enertexbayern gmbh
simulation entwicklung consulting

Handbuch Enertex® EibPC und EibPC²

Voraussetzungen

Enertex® EibPC: **Firmware 4.000 oder höher**
Optionen NP,V3

Enertex® EibStudio: **Version 4.000 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

Sicherheitshinweise **8**
Lizenzhinweise **8**
E-Mail 8
Support-Export 8
HOTLINE 8
KNX-User-Forum 8
Lehrvideos 8
Updates **8**
Enertex® EibPC² **9**
Übersicht *9*
Inbetriebnahme *11*
Anschlüsse und Bedienelemente 11
Installation 12
Gerätestart *13*
Firmwareupdates *13*
Werkseinstellungen *13*
Enertex® EibPC **14**
Übersicht *14*
Inbetriebnahme *16*
Anschlüsse und Bedienelemente 16
Installation 17
Gerätestart *19*
Firmwareupdates *19*
Werkseinstellungen *19*
Update auf Firmware-Version 4 ..... *20*
Downgrade auf Version 3.107 20
Kurzanleitung EibStudio 4 ..... **21**
Unterschiede zum Enertex® EibStudio 3 ..... **22**
Enertex® EibStudio 4 ..... **23**
Installation *23*
Fenstermenü *23*
Projektübersicht *23*
Projektverzeichnis 23
Projekt aus Enertex® EibStudio 3 importieren 23
Einstellungen *23*
Konfigurationsverzeichnis 23
Bedienelemente *24*
Übersicht *26*
Objekte *26*
Gruppenadressimport 26
Topologie 26
Interne Variablen 26
Konstanten 26
Logik *27*
Definitionen 27
Beispiel: Lichtautomatik 28
Debug-Modus 28

Visualisierungsobjekte....30
Visu....*31*
Elemente....31
Funktionen....31
Eigene Vorlagen....31
Vorlagen....31
Verknüpfung mit Logik und Experte....31
Experte....*32*
Autovervollständigung....32
Makros....32
Eigene Visualisierungsseiten....32
Zugriff auf Visualisierungselemente....32
Syntax....33
Online Debuggen zur Laufzeit....34
Projekteinstellungen....*35*
Suche....35
KNX-Verbindung....35
Netzwerkverbindung....35
Namensauflösung....35
Datum und Uhrzeit....35
Position....35
SHUTDOWN-Variable....35
FTP....35
E-Mail....35
Sicherung....35
Dateien....35
HTTPS....35
VPN....35
IDs....36
Export und Import....*37*
Debugger....*37*
Gruppenmonitor....*37*
Langzeitpuffer....*37*
Ereignisspeicher....*37*
Simulation....*37*
Objekte....38
Datentyp....*38*
Zahlen (Konstanten)....39
Datentypen in der Übersicht....41
Variablen....*42*
Gruppenadressen....*42*
Manuelle Gruppenadresse....42
Initialisieren von Gruppenadressen....43
Validierungsschema....*44*
Objektbaum....44
Schreiben auf Warteschlangen....47
Asynchroner Rückgabewert....47
Makros....48
Rekursion....49
Befehle und Funktionen....50
Logische Verknüpfungen....*50*

Und - Verknüpfung....50
Oder - Verknüpfungen....50
Exklusiv-Oder....51
Vergleichsoperatoren....52
Hysterese-Vergleich....52
Invertierung....53
Shift....53
Zeitfunktionen....54
Die Zeit des Enertex® EibPC neu setzen....54
Die Zeit des Enertex® EibPC auf den KNX-Bus schreiben....54
Das Datum des Enertex® EibPC neu setzen....55
Das Datum des Enertex® EibPC auf den KNX-Bus schreiben....55
GetTimeDate - Die Zeit und das Datum des Enertex® EibPC neu setzen....56
SetTimeDate - Die Zeit und das Datum des Enertex® EibPC auf den KNX-Bus schreiben....56
Hour....56
Minute....57
Second....57
Changehour....58
Changeminute....58
Changesecond....58
Utc....59
Utctime....59
Utcconvert....59
Datumssteuerung....60
Date- Datumsvergleich....60
Month - Monatsvergleich....60
Day - Tagesvergleich....61
DayOfWeek – Wochentag....61
Easterday....61
Eastermonth....61
Beschattung und Sonnenstand....62
Sun - Sonnenstand....62
Azimuth....62
Elevation....63
Presun....63
Sunrisehour - Stunde des Sonnenaufgangs....64
Sunriseminute - Minute des Sonnenaufgangs....64
Sunsethour - Stunde des Sonnenuntergang....64
Sunsetminute - Minute des Sonnenuntergangs....64
Zeitschaltuhren....65
Tageszeitschaltuhr....65
Stundenzeitschaltuhr....66
Minutenzeitschaltuhr....66
Vergleichszeitschaltuhren....67
Grundsätzliches....67
Wochenvergleichszeitschaltuhr....67
Tagesvergleichszeitschaltuhr....68
Stundenvergleichszeitschaltuhr....68
Minutenvergleichszeitschaltuhr....69
Spezielle Zeitfunktionen....70

Präzisionstimer - programmierbare Verzögerungszeit....70
Zyklustimer – cycle....74
Remanentspeicher....75
Readflash....75
Writeflash....75
Readflashvar....76
Writeflashvar....77
Arithmetische Verknüpfungen („Berechnungen“)....78
Grundsätzliches....78
Abs- Absolutwert....78
Addition....78
Acos - Arkuscosinus....78
Asin - Arkussinus....79
Atan - Arkustangens....79
Cos - Cosinus....79
Ceil - kleinste ganze Zahl....80
Division....80
Average - Durchschnitt....80
Exp - Exponentialfunktion....81
Floor - Größte ganze Zahl....81
Log- Logarithmus....81
Max - Maximum....82
Min - Minimum....82
Mod - Modulo....83
Mutliplikation....83
Pow - Potenz....83
Sqrt - Quadratwurzel....84
Sin - Sinus....84
Subtraktion....84
Tan - Tangens....85
Sonderfunktionen....86
Change....86
Convert - Konvertieren von Datentypen....87
Devicenr....88
Elog....88
Elognum....88
Eval....89
Processingtime....89
Systemstart....91
Programmende....91
Random – Zufallszahl....91
Comobject - Kommunikationsobjekt....92
Sleep – Passivmodus....92
Eibtelegramm....93
Lichtszenen....94
Scene - Szenenaktor....94
Presetscene – Vorbelegung für Szenenaktor....94
Storescene - Szene speichern....95
Callscene - Szene aufrufen....96
Stringfunktionen....97
Verketten....97

Find....97
Stringcast....98
Stringset....98
Stringformat....100
Split....102
Size....102
Capacity....103
Tostring....103
Encode....103
Urldecode....104
Urlencode....104
RS232 Schnittstelle....105
Konfiguration....105
Readrs232....105
Resetrs232....106
Sendrs232....106
KNX-Telegramme....107
write....107
read....107
event....108
eventread....109
eventresponse....109
eventwrite....109
writeresponse....109
initga....110
eibtelegramm....111
KNX-Telegramm-Routing....113
Address....113
Getaddress....113
Gaimage....114
Getganame....114
ReadKnx....115
Readrawknx....116
Netzwerkfunktionen....119
Standard-Ports....119
UDP Telegramme....119
TCP Server und Client....122
Md5sum....125
Ping....125
Resolve....126
Sendmail....126
Sendhtmlmail....127
VPN Server....128
FTP-Funktionen....130
Webserverfunktionen....132
Button (Webbutton)....132
Chart (Webchart)....132
Display (Webdisplay)....133
Getslider....134
Getpslider....134
Geteslider....134

Getpeslider....134
link....135
Mbutton....135
Mchart....136
Mpchart....137
Mpbutton....137
Pdisplay....138
Pchart....138
Pbutton....139
Picture....139
Plink....140
Setslider....141
Setpslider....141
Seteslider....141
Setpeslider....142
Timebufferconfig....142
Timebufferadd....142
Timebufferclear....143
Timebufferstore....143
Timebufferread....143
Timebuffersize....143
Timebuffervalue....144
mtimechartpos....145
mtimechart....145
Webinput....146
Weboutput....146
Webserverkonfiguration....147
Das Webserverdesign....147
Webserverelemente....149
Konfiguration....154
Webicons....172
Verhalten des Webservers bei Benutzereingaben....174
Makros
Funktionsblöcke....195
Grundsätzliches....195
Mitgelieferte Bibliotheken....195
Programmierung eines Makros....195
Grundsätzliches....195
Definition....195
Sonderzeichen....196
Laufzeit- und Syntaxfehler....196
Makroasisstent....196
Lokale Variablen und Rückgabewerte....196
Schlüsselwörter - Referenz....198
Schlüsselwörter - Referenz - Alphanumerische Sortierung....210
Quellen....218
Fragen und Antworten....219
Ereignisse....219
Probleme und Lösungen....222
ChangeLog....223

Version 32 (> Firmware 4.000, EibStudio 4.000)....223

Wir freuen uns, dass Sie sich für einen Enertex® EibPC oder EibPC² entschieden haben.

Sicherheitshinweise

Bitte beachten Sie folgende Punkte:

- Einbau und Montage elektrischer Geräte dürfen nur durch Elektrofachkräfte erfolgen.
- Bei Anschluss und Inbetriebnahme von KNX-Geräten werden Fachkenntnisse durch 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.

Hilfe

E-Mail

Bei Fragen und Problemen zu Ihrem Enertex® EibPC und EibPC² können Sie sich an eibpc@enertex.de wenden.

Support-Export

Um ihr Anliegen schneller bearbeiten zu können, fügen Sie Ihrer Anfrage bitte das Projekt hinzu. Klicken Sie im Fenstermenü auf **Hilfe → Export für Support**. Senden Sie uns die Datei (*.esp).
Dies ist zip-Archiv und enthält ihr Projekt und alle hinzugefügten Webserver-Dateien, sowie Informationen zum System und die Log-Datei. Vertrauliche Informationen wie FTP-, E-Mail, VPN-Passwörter sind jedoch **nicht** enthalten.

HOTLINE

Zu den üblichen Öffnungszeiten steht Ihnen auch der kostenlose Telefonsupport unter der Rufnummer 09191/73395-0 zur Verfügung.

KNX-User-Forum

Unter http://www.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.

Lehrvideos

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 und Enertex® EibPC².

Enertex® EibPC²

Übersicht

Abbildung 1: Der Enertex® EibPC²

Zusammenfassung

Der Enertex® EibPC² stellt eine Steuerung für die Hutschienenmontage (4 TE) für KNX-Bussysteme dar. Er ermöglicht mit ca. 1,8 W Leistungsaufnahme stromsparend und umweltschonend vollständige Kontrolle über das KNX-Bussystem.

Der Enertex® EibPC² ist Szenenaktor, Berechnungscomputer, Logikzentrale, SPS, Schaltzeituhr, LAN-und Internet-Anbindung, Webserver und E-Mail-Client in einem Gerät.

Er verfügt über alle Funktionen des Enertex® EibPC. Über den integrierten Busankoppler stellt der Enertex® EibPC² der ETS einen KNX-Tunnel zur Programmierung zur Verfügung.

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

KNX-Funktionen

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

Alle Funktionen des KNX-Bussystems sind ansprechbar

- 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 eine Hausautomatisierung erstellt. Damit lässt sich der Enertex® EibPC ohne ETS auf einfache Weise programmieren.

Programmierung - Die Programmierung erfolgt wahlweise durch Verbinden von vorgefertigten Funktionsblöcken in einem grafischen Editor oder durch Programme in einer einfachen Programmiersprache, für die keine aufwändigen Lehrgänge notwendig sind.

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

Inbetriebnahme

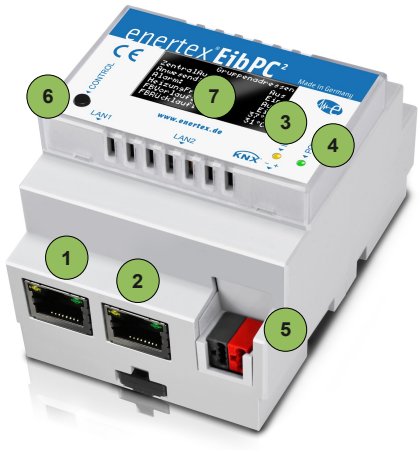


Abbildung 2: Anschlüsse und Bedienelemente

Anschlüsse und Bedienelemente

In Abbildung 2 sind die folgenden Anschlüsse und Bedienelemente des Enertex® EibPC² dargestellt:

1. LAN-Anschluss 1
2. LAN-Anschluss 2
3. Info-LED (orange)
4. Power-LED (grün)
5. KNX-Klemme
6. Control-Taste
7. Anzeige

Stromversorgung

Der Enertex® EibPC² wird direkt vom KNX-Bus gespeist und benötigt eine Spannung von 27V bis 30V. Prüfen Sie unbedingt die Spannung am Installationsort, falls der Enertex® EibPC² nicht in unmittelbarer Nähe zur KNX-Spannungsversorgung installiert wird.

Es ist auch möglich, den Enertex® EibPC² mit einem normalen Netzteil zu betreiben, falls der Buszugriff über eine KNXnet/IP-Schnittstelle erfolgt.

Die Leistungsaufnahme beträgt ca. 1,5 Watt bei 30 VDC

Die Spannungsversorgung muss mindestens 3,2 Watt (110 mA bei 29 V Busspannung) Ausgangsleistung zur Verfügung stellen.

Buszugriff

Der Enertex® EibPC² verfügt über einen integrierten Busankoppler. Alternativ kann im EibStudio eine externe KNXnet/IP-Schnittstelle ausgewählt werden. Dadurch haben Sie die Möglichkeit, den Enertex® EibPC² unabhängig vom KNX-Bus zu installieren.

Grundsätzlich sind alle KNX-zertifizierte IP Schnittstellen mit dem Enertex® EibPC² kompatibel.

Besonders zu empfehlen sind

- Enertex® KNX IP Secure Router
- Enertex® KNX IP Secure Interface
- Enertex® KNXnet/IP Router
- Enertex® KNXnet/IP Interface

Die Router biete viele Zusatz- und Diagnosefunktionen und können den Enertex® EibPC nach einem Reset ohne Internetverbindung direkt mit der Uhrzeit synchronisieren.

Der Enertex® EibPC² nutzt das sog. Tunneling der KNXnet/IP-Schnittstelle. Diese wird dadurch exklusiv belegt, d.h. Sie können die Schnittstelle in diesem Modus dann nicht von der ETS aus ansprechen.

Installation

Integrierter Busankoppler

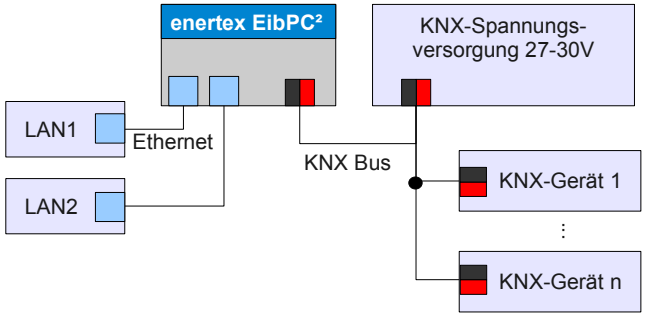


Abbildung 3: Anschluss des Enertex® EibPC an den KNX-Bus

Abbildung 3 zeigt den Anschluss des Enertex® EibPC². Gehen Sie wie folgt vor:

1. Über den LAN 1 oder LAN 2-Anschluss (1,2) mit dem Netzwerk verbinden.
2. Der freie LAN-Anschluss (2) kann verwendet werden, um weitere Geräte ins gleiche Netz einzubinden.
3. Den Enertex® EibPC² an eine KNX-Spannungsversorgung anschließen.
   Sie können auch ein anderes Netzteil entsprechender Spannung und Leistung verwenden, falls Sie eine KNXnet/IP-Schnittstelle statt der integrierten verwenden.

Integrierter Switch

Achtung: Kommunikation zwischen LAN1 und LAN2 ist nur möglich, wenn der Enertex® EibPC² gestartet und betriebsbereit ist.

Anderenfalls ist keine Kommunikation zwischen LAN1 und LAN2 möglich. Dies ist auch der Fall, wenn der Enertex® EibPC² neu gestartet wird.

Wird das Anwendungsprogramm neu gestartet oder ein neues übertragen, bleibt die Verbindung bestehen.

KNXnet/IP-Schnittstelle

Abbildung 4 zeigt den Anschluss bei Verwendung einer KNXnet/IP-Schnittstelle.

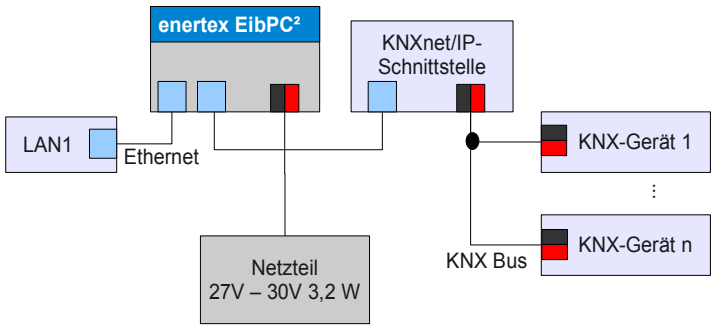


Abbildung 4: Anschluss des Enertex® EibPC² an den KNX-Bus über eine KNXnet/IP- Schnittstelle

Jetzt haben Sie die Installation des Enertex® EibPC² an Ihren KNX-Bus abgeschlossen.

Gerätestart

Die LEDs erleichtern die Inbetriebnahme:

Einschalten:
Info- und Power-LEDs leuchten

Bootvorgang beendet:
Info-LED blinkt

Beim Gerätestart des Enertex® EibPC² können Sie folgendes beobachten:

1. Info- und Power-LED leuchten, der Bootvorgang wird gestartet.
2. Im Betrieb leuchtet die Power-LED durchgängig.
3. Ca. 2 Minuten nach dem Start blinkt die Info-LED im Sekundentakt mit gleichlanger Ein- und Ausschaltzeit. Während dieser Phase ist ein Zurücksetzen auf Werkseinstellungen möglich (s.u.).
4. Kurz danach wird die Bus-Schnittstelle initialisiert - Ein paar kurze Impulse können dabei an der Info-LED beobachtet werden.
5. Nach erfolgreichem Start der Netzwerkschnittstelle wird die aktuelle IP auf der Anzeige angezeigt. Die Anzeige erlischt automatisch nach 30 s. Sie kann durch Drücken der Control-Taste erneut aktiviert werden.
6. Wenn ein Telegramm am Bus ausgelöst wird (KNX-Schalter etc.), wird auch eine kurze Impulsfolge an der Info-LED sichtbar. Dazu muss kein Anwenderprogramm geladen sein.

Firmwareupdates

Aktualisierungen der Firmware können Sie selbst durchführen. Dazu benötigen Sie eine Aktualisierungsdatei (Dateiname: *eibpc2-patch-x.xxxx.ptc*) die Sie über das Enertex® EibStudio installieren können. Der Vorgang dauert insgesamt etwa 5 Minuten. Während dieser Zeit muss eine stabile Spannungsversorgung garantiert sein.

Sollte das Gerät 20 Minuten nach Start der Aktualisierung noch nicht wieder im Enertex® EibStudio ansprechbar sein und keine Aktivität signalisieren (beide LEDs dauerhaft aus oder ein, Display ohne Anzeige bei Drücken der Control-Taste), trennen Sie das Gerät kurz von der Spannungsversorgung, um es neu zu starten.

Sollte es danach weiterhin nicht erreichbar sein, wenden Sie sich bitte an unseren Support.

Werkseinstellungen

Zurücksetzen beim Start

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

Durch Drücken der Control-Taste wird folgendes ausgelöst:

1. Netzwerkeinstellungen auf DHCP zurücksetzen
2. Anwenderprogramm löschen
3. Daten zum Sonnenstand löschen
4. VPN-Benutzer und IP-Filter löschen
5. HTTPS-Benutzer löschen
6. Szenen und Timebuffer löschen

Nach dem Zurücksetzen blinkt die Power-LED mehrmals kurz auf und das Gerät wird neu gestartet.

Zurücksetzen während des Betriebs

Wurde das Gerät bereits gestartet, kann es auf Werkseinstellungen zurückgesetzt werden, indem die Control-Taste für mindestens 25 Sekunden gedrückt wird. Auf dem Display erscheint eine Bestätigung, die Power-LED blinkt mehrmals kurz auf und das Gerät startet neu.

Enertex® EibPC

Übersicht

Abbildung 5: Der Enertex® EibPC

Fertige Funktionsblöcke

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.

KNX-Funktionen

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).

Alle Funktionen des KNX-Bussystems sind ansprechbar

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

Abbildung 6: Enertex® EibPC Anschlussbelegung

Anschlüsse und Bedienelemente

In Abbildung 6 sind die Anschlüsse und Bedienelemente des Enertex® EibPC dargestellt:

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

Stromversorgung

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 eine KNX-RS232 Schnittstelle für das FT 1.2 Protokoll oder eine IP Schnittstelle geeignet.

FT1.2 Schnittstelle

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 sind

- Enertex® KNX IP Secure Router
- Enertex® KNX IP Secure Interface
- Enertex® KNXnet/IP Router
- Enertex® KNXnet/IP Interface

Die Router biete viele Zusatz- und Diagnosefunktionen und können den Enertex® EibPC nach einem Reset ohne Internetverbindung direkt mit der Uhrzeit synchronisieren.

Installation

Anschluss an FT1.2-Schnittstelle über RS232

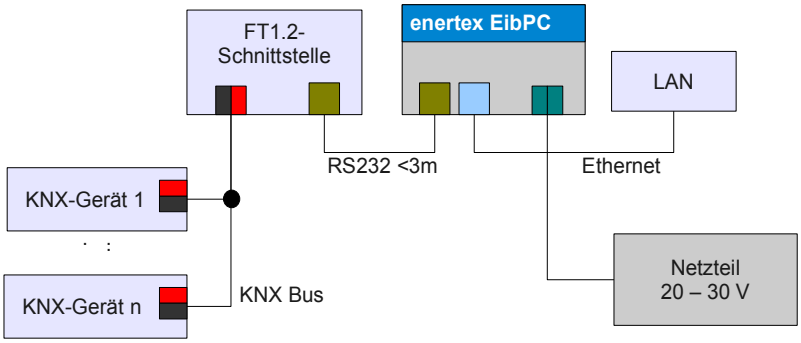


Abbildung 7: Anschluss des Enertex® EibPC über eine FT1.2-Schnittstelle

Die Abbildungen 7 zeigt den Anschluss des Enertex® EibPC an den KNX-Bus über eine FT1.2-Schnittstelle.

Gehen Sie wie folgt vor:

1. Enertex® EibPC an ein separates 20V - 30V DC-Netzteil anschließen (1)
2. Über den LAN-Anschluss (2) muss der Enertex® EibPC mit dem LAN verbunden werden.
3. Die RS232-Schnittstelle (3) wird mit einer FT1.2-Schnittstelle verbunden. Als Kabel nimmt man dafür ein Verlängerungskabel („Male“ auf „Female“). Die Pinbelegung der RS232-Schnittstelle des Enertex® EibPC entspricht dem Standard EIA-232.
4. EibPC neu starten (über EibStudio oder Stromzufuhr unterbrechen).

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.

Wenn der EibPC startet, muss die FT1.2-Schnittstelle in jedem Fall mit dem eingeschalteten Bus verbunden sein.

Also nicht: Enertex® EibPC Einschalten und dann RS232 Verbindung herstellen. Die FT1.2-Schnittstelle kann mit dem Enertex® EibPC gleichzeitig eingeschalteten werden, nicht aber nach dem Bootvorgang des Enertex® EibPC.

Verwendung einer KNXnet/IP-Schnittstelle

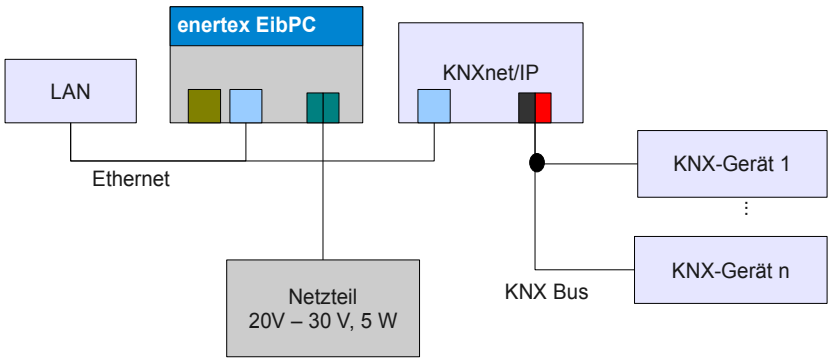


Abbildung 8: Anschluss des Enertex® EibPC an den KNX-Bus über eine KNXnet/IP-Schnittstelle

Abbildung 8 zeigt den Anschluss des Enertex® EibPC an den KNX-Bus über eine KNXnet/IP-Schnittstelle. Gehen Sie dabei wie oben beschrieben vor, abgesehen vom RS232-Anschluss (3).

Hinweis:

Bei Problemen bein Buszgriff versuchen Sie die KNXnet/IP-Schnittstelle neu zu starten, da diese mitunter auch „hängen“ können.

Aktivierung im Enertex® EibStudio

Die Schnittstelle wird erst nach Überspielen und anschließendem Starten eines Anwendungsprogramms 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 Verbindung zur Schnittstelle im Enertex® EibStudio vorübergehend trennen und so die Tunnelverbindung freigeben.

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

Gerätestart

Nach dem Einschalten oder Neustart via EibStudio des Enertex® EibPC können Sie folgendes beobachten:

1. Zuerst leuchtet die LED dauerhaft (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 Zurücksetzen auf Werkseinstellungen möglich (s.u.).
3. Initialisiere KNX-Verbindung. 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).

Firmwareupdates

Aktualisierungen der Firmware können Sie selbst durchführen. Dazu benötigen Sie eine Aktualisierungsdatei (Dateiname: *eibpc1-patch-x.xxx.ptc*) die Sie über das Enertex® EibStudio installieren können. Der Vorgang dauert insgesamt etwa 5 Minuten. Während dieser Zeit muss eine stabile Spannungsversorgung garantiert sein.

Sollte das Gerät 20 Minuten nach Start der Aktualisierung noch nicht wieder im EibStudio ansprechbar sein, und keine Aktivität signalisieren (LED dauerhaft aus oder ein), trennen Sie das Gerät kurz von der Spannungsversorgung, um es neu zu starten.

Sollte es danach weiterhin nicht erreichbar sein, wenden Sie sich bitte an unseren Support.

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 DHCP zurücksetzen
2. Anwenderprogramm löschen
3. Daten zum Sonnenstand löschen.

Update auf Firmware-Version 4

Um das Enertex® EibStudio 4 verwenden zu können, muss Firmware-Version v4.000 oder höher auf dem Enertex® EibPC installiert sein. Die Aktualisierung ist nur für Geräte mit der Option V3 verfügbar.

Enertex® EibStudio 4: Firmware-Version v4.xxx

Auf Geräten ohne Option V3 kann Enertex® EibStudio 4 nicht verwendet werden. Es besteht die Möglichkeit, die Option vorher zu erwerben. Sie erhalten dazu einen Freischaltkey, den Sie selbst aktivieren können. Wenden Sie sich bitte bei Interesse an unseren Vertrieb.

Enertex® EibStudio 3: Firmware-Version 3.107

Bevor Sie fortfahren, laden Sie die aktuellsten Versionen von www.enertex.de herunter:

- Enertex® EibStudio v3.xxx
- Enertex® EibStudio v4.xxx
- Firmware v4.xxx

Prüfen Sie außerdem, ob Enertex® EibStudio 4 auf Ihrem PC gestartet werden kann. Ist dies nicht der Fall, installieren Sie das Firmware-Update nicht.

Gehen Sie zum Aktualisieren wie folgt vor:

- Starten Sie das Enertex® EibStudio 3
- Entpacken Sie die Firmware v4.xxx und prüfen Sie den Dateinamen:
  Dieser muss wie folgt sein: *eibpc1-patch-4.xxx.ptc* (xxx beliebige Zahlen)
- Installieren Sie die Firmware-Datei. Dieser Vorgang dauert etwa 5 Minuten. Der Enertex® EibPC startet am Ende automatisch neu. Zwischenzeitlich geht die INFO-LED aus. Nach dem Neustart blinkt diese wieder wie gewohnt.
- Der Enertex® EibPC wird nun nicht mehr im Enertex® EibStudio 3 erreichbar sein. Beenden Sie es und starten Sie das Enertex® EibStudio 4.
- Prüfen Sie die Verbindungseinstellungen und bauen Sie eine Verbindung auf. Verifizieren Sie die installierte Firmware-Version (Übersicht, S. 27).

Downgrade auf Version 3.107

Es besteht die Möglichkeit, nach dem Update auf Version 4 wieder die Version 3.107 zu installieren, falls unerwartet Probleme auftreten. Dazu muss der Enertex® EibPC im Enertex® EibStudio 4 erreichbar sein.

Gehen Sie wie folgt vor:

- Laden Sie sich die Firmware v3.107 herunter und entpacken Sie sie. Prüfen Sie, dass die Datei *eibpc-patch-3.107.ptc* vorhanden ist.
- Starten Sie das Enertex® EibStudio 4 und prüfen Sie die Verbindung zum Enertex® EibPC
- Installieren Sie die Firmware-Datei.
- Dieser Vorgang dauert etwa 5 Minuten. Der Enertex® EibPC startet am Ende automatisch neu. Zwischenzeitlich geht die INFO-LED aus. Nach dem Neustart blinkt diese wieder wie gewohnt.
- Der Enertex® EibPC ist nun nicht mehr im Enertex® EibStudio 4 erreichbar. Starten Sie Enertex® EibStudio 3.
- Prüfen Sie die Verbindungseinstellungen und bauen Sie eine Verbindung auf. Verifizieren Sie die installierte Firmware-Version.

Kurzanleitung Eib-Studio 4

Das Gerät wurde wie oben beschrieben mit dem Netzwerk verbunden und gestartet. Im Auslieferungszustand sucht es nach einen DHCP-Server, um eine IP-Adresse zugewiesen zu bekommen. Dieses Verhalten können Sie später in den Einstellungen ändern.

Zur Programmierung wird das Enertex® EibStudio ab Version 4 verwendet. Es dient sowohl zur Änderung der Einstellungen als auch der Erstellung eines Anwenderprogramms.

Das Enertex® EibStudio 4 muss lediglich entpackt werden, es erfolgt keine Installation.

Enertex® EibStudio 4 starten

Wichtig: Eine aktive Firewall verhindert u.U. die Kommunikation zwischen Enertex® EibStudio 4 und Enertex® EibPC und muss entsprechend konfiguriert werden.

Projektverzeichnis

Beim ersten Start fragt Enertex® EibStudio 4 nach einem Verzeichnis, in dem die Projekte gespeichert werden sollen.

Außerhalb dieses Verzeichnisses und dem Konfigurationsverzeichnis (S. 24) werden keine Daten verändert.

Beim Import von bestehenden Projekten werden diese automatisch ins Projektverzeichnis kopiert.

Sie können das Projektverzeichnis über die Benutzer-Einstellungen ändern. Dabei wird ein eventuell geöffnetes Projekt geschlossen und alle vorhandenen Projekte im geänderten Verzeichnis werden angezeigt.

Projektübergreifende Einstellungen

Einstellungen, die für alle Projekte gelten, erreichen Sie über das Fenstermenü unter **Bearbeiten → Einstellungen**.

Projekt

Enertex® EibStudio 4 startet immer in der Projektübersicht. Darin können Sie neue Projekte anlegen, Projekte importieren oder löschen. Dabei werden nur die für das jeweilige Projekt gespeicherten Daten im aktuellen Projektverzeichnis gelöscht.

Ein Projekt beinhaltet alle Informationen, die für die Konfiguration und den Betrieb eines Geräts notwendig sind.

Projektmenü

Innerhalb eines geöffneten Projekts wird das Projektmenü verwendet. Es bietet Zugriff auf:

- **Übersicht**: Gerätezustand, Programminformationen, Logbuch
- **Objekte**: Alle Gruppenadressen und Variablen
- **Logik**: Logikeditor zur grafischen Erstellung von Programmen
- **Visu**: Erstellen der Web-Visualisierung
- **Experte**: Texteditor zur Erstellung von Programmen
- **Einstellungen**: Projektspezifische Konfiguration des Enertex® EibPC

Bus-Verbindung

Um das erste Programm ausführen zu können muss die Verbindung zum Enertex® EibPC im Projektmenu unter **Einstellungen → Verbindung** konfiguriert werden. Befindet sich das Gerät im gleichen Netz wird er über die Suche gefunden.

Je nach Gerät und Installation kann hier auch die Verbindung zum KNX-Bus eingestellt werden.

Programm

Im Fenstermenü **Projekt** kann die Kompilierung des Programms gestartet werden. Das Programm wird dabei aus den einzelnen Konfigurationen zusammengesetzt, d.h. es beinhaltet Logik, Visualisierung, sowie alle Expertenprogramme und Einstellungen.

Im gleichen Menü kann die Programmübertragung auf den Enertex® EibPC gestartet werden.

Objekte

Im Projektmenü **Objekte → ETS-Import** können *.esf* und *.knxproj-Dateien* importiert werden, um Typinformationen der verfügbaren Gruppenadressen zu erhalten. Diese können bei Bedarf direkt unter **Objekte → Gruppenadressen** geändert werden.

Typinformationen sind insbesondere für die Verwendung des Debuggers und des Gruppenmonitors notwendig.

Die Liste der verwendeten Variablen wird bei jeder Programmübersetzung neu erstellt und kann nicht selbst geändert werden.

Unterschiede zum Enertex® EibStudio 3

Hier finden Sie kurz zusammengefasst, wo sich bestimmte Einstellungen befinden und was sich bei der Programmierung geändert hat.

Falls Sie bisher keine Erfahrungen mit dem Enertex® EibStudio 3 oder früher gesammelt haben, können Sie dieses Kapitel beruhigt überspringen.

Projekte und Struktur

Ein Projekt besteht nicht wie bisher aus einzelnen Dateien, die per #include verknüpft werden, sondern wird über die einzelnen Funktionen im Projektmenü verwaltet. Ein direktes Ändern von Projektdateien, die im Projektverzeichnis gespeichert sind, ist nicht zu empfehlen.

Die Sektionen wie [VPN], [Performance], [FTP], ... sind deshalb nicht mehr zu verwenden. Alle Einstellungen wie sind direkt in den Projekteinstellungen (S. 36) zu finden.

ETS-Import

Die Quelldatei für die importierten Gruppenadressen (.esf) ist nicht mehr Teil des Projektes, die Informationen werden statt dessen direkt im Projekt gespeichert.

Initialisierung

Statt der Sektion [InitGA] wird der Haken in der Gruppenadressübersicht unter **Objekte** gesetzt. Werden Gruppenadressen in der Visualisierung verwendet, so werden diese automatisch initialisiert. Die Liste der zu initialisierenden Gruppenadressen wird bei jedem Kompilieren aktualisiert.

Visualisierungsassistent

Visualisierungen werden direkt über die Funktion **Visu** im Projektmenü erstellt. Es muss nicht mehr zwischen vorgefertigten Seiten gewählt werden. Statt dessen werden leere Seiten angelegt, Elemente hinzugefügt, platziert und jedes Element direkt konfiguriert. Hier stehen die gängigsten Funktionen direkt zur Verfügung.

Programme

Das eigentliche Programm aus der Sektion [EibPC] kann nun über die Funktion **Experte** auf einzelne Programme aufgeteilt werden. Diese werden bei der Kompilierung automatisch zusammengefügt. Variablen, die in einem Programm verwendet werden, werden also global angelegt und stehen in anderen Programmen ebenfalls zur Verfügung (dürfen aber auch nicht erneut global belegt werden).

Es gilt also weiterhin, dass Variablen eindeutig global angelegt werden.

Es werden ebenso automatisch Variablen z.B. bei der Verwendung von Elementen in der Visualisierung angelegt. Diese beginnen stets mit INT_. Im Kollisionen zu vermeiden, legen Sie also keine Variablen in diesem Namensanfang an.

Eigene Visualisierungsseiten

Bestehende Visualisierungsseiten können weiterverwendet werden. Dabei ist wichtig, dass die Seiten-ID nicht mit automatisch vergebenen IDs aus der **Visu** kollisiert. Um dies zu verhindern können Sie die Start-ID unter **Projekteinstellungen → IDs** (S. 37) anpassen. Dies geht neben Seiten für alle globalen Elemente wie Timebuffer, Webinputs und Weboutputs (S. 33).

Makros

Die Verwendung von Makros ist weiterhin möglich. Die bisher extern mitgelieferte Makrobibliothek ist nun integriert. Makros werden direkt in entsprechenden Expertenprogramm durch `Doppelklick` auf den Namen des Makros hinzugefügt. Eingefügte Makros können durch `Doppelklick` editiert werden.

Integrierte Makrobibliotheken können nicht verändert werden. Erstellen Sie eine neue Makrobibliothek und kopieren Sie das zu ändernde Makro. Interne Makrobibliotheken werden deaktiviert, wenn ein eigenes Makro mit dem gleichen Namen existiert.

Wird eine eigene Makrobibliothek definiert, so muss die genaue Formatierung eingehalten werden, dass die definierten Makros in der Liste zur Verfügung stehen und der Parameterdialog korrekt erstellt wird.

Vordefinierte Visualisierungsseiten

Komplexe Vorgabeseiten aus dem Visuassi stehen als „Funktionen“ zur Verfügung. Dies sind Kombinationen von Elementen, die nicht einzeln Verändert werden können sondern einen gemeinsamen Einstellungsdialog besitzen. Alternativ können Sie vorhandene Vorlagen einbinden oder eigene Seiten als Vorlagen speichern, um sie in mehreren Projekten nutzen zu können.

Enertex® EibStudio 4

Dieses Kapitel gibt einen Überblick über den Programmaufbau und die grundlegende Bedienung.

Sofern nicht ausdrücklich unterschieden, bezieht sich die Bezeichnung EibPC im Folgenden auf alle Generationen. Die Bezeichnung EibStudio ohne explizite Versionsangabe bezieht sich auf Version 4.

Installation

Enertex® EibStudio 4, ebenso wie Enertex® EibStudio 3, benötigt keine Installation sondern muss lediglich entpackt werden. Prüfen Sie in jedem Fall, dass Sie die nötigen Rechte in dem Verzeichnis besitzen. Dies gilt insbesondere, wenn Sie das Enertex® EibStudio an einen zentralen Ort (z.B. das allgemeine Programmverzeichnis) verschieben möchten.

Beachten Sie auch, dass die Datei *eibparser.exe* im Unterverzeichnis *bin* ausgeführt werden können muss.

Fenstermenü

Das Fenstermenü beinhaltet alle zentralen Funktionen, die unabhängig von einen Projekt aufgerufen werden können, wie die Einstellungen oder Hilfe.

Ist ein Projekt geöffnet, befinden sich dort zusätzlich häufig verwendete Funktionen, wie das Kompilieren und Übertragen des Programms.

Projektübersicht

Hier haben Sie die Möglichkeit, neue Projekte anzulegen oder bestehende Projekte aus Enertex® EibStudio 3 oder Enertex® EibStudio 4 zu importieren. Projekte kapseln alle Informationen, die einen bestimmten Enertex® EibPC bzw. eine Installation betreffen. Alle Projekte werden im Projektverzeichnis gespeichert.

Verändern Sie keine Dateien innerhalb des Projektverzeichnisses!

Projektverzeichnis

Beim ersten Start müssen Sie den Speicherort des Projektverzeichnisses wählen. Stellen Sie sicher, dass sie die nötigen Berechtigungen (lesen, schreiben) für das Verzeichnis besitzen.

Enertex® EibStudio 4 greift nur auf Dateien im Projektverzeichnis und dem Konfigurationsverzeichnis zu. Sie können das Projektverzeichnis jederzeit in den Einstellungen ändern.

Projekt aus Enertex® EibStudio 3 importieren

Enertex® EibStudio 3-Projekte bestehen aus einer oder mehreren Quellcodedateien mit der Endung .epc. Dabei werden alle weiteren Quellcodedateien vom Hauptprogramm über #include-Anweisungen eingebunden.

Starten Sie den Importvorgang und wählen Sie zunächst das Hauptprogramm, das bisher mit Enertex® EibStudio 3 bearbeitet wurde. Wählen Sie im nächsten Dialog das Verzeichnis, in dem sich die Programmdatei von Enertex® EibStudio 3 befindet. Dies ist nötig, falls das Hauptprogramm #include-Anweisungen mit relativen Pfadangaben enthält.

Daraufhin wird ein neues Projekt angelegt. Der Name entspricht dem Dateinamen des Hauptprogramms. Werden referenzierte Dateien nicht gefunden, so bricht der Import-Vorgang mit einem entsprechenden Hinweis ab, um welche Datei es sich handelt. Prüfen Sie deren Pfad und editieren Sie falls nötig das zu importierende Programm. Starten Sie anschließend den Vorgang neu.

Wurde der Import erfolgreich abgeschlossen, so befinden sich folgende Einstellungen bzw. Einträge im neu angelegten Projekt:

[ETS-ESF]: Die .esf-Datei wird importiert und die Gruppenadressen unter **Objekte** angelegt

[InitGA]: Die Gruppenadressen unter **Objekte** werden mit dem Initialisierungsflag versehen

[FTP], [MailConf], [Performance], [VPN], [HTTPS], [Location]: Die Einstellungen werden unter **Projekteinstellungen → EibPC** bzw. **Projekteinstellungen → Fernzugriff** übernommen

[MacroLibs]: Die Quelldateien der Bibliotheken werden als **Eigene Makros** unter **Experte** angelegt. Die meisten Makrobibliotheken aus Enertex® EibStudio 3 sind bereits als **Interne Makros** vorhanden. Diese werden deaktiviert, falls es Namensüberschneidungen mit den eigenen Makros gibt.

Das gesamte Programm wird als neues Programm unter **Experte** angelegt. Die o.g. Sektionen werden auskommentiert, und die Sektionen [EibPC], [Macros] und [Webserver] werden durch #addto [EibPC] usw ersetzt.

Einstellungen

Konfigurationsverzeichnis

Die projektübergreifenden Einstellungen werde über das Fenstermenü **Bearbeiten → Einstellungen** geöffnet. Sie gelten für alle Projekte und werden als Datei *eibstudio.json* im Konfigurationsverzeichnis gespeichert. Je nach Betriebssystem befindet sich dies unter:

- Windows 10: *%LOCALAPPDATA%\eibstudio\User Data\Default*
- Linux ~/.config/eibstudio/Default/
- OSX: ~/Library/Application Support/eibstudio/Default

Bedienelemente

Navigationselemente

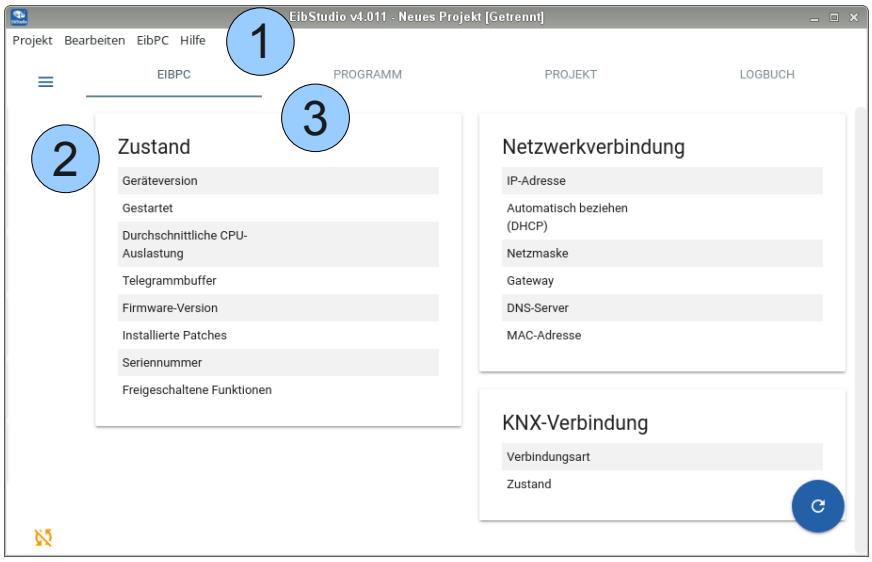


Abbildung 9: Übersicht

Erweiterte Navigation

Ist ein Projekt geöffnet so beinhaltet das Fenstermenü (1) zusätzlich häufig verwendete Projekt-Funktionen. Das Projektmenü wird über den Projektmenü-Knopf (2) eingeblendet. Diese wird verwendet um zwischen den verschiedenen Aufgaben innerhalb eines Projekts zu wechseln. Zur weiteren Strukturierung werden Registerkarten (3) verwendet.

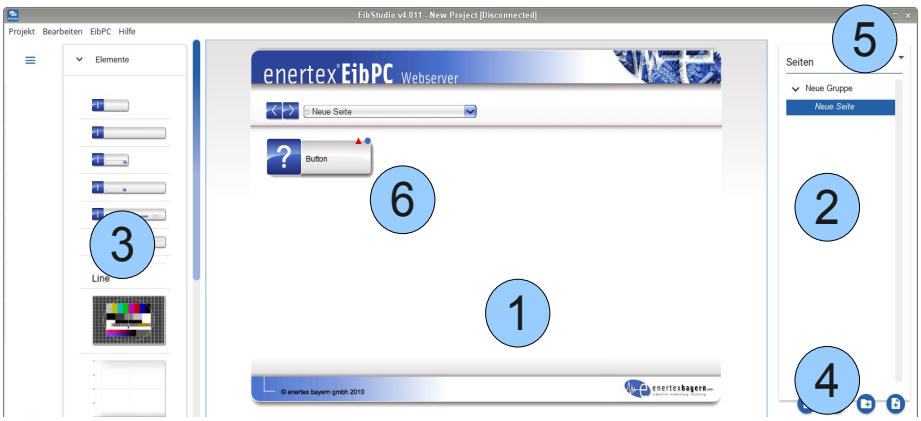


Abbildung 10: Erweiterte Navigation

Für Logik, Visualisierung und Experte wird eine erweiterte Navigation wie in Abbildung 10 dargestellt verwendet.

Der zentrale Editor-Bereich (1) stellt den aktuell in (2) ausgewählten Eintrag dar.

Einträge in der Liste links (3) können in (1) durch `Klicken` oder Ziehen mit der Maus hinzugefügt werden. Um sie aus (1) zu entfernen, mit einem `Klick` markieren und `Entf` drücken. Bei gedrückter `Shift` oder `Strg`-Taste können mehrere Elemente markiert werden.

Einträge in (2) werden über die Knöpfe in (4) hinzugefügt, entfernt, usw.

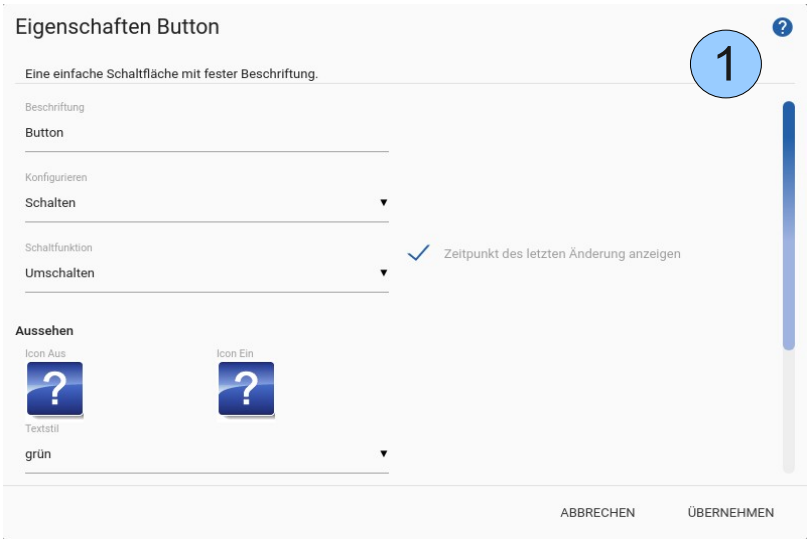
Um den Editor-Bereich (1) zu vergrößern, kann (2) bei Bedarf ausgeblendet werden (5).

Ein Doppelklick auf Einträge in (1) und (2) öffnen den entsprechenden Dialog zum Bearbeiten ihrer Eigenschaften.

Das rote Dreieck bei (6) signalisiert, dass die Konfiguration nicht korrekt oder nicht vollständig ist, und dass dadurch das Programm wahrscheinlich nicht wie erwartet funktioniert.

Der blaue Kreis signalisiert, dass das Element seit der letzten Speicherung geändert wurde.

Eigenschaftsdialog



Im Eigenschaftsdialog werden die Eigenschaften eines Objekts geändert. Die meisten Dialoge verfügen über eine integrierte Hilfe (1), die bei Bedarf eingeblendet werden kann.

Im Folgenden werden die Funktionen im Projektmenü näher beschrieben.

Übersicht

In der Übersicht werden Informationen zum verbundenen Enertex® EibPC und zum zuletzt kompilierten Programm dargestellt. In Anlehnung an die ETS können zudem projektbezogene Verwaltungsinformationen und Logbuch-Einträge gespeichert werden. Diese sind nicht mit einem internen Zustand verknüpft sondern dienen lediglich der Dokumentation.

Objekte

Die Objektübersicht listet alle Gruppenadressen (Manuelle Gruppenadressen werden nicht in der Liste aufgeführt), Variablen und vordefinierten Konstanten auf. Eine genaue Erklärung zu den einzelnen Objekttypen finden Sie im Kapitel Objekte. Bei einem neuen Projekt sind diese Listen zunächst leer. Bei jeder erfolgreichen Kompilierung werden die Listen aktualisiert. **Treten also bei der Kompilierung Fehler auf, so müssen diese zunächst behoben werden.**

Die Gruppenadress- und Variablenübersicht bieten zudem die Möglichkeit, den aktuellen Zustand abzufragen. Wählen Sie dazu einen Eintrag und klicken Sie auf das entsprechende Symbol oben rechts. Alternativ: `Doppelklick` zur Wertabfrage, `Strg+Klick` um ein Bustelegramm zu senden, bzw. den Variablenwert zu ändern.

Der Debugger bietet weitere Möglichkeiten wie das Abfragen mehrerer Objekte oder das Senden von Leseanforderungen.

Gruppenadressimport

.knxproj oder .esf aus ETS4/ETS5

Um die Konsistenz mit anderen Busteilnehmern zu gewährleisten, können Gruppenadressen nicht angelegt werden, sondern müssen aus der ETS importiert werden. Dazu stehen grundsätzlich zwei Formate zur Verfügung. Der bevorzugte Weg ist der Import der .knxproj-Datei. Diese erzeugen sie, indem Sie das ETS-Projekt exportieren.

Aktuell werden ETS4 und ETS5-Projekte unterstützt. Voraussetzung ist, dass das Projekt **nicht durch ein Passwort geschützt** ist und dreistufige Gruppenadressen verwendet.

Datentyp-Inferenz

Beim Import wird versucht, jeder Gruppenadresse einen Datentyp zuzuordnen. Besitzt weder die Gruppenadresse noch die Verknüpfungen einen DPT, so wird ein vorzeichenloser Ganzzahltyp der entsprechenden Größe des Kommunikationsobjektes angenommen. Bei nicht-verknüpften Gruppenadressen kann kein Typ bestimmt werden. **Typenlose Gruppenadressen können nicht verknüpft werden.**

Alternativ steht der .esf-Import (wie unter Enertex® EibStudio 3) zur Verfügung. Dieser beinhaltet allerdings nur unzureichende Typinformationen und sollte nur verwendet werden, falls der .knxproj-Import nicht möglich ist. Die .esf-Datei wird in der ETS mit der Funktion „OPC-Export“ erstellt.

Der Typ jeder Gruppenadresse kann beliebig geändert werden.

Falls ein falscher Typ gewählt wird, findet keine korrekte Interpretation der Telegramme statt!

Topologie

Beim .knxproj-Import wird zudem die Topologie des Projekts ausgelesen und als Baum dargestellt. Dies ist insbesondere für die Zuordnung von phsikalischen Adressen im Gruppenmonitor notwendig (S. 38).

Interne Variablen

Variablen können gezielt vom Benutzer angelegt werden, um einen internen Zustand zu speichern, ohne ihn auf den KNX-Bus senden zu müssen. Außerdem werden interne Variablen für die Logik, die Visualisierung, sowie für Makros in Expertenprogrammen deklariert. Diese Variablen werden normalerweise ausgeblendet, können aber bei Bedarf in der Variablenübersicht sowie im Debugger eingeblendet werden.

Konstanten

Im Enertex® EibStudio sind Variablen vordefiniert, um das Erstellen eines Anwenderprogramms möglichst einfach zu gestalten. Diese Variablen sind im Enertex® EibStudio unter **Objekte** → **Konstanten** gelistet. Vordefinierte Variablen können nicht überschrieben werden.

Wird ein neues Projekt angelegt, so muss das (leere) Projekt ein erstes Mal kompiliert werden, um alle Konstanten zu laden.

Logik

Die Logikmodellierung erlaubt es auf einfache Weise, Objekte mit Operationen zu verknüpfen.
Folgende Begrifflichkeiten werden hierbei unterschieden:

Definitionen

Knoten
: Hat einen festgelegten Typ und repräsentiert entweder ein Objekt oder eine Operation.

Eingang
: Anfasser auf der linken Seite eines Knotens. Kann mit mindestens einem Ausgang durch eine Kante verbunden werden, jedoch nicht mehrfach mit demselben Ausgang.

Ausgang
: Anfasser auf der rechten Seite eines Knotens. Kann mit mindestens einem Eingang durch eine Kante verbunden werden, jedoch nicht mehrfach mit demselben Eingang.

Port
: Eingang oder Ausgang.

Kante
: Verbindet genau einen Ausgang eines Knoten mit genau einem Eingang eines anderen Knotens.

Auslöser
: Port, der beim Wechsel von 0b01 auf 1b01 die entsprechende Funktion des Knotens startet. Bleibt der Auslöser auf 1b01, so wird die Funktion nicht erneut ausgeführt.

Sammeleingänge

Besitzt ein Port mehrere Kanten, so sind diese gleichwertig. Knoten, bei denen die Parameterreihenfolge keine Rolle spielt (z.B. ADDITION), besitzen der Übersichtlichkeit halber lediglich einen einzigen Eingang, mit dem alle Parameter verbunden werden.

Typen

Jeder Port besitzt einen Typ. Es können nur kompatible Ports miteinander verbunden werden. Folgende Möglichkeiten bestehen bei der Angabe des Typs und der möglichen Verbindung:

*: Alle Typen
: Beliebige Typen

b, u, f: Typklasse
: Beliebige Typen der gleichen Klasse

b01, u08, f16: Spezifischer Typ
: Verbindung mit genau diesem Typ

Beispiele:

Ein Eingang vom Typ b01 darf mit Ausgängen der Typen *, b, b01 verbunden werden.

Ein Ausgang vom Typ u,s darf mit Eingängen der Typen *, u, uXX, s, sXX verbunden werden.

Entfernen von Kanten

Dabei gilt allerdings: Zur Übersetzung muss ein spezifischer Datentyp feststehen. Mit jeder neuen Knotenverbindung werden die Typen der verbundenen Knoten und ihrer Vorgänger und Nachfolger aktualisiert.

Wird eine Verbindung entfernt, bleiben die Typen erhalten. **Gegebenenfalls muss also der Knoten entfernt und neu hinzugefügt werden.**

Konvertieren

Falls inkompatible Typen verbunden werden sollen, so müssen diese explizit konvertiert werden (Knoten TYPKONVERTIERUNG). Grundsätzlich kann jeder Typ in jeden anderen Typ konvertiert werden. Ein eventueller Informationsverlust muss mit bedacht werden.

Mehrere Logiken

Logiken können beliebig aufgeteilt werden. Einzelnen Logiken sind gleichberechtigt. Werden also dem gleichen Objekt in unterschiedlichen Logiken Werte zugewiesen, so hat das Objekt jeweils den Wert der zeitlich zuletzt ausgewerteten Zuweisung. Findet die Zuweisung gleichzeitig statt, so ist das Ergebnis undefiniert.

Beispiel: Lichtautomatik

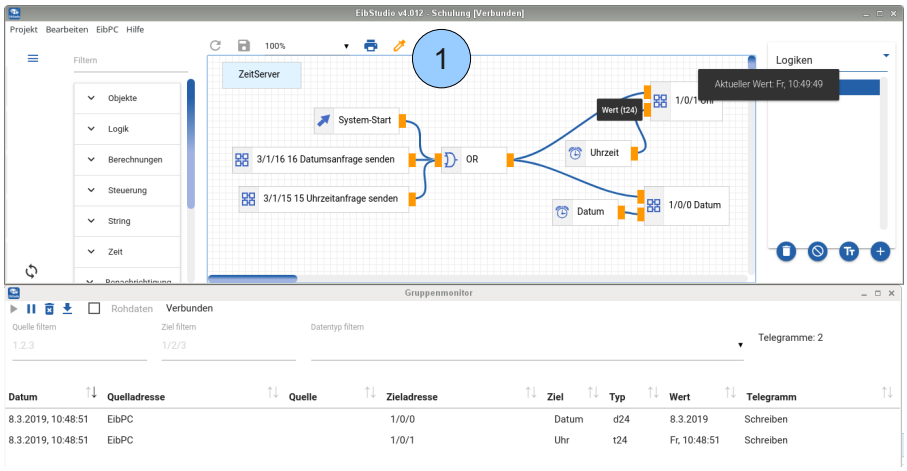
Abbildung 11: Einfache Lichtautomatik

Einfacher Treppenhausautomat: 10 Minuten nach dem Einschalten soll das Licht automatisch wieder ausgeschaltet werden.

- Starten Sie mit einem leeren Projekt, importieren Sie Gruppenadressen und kompilieren Sie das Projekt. Dadurch werden die vordefinierten Konstanten angelegt.
- Erstellen Sie eine neue Logik
- Fügen Sie die Knoten hinzu:
  Objekte/Gruppenadresse
  Objekte/Gruppenadresse
  Objekte/Konstante
  Logik/AND
  Zeit/Verzögerung
- Konfigurieren Sie den ersten Gruppenadressknoten so, dass er den aktuellen Wert der Status-Gruppenadresse ausgibt
- Der zweite Gruppenadressknoten soll bei einem Externen Auslöser Schreiben
- Es soll die Konstante AUS, die dem Wert 0b01 entspricht, geschrieben werden. Wählen Sie diesen in Konstante.
- Konfigurieren Sie die Verzögerung wie gewünscht.
- Verknüpfen Sie die Knoten wie in Abbildung 11.
- Kompilieren und starten Sie das Projekt.

Die Logik wird bei jeder Zustandsänderung der Objekte und Knoten (teilweise) neu ausgewertet (Details siehe Validierungsschema). Ändert sich der Status von 0b01 auf 1b01, wird der Verzögerungstimer neu gestartet. Sobald die Verzögerungszeit abgelaufen ist, geht ihr Auslöser-Ausgang von 0b01 auf 1b01. Ist zu diesem Zeitpunkt der Status immernoch 1b01, so wird der Wert AUS (oder 0b01) als Bustelegramm an die Gruppenadresse gesendet.

Debug-Modus



A

bbildung 12: Debug-Modus im Logikeditor

Der Logikeditor legt Variablen an, die normalerweise vor dem Anwender verborgen werden (S. 27). Um trotzdem den aktuellen Zustand jedes Knotens innerhalb einer Logik zu erfahren, kann der Debug-Modus aktiviert werden (1).

Es werden alle Ports farbig hervorgehoben. Durch einfaches Klicken auf einen Port wird der aktuelle Objektzustand abgefragt. Mittels `Strg+Klick` kann der Zustand auch gezielt verändert werden.

Um ungewünschte Reaktionen anderer Geräte zu vermeiden, empfiehlt sich die Simulation für umfangreiche Tests (S. 38).

Die Logik in Abbildung 12 zeigt den EibPC als Zeitgeber für den KNX-Bus. Beim Systemstart (also nachdem der EibPC sich mit einem NTP-Server synchronisiert hat) wird Datum und Uhrzeit mit den entsprechenden DPTs auf den Bus gesendet.Durch die verknüpften Gruppenadressen kann die aktuelle Zeit auch manuell gesendet werden. Die aktuelle Uhrzeit wurde gerade abgefragt und wird oben rechts eingeblendet. Der Gruppenmonitor zeigt die beiden

Telegramme.

Visualisierungsobjekte

Falls die vordefinierten Visualisierungselemente und Funktionen nicht ausreichen, ist es mit dem Logikeditor sehr leicht, auf Elemente zuzugreifen. Dazu muss in der **Visu** in den Eigenschaften des Elements nur die Konfigurationsoption „Mit Logik verknüpfen“ gewählt werden.

Dadurch steht das Element in der **Logik** zur Verfügung. Fügen Sie den Visualisierungsknoten des entsprechenden Typs hinzu. In dessen Eigenschaften können Sie nun das Visualisierungselement auswählen.

Hinweis: Falls Sie bei komplexeren Logiken auch eine Rückmeldung auf das Visualisierungselement verwenden, können Sie den Visualisierungsknoten auch duplizieren (Kopieren `Strg+c`, Einfügen `Strg+v`) und auf beiden Seiten Ihrer Logik platzieren. Es genügt dabei, jeweils nur die Eingänge oder Ausgänge zu verbinden.

Visu

Visualisierungen können in Enertex® EibStudio 4 bequem und schnell erstellt werden.

Die Visualisierung ist aufgeteilt in Gruppen von Seiten. Jede Seite ist in Größe und Design konfigurierbar. Die Reihenfolge der Seiten entspricht der späteren Darstellung im Webserver und kann durch Ziehen mit der Maus verändert werden.

Grundsätzlich wird zwischen Elementen und Funktionen unterschieden.

Elemente

Elemente sind alle darstellbaren Schaltflächen, Graphen usw. Das Verhalten jedes Elements kann in seinen Eigenschaften konfiguriert werden. Die meisten Funktionen lassen sich direkt im Element realisieren, z.B. ein einfaches Schaltelement, das eine Gruppenadresse umschaltet, oder einen Slider für einen Prozentwert.

Funktionen

Funktionen sind vorkonfigurierte Elemente oder Gruppen von Elementen mit eigenen Eigenschaften. Soll eine Funktion verwendet werden, müssen alle Elemente, die zu dieser Funktion gehören, auf der Seite Platz finden. Anderenfalls wird die gesamte Funktion nicht eingefügt.

Die Platzierung der Elemente in der Vorschau kann durch Ziehen mit der Maus geändert werden und entspricht der späteren Darstellung im Browser.

Eigene Vorlagen

Die aktuell angezeigte Seite kann als eigene Vorlage in den Benutzereinstellungen gespeichert werden. Dadurch ist es möglich, die gleichen Seiten in mehreren Projekten zu nutzen. Diese Vorlagen können nicht editiert werden, sondern müssen bei Bedarf als neue Vorlage gespeichert werden. Es werden alle Objektverknüpfungen kopiert. Unterscheiden sich die Gruppenadressen, müssen sie vor der übersetzung angepasst werden.

Vorlagen

Zusätzlich stehen vordefinierte Vorlagen, z.B. für den Enertex® KNX SmartMeter bereit.

Verknüpfung mit Logik und Experte

Falls die vordefinierten Funktionen nicht ausreichen, ist es über eine Verknüpfung des Visualisierungselements mit der **Logik** oder dem **Experten** möglich, das Verhalten selbst zu definieren.

Hierbei ist die Verwendung in der Logik sehr einfach, mit bewusst reduzierter Flexibilität in der Veränderung des Aussehens. Es kann zwischen dem Grundzustand und dem Aktiv-Zustand gewechselt werden. Es ist also z.B. nicht möglich, zwischen rotem und grünem Text für einen Button umzuschalten, der Grundzustand des Textes ist stets grau (siehe S. 31).

Eine Verknüpfung mit dem Experten bietet die meiste Flexibilität. Hier wird der Grundzustand festgelegt und eine eindeutige Variable definiert, über die im Experten auf das Visualisierungselement zugegriffen wird (siehe S. 33).

Experte

Der Experte bietet vollen Zugriff auf alle Funktionen des Enertex® EibPCs. Eine detaillierte Beschreibung der Funktionen finden Sie im Kapitel Befehle und Funktionen.

Es können beliebig viele Expertenprogramme angelegt werden, die alle im globalen Kontext kompiliert werden. Das bedeutet auch, dass eine Variable, die in einem Expertenprogramm angelegt wurde, in allen anderen verwendet werden kann (und nicht mehr neu definiert!).

Autovervollständigung

Der Texteditor unterstützt die Programmierung, indem automatisch passende Funktionen oder Objekte bei der Eingabe vorgeschlagen werden. Diese Listen werden beim Kompilieren generiert. Legen Sie also eine neue Variable an, so wird diese erst nach dem ersten Kompiliervorgang für die weitere Verwendung vorgeschlagen.

Für den Zugriff auf Gruppenadressen genügt es, doppelte Anführungszeichen für den Beginn einer Gruppenadresse, gefolgt von aussagekräftigen Namensbestandteilen einzugeben, z.B. "123“ um Vorschläge für die Gruppenadresse 1/2/3 zu erhalten.

Makros

Makros sind Funktionsblöcke, mit denen Expertenprogramme zusätzlich strukturiert werden können. Zudem wird eine umfangreiche Sammlung an Makros in Form von internen Makrobibliotheken mitgeliefert.

Eigene Visualisierungsseiten

Innerhalb von Expertenprogrammen können Sie auf den Webserver zugreifen, indem Sie die Direktive #addto [Webserver] verwenden.

Möchten Sie eigene Visualisierungsseiten oder globale Visualisierungselemente im Experten nutzen, so muss gewährleistet werden, dass diese nicht mit der **Visu** kollidieren.

Eine korrekte Vergabe der IDs ist insbesondere wichtig, wenn ein Enertex® EibStudio 3-Projekt importiert wurde, und zu den bereits vorhandenen „programmierten“ Visualisierungsseiten über die **Visu** generierte hinzugefügt werden sollen.

Prüfen Sie dazu, welche IDs bereits verwendet wurden, und tragen Sie die erste freie ID in den **Projekteinstellungen → IDs** (S. 37) ein.

Zugriff auf Visualisierungselemente

Für den Zugriff auf einzelne Visualisierungselemente innerhalb eines Expertenprogramms haben Sie auch die Möglichkeit, diesem einen eindeutigen Variablennamen zuzuordnen. Wählen Sie dazu den passenden Konfigurationseintrag des Elements innerhalb der **Visu** und übersetzten Sie das Projekt, um die Variablenliste zu aktualisieren.

Dieser Name muss den Anforderungen an Variablennamen (S. 44) genügen.

Beim Kompilieren der Visualisierung wird diesem Variablennamen die ID des Visualisierungselements zugewiesen. Handelt es sich um ein seitenbezogenes Element (z.B. Button, Shifter), so wird außerdem eine Variable mit dem gleichen Namen und der Endung „_P“ mit der Seiten-ID angelegt.

Beispiel:

Als eindeutige Variable wird einem Button *ButtonVar* zugewiesen. Da es sich um ein seitenbezogenes Element handelt, wird außerdem die Variable *ButtonVar_P* angelegt. Nach dem nächsten Kompilieren existieren beide Variablen.

Diese können nun von den Webserverfunktionen (S. 139) verwendet werden:

```
pdisplay(ButtonVar, $MeinButton$, INFO, ACTIVE, GREEN, ButtonVar_P)
```

Falls Sie eigene Visualisierungsseiten in Expertenprogrammen definiert haben, müssen Sie unbedingt die Start-IDs für die Visu entsprechend setzen (S. 37).

Seitenbezogene Visualisierungselemente:

Button, *Shifter*, *Multibutton*, *Multishifter*, *Slider*, *Picture*, *Wertediagramm*, *Zeitdiagramm*.

Nicht-seitenbezogen:

Webinput und *Weboutput*.

Syntax

Variablendefinition

Variablen werden angelegt, indem ihnen ein Initialwert zugewiesen wird. Die Definition muss dabei eindeutig sein. Variablen sind auf Seite 44 detailliert beschrieben.

```
Var=1b01
```

Gruppenadressen

Um auf den aktuellen (interne) Wert einer Gruppenadresse zuzugreifen, wird der Objektname verwendet. Dieser wird als Name unter **Objekte → Gruppenadressen** angezeigt und setzt sich aus dem in der ETS vergebenen Namen und der Gruppenadresse (Haupt-, Mittel-, Untergruppe), getrennt durch ein – zusammen (siehe S. 44). Der Wert der Variablen Var ändert sich mit dem Zustand der Gruppenadresse.

```
Var="GA-1/2/3"
```

if-Anweisungen

Die einfachste Form der if-Anweisung

```
if "GA-1/2/3" then Var=EIN endif
```

eignet sich besonders für einfache wenn-dann-Fälle.

Allgemein ist die if-Anweisung definiert als

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

Die Abfragebedingung muss ein Ausdruck vom Typ b01 sein.

Eine Anweisung entspricht dabei einer Zuweisung, einem Funktionsaufruf oder einem Makroaufruf. Diese werden durch „;“ (Semikolon) getrennt.

Falls zwischen den Anweisungen zur besseren Lesbarkeit Zeilenumbrüche eingefügt werden, muss der Anweisungsblock in {} stehen:

```
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
```

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
```

Online Debuggen zur Laufzeit

Wenn zur Laufzeit die Zustandsänderung beobachtet werden soll, so empfiehlt es sich mit Hilfe von UDP Datagrammen und einem netcat Client (vgl. https://de.wikipedia.org/wiki/Netcat) zu debuggen.

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:

Einen String mit CR auf einen UDP Client schicken

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
```

Projekteinstellungen

In den Projekteinstellungen befinden sich alle Einstellungen, die einen konkreten Enertex® EibPC bzw. eine Installation betreffen.

Änderungen an den Einstellungen des Enertex® EibPCs müssen gezielt auf diesen übertragen werden. Dies geschieht, je nach Einstellungen, entweder beim Übertragen des Programms (P), oder durch Betätigen einer Schaltfläche (S)

Suche

Sie können über alle installierten Netzwerkkarten nach vorhandenen EibPCs suchen.

KNX-Verbindung

(P)

Wählen Sie die passende Verbindungsart. Eine Änderung dieser Einstellung wird mit der nächsten Programmübertragung aktiv.

Netzwerkverbindung

Bei Änderungen an der Netzwerkverbindung wird der EibPC neu gestartet. Sollten Sie den EibPC nicht mehr erreichen, können Sie ihn auf Werkseinstellungen (S. 20) zurücksetzten.

Namensauflösung

Für einige der Netzwerkfunktionen (sendmail, resolve) benötigt der Enertex® EibPC einen Zugang zu einem DNS Server.

Datum und Uhrzeit

Für das korrekte Funktionieren der Zeitfunktionen ist es wichtig, dass der Enertex® EibPC eine korrekte Uhrzeit besitzt. Es bietet sich hier an, diese Zeit mit anderen Busteilnehmern synchron zu halten. Dazu kann der Enertex® EibPC Zeitinformationen vom KNX-Bus verwenden, um seine eigene Uhr zu synchronisieren. Ist in der KNX-Installation keine zuverlässige Zeitquelle vorhanden, kann dies auch der EibPC übernehmen, und seine Systemzeit als Telegramm auf den Bus senden.

Zusätzlich kann der Enertex® EibPC seine Systemuhr über das NTP-Protokoll mit einem Server im lokalen Netzwerk oder im Internet synchronisieren.

Falls die Verwendung eines NTP-Servers konfiguriert ist, wird dieser zum Setzten der Zeit verwendet, auch wenn zwischenzeitlich die Uhr manuell verstellt wurde. Die Synchronisierung erfolgt initial beim Programmstart. Wenn nach fünf Minuten kein NTP-Server erreicht werden konnte, beginnt das Anwendungsprogramm trotzdem.

Position

(P)

Der Enertex® EibPC berechnet für das aktuelle Jahr den Sonnenstand voraus, so dass zu jedem 5-Minuten-Intervall die Sonnenposition zur Verfügung steht. Dazu werden die Koordinaten des Aufstellortes benötigt. Müssen die Sonnendaten neu berechnet werden, dauert dies ca 5 Minuten.

SHUTDOWN-Variable

Es besteht die Möglichkeit, bei jedem Neustart des EibPC-Programms die Variable *SHUTDOWN* auf 1b01 zu setzen. Dies ermöglicht das persistente Speichern von Daten auf dem Flash. Eine Verzögerung von 5 Sekunden ist hierbei eine sinnvolle Konfiguration.

FTP

(P)

Der Enertex® EibPC kann alle Telegramme vom KNX-Bus zur Protokollierung auf einen FTP-Server hochladen. Dazu muss der Server auf Port 21 auf eingehende Verbindungen warten. (P)

E-Mail

(P)

Um E-Mails senden zu können, müssen Sie hier die Verbindungseinstellungen festlegen. (P)

Sicherung

Es ist möglich, bei jeder Programmübertragung das aktuelle Projekt zur Sicherung abzulegen. Über die entsprechende Schaltfläche kann das Projekt heruntergeladen werden. Sie können es bei Bedarf als neues Projekt importieren.

Dateien

Um eigene Grafiken in der Visualisierung verwenden zu können, müssen diese auf den Enertex® EibPC übertragen werden. Die Grafikdateien werden außerdem im Projektverzeichnis gespeichert. Wird das Projekt auf einen anderen Enertex® EibPC übertragen, so werden automatisch die hinzugefügten Dateien synchronisiert. Gleiches gilt in die Rückrichtung. Sind also Dateien auf dem Enertex® EibPC gespeichert, die noch nicht dem Projekt hinzugefügt wurden, werden diese automatisch heruntergeladen.

HTTPS

(S)

Mit dem Enertex® EibPC ist eine sichere Kommunikation via HTTPS zwischen WebServer und Browser möglich. Dazu muss lediglich ein SSL-Zertifikat auf dem EiBPC vorhanden sein und ein Benutzer mit Passwort eingetragen werden.

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.

VPN

(S)

Der Enertex® EibPC verfügt über einen OpenVPN-Server. Auch hierfür muss vorher ein Zertifikat generiert werden.

IDs

Die Firmware des EibPCs organisiert den Zugriff auf interne Objekte über eindeutige Nummern, sog. IDs. Diese werden bei der Definition angegeben und müssen beim Zugriff verwendet werden, um das entsprechende interne Objekt anzusprechen.

Export und Import

Projekte werden über das Fenstermenü **Projekt → Exportieren** gepackt. Im Vergleich zum **Hilfe → Export für Support** beinhaltet der Export auch alle „sensiblen“ Daten wie das E-Mail-Passwort.

Debugger

Der Debugger kann über das Fenstermenü **EibPC → Debugger** geöffnet werden. Er bietet die Möglichkeit, den internen Status von Verarbeitungsobjekten zu verändern, Gruppentelegramme zu senden und aktuelle Werte abzufragen.

Gruppenmonitor

Der Gruppenmonitor kann über das Fenstermenü **EibPC → Gruppenmonitor** geöffnet werden. Er zeigt alle Telegramme, die der EibPC empfängt. Wurde eine .knxproj-Datei importiert, so werden zu den physikalischen Adressen die Gerätenamen angezeigt.

Es werden stets die letzten 100 Telegramme angezeigt. Diese können als .csv-Datei exportiert werden.

Langzeitpuffer

In Ergänzung zum Gruppenmonitor, mit dem stets nur die letzten Telegramme ausgewertet werden können, Speichert der EibPC die letzten 500.000 Telegramme (bei Verwendung der Visualisierung: 150.000 Telegramme) im Langzeitpuffer. Dieser kann über das Fenstermenü **EibPC → KNX-Telegramme abholen** ausgelesen und als .csv-Datei gespeichert werden.

Ereignisspeicher

Sobald vbei der Verarbeitung im Enertex® EibPC etwas unvorhergesehenes passiert, wird das Ereignis protokolliert. Das Ereignisprotokoll kann über das Fenstermenü **EibPC → Ereignisse** ausgelesen werden. Zur Erklärung der Ereignisse siehe S. 227.

Simulation

Bei komplexeren Logiken ist es für die Erstellung hilfreich, diese vorab simulieren zu können. Dazu genügt es, die Verbindung zum KNX-Bus auf FT1.2 Busmonitor-Modus zu stellen (S. 36) und die Schnittstelle abzustecken, falls diese verwendet wird. Im Gruppenmonitor werden nun weiterhin alle Telegramme angezeigt, die der EibPC sendet, ohne andere Geräte zu beeinflussen.

Um die Kommunikation mit anderen Geräten nachzubilden, können Sie z.B. in der Logik auf Leseanfrage reagieren und gezielt Testwerte senden. Eine einfache Simulation ist in Abbildung 9 dargestellt.

Fügen Sie drei Gruppenadress-Knoten hinzu und konfigurieren Sie sie wie folgt:

1. Ereignis erzeugen, sobald eine Leseanfrage an die Gruppenadresse gesendet wird
2. Den aktuellen Wert am Ausgang ausgeben
3. Ein Antwort-Telegramm senden, mit der Leseanfrage als externem Trigger

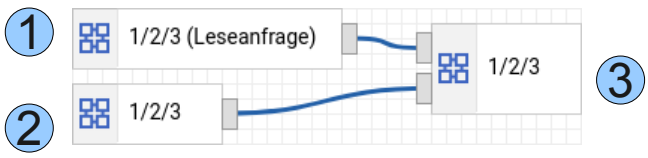


Figure 1: Simulation Leseanfrage

Auf diese Weise können Sie schnell komplette Testumgebungen schaffen, z.B. für eine automatische Beschattung. Dies ist deutlich einfacher, als im Debugger immer wieder manuell Gruppenadressen zu beschreiben.

Beachten Sie, dass ohne Busverbindung Leseanforderungen beim Start nicht beantwortet werden können. Pro Anfrage wird 1,5 Sekunden gewartet, was den Systemstart unnötig verzögert. Um diese unnötige Wartezeit zu vermeiden, können Sie in den Einstellungen der Bus-Verbindung die Initialisierung komplett deaktivieren.

Achten Sie aber darauf, diese wieder zu aktivieren, wenn Sie die Simulation beenden.

Objekte

Objekte repräsentieren Zustände, auf die der Enertex® EibPC mit der Erzeugung neuer Zustände reagieren kann. Vereinfacht ausgedrückt lassen sich die meisten Aufgaben auf *Wenn dies, dann tue jenes* abbilden. Objekte sind dabei sowohl die Bedingung als auch das Ziel.

Es gibt im Enertex® EibPC zwei verschiedene Arten von Objekten: Gruppenadressen und Variablen.

Gruppenadressen

Gruppenadressen sind Objekte, die für andere Busteilnehmer relevant sind. Jeder Teilnehmer muss die für ihn relevanten Zustandsänderungen, die als Bustelegramme an Gruppenadressen gesendet werden, mit seinem intern gespeicherten abgleichen und bei Bedarf auswerten.

Variablen

Darüber hinaus speichern Busteilnehmer eigene interne Zustände, die zunächst auch nur intern verarbeitet werden. Der Enertex® EibPC kennt diese als Variablen.

Beispiel: Ein Schaltaktor reagiert auf eine Gruppenadresse, die mit seinem Kommunikationsobjekt *Schalte Kanal 1 um* verknüpft ist. Der Schaltaktor weiß dabei seinen internen Zustand, den er benötigt, um entweder ein oder aus zu schalten. Außerdem informiert er andere Busteilnehmer über die Änderung, indem er ein Status-Telegramm auf den Bus sendet.

Bei diesem Schaltvorgang sind die Gruppenadressen für Aktor und Status, aber auch der interne Schaltzustand beteiligt.

Der Enertex® EibPC als universelle Logikmaschine funktioniert nach dem gleichen Prinzip, nur dass das Regelwerk durch das Anwenderprogramm und damit Sie selbst vorgegeben wird. Es kann also jedes Objekt mit jedem anderen Objekt über eine Vielzahl interner Funktionen verknüpft werden.

Datentyp

In der ETS liegen Datentypen als Datenpunkttyp (DPT) vor. Dieser spezifiziert die Größe eines Objekts und legt ggf. auch seine Interpretation fest. Ein 1-Bit Objekt vom Typ DPT 1 kann z.B. als DPT 1.001 Ein/Aus oder DPT 1.008 Hoch/Runter interpretiert werden.

Beim Import aus der ETS werden diese Typen auf interne abgebildet, die ausschließlich ihre Größe beschreiben. Diese setzen sich aus dem Typ und der Bitlänge wie folgt zusammen:

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 91.

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 90).

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	Wertebereich	Datenpunkttyp	EIS Datentyp
Binär	b01	1b01	Schaltaktor, Jalousieaktor	0, 1	1000	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.

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 90).

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
```

Gruppenadressen

Der sinnvollste Weg, Gruppenadressen einem Enertex® EibStudio-Projekt hinzuzufügen, ist der ETS-Import (S. 27).

Manuelle Gruppenadresse

Gruppenadressen können auch manuell definiert werden. Dazu steht die Gruppenadresse in einfachen Anführungszeichen, gefolgt von Datentyp, z.B. '1/2/3'b01.

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. 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.

Um sich mit dem KNX Bus vor der eigentlichen Verarbeitung zu synchronisieren, kann der Anwender Gruppenadressen mit einem Lesetelegramm vom Enertex® EibPC abfragen. Dazu kann für jede importierte Gruppenadresse die Initialisierung unter **Objekte → Gruppenadressen** aktiviert werden.

Wichtige Erklärungen

- 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 1,5 s.
- Nach dem Abfragen der letzten Gruppenadresse wird mit der Verarbeitung begonnen.
- Alle Anweisungen die als Bestandteil einer der zu initialisierenden Gruppenadressen beinhalten, werden ungültig und die Verarbeitung im ersten Durchlauf ausgeführt.
- Wenn eine Gruppenadresse nicht rechtzeitig auf die Leseanforderung antwortet, wird ein Event generiert.

Validierungsschema

Objektbaum

Dieses Kapitel beschreibt das Validierungsschema. Bei der Kompilierung des Projekts wird ein Programm generiert, das von der EibPC-Firmware verarbeitet wird.

Dies ist jedoch kein sequentielles Programm, wie man es für Mikroprozessoren kennt, sondern eine Datenabhängigkeitsbeschreibung, ein sogenannter Abhängigkeitsbaum. Die Knoten in diesem Graph werden als Programmobjekte bezeichnet. Diese sind nicht zu verwechseln mit den Objekten. Die Menge der Programmobjekte umfasst sowohl alle verwendeten Objekte, es werden aber auch alle Befehle und Funktionen (Seite 53) auf Programmobjekte abgebildet.

Statt einen Befehl nach dem anderen auszuführen wird die Ausführung in Zeitschritte (Zyklen) aufgeteilt. Innerhalb jedes Zyklus erfolgt die Abarbeitung quasi-parallel, .d.h. jede Änderung ist gleichberechtigt. Um unnötige Arbeit zu vermeiden, werden stets nur diejenigen Programmobjekte betrachtet, die sich geändert haben.

Jedes Programmobjekt 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 Programmobjekt 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.

Die Verarbeitung erfolgt in folgenden Schritten in jedem Zyklus, solange bis kein Objekt mehr invalid ist.

Invalidierung

Wenn ein Verarbeitungsobjekt invalid ist, heißt das zunächst nur, dass es neu berechnet werden muss. Bei Initialisierung (erster Verarbeitungszyklus) ist jedes Programmobjekt invalid. Danach muss ein Ereignis eingetreten sein. Trifft z.B. ein Telegramm vom Bus ein, wir das entsprechende Programmobjekt invalidiert. Invalidiert werden können nur Programmobjekte, die von einer Gruppenadresse, dem Systemtimer, TCP/UDP, RS232 oder einer if-Anweisung abhängig sind.

Berechnung

Das Verarbeitungsobjekt wird mit den neuen Eingabewerten neu berechnet. Hat sich der Zustand geändert, wird der nächste Schritt ausgeführt.

Bedingte Invalidierung

Invalidiere alle Verarbeitungsobjekte in der Abhängigkeitsliste.

Dabei ist das genaue Verhalten abhängig vom Typ des Verarbeitungsobjekts.

Variablen

Die folgenden Programmbeispiele können so direkt als Expertenprogramm angelegt werden.

Es wird das folgende Programm betrachtet:

```
x=2
y="SaunaDimmer-1/0/1"+3%+x
z='1/2/3'b01 or '1/2/4'b01
```

Durch die Kompilierung wird der in Abbildung 13 dargestellte Abhängigkeitsbaum erstellt.

Nach der Initialisierung hat x den Wert 2 und y den 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 für y: Wenn vom Bus die Gruppenadresse eintrifft und sich deren Wert verändert hat, so wird y neu berechnet. Y ist also von einem Ausdruck abhängig, der sich geändert hat und wird invalidiert. Dabei wird y im 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.

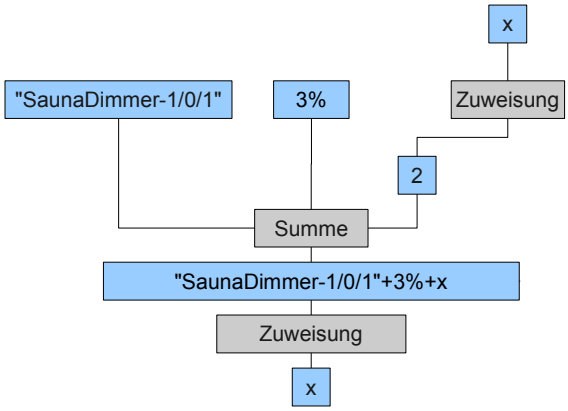


Abbildung 13: Baumstruktur für y="SaunaDimmer-1/0/1"+3%+x und x=2

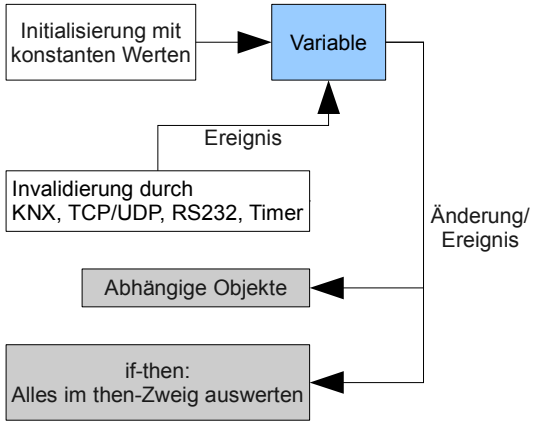


Abbildung 14: Verarbeitung von Variablen

Funktionen

Funktionen werden durch ihre Argumente invalidiert. Ändert sich also ein Argument, wird der Wert des Funktionsobjektes neu berechnet. Ändert sich dadurch das Ergebis, werden alle abhängigen Programmobjekte invalidiert.

```
x=sin(3.14f32)
tan(2.0f32)
y=cos("Temperatur-1/0/1")
z=event("Temperatur-1/0/1")
```

Ausgabefunktionen

Eine Besonderheit stellen Ausgabefunktionen dar. Diese haben bei ihrer Ausführung Seiteneffekte, neben der Änderung ihres Zustandes. Um zu gewährleisten, dass diese nicht ungewollt angestoßen werden, werden Ausgabefunktionen nie durch ihre Argumente invalidiert, sondern nur, wenn sie innerhalb von if-then-Anweisungen vorkommen und somit durch die Bedingung invalidiert werden.

```
write("Temperatur-1/2/1",22.3)
write("Schalter-1/2/10",!"Schalter-1/2/10")
read("Temperatur-1/2/1")
```

Das Programm führt nie etwas aus. Die zweite Anweisung würde sonst im Zyklustakt des Enertex® EibPC den Bus beschreiben.

Timerfunktionen

Ähnliches gilt für Timerfunktionen. Diese werden nicht von ihren Argumenten invalidiert, sondern durch die interne Uhr des EibPCs.

```
o=stime(19)
```

O wird nur minütig bei 19 Sekunden nach der letzten vollen Minute für einen Verarbeitungszyklus auf EIN gehen.

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.

```
a=1
if '1/2/3'b01 then a=3 endif
```

Nachdem ein Gruppentelegramm für die Adresse '1/2/3' mit dem Wert 1b01 empfangen wurde, bleibt a auf 3, da sich 1 (von der Zuweisung a=1) nie ändert und dadurch a nicht mehr invalidiert wird.

Verschachteltung von if-Anweisungen

Innere if-Anweisungen werden nicht durch ihre Bedingung sondern durch die Bedingung der äußeren if-Anweisung invalidiert. So wird gewährleistet, dass die äußere Bedingung beachtet wird. Dadurch ist die Auswertung des inneren then-Zweigs allerdings nicht mehr vom Ändern der inneren Bedingung abhängig.

```
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
```

Nich ausprobieren!

Das Beispiel verdeutlicht die abweichende Invalidierung bei geschachtelten if-Anweisungen:

```
if change('0/0/1'b01) then {
   if EIN then write('0/0/1'b01, !'0/0/1'b01) endif
} endif
```

Wäre die innere Anweisung nicht im then-Zweig einer anderen, so würde nie auf den Bus geschrieben werden, da die Bedingung sich nicht ändert. Durch die Verschachtelung wird die innere if-Anweisung mit jeder Änderung der Gruppenadresse invalidiert. Da die innere Bedingung erfüllt ist, wird ein Schreibtelegramm gesendet, dass den Wert der Gruppenadresse invertiert. Dadurch wird die Bedingung der äußeren if-Anweisung neu ausgewertet, die wiederrum die Auswertung der inneren bedingt. Es kommt also zu einer endlosen Telegrammauslösung.

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.

```
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.

else-Zweig

Der else-Zweig einer if-Funktion ist wie eine zweite eigenständige if-Funktion mit negierter Abfragebedingung zu verstehen.

```
Taste='1/2/3'b01
if Taste then write('4/5/6'b01, AUS) else write('4/5/6'b01, EIN) endif
```

Dies ist gleichwertig zu

```
Taste='1/2/3'b01
if Taste then write('4/5/6'b01, AUS) endif
if !Taste then write('4/5/6'b01, EIN) endif
```

Schreiben auf Warteschlangen

Wenn die Verarbeitung eines „Programmdurchlaufs“ abgeschlossen sind, werden die internen Warteschlangen für die Ausgabe beschrieben. Dabei werden die zu diesem Zeitpunkt aktuellen Zustände der Argumente ausgewertet, d.h. ein Objekt kann bereits durch eine nachfolgende Operation verändert worden sein. Dies betrifft die folgenden Funktionen:

- sendudp
- sendudparray
- resolve
- sendmail
- sendhtmlmail
- sendcp
- sendtcparray
- connecttcp
- closetcp
- resetasyncserial
- sendasyncserial
- startvpn
- stopvpn
- openvpnuser
- closevpnuser
- ping
- writeflash
- writeflashvar

Netzwerkfunktionen

Beispiele:

```
uPing=10
uIp=192.168.1.1
if after(systemstart(),1000u64) then {
   uPing=ping(uIp);
   uIp=192.168.1.100;
} endif
```

Die Variable uIP wird mit der Adresse 192.168.1.1 initialisiert. Eine Sekunde nach dem Start wird die If-Bedingung invalidiert, und damit die Anweisungen im then-Zweig. Die ping-Anweisung wird in die Warteschlange eingereiht, die Zuweisung uIp=192.168.1.100 hingegen während des Zyklusses ausgeführt. Am Ende des Zyklus wird nun die ping-Anweisung ausgeführt, jedoch mit der bereits aktualisierten Adresse.

```
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.*

Schreiben auf Flash-Speicher

Gleichermaßen gilt diese Aussage für die Warteschlange für das Schreiben in den Flash.

```
b=1
if systemstart() then {
  writeflash(b,0u16);
  writeflashvar(b);
  b=b+1;
} endif
```

Hier wird b zuerst um eins erhöht, und dann der Wert in das Array bzw. Flash geschrieben.

```
b=5
if systemstart() then {
  writeflash(b,0u16);
  b=b+1;
  readflash(b,0u16);
} endif
```

Das Lesen erfolgt sofort. Dadurch wird der um 1 erhöhte Wert von b durch den gespeicherten Wert ersetzt und schlussendlich, am Ende des Zyklus, in den Flash geschrieben.

Asynchroner Rückgabewert

Bei einigen Funktionsaufrufen (wie etwa das Anlegen einer TCP-Verbindung) kann nicht gewährleistet werden, dass diese Ihren Rückgabewert innerhalb einer Verarbeitungsschleife des EibPC aktualisieren. Diese Funktionen aktualisieren ihren Rückgabewert entsprechend ihrer eigenen Verarbeitung „asynchron“ zur Hauptverarbeitungsschleife.

Dazu folgendes Beispiel:

```
// 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-Anweisung 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). Alle vom Rückgabewert abhängigen Programmobjekte werden daraufhin im nächsten Zyklus neu ausgewertet.

Makros

Makros sind wie eine Art Textersetzung zu verstehen.

Man betrachte das folgende Makro:

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 nach der :return Anweisung eine Verknüpfung mit dem Validierungskonzept aufweisen. Daher wird folgender Code

```
if sun()  then MyFunction($Licht$) endif
```

bei Sonnenaufgang nichts ausgeben. Er wird nämlich von Eibparser „expandiert“ in die Form:

```
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
```

Alle write's sind global!

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“.

```
: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

„Umleiten der Valdierung“

```
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.

Weiteres Beispiel:

```
:begin Akt_3(Aktor,Now)
   Variable=3
    if Now then write(Aktor,Variable) endif
:return AUS
:endif
```

Wenn das Makro wie folgt genutzt wird

```
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.

Nur globale Definitionen

: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:

```
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

```
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:

```
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

Das Programm

```
a=AUS
if a==EIN then a=!a else a=!a endif
```

verursacht eine Rekursion, wie man anhand Abbildung 15 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.

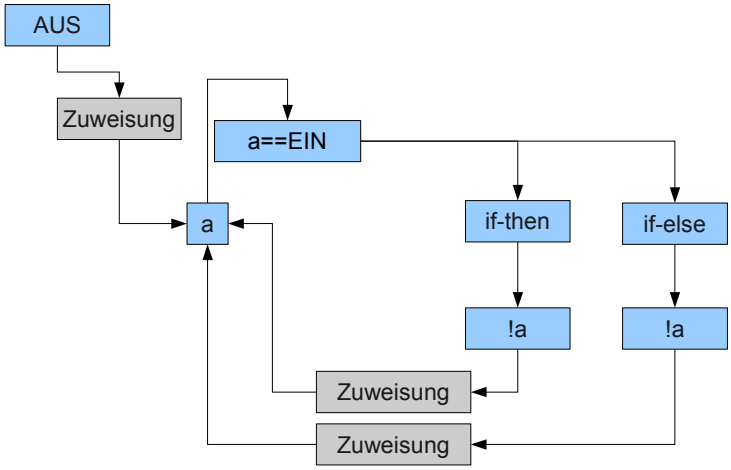


Abbildung 15: Baumstruktur für a=AUS; if a==EIN then a=!a else a=!a endif

Insgesamt gewährleistet das Validierungskonzept, dass

- der Enertex® EibPC auch bei umfangreichen Programmen extrem effizient ist
- Standardaufgaben (Wenn Taste, dann Licht) einfach programmierbar sind
- alle Anweisungen parallel in einem Zyklus abgearbeitet werden.

Befehle und Funktionen

Dieses Kapitel ist nur relevant, falls Sie eigene Expertenprogramme erstellen möchten.

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:

Oder - Verknüpfungen

```
if ((TasterEin==1u08) and (Windgeschwindigkeit==2.9f16)) then LichtAktorEin=1u08 endif
```

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 40 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 40 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*

Zeitfunktionen

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. 60

Minute

Definition

- Funktion minute()

Argumente

- keines

Wirkung

- Die Systemzeit (Minute) wird in eine Variable gespeichert

Datentyp Ergebnis (Rückgabe)

- Datentyp u08

Beispiel:

Stoppuhr siehe S. 60

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. 106.

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. 36).

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. 36).

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. 36).

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

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

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=east
ermonth(-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. 36 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 Konstante mit dem Binärwert 1b01 und kann deshalb als Vergleichsoperator an Stelle von 1b01 angegeben werden.

Azimuth

Definition

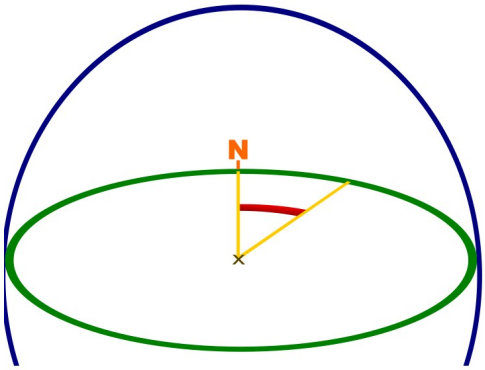
- Funktion azimuth()

Argumente

- keine. Der Enertex® EibPC muss die geographische Länge und Breite des betreffenden Ortes kennen. Diese können im Enertex® EibStudio eingegeben werden (s. Seite 36).

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

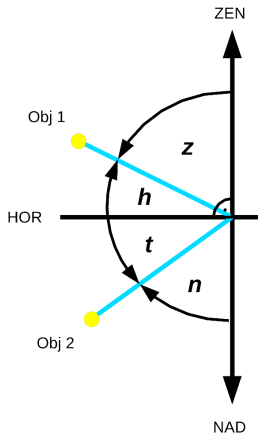
- Funktion elevation()

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 36).

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 b01

```
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

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.

Wochenzeitschaltuhr

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.)

Tageszeitschaltuhr

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.

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,7=Werktags, 8=Wochenende)

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

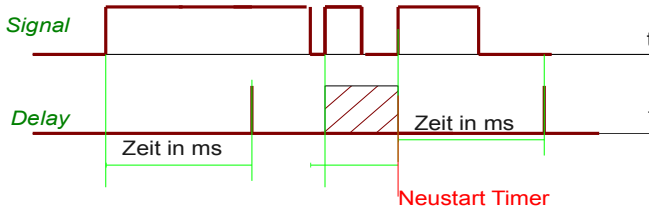
- Funktion delay(*Signal*, *Zeit*)

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. 74).
- 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

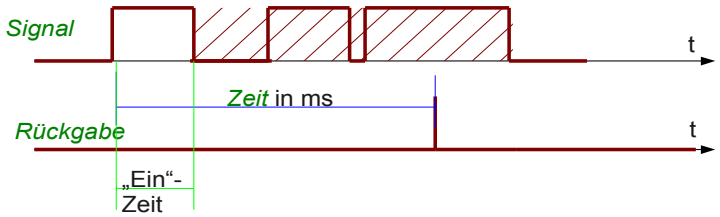
- Funktion after(*Signal*, *Zeit*)

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
```

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. 75.
- 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 40).

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 40). 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. 81).**

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

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 40) 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. 81).**

Datentyp Ergebnis (Rückgabe)

- Datentyp b01

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 40). 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

Writeflashvar

Definition

- Funktion writeflashvar(*Variable)*

Argumente

- *Variable* beliebiger Datentyp

Wirkung

- Es werden im eingebauten Flash die Binärdaten des Speicherinhalts (vgl. Bitlänge Seite 40) 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

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 95)

```
ValuePowerK1="K1-Wirkenergiezähler (Verbrauch)-14/2/76"
if htime(0,0,0) or SHUTDOWN then {
        writeflashvar(ValuePowerK1)
} endif
if systemstart() then readflashvar(ValuePowerK1) endif
```

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 40 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 40 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 90)

Datentyp Ergebnis (Rückgabe)

- Datentyp wie Argumente

Hinweis:

Mit gleicher Syntax können Sie Zeichenketten verketten (siehe Seite 102).

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 40 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 40 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:Basis
- *Variable2*:Numerus

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 40 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 40 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 40 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 40 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.
- Aktualisierung der Abhängigkeiten bei Wertänderung. Falls eine negative Basis vorgegeben wird, erfolgt keine Berechnung.

Datentyp Ergebnis (Rückgabe)

- Datentyp f32

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 40 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 114) 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

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.
- 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.
- Wird von einer Zeichenkette in einen numerischen Datentyp konvertiert, wird der Zahlenwert interpretiert. Beginnt die Zeichenkette mit 0x oder 0X, wird die Zahl hexadezimal interpretiert.
- 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. 92

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 $,LEFTGRAF, 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(34f32)+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)+tan(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. 81.

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 40 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

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 **Projekteinstellungen → Dateien** 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. 100)

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. 100 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 1 bis 64)

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.

Storescene - Szene speichern

- *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.

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.

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“

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
```

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. 103

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. 38) 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 40 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. 60).

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. 60) 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)
```

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)
```

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)
```

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.

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-Telegramme

write

Gruppentelegramm senden

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

read

Leseanforderung senden

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

Hinweis:

Damit der Aktor im KNX™ Netzwerk auch antwortet, muss das Lese-Flag bei dessen ETS Programmierung auch gesetzt werden.

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 16 dargestellt werden.

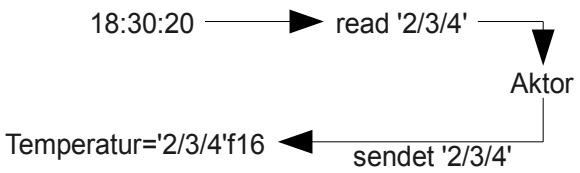


Abbildung 16: Funktionsweise von read

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 Tempera-

tur.

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 auf das Eintreffen von Paketen.

Eine Event-Funktion ist wie folgt definiert:

Definition

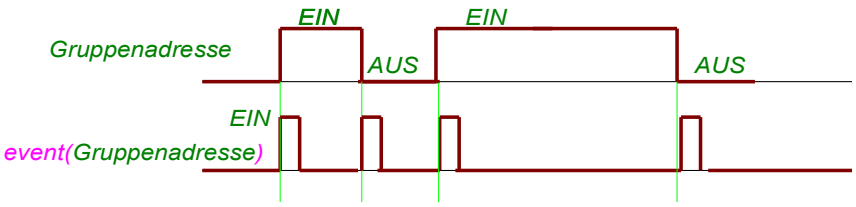
- Funktion event(*Gruppenadresse*)

Argumente

- *Gruppenadresse*: Importierte oder manuelle KNX Gruppenadresse
- Bei UDP, TPC/IP oder RS232 Telegrammen kann die event-Funktion 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.

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
```

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. 96) genutzt werden

Datentyp Ergebnis (Rückgabe)

- keine

Beispiel

Folgendes Beispiel zeigt die Verwendung alternativ zu obiger [InitGA] Sektion

```
// 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.

```
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.

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 17: 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 18: 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 19: 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
```

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. 97

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. 125).

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 S. 46).

Netzwerkfunktionen

Standard-Ports

Die Ports, über welche der Enertex® EibPC kommuniziert, können über die **Projekteinstellungen → Verbindung** geändert werden.

UDP Telegramme

UDP Ports

Der Enertex® EibPC selbst verschickt die Daten beim UDP Transfer standardmäßig 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.

Sendudp

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.

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. 125 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. Fehler: Referenz nicht gefunden)

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

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.Fehler: Referenz nicht gefunden)

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. Fehler: Referenz nicht gefunden)

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 36).

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. Fehler: Referenz nicht gefunden)

Hinweis:

Die Funktion sendmail sendet reine Textmails. Für das Verschicken von html-Mails steht sendhtmlmail 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 36).

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

Startvpn

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

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

Getvpnusers

Definition

- Funktion getvpnusers()

Argumente

- keine

Wirkung

- Liefert eine Liste mit derzeit aktiven VPN-Benutzern

Datentyp Ergebnis (Rückgabe)

- keine

Hinweis: In der EnertexVPN.lib finden sich Funktionen, die das Handling von VPN mit dem Enertex® EibPC einfach gestaltet.

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. 137 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. Nicht zu verwechseln ist dies mit der zyklischen Auslagerung der KNX Telegramme (vgl. S 36).

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 155 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.

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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 4 (Seite 181).
- 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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 147.

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 155 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.

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 155 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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zu-dem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 4 (Seite 181).
- 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 155 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 147.

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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 4 (Seite 181).

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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 4 (Seite 181).

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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 4 (Seite 181).

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 180 dargestellt und werden über vordefinierte Zahlen (u08) ausgewählt. Zudem erleichtern vordefinierte Konstanten die Auswahl. Die Zuordnung hierfür ist in Tabelle 3 (Seite 181) 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 4 (Seite 181).

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 Datentyp muss u16 sein.
- 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 151 und timebufferread 151.

Datentyp Ergebnis (Rückgabe)

- Werte: 0 Erfolg, 1 Fehler: maximale Anzahl von Zeitpuffern überschritten, 2 Fehler: Zeitpuffer bereits definiert

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

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

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

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

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.

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,ChartIdx,ChartBuffer,StartPos,EndPos)

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.
- StartPos Startposition der Anzeige
- EndPos Endposition der Anzeige

Wirkung

- Den darzustellenden Bereich eines Zeitpuffers für das Webelement vorgeben.

Datentyp Ergebnis (Rückgabe)

- keine

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

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.

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

```
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

Dieses Kapitel ist nur relevant, falls Sie eigene Seiten innerhalb eines Expertenprogramms definieren möchten.

Alternativ können Sie auf einzelne Visualisierungselemente innerhalb von Expertenprogrammen zugreifen, indem Sie ihnen eine ID-Variable zuweisen (siehe S.

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 20). Es gibt dabei grundsätzlich die Möglichkeit, auch Felder freizulassen sowie Trennlinien einzufügen.

Das Design der Knöpfe ist vorgegeben

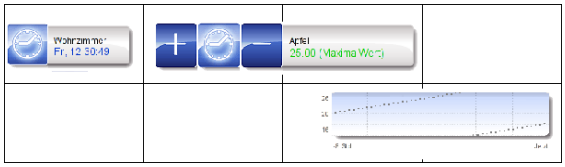


Abbildung 20: Schema Webserver

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 180) 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. 166), das z.B. für die Verwendung für Wandpanels oder Smartphones sehr geeignet ist. Auf S. 162 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 21) können Sie die nach Gruppen geordneten Seiten auswählen.

Abbildung 21: 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 21) 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

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.

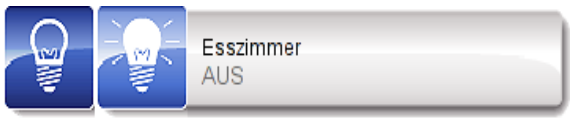
Beispiel

```
[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] $Seite 5$
```

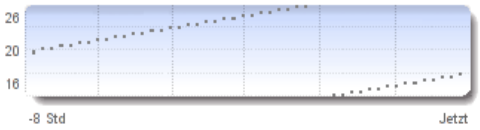
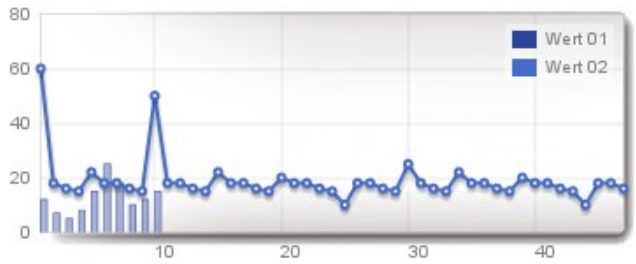
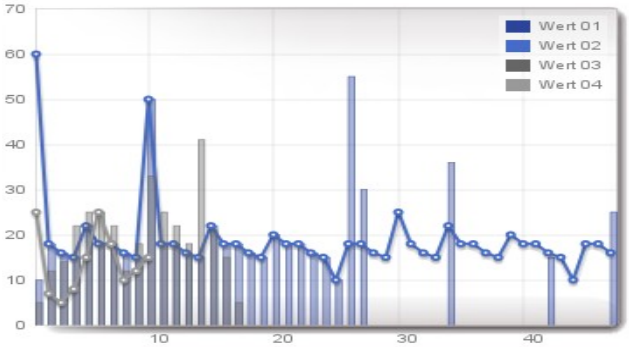
Webserverelemente

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 2). Neben den gezeigten Standard-Blau kann mit Hilfe des Designkommandos auf ein schwarzes Design umgeschaltet werden (vgl. 166).

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. 141) bzw. mbbutton (S. 144) 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. 141) bzw. mpbutton (S. 144) 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. 141) bzw. mpbutton (S. 144) 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. 141) bzw. mpbutton (S. 144) 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. 141) bzw. mpbutton (S. 144) 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. 141) bzw. mpbutton (S. 144) 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. 141) bzw. mpbutton (S. 144) 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 143) 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 2: Übersicht über Webelemente.

Konfiguration

Das Design des Webservers ist fest im Enertex® EibPC eingebaut. Das Schema nach Seite 155, Abbildung 20 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 20). 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 20 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 22.

generiert einen Freiraum unter dem 4-Fach Multi-Knopf

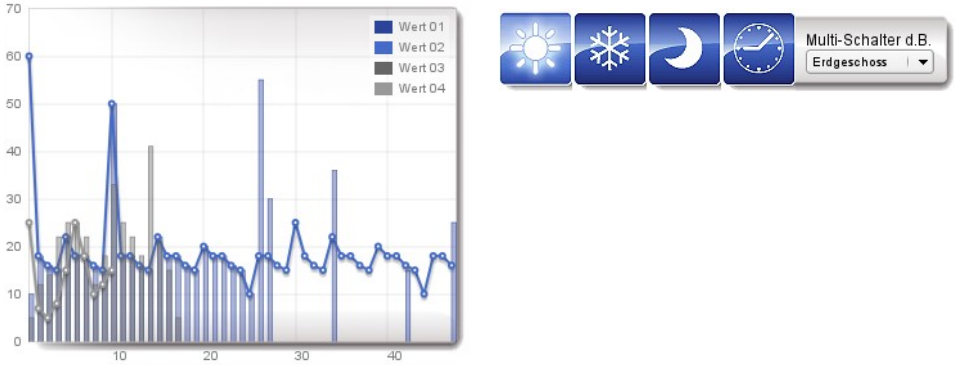


Abbildung 22: Freiraum

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 23 zeigt diese „Belegung“ der Einheitsgrößen (in Blau dargestellt) der obigen Webkonfiguration.

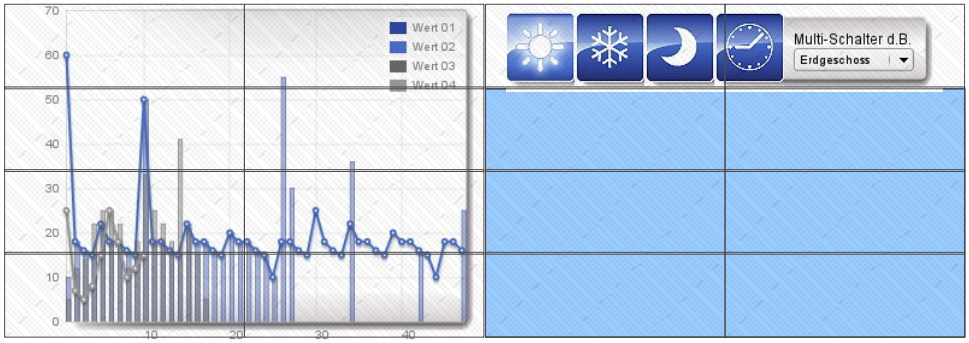


Abbildung 23: Veranschaulichung der Einheitsgrößen.

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 23 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 23,

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 24).

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 23 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 24, welche nun der Webserver ausgibt.

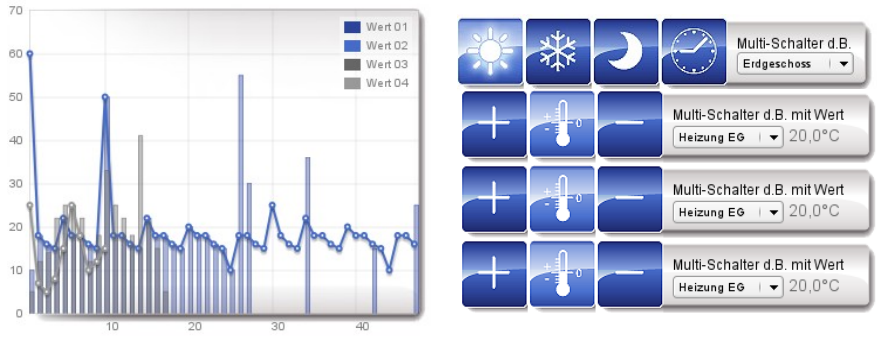


Abbildung 24: Compactmodus

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 25 zeigt hierzu nochmals schematisch die Anordnung der Elemente, wie das auch schon im eibparser ausgegeben wird.

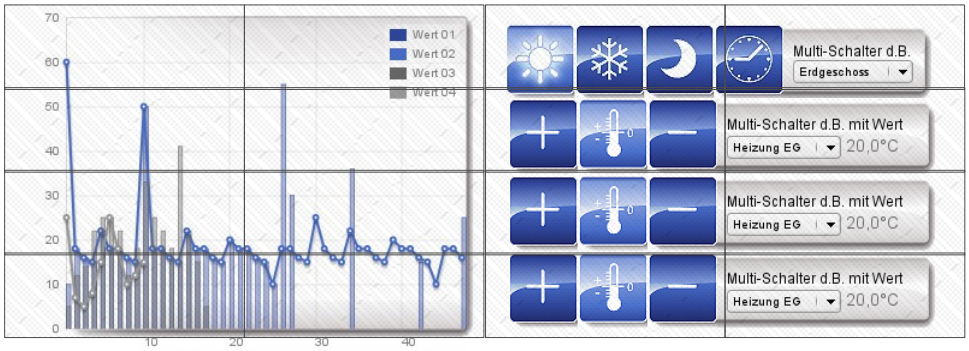


Abbildung 25: „compact“ mit Raster (zur Veranschaulichung)

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 26 ausgeben.

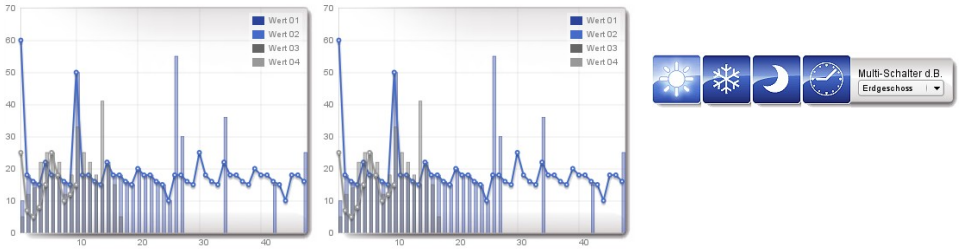


Abbildung 26: Darstellungsbeispiel für den Zeilenvorschub

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 180).
- Text: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion Webdisplay (Seite 140) 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 139) 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. 36). Die Hintergrundgrafik wird nicht skaliert. Die Anordnung der Webelemente bleibt davon unberücksichtigt, none-Elemente werden transparent.
- Der Pfad auf den internen Server ist „/upload/“

Abbildung 27: 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 180).
- 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 140) 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 180).
- Text: Ein statischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Die Grafik und der Text werden über die Funktion pdisplay (Seite 145) angesprochen.
- Es handelt sich um ein Element, das immer nur einer Seite zugeordnet ist.
- Die Betätigung der Knöpfe ist über die Funktion pbutton (Seite 146) 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 180).
- 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 145 (Seite 145) 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 144) 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 180).
- 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. 145) angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion button (Seite 139) 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 180).
- 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. 145) angesprochen.
- Die Betätigung der Knöpfe ist über die Funktion pbutton (Seite 146) 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 180).
- 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. 145) 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 139) 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 180).
- 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. 145) 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 139) 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 139) 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 143) 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 143.

Mtimechart

Element *mtimechart*

- *mtimechart*(ID)[Format,Typ,Länge,YLMIN,YLMAX,YRMIN,YRMAX] ($Beschriftung1$,ChartPos1,Buffer1,$Beschriftung2$,ChartPos2,Buffer2, $Beschriftung3$,ChartPos3,Buffer3,$Beschriftung4$,ChartPos4,Buffer4)
- $Beschriftung1$,CHARTPOS1,verkn.Buffer1, $Beschriftung2$,... (bis zu 4 Graphen)

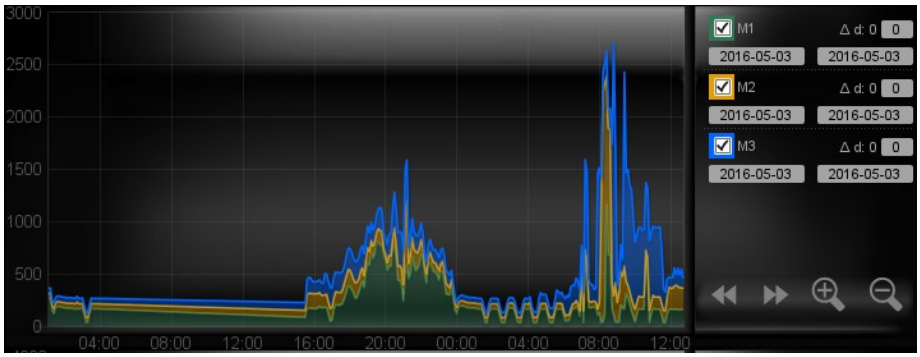
Argumente

- *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 (0=AUTOSCALELEFT)

  1 für autoscale der rechten Achse, in diesem Fall wird YRMAX etc. ignoriert (1=AUTOSCALERIGHT)

  2 für autoscale der beiden Achsen (2=AUTOSCALE)

  3 für kein autoscale (3=NOAUTOSCALE)
- *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 (RIGHTGRAF) (0 für Beschriftung an der linken y-Achse, 1 für Beschriftung an der rechten y-Achse) oder 2 (STACK) für grafisches Addieren von zwei Graphen: 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. 149) 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. 149 und timebufferconf S. 149 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. 149 und timebufferconf S. 149.
- mtimecharts sind immer global.

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 143) 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 143.

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 139) 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 180).
- *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 140) angesprochen.
- Die Betätigung des Sliders ist über die Funktion getslider (Seite 141) auszuwerten.
- Die Verstellung des Sliders ist über die Funktion setslider (Seite 148) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion button (Seite 139) 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 180).
- *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 140) angesprochen.
- Die Betätigung des pSliders ist über die Funktion getpslider (Seite 141) auszuwerten.
- Die Verstellung des pSliders ist über die Funktion setpslider (Seite 148) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion pbutton (Seite 146) 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 180).
- *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 140) angesprochen.
- Die Betätigung des esliders ist über die Funktion geteslider (Seite 141) auszuwerten.
- Die Verstellung des esliders ist über die Funktion seteslider (Seite 148) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion button (Seite 139) 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 180).
- *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 140) angesprochen.
- Die Betätigung des Sliders ist über die Funktion getpeslider (Seite 141) auszuwerten.
- Die Verstellung des Sliders ist über die Funktion setpeslider (Seite 149) anzustoßen.
- Die Betätigung des Knopfes ist über die Funktion pbutton (Seite 146) 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 180).
- *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. 154) 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. 154) 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. 154) erfolgt als String als 24-Bit Zahl.

Zugriff im Anwenderprogramm

- Das Element wird über die Funktion webinput (s. 154) 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.
- *Stil*:Wert 0,1,2 (bzw. Konstante ICON und NOICON, NOCOLOR)

Zugriff im Anwenderprogramm

- Das Element wird über die Funktion weboutput (s. 154) angesprochen.
- Weboutputelemente sind immer global.

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 21) 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 180).
- 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. 147) 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 180).
- Text: Ein dynamischer Beschriftungstext (erste Zeile).

Zugriff im Anwenderprogramm

- Mit Hilfe der Seiten bezogenen Funktion link (S. 142) 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

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 183), 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 3: 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 4: Übersicht über Zustände.

GREY	0u08
GREEN	1u08
BLINKRED	2u08
BLINKBLUE	3u08

Tabelle 5: 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.

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										

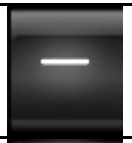
WASTE	52u08										
ASK	53u08										
WARN	54u08										
NEAR	55u08										
CAMERA	56u08										
SIGNAL	57u08										
DOOR	58u08										
GARAGE	59u08										
CURTAIN	60u08										

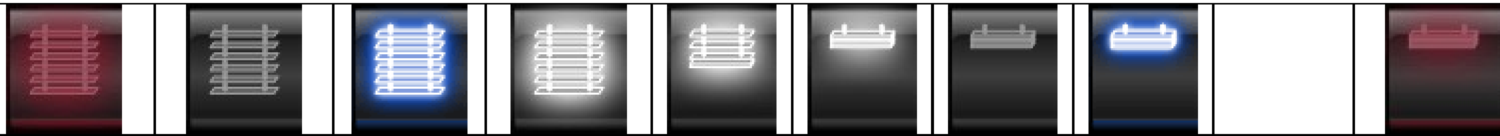
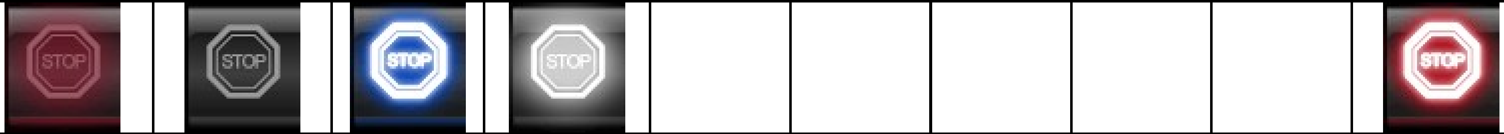
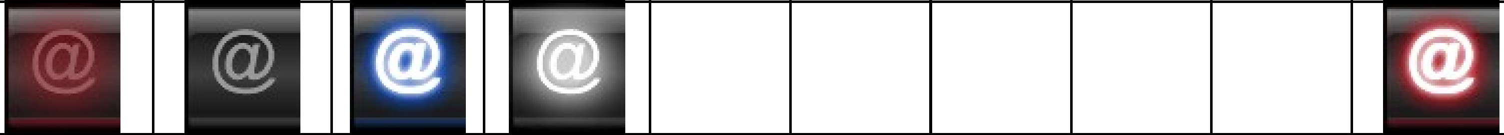
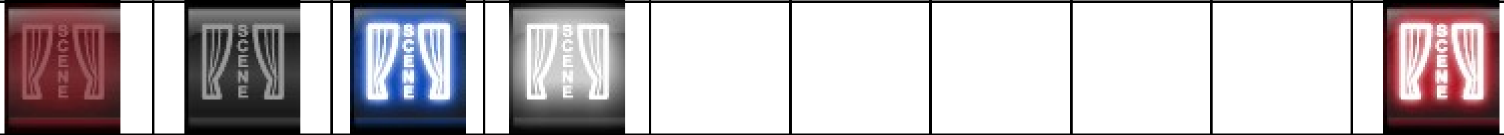
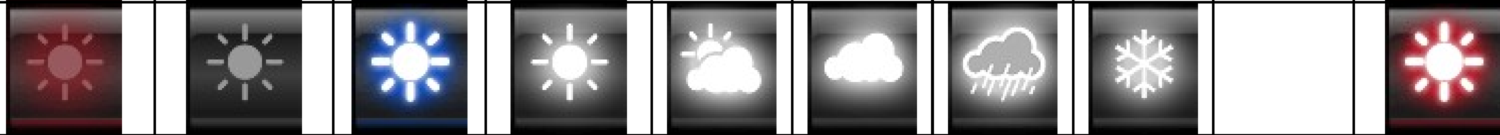
ANGLE	61u08										
ROLLER	62u08										
EMAIL	63u08										
PETS	64u08										
PHONE	65u08										
PERSON	66u08										
TV	67u08										
BEAMER	68u08										
RADIO	69u08										

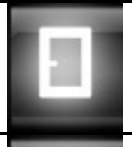
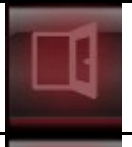
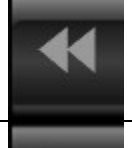
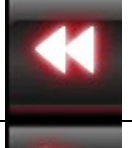
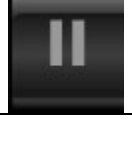
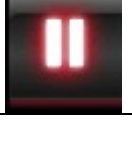
RECIEVER	70u08										
MEDIA	71u08										
STOVE	72u08										
FRIDGE	73u08										
WASHER	74u08										
DISHWASHER	75u08										
HOLIDAY	76u08										
SLEEP	77u08										

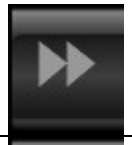
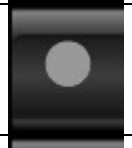
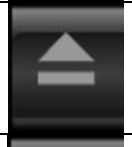
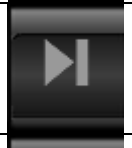
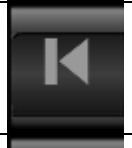
UPDATE	78u08										

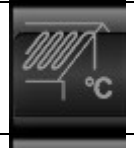
Tabelle 6: Übersicht Symbole blue-design

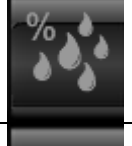
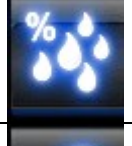
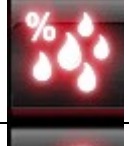
Symbol	Index	DARKRED 0u08	INACTIVE 1u08	ACTIVE 2u08	DISPLAY 3u08	STATE4 4u08	STATE5 5u08	STATE6 6u08	STATE7 7u08	STATE8 8u08	BRIGHT-RED 9u08
INFO	0u08										
SWITCH	1u08										
UP	2u08										
DOWN	3u08										
PLUS	4u08										
MINUS	5u08										
LIGHT	6u08										
TEMPERATURE	7u08										

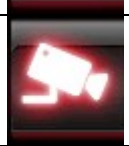
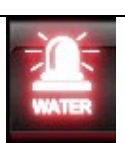
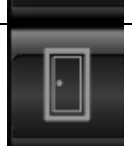
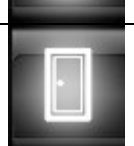
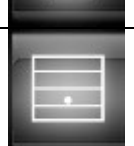
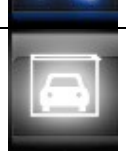
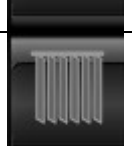
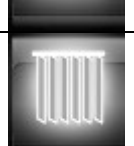
BLIND	8u08	
STOP	9u08	
MAIL	10u08	
SCENES	11u08	
MONITOR	12u08	
WEATHER	13u08	
ICE	14u08	
NIGHT	15u08	

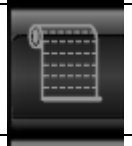
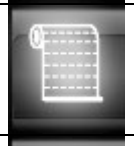
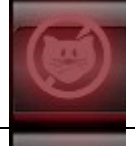
CLOCK	16u08											
WIND	17u08											
WINDOW	18u08											
DATE	19u08											
PRESENT	20u08											
ABSENT	21u08											
REWIND	22u08											
PLAY	23u08											
PAUSE	24u08											

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										

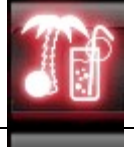
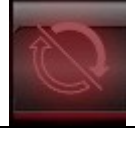
RECIEVER	70u08										
MEDIA	71u08										
STOVE	72u08										
FRIDGE	73u08										
WASHER	74u08										
DISHWASHER	75u08										
HOLIDAY	76u08										
SLEEP	77u08										
UPDATE	78u08										

Tabelle 7: 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 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

Programmierung eines Makros

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.

Grundsätzliches

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:

Definition

```
[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:

```
: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
```

Ein Makro beginnt mit :begin

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.

... endet mit :end

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
```

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 = Basis Var2 = Numerus
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 geogra - phische 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 KNX 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 KNX 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 65535u16 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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.
mtimechart(ID) [Format,Typ,Länge,YLMAX,YLMIN,YRMAX,YRMIN ($Beschriftung1$,ChartPos1,Buffer1, [$Beschriftung2$,ChartPos2,Buffer2, $Beschriftung3$,ChartPos3,Buffer3,$Beschriftung4$,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 YRMAX 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 180). 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 QUAD)
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(TimeBufferID, 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 180). 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 180). 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 KNX 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 = Basis Var2 = Numerus
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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 180). 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 KNX 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)

Quellen

Enertex® EibPC und Enertex® EibPC² verwenden Software, die unter verschiedenen Lizenzen kostenlos zur Verfügung gestellt wird. Die entsprechenden Softwarequellen werden auf Anfrage jederzeit gegen eine Bearbeitungsgebühr von uns zur Verfügung gestellt.

Enertex® EibPC

Betriebssystem**:** Debian Linux: 2.6.24-rt1.

Enertex® EibPC²

Betriebssystem: Debian Linux 9.6: 4.14.16.

Enertex® EibStudio 4

Eine Liste der verwendeten Bibliotheken kann unter **Hilfe → Lizenzen** eingesehen werden.

Fragen und Antworten

Ereignisse

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 32 (> Firmware 4.000, EibStudio 4.000)

- Grundlegende Überarbeitung für Enertex® EibPC² und Enertex® EibStudio 4