPC geschickt werden. Als Nutzendaten soll ein 32-Bit Zähler für die Telegramme und der String „Ich lebe noch“ geschickt werden.

Umsetzung im Anwenderprogramm:

```
Count=0u32
if cycle(2,00) then sendudp(5555u16,resolve($www.enertex.de$, Count,$Ich lebe noch$); \\
                    Count=Count+1u32 endif
```

Sendudparray

Definition

- Funktion sendudparray(*Port, IP, arg* ,*size*)

Argumente

- Argument *Port* vom Datentyp u16
- Argument *IP* vom Datentyp u32 (Adresse des Empfängers in der üblichen Schreibweise z.B. 192.168.22.100) oder über die Funktion resolve
- *arg* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge
- Argument *size* vom Datentyp u16 (Anzahl der zu sendenden Bytes)

Wirkung

- Argument *Port* ist der Port, an dem der Enertex® EibPC Daten verschickt.
- Der Enertex® EibPC selbst verschickt die Daten über seinen Port 4807.
- Übertragene „Nutzdaten“ beginnen ab dem 3 Argument. Diese sind beliebig in Anzahl und Datentyp.
- Die IP Adresse (Variable *IP*) wird in der üblichen Schreibweise (xxx.xxx.xxx.xxx mit xxx: Zahl zwischen 0 und 255) abgegeben.
- Wenn Ihr LAN Gerät über DNS einen Namen ansprechbar ist, kann die Funktion resolve anstelle einer IP Adresse stehen.
- Keine terminierenden Nullzeichen.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: UDP Telegramme versenden

Alle 2 Minuten soll ein UDP Telegramm soll an den Empfänger www.enertex.de an den Port 5555u16 vom Enertex® EibPC geschickt werden. Als Nutzendaten sollen die ersten 5 Bytes des String „Ich lebe noch“ geschickt werden.

Umsetzung im Anwenderprogramm:

```
String =$Ich lebe noch$
if cycle(2,00) then sendudparray(5555u16,resolve($www.enertex.de$), String,
size(String),5u16)  endif
```

TCP Server und Client

Server und Client

Der Enertex® EibPC arbeitet sowohl als Server als auch als Client. Alle 100 ms wird auf eine neue Verbindungsanfrage reagiert. Ist der Enertex® EibPC verbunden, so beantwortet er die Anfragen mit der Zykluszeit der Verarbeitungsschleife.

TCP Ports

Der TCP/IP Server des Enertex® EibPC nimmt Verbindungsanfragen immer über seinen Port 4809 entgegen. Alternativ kann der Port wie auf S. 245 verändert werden.

Connecttcp

Definition

- Funktion connecttcp(*Port, IP*)

Argumente

- Argument *Port* vom Datentyp u16
- Argument *IP* vom Datentyp u32 (Adresse des Empfängers in der üblichen Schreibweise z.B. 192.168.22.100)

Wirkung

- Der Enertex® EibPC arbeitet als Client. Er baut eine Verbindung zum angegeben Zielrechner (definiert durch dessen *IP*-Adresse und *Port*) auf.
- Die Funktion liefert den Status ihrer Verarbeitung zurück:
  - bei Erfolg = 0
  - bei andauernder Verarbeitung = 1
  - bei Fehler = 2
  - bei Fehler, wenn eine entsprechende Verbindung schon besteht = 3
  - bei Fehler, da zu viele Verbindungen in Betrieb sind = 4
  - bei automatisch getrennter Verbindung nach timeout = 6
  - nach vom Anwender mit closetcp getrennter Verbindung = 7
  - TCP Gegenstelle hat die Verbindung beendet = 8
  - Initialwert = 9
- nach 30 Sekunden ohne Aktivität auf einer bestehenden Verbindung trennt der Enertex EibPC diese automatisch

Datentyp Ergebnis (Rückgabe)

- u08 (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Closetcp

Definition

- Funktion closetcp(*Port, IP*)

Argumente

- Argument *Port* vom Datentyp u16
- Argument *IP* vom Datentyp u32 (Adresse des Empfängers in der üblichen Schreibweise z.B. 192.168.22.100)

Wirkung

- Der Enertex® EibPC schließt die Client-Verbindung zum angegeben Zielrechner (definiert durch dessen *IP*-Adresse und *Port*).
- Die Funktion liefert den Status ihrer Verarbeitung zurück:
  - bei Erfolg = 0,
  - bei andauernder Verarbeitung = 1 und
  - bei Fehler = 2
  - bei nicht existenter Verbindung = 5
  - Initialwert = 9

Datentyp Ergebnis (Rückgabe)

- u08

Readtcp

Definition

- Funktion readtcp(*Port, IP, arg 1*[*, arg2, ... argN*])

Argumente

- Argument *Port* vom Datentyp u16 (und ist der Port, über den der Absender Daten verschickt).
- Argument *IP* vom Datentyp u32 (die Adresse des Absenders in der üblichen Schreibweise z.B. 192.168.22.100)
- *arg2* bis *argN* beliebig

Wirkung

- Übertragene „Nutzdaten“ beginnen ab dem 3 Argument. Diese sind beliebig in Anzahl und Datentyp.
- Wenn ein beliebiges TCP/IP Telegramm an den Enertex® EibPC geschickt wird, aktualisiert jede Funktion readtcp ihre Argumente. Wenn dies der Fall ist, so werden die Argumente der Funktion soweit mit Daten „gefüllt“, bis die empfangene Datenmenge (eingetroffenes TCP/IP Telegramm) zur Datenlänge der Argumente der readtcp Funktion passt. In jedem Fall werden die Variablen *Port* und *IP* von der Funktion readtcp bei jedem neu eintreffenden Telegramm mit den aktuellen Daten des Versenders überschrieben.
- Die IP Adresse (Variable *IP*) wird in der üblichen Schreibweise (xxx.xxx.xxx.xxx mit xxx: Zahl zwischen 0 und 255) abgegeben.
- Wenn Ihr LAN Gerät über DNS einen Namen ansprechbar ist, kann die Funktion resolve anstelle einer IP Adresse stehen.
- Um zu prüfen, ob ein Telegramm eingetroffen ist, kann die Funktion event auf readtcp angewendet werden. Dies wird notwendig, wenn Telegramme mit identischen Inhalt ausgewertet werden müssen (s.u.).

Datentyp Ergebnis (Rückgabe)

- keine

Sendtcp

Definition

- Funktion sendtcp(*Port, IP, arg 1*[*, arg2, ... argN*])

Argumente

- Argument *Port* vom Datentyp u16
- Argument *IP* vom Datentyp u32 (Adresse des Empfängers in der üblichen Schreibweise z.B. 192.168.22.100)
- *arg2* bis *argN* beliebig

Wirkung

- Argument *Port* ist der Port, an dem der Enertex® EibPC Daten verschickt.
- Übertragene „Nutzdaten“ beginnen ab dem 3 Argument. Diese sind beliebig in Anzahl und Datentyp.
- Die IP Adresse (Variable *IP*) wird in der üblichen Schreibweise (xxx.xxx.xxx.xxx mit xxx: Zahl zwischen 0 und 255) abgegeben.
- Wenn Ihr LAN Gerät über DNS einen Namen ansprechbar ist, kann die Funktion resolve anstelle einer IP Adresse stehen.
- Wenn *arg2* bis *argN* vom Datentyp c1400 sind, werden die terminierenden Nullzeichen der Strings mit übermittelt.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: Siehe Erläuterungen auf S. 74

Sendtcparray

Definition

- Funktion sendtcparray(*Port, IP, arg* ,*size*)

Argumente

- Argument *Port* vom Datentyp u16
- Argument *IP* vom Datentyp u32 (Adresse des Empfängers in der üblichen Schreibweise z.B. 192.168.22.100)
- *arg* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge
- Argument *size* Datentyp u16 (Anzahl der zu sendenden Bytes)

Wirkung

- Argument *Port* ist der Port, an dem der Enertex® EibPC Daten verschickt.
- Übertragene „Nutzdaten“ beginnen ab dem 3 Argument. Diese sind beliebig in Anzahl und Datentyp.
- Die IP Adresse (Variable *IP*) wird in der üblichen Schreibweise (xxx.xxx.xxx.xxx mit xxx: Zahl zwischen 0 und 255) abgegeben.
- Wenn Ihr LAN Gerät über DNS einen Namen ansprechbar ist, kann die Funktion resolve anstelle einer IP Adresse stehen.
- Keine terminierenden Nullzeichen.

Datentyp Ergebnis (Rückgabe)

- keine

TCP Telegramme versenden

Alle 2 Minuten soll ein TCP Telegramm an den Empfänger www.enertex.de an den Port 5555u16 vom Enertex® EibPC geschickt werden. Als Nutzendaten sollen die ersten 5 Bytes des Strings „Ich lebe noch“ geschickt werden.

(Die Socketverbindung wurde bereits mit connecttcp geöffnet und besteht.)

Umsetzung im Anwenderprogramm:

```
String =$Ich lebe noch$
if cycle(2,00) then sendtcparray(5555u16,resolve($www.enertex.de$), String,
size(String),5u16)  endif
```

Md5sum

Definition

- Funktion md5sum(*string*)

Argumente

- Argument *string* beliebiger Länge

Wirkung

- Es wird die MD5-Summe des Strings berechnet. Das Ergebnis wird als String zurückgegeben.
- **Ergebnis (Rückgabe)**
- Datentyp cXXXXX mit der gleichen Stringlänge, wie der Ausgangsstring.

Beispiel ping

Der Wert der MD5-Summe des String $fdzehkdkhfckdhk%%$ soll ermittelt werden

```
string=$fdzehkdkhfckdhk%%$
md5=md5sum(string)
```

Ping

Definition

- Funktion ping(*IP*)

Argumente

- Argument *IP* vom Datentyp u32 (Adresse des Empfängers in der üblichen Schreibweise z.B. 192.168.22.100)

Wirkung

- Ausführung des Ping Befehls
- Die Funktion liefert den Status ihrer Verarbeitung zurück:
  - bei Erfolg = 0
  - bei andauernder Verarbeitung = 1
  - bei Fehler bzw. nicht gefunden = 2
- **Ergebnis (Rückgabe)**
- Datentyp u08
  (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Beispiel ping

Die Adresse www.enertex.de soll kurz nach dem Systemstart angepingt werden. Bei Erfolg soll auf die GA 2/2/2 geschrieben werden.

```
A=3
If after(systemstart(),1u64) then IP=resolve($www.enertex.de$) endif
If after(systemstart(),10u64) then a=ping(IP) endif
if a==0 then write('2/2/2'c14,$gefunden$c14) endif
```

Hinweis:

Beachten Sie, dass wohl resolve als auch ping asynchrone Funktionen sind. Dies bedeutet, dass bei etwa ein ineinander setzen von diesen Funktionen zu unerwarteten Verhalten führen kann:

```
If after(systemstart(),10u64) then a=ping(resolve($www.enertex.de$)) endif
```

Wenn die Funktion ping und resolve angestoßen werden und unterschiedlich abgearbeitet werden, ist das Verhalten der Abfragen und damit der Wert von *a* u.U. nicht reproduzierbar.

Resolve

Definition

- Funktion resolve(*hostname*)

Argumente

- Ein Argument *hostname* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge

Wirkung

- Die Funktion ermittelt die IP-Adresse des angegebenen Hostenamens
- Im Fehlerfall wird 0u32 zurück gegeben.

Datentyp Ergebnis (Rückgabe)

- Datentyp u32

  (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Beispiel resolve

Der Hostname enertex.de soll aufgelöst werden.

Umsetzung im Anwenderprogramm:

```
hostname=$www.enertex.de$
IP=resolve(hostname)
```

Sendmail

Um die Funktion sendmail nutzen zu können, muss die Grundkonfiguration des E-Mail vorgenommen werden (Seite 145).

Definition

- Funktion sendmail(*Empfängeradresse, Betreff, Nachricht*)

Argumente

- 3 Argumente vom Datentyp c1400 bzw. einer selbst definierten Stringlänge

Wirkung

- Es wird an die *Empfängeradresse* (Zeichenkette) eine E-Mail mit dem *Betreff* und der *Nachricht* verschickt.
- Alle Zeichenketten werden auf 1400 Zeichen begrenzt.
- Ein Zeilenumbruch wird durch Einfügen der beiden Zeichen '\n' generiert.
- Rückgabewert: 0 = E-Mail erfolgreich versendet
  1 = in Bearbeitung
  2 = Fehler

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

  (Der Rückgabewert erfolgt asynchron zur Hauptverarbeitungsschleife - siehe S.129)

Hinweis:

Die Funktion sendmail sendet reine Textmails. Für das Verschicken von html-Mails steht sendhtml-mail zur Verfügung.

Beispiel: sendmail

Jeden Montag um 8.00 Uhr soll eine E-Mail an eibpc@enertex.de versendet werden.
Der Betreff ist “EibPC“ und der Nachrichtentext “alles prima“ in der ersten und „mit dem EibPC“ in der zweiten Zeile.

Umsetzung:

```
email=$eibpc@enertex.de$
betreff=$EibPC$
nachricht=$alles prima\nmit dem EibPC$
if wtime(08,00,00,MONTAG) then sendmail(email, betreff, nachricht) endif
```

Das geht auch mit zwei oder mehr Empfängern

```
email=$eibpc@enertex.de, knxrouter@enertex.de$
```

Sendhtmlmail

Um die Funktion sendhtmlmail nutzen zu können, muss die Grundkonfiguration des E-Mail vorgenommen werden (Seite 145).

Definition

- Funktion sendhtmlmail(*Empfängeradresse, Betreff, Nachricht*)

Argumente

- Drei Argumente vom Datentyp c1400 bzw. einer selbst definierten Stringlänge

Wirkung

- Es wird an die *Empfängeradresse* (Zeichenkette) eine E-Mail mit dem *Betreff* und der *Nachricht* verschickt.
- Alle Zeichenketten werden auf 1400 Zeichen begrenzt.
- Ein Zeilenumbruch wird durch Einfügen der beiden Zeichen '\n' generiert.
- Rückgabewert: 0 = E-Mail erfolgreich versendet
  1 = in Bearbeitung
  2 = Fehler

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Beispiel: sendmail

Jeden Montag um 8.00 Uhr soll eine E-Mail an eibpc@enertex.de versendet werden.
Der Betreff ist “EibPC“ und der Nachrichtentext “alles prima“ in der ersten und „mit dem EibPC“ in der zweiten Zeile.

Umsetzung im Anwenderprogramm

```
email=$eibpc@enertex.de$
betreff=$EibPC$
nachricht=$<html><head><meta name="qrichtext" content="1" /></head><body style="font-size:11pt;font-family:Sans Serif"> <p><span style="font-weight:600">alles prima, </span></p> <p>mit dem EibPC</p> </body></html>$
if wtime(08,00,00,MONTAG) then sendhtmlmail(email, betreff, nachricht) endif
```

Hinweis:

Die Funktion sendhtmlmail sendet HTML-Mails. Für das Verschicken von rein textbasierten Mails steht sendlmail zur Verfügung.

VPN Server

Der Enertex ® EibPC kann als VPN-Server und VPN-Router in Ihrem LAN betrieben werden. Um diese Funktionalität nutzen zu können, benötigen Sie einen gültigen Freischaltcode für die Option NP.

Im Home-Router muss für den UDP Port 1194 eine Weiterleitung zum EibPC eingerichtet werden.

Mit dem Enertex® EibPC kann der VPN Dienst direkt gestartet und gestoppt werden.

Eine Zugriffsverwaltung von verschiedenen Benutzern ist möglich.

Mit dem Eingabedialog unter Option-VPN Konfiguration wird der EibPC für VPN konfiguriert (Abb. 123).

Folgende Angaben sind konfigurierbar:

- die Netzwerkadresse des VPNs
- die Netzwerkadresse des LANs in dem sich der Enertex® EibPC befindet
- die IP-Adressen, die im entfernten LAN erreichbar sein sollen
- Benutzerverwaltung

Zusätzlich ist es nötig, Schlüssel- und Zertifikationsdateien für das VPN zu erstellen. Diese sind einmalig und werden vom Enertex® EibPC generiert. Unter www.youtube.com, Stichwort EibPC, finden Sie ein Demovideo mit weiteren Informationen.

Die Bibliothek EnertexVPN.lib vereinfacht die Anwendung der VPN Funktionen.

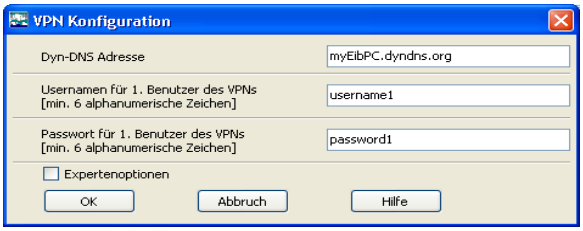


Abbildung 123: Konfigurationsdialog für VPN

Startvpn

Über den Info-Button im EibStudio® kann ausgelesen werden, ob der VPN-Dienst läuft und welche Benutzer freigeschaltet sind.

Definition

- Funktion startvpn()

Argumente

- keine

Wirkung

- Startet den VPN Dienst im Enertex® EibPC. Zuvor muss das VPN im Anwendungsprogramm mit dem EibStudio® konfiguriert werden.
- Nach einem Neustart ist das VPN standardmäßig gestoppt. Deshalb sollte der Dienst mit if systemstart() gestartet werden (siehe Beispiel).
- Für alle eingerichteten Benutzer ist VPN nach diesem Funktionsaufruf sofort geöffnet (um für einen Benutzer VPN zu öffnen vgl. openvpnuser()).
- Beim Überspielen eines neuen Anwendungsprogramms an den EibPC bleibt VPN geöffnet. Ein empfohlener, zusätzlicher Aufruf von startvpn() unterbricht eine bestehende Verbindung nicht. Nur bei einem Neustart wird der Dienst gestoppt.
- Über den Info-Button im EibStudio® kann ausgelesen werden, ob der VPN-Dienst läuft und welche Benutzer freigeschaltet sind.

Datentyp Ergebnis (Rückgabe)

- keine

Getvpnusers

Definition

- Funktion getvpnusers()

Argumente

- keine

Wirkung

- Liefert eine Liste mit derzeit aktiven VPN-Usern

Datentyp Ergebnis (Rückgabe)

- keine

Hinweis: In der EnertexVPN.lib finden sich Funktionen, die das Handling von VPN mit dem Enertex® EibPC einfach gestaltett

Stopvpn

Definition

- Funktion stopvpn()

Argumente

- keine

Wirkung

- Stoppt den VPN Dienst im Enertex® EibPC.
- Nach einem Neustart ist das VPN standardmäßig gestoppt.
- Für alle eingerichteten Benutzer ist VPN nach diesem Funktionsaufruf sofort geschlossen (um für einen Benutzer VPN zu öffnen vgl. openvpnuser()).
- Über den Info-Button im EibStudio® kann ausgelesen werden, ob der VPN-Dienst läuft und welche Benutzer freigeschaltet sind.

Datentyp Ergebnis (Rückgabe)

- keine

Openvpnuser

Definition

- Funktion openvpnuser(*Benutzername*)

Argumente

- Ein Argument *Benutzername* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge

Wirkung

- Öffnet einen VPN Benutzerzugang. Der Zugang wird erst aktiv, wenn die Funktion startvpn() aufgerufen wurde.
- Nach einem Neustart bleibt der Benutzerzugang aktiv, allerdings muss der VPN Dienst mit startvpn() neu gestartet werden.
- Über den Info-Button im EibStudio® kann ausgelesen werden, ob der VPN-Dienst läuft und welche Benutzer freigeschaltet sind.

Datentyp Ergebnis (Rückgabe)

- keine

Closevpnuser

Definition

- Funktion closevpnuser(*Benutzername*)

Argumente

- Ein Argument *Benutzername* vom Datentyp c1400 bzw. einer selbst definierten Stringlänge

Wirkung

- Schließt einen VPN Benutzerzugang. Der Zugang wird inaktiv unabhängig davon, ob der VPN-Dienst ausgeführt wird oder nicht.
- Nach einem Neustart bleibt der Benutzerzugang aktiv, allerdings muss der VPN Dienst mit startvpn() neu gestartet werden.
- Über den Info-Button im EibStudio® kann ausgelesen werden, ob der VPN-Dienst läuft und welche Benutzer freigeschaltet sind.

Datentyp Ergebnis (Rückgabe)

- keine

Hinweis:

Wenn ein Benutzerzugang mit closevpnuser geschlossen wird, so ist eine noch bestehende Verbindung davon nicht betroffen. Erst nach Abmelden der Verbindung – oder einen Stop und Start des VPN Service - ist ein geschlossener Zugang tatsächlich gesperrt.

Beispiel

Über eine Gruppenaddresse 1/1/1 soll bei einem EIN-Telegramm ein bereits angelegter VPN Teilnehmer „USER1“ der Zugang frei geschaltet werden. Bei einem AUS Telegramm, soll dieser Zugang wieder geschlossen werden. Über die Adresse 1/1/2 soll ein zweiter Anwender „USER2“ frei geschaltet werden. Über die Adresse 1/2/3 soll bei einem EIN Telegramm der gesamte VPN gesperrt werden.

```
[EibPC]
if after(systemstart(),500u64) then startvpn() endif
if "OpenUser1-1/1/1"==EIN then openvpnuser($User1$) else closevpnuser($User1$) endif
if "OpenUser2-1/1/2"==EIN then openvpnuser($User2$) else closevpnuser($User2$) endif
if "StopVPN-1/1/3"==EIN then stopvpn() endif
```

FTP-Funktionen

FTP Transfer zum beliebigen Datenloggen.

Der FTP Transfer schreibt Dateien auf einen externen FTP Server, wobei die maximale Dateigröße 64 kB ist.

Dazu können verschiedene Handles angelegt werden, die ihrerseits per Warteschlange gepuffert bis zu 64 kB große Dateien auf dem Server anlegen. Die Dateien werden per Timeout auch früher (und dann ggf. weniger Bytes) oder per flushftp() vom User initiiert geschrieben.

Die Dateien werden von der Firmware automatisch nach Datum und Uhrzeit benannt.

Als Eingabe können Strings geschrieben werden. Die Datei ist demnach im ASCII Format und werden mit der Funktion sendftp() s. 257 in die Warteschlange geschrieben.

Dabei wird am Ende der Datenübermittlung von sendftp ein LF-CR (Zeilenumbruch für Windows geeignet) eingefügt. Ein Aufruf von sendftp kann mehrere Teilstrings übergeben, aber nicht mehr als 1400 Bytes insgesamt übernehmen. Es können nicht mehr als vier Handles definiert werden. Auf S. 114 finden Sie ein ausführliches Beispiel. Nicht zu verwechseln ist dies mit der zyklischen Auslagerung der KNX™ Telegramme (vgl. S 150).

Ftpconfig

Definition

- Funktion ftpconfig(server,user,password,path,timeout)

Argumente

- Argument *server* vom Datentyp c1400
- Argument *user* vom Datentyp c1400
- Argument *password* vom Datentyp c1400
- Argument *path* vom Datentyp c1400
- Argument *timeout* vom Datentyp u32 in Sekunden

Wirkung

- Konfiguration eines FTP-Servers
- Aktualisierung der Abhängigkeiten bei Wertänderung oder während des möglichen Aufrufs der Systemstart-Funktion.
- Der FTP Transfer schreibt Dateien auf einen externen FTP Server, wobei die maximale Dateigröße 64 kB ist. Dazu können verschiedene Handles angelegt werden, die ihrerseits per Warteschlange gepuffert bis zu 64 kB große Dateien auf dem Server anlegen. Die Dateien werden per Timeout auch früher (und dann ggf. weniger Bytes) oder per flushftp() vom User initiiert geschrieben. Die Dateien werden von der Firmware automatisch nach Datum und Uhrzeit benannt.
- Es können nicht mehr als vier Handles definiert werden.

Datentyp Ergebnis (Rückgabe)

- Im Fehlerfall = 0
- bei Erfolg wird eine Handle-Nummer zwischen 1 und 4 zurückgegeben

Sendftp

Definition

- Funktion sendftp(handle,data1,[data2],[...])

Argumente

- Argument *handle* vom Datentyp u08
- Argument *data[x]* von beliebigem Datentyp, maximal 1400 Bytes

Wirkung

- Beliebige Daten werden in die Warteschlange des Handles geschrieben.
- Die Zuweisung erfolgt asynchron.

Datentyp Ergebnis (Rückgabe)

- bei Erfolg = 0
- Im Fehlerfall = 1

Ftpstate

Definition

- Funktion ftpstate(handle)

Argumente

- Argument *handle* vom Datentyp u08

Wirkung

- Gibt Informationen über den Status der FTP-Konfiguration zurück.

Datentyp Ergebnis (Rückgabe)

- u08
- Konfiguriert / fehlerfrei = 0
- Letzte Übertragung fehlerfrei = 1
- Server nicht erreichbar = 2
- Passwort/User nicht erlaubt = 3
- Fehler Verzeichnis nicht existent und konnte nicht angelegt werde = 4
- Warteschlange Überlauf, wenn vorher Fehler = 5
- Handle nicht definiert = 6

Ftptimeout

Definition

- Funktion ftptimeout(handle)

Argumente

- Argument *handle* vom Datentyp u08

Wirkung

- Gibt die bereits verstrichene Zeit seit dem letzten Transfer in Sekunden zurück

Datentyp Ergebnis (Rückgabe)

- u32

Ftpbuffer

Definition

- Funktion ftpbuffer(handle)

Argumente

- Argument *handle* vom Datentyp u08

Wirkung

- Gibt den Füllstand der Warteschlange des Transfers aus.

Datentyp Ergebnis (Rückgabe)

- u16

Flushftp

Definition

- Funktion flushftp(handle)

Argumente

- Argument *handle* vom Datentyp u08

Wirkung

- Daten manuell auf den FTP-Server schreiben

Datentyp Ergebnis (Rückgabe)

- Erfolg = 0
- Server nicht erreichbar = 1
- Fehler beim Hochladen der Datei = 2
- Passwort/User nicht erlaubt = 3
- Fehler Verzeichnis nicht existent und konnte nicht angelegt werde = 4
- Übertragung wird eben durchgeführt (asynchrone Aktualisierung) = 5

Webserverfunktionen

Um den Webserver des EibPC nutzen zu können, müssen Sie die Option NP im Enertex® EibPC freischalten Der Freischaltcode ist immer an die Seriennummer des Gerät gebunden und ist nicht übertragbar auf andere Geräte.

Beispiele zur Anwendung finden Sie mit der EnertexWebV2.lib

Button (Webbutton)

Definition

- Funktion button(*ID*)
- identisch zur Funktion webbutton früherer Releases

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.

Wirkung

- Bei „Betätigen“ bzw. Drücken der Schaltfläche eines Webbuttons (z.B. *button* oder *shifter*) mit der *ID* (Seite 274 ff.) geht die Funktion für einen Verarbeitungszyklus auf einen Wert ungleich 0. In allen anderen Fällen ist der Rückgabewert 0.
- Bei einem *button*-Element wechselt bei Betätigung die Rückgabe auf 1.
- Bei einem *shifter*-Element wechselt bei Betätigung die Rückgabe auf 1, 2, 3 oder 4 (u08), je nachdem, welches Schaltelement des Webbuttons betätigt wird. Die Nummern beziehen sich dabei auf die Anordnung der Schaltelemente von links nach rechts

Datentyp Ergebnis (Rückgabe)

- Typ u08, Werte 0,1,2,3,4.

Beispiel: Webserver

Ein ausführliches Beispiel finden Sie in der Einführung auf Seite 75ff.

Chart (Webchart)

Definition

- Funktion chart(*ID, Var, X1, X2*)
- kompatibel zur Funktion webchart

Argumente

- Argumente *ID, Var* vom Datentyp u08
- Argumente *X1, X2* Datentyp c14

Wirkung

- Die Funktion spricht das XY-Diagramm *chart* an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
- Bei Aufruf der Funktion wird die XY-Darstellung des Wertes *Var* aktiviert. Es können Werte aus dem Bereich 1..30 dargestellt werden. 0 bedeutet keine Darstellung, Werte größer als 30 sind unzulässig und werden wie 0 interpretiert. Bei jedem Aufruf der Funktion werden zunächst die Werte von links beginnend dargestellt. Wenn nach 47 Aufrufen das Ende erreicht ist, werden die Werte nach links geschoben.
- Die X-Achsenberschriftung wird mit den Argumenten *X1, X2* (Datentyp c14) vorgegeben.

Datentyp Ergebnis (Rückgabe)

- u08 (interner Zustand des Webcharts).

Beispiel Prozentwert Anzeigen

An einer XY-Grafik im Webserver (*chart*-Element) soll ein Prozentwert angezeigt werden

Umsetzung im Anwenderprogramm:

```
[WebServer]
chart(ChartWebID)[$0%$,$50%$,$100%$]
[EibPC]
Prozentwert='1/3/5'u08
ChartWebID=0
if stime(0) then\\
webchart(ChartWebID,convert(convert(Prozentwert,0f32)/8.5f32,0), $jetzt$c14,$-47min$c14)
endif
```

Display (Webdisplay)

Definition

- Funktion webdisplay(*ID, Text, Icon, State, TextStil,[Mbutton]*)

Argumente

- Argumente *ID, Icon, State, Textstil* und *Mbutton* vom Datentyp u08
- Argument *Text* beliebiger Datentyp

Wirkung

- Die Funktion spricht den (*button* oder *shifter*) an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
- Mit Hilfe des optionalen Arguments *Mbutton* kann die angezeigte Auswahl des Dropdown-menüs verändert werden.
- Bei Aufruf der Funktion wird das Icon des Webelements mit der *ID* auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Das Argument *Text* bezeichnet eine beliebige Variable, deren in einen String gewandelter Wert auf die variable Textzeile des Webelements ausgegeben wird.
- Jedes Icon hat mindestens die Zustände ACTIVE (==1), INACTIVE (==2), DARKRED (==0) und BRIGHTRED (==9). Einer dieser Zustände kann im Argument *State* übergeben werden. Eine Übersicht über die möglichen Zustände gibt Tabelle 5 (Seite 301).
- Der auszugebende Text kann in den Stilen GREY (==0), GREEN (==1), BLINKRED(==2) und BLINKBLUE (==3) dargestellt werden.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel Uhrzeit anzeigen

An einem *button*-Element soll die Uhrzeit angezeigt werden.

Umsetzung im Anwenderprogramm:

```
[WebServer]
button(ClockWebID)[CLOCK]$Uhrzeit$
[EibPC]
ClockWebID=0
if stime(0) then webdisplay(ClockWebID,settime(),CLOCK,INACTIVE,GREY) endif
```

Hinweis:

1. Der Rückgabetyp von *settime*() ist t24. Er wird in diesem Fall in einen String der lesbaren Form „Fr. 12:33:55“ gewandelt.
2. Sie können auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Beachten Sie, dass der Webserver dabei die Variable statisch auswertet. Wenn sich also die Variable *ClockWebID* während der Laufzeit ändert, bezieht sich der Index immer noch auf den Anfangswert 0.

Getslider

Definition

- Funktion getslider(*ID*)

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.

Wirkung

- Die Funktion spricht den *slider* an und gibt dessen Stellung (0 bis 255) zurück. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Getpslider

Definition

- Funktion getpslider(*ID, PageID)*

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.
- Argument *PageID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.

Wirkung

- Die Funktion spricht den seitenbezogenen *pslider* an und gibt dessen Stellung (0 bis 255) zurück. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Geteslider

Definition

- Funktion geteslider(*ID*)

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.

Wirkung

- Die Funktion spricht den *eslider* an und gibt dessen Stellung (0 bis 255) zurück. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

Getpeslider

Definition

- Funktion getpeslider(*ID, PageID)*

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.
- Argument *PageID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.

Wirkung

- Die Funktion spricht den seitenbezogenen *peslider* an und gibt dessen Stellung (0 bis 255) zurück. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

link

Definition

- Funktion link(*ID, Text, Icon, Iconzustand, PageID, Website*)

Argumente

- Argument *ID* vom Datentyp u08.
- Argument *Text* beliebiger Datentyp
- Argument *Icon* vom Datentyp u08
- Argument *Iconzustand* vom Datentyp u08
- Argument *PageID* vom Datentyp u08
- Argument *Website* vom Datentyp c1400

Wirkung

- Die Funktion spricht den (*link*) an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.
- Bei Aufruf der Funktion wird das Icon des Webelements mit der *ID* auf der Webseite mit der *PageID* auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Das Argument *Text* bezeichnet eine beliebige Variable, deren in einen String gewandelter Wert auf die variable Textzeile des Webelements ausgegeben wird.
- Im Argument *Website* (http-Adresse (inkl. Pfad und führenden http://) des Ziels) wird die neue URL als Sprungziel angegeben.

  Aus Kompatibilitätsgründen kann der Weberver nur die ersten 479 Zeichen des Links auswerten. Längere Links werden abgeschnitten.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel

Ein Beispiel finden sie auf Seite 267.

Mbutton

Definition

- Funktion mbutton(*ID, Auswahl*)

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.
- Argument *Auswahl* vom Datentyp u08.

Wirkung

- Bei „Betätigen“ bzw. Drücken der Schaltfläche eines Multibuttons und der gegebenen Auswahl mit Index *Auswahl* (z.B. *mbutton* oder *mshifter*) mit der *ID* (Seite 274 ff.) geht die Funktion für einen Verarbeitungszyklus auf den Wert *Auswahl*. *Auswahl* ist dabei ein Wert 1, 2 ... 254 und bezeichnet die Stelle im Auswahlfeld, an welcher das Auswahlelement steht. Wenn das Auswahlfeld betätigt wurde, ist der Rückgabewert 255. In allen anderen Fällen ist die Rückgabe 0.
- Bei einem *mbutton*-Element wechselt bei Betätigung die Rückgabe auf 1.
- Bei einem *mshifter*-Element wechselt bei Betätigung die Rückgabe auf 1, 2, 3 oder 4 (u08), je nachdem, welches Schaltelement des Webbuttons betätigt wird. Die Nummern beziehen sich dabei auf die Anordnung der Schaltelemente von links nach rechts.

Datentyp Ergebnis (Rückgabe)

- Typ u08.

Beispiel: S. 86

Mchart

Definition

- Funktion mchart(*ID, VarX, VarY, Index*)

Argumente

- Argumente *ID, Index* vom Datentyp u08
- Argumente *VarX, VarY* vom Datentyp f16

Wirkung

- Die Funktion spricht das Element *mchart* mit der *ID* an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
- Ein *mchart* stellt vier verschiedene Graphen dar. *Index* (0,1,2,3) gibt an, welcher Graph angesprochen wird.
- Es werden 48 Werte gespeichert. Wenn mehr als 48 Werte in den selben *Index* von mchart gespeichert werden, fällt der an erste Stelle gespeicherte Wert weg.
- Die Einordnung der Werte im Graphenverlauf erfolgt durch die Vorgabe der Wertepaare.
- Die Achsenbeschriftung wird automatisch generiert.

Datentyp Ergebnis (Rückgabe)

- u08 (interner Zustand).

Mpchart

Definition

- Funktion mpchart(*ID, VarX, VarY, Index,PageID*)

Argumente

- Argumente *ID, PageID, Index* vom Datentyp u08
- Argumente *VarX, VarY* Datentyp f16

Wirkung

- Die Funktion spricht das seitenbezogene Element *mpchart* mit der *ID* auf der Webseite mit der *PageID* an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
- Ein *mpchart* stellt vier verschiedene Graphen dar. *Index* (0,1,2,3) gibt an, welcher Graph angesprochen wird.
- Es werden 48 Werte gespeichert. Wenn mehr als 48 Werte in den selben *Index* von mpchart gespeichert werden, fällt der an erste Stelle gespeicherte Wert weg.
- Die Einordnung der Werte im Graphenverlauf erfolgt durch die Vorgabe der Wertepaare.
- Die Achsenbeschriftung wird automatisch generiert.

Datentyp Ergebnis (Rückgabe)

- u08 (interner Zustand)

Mpbutton

Definition

- Funktion mpbutton(*ID, Auswahl,PageID*)

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.
- Argument *PageID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.
- Argument *Auswahl* vom Datentyp u08.

Wirkung

- Bei „Betätigen“ bzw. Drücken der Schaltfläche eines seitenbezogenen Multibuttons und der gegebenen Auswahl mit Index *Auswahl* (z.B. *mpbutton* oder *mpshifter*) mit der *ID* (Seite 274 ff.) auf der Webseite mit der *PageID* geht die Funktion für einen Verarbeitungszyklus auf den Wert *Auswahl*. *Auswahl* ist dabei ein Wert 1, 2 ... 254 und bezeichnet die Stelle im Auswahlfeld, an welcher das Auswahlelement steht. Wenn das Auswahlfeld betätigt wurde, ist der Rückgabewert 255. In allen anderen Fällen ist die Rückgabe 0.
- Bei einem *mpbutton*-Element wechselt bei Betätigung die Rückgabe auf 1.
- Bei einem *mpshifter*-Element wechselt bei Betätigung die Rückgabe auf 1, 2, 3 oder 4 (u08), je nachdem, welches Schaltelement des Webbuttons betätigt wird. Die Nummern beziehen sich dabei auf die Anordnung der Schaltelemente von links nach rechts.

Datentyp Ergebnis (Rückgabe)

- u08 (interner Zustand).

Pdisplay

Definition

- Funktion pdisplay(*ID, Text, Icon, State, TextStil, PageID, [Mbutton]*)

Argumente

- Argument *ID* vom Datentyp u08.
- Argument *PageID* vom Datentyp u08.
- Argumente *Icon, State* und *Textstil* vom Datentyp u08
- Argument *Text* beliebiger Datentyp

Wirkung

- Die Funktion spricht den (*pbutton* oder *pshifter*) an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.
- Mit Hilfe des optionalen Arguments *Mbutton* kann die angezeigte Auswahl des Dropdown-menüs verändert werden.
- Bei der Funktion *plink* gibt dieses Argument den Sprungindex an.
- Bei Aufruf der Funktion wird das Icon des Webelements mit der *ID* auf der Webseite mit der *PageID* auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zu-dem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Das Argument *Text* bezeichnet eine beliebige Variable, deren in einen String gewandelter Wert auf die variable Textzeile des Webelements ausgegeben wird.
- Bei der Funktion *link* gibt dieses Argument den Sprungindex an.
- Jedes Icon hat mindestens die Zustände ACTIVE (==1), INACTIVE (==2), DARKRED (==0) und BRIGHTRED (==9). Einer dieser Zustände kann im Argument *State* übergeben werden. Eine Übersicht über die möglichen Zustände gibt Tabelle 5 (Seite 301).
- Der auszugebende Text kann in den Stilen GREY (==0), GREEN (==1), BLINKRED(==2) und BLINKBLUE (==3) dargestellt werden.

Datentyp Ergebnis (Rückgabe)

- keine

Pchart

Definition

- Funktion pchart(*ID, Var, X1, X2, PageID*)

Argumente

- Argumente *ID, Var, PageID* vom Datentyp u08
- Argumente *X1, X2* Datentyp c14

Wirkung

- Die Funktion spricht das XY-Diagramm *chart* an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.
- Bei Aufruf der Funktion wird die XY-Darstellung des Wertes *Var* aktiviert. Es können Werte aus dem Bereich 1..30 dargestellt werden. 0 bedeutet keine Darstellung, Werte größer als 30 sind unzulässig und werden wie 0 interpretiert. Bei jedem Aufruf der Funktion werden zunächst die Werte von links beginnend dargestellt. Wenn nach 47 Aufrufen das Ende er-reicht ist, werden die Werte nach links geschoben.
- Die X-Achsenberschriftung wird mit den Argumenten *X1, X2* (Datentyp c14) vorgegeben.

Datentyp Ergebnis (Rückgabe)

- u08 (interner Zustand des Webcharts).

Pbutton

Definition

- Funktion pbutton(*ID,PageID*)

Argumente

- Argument *ID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.
- Argument *PageID* vom Datentyp u08. Dieses Argument darf sich zur Laufzeit des Programms nicht ändern.

Wirkung

- Bei „Betätigen“ bzw. Drücken der Schaltfläche eines seitenbezogenen Buttons (z.B. *pbutton* oder *pshifter*) mit der *ID* (Seite 274 ff.) auf der Webseite mit der *PageID* geht die Funktion für einen Verarbeitungszyklus auf einen Wert ungleich 0. In allen anderen Fällen ist der Rückgabewert 0.
- Bei einem *pbutton*-Element wechselt bei Betätigung die Rückgabe auf 1.
- Bei einem *pshifter*-Element wechselt bei Betätigung die Rückgabe auf 1, 2, 3 oder 4 (u08), je nachdem, welches Schaltelement des Webbuttons betätigt wird. Die Nummern beziehen sich dabei auf die Anordnung der Schaltelemente von links nach rechts.

Datentyp Ergebnis (Rückgabe)

- Typ u08, Werte 0,1,2,3,4.

Picture

Definition

- Funktion picture(*ID, Beschriftung, PageID, www-LINK*)

Argumente

- Argument *ID* vom Datentyp u08
- Argument *Beschriftung* beliebiger Datentyp
- Argument *PageID* vom Datentyp u08
- Argument *www-LINK* vom Datentyp c1400

Wirkung

- Die Funktion ändert das Element (*picture*). Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.
- Das Argument *Text* bezeichnet eine beliebige Variable, deren in einen String gewandelter Wert auf die variable Textzeile des Webelements ausgegeben wird.
- www-LINK: Gültig WWW-Adresse (optional mit führenden http://) zur externen Grafik als neues Sprungziel.

  Aus Kompatibilitätsgründen kann der Weberver nur die ersten 383 Zeichen des Links auswerten. Längere Links werden abgeschnitten.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel

Ein Beispiel finden sie auf Seite 267.

Plink

Definition

- Funktion plink(*ID, Text, Icon, Iconzustand, PageID, PageSprungIndex*)

Argumente

- Argument *ID* vom Datentyp u08.
- Argument *Text* beliebiger Datentyp
- Argument *Icon* vom Datentyp u08
- Argument *Iconzustand* vom Datentyp u08
- Argument *PageID* vom Datentyp u08
- Argument *PageSprungIndex* vom Datentyp u08

Wirkung

- Die Funktion spricht den (*plink*) an. Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.
- Bei Aufruf der Funktion wird das Icon des Webelements mit der *ID* auf der Webseite mit der *PageID* auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Das Argument *Text* bezeichnet eine beliebige Variable, deren in einen String gewandelter Wert auf die Textzeile des Webelements ausgegeben wird.
- Im Argument *PageSprungIndex* wird die Seitennummer des neuen Sprungziels angegeben.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel

Dynamisches Verändern von Web-Links im Webserver

```
[WebServer]
page (1) [$Haus$,$OG$]
plink(2) [INFO] [3] $Zu Seite 3$
picture(3) [DOUBLE,ZOOMGRAF] ($Wetter$,
$http://eur.yimg.com/w/wcom/eur_germany_outlook_DE_DE_440_dmy_y.jpg$)
link(4) [BLIND]
[$http://eur.yimg.com/w/wcom/eur_germany_outlook_DE_DE_440_dmy_y.jpg$] $Mein Link$

page (2) [$Haus$,$Seite2$]
plink(2) [INFO] [3] $Zu Seite 3$

page (3) [$Haus$,$Seite3$]
plink(2) [WEATHER] [1] $Zu Seite 1$

[EibPC]
SprungZiel=3
if after(systemstart(),5000u64) then plink(2,$Doch zu Seite 2$,MONITOR,DISPLAY,
1,SprungZiel) endif

// Achtung: picture verwendet nur die ersten 479 Zeichen für den Link
if after(systemstart(),5000u64) then picture(3,$Neues
Wetter$,1,$http://eur.yimg.com/w/wcom/eur_satintl_440_dmy_y.jpg$) endif

// Achtung: link verwendet nur die ersten 479 Zeichen für den Link
if after(systemstart(),5000u64) then link(4,$Neuer
Link$,MONITOR,DISPLAY,1,$http://eur.yimg.com/w/wcom/eur_satintl_440_dmy_y.jpg$) endif
```

Setslider

Definition

- Funktion setslider(*ID*,*Value*,*Icon*, *State*)

Argumente

- Alle Argumente vom Datentyp u08

Wirkung

- Die Funktion spricht den *slider* an und stellt diesen auf den Wert von *Value.* Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
- Bei Aufruf der Funktion wird das Icon zudem auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Jedes Icon hat mindestens die Zustände ACTIVE (==1), INACTIVE (==2), DARKRED (==0) und BRIGHTRED (==9). Einer dieser Zustände kann im Argument *State* übergeben werden. Eine Übersicht über die möglichen Zustände gibt Tabelle 5 (Seite 301).

Datentyp Ergebnis (Rückgabe)

- keine

Setpslider

Definition

- Funktion setpslider(*ID, Value, Icon, State, PageID*)

Argumente

- Alle Argumente vom Datentyp u08

Wirkung

- Die Funktion spricht den seitenbezogenen *slider* an der *ID* auf der Seite *PageID* und stellt diesen auf den Wert von *Value.* Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.
- Bei Aufruf der Funktion wird das Icon zudem auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Jedes Icon hat mindestens die Zustände ACTIVE (==1), INACTIVE (==2), DARKRED (==0) und BRIGHTRED (==9). Einer dieser Zustände kann im Argument *State* übergeben werden. Eine Übersicht über die möglichen Zustände gibt Tabelle 5 (Seite 301).

Datentyp Ergebnis (Rückgabe)

- keine

Seteslider

Definition

- Funktion seteslider(*ID*,*Value*,*Icon*, *State*)

Argumente

- *ID, Icon, State, PageID* vom Datentyp u08
- *Value* vom Datentyp f32

Wirkung

- Die Funktion spricht den *eslider* an und stellt diesen auf den Wert von *Value.* Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
- Bei Aufruf der Funktion wird das Icon zudem auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Jedes Icon hat mindestens die Zustände ACTIVE (==1), INACTIVE (==2), DARKRED (==0) und BRIGHTRED (==9). Einer dieser Zustände kann im Argument *State* übergeben werden. Eine Übersicht über die möglichen Zustände gibt Tabelle 5 (Seite 301).

Datentyp Ergebnis (Rückgabe)

- keine

Setpeslider

Definition

- Funktion setpeslider(*ID, Value, Icon, State, PageID*)

Argumente

- *ID, Icon, State, PageID* vom Datentyp u08
- *Value* vom Datentyp f32

Wirkung

- Die Funktion spricht den seitenbezogenen *peslider* an der *ID* auf der Seite *PageID* und stellt diesen auf den Wert von *Value.* Wenn *ID* mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der *PageID* angesprochen.
- Bei Aufruf der Funktion wird das Icon zudem auf das Symbol mit der Nummer (Typ u08) *Icon* gesetzt. Mögliche Grafiken sind auf Seite 300 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 4 (Seite 301) aufgelistet.
- Jedes Icon hat mindestens die Zustände ACTIVE (==1), INACTIVE (==2), DARKRED (==0) und BRIGHTRED (==9). Einer dieser Zustände kann im Argument *State* übergeben werden. Eine Übersicht über die möglichen Zustände gibt Tabelle 5 (Seite 301).

Datentyp Ergebnis (Rückgabe)

- keine

Timebufferconfig

Definition

- Funktion timebufferconfig(*ChartBufferID, MemTyp, Laenge, DataTyp*)

Argumente

- *ID* vom Datentyp u08
- *MemTyp* Speichertyp, wobei „0“ Ringspeicher und „1“ einen linearer Speicher repräsentiert.
- *Laenge* der Daten im Puffer. Maximal 65535 Datensätze mit max. 4 Bytes Länge
- Der Speicher ist vom Datentyp *DataTyp* des Eingangsobjektes

- **Wirkung**
  - Es wird hier ein Wertepaarbuffer angelegt bzw. konfiguriert. Dabei kann mit dem Speichertyp festgelegt werden, ob dieser nach Befüllen mit den Werten vollläuft oder ob der jeweils älteste Wert verworfen wird.
  - ACHTUNG: Der EibPC hat einen RAM von 64MB, wovon ca. 40 MB maximal vom Anwender genutzt werden können.
    Um einen ordnungsgemäßen Betrieb zu garantieren, müssen die Buffer und mtimecharts so dimensioniert werden, damit der Speicher des EibPC nicht überlastet wird. Mit der Funktion können bis zu 255 Puffer zum Speichern von Verlaufswerte definiert werden. Dabei gilt für den benötigten Speicherbedarf = (Anzahl der Werte) * 12. Daher belegt z.B. ein Puffer mit 65000 Werten etwa 780 kB.
  - Sie können den Puffer auch jederzeit im Flash ablegen, sodass bei einem Neustart die Werte nicht verloren sind, siehe timebufferstore 270 und timebufferread 270.

Datentyp Ergebnis (Rückgabe)

- Werte: 0 Erfolg, 1 Fehler: maximale Anzahl von Zeitpuffern überschritten, 2 Fehler: Zeitpuffer bereits definiert

Ausführliches Beispiel: S. 119

Timebufferadd

Definition

- Funktion timebufferadd(*ChartBufferID, Daten*)

Argumente

- *ID* vom Datentyp u08
- *Daten* Wert (max 32 Bit), welcher in den Speicher am Ende eingefügt werden soll.

- **Wirkung**
  - Es wird hier ein Wertepaar in den Zeitpuffer am Ende eingefügt.

Datentyp Ergebnis (Rückgabe)

- 0 Erfolg, 1 Fehler

Ausführliches Beispiel: S. 119

Timebufferclear

Definition

- Funktion timebufferclear(*ChartBufferID*)

Argumente

- *ChartBufferID* vom Datentyp u08

- **Wirkung**
  - Den aktuellen Zeitpuffer löschen (im Speicher und ggf. auf den Flash, falls vorhanden)

Datentyp Ergebnis (Rückgabe)

- Füllstand des Zeitpuffers vom Datentyp u16

Beispiel

```
if systemstart() then timebufferclear(2) endif
```

Timebufferstore

Definition

- Funktion timebufferstore(*ChartBufferID*)

Argumente

- *ChartBufferID* vom Datentyp u08

- **Wirkung**
  - Es wird ein Buffer dauerhaft im Flash abgelegt.

Datentyp Ergebnis (Rückgabe)

- 0 Erfolg, 1 Fehler, 2 andauernde Verarbeitung

Ausführliches Beispiel: S. 119

Timebufferread

Definition

- Funktion timebufferread(*ChartBufferID*)

Argumente

- *ChartBufferID* vom Datentyp u08

- **Wirkung**
  - Es wird ein Buffer aus dem Flash gelesen.

Datentyp Ergebnis (Rückgabe)

- 0 Erfolg, 1 Fehler, 2 andauernde Verarbeitung, Datentyp u08

Ausführliches Beispiel: S. 119

Timebuffersize

Definition

- Funktion timebuffersize(*ChartBufferID*)

Argumente

- *ChartBufferID* vom Datentyp u08

- **Wirkung**
  - Den aktuellen Füllstand des Zeitpuffers ausgeben.

Datentyp Ergebnis (Rückgabe)

- Füllstand des Zeitpuffers vom Datentyp u16

Ausführliches Beispiel: S. 119

Timebuffervalue

Definition

- Funktion timebuffervalue(*ChartBufferID, utcZeit,Data, utcZeitWert)*

Argumente

- *ID* vom Datentyp u08
- *utcZeit* vom Datentyp u64, welcher der Zeitstempel angibt, der größer gleich dem Zeitpunkt des nächsten Datenpunkts in der Zeitreihe ist.
- *Data* Wert (max 32 Bit), welcher in den Speicher am Ende eingefügt werden soll. Die Funktion ändert beim Aufruf den Wert dieses Arguments auf den gespeicherten Wert zum gesuchten Zeitpunkt. Der Datentyp muss zum Datentyp des timebuffers (timebufferconfig) passen.
- *utcZeitWert* Die exakte Zeit des Aufnahmezeitpunkts des Wertes *Data.* . Die Funktion ändert beim Aufruf den Wert dieses Arguments auf den Wert

- **Wirkung**
  - Es wird ein Wertepaar im Zeitpuffer gesucht.

Datentyp Ergebnis (Rückgabe)

- 0 Erfolg, 1 Fehler, 2 andauernde Verarbeitung.

Vergleiche ausführliches Beispiel zum Thema Zeitpuffer auf : S. 119

Beispiel: Werte auslesen

Ein Timebuffer trägt f16-Datentypen und zeichnet seit 1.1.2016 Werte auf. Es soll der Wert im Zeitpuffer zum Zeitpunkt 12:00:00 am 2.1.2016 täglich um 9:30:00 ausgelesen werden. Wenn ein Wert im Puffer vorhanden ist, der plus oder minus eine Sekunde zu diesem Zeitpunkt mit timebufferadd in den Puffer geschrieben wurde, soll dieser auf die GA '1/2/3'f16 ausgegeben werden.

```
uBf=0
timebufferconfig(uBf,0,2500u16,0f16)
// gesuchter Zeitpunkt
uTime=utc($2016-01-02 12:00:00$)
fVal=0f16
uSampleTime=0u64
uRet=3

if htime(9,30,00) then {
  uRet=timebuffervalue(uBf,uTime,fVal,uSampleTime);
} endif
if uRet==0 then {
    if hysteresis(uSampleTime, uTime-1000u64,uTime+1000u64) then {
        write('1/2/3'f16, fVal) ;
    } endif
} endif
```

mtimechartpos

Definition

- Funktion mtimechartpos(TimeChartID,TimeChartID,ChartBuffer,StartPos,EndPos)

Argumente

- TimeChartID vom Datentyp u08
- TimeChartID Index des Charts (0..3)
- ChartBuffer Handle auf den Zeitpuffer, der vom Webelement angezeigt werden soll. Das Webelementmuss vorher entsprechend konfiguriert werden.
- StartPos Startposition der Anzeige
- EndPos Endposition der Anzeige

Wirkung

- Den darzustellenden Bereich eines Zeitpuffers für das Webelement vorgeben.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiele S. 119 bis 123

mtimechart

Definition

- Funktion mtimechart(TimeChartID,ChartIdx,ChartBuffer,StartZeit,EndZeit)

Argumente

- TimeChartID vom Datentyp u08
- ChartIdx-Index des Charts (0..3)
- ChartBuffer Handle auf den Zeitpuffer, der vom Webelement angezeigt werden soll. Das Webelement muss vorher entsprechend konfiguriert werden.
- StartZeit Startposition der Anzeige als UTC Zeit-Tics
- EndZeit Endposition der Anzeige als UTC Zeit-Tics

Wirkung

- Den darzustellenden Bereich eines Zeitpuffers für das Webelement vorgeben.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiele S. 119 bis 123

Webinput

Definition

- Funktion webinput(ID)

Argumente

- *ID* des Webinputelements vom Datentyp u08

Wirkung

- Liest die Daten des Webinputelementes aus. Dabei werden diese Daten in den Rückgabe-String geschrieben.
- Webinputelemente sind immer global.

Datentyp Ergebnis (Rückgabe)

- String c1400 als Ergebnis der Betätigung.

Beispiel: s. Beispiel weboutput S.273, S. 116)

Weboutput

Definition

- Funktion weboutput(ID,Data)

Argumente

- *ID* des Weboutputelements vom Datentyp u08
- *Data* Daten, die im Weboutputelement angezeigt werden

Wirkung

- Gibt Daten an das Weboutputelement aus.
- Weboutputelemente sind immer global.

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel: webinput, weboutput (vgl. auch S. 116)

```
WebServer]
page(1)[$Enertex$,$Webserver$]
webinput(1)[INFO] $Eingabe hier -> Ausgabe in Outputfeldern$
weboutput(2)[SINGLE,ICON]

[EibPC]
inputstring=webinput(1)
if change(inputstring) then weboutput(2,inputstring) endif
```

Webserverkonfiguration

Mit dem eingebauten Webserver können Daten und Automatisierungsabläufe visualisiert werden. Sie benötigen hierzu lediglich einen Webbrowser.

Mozilla Firefox auf dem Client

Der Webserver des Enertex® EibPC wurde erfolgreich mit den Webbrowsern Microsoft Internet Explorer 7 und Mozilla Firefox 3 getestet. Die beste Performance erreichen Sie mit dem Mozilla Firefox 3.

Das Webserverdesign

Der eingebaute Webserver ordnet fix wie in einem Schachbrettmuster seine Elemente an, die entweder eine vorgegebene einfache oder doppelte Länge aufweisen (vgl Abbildung 124). Es gibt dabei grundsätzlich die Möglichkeit, auch Felder freizulassen sowie Trennlinien einzufügen.

Das Design der Knöpfe ist vorgegeben

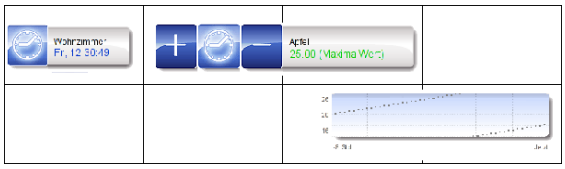


Abbildung 124: Schema Webserver

Die Anordnung und die Konfiguration der Webelemente werden in der Sektion [WebServer] vorgenommen. Durch die fixe Vorgabe von Icons, Längen und Variablen kann die Konfiguration ohne grafischen Aufwand erfolgen und auf sehr einfache Weise ein professionelles Web-Interface erstellt werden. Die Icons (Übersicht s. Seite 300) sind in einem einheitlichem Design erstellt. Neben den gezeigten Standard-Blau der Schaltflächen kann mit Hilfe des Designkommandos auf ein schwarzes Design umgeschaltet werden (vgl. 285), das z.B. für die Verwendung für Wandpanels oder Smartphones sehr geeignet ist. Auf S. 281 wird gezeigt, wie der Aufbau des Webseite so gestaltet werden kann, damit keine Freiräume in der Visualisierung entstehen, wenn Seiten mit Elementen unterschiedlicher Größe verwendet werden.

Sie können den Webserver mit einer einzigen Seite (wie in der Firmwareversion <1.200 vorgesehen) oder in der mehrseitigen Version konfigurieren.

Pro Seite können Sie maximal 60 Elemente konfigurieren und nutzen.

Navigation bei mehreren Seiten

Bei der Verwendung von mehreren Seiten, können sie jede Seite einer Gruppe zuordnen. Über den Listbox der Seitennavigation (vgl. Abbildung 125) können Sie die nach Gruppen geordneten Seiten auswählen.

Abbildung 125: Seitennavigation

Auch die Seitennavigation wird automatisch generiert. Dabei werden die Seiten über das page-Konfigurationskommando in der Sektion [WebServer] definiert. Wenn eine Seite mit diesem Kommando einer Gruppe zugeordnet wird, erscheint sie dann so geordnet in der Auswahlbox. Auf diese Weise können Gruppen wie „Keller“, „Erdgeschoß“ etc. generiert werden.

Die Schnellauswahl (Vor- bzw. Zurückknopf in Abbildung 125) ist durch die Reihenfolge der Definition gegeben.

Für die Anzeige gibt es folgende Elementgruppen, wobei diese wiederum globale oder lokale Elemente darstellen. Global bedeutet, das das Element auf verschiedenen Seiten verwendet werden kann, aber nur ein einziges mal erzeugt wurde. Ein Zugriff, Veränderung im Anwendungsprogramm des Elements ist daher für alle Seiten gleich.

Hingegen kann ein lokales Element nur in einer Seite verändert werden. Vom Design sind lokale und globale Elemente identisch. Lokale Elemente sind durch den Zusatz „p“ (page) gekennzeichnet

Benutzerverwaltung

Ab der Patch-Version 3.xxx ist eine seitenbezogene Benutzerveraltung des Webservers möglich. Jede Seite kann über Benutzer- und Passworteingabe gesichert werden. Dabei können mehrere Benutzer pro Seite zugelassen werden.

Jedem Benutzer kann ein Passwort zugewiesen werden, welches bei der ersten Definition des Benutzernamens angegeben werden muss.

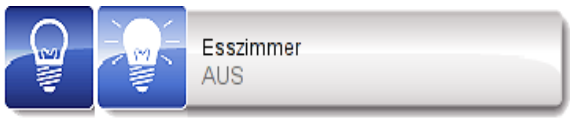
Beispiel

```
[WebServer]
page(1) [$Benutzerverwaltung$,$Seite 1$]
user $Michael$ [PasswortM]
user $Florian$ [PasswortF]
button(1) [INFO] $Seite 1$

page(2) [$Benutzerverwaltung$,$Seite 2$]
// Passwörter werden übernommen
user $Michael$
user $Florian$
button(1) [INFO] $Seite 2$

page(3) [$Benutzerverwaltung$,$Seite 3$]
// Diese Seite ist nur für Michael
// Passwort wird übernommen
user $Michael$
button(1) [INFO] $Seite 3$

page(4) [$Benutzerverwaltung$,$Seite 4$]
// Diese Seite ist nur für Stefanie
// Passwort muss mit angegeben werden, da dieser User auf dern vorherigen Seiten nicht genannt war
user $Stefanie$ [SGut]
button(1) [INFO] $Seite 4$

page(5) [$Benutzerverwaltung$,$Seite 5$]
// Alles User
button(1) [INFO] $Seite 5$
```

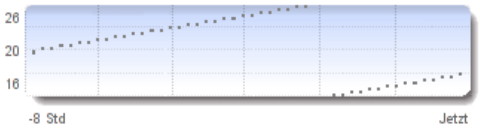
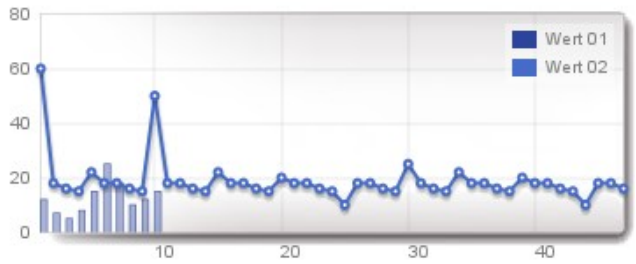
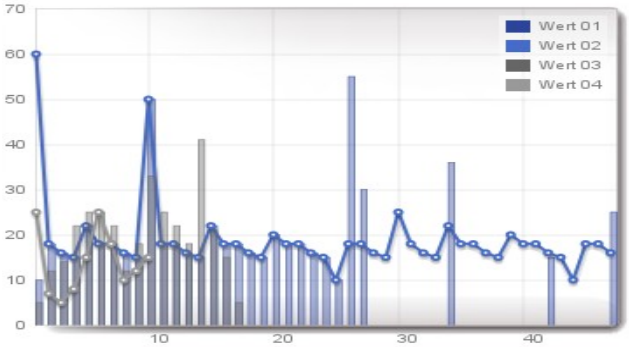
Webserverelemente

Für die Anzeige gibt es Elementgruppen, den W*ebbutton* und das W*ebdisplay*. Das Element b*utton* (Untergruppe von W*ebbutton*) ist dabei das einzige Element, das die einfache Breite aufweist. Schließlich gibt es noch Gestaltungselemente. Hier ist die Kopf- und Fußzeile (*header*) und Trennlinie (*line*) zu nennen (vgl. Tabelle 3). Neben den gezeigten Standard-Blau kann mit Hilfe des Designkommandos auf ein schwarzes Design umgeschaltet werden (vgl. 285).

Jeder Webbutton kann eine Grafik und eine Textzeile während der Laufzeit des Programms dynamisch verändern.

Gruppe	Element	Erläuterung
button		
	button, pbutton	 Die Grafik, die das eigentliche Bedienfeld darstellt, kann durch das Anwenderprogramm verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden, z. B. zum Anzeigen von Variablen.
	shifter, pshifter	 Die Grafik kann durch das Anwenderprogramm verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden.
	shifter, pshifter	 Die **rechte** Grafik kann durch das Anwenderprogramm verändert werden. Die linke Grafik kann nur bei der Konfiguration verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden.
	shifter, pshifter	 Die **mittlere** Grafik kann durch das Anwenderprogramm verändert werden. Die beiden äußeren Grafiken können nur bei der Konfiguration verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden.
	shifter	 Die **rechte** Grafik kann durch das Anwenderprogramm verändert werden. Die anderen Grafiken können nur bei der Konfiguration verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden.
mbutton		
	mbutton, mpbutton	 Die Grafik, die das eigentliche Bedienfeld darstellt, kann durch das Anwenderprogramm verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die aktive Auswahl kann durch das Anwenderprogramm verändert werden. Dabei muss dieses dann den Zustand der Grafik anpassen. In der zweiten Zeile kann kein Text angezeigt werden. Die Listbox kann maximal 254 Einträge verwalten. Bei Betätigung der Listbox wird ein Signal an das Anwendungsprogramm geschickt, das mit der Funktion mbutton (S. 261) bzw. mbbutton (S. 264) abgefragt werden kann.

Gruppe	Element	Erläuterung
Mit mshifter können Sie bis zu 254 verschiedene Aktionen in einen Knopf packen.	*mshifter, mpshifter*	 Die Grafik kann durch das Anwenderprogramm verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden. Die Listbox kann maximal 254 Einträge verwalten. Bei Betätigung der Listbox wird ein Signal an das Anwendungsprogramm geschickt, welches mit der Funktion mbutton (S. 261) bzw. mpbutton (S. 264) abgefragt werden kann.
	mshifter, mpshifter	 Die **rechte** Grafik kann durch das Anwenderprogramm verändert werden. Die linke Grafik kann nur bei der Konfiguration verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden. Die Listbox kann maximal 254 Einträge verwalten. Bei Betätigung der Listbox wird ein Signal an das Anwendungsprogramm geschickt, welches mit der Funktion mbutton (S. 261) bzw. mpbutton (S. 264) abgefragt werden kann.
	mshifter, mpshifter	 Die **mittlere** Grafik kann durch das Anwenderprogramm verändert werden. Die beiden äußeren Grafiken können nur bei der Konfiguration verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). Die zweite Zeile kann durch das Anwenderprogramm verändert werden. Die Listbox kann maximal 254 Einträge verwalten. Bei Betätigung der Listbox wird ein Signal an das Anwendungsprogramm geschickt, welches mit der Funktion mbutton (S. 261) bzw. mpbutton (S. 264) abgefragt werden kann.
	mshifter, mpshifter	 Die rechte Grafik kann durch das Anwenderprogramm verändert werden. Die anderen Grafiken können nur bei der Konfiguration verändert werden. Die erste Textzeile ist statisch (nur bei der Konfiguration veränderbar). In der zweiten Zeile kann kein Text angezeigt werden. Die Listbox kann maximal 254 Einträge verwalten. Bei Betätigung der Listbox wird ein Signal an das Anwendungsprogramm geschickt, welches mit der Funktion mbutton (S. 261) bzw. mpbutton (S. 264) abgefragt werden kann.
	Slider pslider	 Die Grafik und die Stellung des Sliders kann durch das Anwenderprogramm mit den Funktionen setslider bzw. setpslider verändert werden. Der Knopfdruck kann mit der Funktion mbutton (S. 261) bzw. mpbutton (S. 264) abgefragt werden.
	eslider peslider	Die Grafik und die Stellung des Sliders kann durch das Anwenderprogramm mit den Funktionen setslider bzw. setpslider verändert werden. Der Knopfdruck kann mit der Funktiion mbutton (S. 261) bzw. mpbutton (S. 264) abgefragt werden. Minimale und Maximale Stellung, sowie Inkrement des Sliders kann vorgegeben werden.

Gruppe	Element	Erläuterung
chart		
Mit mchart können Sie bis zu vier verschiedene Kurvenverläufe darstellen...	*chart, pchart*	 Dieses Element dient zum Anzeigen von Verläufen. Die Beschriftung der Y-Achse wird bei der Konfiguration vorgeben. Die X-Achsen Beschriftung kann durch das Anwenderprogramm verändert werden. Bei Aufruf der Funktion webdisplay wird die XY-Darstellung aktiviert. Es können Werte aus dem Bereich 1..30 dargestellt werden. 0 bedeutet keine Darstellung. Die Werte werden von links beginnend dargestellt. Wenn nach 47 Aufrufen das Ende erreicht ist, werden die Werte nach links geschoben.
... entweder in „einfacher“ ... *.. oder „doppelter“ Höhe ...*	*mchart* *mpchart*	 Die XY-Werte werden im Anwenderprogramm über die Funktion mchart (Seite 263) angesprochen. Ein mchart-Element verwaltet bis zu 4 XY-Diagramme, die über die gleich lautende Funktion mchart im Anwendungsprogramm mit Daten gespeist wird. Es können max. 4 Diagramme vorgegeben werden. Jedes der 4 Diagramme hat eine eigene Legende (rechts oben eingeblendet). Es werden 47 Fließkommawerte dargestellt. Die Skala wird automatisch generiert.
	mchart *mpchart*	 wie oben, jedoch doppelte Höhe.
... oder externe Bildquellen einbinden...	*picture*	 Anzeige eines externen Bildes z. B. Wetterkarte für Deutschland Es wird ein externer Link auf eine Grafik eingebunden. Die Grafik kann linksbündig, zentriert oder gestreckt eingebunden werden.

picture

Anzeige eines externen Bildes z. B. Wetterkarte für Deutschland

Es wird ein externer Link auf eine Grafik eingebunden. Die Grafik kann linksbündig, zentriert oder gestreckt eingebunden werden.

...komplette Webpages einbinden...

Gruppe	Element	Erläuterung
Link		
	frame *dframe*	Einbetten einer einer externen Seite
	pLink	Link auf eine interne Seite als einfacher Knopf
	Link	Link auf eine externe Seite als einfacher Knopf
Dekos		
	line	Überschrift Erzwingt eine freie Zeile mit Trennlinie im Webserveraufbau. Die Überschrift ist optional.
	header	 Kopfzeile. Die kann abgeschaltet werden, um so die Bedienung an Touchpanels zu erleichtern. Ebenso ist ein Link auf eine externe Bildquelle möglich. Die Skalierung sollte dabei auf die Größe angepasst sein.
	footer	 Fußzeile. Die kann abgeschaltet werden, um so die Bedienung an Touchpanels zu erleichtern. Ebenso ist ein Link auf eine externe Bildquelle möglich. Die Skalierung sollte dabei auf die Größe angepasst sein.
	none	Ein Freifeld einfacher Breite

Tabelle 3: Übersicht über Webelemente.

HTTPS

Mit dem Enertex ® EibPC ist eine sichere Kommunikation via HTTPS zwischen WebServer und Browser möglich. Dazu muss lediglich im EibStudio® unter Optionen → WebServer Passwortschutz Konfiguration ein Benutzer mit Passwort eingetragen werden(s. Abb. 126).

Um diese Funktionalität nutzen zu können, benötigen Sie einen gültigen Freischaltcode für die Option NP. Zusätzlich muss in Ihrem Router Port Forwarding für den Port 443 eingerichtet sein.

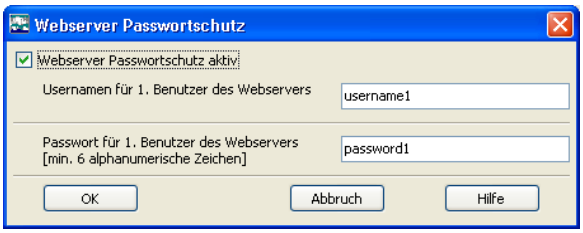


Abbildung 126: Konfigurationsdialog für HTTPS

Konfiguration

Das Design des Webservers ist fest im Enertex® EibPC eingebaut. Das Schema nach Seite 274, Abbildung 124 kann auf zehn Spalten erweitert werden. Der Webserver verwaltet bis zu 60 (IDs von 0 bis 59) Webelemente auf einer Webseite.

Die Konfiguration des Webservers erfolgt in der Sektion [WebServer] im Anwenderprogramm. Die Elemente, die in einer Reihe angeordnet werden, müssen dazu einfach - durch ein oder mehrere Leerzeichen oder Tabulatoren voneinander getrennt – wie folgt konfiguriert werden. Der Compiler erkennt die Anzahl der Elemente pro Reihe und konfiguriert automatisch das „Schachbrettmuster“ (Abbildung 124). Jedes Element muss mit einem Index versehen werden, so dass vom Anwenderprogramm über die entsprechenden Funktionen darauf zugegriffen werden kann.

compact

Element *Compact*

- *compact* (ZUSTAND)

Argumente

- Zustand Wert von 0 / 1 bzw. EIN/AUS

Der Webserver ist in Einheitsgrößen aufgebaut. Alle Elemente passen in dieses Raster bzw. sind ganzzahlige Vielfache hiervon, wie bereits in Abbildung 124 erläutert. Wenn daher ein Element vierfacher Höhe (z.B. mpchart) neben ein Element einfacher Höhe konfiguriert wird,

Ein Beispiel ohne „compact“ Modus

```
[WebServer]
page(1) [$Demo$,$Compact$]
// die nächste Anweisung ist der default
compact(off)
// Zwei Elemente
mpchart(1) [DOUBLE, SXY]($Beschriftung1$,LINE) mpshifter(2) [$Keller$,$OG$][WEATHER, ICE, NIGHT, CLOCK] $Multi$
```

so entsteht in der Darstellung ein Freiraum wie in Abbildung 127.

generiert einen Freiraum unter dem 4-Fach Multi-Knopf

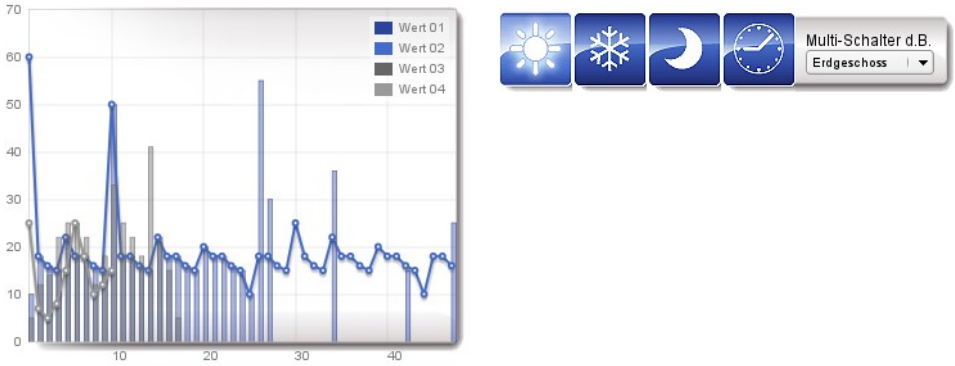


Abbildung 127: Freiraum

Bei der Konfiguration des Webservers steht jede Zeile der Textkonfiguration für eine Darstellungszeile des Webservers. Im „ausgeschalteten“ Compactmodus (compact(off)) sind die Elemente unterschiedlicher Höhe immer in einer Zeile angeordnet, d.h. die eigentliche Zeilenhöhe der Darstellung wird durch die max. Höhe aller Elemente in der jeweiligen Zeile vorgegeben. Dadurch entsteht die freie Stelle im Webserver. Anders ausgedrückt: In der Darstellung werden zusätzliche nicht sichtbare Elemente unter die Elemente gesetzt. Abbildung 128 zeigt diese „Belegung“ der Einheitsgrößen (in Blau dargestellt) der obigen Webkonfiguration.

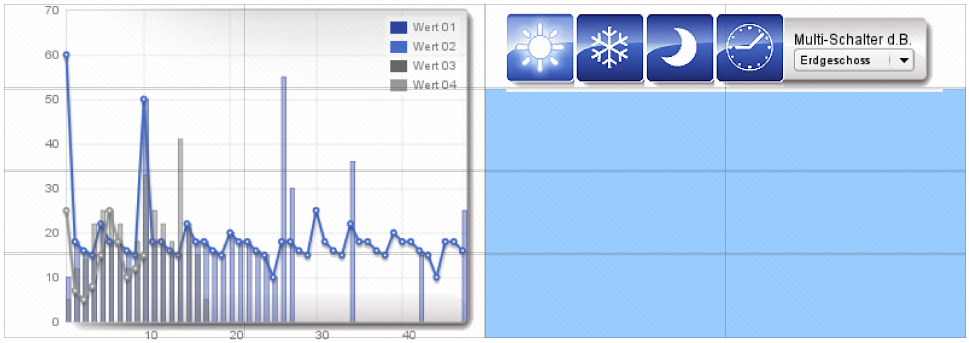


Abbildung 128: Veranschaulichung der Einheitsgrößen.

Der eibparser gibt bereits vorab im Fenster Meldungen die Konfiguration aus:

Die Ausgabe des eibparsers

```
====== Seite: 01/Demo  ======
 mchart (1)    -  mpshifter (2)     -
      |        |          o          o
      |        |          o          o
      |        |          o          o
```

Dabei bedeutet ein Querstrich („-“), dass das Element rechts davon diesen „Platz“, d.h. diese Einheitsgröße belegt, ein senkrechter Strich „|“, dass das Element oberhalb diesen Platz belegt. Ein runder Kreis ist ein automatisch oder per Uservorgabe generiertes leeres Element (none). In Abbildung 128 sind die automatischen generierten Freiräume in blau dargestellt. Diese Ausgabe verdeutlicht somit dem Anwender beim Erstellen der Visu schematisch den Aufbau, wie dieser vom Webserver dargestellt wird. Man vergleiche hierzu die obige Ausgabe des eibparsers mit Abbildung 128,

Wenn man nun den Freiraum rechts neben den Diagramm nutzen will, so muss die Konfiguration geändert wird. Z.b: möchte man neben der Grafik weitere Multibuttons setzen (Abbildung 129).

Compactmodus

```
page(1) [$Demo$,$Compact$]
// die nächste Anweisung ist der default
compact(on)
mpchart(1) [DOUBLE, SXY]($Beschriftung1$,LINE) mpshifter(2) [$Keller$,$OG$][WEATHER, ICE, NIGHT, CLOCK] $Multi$
mpshifter(3) [$Keller$,$OG$][PLUS, TEMPERATURE, Minus] $Multi$
mpshifter(4) [$Keller$,$OG$][PLUS, TEMPERATURE, Minus] $Multi$
mpshifter(5) [$Keller$,$OG$][PLUS, TEMPERATURE, Minus] $Multi$
```

Die erste Zeile ist wie gehabt. Nun können aber die Freiräume von Abbildung 128 genutzt werden, wenn im Compact-Modus gearbeitet wird. Im Compact-Modus werden die Elemente nicht zeilenweise in unterschiedlicher Höhe angeordnet. Da die Zeile

Übertrag von besetzten Einheitselementen über Zeilen hinweg

```
mpchart(1) [DOUBLE, SXY]($Beschriftung1$,LINE) mpshifter(2) [$Keller$,$OG$][WEATHER, ICE, NIGHT, CLOCK] $Multi$
```

ein mpchart doppelter Breite und vierfacher Höhe konfiguriert, ragt dessen Darstellung in drei weitere Zeilen nach unten hinein.

In den Zeilen

```
mpshifter(3) [$Keller$,$OG$][PLUS, TEMPERATURE, Minus] $Multi$
mpshifter(4) [$Keller$,$OG$][PLUS, TEMPERATURE, Minus] $Multi$
mpshifter(5) [$Keller$,$OG$][PLUS, TEMPERATURE, Minus] $Multi$
```

sind Elemente mit doppelter Breite und einfacher Höhe eingebaut. Durch das erste Element sind jeweils zusätzlich 2 Einheitselemente in der Zeile bereits „unsichtbar vorhanden“. Diesen Zeilenüberlauf gibt bereits der eibparser bereits vorab durch die Ausgabe der „-“ bzw. „|“ Zeichen aus:

```
====== Seite: 01/Demo  ======
mchart (1)    -  mpshifter (2)     -
      |        |  mpshifter (3)     -
      |        |  mpshifter (4)     -
      |        |  mpshifter (5)     -
```

an.

Man vergleiche hierzu die Abbildung 129, welche nun der Webserver ausgibt.

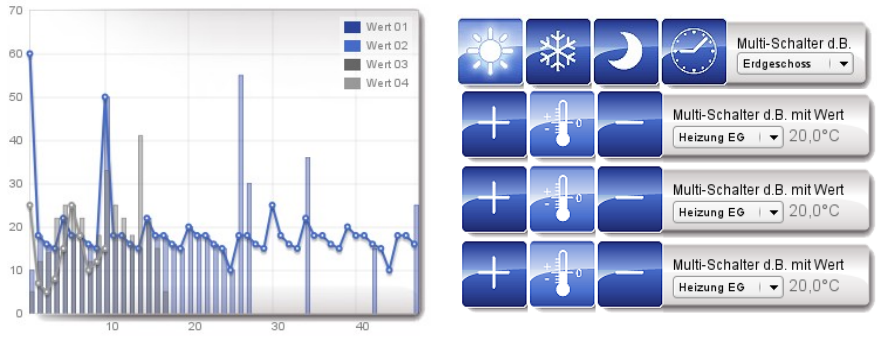


Abbildung 129: Compactmodus

Mit der Anweisung *compact*(EIN) kann das Platzieren von Elementen unterschiedlicher Höhe nebeneinander aktiviert werden. Dabei berechnet der Webserver selbst die Höhenüberläufe in die nächste Zeile. Der Anwender darf hier keine **none** Elelmente platzieren, wenn die Breite nicht vergrößert werden soll. Abbildung 130 zeigt hierzu nochmals schematisch die Anordnung der Elemente, wie das auch schon im eibparser ausgegeben wird.

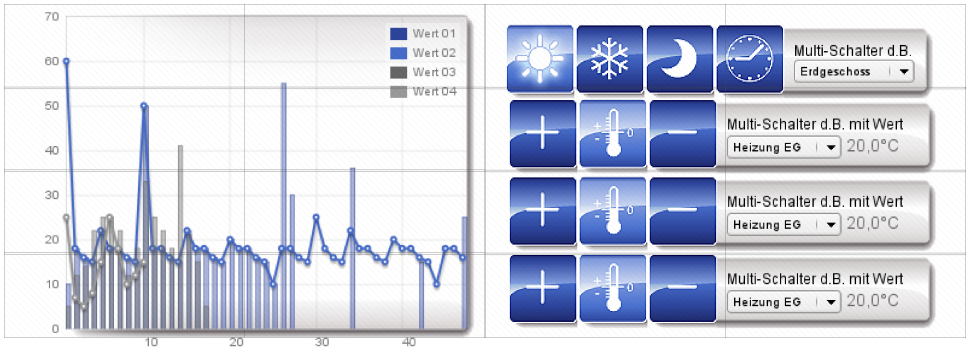


Abbildung 130: „compact“ mit Raster (zur Veranschaulichung)

Im Modus mit *compact*(on) des Webservers muss der Anwender also die Größe des Webelements in die nächste Zeile der Konfiguration mit berücksichtigen, umso die Anordnung der Webelemente zu steuern. Will man eine freie Zeile unter Berücksichtigung der Zeilenüberläufe generieren, so muss mit dem empty-Element gearbeitet werden.

Hierzu folgendes Beispiel

```
page(1) [$Demo$,$Compact$]
// die nächste Anweisung ist der default
compact(on)
mpchart(1) [DOUBLE, SXY]($Beschriftung1$,LINE) mpchart(2) [DOUBLE, SXY]($Beschriftung1$,LINE)
mpshifter(3) [$Keller$,$OG$][WEATHER, ICE, NIGHT, CLOCK] $Multi$
```

Die ersten beiden Elemente belegen je 2 Einheitsbreiten und 4 Einheitshöhen. Nach dem Zeilenumbruch in der Konfiguration der beiden mpcharts beginnt eine neue Zeile in der Darstellung. Diese hat einen „Übertrag“ von zwei mal zwei belegten Einheitselementen. Dann wird in der nächsten Zeile ein mpshifter konfiguriert. Daher muss die Seite mindestens 6 Einheitselemente breit sein. Diese wird auch vom eibparser so ausgegeben:

```
====== Seite: 01/Demo  ======

 mchart (1)   -  mchart (2)   -         o         o
      |       |       |       |  mpshifter (3)    -
      |       |       |       |         o         o
      |       |       |       |         o         o
```

Letztlich wird der Webserver eine Darstellung wie in Abbildung 131 ausgeben.

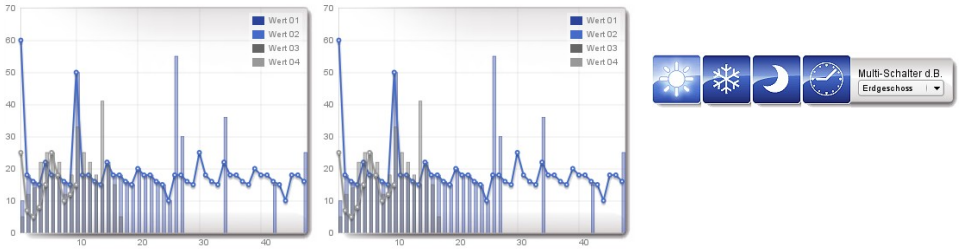


Abbildung 131: Darstellungsbeispiel für den Zeilenvorschub

Will man nun erreichen, dass der 4-Fach Button unterhalb der beiden Grafen angezeigt wird, so müssen empty-Elemente wie folgt konfiguriert werden:

```
page(1) [$Demo$,$Compact$]
// die nächste Anweisung ist der default
compact(on)
mpchart(1) [DOUBLE, SXY]($Beschriftung1$,LINE) mpchart(2) [DOUBLE, SXY]($Beschriftung1$,LINE)
empty
empty
empty
mpshifter(3) [$Keller$,$OG$][WEATHER, ICE, NIGHT, CLOCK] $Multi$
```

Die drei Empty-Elemente fügen nun Leerzeilen ein bzw. überspringen jeweils eine Zeile in der Darstellung. Auch hier kann dies schon schon vorab anhand der vom eibparser angegebenen Ausgabe im Meldungsfenster erkannt werden:

```
====== Seite: 01/Demo  ======

 mchart (1)       -  mchart (2)   -
        |         |       |       |
        |         |       |       |
        |         |       |       |
 mpshifter (3)    -       o       o
```

Der Seiten-Index der lokalen Elemente - , also Elemente, die nur einer Seite zugeordnet werden, - ist derjenige, der im vorangegangenen page-Kommando angegeben ist.

Button

Element *button*

- *button*(ID)[Grafik] $Text$

Argumente

- ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Text: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion Webdisplay (Seite 260) angesprochen.
- Es handelt sich um einen globalen Knopf. Wenn die gleiche Definition auf mehreren Seiten verwendet wird, so werden auf allen Seiten die globalen Elemente mit ID angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion Button (Seite 259) auszuwerten.

Design

Element *design*

- design $DESIGNSTRING$ [$Link/Pfad zu Grafik$]

Argumente

- $DESIGNSTRING$ kann mit $black$ angegeben werden, wenn das „black design“ gewählt werden soll (optimiert für Touchscreens in Wandmontage oder Smart Phones).
- $DESIGNSTRING$ kann mit $blue$ da Standard-Blau Design angebenden werden, wie in den Bildschirmkopieen gezeigt.
- Das design-Kommando kann für jede Seite unterschiedlich angegeben werden.
- $Link/Pfad zu Grafik$ bezeichnet einen Pfad auf dem internen Flash oder einen externen Server (vgl. S. 206). Die Hintergrundgrafik wird nicht skaliert. Die Anordnung der Webelemente bleibt davon unberücksichtigt, none-Elemente werden transparent.
- Der Pfad auf den internen Server ist „/upload/“, vgl. Beispiel S. 116.

Abbildung 132: Hintergrundgrafik

empty

Element *empty*

- empty

Fügt eine Leerzeile bei der Zählung der Webelemente auch im Compact-Modus ein.

Mobilezoom

Element *mobilezoom*

- *mobilezoom(Faktor)*

Argumente

- *Faktor*: ganzzahliger Wert von 0 bis 255 als Zoomfaktor in Prozent für den Zoom der Visualisierung auf mobilen Geräten oder Android-baisierten Panels. Der Zoomfaktor Wirkt nur auf die Seite, die mit einer vorangegangen page-Konfiguration einleitend definiert wurde.

Mbutton

Element *mbutton*

- *mbutton*(ID)[$Text1$,$Text2$,... $Text254$][Grafik] $Beschriftung$

Argumente

- ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Text1, Text2, .. Text254: Beschriftungsfelder für den *mbutton.* Ab dem 2. Element sind die Elemente optional.
- Beschriftung: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion Webdisplay (Seite 151) angesprochen.
- Es handelt sich um einen globalen Knopf. Wenn die gleiche Defintion auf mehreren Seiten verwendet wird, so werden auf allen Seiten die globalen Elemente mit ID angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion Button (Seite 150) auszuwerten.
- Die Umschaltung der Listbox (Vorgabe des aktiven Listboxelements) wird über die Funktion Webdisplay (Seite 260) angesprochen.

Pbutton

Element *pbutton*

- *pbutton*(ID)[Grafik] $Text$

Argumente

- ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Text: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion pdisplay (Seite 265) angesprochen.
- Es handelt sich um ein Element, das immer nur einer Seite zugeordnet ist.
- Die Betätigung der Knöpfe ist über die Funktion pbutton (Seite 266) auszuwerten.

Mpbutton

Element *mpbutton*

- *mpbutton*(ID) [$Text1$,$Text2$,...$Text254$][Grafik] $Beschriftung$

Argumente

- ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Text1, Text2, .. Text254: Beschriftungsfelder für den *mpbutton.* Ab dem 2. Element sind die Elemente optional.
- Beschriftung: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion pdisplay 265 (Seite 265) angesprochen. Die Umschaltung der Listbox (Vorgabe des aktiven Listboxelements) wird auch über diese Funktion angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion mpbutton (Seite 264) auszuwerten.

Shifter

Element *shifter*

- *shifter*(ID)[Grafik1,Grafik2, Grafik3, Grafik4]$Text$

Argumente

- ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Grafik 2 bis Grafik 4 sind optional.
- Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw..
- Text: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion pdisplay (S. 265) angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion button (Seite 259) auszuwerten.

Pshifter

Element *pshifter*

- *pshifter*(ID)[Grafik1,Grafik2, Grafik3, Grafik4]$Text$

Argumente

- ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Grafik 2 bis Grafik 4 sind optional.
- Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw..
- Text: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion pdisplay (S. 265) angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion pbutton (Seite 266) auszuwerten.

Mshifter

Element *mshifter*

- *mshifter*(ID)[$Text1$,$Text2$,...,$Text254$][Grafik1,Grafik2, Grafik3, Grafik4] $Beschriftung$

Argumente

- ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Grafik 2 bis Grafik 4 sind optional.
- Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw..
- Text1, Text2, .. Text254: Beschriftungsfelder für den *mshifter.* Ab dem 2. Element sind die Elemente optional.
- Beschriftung: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden erläutert über die Funktion pdisplay (S. 265) angesprochen. Die Umschaltung der Listbox (Vorgabe des aktiven Listboxelements) wird auch über diese Funktion angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion mbutton (Seite 259) auszuwerten.

Mpshifter

Element *mpshifter*

- *mpshifter*(ID)[$Text1$,$Text2$,...,$Text254$][Grafik1,Grafik2, Grafik3, Grafik4] $Beschriftung$

Argumente

- ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Grafik 2 bis Grafik 4 sind optional.
- Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw..
- Text1, Text2, ..., Text254: Beschriftungsfelder für den mpshifter. Das 2. und 3. Element ist dabei optional.
- Beschriftung: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden wie in Tabelle 1 erläutert über die Funktion pdisplay (S. 265) angesprochen. Die Umschaltung der Listbox (Vorgabe des aktiven Listboxelements) wird auch über diese Funktion angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion mbutton (Seite 259) auszuwerten.

Chart

Element *chart*

- *chart*(ID)[$Y0$,$Y1$,$Y2$]

Argumente

- ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- $Y0$, $Y1$,$Y2$: Achsenbeschriftung der Y-Achse.

Zugriff im Anwenderprogramm

- Die Y-Werte werden wie in Tabelle 1 erläutert im Anwenderprogramm über die Funktion chart (Seite 259) angesprochen.
- Es können Werte aus dem Bereich 1..30 dargestellt werden. Bei jedem Aufruf dieser Funktion werden zunächst die Werte von links beginnend dargestellt. Wenn nach 47 Aufrufen das Ende erreicht ist, werden die Werte nach links geschoben.

Mchart

Element *mchart*

- *mchart*(ID) [Höhe,Typ]($Beschriftung1$,Style1, $Beschriftung2$,Style2, $Beschriftung3$,Style3, $Beschriftung4$,Style4)

Argumente

- *ID:* Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element.
- *Höhe:* Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF)
- *Typ*: Wert 8 (oder Konstante XY): Es wird eine Ortskurve generiert.
- *Typ*: Wert 9 (oder Konstante SXY): Es wird eine X-Y Darstellung gewählt, welche die X-Y Wertepaare vorsortiert (für die Ausgabe von Zeitreihen, z.B: mit Zeitsprung bei 24:00).
- *$Beschriftung1$ .. $Beschriftung4$* Legende des entsprechenden Graphen
- *Style1, Style2, Style3, ,Style4*: Wert 0,1,2 oder 3 (oder Konstante LINE, DOTS, LINEDOTS, COLUMN).

Zugriff im Anwenderprogramm

- Die XY-Werte werden wie in Tabelle 1 erläutert im Anwenderprogramm über die gleichlautende Funktion mchart (Seite 263) angesprochen. Ein mchart verwaltet bis zu 4 XY-Diagramme. Die Anzahl der Diagramme wird durch die Anzahl der Argumente vorgegeben.
- Jedes XY-Diagramm hat eine Legende. Bei Darstellung von 4 XY-Diagramme werden im Diagramm 4 Legenden eingeblendet.
- Es werden 47 Fließkommawerte dargestellt. Die Skala wird automatisch generiert. Beachten Sie hier Hinweise bei der Beschreibung der Anwenderfunktion mchart() auf Seite 263.

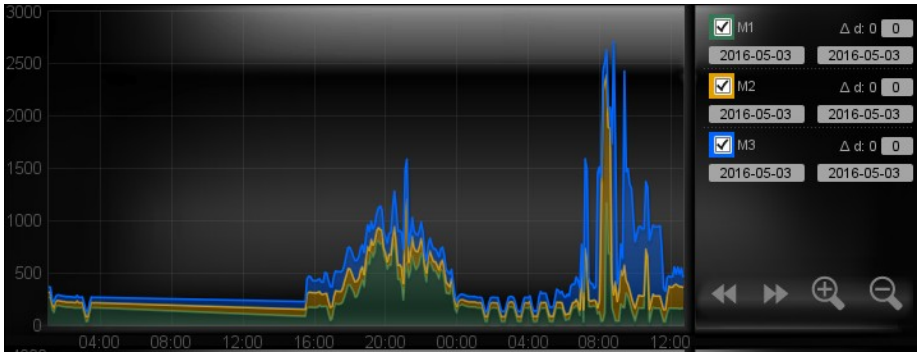
Mtimechart

Element *mtimechart*

- *mtimechart*(ID)[Format,Typ,Länge,YLMIN,YLMAX,YRLMIN,YRLMAX] ($Beschriftung1$,ChartPos1,Buffer1,$Beschriftung2$,ChartPos2,Buffer2, $Beschriftung3$,ChartPos3,Buffer3,$Beschriftung4$,ChartPos4,Buffer4)
- Legendeneintrag1$,CHARTPOS1,verkn.Buffer1, $Legendeneintrag2$,... bis zu 4 Graphen)

Argumente

- *ID:* Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element.
- *Format:* DOUBLE,TRIPLE,QUAD,LONG,EXTDOUBLE,EXTTRIPLE,EXTLONG
- *Typ*: 0 für autoscale der linken Achse, in diesem Fall wird YLMAX etc. ignoriert
  1 für autoscale der rechten Achse, in diesem Fall wird YRMAX etc. ignoriert
  2 für autoscale der beiden Achsen
  3 für kein autoscale
- *Länge:* Maximale Anzahl an Wertepaaren, die pro Graph angezeigt werden können (Mögliche Werte: von 32 bis 256)
- *YLMIN*: Minimalwert linke y-Achse, Fließkommazahlen
- *YLMAX*: Maximalwert linke y-Achse, Fließkommazahlen
- *YRMIN:* Minimalwert rechte y-Achse, Fließkommazahlen
- *YRMAX*: Maximalwert rechte y-Achse, Fließkommazahlen
- *$Beschriftung1$ .. $Beschriftung4$* Legende des entsprechenden Graphen
- *ChartPos*: Wert 0 (LEFTGRAF) oder 1 (RIGHTGRA) (0 für Beschriftung an der linken y-Achse, 1 für Beschriftung an der rechten y-Achse) oder 2 (STACK) für grafisches Addieren von zwei Grafen: Die äußerste Einhüllende ist als die Gesamtsumme der Einzelgrafen zu verstehen:



- *Buffer:* ID des mit dem jeweiligen Graphen verknüpften Timebuffers. Werte zwischen 0 bis 255 als Index für die Programmierung und den Zugriff.
- ACHTUNG: Der EibPC hat einen RAM von 64MB.

  Um einen ordnungsgemäßen Betrieb zu garantieren, müssen die Buffer und mtimecharts so dimensioniert werden, dass der Speicher des EibPC nicht überlastet wird. Siehe hier unter timebufferconf (S. 269) weitere Details.
- Die Formate EXTDOUBLE, EXTTRIPLE, EXTLONG sind Graphen mit intergrierter Zoom-, Verschiebungsfunktion und Zeitversatzeinstellung.

Zugriff im Anwenderprogramm

- Die XY-Werte werden im Anwenderprogramm über die Funktion timebufferadd S. 269 und timebufferconf S. 269 angesprochen. Ein mtimechart verwaltet bis zu 4 XY-Diagramme. Die Anzahl der Diagramme wird durch die Anzahl der Argumente vorgegeben.
- Jedes XY-Diagramm hat eine Legende. Bei Darstellung von 4 XY-Diagramme werden im Diagramm 4 Legenden eingeblendet.
- Es können bis zu 65535 Fließkommawerte dargestellt werden. Für die Skalierung beachten Sie hier Hinweise bei der Beschreibung der Anwenderfunktionen timebufferadd S. 269 und timebufferconf S. 269.
- mtimecharts sind immer global.

Ausführliches Beispiel: S. 119

timechartcolor

Element *timechartcolor*

- *timechartcolor* ID *#HtmlFarbCode*
  Verändert den Farbwert des Graphen mit der ID (1,2,3,4) der timecharts. Die Formatierung ist zur üblichen HTML-Farbcodedefintion identisch, vgl. (https://wiki.selfhtml.org/wiki/Grafik/Farbpaletten)
- Diese Einstellung ist global für alle Graphen gültig und wird hinter ein page-Kommando gestellt.

Beispiel

```
[WebServer]
page (wsMeter) [$Smartmeter$, $Messung$
timechartcolor 1 #337755
timechartcolor 2 #e5a000
timechartcolor 3 #0066ff
timechartcolor 4 #ffff00
```

Picture

Element *picture*

- *picture* (ID) [*Höhe, Typ*]($Beschriftung$,$www-LINK$)

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element.
- *Höhe:*Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF)
- *Typ*: Wert 0,1,2 (oder LEFTGRAF, CENTERGRAF, ZOOMGRAF): Linksbündiges, mittiges oder gestrecktes Einbinden der Grafik.
- www-LINK: Gültig WWW-Adresse (inkl. Pfad und führenden http://) zur externen Grafik

Zugriff im Anwenderprogramm

- Mit Hilfe der seitenbezogenen Funktion picture und dem dort spezifizierten Argumenten Beschriftung und www-LINK kann die Beschriftung und der Link während der Laufzeit dynamisch verändert werden.

Mpchart

Element *mpchart*

- *mpchart*(ID) [*Höhe, Typ*]($Beschriftung1$,Style1, $Beschriftung2$,Style3, $Beschriftung3$,Style3, $Beschriftung4$,Style4)

Argumente

- *ID* : Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element.
- *Höhe*: Wert 0, 1, 2 oder 3 (bzw. Konstante SINGLE, DOUBLE, HALF und LONG)
- *Typ*: Wert 8 (oder Konstante XY): Es wird eine Ortskurve generiert
- *Typ*: Wert 9 (oder Konstante SXY): Es wird eine X-Y Darstellung gewählt, welche die X-Y Wertepaare vorsortiert (für die Ausgabe von Zeitreihen, z.B: mit Zeitsprung bei 24:00)
- *$Beschriftung1$ .. $Beschriftung4$* Legende des entsprechenden Graphen
- *Style1, Style2, Style3, ,Style4*: Wert 0,1,2 oder 3 (oder Konstante LINE, DOTS, LINEDOTS, COLUMN).

Zugriff im Anwenderprogramm

- Die XY-Werte werden wie in Tabelle 1 erläutert im Anwenderprogramm über die gleichlautende Funktion mpchart (Seite 263) angesprochen. Ein mpchart verwaltet bis zu 4 XY-Diagramme. Die Anzahl der Diagramme wird durch die Anzahl der Argumente vorgegeben.
- Jedes XY-Diagramm hat eine Legende. Bei Darstellung von 4 XY-Diagramme werden im Diagramm 4 Legenden eingeblendet.
- Es werden 47 Fließkommawerte dargestellt. Die Skala wird automatisch generiert. Beachten Sie hier Hinweise bei der Beschreibung der Anwenderfunktion mpchart() auf Seite 263.

Pchart

Element p*chart*

- *pchart*(ID)[$Y0$,$Y1$,$Y2$]

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- *$Y0$, $Y1$,$Y2$*: Achsenbeschriftung der Y-Achse.

Zugriff im Anwenderprogramm

- Die Y-Werte werden wie in Tabelle 1 erläutert im Anwenderprogramm über die Funktion pchart (Seite 259) angesprochen.
- Es können Werte aus dem Bereich 1..30 dargestellt werden. Bei jedem Aufruf dieser Funktion werden zunächst die Werte von links beginnend dargestellt. Wenn nach 47 Aufrufen das Ende erreicht ist, werden die Werte nach links geschoben.

Slider

Element *slider*

- *slider*(ID)[Grafik]$Beschriftung$

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- *Grafik*: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- *Beschriftung*: Ein statischer Beschriftungstext (erste Zeile)

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden wie in Tabelle 1 erläutert über die Funktion display (Seite 260) angesprochen.
- Die Betätigung des Sliders ist über die Funktion getslider (Seite 261) auszuwerten.
- Die Verstellung des Sliders ist über die Funktion setslider (Seite 268) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion button (Seite 259) auszuwerten.
- Der Wert kann auch direkt im numerischen Fenster des Sliders geändert werden.

Pslider

Element *pslider*

- *pslider*(ID)[Grafik]$Beschriftung$

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- *Grafik*: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- *Beschriftung*: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden wie in Tabelle 1 erläutert über die Funktion pdisplay (Seite 260) angesprochen.
- Die Betätigung des pSliders ist über die Funktion getpslider (Seite 261) auszuwerten.
- Die Verstellung des pSliders ist über die Funktion setpslider (Seite 268) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion pbutton (Seite 266) auszuwerten.

Eslider

Element e*slider*

- *eslider*(ID)[Grafik] (Min,Inkrement, Max) $Beschriftung$ $Label$

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- *Grafik*: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- *Beschriftung*: Ein statischer Beschriftungstext (erste Zeile)
- *Min*: Minimalwert für den eslider
- *Inkrement*: Inkrement für den eslider
- *Max*: Maximalwert für den eslider
- *Label*: Label für die Werteanzeige, max. zwei Stellen

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden, wie in Tabelle 1 erläutert, über die Funktion display (Seite 260) angesprochen.
- Die Betätigung des esliders ist über die Funktion geteslider (Seite 261) auszuwerten.
- Die Verstellung des esliders ist über die Funktion seteslider (Seite 268) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion button (Seite 259) auszuwerten.
- Der Wert kann auch direkt im numerischen Fenster des Sliders geändert werden.

Peslider

Element *peslider*

- *peslider*(ID)[Grafik] (Min,Inkrement, Max) $Beschriftung$ $Label$

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- *Grafik*: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- *Beschriftung*: Ein statischer Beschriftungstext (erste Zeile).
- *Min*: Minimalwert für den peslider
- *Inkrement*: Inkrement für den peslider
- *Max*: Maximalwert für den peslider
- *Label*: Label für die Werteanzeige, max. zwei Stellen

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden wie in Tabelle 1 erläutert über die Funktion pdisplay (Seite 260) angesprochen.
- Die Betätigung des Sliders ist über die Funktion getpeslider (Seite 261) auszuwerten.
- Die Verstellung des Sliders ist über die Funktion setpeslider (Seite 269) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion pbutton (Seite 266) auszuwerten.
- Der Wert kann auch direkt im numerischen Fenster des Sliders geändert werden.

Webinput

Element *webinput*

- *webinput*(ID)[Grafik,STIL] $Beschriftung$

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- *Grafik*: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- *Beschriftung*: Ein statischer Beschriftungstext
- STIL ist optional. Mögliche Ausprägungen sind
  - PASSWORD: In diesem Fall wird die Eingabe mit Sternchen oder vom Webbrowser vorgegebene Zeichen versteckt.
  - DATEPICK: Die Eingabe eines Datums mit Hilfe eines Standarddialogs (Darstellung abhängig vom Webbrowser). Die Ausgabe mit webinput (s. 273) erfolgt als String in der Darstellung $YYYY-MM-DD$
  - TIMEPICK: Die Eingabe einer Uhrzeit mit Hilfe eines Standarddialogs (Darstellung abhängig vom Webbrowser). Die Ausgabe mit webinput (s. 273) erfolgt als String in der Darstellung $HH-MM-SS$
  - COLORPICK: Die Eingabe einer RGB Farbe mit Hilfe eines Standarddialogs (Darstellung abhängig vom Webbrowser). Die Ausgabe mit webinput (s. 273) erfolgt als String als 24-Bit Zahl.

Zugriff im Anwenderprogramm

- Das Element wird über die Funktion webinput (s. 273) angesprochen.
- Webinputelemente sind immer global.

weboutput

Element *weboutput*

- *weboutput*(ID)[Größe,Stil]

Argumente

- *ID*: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- *Größe*: Wert 0, 1, 2...5 (bzw. Konstante SINGLE, DOUBLE und QUAD, bzw.
  Breite x Höhe: beliebige Zahl für Höhe und Breite als Faktor der Einheitsgröße der Elemente des Webservers also z.B. 3x5
- *Stil*:Wert 0,1,2 (bzw. Konstante ICON und NOICON, NOCOLOR)

Zugriff im Anwenderprogramm

- Das Element wird über die Funktion weboutput (s. 273) angesprochen.
- Weboutputelemente sind immer global.

Beispiel siehe S. 116

Page

Element *page*

- *page*(ID)[$Gruppe$,$Name$]

Argumente

- *ID*: Wert zwischen 1 bis 100 als Seitenindex für die Programmierung und den Zugriff auf lokale Seitenelemente (Anfangsbuchstabe „p“). Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Die Schnellauswahl (Vor- bzw. Zurückknopf in Abbildung 125) ist durch die Reihenfolge der Definition gegeben. Zwischen zwei Definitionen von page-Elementen sind die Elemente auf dieser Seite zu verwenden.
- *Gruppe*: Eine Zuordnung der Seite zu einer Gruppe. Wenn eine Seite einer Gruppe zugeordnet wird, dann bestimmt die Reihenfolge der Definitionen der Seiten die Reihenfolge der Seiten in der Auswahlbox. Auf diese Weise können Gruppen wie „Keller“, „Erdgeschoß“ etc. generiert werden.
- *Name*: Ein statischer Beschriftungstext

Zugriff im Anwenderprogramm

- keiner

User

Element *user*

- *user* $Name$ [Passwort]

Argumente

- Name: Benutzername. Dieser Benutzer hat Zugriff auf die entsprechende Seite.
- Passwort: Der angegeben Benutzer benötigt dieses Passwort um Zugriff auf die entsprechende Seite zu bekommen.

Zugriff im Anwenderprogramm

- keinen

HINWEIS:

Die Benutzerabfrage und das Kennwort sind nicht „sicher“, sondern dient lediglich dazu, Benutzerfehleingaben am Webserver auf einfache Weise abzufangen. Die erzielte IT-Sicherheit ist als gering einzustufen und keineswegs mit der des HTTPS-Zugriffs zu vergleichen.

Line

Element *line*

- *line* $Text$

Argumente

- Keine. Das Element fügt eine Trennlinie zwischen zwei Zeilen ein.
- Der Text wird an die Trennlinie gebunden und ist optional.

Zugriff im Anwenderprogramm

- keinen

Header

Element *header*

- *header*(Nummer) $www.link$

Argumente

- Wenn Nummer den Wert 0 annimmt, wird Kopfzeile ausgeblendet. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Der Link (inkl. Pfad und führenden http://) ist optional. Dieser kann auf eine externe Quelle zugreifen. In diesem Fall muss für Nummer der Wert 2 gesetzt werden.
- Die Kopfzeile ist konfigurierbar, aber für alle Seiten gleich.

Zugriff im Anwenderprogramm

- keinen

Footer

Element *footer*

- *footer*(Nummer) $WWW-Link$

Argumente

- Wenn Nummer den Wert 0 annimmt, wird Kopfzeile ausgeblendet. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Der Link (inkl. Pfad und führenden http://) ist optional. Dieser kann auf eine externe Quelle zugreifen. In diesem Fall muss für Nummer der Wert 2 gesetzt werden.
- Die Fußfzeile ist konfigurierbar, aber für alle Seiten gleich.

Zugriff im Anwenderprogramm

keinen

None

Element *None*

- *none*

Argumente

- Keine. Ein leeres Element einfacher Breite wird im Webserver eingefügt.

Zugriff im Anwenderprogramm

- keinen

Plink

Element plink (Verlinkung auf andere Seite des Webservers)

- *plink*(ID)[Grafik] [PageID] $Text$

Argumente

- ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. (Grafisch ist das Element zum button identisch)
- Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- PageID: Wert von 1 bis 100 als Sprung-Index der Seite, auf welche bei Knopfdruck gesprungen werden soll. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen.
- Text: Ein dynamischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Grafik, Seitenverweis und Text können über die Funktion plink (S. 267) verändert werden.

Link

Element link (Verlinkung auf externe Webseite)

- *link*(ID)[Grafik][$Website$] $Text$

Argumente

- ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. (Grafisch ist das Element zum button identisch)
- $Website$ http-Adresse (inkl. Pfad und führenden http://) des Ziels
- Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300).
- Text: Ein dynamischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Mit Hilfe der Seiten bezogenen Funktion link (S. 262) und den dort spezifizierten Argumenten *Text, Icon, Website* können diese Argumente während der Laufzeit dynamisch verändert werden.

Frame

Element frame (Einbetten einer externen HTML-Seite)

- *frame* [$Text$]

Argumente

- Text: Ein WWW-Link (inkl. Pfad und führenden http://) auf externe Seite, die in den Webserver integriert wird.

Zugriff im Anwenderprogramm

- keinen

Dframe

Element dframe (Einbetten einer externen HTML-Seite)

- *dframe* [$Text$]

Argumente

- Text: Ein WWW-Link (inkl. Pfad und führenden http://) auf externe Seite, die in den Webserver integriert wird. Das eingebettte Fenster ist doppelt so hoch, wie das von frame.

Zugriff im Anwenderprogramm

- keinen

In der Sektion [WebServer] im Anwenderprogramm wird der Webserver konfiguriert. Der Webserver ist wie ein Schachbrettmuster aufgebaut (vgl. Abbildung 124).

Eine Zeile dieses Musters entspricht einer Zeile in dieser Sektion. Die Elemente einer Zeile müssen entsprechend der oben gezeigten Syntax durch ein oder mehrere Leerzeichen oder einen oder mehrere Tabulatoren getrennt angeordnet werden.

Sektion [WebServer]

Die Elemente *header* und *line* müssen einzeln in einer Textzeile in [WebServer] stehen.

Wenn Sie in einer Zeile weniger als vier Elemente angeben, füllt der Webserver automatisch am Ende die Zeile mit *none*-Elementen auf. Die maximale Anzahl von Spalten ist auf zehn begrenzt, wobei beachtet werden muss, dass die Elemente *shifter* und *chart* doppelte Normlänge aufweisen.

Das minimale Design besteht aus fünf Zeilen und vier Spalten, sodass auf Touchpanels mit einer Auflösung von 800x600 Punkte eine Visualisierung erstellt werden kann. Nicht benötigte Elemente müssen allerdings nicht konfiguriert werden.

Eine mögliche Konfiguration unter Zuhilfenahme der Icons (Seite 300) lautet dann:

Die Konfiguration des Webservers ist einfach gehalten

```
[WebServer]
button(0)[INFO]$InfoText$     None      button(1)[Clock]$Heute ist$
line
chart(0)[$10 C$,$21.5 C$,$33 C$] shifter(2)[PLUS,TEMPERATURE,MINUS]$Sollwert$
shifter(3)[PLUS,MINUS]$2er Knopf$  shifter(4)[mail]$1er Knopf$
shifter(5)[PLUS,MINUS,UP,DOWN]$4er Knopf$
[EibPC]
```

Dieses Anwenderprogramm kann direkt eingespielt werden, d. h. die Sektion [EibPC] muss keine weitere Programmierung enthalten, was insbesondere für den Entwurf der Visualisierung von Bedeutung ist. Nach der Übertragung des Programms auf den EibPC wird der Webserver gestartet, falls die Option NP auf diesem Gerät freigeschaltet ist.

Während das Anwenderprogramm mit dem Zugriff auf die KNX™-Schnittstelle etwa acht Sekunden für den Start benötigt, beansprucht der Webserver ca. 15 Sekunden.

Der Webserver benötigt etwa 15 Sekunden nach dem Überspielen der Daten beim Systemstart, bis er verfügbar ist

Abbildung 133: Webserver im blue-Design

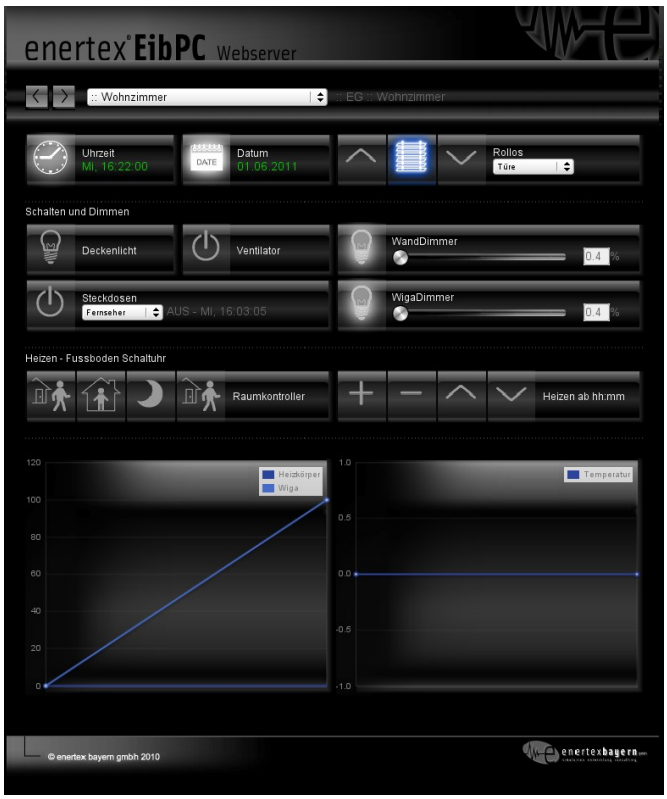


Abbildung 134: Webserver im black-Design

Initialisierung

Jedes Icon des Webservers hat verschiedene Ausprägungen (Farben etc.). In der Konfigurationsdatei wird für die Initialisierung immer der Zustand 1 (INACTIVE) gesetzt. Weitere Zustände (siehe Seite 300) können über die Funktion Webdisplay (Seite 260) eingestellt werden.

Webicons

Der Enertex® EibPC verfügt über einen eingebauten Satz von Grafiken. Diese können direkt über ihren Index (Symbolgruppe) und ihren Subindex (Gestaltung) angesprochen werden.

Es existieren die folgenden Symbolgruppen (Bilder siehe auf 303), die sowohl in der Sektion [WebServer] als auch im Anwenderprogramm als entsprechendes Argument von Webdisplay direkt über den Namen oder die Zahl angesprochen werden.

Symbol	Index
INFO	0u08
SWITCH	1u08
UP	2u08
DOWN	3u08
PLUS	4u08
MINUS	5u08
LIGHT	6u08
TEMPERATURE	7u08
BLIND	8u08
STOP	9u08
MAIL	10u08
SCENES	11u08
MONITOR	12u08
WEATHER	13u08
ICE	14u08
NIGHT	15u08
CLOCK	16u08
WIND	17u08
WINDOW	18u08
DATE	19u08
PRESENT	20u08
ABSENT	21u08
REWIND	22u08
PLAY	23u08
PAUSE	24u08
FORWARD	25u08
RECORD	26u08
STOP	27u08
EJECT	28u08
NEXT	29u08
PREVIOUS	30u08
LEFT	31u08
RIGHT	32u08
CROSSCIRCLE	33u08
OKCIRCLE	34u08
STATESWITCH	35u08
PLUG	36u08

METER	37u08
PVSOLAR	38u08
THERMSOLAR	39u08
PUMP	40u08
HEATINGUNIT	41u08
HEATPUMP	42u08
FLOORHEATING	43u08
WALLHEATING	44u08
COOLER	45u08
MICRO	46u08
SPEAKER	47u08
RGB	48u08
LUX	49u08
RAIN	50u08
KEY	51u08
WASTE	52u08
ASK	53u08
WARN	54u08
NEAR	55u08
CAMERA	56u08
SIGNAL	57u08
DOOR	58u08
GARAGE	59u08
CURTAIN	60u08
ANGLE	61u08
ROLLER	62u08
EMAIL	63u08
PETS	64u08
PERSON	65u08
PHONE	66u08
TV	67u08
BEAMER	68u08
RADIO	69u08
RECIEVER	70u08
MEDIA	71u08
STOVE	72u08
FRIDGE	73u08
WASHER	74u08
DISHWASHER	75u08
HOLIDAY	76u08
SLEEP	77u08

Tabelle 4: Übersicht über Symbolgruppen

Jedes Symbol einer Gruppe kann in unterschiedlichen Ausprägungen angezeigt werden. Hierzu existieren bis zu zehn Zustände, die sowohl in der Sektion [WebServer] als auch im Anwenderprogramm als entsprechendes Argument von Webdisplay direkt über den Namen oder die Zahl angesprochen werden.

Hinweis: Nicht jede Symbolgruppe implementiert alle möglichen Zustände.

Symbol	Index
DARKRED	0u08
INACTIVE	1u08
ACTIVE	2u08
DISPLAY	3u08
STATE4	4u08
STATE5	5u08
STATE6	6u08
STATE7	7u08
STATE7	8u08
BRIGHTRED	9u08

Tabelle 5: Übersicht über Zustände.

GREY	0u08
GREEN	1u08
BLINKRED	2u08
BLINKBLUE	3u08

Tabelle 6: Übersicht über Text-Stilarten

Anmerkung zu ***BLINKRED*** und ***BLINKBLUE:***

In den meisten Browsern ist die Blinkfunktion deaktiviert. Um sie wieder zu aktivieren benötigt man das passende Plugin. Bei Firefox wäre dies das Plugin *Blink Enable 1.1* .

Im Folgenden sehen Sie die Symbolgruppen. Dabei ist der Dateiname in der Form *Symbolgruppe_Zustand.png* angegeben.

Verhalten des Webservers bei Benutzereingaben

Der integrierte Webserver ist so aufgebaut, dass er bei Tastendruck auf die Schaltflächen im Webbrowser sofort reagiert und an die Verarbeitungsschleife eine entsprechende Information schickt. Beim Tastendruck wechselt zusätzlich das Icon einer Symbolgruppe sofort auf den Zustand DISPLAY, der durch einen Beleuchtungseffekt gekennzeichnet ist. Damit soll dem Anwender die Erkennung der Betätigung erleichtert werden und ein visuelles Feedback entstehen.

Das Anwenderprogramm kann nun auf den Tastendruck reagieren, indem z.B. mit webdisplay oder webchart der Zustand der Anzeigen geändert und die HTML-Datei des Webservers modifiziert wird. Die Veränderungen werden in „Echtzeit“ (innerhalb von ca. 300 ms) am Webserver sichtbar.

Das nachladen externer Bildquellen kann zeitgesteuert werden. Weiteres hierzu finden Sie auf S. 135 unter Performance Einstellungen.

Hier nochmals alle Symbole in der Übersicht:

Symbol	Index	DARKRED 0u08	INACTIVE 1u08	ACTIVE 2u08	DISPLAY 3u08	STATE4 4u08	STATE5 5u08	STATE6 6u08	STATE7 7u08	STATE8 8u08	BRIGHT-RED 9u08
INFO	0u08										
SWITCH	1u08										
UP	2u08										
DOWN	3u08										
PLUS	4u08										
MINUS	5u08										
LIGHT	6u08										
TEMPERATURE	7u08										

BLIND	8u08										
STOP	9u08										
MAIL	10u08										
SCENES	11u08										
MONITOR	12u08										
WEATHER	13u08										
ICE	14u08										
NIGHT	15u08										

CLOCK	16u08										
WIND	17u08										
WINDOW	18u08										
DATE	19u08										
PRESENT	20u08										
ABSENT	21u08										
REWIND	22u08										
PLAY	23u08										
PAUSE	24u08										

FORWARD	25u08										
RECORD	26u08										
STOP	27u08										
EJECT	28u08										
NEXT	29u08										
PREVIOUS	30u08										
LEFT	31u08										
RIGHT	32u08										
CROSSCIRCLE	33u08										

OKCIRCLE	34u08										
STATESWITCH	35u08										
PLUG	36u08										
METER	37u08										
PVSOLAR	38u08										
THERMSOLAR	39u08										
PUMP	40u08										
HEATINGUNIT	41u08										
HEATINGPUMP	42u08										

FLOORHEATING	43u08										
WALLHEATING	44u08										
COOLER	45u08										
MICRO	46u08										
SPEAKER	47u08										
RGB	48u08										
LUX	49u08										
RAIN	50u08										
KEY	51u08										

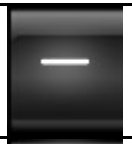
WASTE	52u08										
ASK	53u08										
WARN	54u08										
NEAR	55u08										
CAMERA	56u08										
SIGNAL	57u08										
DOOR	58u08										
GARAGE	59u08										
CURTAIN	60u08										

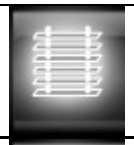
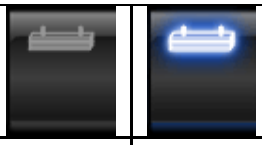
ANGLE	61u08										
ROLLER	62u08										
EMAIL	63u08										
PETS	64u08										
PHONE	65u08										
PERSON	66u08										
TV	67u08										
BEAMER	68u08										
RADIO	69u08										

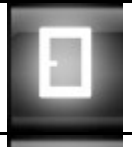
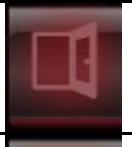
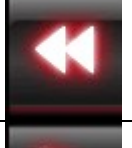
RECIEVER	70u08										
MEDIA	71u08										
STOVE	72u08										
FRIDGE	73u08										
WASHER	74u08										
DISHWASHER	75u08										
HOLIDAY	76u08										
SLEEP	77u08										

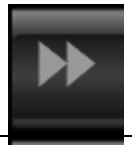
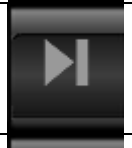
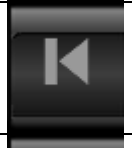
UPDATE	78u08										

Tabelle 7: Übersicht Symbole blue-design

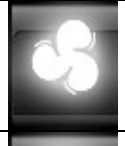
Symbol	Index	DARKRED 0u08	INACTIVE 1u08	ACTIVE 2u08	DISPLAY 3u08	STATE4 4u08	STATE5 5u08	STATE6 6u08	STATE7 7u08	STATE8 8u08	BRIGHT-RED 9u08
INFO	0u08										
SWITCH	1u08										
UP	2u08										
DOWN	3u08										
PLUS	4u08										
MINUS	5u08										
LIGHT	6u08										
TEMPERATURE	7u08										

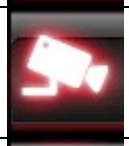
BLIND	8u08																		
STOP	9u08																		
MAIL	10u08																		
SCENES	11u08																		
MONITOR	12u08																		
WEATHER	13u08																		
ICE	14u08																		
NIGHT	15u08																		

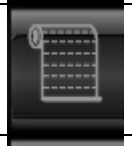
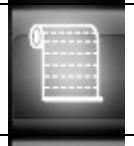
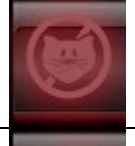
CLOCK	16u08											
WIND	17u08											
WINDOW	18u08											
DATE	19u08											
PRESENT	20u08											
ABSENT	21u08											
REWIND	22u08											
PLAY	23u08											
PAUSE	24u08											

FORWARD	25u08										
RECORD	26u08										
STOP	27u08										
EJECT	28u08										
NEXT	29u08										
PREVIOUS	30u08										
LEFT	31u08										
RIGHT	32u08										
CROSSCIRCLE	33u08										

OKCIRCLE	34u08										
STATESWITCH	35u08										
PLUG	36u08										
METER	37u08										
PVSOLAR	38u08										
THERMSOLAR	39u08										
PUMP	40u08										
HEATINGUNIT	41u08										
HEATINGPUMP	42u08										

FLOORHEATING	43u08										
WALLHEATING	44u08										
COOLER	45u08										
MICRO	46u08										
SPEAKER	47u08										
RGB	48u08										
LUX	49u08										
RAIN	50u08										
KEY	51u08										

WASTE	52u08										
ASK	53u08											
WARN	54u08										
NEAR	55u08										
CAMERA	56u08											
SIGNAL	57u08										
DOOR	58u08										
GARAGE	59u08										
CURTAIN	60u08										

ANGLE	61u08										
ROLLER	62u08										
EMAIL	63u08										
PETS	64u08										
PHONE	65u08										
PERSON	66u08										
TV	67u08										
BEAMER	68u08										
RADIO	69u08										

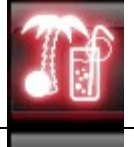
RECIEVER	70u08										
MEDIA	71u08										
STOVE	72u08										
FRIDGE	73u08										
WASHER	74u08										
DISHWASHER	75u08										
HOLIDAY	76u08										
SLEEP	77u08										
UPDATE	78u08										

Tabelle 8: Übersicht Symbole black-design

Makros

Funktionsblöcke

Grundsätzliches

Mit Hilfe von Makros, die wir auch (fertige) Funktionsblöcke nennen, wird

- die Programmierung des Enertex® EibPC für den Anfänger wesentlich vereinfacht bzw.
- die Programmierung des Enertex® EibPC für den versierten Anwender schematisiert. Der Anwender kann komplette Codefragmente von Programmteilen, die er immer wieder verwendet, in eine eigene Bibliothek auslagern und somit die Programmierung bei unterschiedlichen Projekten jederzeit direkt wiederverwenden.
- Sie können den Makroassistenten verwenden, der Ihnen bei der Parametrierung der Makros unterstützt. Dies bedeutet, dass Sie jeden Parameter mit Erklärungen mit Hilfe von Enertex® EibStudio eingeben können. Bei ggf. später notwendiger Änderung können Sie wieder den Assisten benutzten, um die Parameter neu einzugeben.
- Sie können die Makros auch wie eingebaute Funktionen im Anwendungsprogramm im Enertex® EibPC nutzen. In diesem Fall steht Ihnen der Makroassistent allerdings nicht zur Verfügung.

Mitgelieferte Bibliotheken

Auf Seite 26 finden Sie die schrittweise Erklärung der Anwendung von bereits vorliegenden Bibliotheken.

Programmierung eines Makros

Grundsätzliches

Ein Makro ist ein (Teil eines) Anwenderprogramms, das in eine Bibliothek ausgelagert wird. Als eigenständiger Programmteil eines anderen Anwenderprogramm können diese Makros in andere Projekte eingebunden werden. Im Makro kann man verschiedene Eingänge (Argumente) definieren, die projektspezifische Daten enthalten.

Definition

Am einfachsten kann die Makroprogrammierung anhand eines Beispiel erläutert werden. In einem Projekt haben Sie die Doppelbelegung eines KNX™ Tasters programmiert: Ein Tastendruck schreibt ein EIN-Telegramm auf die Adresse 0/0/1. Wenn der Taster innerhalb von 800ms zweimal gedrückt wird, so soll der Enertex® EibPC ein EIN Telegramm auf die Adresse 3/4/6 senden: Es ergibt sich folgendes Anwenderprogramm:

```
[EIBPC]
DoppelKlick=0
if event('0/0/1'b01) and ('0/0/1'b01==EIN) then DoppelKlick=DoppelKlick+1 endif
if after(DoppelKlick==1, 800u64) then write('3/4/5'b01, EIN) endif
if after(DoppelKlick==1, 800u64) and DoppelKlick==2 then write('3/4/6'b01, EIN) endif
if after(DoppelKlick==1, 1000u64) then DoppelKlick=0 endif
```

Wenn diese Funktionalität auf weitere Taster und Gruppenadressen übertragen werden soll, so können Sie den Text per Copy und Paste im Texteditor des Enertex® EibStudio ändern. Allerdings kann diese Methode u.U. fehleranfällig werden.

Mit einem Makro sind Sie in der Lage, in solchen Situationen Schablonen zu schaffen, die die Programmierung einfach gestalten. Hierzu legen Sie sich eine neue Textdatei an (Endung „.lib“) und schreiben nun:

Ein Makro beginnt mit :begin

```
:begin Doppelklick(Name,TasteGA,TasteWert,Klick1GA,Klick1Wert,Klick2GA,Klick2Wert)
Name^DoppelKlick=0
if event(TasteGA) and (TasteGA==TasteWert) then Name^DoppelKlick=Name^DoppelKlick+1 endif
if after(Name^DoppelKlick==1, 800u64) then write(Klick1GA,Klick1Wert) endif
if after(Name^DoppelKlick==1, 800u64) and Name^DoppelKlick==2 then write(Klick2GA,Klick2Wert) endif
if after(Name^DoppelKlick==1, 1000u64) then Name^DoppelKlick=0 endif
:end
```

... endet mit :end

Ein Makro beginnt mit dem Schlüsselwort :begin und endet mit :end. Die Definition selbst ist der Name des Makros gefolgt von durch Komma getrennte Argumente, die von Klammern umgeben sind und befindet sich unmittelbar hinter dem :begin.

Die Argumente des Makros werden als reine Textersetzungen im Makrocode genutzt. Die Syntax ist exakt wie die des „gewöhnlichen“ Anwenderprogramms. Der aus den Makros gleichsam wie aus Textschablonen generierte Code wird vom Compiler intern an die Sektion [EibPC] gebunden. Sie können Ihren vom Compiler generierten Macrocode auch in der Datei „tmpMacroOut.txt“ im Arbeitsverzeichnis des Enertex® EibStudio anschauen.

Wenn das obige Makro z.B. in myMakros.lib gespeichert wird, so vereinfacht sich der „Doppelklick“ auf einen KNX™ Taster:

```
[Macros]
Doppelklick(Keller,'0/0/1'b01,EIN,'3/4/5'b01,EIN,'3/4/6'b01,EIN)
[MacroLibs]
myMakros.lib
[EibPC]
```

Der Compiler schreibt bei unserem Beispiel „tmpMacroOut.txt“ (im Arbeitsverzeichnis vom Enertex® EibStudio):

Die Expansion steht in der Datei „tmpMacroOut.txt“ im Arbeitsverzeichnis

```
KellerDoppelKlick=0
if event('0/0/1'b01) and ('0/0/1'b01==EIN) then KellerDoppelKlick=KellerDoppelKlick+1 endif
if after(KellerDoppelKlick==1, 800u64) then write('3/4/5'b01,EIN) endif
if after(KellerDoppelKlick==1, 800u64) and KellerDoppelKlick==2 then write('3/4/6'b01,EIN) endif
if after(KellerDoppelKlick==1, 1000u64) then KellerDoppelKlick=0 endif
```

Sonderzeichen

Ein Sonderzeichen bei der Textersetzung ist das „^“-Zeichen. Mit diesem kann die Textersetzung so erweitert werden, dass aus zwei Wörtern zusammengesetzte Variablen entstehen. Das „^“-Zeichen wird dabei gelöscht. Die gleiche Wirkung erzielt man mit dem „_“-Zeichen, wobei dieses Zeichen nicht gelöscht wird. Mit Hilfe dieses Vorgehens können (indirekt) Variablen in Makros generiert werden, die aufgrund der Namensgebung gleichsam „gekapselt“ sind.

Auf diese Weise können Sie nun Variablen ähnlich zu objektorientierten Programmiersprachen „kapseln“. Im Beispiel ist die Variable „Doppelklick“ immer wieder genutzt. Wenn es nicht für jedes Makro eine „eigene“ Doppelklick-Variable gäbe, würde das Programm fehlerhaftes Verhalten generieren.

Laufzeit- und Syntaxfehler

Laufzeitfehler oder Syntaxfehler aufgrund falscher Verwendung z.B. Gruppenadressenzuweisungen treten erst bei der „Expansion“ des Makros auf.

Makroasisstent

Sie können Ihre Makros im Quellcode direkt für die Anwendung dokumentieren. Dazu gibt es das Schlüsselwort :info. An erster Stelle nach diesem Schlüsselwort steht hier die Funktionsbeschreibung, gefolgt von einer Beschreibung für jedes Argument. Die Beschreibungen sind von zwei „$“ Zeichen eingefasst.

Die Beschreibung können Sie mit „:info“ selbst erzeugen.

Jede Beschreibung der Argumente ist von zwei $-Zeichen eingerahmt.

```
:info $Mit diesem Funktionsblock können Sie einen Doppelklick auf eine Taste realisieren:\\
      Wenn Sie innerhalb von 0.8 Sekunden zweimal auf den Taster drücken, wird eine
andere\\
      Aktion ausgelöst,als wie wenn Sie einmal drücken. Beide Aktionen können Sie mit \\
      diesem FunktionsblockMakro steuern$ \\
    $Name des Taster (zwecks Eindeutigkeit)$\\
    $Gruppenadresse, auf welcher der Taster Werte sendet$\\
    $Der vom Taster gesendete Wert (z.B. EIN oder AUS) $\\
    $Gruppenadresse für Telegramm bei einfachen Tastendruck$\\
    $Wert für Telegramm bei einfachen Tastendruck (z.B. EIN oder AUS oder 23%$\\
    $Gruppenadresse für Telegramm bei doppelten Tastendruck$\\
    $Wert für Telegramm bei doppelten Tastendruck (z.B. EIN oder AUS oder 23%)$
```

Lokale Variablen und Rückgabewerte

Makros können lokale Variablen definieren, die nur in einem lokalen Kontext des Makros verwendet werden. Wenn ein Makro mehrere Male verwendet wird, werden die lokalen Variablen jeweils eigenständig in jeder Expansion des Makros verwendet. :var VARNAME@ definiert eine lokale Variable ist. Beachten Sie, dass das @-Zeichen am Ende des Namens obligatorisch ist, während VARNAME einen gültigen Variablennamen (Kombination aus Buchstaben und Zahlen und "_" Zeichen) darstellt.
Jedes Makro hat einen Rückgabewert. Entweder ist es mit dem Makro-Befehlszeile :return *Expression* definiert oder wenn dies nicht definiert wurde, wird grundsätzlich die letzte Zeile vor dem :end -Befehl verwendet.

Im folgenden soll eine Funktion $\cosh(x)=\frac{e^x-e^{-x}}{2}$ als Makro so definiert werden, dass diese mit dem berechneten Rückgabewert direkt wie eine „eingebaute“ Funktion genutzt werden kann.

Es können beliebig viele lokale Variablen definiert werden (jede lokale Variable belegt einen der 65.500 Speicherplätze)

```
:begin cosh(x)
:info Berechnet die Cosh-Funktion
:var sum@
:var p_ex@
:var m_ex@
p_ex@=exp(x)
m_ex@=-exp(-x)
sum@=p_ex@+m_ex@
:return sum@ / 2.f32
:end
```

Im obigen Beispiel wäre die Defintion von lokalen Variablen *sum@, p_ex@ u*nd *m_ex@* nicht wirklich notwendig, so dass wir kürzer schreiben können:

Kompakter Code

```
:begin cosh(x)
:info Berechnet die Cosh-Funktion
:return (exp(x)-exp(-x))/2f32
:end
```

Wie bereits erwähnt, ist das :return ebenfalls redundant (um die Kompatibilität mit Makro aus früheren Versionen zu erhalten), weswegen auch

:return wegoptimiert – schlechter zu lesen.

```
:begin cosh(x)
:info Berechnet die Cosh-Funktion
(exp(x)-exp(-x))/2f32
:end
```

möglich wäre. Wenn die letzte Zeile vor dem :end leer ist oder nur Leerzeichen enthält, wird kein Rückgabewert definiert. Grundsätzlich empfiehlt sich die Verwendung des :return Kommandos, alleine aus Gründen der besseren Lesbarkeit des Codes.

:return kann an beliebiger Stelle innerhalb des Makros platziert werden.

Folgender Code hat keinen Rückgabewert

Leerzeile von :end bedeutet, dass kein Wert zurückgegeben wird.

```
:begin cosh(x)
:info Berechnet die Cosh-Funktion
(exp(x)-exp(-x))/2f32
// Achtung : ES WIRD KEIN WERT ZURÜCK GEGEBEN!

:end
```

Ein Makro, das in einer zur Sektion [MacroLibs] beigefügten Makrolib steht, kann wie eine eingebaute Funktion (auch rekursiv) genutzt werden:

Wie eine eingebaute cosh-Funktion

```
[EibPC]
MyVar=cosh(2.3f32)
MyVar2=cosh(cosh('1/3/2'f32)) +cosh('1/3/3'f32) + 32f32
```

Online Debuggen zur Laufzeit

Wenn zur Laufzeit umfangreich Variablen beobachtet werden sollen, so empfiehlt es sich mit Hilfe von UDP Telegrammen und einem netcat Client (vgl. https://de.wikipedia.org/wiki/Netcat) zu debuggen.

Einen String mit CR auf einen UDP Client schicken

Als Debug-Makro wird folgender Code verwendet, der davon ausgeht, dass die Gegenstelle 192.168.1.18 auf Port 9000 hört, z.B. mit dem Unix-Tool `netcat -ul 9000` konfiguriert:

Leermakro

```
#define DEBUG
#ifdef DEBUG
// Debugger an 192.168.1.118 an Port 9000u16
:begin vmDebugUDP(cString)
:return  {
        sendudp(9000u16, 192.168.1.18, cString+tostring(0x0d,0x0a));
}
:end
#endif
#ifndef  DEBUG
:begin vmDebugUDP(cString)
:return __EMPTY()
:end
#endif
```

Je nachdem, ob das Debugging mit dem `#define DEBUG` eingeschaltet ist, wird über UDP eine Nachricht verschickt. Im Falle, dass das `#define DEBUG` auskommentiert wird, werden keine Nachrichten verschickt. Eine Besonderheit stellt dabei die Verwendung von __EMPTY() dar. Diese Anweisung sorgt dafür, dass das Makro nichts expandiert, also auch keinen Code erzeugt.

Effizient bei inaktivem #define von DEBUG

```
x=3
If x>5 then {
    x=x*2;
   vmDebugUDP($x ist nun $+convert(x,$$));
} endif
```

Nun wird bei aktiven `#define DEBUG` über UDP der Wert an den Empfänger zur Laufzeit des Programms automatisch übertragen. Wenn `//#define DEBUG` auskommentiert wird, so erzeugt die Zeile vmDebugUDP(*$x ist nun $*+convert*(x,$$)*); keinen Overhead.

Wenn hingegen eine If-Anweisung **nur** zu Debuggzwecken eingerichtet wird, etwa:

Ineffizient bei inaktivem #define von DEBUG – if Abfrage, die nur dem Debuggen dient.

```
x=3
If x>5 then {
   vmDebugUDP($x ist nun $+convert(x,$$));
} endif
```

so erzeugt der Compiler zwar für vmDebugUDP keinerlei Objekte, allerdings wird ein „verweistes“ if x>5 Objekt angelegt. Diese Art des automatischen Debuggings sollte daher vermieden werden oder eben mit `#define` im Code komplett inaktiviert werden:

... dann lieber so.

```
x=3
#ifdef DEBUG
If x>5 then {
   vmDebugUDP($x ist nun $+convert(x,$$));
} endif
#endif
```

Technische Daten

Versorgung: Der Enertex® EibPC benötigt eine externe Gleichstromversorgung im Bereich 5 bis 30 V DC. Die Leistungsaufnahme ist etwa 1.2 W. Bei LAN Aktivität steigt diese auf 1.7 W.

Zugriff: Um auf den KNX™-Bus zugreifen zu können, benötigt der Enertex® EibPC eine KNX™ Schnittstelle. Dazu wird entweder eine externe EIB-RS232 Schnittstelle für das FT 1.2 Protokoll oder eine IP Schnittstelle benötigt.

Betriebssystem: Debian Linux: 2.6.24-rt1.

Die entsprechenden Softwarequellen werden auf Anfrage jederzeit gegen eine Bearbeitungsgebühr von uns zur Verfügung gestellt.

Schlüsselwörter - Referenz	
Logische Anweisungen	
if ... then ... endif	Wenn – Dann
if ... then ... else ... endif	Wenn – Dann – Sonst
!Var	Bitweise Invertierung
Var1 or Var2	Bitweises Oder
Var1 and Var2	Bitweises And
Var1 xor Var2	Bitweises Exclusiv Oder
Var1 > Var2	Größenvergleich
Var1 < Var2	Größenvergleich
Var1 == Var2	Größenvergleich
Var1 >= Var2	Größenvergleich
Var1 =< Var2	Größenvergleich
Var1 != Var2	Größenvergleich
Hysteresis (Var,UnteresLimit,OberesLimit)	Das Argument Var wird mit den Variablen UnteresLimit und OberesLimit mit einer Hysteresefunktion verglichen.
shift(Operand, Zahl)	Arithmetische Verschiebung des Operanden um Zahl. Bei positiver Zahl Shift nach links, bei negativer Zahl nach rechts. Ge-shiftet werden die Anzahl Bits der Zahl des Eingangs.
Konvertierung	
convert(Var1, Var2)	Konvertiert den Datentypen von Var1 in den von Var2 (Vorsicht: Es können Daten verloren gehen!).
Arithmetische Operatoren	
Var1 + Var2 + VarN	Addition
Var1 – Var2 - VarN	Subtraktion
Var1 * Var2 * VarN	Multiplikation
Var1 / Var2 / VarN	Division
abs(Var1)	Absolutwert
acos(Var1)	Arkuscosinus
asin(Var1)	Arkussinus
atan(Var1)	Arkustangens
ceil(Var1)	Größte Ganzahl ≥ Var1
cos(Var1)	Cosinus
exp(Var1)	E-Funktion
floor(Var1)	Größte Ganzahl ≤ Var1
log(Var1, Var2)	Logarithmus: Var1 = Basiszahl Var2 = Hochzahl

mod(Var1,Var2)	Ergebnis=Var1 modulo Var2
pow(Var1, Var2)	Potenz: Var1= Basiszahl Var2= Exponent
sin(Var1)	Sinus
sqrt(Var1)	Wurzel
tan(Var1)	Tangens
Größenermittlung	
average(Var1, Var2, ... VarN)	Rückgabewert: Durchschnitt der angegebenen Variablen,die alle vom selben Datentyp sein müssen.
min(Var1, Var2, ... VarN)	Rückgabewert: Das Minimum der angegebenen Variablen, die alle vom selben Datentyp sein müssen.
max(Var1, Var2, ... VarN)	Rückgabewert: Das Maximum der angegebenen Variablen, die alle vom selben Datentyp sein müssen.
Zeitsteuerung (Timer)	
after(Steuerung, ms-Zeit)	Steuerung: Binärwert ms-Zeit: ZeitAngabe in ms (<2^{64})
afterc(Steuerung, Zeit, xT)	Steuerung: Binärwert ms-Zeit: ZeitAngabe in ms (<2^{64}) xT: verbleibende Zeit in ms
delay(Signal, Zeit)	Die Funktion startet beim Übergang der Variablen Signal von AUS auf EIN einen Timer und setzt den Rückgabewert der Funktion auf EIN. Nach Ablauf der Zeit in ms springt der Ausgang wieder auf AUS
delayc(Signal, Zeit, xT)	Die Funktion startet beim Übergang der Variablen Signal von AUS auf EIN einen Timer und setzt den Rückgabewert der Funktion auf EIN. Nach Ablauf der Zeit in ms springt der Ausgang wieder auf AUS, in der Variable xT steht die verbleibende Zeit in ms
cycle(mm,ss)	Der Rückgabewert ist ungleich null, wenn die Zeit eingetroffen ist. Wenn der Zeitpunkt eintrifft (und exakt übereinstimmt), geht der Ausgabewert auf 1. mm= Minuten (0...255) ss = Sekunden (0..59)
wtime(hh,mm,ss,dd)	Wochenzeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23) dd: Tag (0=Sonntag, 6=Samstag)
htime(hh,mm,ss)	Tageszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23)
mtime(mm,ss)	Stundenzeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59)
stime(ss)	Minutenzeitschaltuhr: ss: Sekunden (0..59)

cwtime(hh,mm,ss,dd)	Wochenvergleichszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23) dd: Tag (0=Sonntag, 6=Samstag)
chtime(hh,mm,ss)	Tagesvergleichszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23)
cmtime(mm,ss)	Stundenvergleichszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59)
cstime(ss)	Minutenvergleichszeitschaltuhr: ss: Sekunden (0..59)
Synchronisierung mit dem KNX Bus	
dayofweek()	Gibt den aktuellen Tag zurück (0=Sonntag, 6=Samstag)
gettime(Adresse)	Setzt die Zeit des Enertex® EibPC neu
settime()	Schreibt die Zeit des Enertex® EibPC auf den KNX Bus
getdate(Adresse)	Setzt das Datum des Enertex® EibPC neu
setdate()	Schreibt das Datum des Enertex® EibPC auf den KNX Bus
gettimedate(Adresse)	Setzt die Zeit und das Datum des Enertex® EibPC neu
settimedate()	Schreibt das Datum und die Zeit des Enertex® EibPC auf den KNX Bus
hour()	Gibt die aktuelle Zeit (Stunde) als Datentyp u08 zurück
minute()	Gibt die aktuelle Zeit (Minute) als Datentyp u08 zurück
second()	Gibt die aktuelle Zeit (Sekunde) als Datentyp u08 zurück
changehour(Arg)	Ändert die aktuelle Zeit (Stunde)
changeminute(Arg)	Ändert die aktuelle Zeit (Minute)
changesecond(Arg)	Ändert die aktuelle Zeit (Sekunde)
utc(YYYY-MM-DD-HH-MM-SS)	Wandelt eine Zeitangabe im utc-Format in ein c1400 String-Format YYYY-MM-DD-HH-MM-SS um
utcconvert(utc)	Wandelt eine Zeitangabe im YYYY-MM-DD-HH-MM-SS Format zurück in einen UTC-Wert. Dieser Wert ist kompatibel zum Unix-Zeitstempel, aber anstelle von Sekunden wird in Millisekunden gerechnet.
utctime	Gibt die Anzahl der verstrichenen Millisekunden seit Jan 1, 1970 00:00 an. Die Funktion ist kompatibel zum Unix-Zeitstempel, aber anstelle von Sekunden Millisekunden, um für Flot-Zeitreihen bereits das richtige Zeitformat vorzuhalten.
Datumsteuerung	
date(mm,dd,yyy)	Datumsvergleichszeitschaltuhr: yyy: Jahresdifferenz (0..255) vom Jahr 2000 an mm: Monat (1=Januar, 12= Dezember) dd: Tag (1..31)

month(dd,mm)	Monatsvergleichszeitschaltuhr: mm: Monat (1=Januar, 12= Dezember) dd: Tag (1..31)
day(dd)	Tagesvergleichszeitschaltuhr dd: Tag (1..31)
Sonnestand	
azimuth()	DieseFunktion berechnet zyklisch (Zeitraster: 5 Minuten) den Azimuthwinkel der Sonne in Grad, Nord über Ost
elevation()	Dieser Funktion berechnet zyklisch (Zeitraster: 5 Minuten) den Höhenwinkel der Sonne in Grad.
presun(hh:mm)	Gibt in der angegeben Zeit vor dem Wechsel auf Tag eine 1 aus, bzw. vor dem Wechsel auf Nacht eine 0 aus. Das Programm muss dazu die geographische Länge und Breite des betreffenden Ortes kennen
sun()	Gibt aus, ob es Tag oder Nacht ist (Sonnenstand). Das Programm muß dazu die geographische Länge und Breite des betreffenden Ortes kennen.
sunriseminute()	Minute des Sonnenaufgangs am akutellem Tag
sunrisehour()	Stunde des Sonnenaufgangs am akutellem Tag
sunsetminute()	Minute des Sonnenuntergangs am akutellem Tag
sunsethour()	Stunde des Sonnenuntergangs am akutellem Tag
Lesen und Schreiben	
write(Gruppenadresse, Wert)	Auf dem Bus wird ein gültiges EIB-Telegramm generiert.
read(Gruppenadresse)	Auf den Bus wird ein EIB-Telegramm mit Leseanforderung an die Gruppenadresse generiert. Falls die Aktoren etc. antworten, wird das Ergebnis als Rückgabewert der Funktion ermittelt.
Busaktivitäten	
event(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus ein Telegramm mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
eventread(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNXTM Bus eine Leseanforderung mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
eventresponse(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus eine Antwort auf eine Leseanforderung mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
eventwrite(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus eine Antwort auf Telegramm mit der Gruppenadresse geschrieben wird, unabhängig von dessen Inhalt.
writeresponse(Gruppenadresse, Wert)	Antwortet auf eine Leseanforderung, indem ein gültiges KNXTM Telegramm generiert wird, welches den Wert auf die Gruppenadresse schreibt.
Lichtszene	
scene (GruppenadrSzenebaustein, GruppenadresseAktor1, ... GruppenadresseAktorN)	Es wird ein KNX™ Szenenaktor mit der Gruppenadresse GruppenadrSzenebaustein erstellt.
storescene (GruppenadrSzenebaustein, Nummer)	Ein Szenebaustein soll seine Szene mit der entsprechenden Nummer neu abspeichern.
callscene (GruppenadrSzenebaustein, Nummer)	Ein Szenebaustein soll seine Szene mit der entsprechenden Nummer aufrufen.

presetscene(GruppenadresseSzenebaustein, SzenenNummer, OptionOverwrite, WertVariable1,StatusWertVariable1, [WerVariable2,StatusWertVariable2,Variable Wertn, StatusVariableWertn)	Vorbelegung für den Szenenaktor mit der Gruppenadresse GruppenadresseSzenebaustein und entsprechender SzenenNummer erstellen.
RS232 Schnittstelle	
readrs232(data,len)	Der Enertex® EibPC empfängt RS232-Telegramme (nur bei IP-Ankopplung an den KNX™ Bus)
sendrs232(arg1 [, arg2, ... arg n])	Der Enertex® EibPC sendet RS232-Telegramme (nur bei IP-Ankopplung an den KNX™ Bus)
Sonderfunktionen	
change(Var1)	Rückgabewert: 1b01 bei Änderung der überwachten Adresse oder Variable
comobject(Var1, Var2, ... VarN)	Die Rückgabe ist der Wert der zuletzt veränderten Variablen oder Gruppenadresse.
devicenr()	Rückgabewert: Seriennummer des EibPCs
elog()	Auslesen des ältesten Eventspeicher Eintrags
elognum()	Gibt die Anzahl der Einträge im Fehlerspeicher zurück.
eval(Var)	Unterbrechung des Valdidierungsschemas
event(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus ein Telegramm mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
eibtelegramm(Kontrollfeld, Zieladresse, Telegramminformation, Nutzinformation1 .. Nutzinformation18)	Es können eigenen EibTelegramme erstellt werden.
initga(Addresse)	Äquivalent zum Eintrag der Gruppenadresse in [InitGA]
random(Max)	Gibt eine Zufallszahl im Bereich von 0 bis Max zurück.
Sleep(Status)	Bei Status == EIN wird der Enertex® EibPC in den Passivmodus versetzt.
systemstart()	Beim Systemstart werden alle angegebenen Variablen aktualisiert.
Netzwerkfunktionen	
closetcp(port, adresse)	Der Enertex® EibPC schließt eine TCP-Verbindung.
connecttcp(port, adresse)	Der Enertex® EibPC baut eine TCP-Verbindung auf.
ping(adresse)	Der Enertex EibPC führt einen ping auf die angegeben Adresse durch.
readtcp(port, adresse, arg 1 [, arg2, ... arg n]	Der Enertex® EibPC empfängt TCP-Telegramme.
readudp (port, adresse, arg 1 [, arg2, ... arg n])	Der Enertex® EibPC empfängt UDP-Telegramme.
resolve(hostename)	Die Funktion ermittelt die IP-Adresse des angegebenen Hostenamens.

sendmail(empfängeradresse, betreff, nachricht)	Es wird an die Empfängeradresse (Zeichenkette) eine E-Mail mit dem Betreff und der Nachricht verschickt. Alle Zeichenketten werden auf 255 Zeichen begrenzt, auch wenn diese mehr Zeichen beinhalten. Rückgabewert: 0 = E-Mail erfolgreich versendet 1 = in Bearbeitung 2 = Fehler
sendtcp(port, adresse, arg 1 [, arg2, ... arg n]	Der Enertex® EibPC verschickt TCP-Telegramme.
sendtcparray(port, adresse, arg 1 [, arg2, ... arg n, size]	Der Enertex® EibPC verschickt TCP-Telegramme, ohne Nullterminierung.
sendudp(port, adresse, arg 1 [, arg2, ... arg n])	Der Enertex® EibPC verschickt UDP-Telegramme.
sendudp(port, adresse, arg 1 [, arg2, ... arg n], size)	Der Enertex® EibPC verschickt UDP-Telegramme, ohne Nullterminierung.
webbutton(Index)	Gibt das Ereignis aus, welches das Webelement mit Index (0...255) bei dessen Betätigung ausgelöst hat.
webdisplay(Index,Text,Grafik)	Schreibt auf das Webelement Index (0...255) den Text und setzt die Grafik. Die verfügbaren Standard-Grafiken sind codiert als Predefine (rechts unten im Enertex® Enertex® EibStudio) wählbar.
webchart(ID, Var, X1, X2)	Die Funktion spricht das XY-Diagramm chart an. Bei Aufruf wird die XY-Darstellung des Wertes Var aktiviert. ID, Var vom Datentyp u08 X1, X2 Datentyp c14
Stringfunktionen	
capacity(String)	Von der Zeichenkette *String* soll die maximal verfügbare Länge bestimmt werden.
encode(String,Quellcodierung, Zielcodierung)	Eine Zeichenkette *String*, die in der Quellcodierung vorliegt, wird in die Zielcodierung übertragen.
String1 + String2 [+ String3 ... String n]	Die Strings werden aneinander gefügt. Übersteigt die resultierende Länge die maximale Länge des Datentyps, so werden diese Zeichen „abgeschnitten“.
find(String1, String2, Pos1)	String1: Zeichenkette, in der eine (Teil-)Zeichenkette gesucht werden soll. String2: zu findende Zeichenkette Pos1: Die ersten Pos1 - Vorkommen der zu findenden Zeichenkette ignorieren. Die Funktion gibt die Position des ersten Zeichens der gefundenen Zeichenkette (0..1399u16). Sie gibt 1400u16 zurück, falls die Zeichenkette nicht gefunden wurde. Für 65534u16 ist die Konstante END definiert.
size(String)	Von der Zeichenkette String soll die Länge bestimmt werden. Die Länge ist durch das Terminierungszeichen „\0“ am Ende von Zeichenketten gegeben.
split(String, Pos1, Pos2)	String: Zeichenkette, aus der eine Zeichenkette entnommen werden soll Pos1: erstes Zeichen der abzutrennenden Zeichenkette (0...1399u16) Pos2: letztes Zeichen der abzutrennenden Zeichenkette (0...1399u16). Wenn Pos2 gleich 65534u16 (vordefinierte Konstante END) ist, so wird die Zeichenkette bis zu deren Ende abgetrennt. Die Variable String muss vom Datentyp c1400 sein. Rückgabewert: die von Variable String abgetrennte Zeichenkette
stringcast(String, Data, Pos)	Interpretation von Binärdaten eines c1400 Strings
stringformat(Data, Umwandlungtyp, Format, Feldbreite,[Präzision])	Umwandlung von Zahlen in c1400 Strings unter Berücksichtigung verschiedener Formate

stringset(String, Data, Pos)	Schreiben von Binärdaten in einen c1400 String.
VPNServer (Option NP)	
startvpn()	VPN Service starten.
stopvpn()	VPN Service stoppen.
openvpnuser(Name)	Userzugang öffnen.
closevpnuser(Name)	Userzugang schließen.
FTP Funktionen	
Ftpconfig (server,user,password,path,timeout)	Konfiguration eines FTP-Servers
sendftp(handle,data1,[data2],[...])	Beliebige Daten in die Warteschlange des FTP-Handles schreiben.
flushftp(handle)	Daten manuell auf den FTP-Server schreiben.
ftpstate(handle)	Gibt Informationen über den Status der FTP-Konfiguration zurück.
Ftptimeout(handle)	Gibt die bereits verstrichene Zeit seit dem letzten Transfer in Sekunden zurück.
KNX™ Routing (Option NP)	
address(Nummer)	Indirekte Adressierung mit Gruppenadressen
getaddress(Gruppenaddresse)	Gruppenadresse in 16-Bit Zahl umwandeln.
gaimage(Nummer)	Abbild einer Gruppenaddresse ermitteln.
readknx(Adr,String)	Umwandlung eines Telegramms am Bus in Adresse und dezidierten Informationen
readrawknx(Kontrollfeld, PhyAdresse, Zieladresse, IsGroubAdress, RoutingZähler, BitLänge, Nutzdaten)	Umwandlung eines Telegramms am Bus in Adresse und dezidierten Informationen
Webserver-Elemente Konfiguration	
design $DESIGNSTRING$	$DESIGNSTRING$ kann mit $black$ oder $blue$ angegeben werden Das design-Kommando kann für jede Seite unterschiedlich angegeben werden.
page(ID)[$Gruppe$,$Name$]	ID: Wert zwischen 1 bis 100 als Seitenindex für die Programmierung und den Zugriff auf lokale Seitenelemente (Anfangsbuchstabe „p“). Gruppe: Eine Zuordnung der Seite zu einer Gruppe Name: Ein statischer Beschriftungstext
header(Nummer) $www.link$	Wenn Nummer den Wert 0 annimmt, wird Kopfzeile ausgeblendet. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Der Link (inkl. Pfad und führenden http://) ist optional. Dieser kann auf eine externe Quelle zugreifen. In diesem Fall muss für Nummer der Wert 2 gesetzt werden. Die Kopfzeile ist konfigurierbar, aber für alle Seiten gleich.
footer(Nummer) $WWW-Link$	Wenn Nummer den Wert 0 annimmt, wird Kopfzeile ausgeblendet. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Der Link (inkl. Pfad und führenden http://) ist optional. Dieser kann auf eine externe Quelle zugreifen. In diesem Fall muss für Nummer der Wert 2 gesetzt werden. Die Fußfzeile ist konfigurierbar, aber für alle Seiten gleich.

Line $Text$	Das Element fügt eine Trennlinie zwischen zwei Zeilen ein. Der Text wird an die Trennlinie gebunden und ist optional.
none	Ein leeres Element einfacher Breite wird im Webserver eingefügt.
button(ID)[Grafik] $Text$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text: Ein statischer Beschriftungstext (erste Zeile).
pbutton(ID)[Grafik] $Text$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text: Ein statischer Beschriftungstext (erste Zeile).
mbutton(ID)[$Text1$,$Text2$,... $Text254$] [Grafik] $Beschriftung$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text1, Text2, .. Text254: Beschriftungsfelder für den mbutton. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).
mpbutton(ID)[$Text1$,$Text2$,... $Text254$] [Grafik] $Beschriftung$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text1, Text2, .. Text254: Beschriftungsfelder für den mpbutton. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).
shifter(ID)[Grafik1,Grafik2, Grafik3, Grafik4]$Text$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text: Ein statischer Beschriftungstext (erste Zeile).
pshifter(ID)[Grafik1,Grafik2, Grafik3, Grafik4]$Text$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text: Ein statischer Beschriftungstext (erste Zeile).
mshifter(ID)[$Text1$,$Text2$,...,$Text254$] [Grafik1,Grafik2, Grafik3, Grafik4] $Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text1, Text2, .. Text254: Beschriftungsfelder für den mshifter. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).
mpshifter(ID)[$Text1$,$Text2$,...,$Text254$] [Grafik1,Grafik2, Grafik3, Grafik4] $Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text1, Text2, .. Text254: Beschriftungsfelder für den mpshifter. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).
chart(ID)[$Y0$,$Y1$,$Y2$]	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. $Y0$, $Y1$,$Y2$: Achsenbeschriftung der Y-Achse.

pchart(ID)[$Y0$,$Y1$,$Y2$]	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. $Y0$, $Y1$,$Y2$: Achsenbeschriftung der Y-Achse.
mchart(ID) [Höhe,Typ] ($Beschriftung1$,Style1, $Beschriftung2$,Style2, $Beschriftung3$,Style3, $Beschriftung4$,Style4)	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Höhe: Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF) Typ: Wert 8 (oder Konstante XY): Es wird eine Ortskurve generiert. Typ: Wert 9 (oder Konstante SXY): Es wird eine X-Y Darstellung gewählt, welche die X-Y Wertepaare vorsortiert (für die Ausgabe von Zeitreihen, z.B: mit Zeitsprung bei 24:00) $Beschriftung1$ .. $Beschriftung4$ Legende des entsprechenden Graphen Style1, Style2, Style3, ,Style4: Wert 0,1,2 oder 3 (oder Konstante LINE, DOTS, LINEDOTS, COLUMN).
mpchart(ID) [Höhe, Typ] ($Beschriftung1$,Style1, $Beschriftung2$,Style3, $Beschriftung3$,Style3, $Beschriftung4$,Style4)	ID : Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Höhe: Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF) Typ: Wert 8 (oder Konstante XY): Es wird eine Ortskurve generiert Typ: Wert 9 (oder Konstante SXY): Es wird eine X-Y Darstellung gewählt, welche die X-Y Wertepaare vorsortiert (für die Ausgabe von Zeitreihen, z.B: mit Zeitsprung bei 24:00) $Beschriftung1$ .. $Beschriftung4$ Legende des entsprechenden Graphen Style1, Style2, Style3, ,Style4: Wert 0,1,2 oder 3 (oder Konstante LINE, DOTS, LINEDOTS, COLUMN).
slider(ID)[Grafik]$Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile)
pslider(ID)[Grafik]$Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile)
eslider(ID)[Grafik] (Min,Inkrement, Max) $Beschriftung$ $Label$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile) Min: Minimalwert für den eslider Inkrement: Inkrement für den eslider Max: Maximalwert für den eslider Label: Label für die Werteanzeige, max. zwei Stellen
peslider(ID)[Grafik] (Min,Inkrement, Max) $Beschriftung$ $Label$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile) Min: Minimalwert für den peslider Inkrement: Inkrement für den peslider Max: Maximalwert für den peslider Label: Label für die Werteanzeige, max. zwei Stellen
picture(ID) [Höhe, Typ]($Beschriftung$, $www-LINK$)	ID : Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Höhe:Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF) Typ: Wert 0,1,2 (oder LEFTGRAF, CENTERGRAF, ZOOMGRAF): Linksbündiges, mittiges oder gestrecktes Einbinden der Grafik. www-Link: Gültig WWW-Adresse (inkl. Pfad und führenden http://) zur externen Grafik
link(ID)[Grafik][$Website$] $Text$	Verlinkung auf externe Seite

plink(ID)[Grafik] [PageID] $Text$	Verlinkung auf andere Seite des Webservers
frame[$Text$]	Einbetten einer externen HTML-Seite
dframe[$Text$]	Einbetten einer externen HTML-Seite. Das eingebettte Fenster ist doppelt so hoch, wie das von frame.
art(ID) [Format,Typ,Länge,YLMAX,YLMIN,YRLMAX, YRLMIN ($Beschriftung1$,ChartPos1,Buffer1, [$Beschriftung2$,ChartPos2,Buffer2, $Beschriftung3$,ChartPos3,Buffer3,$Beschr iftung4$,ChartPos4,Buffer4])	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Format: DOUBLE|TRIPLE|QUAD|LONG|EXTDOUBLE|EXTTRIPLE|EXTLONG Typ: 0 für autoscale der linken Achse, in diesem Fall wird YLMAX etc. ignoriert 1 für autoscale der rechten Achse, in diesem Fall wird YLMAX etc. ignoriert 2 für autoscale der beiden Achsen 3 für kein autoscale Länge: Maximale Anzahl an Wertepaaren, die pro Graph angezeigt werden können. YLMAX: Maximalwert linke y-Achse, Fließkommazahlen YLMIN: Minimalwert linke y-Achse, Fließkommazahlen YRMAX: Maximalwert rechte y-Achse, Fließkommazahlen YRMIN: Minimalwert rechte y-Achse, Fließkommazahlen $Beschriftung1$ .. $Beschriftung4$ Legende des entsprechenden Graphen ChartPos: Wert 0 oder 1 (0 für Beschriftung an der linken y-Achse, 1 für Beschriftung an der rechten y-Achse) Buffer: ID des mit dem jeweiligen Graphen verknüpften Timebuffers
webinput(ID)[Grafik] $Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext.
weboutput(ID)[Höhe]	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Höhe: Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF)
Webserver-Elemente Funktionalität	
button(ID)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines Webbuttons (z.B. button oder shifter) mit der ID geht die Funktion für einen Verarbeitungszyklus auf einen Wert ungleich 0. In allen anderen Fällen ist der Rückgabewert 0. Identisch zur Funktion webbutton früherer Releases.
pbutton(ID, PageID)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines seitenbezogenen Buttons (z.B. pbutton oder pshifter) mit der ID auf der Webseite mit der PageID geht die Funktion für einen Verarbeitungszyklus auf einen Wert ungleich 0. In allen anderen Fällen ist der Rückgabewert 0.
mbutton(ID, Auswahl)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines Multibuttons und der gegebenen Auswahl mit Index Auswahl (z.B. mbutton oder mshifter) mit der ID geht die Funktion für einen Verarbeitungszyklus auf den Wert Auswahl. Auswahl ist dabei ein Wert 1, 2 ... 254 und bezeichnet die Stelle im Auswahlfeld, an welcher das Auswahlelement steht. Wenn das Auswahlfeld betätigt wurde, ist der Rückgabewert 255. In allen anderen Fällen ist die Rückgabe 0.
mpbutton(ID, Auswahl, PageID)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines seitenbezogenen Multibuttons und der gegebenen Auswahl mit Index Auswahl (z.B. mpbutton oder mpshifter) mit der ID auf der Webseite mit der PageID geht die Funktion für einen Verarbeitungszyklus auf den Wert Auswahl. Auswahl ist dabei ein Wert 1, 2 ... 254 und bezeichnet die Stelle im Auswahlfeld, an welcher das Auswahlelement steht. Wenn das Auswahlfeld betätigt wurde, ist der Rückgabewert 255. In allen anderen Fällen ist die Rückgabe 0.

chart(ID, Var, X1, X2)	Die Funktion spricht das XY-Diagramm chart an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Bei Aufruf der Funktion wird die XY-Darstellung des Wertes Var aktiviert. X1, X2: Achsenbeschriftung
pchart(ID, Var, X1, X2, PageID)	Die Funktion spricht das XY-Diagramm pchart an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen. Bei Aufruf der Funktion wird die XY-Darstellung des Wertes Var aktiviert. X1, X2: Achsenbeschriftung
mchart(ID, VarX, VarY, Index)	Die Funktion spricht das Element mchart mit der ID an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Ein mchart stellt vier verschiedene Graphen dar. Index (0,1,2,3) gibt an, welcher Graph angesprochen wird. VarX, VarY: Achsenbeschriftung
mpchart(ID, VarX, VarY, Index,PageID)	Die Funktion spricht das seitenbezogene Element mpchart mit der ID auf der Webseite mit der PageID an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Ein mpchart stellt vier verschiedene Graphen dar. Index (0,1,2,3) gibt an, welcher Graph angesprochen wird. VarX, VarY: Achsenbeschriftung
display(ID, Text, Icon, State, TextStil, [Mbutton])	Die Funktion spricht den (button oder shifter) an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Identisch zur Funktion webdisplay früherer Releases.
pdisplay(ID, Text, Icon, State, TextStil, PageID, [Mbutton])	Die Funktion spricht den (pbutton oder pshifter) an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
getslider(ID)	Die Funktion spricht den slider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
getpslider(ID,PageID)	Die Funktion spricht den seitenbezogenen pslider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
geteslider(ID)	Die Funktion spricht den eslider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
getpeslider(ID, PageID)	Die Funktion spricht den seitenbezogenen peslider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
setslider(ID,Value,Icon, State)	Die Funktion spricht den slider an und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
setpslider(ID,Value,Icon, State, PageID)	Die Funktion spricht den seitenbezogenen slider an der ID auf der Seite PageID und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
seteslider(ID,Value,Icon, State)	Die Funktion spricht den eslider an und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
setpeslider(ID,Value,Icon, State,PageID)	Die Funktion spricht den seitenbezogenen peslider an der ID auf der Seite PageID und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
timebufferadd(ChartBufferID, Daten)	Es wird hier ein Wertepaar in den Zeitpuffer am Ende eingefügt.
timebufferconfig(ChartBufferID, MemTyp, Länge, DataTyp)	Es wird hier ein Wertepaarbuffer angelegt bzw. konfiguriert. Dabei kann mit dem Speichertyp festgelegt werden, ob dieser nach Befüllen mit den Werten vollläuft oder ob der jeweils älteste Wert verworfen wird.
timebufferstore(ChartBufferID)	Es wird ein Buffer dauerhaft im Flash abgelegt.
timebufferread(ChartBufferID)	Es wird ein Buffer aus dem Flash gelesen.
timebuffersize(ChartBufferID)	Den aktuellen Füllstand des Zeitpuffers ausgeben.
mtimechartpos(TimeChartID,ChartIdx,ChartBuffer,StartPos,EndPos)	Den darzustellenden Bereich eines Zeitpuffers für das Webelement vorgeben.

mtimechart(TimeChartID,ChartIdx,ChartBuffer,StartZeit,EndZeit)	Den darzustellenden Bereich eines Zeitpuffers für das Webelement vorgeben.
mtimechartupdate(TimeChartID,ChartIdx,ChartBuffer,Strategy)	Das Webelement updaten.
webinput	Liest die Daten des Webinputelementes aus.
weboutput	Gibt Daten an das Weboutputelement aus.
Synohr MultiSense Funktionen	
synohrmaster	

Schlüsselwörter - Referenz - Alphanumerische Sortierung

!Var	Bitweise Invertierung
abs(Var1)	Absolutwert
acos(Var1)	Arkuscosinus
address(Nummer)	Indirekte Adressierung mit Gruppenadressen
after(Steuerung, ms-Zeit)	Steuerung: Binärwert ms-Zeit: ZeitAngabe in ms (<2^{64})
afterc(Steuerung, Zeit, xT)	Steuerung: Binärwert ms-Zeit: ZeitAngabe in ms (<2^{64}) xT: verbleibende Zeit in ms
asin(Var1)	Arkussinus
atan(Var1)	Arkustangens
average(Var1, Var2, ... VarN)	Rückgabewert: Durchschnitt der angegebenen Variablen,die alle vom selben Datentyp sein müssen.
azimuth()	DieseFunktion berechnet zyklisch (Zeitraster: 5 Minuten) den Azimuthwinkel der Sonne in Grad, Nord über Ost
button(ID)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines Webbuttons (z.B. button oder shifter) mit der ID geht die Funktion für einen Verarbeitungszyklus auf einen Wert ungleich 0. In allen anderen Fällen ist der Rückgabewert 0. Identisch zur Funktion webbutton früherer Releases.
button(ID)[Grafik] $Text$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text: Ein statischer Beschriftungstext (erste Zeile).
callscene (GruppenadrSzenebaustein, Nummer)	Ein Szenebaustein soll seine Szene mit der entsprechenden Nummer aufrufen.
ceil(Var1)	Größte Ganzahl ≥ Var1
change(Var1)	Rückgabewert: 1b01 bei Änderung der überwachten Adresse oder Variable
changehour(Arg)	Ändert die aktuelle Zeit (Stunde)
changeminute(Arg)	Ändert die aktuelle Zeit (Minute)
changesecond(Arg)	Ändert die aktuelle Zeit (Sekunde)
chart(ID, Var, X1, X2)	Die Funktion spricht das XY-Diagramm chart an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Bei Aufruf der Funktion wird die XY-Darstellung des Wertes Var aktiviert. X1, X2: Achsenbeschriftung
chart(ID)[$Y0$,$Y1$,$Y2$]	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. $Y0$, $Y1$,$Y2$: Achsenbeschriftung der Y-Achse.
chtime(hh,mm,ss)	Tagesvergleichszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23)
closetcp(port, adresse)	Der Enertex® EibPC schließt eine TCP-Verbindung
closevpnuser(Name)	Userzugang schließen
cmtime(mm,ss)	Stundenvergleichszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59)
comobject(Var1, Var2, ... VarN)	Die Rückgabe ist der Wert der zuletzt veränderten Variablen oder Gruppenadresse.
connecttcp(port, adresse)	Der Enertex® EibPC baut eine TCP-Verbindung auf.
convert(Var1, Var2)	Konvertiert den Datentypen von Var1 in den von Var2 (Vorsicht: Es können Daten verloren gehen!).
cos(Var1)	Cosinus
cstime(ss)	Minutenvergleichszeitschaltuhr: ss: Sekunden (0..59)

cwtime(hh,mm,ss,dd)	Wochenvergleichszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23) dd: Tag (0=Sonntag, 6=Samstag)
cycle(mm,ss)	Der Rückgabewert ist ungleich null, wenn die Zeit eingetroffen ist. Wenn der Zeitpunkt eintrifft (und exakt übereinstimmt), geht der Ausgabewert auf 1. mm= Minuten (0...255) ss = Sekunden (0..59)
date(yyy,mm,dd)	Datumsvergleichszeitschaltuhr: yyy: Jahresdifferenz (0..255) vom Jahr 2000 an mm: Monat (1=Januar, 12= Dezember) dd: Tag (1..31)
day(dd)	Tagesvergleichszeitschaltuhr dd: Tag (1..31)
dayofweek()	Gibt den aktuellen Tag zurück (0=Sonntag, 6=Samstag)
delay(Signal, Zeit)	Die Funktion startet beim Übergang der Variablen Signal von AUS auf EIN einen Timer und setzt den Rückgabewert der Funktion auf EIN. Nach Ablauf der Zeit in ms springt der Ausgang wieder auf AUS
delayc(Signal, Zeit, xT)	Die Funktion startet beim Übergang der Variablen Signal von AUS auf EIN einen Timer und setzt den Rückgabewert der Funktion auf EIN. Nach Ablauf der Zeit in ms springt der Ausgang wieder auf AUS, in der Variable xT steht die verbleibende Zeit in ms
design $DESIGNSTRING$	$DESIGNSTRING$ kann mit $black$ oder $blue$ angegeben werden Das design-Kommando kann für jede Seite unterschiedlich angegeben werden.
devicenr()	Rückgabewert: Seriennummer des EibPCs
dframe[$Text$]	Einbetten einer externen HTML-Seite. Das eingebettte Fenster ist doppelt so hoch, wie das von frame.
display(ID, Text, Icon, State, TextStil, [Mbutton])	Die Funktion spricht den (button oder shifter) an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Identisch zur Funktion webdisplay früherer Releases.
eibtelegramm(Kontrollfeld, Zieladresse, Telegramminformation, Nutzinformation1 .. Nutzinformation18)	Es können eigenen EibTelegramme erstellt werden.
elevation()	Dieser Funktion berechnet zyklisch (Zeitraster: 5 Minuten) den Höhenwinkel der Sonne in Grad.
elog()	Auslesen des ältesten Eventspeicher Eintrags
elognum()	Gibt die Anzahl der Einträge im Fehlerspeicher zurück.
eslider(ID)[Grafik] (Min,Inkrement, Max) $Beschriftung$ $Label$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile) Min: Minimalwert für den eslider Inkrement: Inkrement für den eslider Max: Maximalwert für den eslider Label: Label für die Werteanzeige, max. zwei Stellen
eval(Var)	Unterbrechung des Valdidierungsschemas
event(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus ein Telegramm mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
event(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus ein Telegramm mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
event(readudp(...))	Rückgabewert: 1b01, wenn ein LAN-UDP-Telegramm eintrifft.
eventread(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNXTM Bus eine Leseanforderung mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
eventresponse(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus eine Antwort auf eine Leseanforderung mit der Gruppenadresse geschickt wird, unabhängig von dessen Inhalt.
eventwrite(Gruppenadresse)	Rückgabewert: 1b01 (EIN-Impuls), wenn am KNX™ Bus eine Antwort auf Telegramm mit der Gruppenadresse geschrieben wird, unabhängig von dessen Inhalt.

exp(Var1)	E-Funktion
find(String1, String2, Pos1)	String1: Zeichenkette, in der eine (Teil-)Zeichenkette gesucht werden soll. String2: zu findende Zeichenkette Pos1: Die ersten Pos1 - Vorkommen der zu findenden Zeichenkette ignorieren. Die Funktion gibt die Position des ersten Zeichens der gefundenen Zeichenkette (0..1399u16). Sie gibt 1400u16 zurück, falls die Zeichenkette nicht gefunden wurde. Für 65534u16 ist die Konstante END definiert.
floor(Var1)	Größte Ganzahl ≤ Var1
footer(Nummer) $WWW-Link$	Wenn Nummer den Wert 0 annimmt, wird Kopfzeile ausgeblendet. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Der Link (inkl. Pfad und führenden http://) ist optional. Dieser kann auf eine externe Quelle zugreifen. In diesem Fall muss für Nummer der Wert 2 gesetzt werden. Die Fußfzeile ist konfigurierbar, aber für alle Seiten gleich.
frame[$Text$]	Einbetten einer externen HTML-Seite
gaimage(Nummer)	Abbild einer Gruppenaddresse ermitteln.
getaddress(Gruppenaddresse)	Gruppenadresse in 16-Bit Zahl umwandeln.
getdate(Adresse)	Setzt das Datum des Enertex® EibPC neu
geteslider(ID)	Die Funktion spricht den eslider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
getpeslider(ID, PageID)	Die Funktion spricht den seitenbezogenen peslider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
getpslider(ID,PageID)	Die Funktion spricht den seitenbezogenen pslider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
getslider(ID)	Die Funktion spricht den slider an und gibt dessen Stellung (0 bis 255) zurück. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
gettime(Adresse)	Setzt die Zeit des Enertex® EibPC neu
gettimedate(Adresse)	Setzt die Zeit und das Datum des Enertex® EibPC neu
header(Nummer) $www.link$	Wenn Nummer den Wert 0 annimmt, wird Kopfzeile ausgeblendet. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Der Link (inkl. Pfad und führenden http://) ist optional. Dieser kann auf eine externe Quelle zugreifen. In diesem Fall muss für Nummer der Wert 2 gesetzt werden. Die Kopfzeile ist konfigurierbar, aber für alle Seiten gleich.
hour()	Gibt die aktuelle Zeit (Stunde) als Datentyp u08 zurück.
htime(hh,mm,ss)	Tageszeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23)
Hysteresis (Var,UnteresLimit,OberesLimit)	Das Argument Var wird mit den Variablen UnteresLimit und OberesLimit mit einer Hysteresefunktion verglichen.
if ... then ... else ... endif	Wenn – Dann – Sonst
if ... then ... endif	Wenn – Dann
initga(Addresse)	Äquivalent zum Eintrag der Gruppenadresse in [InitGA]
Line $Text$	Das Element fügt eine Trennlinie zwischen zwei Zeilen ein. Der Text wird an die Trennlinie gebunden und ist optional.
link(ID)[Grafik][$Website$] $Text$	Verlinkung auf externe Seite
log(Var1, Var2)	Logarithmus: Var1 = Basiszahl Var2 = Hochzahl
max(Var1, Var2, ... VarN)	Rückgabewert: Das Maximum der angegebenen Variablen, die alle vom selben Datentyp sein müssen.
mbutton(ID, Auswahl)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines Multibuttons und der gegebenen Auswahl mit Index Auswahl (z.B. mbutton oder mshifter) mit der ID geht die Funktion für einen Verarbeitungszyklus auf den Wert Auswahl. Auswahl ist dabei ein Wert 1, 2 ... 254 und bezeichnet die Stelle im Auswahlfeld, an welcher das Auswahlelement steht. Wenn das Auswahlfeld betätigt wurde, ist der Rückgabewert 255. In allen anderen Fällen ist die Rückgabe 0.

mbutton(ID)[$Text1$,$Text2$,... $Text254$] [Grafik] $Beschriftung$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text1, Text2, .. Text254: Beschriftungsfelder für den mbutton. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).
mchart(ID, VarX, VarY, Index)	Die Funktion spricht das Element mchart mit der ID an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Ein mchart stellt vier verschiedene Graphen dar. Index (0,1,2,3) gibt an, welcher Graph angesprochen wird. VarX, VarY: Achsenbeschriftung
mchart(ID) [Höhe,Typ] ($Beschriftung1$,Style1, $Beschriftung2$,Style2, $Beschriftung3$,Style3, $Beschriftung4$,Style4)	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Höhe: Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF) Typ: Wert 8 (oder Konstante XY): Es wird eine Ortskurve generiert. Typ: Wert 9 (oder Konstante SXY): Es wird eine X-Y Darstellung gewählt, welche die X-Y Wertepaare vorsortiert (für die Ausgabe von Zeitreihen, z.B: mit Zeitsprung bei 24:00) $Beschriftung1$ .. $Beschriftung4$ Legende des entsprechenden Graphen Style1, Style2, Style3, ,Style4: Wert 0,1,2 oder 3 (oder Konstante LINE, DOTS, LINEDOTS, COLUMN).
min(Var1, Var2, ... VarN)	Rückgabewert: Das Minimum der angegebenen Variablen, die alle vom selben Datentyp sein müssen.
minute()	Gibt die aktuelle Zeit (Minute) als Datentyp u08 zurück.
mod(Var1,Var2)	Ergebnis=Var1 modulo Var2
month(mm,dd)	Monatsvergleichszeitschaltuhr: mm: Monat (1=Januar, 12= Dezember) dd: Tag (1..31)
mpbutton(ID, Auswahl, PageID)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines seitenbezogenen Multibuttons und der gegebenen Auswahl mit Index Auswahl (z.B. mpbutton oder mpshifter) mit der ID auf der Webseite mit der PageID geht die Funktion für einen Verarbeitungszyklus auf den Wert Auswahl. Auswahl ist dabei ein Wert 1, 2 ... 254 und bezeichnet die Stelle im Auswahlfeld, an welcher das Auswahlelement steht. Wenn das Auswahlfeld betätigt wurde, ist der Rückgabewert 255. In allen anderen Fällen ist die Rückgabe 0.
mpbutton(ID)[$Text1$,$Text2$,... $Text254$] [Grafik] $Beschriftung$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text1, Text2, .. Text254: Beschriftungsfelder für den mpbutton. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).
mpchart(ID, VarX, VarY, Index,PageID)	Die Funktion spricht das seitenbezogene Element mpchart mit der ID auf der Webseite mit der PageID an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen. Ein mpchart stellt vier verschiedene Graphen dar. Index (0,1,2,3) gibt an, welcher Graph angesprochen wird. VarX, VarY: Achsenbeschriftung
mpchart(ID) [Höhe, Typ] ($Beschriftung1$,Style1, $Beschriftung2$,Style3, $Beschriftung3$,Style3, $Beschriftung4$,Style4)	ID : Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Höhe: Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF) Typ: Wert 8 (oder Konstante XY): Es wird eine Ortskurve generiert Typ: Wert 9 (oder Konstante SXY): Es wird eine X-Y Darstellung gewählt, welche die X-Y Wertepaare vorsortiert (für die Ausgabe von Zeitreihen, z.B: mit Zeitsprung bei 24:00) $Beschriftung1$ .. $Beschriftung4$ Legende des entsprechenden Graphen Style1, Style2, Style3, ,Style4: Wert 0,1,2 oder 3 (oder Konstante LINE, DOTS, LINEDOTS, COLUMN).
mpshifter(ID)[$Text1$,$Text2$,...,$Text254$] [Grafik1,Grafik2, Grafik3, Grafik4] $Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text1, Text2, .. Text254: Beschriftungsfelder für den mpshifter. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).

mshifter(ID)[$Text1$,$Text2$,...,$Text254$] [Grafik1,Grafik2, Grafik3, Grafik4] $Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text1, Text2, .. Text254: Beschriftungsfelder für den mshifter. Ab dem 2. Element sind die Elemente optional. Beschriftung: Ein statischer Beschriftungstext (erste Zeile).
mtime(mm,ss)	Stundenzeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59)
none	Ein leeres Element einfacher Breite wird im Webserver eingefügt.
openvpnuser(Name)	Userzugang öffnen
page(ID)[$Gruppe$,$Name$]	ID: Wert zwischen 1 bis 100 als Seitenindex für die Programmierung und den Zugriff auf lokale Seitenelemente (Anfangsbuchstabe „p“). Gruppe: Eine Zuordnung der Seite zu einer Gruppe. Name: Ein statischer Beschriftungstext
pbutton(ID, PageID)	Bei „Betätigen“ bzw. Drücken der Schaltfläche eines seitenbezogenen Buttons (z.B. pbutton oder pshifter) mit der ID auf der Webseite mit der PageID geht die Funktion für einen Verarbeitungszyklus auf einen Wert ungleich 0. In allen anderen Fällen ist der Rückgabewert 0.
pbutton(ID)[Grafik] $Text$	ID: Wert von 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Text: Ein statischer Beschriftungstext (erste Zeile).
pchart(ID, Var, X1, X2, PageID)	Die Funktion spricht das XY-Diagramm pchart an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen. Bei Aufruf der Funktion wird die XY-Darstellung des Wertes Var aktiviert. X1, X2: Achsenbeschriftung
pchart(ID)[$Y0$,$Y1$,$Y2$]	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. $Y0$, $Y1$,$Y2$: Achsenbeschriftung der Y-Achse.
pdisplay(ID, Text, Icon, State, TextStil, PageID, [Mbutton])	Die Funktion spricht den (pbutton oder pshifter) an. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
peslider(ID)[Grafik] (Min,Inkrement, Max) $Beschriftung$ $Label$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile) Min: Minimalwert für den peslider Inkrement: Inkrement für den peslider Max: Maximalwert für den peslider Label: Label für die Werteanzeige, max. zwei Stellen
picture(ID) [Höhe, Typ]($Beschriftung$, $www-LINK$)	ID : Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Höhe:Wert 0, 1 oder 2 (bzw. Konstante SINGLE, DOUBLE und HALF) Typ: Wert 0,1,2 (oder LEFTGRAF, CENTERGRAF, ZOOMGRAF): Linksbündiges, mittiges oder gestrecktes Einbinden der Grafik. www-Link: Gültig WWW-Adresse (inkl. Pfad und führenden http://) zur externen Grafik
ping(adresse)	Der Enertex EibPC führt einen ping auf die angegeben Adresse durch
plink(ID)[Grafik] [PageID] $Text$	Verlinkung auf andere Seite des Webservers
pow(Var1, Var2)	Potenz: Var1= Basiszahl Var2= Exponent

presetscene(GruppenadresseSzenebaustein, SzenenNummer, OptionOverwrite, WertVariable1,StatusWertVariable1, [WerVariable2,StatusWertVariable2,Variable Wertn, StatusVariableWertn)	Vorbelegung für den Szenenaktor mit der Gruppenadresse GruppenadresseSzenebaustein und entsprechender SzenenNummer erstellen.
pshifter(ID)[Grafik1,Grafik2, Grafik3, Grafik4]$Text$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text: Ein statischer Beschriftungstext (erste Zeile).
pslider(ID)[Grafik]$Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile)
random(Max)	Gibt eine Zufallszahl im Bereich von 0 bis Max zurück.
read(Gruppenadresse)	Auf den Bus wird ein EIB-Telegramm mit Leseanforderung an die Gruppenadresse generiert. Falls die Aktoren etc. antworten, wird das Ergebnis als Rückgabewert der Funktion ermittelt.
readknx(Adr,String)	Umwandlung eines Telegramms am Bus in Adresse und dezidierten Informationen
readrawknx(Kontrollfeld, PhyAdresse, Zieladresse, IsGroubAdress, RoutingZähler, BitLänge, Nutzdaten)	Umwandlung eines Telegramms am Bus in Adresse und dezidierten Informationen
readrs232(data,len)	Der Enertex® EibPC empfängt RS232-Telegramme (nur bei IP-Ankopplung an den KNX™ Bus).
readtcp(port, adresse, arg 1 [, arg2, ... arg n]	Der Enertex® EibPC empfängt TCP-Telegramme.
readudp (port, adresse, arg 1 [, arg2, ... arg n])	Der Enertex® EibPC empfängt UDP-Telegramme.
resolve(hostename)	Die Funktion ermittelt die IP-Adresse des angegebenen Hostenamens.
scene (GruppenadrSzenebaustein, GruppenadresseAktor1, ... GruppenadresseAktorN)	Es wird ein KNX™ Szenenaktor mit der Gruppenadresse GruppenadrSzenebaustein erstellt.
second()	Gibt die aktuelle Zeit (Sekunde) als Datentyp u08 zurück.
sendmail(empfängeradresse, betreff, nachricht)	Es wird an die Empfängeradresse (Zeichenkette) eine E-Mail mit dem Betreff und der Nachricht verschickt. Alle Zeichenketten werden auf 255 Zeichen begrenzt, auch wenn diese mehr Zeichen beinhalten. Rückgabewert: 0 = E-Mail erfolgreich versendet 1 = in Bearbeitung 2 = Fehler
sendrs232(arg1 [, arg2, ... arg n])	Der Enertex® EibPC sendet RS232-Telegramme (nur bei IP-Ankopplung an den KNX™ Bus)
sendtcp(port, adresse, arg 1 [, arg2, ... arg n]	Der Enertex® EibPC verschickt TCP-Telegramme.
sendtcparray(port, adresse, arg 1 [, arg2, ... arg n, size]	Der Enertex® EibPC verschickt TCP-Telegramme, ohne Nullterminierung.
sendudp(port, adresse, arg 1 [, arg2, ... arg n], size)	Der Enertex® EibPC verschickt UDP-Telegramme, ohne Nullterminierung.
sendudp(port, adresse, arg 1 [, arg2, ... arg n])	Der Enertex® EibPC verschickt UDP-Telegramme.
setdate()	Schreibt das Datum des Enertex® EibPC auf den KNX Bus.

seteslider(ID,Value,Icon, State)	Die Funktion spricht den eslider an und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
setpeslider(ID,Value,Icon, State,PageID)	Die Funktion spricht den seitenbezogenen peslider an der ID auf der Seite PageID und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
setpslider(ID,Value,Icon, State, PageID)	Die Funktion spricht den seitenbezogenen slider an der ID auf der Seite PageID und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID auf der Webseite mit der PageID angesprochen.
setslider(ID,Value,Icon, State)	Die Funktion spricht den slider an und stellt diesen auf den Wert von Value. Wenn ID mehrfach vorhanden ist, dann werden alle Elemente dieser ID angesprochen.
settime()	Schreibt die Zeit des Enertex® EibPC auf den KNX Bus
settimedate()	Schreibt das Datum und die Zeit des Enertex® EibPC auf den KNX Bus
shifter(ID)[Grafik1,Grafik2, Grafik3, Grafik4]$Text$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik 1 bis Grafik 4: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Grafik 2 bis Grafik 4 sind optional. Wenn nur drei Grafiken angegeben werden, so hat das Element nur drei Knöpfe usw.. Text: Ein statischer Beschriftungstext (erste Zeile).
sin(Var1)	Sinus
size(String)	Von der Zeichenkette String soll die Länge bestimmt werden. Die Länge ist durch das Terminierungszeichen „\0“ am Ende von Zeichenketten gegeben.
Sleep(Status)	Bei Status == EIN wird der Enertex® EibPC in den Passivmodus versetzt.
slider(ID)[Grafik]$Beschriftung$	ID: Wert zwischen 0 bis 59 als Index für die Programmierung und den Zugriff auf dieses Element. Sie können auch auf u08 Variablendefinitionen in der Sektion [EibPC] zugreifen. Grafik: Wert zwischen 0 und 99. Um die Anwendung übersichtlicher zu gestalten, sind vordefinierte Konstanten definiert (Seite 300). Beschriftung: Ein statischer Beschriftungstext (erste Zeile)
split(String, Pos1, Pos2)	String: Zeichenkette, aus der eine Zeichenkette entnommen werden soll Pos1: erstes Zeichen der abzutrennenden Zeichenkette (0...1399u16) Pos2: letztes Zeichen der abzutrennenden Zeichenkette (0...1399u16). Wenn Pos2 gleich 65534u16 (vordefinierte Konstante END) ist, so wird die Zeichenkette bis zu deren Ende abgetrennt. Die Variable String muss vom Datentyp c1400 sein. Rückgabewert: die von Variable String abgetrennte Zeichenkette
sqrt(Var1)	Wurzel
startvpn()	VPN Service starten
stime(ss)	Minutenzeitschaltuhr: ss: Sekunden (0..59)
stopvpn()	VPN Service stoppen
storescene (GruppenadrSzenebaustein, Nummer)	Ein Szenebaustein soll seine Szene mit der entsprechenden Nummer neu abspeichern.
String1 + String2 [+ String3 ... String n]	Die Strings werden aneinander gefügt. Übersteigt die resultierende Länge die maximale Länge des Datentyps, so werden diese Zeichen „abgeschnitten“.
stringcast(String, Data, Pos)	Interpretation von Binärdaten eines c1400 Strings.
stringformat(Data, Umwandlungtyp, Format, Feldbreite,[Präzision])	Umwandlung von Zahlen in c1400 Strings unter Berücksichtigung verschiedener Formate.
stringset(String, Data, Pos)	Schreiben von Binärdaten in einen c1400 String
sun()	Gibt aus, ob es Tag oder Nacht ist (Sonnenstand). Das Programm muß dazu die geographische Länge und Breite des betreffenden Ortes kennen.
sunrisehour()	Stunde des Sonnenaufgangs am akutellem Tag
sunriseminute()	Minute des Sonnenaufgangs am akutellem Tag
sunsethour()	Stunde des Sonnenuntergangs am akutellem Tag
sunsetminute()	Minute des Sonnenuntergangs am akutellem Tag
systemstart()	Beim Systemstart werden alle angegebenen Variablen aktualisiert.
tan(Var1)	Tangens

Var1 – Var2 - VarN	Subtraktion
Var1 != Var2	Größenvergleich
Var1 * Var2 * VarN	Multiplikation
Var1 / Var2 / VarN	Division
Var1 + Var2 + VarN	Addition
Var1 < Var2	Größenvergleich
Var1 =< Var2	Größenvergleich
Var1 == Var2	Größenvergleich
Var1 > Var2	Größenvergleich
Var1 >= Var2	Größenvergleich
Var1 and Var2	Bitweises And
Var1 or Var2	Bitweises Oder
Var1 xor Var2	Bitweises Exclusiv Oder
webbutton(Index)	Gibt das Ereignis aus, welches das Webelement mit Index (0...255) bei dessen Betätigung ausgelöst hat.
webchart(ID, Var, X1, X2)	Die Funktion spricht das XY-Diagramm chart an. Bei Aufruf wird die XY-Darstellung des Wertes Var aktiviert. ID, Var vom Datentyp u08 X1, X2 Datentyp c14
webdisplay(Index,Text,Grafik)	Schreibt auf das Webelement Index (0...255) den Text und setzt die Grafik. Die verfügbaren Standard-Grafiken sind codiert als Predefine (rechts unten im Enertex® Enertex® EibStudio) wählbar.
write(Gruppenadresse, Wert)	Auf dem Bus wird ein gültiges EIB-Telegramm generiert.
writeresponse(Gruppenadresse, Wert)	Antwortet auf eine Leseanforderung, indem ein gültiges KNXTM Telegramm generiert wird, welches den Wert auf die Gruppenadresse schreibt.
wtime(dd,hh,mm,ss)	Wochenzeitschaltuhr: ss: Sekunden (0..59) mm: Minuten (0..59) hh: Stunden (0..23) dd: Tag (0=Sonntag, 6=Samstag)

Predefines

Der Compiler des Enertex® EibStudios kennt vordefinierte Konstanten. Diese finden Sie im Fenster „Definitionen“ (vgl. Abbildung 85, Markierung E). Diese können Sie nutzen um die Symbolgruppen des Webservers besser zu nutzen, den Stil der Grafiken festzulegen oder übersichtlich programmieren zu können.

Fragen und Antworten

Fehlercodes Ereignisspeicher

Fehlercode	Bedeutung
ERR_PROC_OBJECT	Ein Objekt (eine Funktion) konnte nicht verarbeitet werden. Dies kann verschiedene, funktionsspezifische Ursachen haben. Bitte achten Sie auf weitere Fehlermeldungen.
ERR_PROC_OBJECT_MSG_OUT	Ein Ausgabeobjekt konnte nicht verarbeitet werden. Dies kann die folgenden Funktionen betreffen: 1 Schreibzugriff auf den KNX-Bus 1.1 settime 1.2 setdate 1.3 settimedate 1.4 write 1.5 read 1.6 writeresponse 1.7 scene 1.8 storescene 1.9 callscene 1.10 eibtelegramm 2 Netzwerkfunktionen 2.1 closetcp 2.2 connecttcp 2.3 ping 2.4 resolve 2.5 sendhtmlmail 2.6 sendmail 2.7 sendtcp 2.8 sendtcparray 2.9 sendudp 2.10 sendudparray 3. RS232-Schnittstelle 3.1 resetrs232 3.2 sendrs232 4. VPN-Server 4.1 closevpnuser 4.2 openvpnuser 4.3 startvpn 4.4 stopvpn Bitte überprüfen Sie, ob eine entsprechende Verbindung besteht.
ERR_PROC_REPETITIONS	Eine Endlosschleife ist erkannt worden. Die Verarbeitung wurde deshalb abgebrochen.
ERR_POW_OF_NEG_BASE	Bei der Verarbeitung der Funktion pow wurde der Fehler erkannt, dass die Basis negativ ist. Die Berechnung wurde deshalb nicht durchgeführt.
ERR_LOG_OF_NON_POS_BASE_OR_ARG	Bei der Verarbeitung der Funktion log wurde der Fehler erkannt, dass die Basis oder das Argument nicht positiv ist. Die Berechnung wurde deshalb nicht durchgeführt.
ERR_SQRT_OF_NON_POS_ARG	Bei der Verarbeitung der Funktion sqrt wurde der Fehler erkannt, dass das Argument negativ ist. Die Berechnung wurde deshalb nicht durchgeführt.
ERR_ASIN_OF_ARG_OUT_OF_RANGE	Bei der Verarbeitung der Funktion asin wurde der Fehler erkannt, dass das Argument außerhalb des Intervalls [-1; +1] liegt. Die Berechnung wurde deshalb nicht durchgeführt.
ERR_ACOS_OF_ARG_OUT_OF_RANGE	Bei der Verarbeitung der Funktion acos wurde der Fehler erkannt, dass das Argument außerhalb des Intervalls [-1; +1] liegt. Die Berechnung wurde deshalb nicht durchgeführt.
ERR_DIVISION_BY_ZERO	Bei der Verarbeitung einer Division wurde der Fehler erkannt, dass der Divisor gleich 0 ist. Die Berechnung wurde deshalb nicht durchgeführt.
ERR_EIBNET_IP_SETSOCKOPT_0	Es ist ein Fehler bei der Vorbereitung der Verbindung zu einer KNXnet/IP-Schnittstelle aufgetreten.
ERR_EIBNET_IP_SETSOCKOPT_1	s.o.
ERR_EIBNET_IP_SETSOCKOPT_2	s.o.
ERR_EIBNET_IP_SENDTO_0	Es ist ein Fehler beim Senden einer Nachricht an eine KNXnet/IP-Schnittstelle aufgetreten.
ERR_EIBNET_IP_SENDTO_1	s.o.
ERR_EIBNET_IP_SENDTO_2	s.o.
ERR_EIBNET_IP_SENDTO_3	s.o.
ERR_EIBNET_IP_SENDTO_4	s.o.
ERR_EIBNET_IP_SENDTO_5	s.o.
ERR_EIBNET_IP_TIMEOUT_SEARCH	Es konnte keine KNXnet/IP-Schnittstelle gefunden werden. Bitte überprüfen Sie, ob eine betriebsbereite KNXnet/IP-Schnittstelle mit demselben Netzwerk wie der EibPC verbunden ist.
ERR_EIBNET_IP_DISCONNECT_REQUEST_IN	Die Verbindung zwischen EibPC und KNXnet/IP-Schnittstelle wurde getrennt.
ERR_EIBNET_IP_DISCONNECT_REQUEST_OUT	s.o.
ERR_EIBNET_IP_TIMEOUT_CONNECTIONSTATE_REQUEST	s.o.
ERR_EIBNET_IP_E_CONNECTION_ID	s.o.
ERR_EIBNET_IP_E_DATA_CONNECTION	Die KNXnet/IP-Schnittstelle hat einen Fehler der Verbindung zum EibPC festgestellt.
ERR_EIBNET_IP_E_KNX_CONNECTION	Die KNXnet/IP-Schnittstelle hat einen Fehler der Verbindung zum KNX-Bus festgestellt.

ERR_EIBNET_IP_TUNNELLING_TIMEOUT_0	Eine Nachricht wurde erneut zur KNXnet/IP-Schnittstelle gesendet, weil ein Fehler aufgetreten ist.
ERR_EIBNET_IP_TUNNELLING_TIMEOUT_1	Die Verbindung zwischen EibPC und KNXnet/IP-Schnittstelle wurde getrennt.
ERR_EIBNET_IP_L_DATA_CON	Es wurde eine Bestätigung von der KNXnet/IP-Schnittstelle für eine an diese gesendete Nachricht empfangen.
ERR_FT12_LINE_IDLE_TIMEOUT_0	Es ist ein Fehler bei der Verbindung zur FT1.2-Schnittstelle aufgetreten.
ERR_FT12_LINE_IDLE_TIMEOUT_1	s.o.
ERR_FT12_SELECT	s.o.
ERR_FT12_INVALID_TELEGRAM	s.o.
ERR_FT12_READ	s.o.
ERR_FT12_RESET_REQ_IN	Die Verbindung zur FT1.2-Schnittstelle wurde zurückgesetzt.
ERR_FT12_STATUS_REQ_IN	Es wurde eine Statusanfrage von der FT1.2-Schnittstelle empfangen.
ERR_FT12_L_BUSMON_IND	Es wurde eine Nachricht vom KNX-Bus über die FT1.2-Schnittstelle empfangen.
ERR_FT12_FIX_LENGTH_END	Eine Nachricht von der FT1.2-Schnittstelle war fehlerhaft.
ERR_FT12_FIX_LENGTH_CHECKSUM	s.o.
ERR_FT12_VAR_LENGTH_LENGTH_0	s.o.
ERR_FT12_VAR_LENGTH_LENGTH_1	s.o.
ERR_FT12_VAR_LENGTH_START	s.o.
ERR_FT12_VAR_LENGTH_CHECKSUM	s.o.
ERR_FT12_VAR_LENGTH_END	s.o.
ERR_FT12_L_DATA_CON	Es wurde eine Bestätigung von der FT1.2-Schnittstelle für eine an diese gesendete Nachricht empfangen.
ERR_FT12_IN_BUFFER_FULL	Es ist ein Fehler bei der Verbindung zur FT1.2-Schnittstelle aufgetreten.
ERR_MEM_OBJECTS_COUNT	Obsolet in V3
ERR_MEM_OBJECT_OBJECT_TYPE	Obsolet in V3
ERR_MEM_OBJECT_CALC_TYPE	Obsolet in V3
ERR_MEM_OBJECT_BIT_LEN	Obsolet in V3
ERR_MEM_OBJECT_DATA_SIZE	Obsolet in V3
ERR_MEM_OBJECT_NAME	Obsolet in V3
ERR_MEM_OBJECT_EXPRESSION	Obsolet in V3
ERR_MEM_OBJECT_INPUT_COUNTER_0	Obsolet in V3
ERR_MEM_OBJECT_INPUTS_0	Obsolet in V3
ERR_MEM_OBJECT_DEPENDENCY_COUNTER_0	Obsolet in V3
ERR_MEM_OBJECT_DEPENDENCIES_0	Obsolet in V3
ERR_MEM_OBJECT_DEPENDENCY_COUNTER_1	Obsolet in V3
ERR_MEM_OBJECT_DEPENDENCIES_1	Obsolet in V3
ERR_MEM_OBJECT_NULL	Obsolet in V3
ERR_MEM_OBJECT_NO_ERROR	Obsolet in V3
ERR_MSGSND_ASYNC_SERIAL_0	Ein Fehler bei der Kommunikation mit der asynchronen seriellen Benutzer-Schnittstelle wurde festgestellt, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_ASYNC_SERIAL_1	s.o.
ERR_MSGSND_MSGOUT_0	Der Zugriff auf den KNX-Bus war nicht möglich, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.

ERR_MSGSND_MSGOUT_1	s.o.
ERR_MSGSND_MSGOUT_2	s.o.
ERR_MSGSND_MSGOUT_3	s.o.
ERR_MSGSND_MSGOUT_4	s.o.
ERR_MSGSND_MSGOUT_5	s.o.
ERR_MSGSND_RESOLVE_0	Die Funktion resolve konnte nicht ausgeführt werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_INTERFACE_IN_0	Eine vom KNX-Bus empfangene Nachricht konnte nicht an das Anwendungsprogramm übergeben werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_INTERFACE_IN_1	s.o.
ERR_MSGSND_INTERFACE_IN_2	s.o.
ERR_MSGSND_MAIL_0	Eine E-Mail-Nachricht konnte nicht versendet werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_MAIL_1	s.o.
ERR_MSGSND_TCP_OUT_0	Eine TCP-Nachricht konnte nicht versendet werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_TCP_OUT_1	Eine TCP-Verbindung konnte nicht hergestellt werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_TCP_OUT_2	Eine TCP-Verbindung konnte nicht getrennt werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_TCP_IN_0	Eine empfangene TCP-Nachricht konnte nicht an das Anwendungsprogramm übergeben werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_UDP_OUT_0	Eine UDP-Nachricht konnte nicht versendet werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_UDP_IN_0	Eine empfangene UDP-Nachricht konnte nicht an das Anwendungsprogramm übergeben werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_PING_0	Die Funktion ping konnte nicht ausgeführt werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_TCP_OUT_3	Eine TCP-Nachricht ohne Nullterminierung konnte nicht versendet werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_UDP_OUT_1	Eine UDP-Nachricht ohne Nullterminierung konnte nicht versendet werden, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.
ERR_MSGSND_ASYNC_SERIAL_2	Ein Fehler bei der Kommunikation mit der asynchronen seriellen Benutzer-Schnittstelle wurde festgestellt, weil eine interne Warteschlange nicht verfügbar war. Möglicherweise ist der EibPC mit dem aktuellen Anwendungsprogramm zeitweise überlastet.

ERR_EXIT_NCONF_0	Das Anwendungsprogramm wurde beendet. Dieser Vorgang wurde durch eine Aktion im EibStudio ausgelöst.
ERR_EXIT_NCONF_1	s.o.
ERR_EXIT_NCONF_2	s.o.
ERR_EXIT_NCONF_3	s.o.
ERR_EXIT_MAIN_0	Das Anwendungsprogramm wurde aufgrund eines internen Fehlers beendet.
ERR_EXIT_MAIN_1	Das Anwendungsprogramm wurde aufgrund eines internen Fehlers beendet.
ERR_EXIT_MAIN_2	Das Anwendungsprogramm wurde aufgrund eines internen Fehlers beendet.
ERR_EXIT_MAIN_3	Das Anwendungsprogramm wurde aufgrund eines internen Fehlers beendet.
ERR_EXIT_MAIN_4	Das Anwendungsprogramm wurde aufgrund eines internen Fehlers beendet.
ERR_LED_MUTEX_TRYLOCK	Obsolet in V3
ERR_READ_GROUP_ADDRESS	Eine Gruppenaddresse wurde mit initga parametriert, antwortet jedoch nicht auf die Leseanforderung.
ERR_ERRNO	Ein interner Fehler wurde erkannt. Der Fehlertyp kann durch den Hersteller anhand des Fehlercodes genauer bestimmt werden.
ERR_ASYNC_SERIAL_0	Es ist ein Fehler beim Zugriff auf die asynchrone serielle Benutzer-Schnittstelle aufgetreten.
ERR_ASYNC_SERIAL_1	s.o.
ERR_ASYNC_SERIAL_2	s.o.
TIMEBUFFER_DATATYPE_ERROR	Obsolet in V3
TIMEBUFFER_DATATYPE_ERROR	Obsolet in V3
TIMEBUFFER_DATATYPE_ERROR	Obsolet in V3

Probleme und Lösungen

Fehlermeldung	**Lösung**
! Vorgabewert zu groß für gegebenen Datentyp in >xy< !	Wert muss mit Datentyp angegeben werden z.b. Helligkeit<2000u16
! Benutzten Sie Konvertierungsfunktionen: Datentyp der Parameter nicht gleich: >Var1+Var2< !	Var3=convert(Var1,Var2) + Var2
Syntaxfehler in Zeile:[17] >if (("KücheGesamt-1/1/9"==Ein) and wtime(23,00,00,00)) < Gültig bis Position: STOP--> and wtime(23,00,00,00))	Anweisung muss in einer Zeile stehen oder die Zeile muss mit ' \\' abgeschlossen werden. if ..... and \\ then ....
! Vordefinierte Variable kann nicht neu definiert werden in >EIN=1b01< !	Im EibParser sind Variablen vordefiniert, um das Erstellen eines EibProgramms möglichst einfach zu gestalten. Die Predefines werden im Enertex® EibStudio im rechten Bildteil aufgeführt. Sie können nicht neu definiert werden.
Datentyp der Parameter nicht gleich: >sun()==1< !	Der Rückgabewert der Funktion ist binär. Eine Zahl ohne Datentyangabe ist immer ein 8bit Wert ohne Vorzeichen. Als Vergleichsoperator muss ein Binärwert angegeben werden. sun()==1b01
Syntaxfehler in Zeile:[13] >a=4,6e1f32< Gültig bis Position: STOP--> ,6e1f32	Als Dezimaltrennzeichen muss immer ein “.“ verwendet werden.
Syntaxfehler in Zeile:[21] >"Akt1-0/0/5"=after(a,5000u64)<	Eine direkte Zuweisung ist nur bei Variablen, nicht bei Adressen möglich. Das Schreiben einer Information auf den KNX™-Bus wird mit der Hilfe der write-Funktion realisiert. write(„Akt1-0/0/5“, 1b01)
Syntaxfehler in Zeile:[19] >if (a==EIN) then write("Akt1-0/0/5",EIN) write("Akt2-0/0/6",EIN);write("Akt3-0/0/8",EIN); write("Ak4-0/0/7",EIN) endif<	Mehrere Anweisungen in einer if-Anweisung müssen mit “;“ getrennt werden. if(a=EIN) then write(b=EIN); write(c=AUS) endif

Syntaxfehler in Zeile:[26] >write(ein,EIN)< Ungültiger Datentyp in >write(ein<	Die write-Funktion kann nur auf Gruppenadressen wirken (1. Argument), nicht aber Variablen.
Deklaration der Variable muss eindeutig sein in >u=convert(z,r)-r-e<	Jede Variable darf nur einmal deklariert werden. Eine weitere Deklaration bringt diese Fehlermeldung.
Falscher Datentyp in >cycle(0.5,5<	Es dürfen nur ganzzahlige Werte eingegeben werden.

ChangeLog

Version 31 (ab Patch 3.100), EibStudio 3.103)

- Belilebige Größen für das weboutput-Feld (S. 294)

Version 30 (ab Patch 3.100), EibStudio 3.100)

- Online Debuggen zur Laufzeit – direkt im Code S. 326
- Mpchart mit 4-facher Breite (LONG) S. 264
- timechart mit Option von Stapeln der Grafen (STACK) S. 290
- Ergänzung zu connecttcp S. 248
- compact von Webelementen / Ausfüllen von Freiräumen im Webserver - S. 281.
- Farbrad – RGB Wahl mit dem Webserver/ webinput S. 294
- Zeiteingabe mit dem Webserver/ webinput S. 294
- Datumeingabe mit dem Webserver/ webinput S. 294
- Passworteingabe mit dem Webserver/ webinput S. 294
- Neue Funktion easterday S. 188
- Neue Funktion eastermonth S. 189
- SHUTDOWN Variable für die Steuerung von Speichervorgängen im Anwenderprogramm S. 221.
- Neue Funktion timebuffervalue S. 271
- Neue Funktion timebufferclear S. 270
- Neue Funktion writeflashvar S. 205.
- Neue Funktion readflashvar S. 204
- Ausbessern Handbuch-Errata
- Kommunikations-Ports UDP und TCP des Enertex® EibPC verändern (S. 245)
- Neue Größen für weboutput (S. 294)
- Neues Webserver-Element timechartcolor (S. 291)
- Neues Webserver-Element (S. 291)
- Neue Funktion tostring S. 235.
- Graf mit Graph ersetzt
- Beschreibung der mtimechartpos Funktion verbessert

Version 28 (ab Patch 3.018, EibStudio 3.010)

- Windows 7 und Windows 8.1 und Firewalls S. 21.
- Vollständiges Webserver-Beispiel für die Gebäudeautomation S. 90
- Neue Funktion processingtime S.219.
- Neue Funktion urlencode S. 236.
- Neue Funktion urldecode S. 236.
- Neue Funktion getganame S. 240.
- Neue Funktion md5sum S. 251.
- Neue Direktive #addto S. 136.
- Neues Schlüsselwort mobilezoom für Webserver S. 286
- Anmerkung zum Schriftstil Blinkred und Blinkblue S. 302
- Anzahl der Webelemente auf 60 pro Seite erhöht S. 281
- Anzahl der globalen Webelemente auf 60 erhöht S. 281

Version 26 (Patches 3.0xx, EibStudio 3.0xx)

- Codetabelle für Ereignisspeicher S. 349
- Neue Icons S. 312.
- Hintergrundbilder in das Webserverdesign S. 285

Version 24 (Patches 3.0xx, EibStudio 3.0xx)

Neue Funktionen für den Zugriff auf das Internen Flash

- Neues Kapitel Datensicherung S. 206
- Projektdaten auf dem Enertex® EibPC sichern S. 206
- Bilder, Daten, Zeitreihen auf dem Enertex® EibPC sichern S. 207

Neue Webserver-Funktionen:

- Benutzerverwaltung Webserver S. 275, 295

- Neues Webelement und Funktion weboutput S. 294
- Neue Webelement und Funktion webinput S. 273
- Ergänzung Funktion plink S. 267
- Ergänzung Funktion link S. 297.
- picture Element kann relative Pfade verarbeiten (auf internen Bereich), S. 292
- Eigener Bereich zum Hochladen von Dateien auf den Enertex® EibPC S. 207
- Ausführliches Beispiel zum weboutput S. 116
- Ausführliches Beispiel zum Hochladen von Grafiken für picture, header und footer S. 118
- Ausführliches Beispiel mtimechart bzw. Visualisieren von Zeitreihen mit den EXT-mtimechart S. 119
- Ausführliches Beispiel mtimechart bzw. Visualisieren von Zeitreihen, einfaches mtimechart S. 122
- Wechseln von angezeigten Graphen mit mtimechart S. 123
- 49 neue Webicongruppen S. 301

Neue Chart-Funktionen:

- Neues Webelement mtimechart S. 290
- Neue Funktion mtimechartpos S.272
- Neue Funktion mtimechart S.272
- Neue Funktion timebufferconfig S. 269
- Neue Funktion timebufferadd S. 269
- Neue Funktion timebuffersize S. 270
- Neue Funktion timebufferstore S. 270
- Neue Funktion timebufferread S. 270

Neue FTP-Funktionen:

- Neue Funktion ftptimeout S. 258
- Neue Funktion ftpbuffer S. 258
- Neue Funktion ftpstate S.258
- Neue Funktion flushftp S. 258
- Neue Funktion sendftp S. 257
- Neue Funktion ftpconfig S. 257
- Ausführliches Beispiel zur Verwendung der neuen FTP Funktionen S. 114.

Neue und erweiterte Funktionen von Zeichenketten:

- Neue Funktion encode S. 235 und S. 112
- Variable Länge von Zeichenketten möglich S. 229
- Neue Funktion capacity S. 235
- Definition EOS auf 65535 geändert
- Definition END auf 65534 geändert
- Ausführliches Beispiel zum Encoding von c14-Strings und encode S. 112
- Verketten von Strings unterschiedlicher Länge S. 113

Neue Funktionen zur Verarbeitung von Datums- und Zeitangaben:

- Neue Funktion utcconvert S. 186
- Neue Funktion utctime S. 186
- Neue Funktion utc S. 186

Sonstige Neuerungen:

- Fehlercodes des Ereignisspeichers aufgelistet S. 349
- Erklärung zum Visualisierungsassistenten S. 75
- Neue Funktion shift S. 179
- Neue Funktion presun S. 191
- Beschreibung sendmail erweitert
- Neue Konstante HALF bei charts und picture
- Ausbessern div. Fehler
- Neues Kapitel „Programmierung für Experten“ S.105 ff.
- Ergänzung Enertex® KNXNet/IP Router S. 17
- Überarbeitung des Beispiels S. 55

Version 22 (Patches 2.303, EibStudio 2.305)

- Neue Funktion plink S. 267
- Neue Funktion picture S. 267
- Neue Funktion link S. 262
- Ergänzung zum Webelement plink S. 297
- Ergänzung zum Webelement link S. 297
- Erweitertes Beispiel zur Funktion readrawknx S. 242

Version 21 (Patches 2.30x, EibStudio 2.30x)

- Ergänzungen zu den Makrolibs S. 26
- Ergänzung zum Webelement link S. 297.
- Ergänzung zum Webelement picture S. 292
- Neue Funktion resetrs232 S. 238
- Neues Webelement fastinit S. Fehler: Referenz nicht gefunden
- Ergänzung Menüleiste
- Ergänzung der Schlüsselwörter Referenz
- Syntaxergänzung für die Darstellung von physikalischen Adressen S. 162
- Neue Funktion readrawknx S. 242
- Neue Funktion writeresponse S. 174
- Neue Funktion setpeslider S. 269
- Neue Funktion seteslider S. 268
- Neue Funktion getpeslider S. 261
- Neue Funktion geteslider S. 261
- Neue Funktion peslider S. 294
- Neue Funktion eslider S. 293
- Ergänzung zur Funktion pslider S. 293
- Ergänzung zur Funktion slider S. 293
- Ergänzung zur Funktion scene S. 225, S. 66
- Neue Funktion presetscene S. 225
- Neue Funktion elog S. 218
- Neue Funktion elognum S. 218
- Neue Funktion devicenr S. 218
- Neue Funktion ping S. 251
- Neue Funktion afterc S. 201
- Neue Funktion delayc S. 199
- Ergänzung zu Firewall-Problemen und Lösungen
- Ergänzung der NTP Funktionen S 180
- Ausbessern diverser Schreibfehler
- Ergänzung zum readflash and writeflash S. 203
- Ergänzung zu connecttcp S. 248
- Neue Funktion sendtcparray S. 250
- Neue Funktion sendudparray S. 247

Version 19 (Patches 2.10x, EibStudio 2.10x)

- Ergänzung zu Text-Stilarten
- Ergänzung zu VPN S. 254
- Symbolsatz für black-Design S. 322
- Ausbessern diverser Schreibfehler
- Ergänzung zu connecttcp S. 248
- Ergänzung Funktionsweise Sektion [InitGA] S.171
- Ergänzung Asynchroner Zugriff S. 129
- Ergänzung zu FTP S. 150

Version 17 (Firmware 2.00x, Patches 2.00x, EibStudio 2.00x)

- Neue NTP Funktionen S. 180
- Menüleiste aktualisiert
- HTTPS S. 280
- Ausbessern diverser Schreibfehler
- Valdierungskonzept erläutert, S. 125
- Neue Option SXY bei m(p)chart-Webelement S. 289

- Neues Schlüsselwort design for Webserver S. 285
- VPN Server, S. 254
- Neue Makrolibs S. 26
- Makros mit Rückgabewerten, lokalen Variablen und Verwendung als Funktionen im Anwendungsprogramm S. 324
- Neue Funktion gaimage S. 240
- Neue Funktion getaddress S.239
- Schwarzes und Blaues Design des Webservers S. 285
- Neue Sektion [VPN] und [InitGA] S. 133
- [InitGA] S. 171, 44
- Neue Funktion eibtelegramm S.223
- Verhalten von gettime/getdate und gettimedate anpassen S.182
- event kann mit negierten Gruppenadressen arbeiten S. 173
- read event kann mit negierten Gruppenadressen arbeiten S. 170
- Neue Funktion initga S. 172
- Neue #include Direktive S. 137
- Neue #break_if_older_version S. 136
- Zeilenumbruch der sendmail Funktion S. 252
- Neue Funktion sendhtmlmail S. 253
- Neue #define Direktive S. 137
- Neue #ifdef Direktive S. 137
- Neue #endif Direktive S. 137
- Neue #undef Direktive S. 137
- Neue #ifndef Direktive S. 137
- Neue Funktion eventread S. 174
- Neue Funktion eventresponse S.174
- Neue Funktion eventwrite S.174

Version 15 (Firmware 1.30x, Patches 1.30x, EibStudio 1.30x)

- Geschweifte Klammern in if-Anweisungen S. 41, S. 135
- Address-Funktion S. 215
- Eval-Funktion S. 219
- Stringcast-Funktion S. 230
- Stringset-Funktion S. 230
- Button-Funktion S. 259
- hart-Funktion S. 259
- Getslider-Funktion S. 261
- Getpslider-Funktion S. 261
- -Funktion S.
- mchart-Funktion S.263
- Mpchart-Funktion S.264
- Mpbutton-Funktion S.264
- display-Funktion S.265
- Pchart-Funktion S.265
- Setslider-Funktion S.268
- Setpslider-Funktion S.268
- Stringformat-Funktion S.232
- Hexadezimalkonstanten möglich im Compiler S. 161
- Synchronisierung mit dem Anwendungsprogramm., S. 180
- Second-Funktion S.184
- Minute-Funktion S.184
- Hour-Funktion S.183
- Changehour-Funktion S.185
- Changeminute-Funktion S.185
- Changesecond-Funktion S.185
- Komplette Überarbeitung von Webserverelemente (S. 276)
- Einseiten-Version Webserver, S. 78
- Mehr-Seiten Version Webserver, S. 83

- Visualisierung mit dem iPhone, S. 34
- Datentypen in der Übersicht, Ergänzung EIS Typen S.163
- TCP/IP Server und Client, S. 74
- RS232 Schnittstelle, S. 237
- Hinweis auf Changelog Updates, S. 13
- Ergänzungen zum Webserver: Ein-/Mehrseitenversion: S. 274
- display-Funktion: S. 274
- Readflash-Funktion: S. 203
- Writeflash-Funktion: S. 203
- readknx und die Kapitel samt Einleitung zum Thema Routing mit dem Enertex® EibPC, S. 241
- Erweiterte infromationen zur Variablendefinition, S. 48
- Neue Zeichenvorlage WebElement für die Formatierung von Web-Konfigurationskommandos wie z.b. *page*
- Anpassungen der internen Konfigurationsdatei 5 eingepflegt (Webelemente)
- Auslagern auf FTP-Verzeichnis S. 63, 150
- Performance Einstellungen, S. 135
- Alle Eibstudio Screenshots von Eibstudio durch die Screenshots der Eibstudio V1.207 (Win XP) ausgetauscht.
- Neuer Debugger erklärt auf S. 152
- FT1.2 Unterschied Busmonitor und Gruppenmonitor erklärt, S. 24