CL-Software

Copyright © 2024 CL-control

Table of Contents

1. CL-Studio Software

1.1. Titel

CL-Software 5.0

Benutzerhandbuch
2024

1.2. Cover

Versionen

CL Studio

CLX

Version 5.0

copyright (c)

CL-control 2024

www.cl-control.de

1.3. Allgemeine Hinweise

Allgemeine Hinweise

In diesem Dokument sind die Programmversionen CL-Studio und CLX beschrieben.
Da es Unterschiede zwischen den Versionen gibt, kann es an einigen Stellen sein, dass Funktionen beschrieben sind, die in der benutzten Software-Version nicht zur Verfügung stehen.
Die Software läuft auf einer Linux-Zentrale (normalerweise HomeMatic-CCU) und wird über einen Browser konfiguriert und programmiert. Dabei können auch umfangreiche Visualisierungen für beliebige Browser erstellt werden.

Der generelle Unterschied zwischen den Versionen ist, dass mit der CL-Studio-Version nur eine HomeMatic-CCU eingesetzt werden kann, die gleichzeitig auch die Zentrale ist.

Mit der CLX-Version können mehrere CCUs als Gateways benutzt werden können, was die Benutzung einer erheblich grösseren Anzahl von Geräten auch in grossen Gebäuden ermöglicht.

Aufgrund der Unterschiede der Versionen kann es an einigen Stellen der Hilfetexte vorkommen, dass Funktionen beschrieben sind, die für die verwendete Version der Software nicht zur Verfügung stehen.

Die Konfiguration und Programmierung wird über den Webserver der Zentrale mit einem beliebigem Browser erstellt. Konfiguration und Programmierung sollten über ein Gerät mit grösserem Display, also einem PC oder Tablett durchgeführt werden, da die Displaygrösse eines Smartphones bei der Texteingabe insbesondere im Querformat an vielen Stellen nicht ausreicht. Bei einem kleinen Display sollte für die Texteingabe immer das Hochformat benutzt werden, sonst kann es je nach Displaygrösse sein, dass eine Eingabe nicht möglich ist.

Die CL-Software bietet umfangreiche Möglichkeiten eigene individuelle Anwendungsfunktionen für eine intelligente Gebäudeautomation zu erstellen.Grundlage ist dabei eine einfach anzuwendende, leistungsfähige Programmierung in deutscher Sprache. Diese Form der Programmierung ist nicht nur einfacher, sondern bietet erheblich mehr Möglichkeiten und Features als die Programmierung über die integrierte WEB-UI einer HomeMatic-CCU. Das System erlaubt zudem umfangreiche individuelle Visualisierungen für Browser beliebiger Geräte.

Bei Benutzung des HomeMatic-Systems wird zum Anlernen und zur Konfiguration von Modulen immer die WEB-UI der CCU benutzt.

Falls eine CCU benutzt wird, können die internen Programmierungsoptionen der CCU unabhängig von der CL-Software parallel benutzt werden. Dabei muss aber beachtet werden, dass es dann schwierig werden kann den Überblick über die programmierten Funktionen zu behalten.

Einige Hinweise und Erläuterung:

Die CL-Software bietet folgende leistungsfähige Features:
- virtuelle Objekte beliebigen Typs, unabhängig von verwendeten Geräten, z.B. zur Ausgabe von Texten in Visualisierungsansichten
- Rechenfunktionen mit Zahlen, Zeiten
- Variablen zur Zwischenspeicherung und Berechnung von Werten
- Möglichkeit e-mails zu versenden
- integrierte Anwesenheitssimulation
- Funktionen zur Bearbeitung von Texten
- Schreiben eigener Textdateien z.B für Logs
- Historydatenbank
- umfangreiche Visualisierungen über Browser
- Aufruf von Webseiten (http oder https-Requests) um Inhalte von Internetseiten auszulesen oder Geräte anzusprechen
- integrierte Funktion zur Heizungssteuerung über Schaltaktoren z.B. für Fussbodenheizungen, Infrarotheizungen und anderen elektrischen Heizgeräten

In den Hilfeseiten gibt es ausführliche Anleitungen und Beschreibungen zur Programmierung.

Die Ausführung bei Benutzung der CL-Software ist erheblich schneller als Programmierungen über die WEB-UI einer HomeMatic-CCU, da die Programmierung in einen optimierten Ausführungscode übersetzt wird, der von der Exec-Engine auf der Zentrale verarbeitet wird.
Ein weiterer Vorteil der CL-Software ist, dass beim HomeMatic-System eine Erweiterung auf mehrere CCUs möglich ist wenn die Kapazitätsgrenze einer CCU bezüglich der Anzahl der angelernten Geräte oder der Funkreichweite erreicht ist. Neue Geräte werden dann einfach an weiteren CCUs angelernt, die nur als Schnittstellen fungieren und als Software wird die CLX-Version benutzt.

Das Funktionsprinzip ist grundsätzlich einfach:
Jeder Sensor (also z.B. jede Taste einer Fernbedienung) und jeder Aktor (z.B. die einzelnen Ausgänge eines 4-fach-Aktormoduls) werden in der Software als „Objekt“ dargestellt.
Es können auch Objekte ohne zugehöriges Hardware-Modul erstellt werden, diese werden dann als virtuelle Objekte bezeichnet.
Jedes Objekt hat ein eigenes Script.
Makros von Sensor-Objekten werden normalerweise ausgeführt, wenn von der Sensor-Hardware eine Meldung empfangen wird. Dazu ist bei Sensoren standardmässig die Option "Ausführen bei Empfang" aktiviert.
Scripts von Aktoren können in verschiedenen Ausführungsintervallen von mehrmals in der Sekunde bis einmal täglich aktiviert werden.
Die Visualisierungen können optional mit Hintergrundbildern, z.B. Grundrissen eines Hauses, versehen werden.
Die umfangreichen Möglichkeiten der Software erlauben die Realisierung sehr grosser und komplexer Anwendungen.

Beim HomeMatic-System kann eine CCU von eQ-3 oder eine alternative Zentrale benutzt werden, die Zentrale egal welchen Typs wird im Folgenden immer als Zentrale bezeichnet.

Die CL-Software besteht aus mehreren Programmen:

CL-Konfigurationsprogramm
Mit diesem Programm werden die Anwendungsfunktionen über einen Browser erstellt. Diese Konfigurationen und Programmierungen umfassen Zeittabellen zur Anwesenheitssimulation und zum Schalten einzelner Objekte, Scripts zur Programmierung komplexer Anwendungsfunktionen und die Erstellung von Ansichten zur Visualisierung.
Bitte die Namenskonventionen beachten:
Da Umlaute, Leerzeichen und Sonderzeichen vom Browser in spezielle Codes umgewandelte werden können diese Zeichen für Projektnamen, Objektnamen und Modulnamen nicht benutzt werden. In Bezeichnungen von Objektnamen können jedoch alle Zeichen benutzt werden.

Die Browser-Seiten sind für einen PC-Bildschirm konzipiert, bei Benutzung eines es kann auch ein Tablet oder Smartphone benutzt werden. Je nach Seite kann es im Konfigurationsprogramm bei einem Smartphone allerdings je nach Displaygrösse aber etwas eng werden. Die Visualisierungsseiten können bei der Erstellung an die Grösse des Zielgeräts angepasst werden.

Nach Beendigung der Konfiguration und Programmierung, erstellt der Compiler (der aus dem Konfigurationsprogramm aufgerufen wird) eine Datei mit optimiertem Ausführungscode,.
Dieser Ausführungscode wird dann von der

ExecEngine (Zusatzsoftware CL-Studio) verarbeitet.
Die ExecEngine ist eine Software die auf Grundlage der erstellten Ausführungsdatei die Anwendungsfunktionen realisiert und die Schnittstelle für Visualisierungen bereitstellt.

1.4. Unterschiede zu Version 4.1

Unterschiede zu Version 4.1

Der wichtigste Unterschied zwischen Version 4.1 und 5.0 ist, dass für Version 5.0 kein PC mehr erforderlich ist und diese Version direkt auf der Zentrale installiert wird. Die Konfiguration und Programmierung erfolgt über Browser auf einem beliebigem Gerät.

Ein wichtiger Unterschied ist, dass Variablen in Version 5 nicht mehr über eine Tabelle definiert werden, sondern über die Anweisung VARDEF in den Scripten.

1.5. Software-Installation

Software-Installation

Installation der Software auf der HomeMatic-Zentrale (CCU):

Es sind zwei separate Zusatzsoftware-Pakete erforderlich,
einmal das Konfigurationsprogramm für die Version CL 5.0 als Zusatzsoftware zur CCU und
die jeweils aktuelle Version der ExecEngine als Zusatzsoftware.

Diese Zusatzsoftware (CL-Studio 5) für CCUs kann von folgender Seite heruntergeladen werden:

http://www.cl-control.de/software/downloads.html

Die Downloaddatei wird jeweils in einem beliebigen Verzeichnis (normalerweise automatisch das Download-Verzeichnis) gespeichert.

Bitte beachten Sie, dass für die unterschiedlichen Versionen der CCUs bzw. alternativer Zentralen unterschiedliche Versionen der Zusatzsoftware verwendet werden müssen.

Versionen für das Konfigurationsprogramm
Für die CCU2 wird die Installationsdatei CL5*CCU2.tgz* benutzt.
Für die CCU3 und RaspberryMatic wird die Installationsdatei *CL5CCU3.tgz* benutzt.

Versionen für die ExecEngine
Für die CCU2 wird die Installationsdatei *HPInstCCU2.IMG* benutzt.
Für die CCU3 wird die Installationsdatei *CLInstCCU3RM.IMG* benutzt.
Für die RaspberryMatic wird die Installationsdatei *CLInstRaspMatic.IMG* benutzt.

Installieren Sie die Zusatzsoftware auf der Zentrale, indem Sie in der WEB-UI folgenden Menüpunkt aufrufen:
Einstellungen->Systemsteuerung->Button Zusatzsoftware
Wählen Sie die heruntergeladene Zusatzsoftware mit dem Button **Durchsuchen** aus und klicken dann auf **Installieren**.

Die CCU wird bei der Installation neu gestartet, was einige Minuten dauern kann.

Bitte unbedingt beachten:
Bei einer CCU3 (und RaspberyMatic) müssen noch die Standard-Sicherheitseinstellungen angepasst werden, damit auf diese zugegriffen werden kann.
Hinweise dazu sind hier beschrieben:
http://www.cl-control.de/downloads/pdf/SicherheitseinstellungenCCU3.pdf

Die Software muss über die PLN (Persönlichen Lizenz-Nummer) aktiviert werden, um ohne zeitliche Einschränkung benutzt werden zu können.

Wenn die Software ohne PLN benutzt wird, wird automatisch eine temporäre PLN erzeugt, mit der die Benutzung für einen begrenzten Zeitraum möglich ist.

Freigabe der Software zur zeitlich unbeschränkten Nutzung

Wenn Sie eine PLN erworben haben kann die Software ohne zeitliche Begrenzung benutzt werden.

Die Freigabe ist nur einmalig erforderlich und wird gespeichert.
Wenn Sie mit einer Zentrale arbeiten wird die Freigabe dort gespeichert und ist nicht PC-abhängig.

Bitte bewahren Sie Ihre PLN unbedingt gut auf, bei Verlust kann kein Ersatz geliefert werden! .
Sie brauchen die PLN für eine erneute Freigaben bei einem eventuellen Hardwarewechsel.

Die Freigabe zur zeitlich unbegrenzten Nutzung erfolgt unter dem Menüpunkt Software->Lizenz.

1.6. Einstellungen HomeMatic-CCU

Einstellungen HomeMatic-CCU

Bitte unbedingt beachten:
Bei einer CCU3/RaspberyMatic müssen noch die Standard-Sicherheitseinstellungen angepasst werden, damit auf diese zugegriffen werden kann.
Hinweise dazu finden Sie hier:
http://www.cl-control.de/downloads/pdf/SicherheitseinstellungenCCU3.pdf

1.7. Installation Hardware

Installation Hardware

Die Zentrale oder an der Zentrale angeschlossene Funkschnittstellen dürfen nicht unmittelbar neben oder auf anderen elektronischen Geräten wie WLAN-Routerund PCs stehen, insbesondere wenn diese Funkfunktionen beinhalten.

Funkmodule sollten nicht in der Nähe grosser Metallflächen oder direkt neben oder auf elektronischen Geräte aufgestellt werden.

Die Reichweite von Funkmodulen im Gebäude ist abhängig von den individuellen Örtlichkeiten und wird durch Decken, Wände und elektromagnetische Strahlungen vermindert.

Beachten Sie unbedingt die Sicherheitshinweise der jeweiligen Bedienungsanleitungen bei der Installation von elektrischen Komponenten.

1.8. Die wichtigsten Begriffe des Systems

Die wichtigsten Begriffe des Systems

Ein **Projekt** ist die komplette individuell erstellte Anwendung.

Script ist die Bezeichnung für eine oder mehrere Anweisungen, die bestimmte Aktionen bewirken. Das System verfügt über leistungsfähige Anweisungen, mit denen auch sehr komplexe Funktionen realisiert werden können. In Scripts können Aktionen über wenn-Anweisungen von bestimmten Bedingungen abhängig gemacht werden. Das können Zustände bzw. Werte von Objekten, aber auch z.B. Wochentag und Uhrzeit sein.

Aktoren sind Module, die Endgeräte wie Beleuchtungen, Jalousien usw. schalten und steuern.

Sensoren sind Eingabeelemente wie z.B. Taster, Fernbedienungen, Bewegungsmelder und Temperatursensoren. Sensoren wir Taster und Fernbedienungen, die einen Zustand senden (z.B. aus/an/kurz/lang) werden als Binärsensoren oder Schaltsensoren bezeichnet. Sensoren, die einen Zahlenwert senden, also z.B. Temperatur oder Luftfeuchtigkeit werden als Analogsensoren bezeichnet.

Objekt ist der Oberbegriff für Sensoren und Aktoren, wobei der Begriff „Objekt“ auch die Definitionen und Scripts des Sensors bzw. Aktors mit einschliesst. Wenn das Steuerungsprogramm Objektzustände von Aktor-Objekten verändert wird automatisch eine entsprechende Meldung an den Aktor gesendet. Wenn ein Sensor eine Meldung sendet wird das Script des Sensor-Objekts ausgeführt.
In der Studio-Version und höheren Versionen gibt es die Möglichkeit Objekte ohne zugeordnete Hardware zu definieren, sogenannte "virtuelle Objekte".

Visuakisierungsansichten zeigen die Zustände und Werte von Objekten und ermöglichen je nach Objekttyp die Änderung von Zuständen und Werten am Bildschirm.
In der Standard-Version können bis zu drei, in der Studio-Version beliebig viele individuelle Ansichten erstellt werden. Diesen Ansichten kann optional ein Hintergrundbitmap zugewiesen werden.
Welche Darstellungsart verwendet werden soll, kann für jede Ansicht in der jeweiligen Definition eines Objekts definiert werden.

1.9. Funktionsweise

Funktionsweise

Im Folgenden ist eine erste kleine Anwendung beschrieben.

Beim HomeMatic-System:
Das Anlernen der Geräte an CCUs wird ebenso wie die Einstellungen von Grundparametern an den Geräten mit der WEB-UI der CCUs vorgenommen. Informationen zur WEB-UI und zum Anlernen und Konfigurieren der Geräte finden Sie in der jeweiligen Bedienungsanleitung der CCU.
Nach dem Anlernen werden die angelernten Module mit dem Menüpunkt ***Hardware.>Import von CCU*** in die CL-Software importiert. Nach der Erstellung der Anwendungsfunktionen und der optionalen Erstellung individueller Visualisierungsansichten wird eine Ausführungsdatei mit den erstellten Konfigurationen und Programmierungen erstellt.

Funktionsweise

Jeder Sensor (also z.B. jede Taste einer Fernbedienung oder ein Thermometer) und jeder Aktor (z.B. eine Schaltsteckdose oder ein Rollladenaktor) werden in der CL-Software als „Objekt“ dargestellt.
Es können auch eigene Objekte ohne zugehöriges Hardware-Modul erstellt werden, diese werden dann als virtuelle Objekte bezeichnet.
Für jedes Objekt kann ein eigenes Script erstellt werden.
Scripts von Sensor-Objekten werden normalerweise ausgeführt, wenn von der Sensor-Hardware eine Meldung empfangen wird. Dazu ist bei Sensoren standardmässig die Option "Ausführen bei Empfang" aktiviert.
Scripts von Aktoren können in verschiedenen Ausführungsintervallen aktiviert werden, das gewählte Ausführungsintervall sollte dabei nicht kleiner als nötig sein. Ausführungsintervalle unter 5 Sekunden sollten nur in speziellen Ausnahmefällen verwendet werden. Diese Scripts können z.B. den Zustand des Aktors und die Einschaltdauer prüfen um dann bestimmte Aktionen durchzuführen

1.10. Hauptmenü

Hauptmenü

Über das Hauptmenü werden alle Programmfunktionen zur Konfiguration und Programmierung aufgerufen.

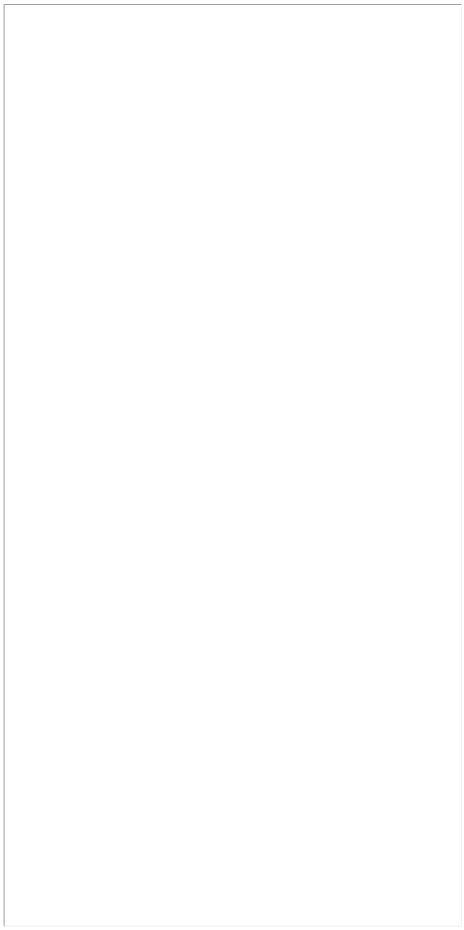
Die Menüstruktur wird automatisch an die Grösse des benutzten Bildschirms angepasst.

Wenn der Bildschirm gross genug ist wird die erste Menüebene in der oberen Zeile angezeigt, das sieht dann so aus:

Bei einem kleinen Bildschirm am PC oder bei einem Tablet erscheint ein Menüsymbol in der oberen linken Ecke und das Menü klappt sich aus wenn dieses Symbol angeklickt wird., Das sieht z.B: so aus:

1.10.1. Projekt

Menüpunkt Projekt



Mit dem ersten Menüpunkt **Visualisierung anzeigen** werden die Ansichten angezeigt, die unter dem Menüpunkt **Programmierung -> Visualisierungsansichten** erstellt worden sind.

Mit dem Menüpunkt **Historyansichten** werden die vordefinierten Historyansichten angezeigt, die unter dem Menüpunkt **Programmierung -> Historyansichten** erstellt worden sind.

Unter dem Menüpunkt **History Schnellansicht grafisch** können schnell und unkompliziert die Historydaten ausgewählter Objekte angezeigt werden (s. Beispiel am Ende dieser Seite).

Mit dem Menüpunkt **History Textanzeige** können zu ausgewählten Objekten der Wert und die Quelle des Werts als Text angezeigt werden. Die ausgewählten Daten können mit einem Button am Ende der Seite aus der Datenbank gelöscht werden.

Mit dem Menüpunkt **Neu** wird ein neues Projekt erstellt.
Mit dem Menüpunkt **Öffnen** wird ein vorhandenes Projekt zur Bearbeitung geöffnet.
Mit dem Menüpunkt **Aktuelles Projekt** wirdder Name des aktuell geöffneten Projekts angezeigt.
Mit dem Menüpunkt **Kopieren oder umbenennen** können Projekte kopiert oder umbenannt werden.
Mit dem Menüpunkt **Löschen** wird ein vorhandenes Projekt gelöscht.

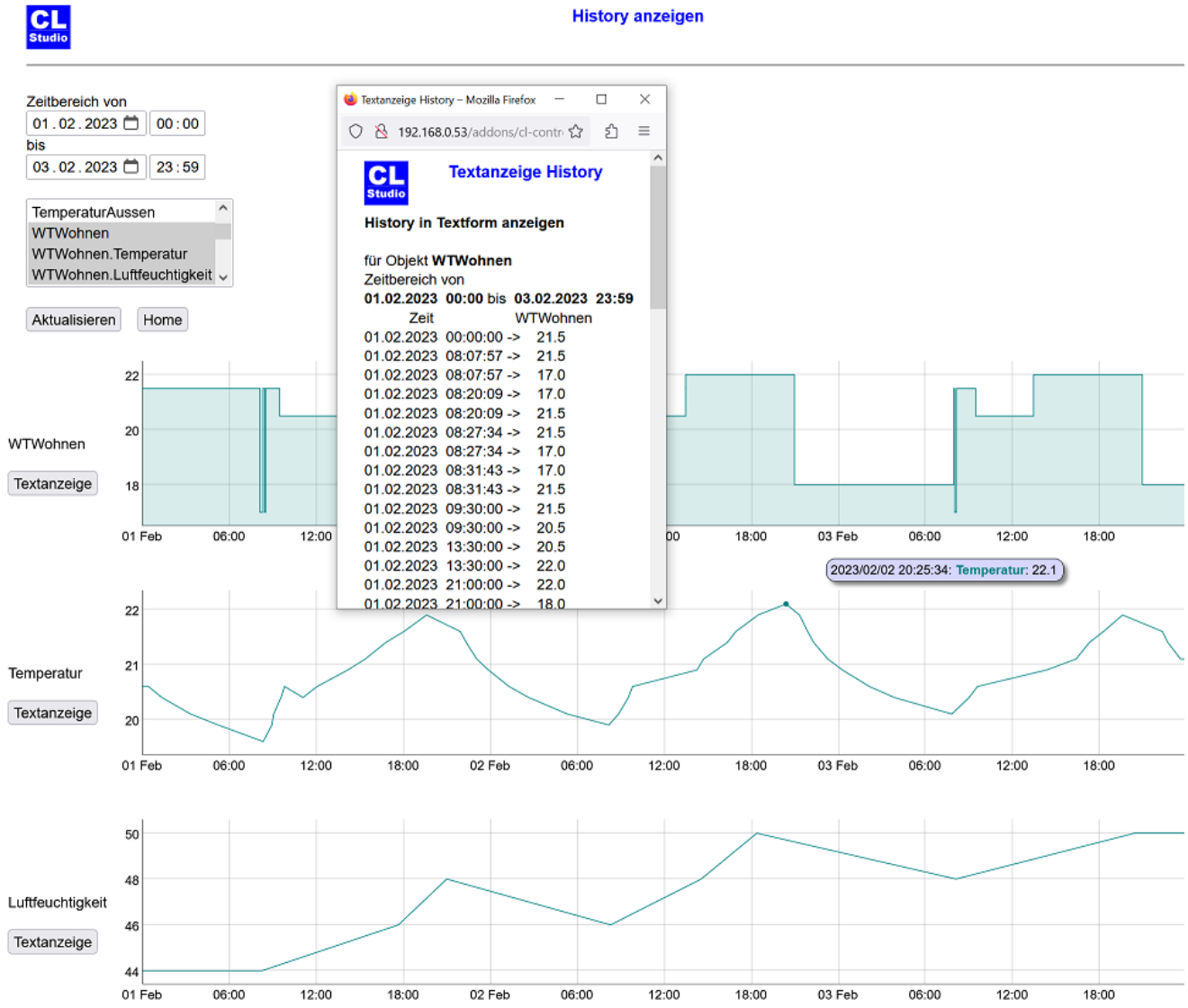
Im Untermenü **Backups** können Backups erstellt und zurückgespeichert werden.

Mit dem Menüpunkt **Projektdateien reorganisieren** werden Projektdateien auf Grundlage der Dateien im Projektverzeichnis neu erstellt. Dieser Menüpunkt sollte nur aufgerufen werden wenn Fehler bei der Konfiguration des Projekts auftreten. Je nach vorhandenem Fehler in den vorhandenen Dateien können eventuell nicht alle Daten korrekt wiederhergestellt werden.

Mit dem Menüpunkt **Moduldateien reorganisieren** wird die Datenkonsistenz zwischen Modul- und Objektdateien geprüft und ggfs. ergänzt oder berichtigt.

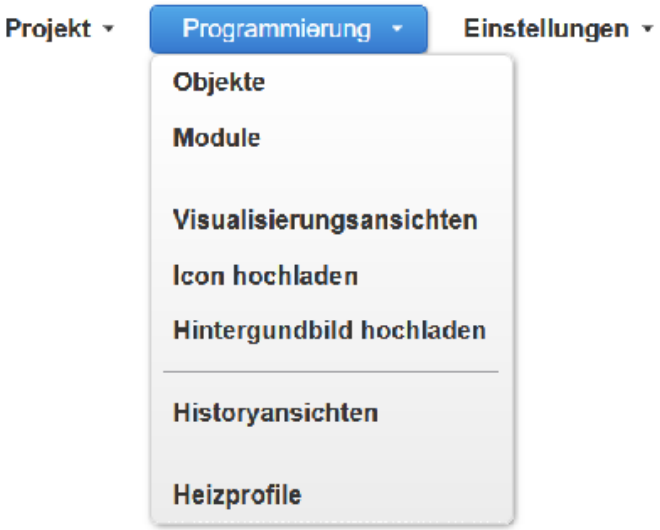
Mit dem Menüpunkt **Benutzer** können bis zu 5 verschiedene Benutzer für das aktuelle Projekt angelegt werden. Wenn Benutzer angelegt sind wird beim ersten Aufruf eines Menüpunkts Benutzer und Kennwort abgefragt.

Beispiel für eine History-Anzeige mit dem Menüpunkt **History allgemein**



1.10.2. Programmierung

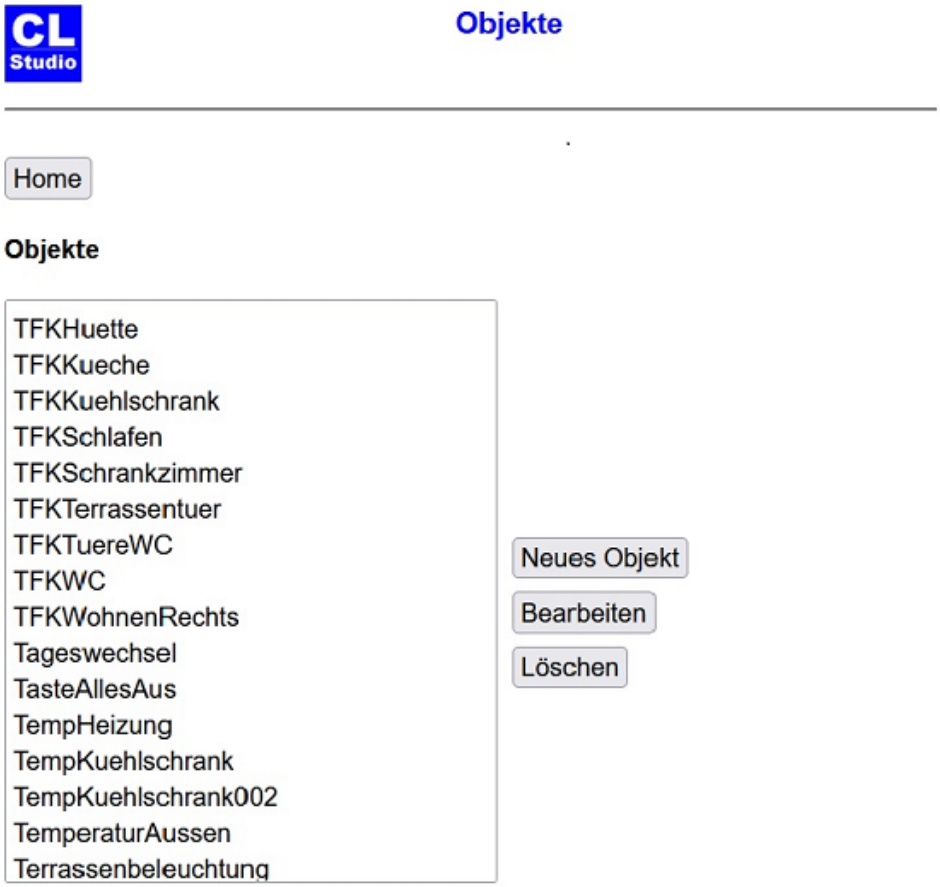
Menüpunkt Programmierung



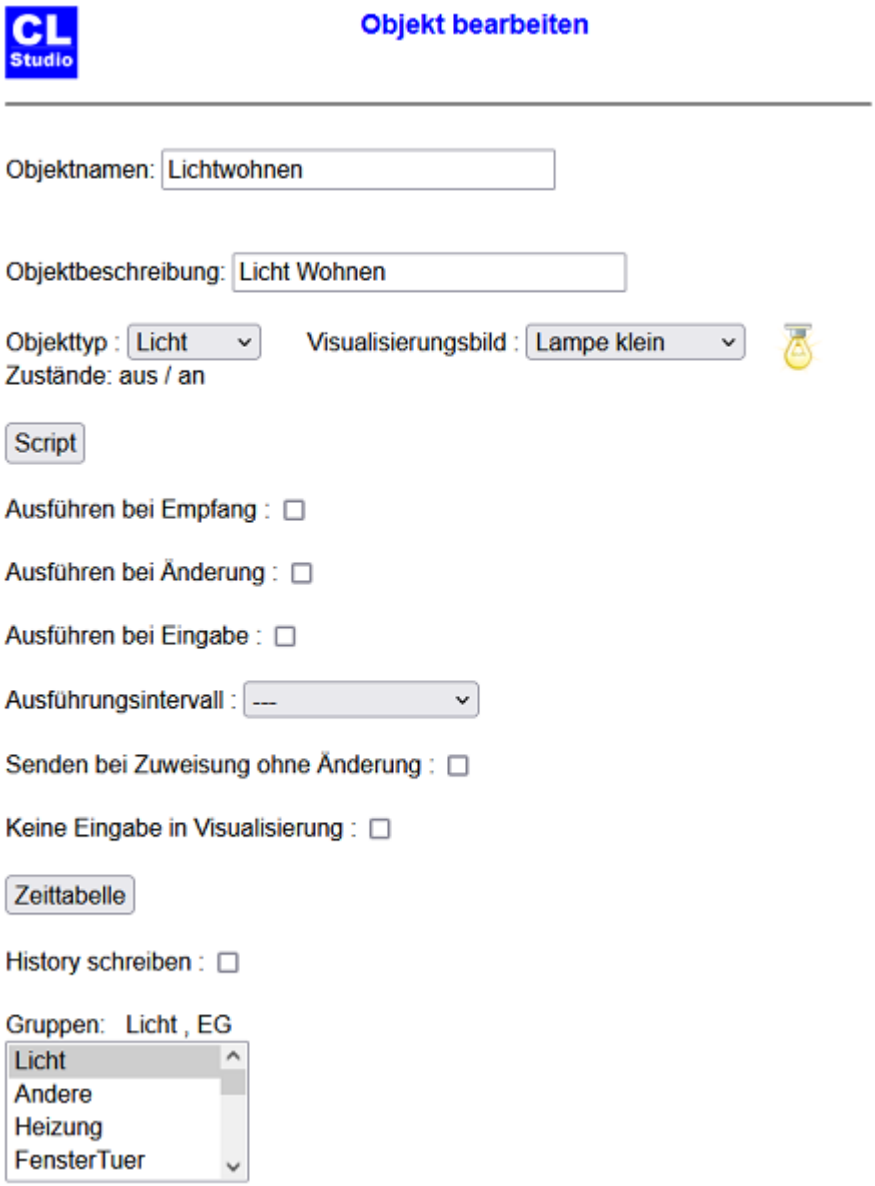
1.10.2.1. Objekte

Objekte

Unter diesem Menüpunkt werden einzelne Objekte verwaltet.
Bei PC-Browsern werden mehrere Objekte in einer Listbox angezeigt, bei Browsern von Tablets und Smartphones wird normalerweise nur ein einzeiliges Listfeld angezeigt, das durch Anklicken geöffnet wird..



Bei Erstellung oder Bearbeitung eines Objekts sieht die Seite je nach Typ beispielsweise so aus:



In den ersten beiden Eingabefeldern werden Objektnamen und Beschreibung angegeben. Beim Objektnamen sind die Namenskonventionen zu beachten, es sind nur Buchstaben und Ziffern erlaubt, keine Leerzeichen oder Sonderzeichen/Umlaute, das erste Zeichen muss ein Buchstabe sein.

In der Auswahlbox *Objekttyp* wird der Typ des Objekts festgelegt. Es werden nur die Objekttypen zur Auswahl angeboten, die auch im Hinblick auf andere grundsätzliche Eigenschaften des Objekts möglich sind. Bei der Auswahl muss darauf geachtet werden, dass der ausgewählte Objekttyp auch im Hinblick auf die für das Objekt verwendete Hardware sinnvoll ist.

In der Auswahlbox Visualisierungsbild wird festgelegt welches Icon für dieses Objekt in der Visualisierung benutzt wird.

Mit dem Button Script wird eine neue Seite zur Eingabe eines Scripts geöffnet.

Die Auswahlbox ***Ausführen bei Empfang*** wird nur angezeigt wenn das Objekt zu einem Sensor gehört.
Wenn diese aktiviert ist wird das Script ausgeführt wenn eine Meldung von dem Sensor empfangen wird.

Wenn die Auswahlbox ***Ausführen bei Änderung*** aktiviert ist, wird das Script immer dann ausgeführt, wenn sich der Wert des Objekts ändert - egal an welcher Stelle diese Änderung erfolgt (also auch innerhalb des eigenen Objektscripts).
Die Benutzung dieser Option ist insbesondere sinnvoll in Scripts von Tür/Fensterkontakten, da bei Verwendung von Ausführung bei Empfang das Script auch beim Empfang von Statusmeldungen aktiviert wird, ohne dass sich der Zustand geändert hat.
Diese Option bietet viele Möglichkeiten, die Benutzung ist jedoch auch gefährlich - wenn mehrere

Objekte diese Option verwenden und es gegenseitige Abhängigkeiten gibt, so dass Scripts wechelseitig aktiviert werden können Endlosschleifen entstehen, die das ganze System blockieren. Diese Option sollte daher nur mit Vorsicht benutzt werden, wenn in dem Script Zustände anderer Objekte verändert werden, die ebenfalls diese Option verwenden.
Achtung: Die unsachgemässe Verwendung dieser Option kann Fehlfunktionen verursachen.

Wenn die Auswahlbox ***Ausführen bei Eingabe*** aktiviert ist, wird das Script ausgeführt wenn eine Eingabe über die Visualisierung erfolgt.

In der Auswahlbox *Ausführungsintervall* wird eingestellt in welchen Zeitabständen das Script ausgeführt werden soll. Bei Sensoren wird diese Option normalerweise nicht benutzt weil die Scripts von Sensoren üblicherweise bei Empfang vom Sensor ausgeführt werden.
Bei Aktoren kann es sinnvoll sein das Script im Zeitintervall auszuführen, z.B. um den Aktor immer nach einer bestimmten Zeit auszuschalten. In den meisten Fällen ist das Zeitintervall "jede Minute" oder "alle 5 Sekunden" die richtige Wahl. Wenn im Script Zeiten abgefragt werden, bei denen Sekunden angegeben sind, muss das Zeitintervall auf alle 5 Sekunden eingestellt werden, da die Funktion "Uhrzeit" die Zeit im 5-Sekunden-Intervall zurückgibt.
Ausführungsintervalle kleiner 5 Sekunden sollten nur gewählt werden wenn diese kurzen Intervalle unbedingt erforderlich sind. Wenn zu viele Scripts unnötigerweise in sehr kurzen Intervallen ausgeführt werden, kann das dazu führen, dass die Anwendung insgesamt langsamer wird und einzelne Aktionen grössere Reaktionszeiten haben, da Scripts die unnötigerweise zu oft ausgeführt werden die Gesamtperformance und die Kommunikation mit der Hardware verlangsamen können.

Bitte beachten Sie:
Die Zeitintervalle *permanent* doder mehrmals in der Sekunde sind nur in Sonderfällen sinnvoll und sollten keinesfalls unnötigerweise gewählt werden. Diese Intervalle sollten auch nicht benutzt werden, wenn es sich um ein grösseres Makro handelt oder andere Scripts in dem permanent aktivierten Script gestartet oder aufgerufen werden.
Keinesfalls dürfen von einem permanenten Script bei jedem Durchlauf Zustands- oder Werteänderungen ausgehen, die zu einer Hardwareaktion oder einer Änderung der Visualisierung führen würden. Das könnte im Extremfall die Netzwerkkommunikation beeinträchtigen und zu einem Überlauf des Duty-Cycle führen, was letztlich Fehlfunktionen verursachen kann. Verwenden Sie also das Zeitintervall *permanent* nur als erfahrender Anwender wenn es unbedingt erforderlich ist und Sie genau wissen was Sie tun.

Sollten Sie permanente Ausführung oder ein Ausführungsintervall kleiner als 1 Sekunde verwenden und Ihr Projekt nicht fehlerfrei laufen erhöhen Sie bei der Fehlersuche als erstes das Zeitintervall.

Wenn die Auswahlbox ***Senden bei Zuweisung ohne Änderung*** aktiviert ist wird auch eine Meldung an die Hardware generiert wenn sich der Zustand des Objekt durch eine Zuweisung nicht verändert hat. Im Normalfann wird dann keine Meldung generiert weil der Zustand der Hardware ja nicht geändert werden muss und unnötige Funkmeldungen vermieden werden sollten um den Duty-Cycle nicht unnötig zu erhöhen und anderen Funkverkehr nicht zu stören.

Mit der Auswahlbox ***Keine Eingabe in Visualisierung*** kann die Änderung des Objekt auf der Visualisierungsseite gesperrt werden.

Mit dem Button ***Zeittabelle*** kann eine Zeittabelle verwaltet werden, mit der der Objektwert zu bestimmten Wochentagen und Uhrzeiten verändert wird. Diese Option steht nicht bei allen Objekten zur Verfügung (z.B. normalerweise nicht bei Sensoren).

Wenn die Auswahlbox ***History schreiben*** aktiviert ist wird jede Änderung des Objekts in die History-Datenbank geschrieben, damit die Historydatenbank geschrieben wird muss die Historyfunktion auf der Seite *Hardware->Zentrale* mit den Auswahlboxen *Historydatenbank* und *History schreiben* aktiviert sein.

Über die Auswahlliste ***Gruppen*** können dem Objekt eine oder mehrere Gruppen zugeordnet werden, damit ist es möglich ist Gruppenzuweisungen an alle Mitglieder einer Gruppe zu machen ohne die Objekte einzeln aufzuführen.

Der untere Teil der Objektseite steht nicht bei allen Objekten in gleicher Form zur Verfügung weil er je nach verwendeter Hardware unterschiedlich sein kann bzw. gar nicht angezeigt wird.
Änderungen in diesem Teil sollten nur vorgenommen werden, wenn man genau weiss was man tut, weil falsche Eingabe dazu führen können, dass das Objekt bzw. die Hardware zu dem Objekt nicht mehr funktionieren oder nicht mehr benutzt oder geändert werden können. Wenn hier falsche

Eingaben gemacht werden kann es sein, dass das Objekt nicht mehr benutzt werden kann und gelöscht und neu angelegt werden muss.
Daher muss die Bearbeitung dieses Teils explizit durch den Button ***Freigabe*** freigeschaltet werden.

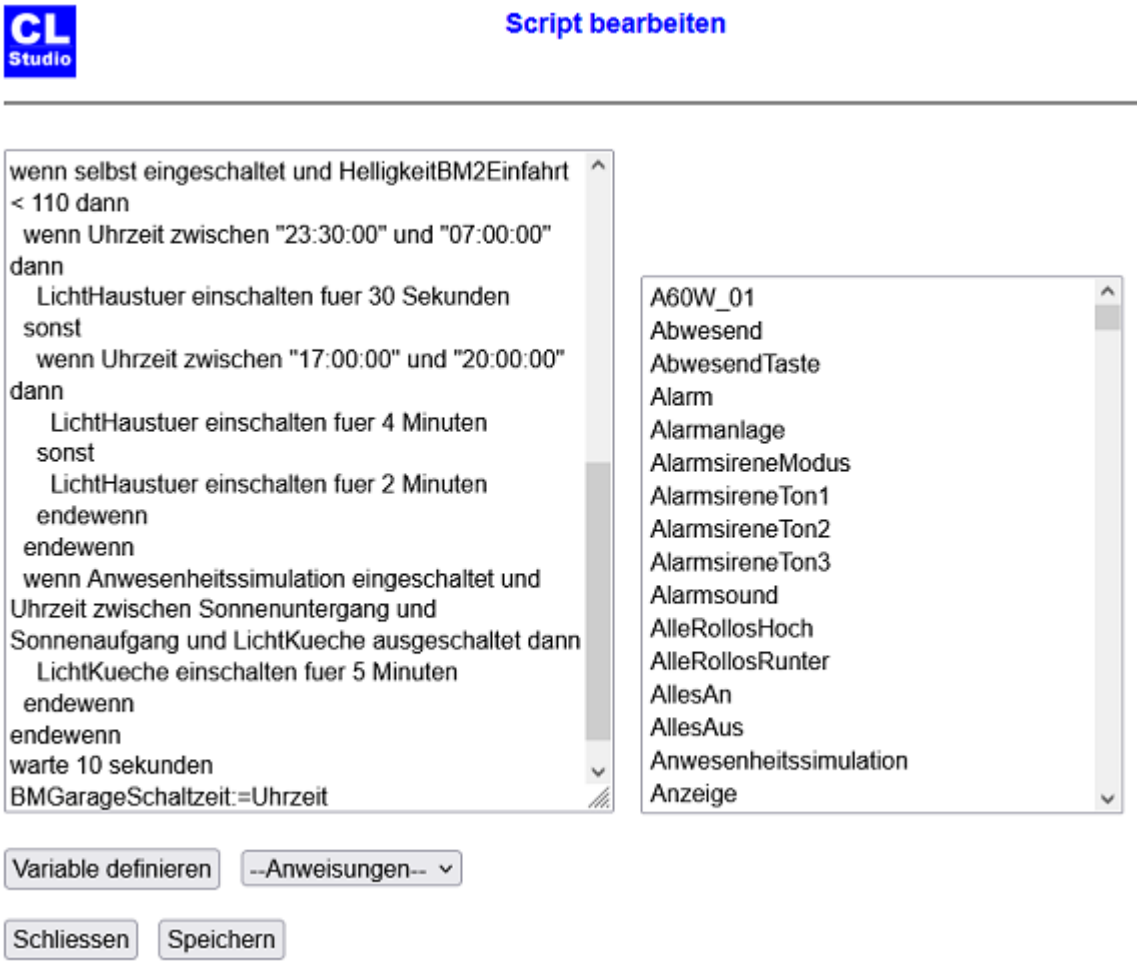
.

1.10.2.1.1. Script

Script

Für jedes Objekt kann ein eigenes Script erstellt werden. In diesem Script wird programmiert welche Funktionen ausgeführt werden sollen. Scripte können durch Empfang vom Sensor eines Objekts, durch eine Änderung des Objektwerts oder in bestimmten Zeitintervallen aktiviert werden.
Welche Option der Ausführungsaktivierung benutzt werden soll, wird auf der Objektseite festgelegt.

Beispiel:



Die Grösse des Eingabefelds für das Script kann durch "Ziehen" der unteren rechten Ecke verändert werden.

In der Listbox rechts vom Script-Eingabefeld werden alle Objekte angezeigt und können durch Anklicken an der Stelle des Cursors eingefügt werden.

Mit dem Button [Variable] definieren werden Eingabefelder zur Variablendefinition geöffnet. Die Variablendefinition wird im Script-Eingabefeld an der Stelle des Cursors eingefügt.
Jede Variablendefinition muss in einer separaten Zeile stehen.

Mit dem Button [Anweisungen] kann eine Anweisung aus der Liste der meistbenutzten Anweisungen ausgewählt werden. Diese wird direkt der Cursorposition im Script eingesetzt. Die Anweisungen sind nicht alphabetisch, sondern nach Verwendungshäufigkeit sortiert.

1.10.2.1.2. Zeit-Tabelle

Seite Zeit-Tabelle

Auf dieser Seite kann festgelegt werden wann ein Aktor-Objekt geschaltet wird bzw. wann ein Script gestartet werden soll.
Bei Sensor-Objekten gibt es diese Seite nicht.

Beispiel:

Auf dieser Seite kann festgelegt werden wann ein Aktor-Objekt geschaltet wird bzw. wann ein Script gestartet werden soll.
Bei Sensor-Objekten gibt es diese Seite nicht.
Die Einschaltdauer steht nur bestimmten Objekten in Abhängigkeit des Modultyps zur Verfügung.
Durch Eingabe von **SA** im Feld Uhrzeit kann die Sonnenaufgangszeit gewählt werden, mit **SU** die Sonnenuntergangszeit.
Die Seite zur Eingabe der geographischen Lage Ihres Standorts wird mit dem Menüpunkt *Einstellungen->Sonnenzeiten* geöffnet.

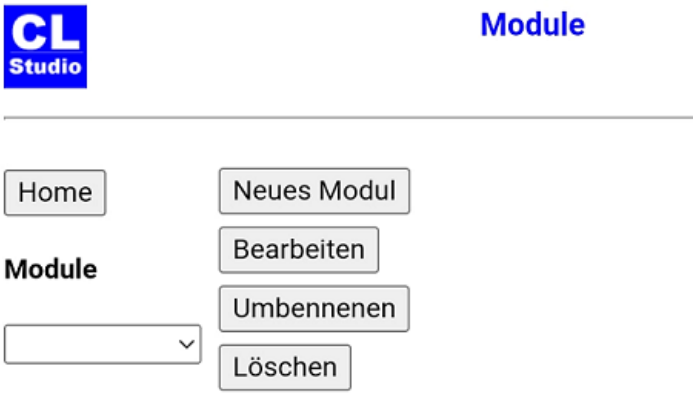
Die Aktivierung der Tabelle kann über Scripts gesteuert werden, indem die Variable des Typs Schalter mit dem Namen *Zeittabelle* ein-/ bzw. ausgeschaltet wird. Diese Variable wird für Aktoren automatisch angelegt. Die in der Tabelle festgelegten Schaltungen werden nur vorgenommen, wenn die Variable *Zeittabelle* des Objekts eingeschaltet ist.
Die Aktivierung der Tabelle kann also z.B. mit den Scriptanweisungen
`Objektname.Zeittabelle einschalten`
und
`Objektname.Zeittabelle ausschalten`
gesteuert werden.

1.10.2.2. Module

Module

Unter diesem Menüpunkt werden die Hardware-Module (bei HomeMatic Geräte genannt) verwaltet.

Bei einem Tablet wird nur eine Zeile der Liste angezeigt:



1.10.2.3. Visualisierungsansichten

Visualisierungsansichten

Auf dieser Seite werden Visualisierungsansichten erstellt und geändert.

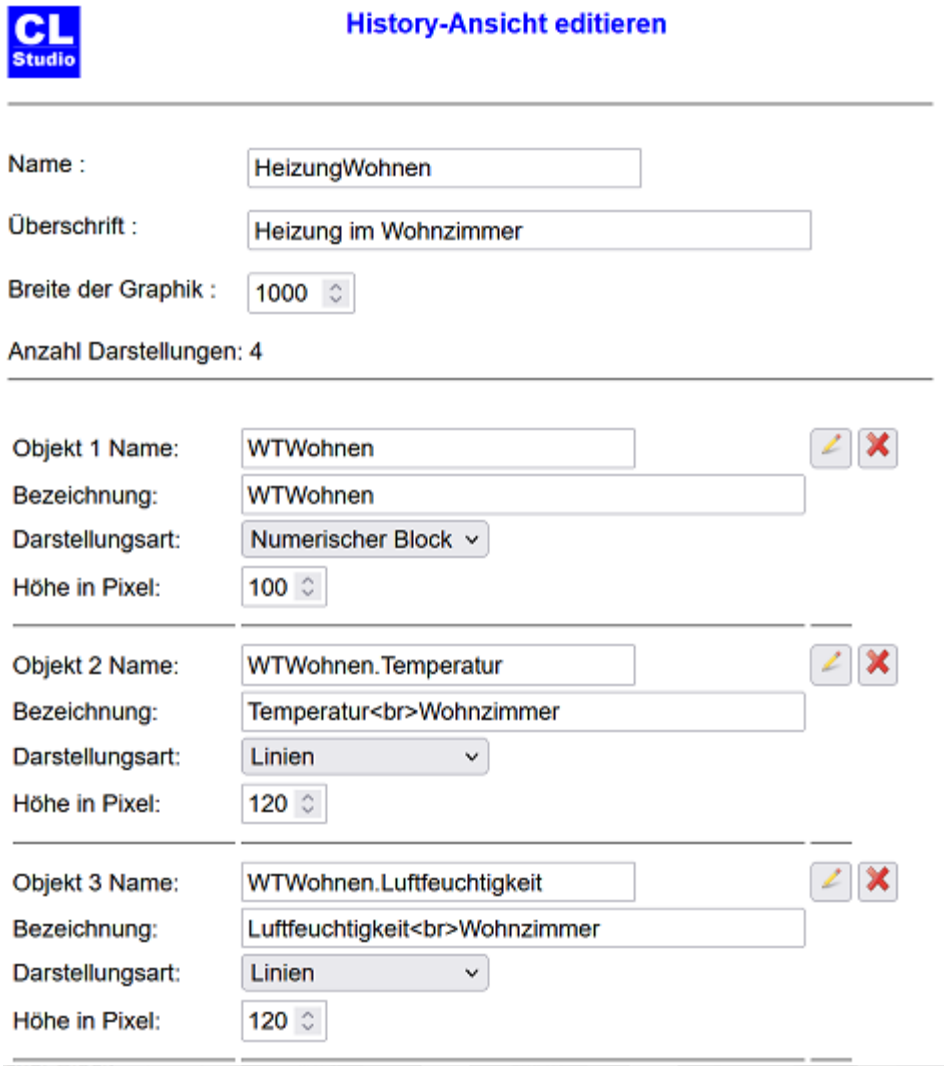
Mit dem Button **Ansicht** können Optionen wie Hintergrundfarbe und Hintergrundbild festgelegt werden, weiterhin kann der Name angegebene werden, unter dem diese Ansicht gespeichert werden soll.
Hinter diesem Button wird das aktuell ausgewählte Objekt angezeigt, durch Klick auf den Button **Objekt-Definitionen** können Positionskoordinaten und Farben für das aktuelle Objekt festgelegt werden. Der Positionierungsmodus wird durch Klick auf das Objekt ein/ausgeschaltet. Im Positionierungsmodus kann das Objekt beim PC mit der Maus positioniert werden, bei Tablet oder Smartphone durch Touch auf die gewünschte Position.

Mit dem Button **Neu** wird ein Objekt des Projekts neu in der Ansicht positioniert, dabei je nach Objekttyp ausgewählt ob nur ein Icon oder ein Objektrahmen dargestellt werden soll. Hier können auch Textfelder und Navigations-Buttons zur Visualisierung und für Hyplerlinks auf beliebige Internetseiten erstellt werden.

Bei Klick auf den Button **Einstellungen** erscheint ein Menüfenster mit Buttons zur Einstellung der Rastergrösse für die Positionierung, Grösse der Objektrahmen und der Rahmenfarben.
Dabei können Grösse und Farbe für das aktuelle Objekt oder für alle Objekt eingestellt werden.

1.10.2.4. Historyansichten

Auf dieser Seite werden Historyansichten erstellt und geändert.
Dabei können mehrere Objekte für eine Ansicht ausgewählt werden.
Weiterhin kann die Breite der Ansicht und die Bezeichnung und Höhe einzelner Objekte angegeben werden.



1.10.2.5. Icon hochladen

Icon hochladen

Mit diesem Menüpunkt werden Icons zur Darstellung von Objektzuständen bei der Visualsierung auf die Zentrale geladen.
Die Icons müssen vom Dateityp .png sein.

1.10.2.6. Hintergrundbild hochladen

Hintergrundbild hochladen

Mit diesem Menüpunkt werden Hintergrund bilder für die Visualisierungsansichten auf die Zentrale geladen.
Dabei können die Dateitypen .jpg, png. und .gif verwendet werden.

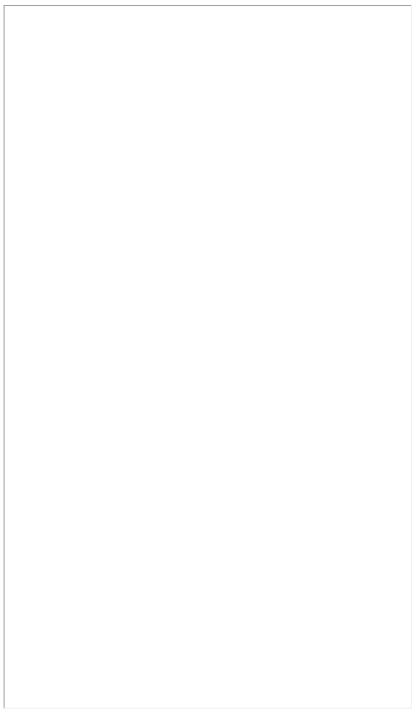
1.10.2.7. Heizprofile

Heizprofile

Unter diesem Menüpunkt werden die Heizprofile für die Steuerung von Heizungen über Schaltaktoren verwaltet.
Diese Funktion wird z.B. für Fussbodenheizungen und Infratot-Heizungen verwendet.

1.10.3. Einstellungen

Einstellungen



1.10.3.1. Sonnenzeiten

Sonnenzeiten

Unter diesem Menüpunkt werden die Sonnenzeiten verwaltet.

In einigen Zeitfenstern kann als Zeitpunkt zur Ausführung einer Aktion die Zeit des Sonnenaufgangs bzw. des Sonnenuntergangs für das aktuelle Datum angegeben werden. In wenn-Bedingungen können diese Zeiten zum Vergleich mit der aktuellen Uhrzeit verwendet werden.

Zur Eingabe des Zeitpunkt Sonnenaufgang geben Sie anstelle der Uhrzeit die Buchstaben SA ein, für den Sonnenuntergang SU.

Es wird der Zeitpunkt der sogenannten „bürgerlichen Dämmerung“ berechnet. Es gibt noch andere Definitionen zu Sonnenaufgangs- und –untergangszeiten (z.B. astronomische Sonnenzeiten, nautische Sonnenzeiten), die teilweise in Abhängigkeit der Jahreszeiten stark voneinander abweichen. Daher sind diese Zeiten eventuell nicht identisch mit Angaben in Tabellen, in denen andere Sonnenzeiten verwendet werden. Die berechneten Zeiten sind bis auf wenige Minuten genau – Vorausetzung ist die Angabe des geografischen Standorts nach Postleitzahl oder Längen- und Breitengrad.

Da je nach Jahreszeit die Zeiten für Sonnenaufgang und Sonnenuntergang sehr früh bzw sehr spät sein können, kann es sinnvoll sein diese Werte zu begrenzen. Es wird dann bei Benutzung dieser Funktionen der angegebene Grenzwert nicht unter- bzw. überschritten.

Weiterhin können individuelle Anpassungen bezüglich der Zeitpunkte festgelegt werden. Dabei werden die Minuten angegeben, um die der berechnete Zeitpunkt verändert wird.

Sonnenzeiten

geografische Länge :

geografische Breite :

Die Sonnenaufgangs- und -untergangszeiten werden nach den Formeln für "bürgerliche Dämmerung" berechnet.
Mit negativen Zahlen kann man die Zeiten jeweils auf eine frühere, mit positiven Werten auf spätere Uhrzeit einstellen.

Korrektur Sonnenaufgang : Minuten

Korrektur Sonnenuntergang : Minuten

Im Winter kann der tatsächliche Sonnenaufgang sehr spät und der Sonnenuntergang sehr früh sein.
Im Sommer hingegen kann der Sonnenaufgang sehr früh und der Sonnenuntergang sehr spät sein.
Ja nach Jahreszeit ist es also nicht immer zweckmässig diese Zeiten zu verwenden, daher gibt es die Möglichkeit die Zeiten individuell anzupassen.

Frühster Sonnenaufgang :

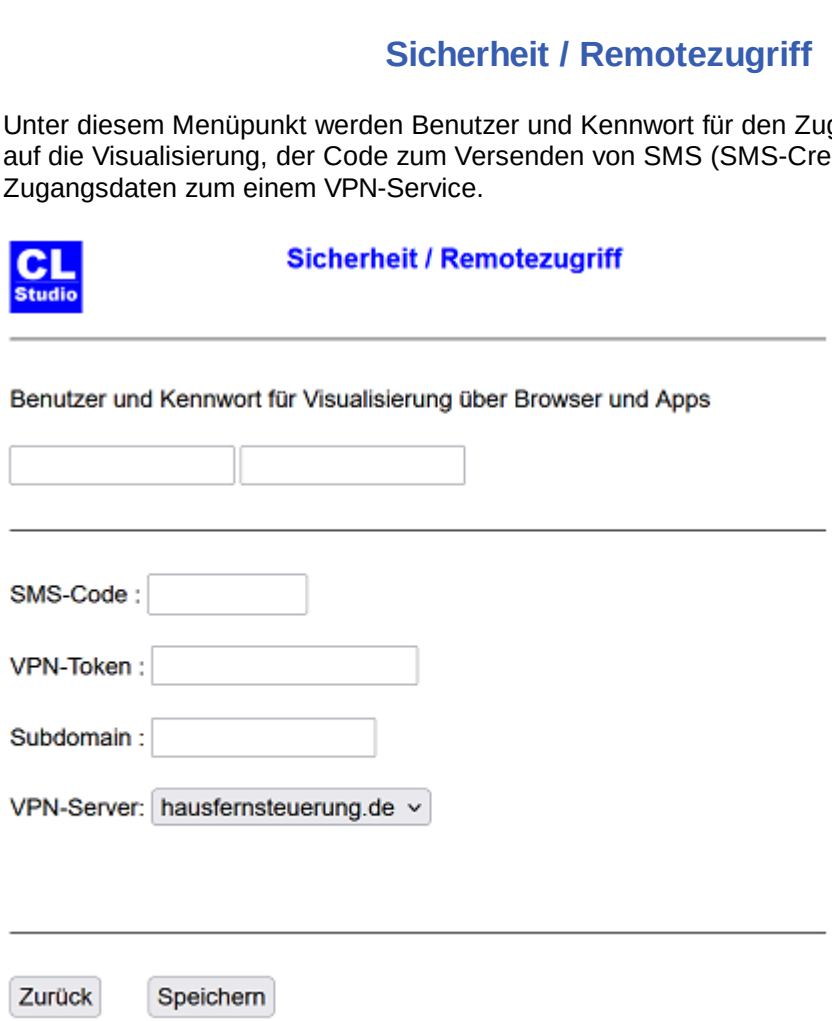
Spätester Sonnenaufgang :

Frühster Sonnenuntergang :

Spätester Sonnenuntergang :

Zurück Speichern

1.10.3.2. Sicherheit/Remotezugriff



1.10.3.3. Anwesenheitssimulation

Anwesenheitssimulation

Das Seite Anwesenheitssimulation wird mit dem Menüpunkt *Einstellungen->Anwesenheitssimulation*.aufgerufen.
Auf dieser Seite kann festgelegt werden, innerhalb welcher Zeiten Aktoren geschaltet werden sollen. Der Aktor wird dann an den ausgewählten Tagen zu einem zufälligen Zeitpunkt innerhalb des angegebenen Zeitraums geschaltet. Wenn keine obere Grenze des Zeitrahmens angegeben wird (Spalte *und*) wird innerhalb von 1-5 Sekunden nach dem in der Spalte *zwischen* festgelegten Zeitpunkt geschaltet.
Wenn keine Dauer angegeben wird bleibt der Schaltzustand erhalten.

Die Anwesenheitssimulation dient dazu, bei Abwesenheit der Bewohner Geräte so zu schalten, dass ein Beobachter den Eindruck bekommt das Haus oder die Wohnung sei nicht verlassen. Normalerweise ist es nicht gewünscht oder sinnvoll, dass die Anwesenheitssimulation dauernd aktiv ist, daher kann die Anwesenheitssimulation über einen Schalter ein- bzw. ausgeschaltet werden.
Sobald Zeiten in der Anwesenheitssimulations-Tabelle eingetragen sind, wird bei der Ausführung ein spezielles Objekt generiert, das Objekt ***Anwesenheitssimulation***. Dieses Objekt wird mit dem Typ *Schalter* angelegt. Somit hat es eine Zeittabelle und kann über die diese wie auch über eine Visualisierung ein- und ausgeschaltet werden.
In den Uhrzeitfeldern können auch Sonnenzeiten angegeben werden, SA für Sonnenaufgang, SU für Sonnenuntergang.
Bei der Verwendung von Sonnenzeiten wird nicht geschaltet, wenn die Sonnenzeiten am aktuellen Tag so sind, dass die Bis-Uhrzeit kleiner als die Von-Uhrzeit ist. In dem Fall wird diese entsprechende Zeitspanne komplett ignoriert.

Wenn es ein Objekt mit dem Namen *Anwesenheitssimulation* (z.B. als Schalter) schon gibt, so wird dieses benutzt um die Anwesenheitssimulation ein- bzw. auszuschalten. Es wird dann kein neues Objekt generiert.

1.10.3.4. Mailserver

Mailserver

Unter diesem Menüpunkt werden die Zugangsdaten zu einem Mailserver angegeben um Mails durch den die Anweisung SENDEMAIL verschicken zu können.

1.10.3.5. Feiertage

Feiertage

Unter diesem Menüpunkt wird festgelegt welche Feiertage bei den Zeittabellen wie Sonntage behandelt werden sollen.

1.10.3.6. Urlaubstage

Urlaubstage

Unter diesem Menüpunkt wird festgelegt welche Zeiträume bei den Zeittabellen wie Sonntage behandelt werden sollen.
Es kann ein von-Datum und ein bis-Datum angegeben werden, wenn kein bis Datum angegeben wird, wird das von Datum als einzelner Tag benutzt..
Wenn keine Jahreszahl angegeben wird werden die Datumsangaben jedes Jahr benutzt.

1.10.3.7. Spezielle Objekte

Spezielle Objekte

Es können spezielle Objekte angelegt werden, die dazu dienen besondere Situationen, Warnungen oder Fehlermeldungen zu bearbeiten oder bestimmte Systemwerte einzustellen.
Spezielle Objekte sind normalerweise virtuelle Objekte, die je nach Art der Verwendung als Objekte vom Typ Zeichen, Schalter oder Zahl angelegt werden.
Existieren keine passenden Objekte für eine Funktion, so bleibt die Auswahlliste für diese Objektart leer.
Details zur Verwendung der einzelnen speziellen Objekte sind jeweils im zugehörigen Abschnitt beschrieben.

Objekt für Sabotagemeldungen
Hier kann ein vorher definiertes Objekt vom Typ Zeichen angegeben werden. Jedesmal wenn eine Sabotagemeldung empfangen wird, wird der Name des Objekts das die Sabotagemeldung gesendet hat als Wert für das *Objekt für Sabotagemeldungen* gesetzt und sein Script wird ausgeführt.

Objekt für Batteriemeldungen
Hier kann ein vorher definiertes Objekt vom Typ Zeichen angegeben werden. Jedesmal wenn eine Batteriewarnung empfangen wird, wird der Name des Objekts, das die Batteriewarnung gesendet hat als Wert für das *Objekt für Batteriemeldungen* gesetzt und das Script wird ausgeführt.

Allgemeines Fehlerobjekt
Hier kann ein vorher definiertes Objekt vom Typ Zeichen angegeben werden. Wenn im System ein Fehler erkannt wird, werden Informationen dazu in dieses Objekt geschrieben und das Script des Objekts wird ausgeführt.

Level für History
Hier kann ein vorher definiertes Objekt vom Typ Zahl angegeben werden. Dieser Wert kann über Scripts während der Programmausführung geändert werden.
Die Zahl kann folgende Werte haben:
0 : Es werden hat werden keine Historydaten geschrieben, auch wenn diese Option für ein Objekt eingestellt ist.
1 : Es werden Historydaten geschrieben für alle Objekte, für die diese Option aktiviert ist.
2 : Es werden Historydaten für alle Hardware-Objekte und für alle Objekte, für die die Historyoption aktiviert ist, geschrieben.
3 : Es werden Historydaten für alle Objekte geschrieben, auch für Objekte für die die Historyoption nicht aktiviert ist.
Bitte beachten Sie unbedingt: Bei Level 2 und 3 können grosse Datenmengen anfallen. Insbesondere bei Verwendung einer CCU muss darauf geachtet werden, dass die Historydatei nicht in den normalen Speicher, sondern auf die SD-Karte oder einen USB-Stick geschrieben, wird um Fehlfunktionen aufgrund eines Speicherüberlaufs zu vermeiden.

Einstellung für Systemlog
Hier kann ein vorher definiertes Objekt vom Schalter oder Zahl angegeben werden. Beim Schalterzustand "ein" werden die Historydaten der Stufe 1 geschrieben, ist das Objekt vom Typ Zahl werden Systemmeldungen der in der Zahl angegebenen Stufe geschrieben.
0=kein Systemlog
1=nur wichtige Systemmeldungen wie Ausführungsstart, Ausführungsende und Fehler,
2=zusätzlich werden Warnungen in die Log-Datei geschrieben,
3=es werden Fehler, Warnmeldungen und Informationen geschrieben.
Die Stufe 3 kann grössere Datenmengen erzeugen und sollte im Normalfall nicht verwendet werden. Beachten Sie, dass es bei einem Speicherüberlauf zu gravierenden Fehlfunktionen und

einem Systemabsturz kommen kann. Das gilt insbesondere bei Verwendung einer CCU ohne DS-Karte oder USB-Stick.

Hauptschalter für Heizen

Hier kann ein vorher definiertes virtuelles Objekt vom Typ Schalter angegeben werden. Beim Schalterzustand "aus" wird die Temperatursteuerung der Thermostate für die Binäraktoren zum Heizen deaktiviert. Mit diesem Schalter ist es möglich in Abhängigkeit von der Jahreszeit oder der aktuell gemessenen Aussentemperatur die Heizfunktion über den Binäraktor komplett zu deaktivieren oder zu aktivieren.

Hauptschalter für Kühlen

Hier kann ein vorher definiertes virtuelles Objekt vom Typ Schalter angegeben werden. Beim Schalterzustand "aus" wird die Temperatursteuerung für den Binäraktoren zum Kühlen deaktiviert. Mit diesem Schalter ist es möglich in Abhängigkeit von der Jahreszeit oder der aktuell gemessenen Aussentemperatur die Kühlfunktion über den Binäraktor komplett zu deaktivieren oder zu aktivieren.

1.10.3.8. Objektgruppen

Objektgruppen

Hier werden Objektgruppen angelegt, die jedem Objekt zugeordnet werden können um mit der Anweisung GRUPPENZUWEISUNG allen Objekten einer Gruppe mit einer Anweisung einen neuen Wert zuweisen zu können.

1.10.3.9. SQL History-Datenbank

SQL History-Datenbank

Auf dieser Seite kann die History-Datenbank mit SQL-Anweisungen bearbeitet werden. Dabei können alle Felder der Datensätze angezeigt werden, Inhalte geändert und Datensätze gelöscht werden.
Dazu werden die SQL-Anweisungen SELECT, UPDATE und DELETE benutzt.

Die Datenbank beinhaltet für jeden Datensatz folgende Felder:
TimeStamp = Zeit in Sekunden seit 1.1.1970, wird als Datum und Uhrzeit eingegeben und für die Ausgabe auch in Datum und Uhrzeit umgerechnet. Der Zeitraum wird in der Abfrage nicht explizit angegeben, sondern automatisch aus den ausgefüllten Eingabefeldern für den Zeitraum eingefügt.
ObjName = Name eines Objekts oder einer Variablen
ValType = Typ bei Objekten (N=numerisch, U=Zustand (z.B. aus/an, offen/geschlossen)
Typ bei Variablen, Spezieller Code z.B. 3=Temperatur, 4=Luftfeuchtigkeit
Value = numerischer Wert (wichtig wenn in Bedingungen kleiner/grösser abgefragt wird)
ValStr = Wert als Text, bei Zahlen mit Komma als Dezimalzeichen, bei Typ U Text z.B. aus/an
Source = Quelle, von der der Wert gesetzt wurde

1 = Wertänderung über Programmierung
2 = Hardware-Bedienung
3 = Status-Meldung von Hardware
4 = Zeittabelle
5 = spezielle Zeitkontrolle (nur c-comatic)
6 = Fensterkontakt (durch Anweisung TEMPERATURABSENKUNG)
7,8 = externer Zugriff über Webserver oder App
9 = aus Wertedatei (z.B. bei Programmstart)
10 = Temperatursteuerung über binäre Schaltaktoren (z.B. Fussbodenheizung)

State.= Status des Datensatzes, Wert kann Ziffer oder Buchstabe sein, wenn ein "X" in State steht werden diese Datensätze bei den Historyansichten im Menüpunkt Projekt ignoriert.

Anweisung SELECT:
Mit dieser Anweisung werden Datensätze aus der Datenbank angezeigt.
Die Syntax ist:
select * from history where <Bedingungen> order by <Feldname>
Bedingungen können mit ***or*** und ***and*** verknüpft werden, es können Klammern zur Verschachtelung benutzt werden, der Operator ***order by*** ist optional, wenn dieser nicht angegeben wird erfolgt die Sortierung immer nach Zeit.
SQL- Anweisungen sind nicht case-sensitive. es kann also Klein und/oder Grossschreibung benutzt werden. Für Text-Werte in Bedingungen muss die exakte Schreibweise benutzt werden, Objektnamen und Variablennamen müssen grundsätzlich immer in doppeltem Hochkomma und Grossbuchstaben geschrieben werden.
Beispiel:
select * from history where objname = "WTWOHNEN" and value < 21 order by value
Diese Anweisung bewrikt, dass alle Datensätze des Objekts WTWOHNEN angezeigt werden, bei denen der Wert kleiner 21 ist, die Sortierung erfolgt hier nach Wert, nicht nach Zeit,
Anstelle des Operators * können auch die Feldnamen die angezeigt werden sollen getrennt durch Komma angegeben werden, dann muss allerdings immer **TimeStamp** als erstes Feld angegeben werden damit die Zeit in der Ausgabe korrekt ausgegeben wird.
Beispiel:
select TimeStamp, ObjName, ValueStr, Source from history where ObjName = "LICHTDIELE"
Hier werden nur die angegebenen Felder aller Datensätze im ausgewählten Zeitraum angezeigt.

Anweisung UPDATE:
Mit dieser Anweisung werden Werte in den Datensätzen geändert.
Die Syntax ist:
update history set <Feldname> = "Neuer Wert" where <Bedingungen>
Bedingungen können mit ***or*** und ***and*** verknüpft werden, es können Klammern zur Verschachtelung benutzt werden.
Beispiel:
update history set State = "X" where ObjName = "AUSSENTEMPERATUR" and Value < 0

Mit dieser Anweisung wird das Feld State in allen ausgewählten Datensätzen auf "X" gesetzt.

Anweisung DELETE:
Mit dieser Anweisung werden Datensätze der Datenbank gelöscht..
Die Syntax ist:
delete from history where <Bedingungen>
Beispiel:
delete from history where ObjName = "AUSSENTEMPERATUR" and Value > 60
Mit dieser Anweisung werden alle Werte des Objekts AUSSENTEMPERATUR gelöscht bei denen ein Wert > 60 eingetragen ist. Eine solche Aktion kann z.B sinnvoll sein wenn ein defekter Sensor falsche Werte gesendet hat..

Bitte unbedingt beachten:

Bei Benutzung der Anweisung SELECT können viele tausend Datensätze sehr schnell gelesen werden.
Bei den Anweisungen UPDATE und DELETE ist jedoch unbedingt zu beachten, dass die Ausführung bei grossen ausgewählten Zeiträumen bzw. Datenmengen sehr lange dauern kann (je nach Anzahl der Datensätze auch länger als 10 Minuten). Während der Ausführung ist kein Zugriff auf die Datenbank möglich, es können in dieser Zeit auch keine neuen Daten in die Datenbank geschrieben werden.
Wenn grössere Zeiträume bearbeitet werden müssen, sollte das daher möglichst in mehreren einzelnen Schritten mit kurzen Zeiträumen erfolgen.
.

1.10.3.10. Typen

Typen

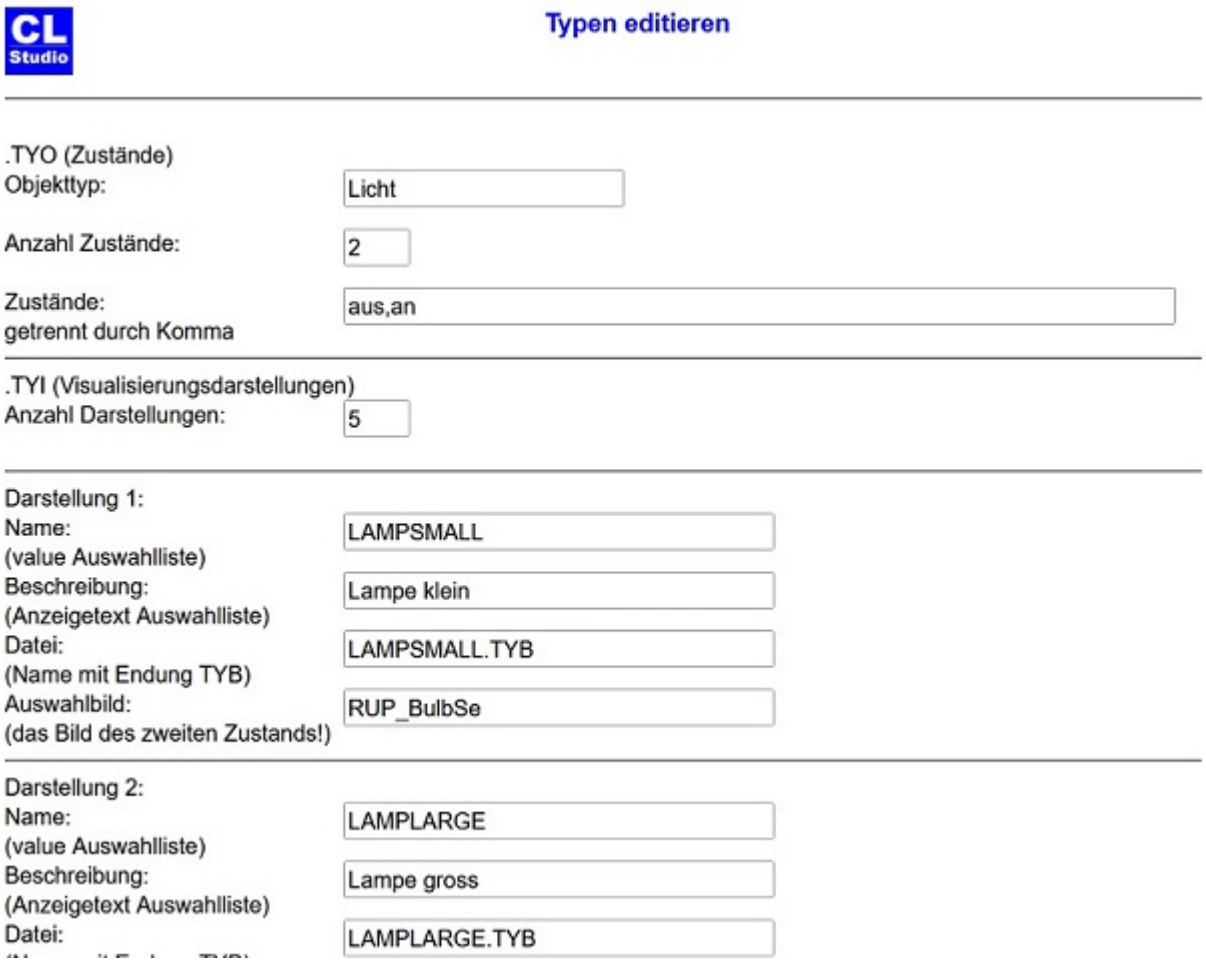
Hier werden Objekttypen und die Darstellung der Objektzustände verwaltet.

1.10.3.10.1. Objekttypen

Objekttypen

Auf dieser Seite werden die Objekttypen verwaltet.
Dabei können vorhandene Objekttypen aus einer Liste zur Bearbeitung ausgewählt und bearbeitet oder auch neue Objekttypen erstellt werden. Dabei werden die möglichen Zustände und die möglichen unterschiedlichen Icons zur Darstellung in der Visualisierung definiert.

Dabei sieht die Definition folgendermassen aus:



Hier werden Name des Typs und die Zustände festgelegt.
Diese Daten werden in der Datei .TYO gespeichert.

Die verschiedenen möglichen Darstellungen für den Objekttyp werden in der Datei .TYI abgespeichert.
Die Anzahl der Darstellungen wird automatisch eingesetzt und darf nicht verändert werden.
Name ist die interne Kennung für die Darstellung, *Beschreibung* ist die Bezeichnung , die in der Objektdefinition angezeigt wird. Unter *Datei* wird der Name angegeben, in dem die Typ-Icons definiert sind.

Diese Datei hat die Endung .TYB, der Dateiname muss derselbe sein, der schon als Name angegeben wurde.

Als *Auswahlbild* wird der Dateiname des Icons angegeben, der auf der Seite für die Objektdefinition

angezeigt werden soll. Das sollte immer das Icon für den zweiten Zustand sein.

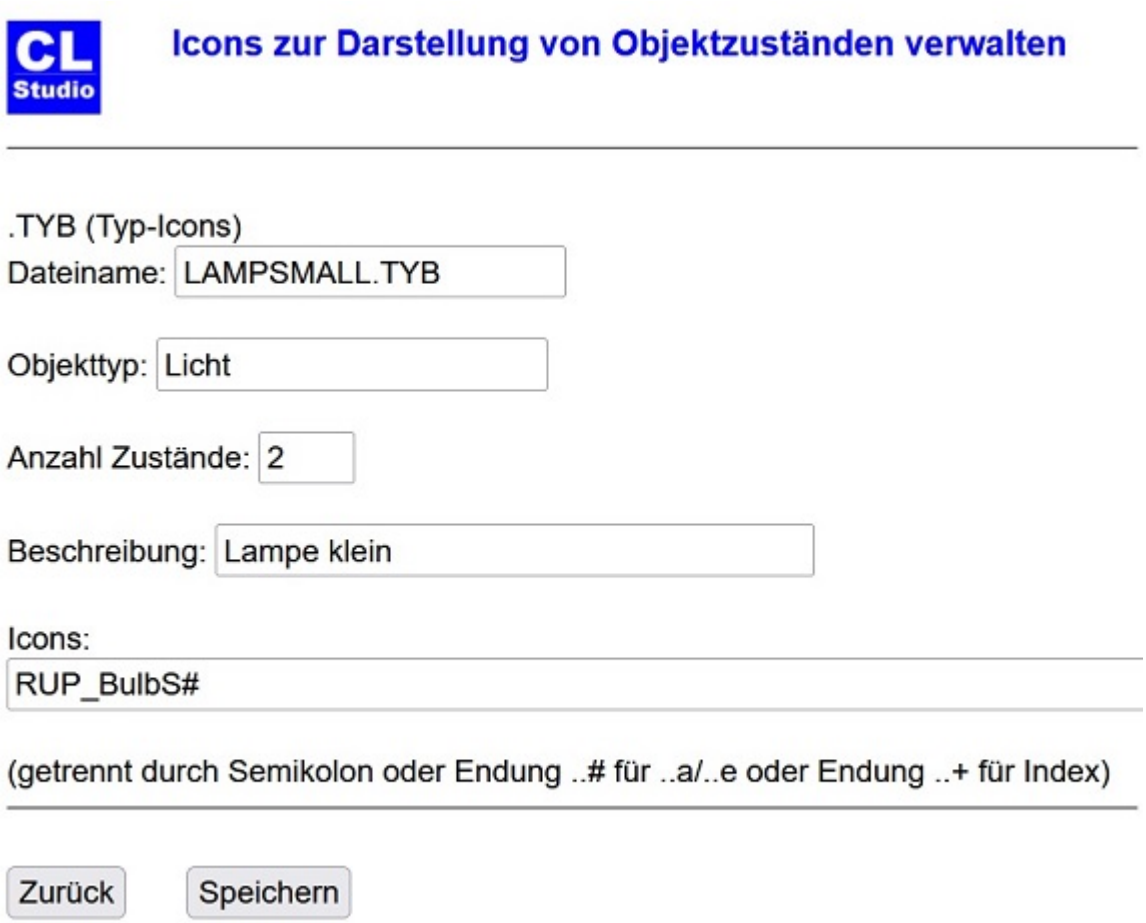
Es folgen beliebig viele unterschiedliche Darstellungsmöglichkeiten für den Objekttyp.

Um eine neue Darstellung hinzuzufügen werden unten auf der Seite im Rahmen "Neue Darstellung die entsprechenden Daten eingegeben und auf den Button Speichern geklickt. Im Beispiel unten wird der Typ Licht um die Darstellung eines Schweinwerfers erweitert.

1.10.3.10.2. Typ-Icons

Typ-Icons

Auf dieser Seite werden die Dateien mit Endung .TYB verwaltet, in diesen befinden sich die Daten für Icons, die für die unterschiedlichen Zustände eines Objekttyps benutzt werden.



Dateiname ist der Name der Datei zur Definition der Icons der unterschiedlichen Zustandstypen, der bei der Definition des Objekttyps angegeben wird.

Unter Objekttyp wird der Name des Objekttyps angegeben.

Im Feld Anzahl Zustände wird die Anzahl der Zustände für diesen Objekttyp angegeben.

Im Feld Beschreibung wird die Beschreibung für die Icons angegeben.

Für Icons wird grundsätzlich der Dateityp .png benutzt, die Dateiendung wird nicht mit angegeben.
Wenn eigene Icons verwendet werden sollen, so können diese diese unter dem Menüpunkt Programmierung->Icons hochladen
auf die Zentrale geladen werden.
Im Feld Icons wird angegeben welche Icons zur Darstellung benutzt werden, es gibt mehrere Möglichkeiten der Eingabe.

Möglichkeit 1:
Es werden die Dateinamen aller Icons beginnend mit dem ersten Zustand getrennt durch Semikolon angegeben. Diese Methode kann immer auch anstelle der weiter unten beschriebenen Verfahren angewendet werden.

Möglichkeit 2:
Wenn es sich nur um zwei Zustände handelt kann man die Dateien bis auf den letzten Buchstaben gleich benennen, der letzte Buchstabe bei der Eingabe im Feld Icons wird dann durch ein '#' ersetzt. In dem Fall muss die Datei mit dem ersten der ersten Zustand als letzten Buchstaben ein kleines 'a' haben und der zweite Zustand als letzten Buchstaben ein kleines 'e' haben.
Beispiel: Dateinamen RUP_BulbSa und RUP_BulbSe, im Feld Icons wird RUP_BulbS# angegeben.

Möglichkeit 3:
Wenn es sich um mehr als zwei Zustände handelt können die Dateinamen bis auf die letzte Stelle gleich benannt werden. Als Kennung wird dann an der letzen Stelle im Feld Icons ein '+' benutzt.
Das Icon für den ersten Zustand muss dann an letzter Stelle eine '1' haben, das für den zweiten Zustand eine '2' usw.
Beispiel: Dateinamen WinS1, WinS2, Wins3, im Feld Icons wird Wins+ angegeben.

1.10.4. Hardware

Hardware

1.10.4.1. Zentrale

Zentrale

Auf dieser Seite werden Einstellungen zur Verbindung mit der Zentrale verwaltet.
Je nach Programmversion können verschiedene Zentralentypen ausgewählt werden.
Bei Auswahl eines Zentralentyps werden automatisch die Verzeichnisse eingetragen, die auf der Zentrale zur Speicherung von Systemlog und Historydaten sowie eigene Dateien verwendet werden.

Die Verzeichnisse dürfen nur verändert werden, wenn andere Verzeichnisse benutzt werden sollen, diese müssen dann auf der Zentrale separat eingerichtet wurden.

1.10.4.2. Gateways

Gateways

Auf dieser Seite werden die verwendeten Gateways angelegt.
Der Typ des Gateways muss ausgewählt werden.

Bei der Version CLX können auch CCUs als Gateways benutzt werden, so dass eine CCU als Zentrale und weitere CCUs als Gateway benutzt werden können.

1.10.4.3. Import von CCU

Import von CCU

Mit diesem Menüpunkt werden die in einer CCU bzw. bei der CLX-Version die in den Gateway-CCUs angelernten Geräte in die CL-Software importiert.

Virtuelle Kanäle werden nicht automatisch importiert weil meistens keine oder nur wenige benutzt werden und die Vielzahl der Objekte die Übersichtlichkeit beeinträchtigt.

Wenn ein virtueller Kanal der CCU benutzt werden soll, so kann dieser als neues Objekt mit dem Typ ***VirtCCUTaste*** angelegt werden.
In diesem Objekt kann dann der verwendete Bus (HmIP. HM-RF oder HMWired) und die Adresse eingetragen werden.

1.10.5. Software

Software

1.10.5.1. Compilieren und starten

Compilieren und starten

Mit diesem Menüpunkt wird eine neue Ausführungsdatei mit den aktuellen Konfigurationen erstellt und ausgeführt.

1.10.5.2. Neustart

Neustart

Mit diesem Menüpunkt wird die aktuelle Ausführung beendet und die Exec-Engine mit der aktuellen Ausführungsdatei neu gestartet.

1.10.5.3. Compilieren

Compilieren

Mit diesem Menüpunkt wird eine neue Ausführungsdatei mit den aktuellen Konfigurationen und Scripten erstellt, aber nicht gestartet. Diese Option ist sinnvoll um zu prüfen ob die aktuelle Programmierung fehlerfrei ist. Wenn die Erstellung der Ausführungsdatei erfolgreich war kann diese mit den Menüpunkt *Neustart* ausgeführt werden.

1.10.5.4. Status

Status

Mit diesem Menüpunkt wird die Ausführungsanzeige aufgerufen, in der Infos zum aktuellen Projekt und die Versionsstände der Software angezeigt werden.

1.10.5.5. Aktuelle Version

Aktuelle Version

Mit diesem Menüpunkt wird die aktuelle Version der Software angezeigt.

1.10.5.6. Lizenz

Lizenz

Unter diesem Menüpunkt wird die PLN (persönliche Lizenznummer) eingetragen. Bei einer gekauften Lizenz ist diese ohne zeitliche Begrenzung benutzbar.
Bei einer PLN für einen begrenzten Testzeitraum kann die Software mit der PLN bis zum Ablauf des durch die PLN festgelegten Zeitraums benutzbar.

1.11. Die Programmierung des Systems

Die Programmierung des Systems

Die Programmiersprache des Systems ist leicht zu erlernen und zur effektiven Programmierung des speziellen Anwendungsbereichs dieses Systems entwickelt. Diese Sprache ermöglicht es jedem Anwender ohne spezielle Programmierkenntnisse komplexe Anwendungsfunktionen zu realisieren. Für jedes Objekt kann ein Script erstellt werden, welches durch die Änderung von Zuständen bzw. Werten die mit den Objekten verbundenen Aktoren steuert. Zustände und Werte von Objekten und Variablen sowie Zeiten können mit vielfältigen Bedingungen abgefragt und durch Zuweisungen verändert werden.

Grundsätzlich kann man zwischen Anweisungen und Funktionen einer Programmiersprache unterscheiden.
Anweisungen beschreiben in der Programmzeile was getan werden soll.
Funktionen liefern z.B. für eine Zuweisung oder eine Bedingungsabfrage Werte zurück.

Scripte von Sensor-Objekten werden normalerweise immer ausgeführt, wenn eine Meldung von dem Sensor empfangen wird, beispielsweise also wenn eine Taste einer Fernbedienung gedrückt wird oder die aktuelle Temperatur eines Temperatursensors empfangen wird. Dazu wird beim Anlegen eines Sensor-Objekts standardmässig automatisch die Checkbox *Ausführen bei Empfang* aktiviert.

Scripte von Aktoren werden üblicherweise immer in einem bestimmten Zeitintervall aktiviert, beispielsweise um den Aktor nach einer bestimmten Zeit auszuschalten oder zu bestimmten Situationen und Abhängigkeiten von den Zuständen mehrerer anderer Objekten oder Zeiten zu steuern. Nach Möglichkeit sollte das gewählte Zeitintervall für die Ausführung nicht kleiner als nötig gewählt werden. Üblicherweise reicht eine Minute, wenn jedoch Uhrzeitabfragen im Script vorkommen, bei denen keine ganzen Minuten sondern Sekunden abgefragt werden muss das Zeitintervall kleiner, normalerweise auf 5 Sekunden eingestellt werden. Bitte beachten Sie dabei, dass die normale Uhrzeit im 5-Sekundentakt aktualisiert wird, die Uhrzeitangabe in den Bedingungen also ebenfalls eine Uhrzeit im 5-Sekundenintervall sein muss.

In jeder Zeile eines Programms darf nur eine Anweisung stehen, Kommentarzeilen müssen mit 2 Schrägstrichen (//) beginnen, diese werden dann bei der Generierung des Codes ignoriert.

In den Beispielen sind Programmanweisungen eingerückt, dies ist nicht erforderlich, aber aufgrund besserer Übersichtlichkeit unbedingt empfehlenswert.

Wenn während der Code-Erstellung Fehler auftreten, wird der fehlerhafte Quelltext mit einer entsprechenden Fehlermeldung angezeigt.

Anweisungsteile, die bei der Erklärung der Syntax in < spitzen Klammern > stehen sind optional und können weggelassen werden.

Die Uhrzeit wird nicht jede Sekunde, sondern alle 5 Sekunden aktualisiert. Bei Zeitvergleichen ist also zu beachten, dass die Sekunden immer durch 5 teilbar sein müssen.

Wenn Schaltmodule eingeschaltet werden, kann die Einschaltdauer angegeben werden.

Die Elemente der Programmiersprache sind die

- ***Objekte*** mit ihren Zuständen, Werten und Variablen. Zustände und Werte von Objekten (und somit die zugehörige Hardware) können abgefragt und verändert werden
- ***Operatoren*** für logische Verknüpfungen und Vergleiche
- ***Schlüsselwörter***, die in der Programmiersprache definierte Bedeutungen haben
- ***Anweisungen***, die in der Programmzeile beschreiben was getan werden soll

- ***Funktionen***, die in der Programmzeile Werte liefern, z.B. für Vergleiche.

Objekte

Hardwareobjekte

sind Objekte, die bestimmter Hardware zugeordnet sind und zum Schalten, Messen, Regeln etc. benutzt werden. Man unterscheidet zwischen Sensoren und Aktoren.

Sensoren sind Eingabeelemente wie Taster von Fernbedienungen, Bewegungsmelder, Fensterkontakte, Temperatursensoren, Wassermelder, Rauchmelder usw. In den Scripts der Sensoren stehen üblicherweise Anweisungen um Aktionen auszuführen wenn eine Meldung von der zugehörigen Hardware empfangen wird. Eine Makroanweisung zu einem Taster könnte also z.B. einen Aktor schalten wird, das sieht z.B. so aus:

```
Stehlampe umschalten
```

Aktoren sind Empfangsmodule, an die Endgeräte wie Beleuchtung, Rollläden, Schalter, Markisen etc. angeschlossen werden. Wenn Scripts von Aktoren benutzt werden, werden diese normalerweise in bestimmten Zeitintervallen ausgeführt. So kann z.B. ein Licht immer nach einer bestimmten Zeitdauer ausgeschaltet werden durch folgendes Script:

```
wenn Badventilator eingeschaltet und
  Schaltdauer(Badventilator) groesser "00:10:00" dann
  Badventilator ausschalten
endewenn
```

Virtuelle Objekte

sind Objekte, die keiner Hardware zugeordnet und z.B. nur zur Bildschirm-Ein- oder Ausgabe benutzt werden.

Im Fenster Einstellungen können über den Button "Neues Objekt" virtuelle Objekte erstellt werden. Dabei wird der Name, die Bezeichnung und der Typ festgelegt: z.B. Name: Anzeige, Bezeichnung: Anzeige, Typ: Zeichen.

In einem Script kann dem virtuelle Objekt Anzeige dann ein Wert zugewiesen werden, z.B:

Anzeige:="aktuelle Temperatur ist " + WandthermostatWohnzimmer.Temperatur+" Grad C"

Variablen

Variablen dienen der Zwischenspeicherung von Werten. Variablen werden wie die Objektprogramme auf der Seite Programmierung der Objektdefinition definiert. Einer Variablen kann im Objektprogramm ein Wert zugewiesen werden. Dieser Wert kann der Inhalt einer anderen Variablen, ein Objektzustand oder eine Konstante sein. Die in einem Objektprogramm definierten Variablen können von allen anderen Objektprogrammen benutzt werden, wenn der Name des Objekts, in dem die Variable definiert wurde, der Variablen getrennt durch einen Punkt vorangestellt wird.

Beispiel:
Anzeige:=RaumthermostatWohnen.Temperatur

Für jede Variable muss ein Typ festgelegt werden. Es können die gleichen Typen wie für Objekte verwendet werden.

Typen sind z.B.: Licht, Schalter, Zeichen, Zahl

Zusätzlich zu diesen Typen können für Variablen auch Zeittypen (Uhrzeit, Datum) benutzt werden.

Bei Zuweisungen zwischen nicht gleichen Typen erfolgt die Zuweisung entsprechend der Reihenfolge der für diesen Typ definierten Zustände, unabhängig von der Bezeichnung des Zustands. Intern werden Zustande als Zahl entsprechend der Reihenfolge in der Definition des Typs behandelt. Bei Zuweisungen wird diese Zahl zugewiesen.

Es gibt eine vordefinierte Variable des Typs Uhrzeit für jedes Objekt. Der Name der Variablen ist CT. In dieser Variablen wird die jeweils letzte Änderungszeit eines Objekts gespeichert. Mit der Funktion STOPPUHR(Objektname.CT) kann ermittelt werden wie lange ein Objekt sich in seinem aktuellen Zustand befindet.

Bitte beachten:
Wenn Sie eine Variable vom Typ Zahl definieren, gibt es zwei unterschiedliche Typen von Zahlen:

Zahlen ohne Komma (Ganzzahl) und Zahlen mit Komma (Gleitkommazahl). Von welchem Typ die Zahl ist, wird durch den Startwert festgelegt. Wird als Startwert eine Zahl mit Komma angegeben (z.B. 1,0) so wird die Variable als Gleitkommazahl behandelt, andernfalls als Ganzzahl.

Es macht bei Rechenoperationen und Zuweisungen einen großen Unterschied wie eine Zahl definiert ist.

Wenn einer Ganzzahl eine Gleitkommazahl zugewiesen wird, werden die Kommastellen einfach abgeschnitten.

Wenn eine Rechenoperation nicht das gewünschte Ergebnis liefert, prüfen Sie wie die in der Anweisung verwendeten Zahlen definiert sind.

Beispiel:
Die Minimal und Maximal-Temperatur sollen täglich auf einer Anzeige ausgegeben werden.

Im Objektprogramm des Objekts TempAussen (das Thermometer im Garten) sind die Variablen MIN und MAX als Zahl definiert. Das Script des Objekts TempAussen lautet:

```
wenn TempWG > MAX dann
  MAX:=TempWG
  endewenn

wenn TempWG < MIN dann
  MIN:=TempWG
  endewenn
```

Zuweisungen

Zustände bzw. Werte von Objekten oder Variablen können durch Zuweisungen geändert werden.
Die grundsätzliche Form einer Zuweisung im Programm ist
Ziel := Quelle
wobei Ziel ein Objekt oder eine Variable sein kann. Quelle kann ein Objekt, eine Variable oder eine Konstante sein kann. Bei Zuweisungen von Werten unterschiedlichen Typs wird soweit möglich automatisch eine Konvertierung entsprechend des Typs des Ziels vorgenommen.

Bei Zuweisungen zwischen nicht gleichen selbst definierten Typen erfolgt die Zuweisung entsprechend der Reihenfolge der für diesen Typ definierten Zustände, unabhängig von der Bezeichnung des Zustands. Zum Beispiel muss bei der Definition neuer Typen mit den Zuständen AUS/AN darauf geachtet werden, dass der Zustand AUS immer zuerst definiert wird, damit bei Zuweisung an andere AUS/AN - Objekte der richtige Zustand übertragen wird.

Bei der Verwendung von Schlüsselwörtern wie einschalten bei Zuweisungen oder ausgeschaltet in Wenn-Bedingungen ist es erforderlich, dass AUS immer als erster Zustand definiert wird, da intern mit der numerischen Reihenfolge der Zustände gearbeitet wird. Durch diese Verfahrensweise ist es möglich, auch Zuweisungen zwischen Typen mit unterschiedlicher Bezeichnung für gleiche Zustände durchführen zu können. Bei der Definition der Zustände von Fenstern und Türen muss offen als erster Zustand und zu als zweiter Zustand definiert werden, damit die entsprechenden Schlüsselwörter bei Zuweisungen und Bedingungen benutzt werden können.

Beispiele für Zuweisungen:

```
LichtBad:=SchalterBad hat die gleiche Wirkung wie

LichtBad wie SchalterBad

Licht1Garten einschalten hat die gleiche Wirkung wie
```

```
Licht1Garten:=an oder Licht1Garten:=1
```

Die besonderen Schlüsselwörter AN und AUS können bei einer Zuweisung mit dem Zuweisungsoperator := ohne Hochkommas verwendet werden.

Wird einem Objekt ein anderer Zustand als Text zugewiesen, so muss dieser in Hochkommas gesetzt werden und genau der Schreibweise der Typdefinition entsprechen, auch Groß - / Kleinschreibung muss berücksichtigt werden.

Weitere Beispiele für Zuweisungen:

Objekt einschalten für Zeitdauer

als Zeitdauer kann die Zeit in Hochkommas oder als Variable angegeben werden.

```
Stehlampe einschalten für "00:30:00"
```

oder

```
Gartenlicht einschalten für Dauer1
```

Dauer1 muss dabei eine Variable vom Typ Uhr oder Zeichen sein., also z.B. so definiert worden sein:

```
vardef Zeichen:Dauer1 = 00:30:00
```

Rechenfunktion bei Zuweisungen numerischer Werte

Bei Zuweisungen an ein Objekt bzw. eine Variable des Typs Zahl können die vier Grundrechenarten benutzt werden. Damit ist es z.B. möglich Eingabewerte von Sensoren für die weitere Verarbeitung zu verändern.
Beispiel:

```
NeuTemp:=AltTemp - 15
```

Rechenfunktion bei Zuweisungen von Uhrzeiten

Uhrzeiten können mit Hilfe der Operatoren + und - addiert bzw. subtrahiert werden.
Beispiel:

```
Aktionszeit:= Sonnenaufgang + "00:15:00"
```

Rechenfunktion bei Zuweisungen von Datumswerten

Es ist auch möglich ein Datum durch eine Rechenoperation zu ermitteln.
Das Datum 11 Tage nach dem aktuellen Tag kann man z.B. ermitteln mit der Anweisung:

```
Zukunft:=Datum+11
```

wobei die Variable *Zukunft* vom Typ Datum sein muss.

Verknüpfungsfunktion bei Zuweisungen von Texten

Mit Hilfe von Zuweisungen können zwei oder mehr Zeichenketten verbunden werden. Wenn in der Anweisung Operanden eines anderen Typs als Zeichenkette benutzt wird, werden diese soweit möglich automatisch konvertiert.

Beispiel:

```
Anzeige:="Licht in der Küche ist " + LichtKueche
```

Wenn das Objekt *LichtKueche* eingeschaltet ist erscheint in der Anzeige
Licht in der Küche ist an

Bei der Anweisung

```
AnzeigeBad:=Uhrzeit + " Uhr"
```

erschaint in der Anzeige:
22:30:00 Uhr

wenn - Bedingungen

In Wenn-Anweisungen wird aufgrund von Bedingungen entschieden, welche weiteren Anweisungen ausgeführt werden. Eine Bedingung ist entweder WAHR oder FALSCH.

Vergleichsoperatoren

= oder gleich

<> oder ungleich

< oder kleiner

> oder groesser

<= (kleiner oder gleich)

>= (größer oder gleich)

Logische Operatoren

UND (im Sinne von und zugleich), **ODER** (im Sinne von oder auch),

NICHT (zur Negation der nachfolgenden Bedingung)

Bedingungen sind folgendermaßen aufgebaut (Wörter in <>-Klammern sind optional):

```
WENN <NICHT> Bedingung <UND/ODER> Bedingung DANN
  Anweisungen
<SONST>
  Anweisungen
ENDEWENN
```

Die Wenn-Anweisung ist eine Anweisung, die sich über mehrere Zeilen erstreckt.

Mit der WENN-Anweisung ist es möglich, den weiteren Programmablauf von einer oder mehreren Bedingungen abhängig zu machen. Wenn-Anweisungen können auch verschachtelt werden, d.h. zwischen dem wenn und dem endewenn (bzw. SONST) können weitere Wenn-Anweisungen stehen. Jede Wenn-Anweisung muss mit einer endewenn-Anweisung beendet werden, ansonsten wird bei der Code-Generierung eine entsprechende Fehlermeldung ausgegeben.

Beispiel:

```
WENN LichtBad eingeschaltet UND
           SCHALTDAUER(LichtBad) groesser "00:00:30" DANN
  Ventilator einschalten
  ENDEWENN
```

Vor jeder Bedingung kann ein **NICHT** gesetzt werden, dann wird die Anweisung hinter **dann** ausgeführt wenn die Bedingung nicht zutrifft.
Bitte beachten:
Das Wort **NICHT** muss vor der eigentlichen Bedingung stehen, es darf nicht in der Bedingung stehen.
Beispiel:
Falsch wäre die umgangsprachliche Formulierung:

```
wenn LichtBad NICHT ausgeschaltet oder Tag NICHT=“Montag“ dann
```

es würde ein Syntaxfehler angezeigt.
Richtig ist:

```
wenn NICHT LichtBad ausgeschaltet oder NICHT Tag=“Montag“ dann
```

Bitte beachten:
Häufige Syntaxfehler in wenn-Anweisungen sind, dass das Wort **dann** vergessen wird oder dass die Anweisung nicht mit einem **endewenn** abgeschlossen wird.

Da wenn-Anweisungen sich immer über mehrere Zeilen erstrecken, kann die Zeile für einen Syntaxfehler oft nicht bestimmt werden. Wenn in einem Script mit einer wenn-Anweisung ein Syntaxfehler ohne Fehlerbeschreibung auftritt, prüfen Sie alle Elemente der wenn-Anweisung um den Fehler zu finden.

Beispiel einer WENN-Anweisung:

```
WENN Temperatur < 21,5 DANN
```

Wenn die Operanden unterschiedlichen Typs sind, findet soweit möglich eine automatische Konvertierung statt.
Wenn einer der Operanden eine Konstante ist, muss diese als Operand2 stehen, damit eine korrekte Konvertierung durchgeführt werden kann.

Beispiel:
Richtig:
```
wenn Uhrzeit = "15:15:00" dann
```

Falsch:
```
wenn "15:15:00" = Uhrzeit dann
```

Besondere Vergleichsbedingungen:

```
wenn Objekt eingeschaltet dann
```
hat die gleiche Wirkung wie
```
wenn Objekt = 1 dann
```
oder auch
```
wenn Objekt = "an" dann
```

Schlüsselwörter für Bedingungen

Für Schalter: *eingeschaltet, ausgeschaltet*
Für Türen und Fenster: *geoeffnet, geschlossen*

Beispiel:
```
wenn Fenster geoeffnet dann
```

Zeit-Vergleiche

Bei Zeitvergleichen mit Uhrzeiten ist zu beachten, dass die Uhrzeit im 5-Sekunden-Takt aktualisiert wird, die Sekunden der Uhrzeit bei der Prüfung auf Zeitgleichheit also immer durch 5 teilbar sein müssen.

```
wenn Uhrzeit = "HH:MM:SS" dann
```

bei dieser Bedingung ist zu beachten, dass die Uhrzeit in Hochkommas gesetzt werden muss.
Beispiel:
```
wenn Uhrzeit = "14:30:00" dann
```

Bei einem Datum:

```
wenn Datum = "TT.MM.JJJJ" dann
```

bei dieser Bedingung ist zu beachten, dass das Datum in Hochkommas gesetzt wird und die Jahreszahl

4-stellig angegeben wird.
Beispiel:
```
wenn Datum = "24.12.2023" dann
```

Bei Wochentagen:
```
wenn Wochentag = "Sonntag" dann
```

Der Wochentag wird in Hochkommas gesetzt und muss mit einem Großbuchstaben und nachfolgenden Kleinbuchstaben geschrieben werden.

Bei Prüfung auf den Tag eines Datums.
```
wenn Monatstag = 29 dann
```

Bei Monaten, die Anweisung

```
wenn Monat > 4 und Monat < 10 dann
```

bewirkt, dass die folgenden Anweisungen nur zwischen Mai und September ausgeführt werden.

Vergleichsoperator =* für Vergleiche mit Jokerzeichen

Bei Vergleichen von Uhrzeit und Datum mit einer Konstanten können auch Jokerzeichen verwendet werden. Als Vergleichsoperator muss **=*** verwendet werden. Operand2 muss eine Konstante in Hochkommas sein.

Um Anweisungen jede volle und halbe Stunde auszuführen:
```
wenn Uhrzeit =* "**:00:00" oder Uhrzeit =* "**:30:00" dann
```

Um Anweisungen immer am Ersten eines Monats auszuführen:
```
wenn Datum =* "01.**.****" dann
```

Bedingung für Zeiträume

Manchmal ist es erforderlich Aktionen nur innerhalb bestimmter Zeiträume auszuführen.
Das ist möglich mit der Bedingungsanweisung :

wenn Uhrzeit/Datum zwischen "Zeitkonstante" und "Zeitkonstante" dann

Beispiel:
```
wenn Uhrzeit zwischen "09:00:00" und "11:30:00" dann
```

Vergleichsoperator =+ für Vergleiche mit Wochenmaske

Mit diesem Vergleichsoperator ist es möglich, Anweisungen nur an bestimmten Wochentagen ausführen zu lassen. Als Operand2 muss eine 7-stellige Konstante bestehend aus Nullen und Einsen in Hochkommas verwendet werden. Jede Stelle der Konstanten steht für einen Wochentag, beginnend mit Sonntag. Die Bedingung ist wahr wenn an der Stelle des aktuellen Wochentags eine Eins steht.
Beispiel:
Es soll geprüft werden, ob der aktuelle Tag ein Freitag, Samstag oder Sonntag ist, die Anwesung sieht so aus:

```
wenn Wochentag =+ "1000011" dann
```

1.11.1. Skripte

Skripte

Skripte sind Programme, die der Benutzer erstellen kann um die individuellen Anwendungsfunktionen seines Systems zu programmieren.
Jedes Objekt eines Moduls kann ein Script beinhalten. Es können auch Objekte ohne Hardwarebezug (virtuelle Objekte) erstellt werden, für die ein Script geschrieben wird.
Scripte können durch Empfang vom Sensor eines Objekts, durch eine Änderung des Objektwerts oder in bestimmten Zeitintervallen aktiviert werden.
Welche Option der Ausführungsaktivierung benutzt werden soll, wird auf der Seite zur Verwaltung von Objekten festgelegt.

Scripte können in bestimmten Zeitintervallen, durch den Empfang vom zugeordneten Hardwaresensor, durch Aufruf aus anderen Scripts oder durch Bedienung über eine Visualisierung aktiviert werden.

In Scripten können die Zustände bzw. Werte des eigenen und aller anderen Objekte abfragen und ändern. Zustände von Objekten und den diesen zugeordneten Hardwaremodulen werden automatisch abgeglichen, d.h. wenn ein zugeordneter Sensor eine Zustandsänderung meldet, ändert sich der Zustand des Objekts. Wenn das Programm einen Objektzustand ändert wird eine Meldung an den zugeordneten Aktor (Empfänger) geschickt.

In einem Script kann die Funktion SELBST verwendet werden, um das aktuelle Objekt, das das Script beinhaltet zu benutzen.

Beispiel um eine Beleuchtung immer nach maximal einer Stunde auszuschalten in einem Script, das im Zeitintervall von 1 Minute aktiviert wird:

```
wenn SELBST eingeschaltet und Schaltdauer(SELBST) > "01:00:00" dann
  SELBST ausschalten
endewenn
```

Es gibt zwei spezielle Scripttypen, die jeweils beim Start und bei der Beendigung der ExecEngine ausgeführt werden.

Scripts, deren Namen mit "INIT_" beginnen werden beim Start der ExecEngine vor allen anderen Scripts ausgeführt.

Scripts deren Namen mit "END_" beginnen werden beim Beenden der ExecEngine ausgeführt. Das geschieht allerdings nur wenn die ExecEngine "normal" beendet wird, d.h. durch Compilieren und Übertragen eines Projekts oder durch das Kontrollprogramm ExecEngineWin.
Da das Starten der END-Scripts erst nach Beendigung der Ausführungssteuerung beginnt, dürfen in END-Scripts keine WARTE-Anweisungnen verwendet werden. Bei Ausführung einer WARTE-Anweisung wird das End-Script endgültig beendet, da es nach Ablauf der Wartzezeit nicht mehr durch die Ausführungssteuerung aktiviert werden kann. Das gilt nicht wenn das End-Script im normalen Betrieb z.B. von einem anderen Script aufgerufen wird.
Weiterhin dürfen in einem END-Script keine Aktionen gestartet werden, die im Hintergrund ablaufen, wie das Versenden von Mails oder ein http-Request mit GETSITE.
Diese Hintergrundaktivitäten können dann eventuell nicht mehr komplett ausgeführt werden.

Die END-Scripts werden nicht ausgeführt wenn die ExecEngine nicht über ExecEngineWin oder die Übertragung einer Ausführungsdatei beendet wird, also z.B. durch einen Neustart der Zentrale, einen Betriebssystembefehl in der Konsole oder Webserverfunktionen.

Kommentare und Bemerkungen in Scripts beginnen mit zwei Schrägstrichen //, diese können am Anfang einer Zeile oder auch hinter einer Anweisung benutzt werden.
Beispiel:
// jetzt folgt die Berechnung
DWert1:=(a + b) / 2   // Durchschnittswert

Üblicherweise sind Skripte nicht länger als 0,1 bis 0,2 Sekunden aktiv. Bei sehr langen Skripten, die mehr machen als nur einige Aktoren zu schalten und mehrere hundert Zeilen haben kann die Ausführungszeit aber auch deutlich länger sein. Eine zu lange Ausführungszeit sollte aber möglichst vermieden werden, um die Reaktionszeiten zum Schalten von Aktoren bei der Bedienung nicht zu hoch werden zu lassen.
Daher wird während der Ausführung überprüft, ob ein Skript aussergewöhnlich lange läuft und in dem Fall ein Eintrag in das Systemlog geschrieben. Das passiert standardmässig wenn ein Skript länger als 2 Sekunden aktiv ist. Dann sollte das Skript ggfs. entsprechend verändert werden bzw. die Funktionen auf mehrere Skripte aufgeteilt werden. Falls das nicht gewünscht ist, kann die Standardzeit, ab der eine solche Warnung geschrieben wird verändert werden.
Dazu wird ein Objekt des Typs Zahl mit dem Namen SCRIPTTIMEOUT erstellt und mit einem Startwert versehen, der die maximale Zeit in Sekunden angibt, die ein Skript aktiv sein darf bevor eine Meldung in das Systemlog geschrieben wird.

1.11.2. Variablen

Variablen

Variablen dienen der Zwischenspeicherung von Werten. Variablen werden mit der Anweisung VARDEF in Scripten definiert. Einer Variablen kann im Script ein Wert zugewiesen werden. Dieser Wert kann der Inhalt einer anderen Variablen, ein Objektzustand oder eine Konstante sein. Die in einem Script definierten Variablen können von allen anderen Scripten in anderen Objekten benutzt werden. Wenn der Name des Objekts, in dem die Variable definiert wurde der Variablen getrennt durch einen Punkt vorangestellt wird.
Der Name einer Variablen kann die Buchstaben A-z, Ziffern und den Unterstrich beinhalten. Er muss mit einem Buchstaben beginnen, landesspezifische Buchstaben und Sonderzeichen sind nicht erlaubt (Beispiel: Var_99).

Für jede Variable muss ein Typ festgelegt werden. Es können dieselben Typen wie für Objekte verwendet werden. Zusätzlich zu diesen Typen können für Variablen auch Zeittypen (Uhrzeit, Datum) benutzt werden.
Bei Zuweisungen zwischen nicht gleichen Typen erfolgt die Zuweisung entsprechend der Reihenfolge der für diesen Typ definierten Zustände, unabhängig von der Bezeichnung des Zustands. Intern werden Zustande als Zahl entsprechend der Reihenfolge in der Definition des Typs behandelt. Bei Zuweisungen wird diese Zahl zugewiesen.

Es gibt eine vordefinierte Variable des Typs Uhrzeit für jedes Objekt. Der Name der Variablen ist **CT** (für **C**hange**T**ime). In dieser Variablen wird die jeweils letzte Änderungszeit eines Objekts gespeichert. Mit der Funktion STOPPUHR(Objektname.CT) kann ermittelt werden wie lange ein Objekt sich in seinem aktuellen Zustand befindet. Die aktuelle Schaltdauer eines Objekts kann auch mit der Funktion SCHALTDAUER(Objektname) ermittelt werden.
Bitte beachten Sie:
Wenn Sie eine Variable vom Typ Zahl definieren, gibt es zwei unterschiedliche Typen von Zahlen: Zahlen mit Komma (Gleitkommazahlen) und Zahlen ohne Komma. Von welchem Typ die Zahl ist, wird durch den Startwert festgelegt. Wird als Startwert eine Zahl mit Komma angegeben (z.B. 1,0) so wird die Variable als Zahl mit Kommastellen behandelt, andernfalls als Zahl ohne Kommastelle. Als Dezimaltrennzeichen kann anstelle eines Kommas auch ein Punkt (amerikanische Schreibweise für Gleitkommazahlen) verwendet werden.
Es macht bei Rechenoperationen und Zuweisungen einen grossen Unterschied wie eine Zahl definiert ist. Wenn einer Zahl ohne Kommastellen eine Zahl mit Kommastellen zugewiesen wird, werden die Kommastellen einfach abgeschnitten.
Wenn eine Rechenoperation nicht das gewünschte Ergebnis liefert, prüfen Sie wie die in der Anweisung verwendeten Zahlen definiert sind.

Variablen werden in Skripten mit folgender Anweisung definiert:
vardef Typ:Variablenname = Startwert
Beispiele:

```
vardef Zeichen:Dauer = 00:05:00
vardef Licht:Stehlampe=aus
vardef Zahl:Temperatur = 21,5
```

Es gibt vordefinierte Variablen, die verwendet werden können. Jedes Objekt hat beispielsweise eine Variable vom Typ Zeit mit dem Namen **CT**, in der die letzte Schaltzeit des Objekt abgespeichert ist. Um die aktuelle Schaltdauer eines Objekts zu ermitteln kann man folgende Anweisung verwenden:

```
Anzeige:=Schaltdauer(Lampe.CT)
```

Ein Beispiel zur Verwendung von Variablen :
Die minimal und maximal-Temperatur sollen jede Stunde auf einer Anzeige ausgegeben werden.
Im Script des Objekts TempWG sind die Variablen MIN und MAX als Zahl definiert.
Das Script des Objekts TempWG:

```
wenn TempWG > MAX dann
  MAX:=TempWG
endewenn
```

```
wenn TempWG < MIN dann
  MIN:=TempWG
endewenn
```

Variablen sind dem Objekt zugeordnet, in dessen Script es definiert wird, kann jedoch auch in anderen Scripten verwendet werden, wenn der Name der Variablen qualifiziert, d.h. mit vorangestelltem zugehörigem Objektnamen, von der Variablen durch einen Punkt getrennt, angegeben wird.

Wenn Beispielsweise im Script *Gartenlicht* die Variable *Autoschalter* definiert ist, so kann diese in anderen Scripten unter dem Namen *Gartenlicht.Autoschalter* verwendet werden.

1.11.3. Zuweisungen

Zuweisungen und Rechenfunktionen

Zustände bzw. Werte von Objekten oder Variablen können durch Zuweisungen geändert werden.
Die grundsätzliche Form einer Zuweisung im Programm ist
Ziel := Quelle
wobei Ziel ein Objekt oder eine Variable sein kann. Quelle kann ein Objekt, eine Variable oder eine Konstante sein kann.

Bei Zuweisungen von Werten unterschiedlichen Typs wird soweit möglich automatisch eine Konvertierung entsprechend des Typs des Ziels vorgenommen, d.h. das Ergebnis hat den Typ der Zielvariable-
Wenn die Zielvariable z.B. eine Uhrzeit ist und es werden zwei Zeichen-Variablen mit Werten im Zeitformat addiert ist das Ergebnis die Summe der Zeiten, ist die Zielvariable aber eine Zeichenvariable werden nur die Texte aneinandergehängt.
Beispiele
```
ZeichenVariable1 := "14:00:00"
Zeitvariable1 := "03:10:00"
```

Bei einer Zeichenvariablen als Zielvariable:
`ZeichenvariableX := ZeichenVariable1 + Zeitvariable1` ist das Ergebnis "14:00:0003:10:00"

Bei einer Variable vom Typ Uhrzeit als Zielvariable:
`ZeitvariableX := ZeichenVariable1 + Zeitvariable1` ist das Ergebnis "17:10:00"

Bitte beachten Sie, dass Zahlen mit bzw. ohne Kommastellen intern in unterschiedlichen Formaten dargestellt werden und es daher zu Problemen kommen kann wenn diese unterschiedlichen Zahlentypen einander zugewiesen werden. Bei Zuweisungen von Konstanten an Zahlenvariablen mit Nachkommastellen sollte die Konstante immer ein Komma beinhalten damit diese als Gleitkommazahl gespeichert wird, dabei kann hinter dem Komma natürlich auch eine Null stehen.
Es wird zwar bei der Ausführung versucht solche Probleme automatisch auszuschalten, wenn bestimmte Grössen überschritten werden funktioniert das aber nicht mehr (wenn eine Integer-Zahl grösser als 2147483647 einer Gleitkommavariablen zugewiesen wird, wird diese Zahl als negative Zahl interpretiert).

Bei Zuweisungen zwischen nicht gleichen selbstdefinierten Typen erfolgt die Zuweisung entsprechend der Reihenfolge der für diesen Typ definierten Zustände, unabhängig von der Bezeichnung des Zustands. Zum Beispiel muss bei der Definition neuer Typen mit den Zuständen AUS/AN darauf geachtet werden, dass der Zustand AUS immer zuerst definiert wird, damit bei Zuweisung an andere AUS/AN-Objekte der richtige Zustand übertragen wird.

Bei der Zuweisung von Zuständen können die Zustandstexte in Hochkomma oder der Index eines Zustands verwendet werden, wobei der erste Zustand den Index 0 hat. Bei Verwendung von Zustandstexten muss das Wort exakt so geschrieben werden wie der definierte Zustand in der Typdefinition, auch Gross/Kleinschreibung muss berücksichtigt werden.
Wenn eine Lampe also den Objektytp Licht mit den Zuständen "aus" und "an" hat kann folgende Anweisung zum Ausschalten verwendet werden:
`Lampe:=`**`"aus"`**
oder auch
`Lampe:=0`

Bitte unbedingt beachten:
Wenn bei Zuweisungen ein Text für den Zustand verwendet wird, muss dieser (auch in der Schreibweise) genau einem Zustand des Objekttyps entsprechen, sonst wird die Zuweisung nicht ausgeführt und der alte Zustand bleibt erhalten.

Für bestimmte Objektypen können auch umgangssprachliche Begriffe als Schlüsselwörter

verwendet werden.
Bei der Verwendung von Schlüsselwörtern wie *einschalten* bei Zuweisungen oder *ausgeschaltet* in wenn-Bedingungen ist es erforderlich, dass AUS immer als erster Zustand definiert wird, da intern mit der numerischen Reihenfolge der Zustände gearbeitet wird. Durch diese Verfahrensweise ist es möglich, auch Zuweisungen zwischen Typen mit unterschiedlicher Bezeichnung für gleiche Zustände durchführen zu können. Bei der Definition der Zustände von Fenstern und Türen muss *offen* als erster Zustand und *zu* als zweiter Zustand definiert werden, damit die entsprechenden Schlüsselwörter bei Zuweisungen und Bedingungen benutzt werden können.
Beispiele für Zuweisungen:
LichtBad:=SchalterBad
hat die gleiche Wirkung wie
LichtBad wie SchalterBad

Gartenlicht einschalten
hat die gleiche Wirkung wie
GartenLicht:=1

Wird einem Objekt ein Zustand als Text zugewiesen, so muss dieser in Hochkommas gesetzt werden und genau der Schreibweise der Typdefinition entsprechen, auch Gross/Kleinschreibung muss berücksichtigt werden.
Beispiel:
Anstatt
Gartenlicht einschalten
könnte man auch die Anweisung
Gartenlicht:=“an“
verwenden, das Wort **an** muss genau wie in der Typdefinition geschrieben sein und in Hochkommas gesetzt werden.
Eine besondere Anweisung zum Einschalten bei Schaltaktoren ist die Anweisung

Objekt **einschalten für** *Zeitdauer*

als Zeitdauer kann die Zeit in Hochkommas oder als Variable angegeben werden.

Beispiel:

Stehlampe einschalten für "00:30:00"
oder

Dauer1:="00:10:00"
Gartenlicht einschalten für Dauer1

wobei Dauer1 eine Variable vom Typ *Uhr* oder *Zeichen* sein muss.

Ein weiteres wichtiges Schlüsselwort ist **ausschalten** um einen Schaltaktor auszuschalten.

Die Art der Rechenoperation bei Zuweisungen wird durch den Typ der Zielvariablen bestimmt. Wenn der Zieltyp vom Typ Zeichen ist, werden die Zeichen in der Zuweisung als Zeichenkette aneinander gereiht.
Wenn die Zielvariable vom Typ Zahl ist, wird eine normale Rechenoperation vorgenommen falls dies mit den angegebenen Operanden und Operatoren möglich ist, dabei können auch Klammenr gemäss den üblichen Rechenregeln verwendet werden.

Rechenfunktionen für Zahlen
Es können die vier Grunrechenarten benutzt werden und die Modulo-Operation. Der Operator für die Modulo-Operation ist das Prozentzeichen %. Das Ergebnis der Modulo-Operation ist der ganzzahlige Rest einer Division, wenn einer der Operanden eine Zahl mit Nachkommastellen ist werden diese abgeschnitten.
Weiterhin werden folgende mathematischen Funktionen unterstützt:
LN = natürlicher Logarithmus
EXP = Exponentialfunktion (Umkehrfunktion von LN)
LOG = dekadischer Logarithums (zur Basis 10)
SIN = Sinus
ASIN = Arcus Sinus
COS = Cosinus
ACOS = Arcus Cosinus

TAN = Tangens
ATAN = Arcus Tangens
SQRT = Quadratwurzel
ABS = Absolutwert

Beispiel:
Durchschnitt:=Summe / Anzahl
Var1:=Wert1 * (Wert2 + Wert3)
QW:=SQRT(Var1)

Bitte unbedingt beachten:
Innerhalb der Klammer einer Funktion dürfen keine Rechenoperatioenn stehen, nur eine Konstante oder eine Variable, falls ger Parameter berechnet werden muss, kann er vorher in einer Variablen gespeichert werden..

Rechenfunktion bei Zuweisungen von Uhrzeiten
Uhrzeiten können mit Hilfe der Operatoren **+** und **-** addiert bzw. subtrahiert werden.
Beispiel:
AktuelleWeckzeit:=Weckzeit + "00:30:00"

Anstatt von Uhrzeiten in Hochkommas kann auch die Zeit in Sekunden angegeben werden.
Diese Anweisung addiert 2 Minuten zu einer Uhrzeit**:**
ZeitLicht1:=ZeitLicht1 + 120

Rechenfunktion bei Zuweisungen von Datumswerten

Es ist auch möglich ein Datum durch eine Rechenoperation zu ermitteln.
Das Datum 11 Tage nach dem aktuellen Tag kann man z.B. einfach ermitteln mit der Anweisung:

Zukunft:=Datum+11

wobei die Variable Zukunft natürlich vom Typ Datum sein muss.

Verknüpfungsfunktion bei Zuweisungen von Texten
Mit Hilfe von Zuweisungen können zwei oder mehr Zeichenketten verbunden werden. Wenn in der Anweisung Operanden eines anderen Typs als Zeichenkette benutzt wird, werden diese soweit möglich automatisch konvertiert.
Beispiel:

AnzeigeKueche:="Garage ist "+Garagentor

Wenn das Objekt Garagentor die Typen *auf* und *zu*hat, und das Garagentor auf ist erscheint in der Anzeige ***Garage ist auf***

AnzeigeBad:=Uhrzeit+" Uhr"

Der Text in der Anzeige sieht dann so aus: ***22:30:00 Uhr***

Bitte beachten:
Wenn das Ziel der Wert oder Zustand eines Hardwaregeräts ist, muss der zugewiesene Wert im Wertebereich und Format sein, der für das Hardwaregerät gültig ist. Dies zu prüfen liegt in der Verantwortung des Anwenders, da eine solche Prüfung weder im Editor noch im Compiler vorgenommen werden kann weil die Werte vom jeweiligen Gerät und bei demselben Gerät teilweise von der im Gerät installierten Firmware abhängig sein kann.

1.11.4. Bedingungen

Bedingungen

Im Folgenden werden die Formen der Bedingungen in wenn-Anweisungen beschrieben, nicht die Wenn-Anweisung selber.
Details zu dieser Anweisung finden Sie im Kapitel wenn-Anweisungen.
In wenn-Anweisungen wird aufgrund von Bedingungen entschieden welche weiteren Anweisungen ausgeführt werden. Eine Bedingung ist entweder WAHR oder FALSCH. Bedingungen können mit UND bzw. ODER verknüpft werden.
Bedingungen sind folgendermaßen aufgebaut:
wenn *Operand1 Vergleichsoperator Operand2*
Beispiel:
wenn Temperatur kleiner 21,5

Wenn die Operanden unterschiedlichen Typs sind, findet soweit möglich eine automatische Konvertierung statt. Dazu ist es erforderlich, dass das in der Bedingung abgefragte Objekt bzw. die Variable ímmer *Operand1* in der Vergleichsabfrage ist, wie es der üblichen Logik einer Abfrage entspricht. Nur dann ist eine korrekte Konvertierung möglich. Wenn also für ein Objekt mit Zuständen eine Abfrage des Zustands in Textform (als Konstante oder Variable) erfolgen soll, muss der Text immer als zweiter Operand stehen. So kann der *Operand2* dann entsprechend des Typs des ersten Operanden in einen Zustand dieses Typs konvertiert werden. Dabei ist wichtig, dass der Text einem tatsächlich vorhandenen Zustand des Typs entspricht (auch Gross/Kleinschreibung), sonst ist eine Konvertierung nicht möglich und das Ergebnis des Vergleichs ist unbestimmt und somit falsch.
Wenn einer der Operanden eine Konstante ist muss diese immer *Operand2* sein damit eine Konvertierung auf den Typ des abzufragenden Objekts/Variablen durchgeführt werden kann.
Beispiel einer Uhrzeitabfrage:
Richtig: wenn Uhrzeit = "15:15:00" dann
Falsch: wenn "15:15:00" = Uhrzeit dann

Standard-Vergleichsoperatoren
= oder **gleich**
<> oder **ungleich**
< oder **kleiner**
> oder **groesser**
<= (kleiner oder gleich)
>= (grösser oder gleich)

Vergleichsbedingungen

Bei Vergleichen von Zuständen können die Zustandstexte in Hochkomma oder der Index eines Zustands verwendet werden, wobei der erste Zustand den Index 0 hat.
Die Abfrage ob eine Lampe ausgeschaltet ist kann also mit der Anweisung
```
wenn Lampe = "aus" dann
```
oder auch
```
wenn Lampe = 0 dann
```
durchgeführt werden.
Bitte unbedingt beachten:
Wenn bei Bedingungen Text für die Abfrage des Zustands verwendet wird muss dieser (auch in der Schreibweise) einem gültigen Zustand des Objekttyps entsprechen, sonst kann der intern verwendete Zustandsindex nicht ermittelt und die Bedingung kann nicht korrekt ausgewertet werden. Wenn Text verwendet wird, der keinem gültigen Zustand des Objekttyps entspricht ist das Ergebnis der Abfrage unbestimmt.

Schlüsselwörter in Vergleichsbedingungen

Bei Vergleichen können für bestimmte Objekttypen auch Schlüsselwörter benutzt werden.

wenn *Lampe* **eingeschaltet**
hat die gleiche Wirkung wie
wenn *Lampe* = **"an"**
bzw.
wenn *Lampe* = 1

wenn *Lampe* **ausgeschaltet**
hat die gleiche Wirkung wie
wenn *Lampe* = **"aus"**
bzw.
wenn *Lampe* = 0

weitere besondere Schlüsselwörter für Bedingungen sind
GEÖFFNET und **GESCHLOSSEN** für Tür/Fensterkontakte (**nicht für Rollladen, Jalousien, KeyMatic, Winmatic !**),
also z.B.:
wenn Fenster **geoeffnet** dann

Um bei einer Fernbedienungstaste festzustellen ob diese kurz oder lang gedrückt wurde kann z.B. folgende Abfrage verwendet werden:

```
WENN FBWohnenTaste2 = "lang" dann
  Stehlampe einschalten
SONST
  Stehlampe ausschalten
ENDEWENN
```

Zeit-Vergleiche
Bei Zeitvergleichen ist zu beachten, dass die Uhrzeit im 5-Sekunden-Takt aktualisiert wird, die Sekunden der Uhrzeit bei der Prüfung auf Zeitgleichheit also immer durch 5 teilbar sein müssen. Wenn Uhrzeitvergleiche verwendet werden, muss sichergestellt sein, dass das Makro zu dem Zeitpunkt zu dem die Bedingung zutrifft könnte auch ausgeführt wird. Wenn also die Vergleichszeit volle Minuten hat, muss das Zeitintervall für das Makro mindestens auf 1 x in der Minute gesetzt sein, wenn es eine durch 5 Sekunden teilbare Uhrzeit ist muss das Ausführungsintervall auf 5 Sekunden gesetzt werden.

wenn **Uhrzeit = "HH:MM:SS" .....**

bei dieser Bedingung ist zu beachten, dass die Uhrzeit in Hochkommas gesetzt wird. Weiterhin müssen die Sekunden durch 5 teilbar sein, da die Bedingung sonst nie zutrifft.

wenn **Datum = "TT.MM.JJJJ"**

bei dieser Bedingung ist zu beachten, dass das Datum in Hochkommas gesetzt wird und das Jahr vollständig, also 4-stellig mit Jahrtausend angegeben werden muss.

wenn **Wochentag =** "*Wochentag"*

Prüfung auf einen Wochentag. Der Wochentag wird in Hochkommas gesetzt und muss mit einem Großbuchstaben und nachfolgenden Kleinbuchstaben geschrieben werden.
Beispiel : **wenn Wochentag = "Freitag" dann**

wenn **Monat** Vergleichsoperator *Zahl*

Beispiel :

wenn Monat groesser 4 und Monat kleiner 10 dann

bewirkt, dass die folgenden Anweisungen nur zwischen Mai und September ausgeführt werden.
Vergleichsoperator =* für Vergleiche mit Jokerzeichen
Beim Vergleich von Uhrzeit und Datum mit einer Konstanten können auch Jokerzeichen verwendet werden. Als Vergleichsoperator muss dann =* verwendet werden. Operand2 muss eine Konstante in Hochkommas sein.
Beispiele:

Um Anweisungen jede volle und halbe Stunde auszuführen:

wenn **Uhrzeit =* "**:00:00"** oder **Uhrzeit =* "**:30:00"** dann
.....
Um Anweisungen immer am Ersten eines Monats auszuführen:

wenn **Datum =* "01.**.****"** dann

Bedingung für Bereiche
Mit diesem besonderen Bedingungstyp ist es möglich z.B. Aktionen nur innerhalb bestimmter Bereiche und Zeiträume auszuführen.

wenn *Prüfwert* **zwischen** *Startwert* **und** *Endwert* **dann**
.....
Beispiele für Zeiträume:

wenn Uhrzeit/Datum zwischen "*Zeitkonstante*" **und** "*Zeitkonstante*" **dann**

wenn Uhrzeit zwischen "23:00:00" und "05:00:00" dann

wenn Monat zwischen 5 und 9 dann
ist wahr von Mai bis September

wenn Monat zwischen 10 und 4 dann
ist wahr von Oktober bis April

wenn Monat zwischen 6 und 6 dann
ist nur im Juni wahr

Dieser Bedingungstyp kann nicht nur für Zeiträume, sondern grundsätzlich für alle Werte benutzt werden.

wenn Temperatur zwischen 18,0 und 21,0 dann
ist von 18,0 bis 21,0 Grad wahr

Bitte beachten - grundsätzlich gilt:
wenn der *Startwert* kleiner ist als der *Endwert* ist die Bedingung **wahr** wenn der *Prüfwert* im zu prüfenden Bereich liegt, also auch wenn er gleich mit dem *Start-* oder *Endwert* ist.
wenn der *Startwert* grösser ist als der *Endwert* ist die Bedingung **wahr** wenn der *Prüfwert* **nicht** zwischen *Startwert* und *Endwert* liegt.
Diese Verfahrensweise bietet die grösste Flexibilität und ergibt auch bei Bedingungen für Zeiträume mit Tageswechsel wie z.B. "*wenn Uhrzeit zwischen "23:00:00" und "05:00:00" dann*" das richtige Ergebnis, aber es muss genau darauf geachtet werden in welcher Reihenfolge Start- und Endwert angegeben werden.

Vergleichsoperator =+ für Vergleiche mit Wochenmaske
Mit diesem Vergleichsoperator ist es möglich Anweisungen nur an bestimmten Wochentagen ausführen zu lassen. Als Operand2 muß eine 7-stellige Konstante oder Variable vom Ty Zeichen bestehend aus Nullen und Einsen in Hochkommas verwendet werden. Jede Stelle steht für einen Wochentag, beginnend mit Sonntag. Die Bedingung ist wahr wenn an der Stelle des aktuellen Wochentags eine Eins steht.

Beispiel : Es soll geprüft werden, ob der aktuelle Tag ein Freitag, Samstag oder Sonntag ist.

wenn Wochentag =+ "1000011" dann

1.11.5. Anweisungen

Anweisungen

Auf den folgenden Seiten werden die Anweisungen unf Funktionen aufgeführt, die für die Programmierung zur Verfügung stehen.

Da von Anwendern, die mit Programmierungen vertraut sind, oft die Frage kommt, ob die Programmierung nicht wie bei Programmiersprachen üblich mit englischen Begriffen durchgeführt werden kann, hier eine Liste der möglichen englischen Anweisungen.
Ein Hinweis dazu noch:
In deutschen Scripten wird beim Schalten von Aktoren üblicherweise die Form z.B.
`MeinLicht einschalten`
benutzt. Da die Wortstellung im Englischen so nicht üblich ist, kann auch die Form
`Switchon MyLight`
benutzt werden.

Hier eine Tabelle der Anweisungen und Funktionen mit den Begriffen, die für die Programmierung in englischer Sprache benutzt werden können:

Für Abfragen / Wenn- Bedingungen

WENN	IF
SONST	ELSE
ENDEWENN	ENDIF
DANN	THEN
GLEICH	EQUAL
UNGLEICH	NOTEQUAL
IST	IS
KLEINER	LESSER
GROESSER	GREATER
KLEINERGLEICH	LESSEREQUAL
GROESSERGLEICH	GREATEREQUAL
ZWISCHEN	BETWEEN
ODER	OR
UND	AND
NICHT	NOT
AUSGESCHALTET	SWITCHEDOFF
EINGESCHALTET	SWITCHEDON
GEOEFFNET	OPENED
GESCHLOSSEN	CLOSED
ENTRIEGELT	UNLOCKED
VERRIEGELT	LOCKED

Für Zuweisungen

EINSCHALTEN	SWITCHON
AUSSCHALTEN	SWITCHOFF
UMSCHALTEN	TOGGLE
RUNTERFAHREN	MOVEDOWN
RAUFFAHREN	MOVEUP
SCHLIESSEN	CLOSE
OEFFNEN	OPEN
ENTRIEGELN	UNLOCK
VERRIEGELN	LOCK
TUEROEFFNEN	OPENDOOR

Weitere Anweisungen

ABBRECHEN	CANCEL
ABFRAGE	REQUESTDEVICE
ABSPIELEN	PLAY
AKTIVIEREN	ACTIVATE
ALLEABFRAGEN	REQUESTALLDEVICES
ALLEWERTESICHERN	SAVEALLVALUES
ANZEIGEN	DISPLAY

AUFRUFEN	CALL
DEAKTIVIEREN	DEACTIVATE
DIMMEN	DIM
ERLEDIGT	DONE
GEHEZU	GOTO
GRUPPENZUWEISUNG	SETGROUP
LESEINTERNETSEITE	GETSITE
LESEWERTEDATEI	READVALUEFILE
LOESCHEDATEI	DELETEFILE
LÖSCHEANZEIGE	CLEARDISPLAY
RAUFDIMMEN	DIMUP
RUNTERDIMMEN	DIMDOWN
SCHLIESSEDATEI	CLOSEFILE
SCHLIESSEDATEIEN	CLOSEFILES
SCHREIBEDATEI	WRITEFILE
SENDE	SENDDEVICE
SENDEMAIL	SENDMAIL
SENDEMAILANHANG	SENDMAILATTACH
SENDESMS	SENDSMS
SETZEHISTORYDIFFERENZ	SETHISTORYSENSIBILITY
SETZEKONFIG	SETCONGIG
SETZEWERT	SETVALUE
SETZEZEITTABELLENWERT	SETTIMETABVAL
SICHTBAR	VISIBLE
SPRACHAUSGABE	SPEECHOUTPUT
STARTE	RUN
STARTEPROGRAMM	STARTWIN
STARTPROGRAMM	EXECUTE
STARTUHR	STARTWATCH
STOPPDIMMEN	STOPDIM
STOPPE	STOPDEVICE
TEMPERATURABSENKUNG	TEMPREDUCTION
UNSICHTBAR	INVISIBLE
VERLASSEN	RETURN
WARTE	WAIT

Funktionen

AKTIVIERT	ISACTIVE
BATTERIELEER	LOWBAT
DATEIVORHANDEN	FILEEXISTS
DATUM	DATE
FEIERTAG	HOLIDAY
GESCHALTET	SWITCHED
HOLEWERT	GETVALUE
JAHR	YEAR
MINUTE	MINUTE
MONAT	MONTH
MONATSTAG	DAYOFMONTH
SCHALTDAUER	SWITCHDURATION
SEKUNDE	SECOND
SONNENAUFGANG	SUNRISE
SONNENUNTERGANG	SUNSET
STOPPUHR	STOPWATCH
STUNDE	HOUR
UHRZEIT	CLOCKTIME
UHRZEITSEKUNDE	CLOCKTIMESEC
VERBINDUNGSFEHLER	COMERROR
WOCHENTAG	DAYOFWEEK
ZEIT	TIME
ZUFALLSZEIT	RANDOMTIME

Funktionen zur Textbarbeitung

ERSETZEN	REPLACE
GROSSBUCHSTABEN	UPPERCASE
KLEINBUCHSTABEN	LOWERCASE
LESETEXTPAR	GETSTRPAR
LINKERTEIL	LEFTPART
OBJEKTBEZ	OBJECTDESCR
OBJEKTNAME	OBJECTNAME

RECHTERTEIL	RIGHTPART
TEXTLÄNGE	STRLEN
TEXTPOSITION	STRPOS
TEXTTEIL	COPYTEXT
TEXTZWISCHEN	TEXTBETWEEN

1.11.5.1. ABBRECHEN

ABBRECHEN

Syntax :
ABBRECHEN(*Objektname***)**

Mit dieser Anweisung wird die Ausführung des angegebenen Objektscripts abgebrochen, wenn es sich im WARTE-Status befindet (also aktuell eine WARTE-Anweisung aktiviert ist).

Beispiel:
ABBRECHEN(Lichtspiel)

1.11.5.2. ABFRAGE

ABFRAGE

Syntax :
ABFRAGE(*Objektname***)**

Mit dieser Anweisung kann der aktuelle Zustand/Wert der Hardware eines Objekts abgefragt und somit aktualisiert werden. Normalerweise ist das nicht nötig, da Änderungen automatisch aktualisiert werden. In speziellen Anwendungen kann es sinnvoll sein diese Anweisung zu verwenden, sie sollte jedoch nicht zu häufig und möglichst nur bei sicherheitsrelevanten Objekten benutzt werden, da sie relativ zeitintensiv ist und es bei zu häufiger Verwendung zu Verzögerungen bei der normalen Kommunikation kommen kann.

Beispiel:
ABFRAGE(Tiefkuehltruhe)

Bitte beachten Sie unbedingt:
Diese Anweisung generiert bei jeder Ausführung eine Meldung an das Gerät.
Wenn es sich um ein Funkmodul handelt kann die hemmungslose und unsachgemässe Benutzung dieser Anweisung viele unnötige Funkmeldungen generieren, was dazu führen kann, dass das Duty-Cycle-Konto überlauft und alle Funkmeldungen an Aktoren zeitweise blockiert werden.
Bei batteriebetriebenen Geräten wird ja nach Typ keine Meldung an das Gerät geschickt, wenn dieses nicht ständig empfangsbereit ist. In dem Fall wird der Wert gemeldet, den das Gerät in der WEB-UI hat. Dieser kann in seltenen Fällen vom tatsächlicvhen Wert des Geräts abweichen.

Diese Anweisung funktioniert nicht mit FHZ-Schnitttstellen weil im Protokoll der Geräte eine solche Abfage nicht vorgesehen ist.

1.11.5.3. AKTIVIEREN

AKTIVIEREN

Syntax :
AKTIVIEREN(*OBJEKT***)**

Die Ausführung im Zeitintervall wird aktiviert. Natürlich muss ein Zeitintervall definiert worden sein, damit das Programm in diesem Intervall ausgeführt werden kann.
siehe auch DEAKTIVIEREN

1.11.5.4. ALLEABFRAGEN

ALLEABFRAGEN

Syntax :
ALLEABFRAGEN

Diese Anweisung startet eine Hintergrundabfrage bei der der aktuelle Zustand aller Hardwaremodule abgefragt wird.

Bitte beachten Sie:
Da diese Abfrage intern beim Start automatisch ausgeführt wird und danach die Synchronisation immer auch automatisch erfolgt wird diese Anweisung normalerweise nicht verwendet und ist nur in speziellen Fällen sinnvoll wenn der Verdacht besteht, dass die automatische Synchronisation gestört wurde.

Da diese Anweisung erheblichen Funkverkehr verursachen kann und Ressourcen benötigt sollte sie keinesfalls unnötigerweise verwendet werden.

Bitte beachten Sie unbedingt:
Diese Anweisung generiert bei jeder Ausführung eine Meldung an das Gerät.
Wenn es sich um ein Funkmodul handelt kann die hemmungslose und unsachgemässe Benutzung dieser Anweisung viele unnötige Funkmeldungen generieren, was dazu führen kann, dass das Duty-Cycle-Konto überlauft und alle Funkmeldungen an Aktoren zeitweise blockiert werden.

1.11.5.5. ALLEWERTESICHERN

ALLEWERTESICHERN

Syntax :
ALLEWERTESICHERN

Mit dieser Anweisung werden alle Werte von Objekten und Variablen gesichert, so wie es beim normaler Beendigung der ExecEngine passiert wenn die Option "Aktuelle Werte der ExecEngine speichern/laden" aktiviert ist.
Die Datei wird auf einer Linux-Zentrale im Programmverzeichnis unter dem Namen <Projektname>.### gespeichert.

Diese Datei wird bei einem Neustart der ExecEngine eingelesen und die in der Datei gespeicherten Werte werden als Startwerte für die Objekte gesetzt.
Ob beim Neustart eine Hardwareabfrage durchgeführt wird, kann im Reiter Allgemein des Hardwarefensters in Abhängigkeit vom Alter der Datei eingestellt werden.

1.11.5.6. ANZEIGEN

ANZEIGEN

Syntax :
ANZEIGEN(*OBJEKT***)**

Mit dieser Anweisung wird der Text und die über Variablen eingestellten Zusatzinformationen zu einem speziellen Anzeige-Hardwaremodul (z.B. 19-Tasten-Fernbedienung) gesendet.
Für die 19-Tasten-Fernbedienung ist das Objekt, das als Parameter eingetragen werden muss das letzte Objekt der Fernbedienung,
also zum Beispiel:

```
Anzeigen(FernbedienungW_18)
```

1.11.5.7. AUFRUFEN

AUFRUFEN

Syntax :
AUFRUFEN (*OBJEKT***)**

Diese Anweisung führt das angegebene Script sofort aus. Nach Beendigung des aufgerufenen Scripts wird die Ausführung mit der nächsten Anweisung fortgesetzt.

1.11.5.8. DEAKTIVIEREN

DEAKTIVIEREN

Syntax :
DEAKTIVIEREN(*OBJEKT***)**

Die Ausführung im Zeitintervall wird deaktiviert. Alle anderen Möglichkeiten das zur Aktivierung des Objektprogramms sind davon nicht betroffen. Das heisst das Programm kann weiterhin durch die Optionen "Ausführung bei Empfang", "Ausführen bei Änderung" und die Anweisungen STARTE oder AUFRUFEN aktiviert werden.
siehe auch AKTIVIEREN

1.11.5.9. ERLEDIGT

ERLEDIGT

Syntax :
ERLEDIGT(*OBJEKT***)**

Das angegebene Objekt wird als erledigt gekennzeichnet, d.h. die nächste GESCHALTET - Abfrage gibt den Wert FALSCH zurück, falls sich der Zustand des Objekts nach der ERLEDIGT - Anweisung nicht wieder geändert hat.
siehe auch GESCHALTET

1.11.5.10. GEHEZU

GEHEZU

Syntax :
GEHEZU *ADRESSE*
.....
ADRESSE:
.....

Achtung: Bei falscher Anwendung dieser Anweisung können Endlosschleifen entstehen, durch die das aktuelle Script ständig läuft ohne beendet zu werden.
Eine gefährliche Anweisung, die im Normalfall nicht gebraucht wird und die man vermeiden sollte. Die GEHEZU-Anweisung bewirkt, dass die Programmausführung an der durch *ADRESSE* festgelegten Stelle fortgesetzt wird. Als Adresse kann ein beliebiger Name aus Buchstaben und Ziffern gewählt werden, das erste Zeichen muss ein Buchstabe sein. Die Adresse muss immer am Zeilenanfang beginnen und mit einem Doppelpunkt abgeschlossen werden. Innerhalb eines Objektprogramms muss eine Adresse eindeutig sein.
Diese Anweisung sollte möglichst vermieden werden, da bei falscher Anwendung Endlosschleifen entstehen können und die Ausführung des aktuellen Objektprogramms nicht beendet wird. Wenn die Sprungadresse vor der GEHEZU-Anweisung liegt, wird die Ausführung des Scripts für einen Prozesstakt unterbrochen, damit andere Scripts und Ein/Ausgaben ausgeführt werden können, und eine eventuelle Endlosschleife nicht das komplette System blockiert.

Beispiel:
WENN Uhrzeit > "22:00:00" DANN
GEHEZU ABEND
ENDEWENN
Anweisungen
.....
GEHEZU ENDE
ABEND:
Anweisungen
.....
ENDE:
Anweisungen

1.11.5.11. GETCCUSYSVAR

GETCCUSYSVAR

Syntax :
GETCCUSYSVAR(*Systemvariablenname,Zielvariable,IP_der_CCU***)**

Mit dieser Anweisung können Systemvariablen der CCU gelesen werden. Der Wert der Systemvariablen steht nach Ausführung der Anweisung in der angegebenen Variablen bzw. im Objekt zur Verfügung. Der Typ der als Ziel angegebenen Variablen oder des Objekts muss kompatibel mit dem Typ der Systemvariablen sein, soweit möglich wird der Typ konvertiert.
Bei Systemvariablen des Typs "Logikwert" wird in einer Textvariablen "true" oder "false" zurückgegeben, in einer Zahlenvariablen 1 oder 0.
Mit dieser Anweisung können nicht nur Systemvariablen der CCU, sondern auch Systemwerte ausgelesen werden. Dazu wird als Variablenname der Systemwert mit einem führenden * angegeben. Also z.B. "*SunsetTime". Wichtig ist, dass Gross/Kleinschreibung des Systemwerts genau eingehalten werden.

Als IP der CCU wird für die aktuelle Zentrale normalerweise "127.0.0.1" angegeben, es kann allerdings auch die IP-Adresse einer beliebigen CCU im Netzwerk angegeben werden. Die Angabe "localhost" funktioniert bei der CCU nicht zuverlässig, daher sollte besser die localhost-IP "127.0.0.1" verwendet weden.
Die Ausführung dieser Anweisung wird im Hintergrund durchgeführt. Während das passiert werden andere Scripts weiter ausgeführt. Nachdem die CCU-Logikschicht geantwortet hat wird die Ausführung des Scripts fortgesetzt. Die Ausführung der Anweisung kann je nach Auslastung der CCU und individueller Umgebung einige Sekunden dauern.
Daher muss unbedingt darauf geachtet werden dass die Abfragen nicht ständig in einem kürzeren Zeitintervall durchgeführt werden. Wenn das Zeitintervall bei permanenten Abfragen unter Antwortzeit des Requests liegt, kommt es nach einiger Zeit zu einem Überlauf des Buffers, was zur Folge hat, dass Speicher blockiert wird und keine weiteren HTTP-Requests mehr möglich sind. Daher sollte das Abfrageintervall für ständige Abfragen keinesfalls unter 5 Sekunden gewählt werden.

Beispiele:
```
GetCCUSysVar("Anwesenheit",CCU.Anwesenheit,"127.0.0.1")
GetCCUSysVar("*SunriseTime",Text1,"192.168.0.81")

Varname:="Anwesenheit"
GetCCUSysVar(Varname,MeineVariable,"127.0.0.1")
```

Bei diesem Beispiel wird nicht der Name der CCU-Systemvariablen angegeben, sondern die CL-Variable ***Varname***, in der der Name der CCU-Systemvariablen steht.

1.11.5.12. GETSITE

GETSITE

Syntax :
GETSITE(*Webseite,Port,Zielvariable***)**

Mit dieser Anweisung können beliebige Webseiten von einem Webserver aus dem lokalen Netzwerk oder aus dem Internet eingelesen werden indem ein http-Request erzeugt wird.
Als erster Parameter wird die aufzurufende Seite angegeben, falls ein anderer als der Standardport (80 bei http, 443 bei https) verwendet wird, kann dieser mit einem Doppelpunkt hinter der Adresse oder als zweiter Parameter der Anweisung angegeben werden. Falls der Port mit einem Doppelunkt hinter Adresse angegeben wird, wird der zweite Parameter ignoriert.
Falls https-Seiten aufgerufen werden, muss der Seitenname mit den Zeichen "https://" beginnen.
Als zweiter Parameter wird der Port angegeben. Wenn die Standardports verwendet werden kann hier 0 eingesetzt werden, es wird dann automatisch der richtige Port benutzt.
Wenn der Port mit einem Doppelunkt hinter dem Servernamen angegeben wird, wird der zweite Parameter ignoriert.
Als dritter Parameter wird die Variable oder das Objekt angegeben, in dem der Seitenquelltext empfangen wird. Diese Variable bzw. das Objekt muss vom Typ Zeichen sein.

Bitte beachten Sie, dass die einzulesenden Daten der Webseite eventuell sehr gross sein können. Die Ausführung des Befehls kann daher länger dauern und einiges an Ressourcen brauchen. Achten Sie daher darauf, dass die Seiten, die aufgerufen werden möglichst nicht zu gross sind. Die maximale Grösse, die in der Zielvariablen gespeichert wird ist abhängig von der verwendeten Hardware.

Die Ausführung dieses Befehls kann je nach Grösse der Seite einige Sekunden dauern. Die Ausführung dieser Anweisung wird im Hintergrund durchgeführt. Während das passiert werden andere Scripts weiter ausgeführt. Nachdem die Seite in der angegebenen Variablen gespeichert wurde wird die Ausführung des aktuellen Scripts fortgesetzt.

Beispiele:
```
GetSite("www.meine.seite.de/html/CCUSeite.html",0,Webseitentext)

GetSite("www.meine.seite.de/html/CCUSeite.html",8081,Webseitentext)
GetSite("www.meine.seite.de:8081/html/CCUSeite.html",0,Webseitentext)

GetSite("https://meine.spezielleseite.de/html/Seite1.html",0,Webseitentext)
```

Bitte beachten Sie:
Die Anweisung GETSITE eröffnet einen separaten Thread im Hintergrund, damit das Warten auf eine Internetseite die übrige Ausführung nicht behindert und Zeitverzögrungen vermieden werden. Die Verwaltung der Threads geschieht automatisch durch das Betriebssystem und beansprucht einige Ressourcen. Daher sollte diese Anweisung nur sparsam und mit möglichst grossem Zeitabstand verwendet werden.
Je nach Zentralenhardware, Ressourcenauslastung und individueller Umgebung kann es evenuell sein, dass es bei ständiger Ausführung der Anweisung in kürzeren Zeitabstände zu Problemen kommt.
Wir empfehlen diese Anweisung nur sparsam einzusetzen und bei ständiger Verwendung mindestens 30 Sekunden Abstand zwischen den Aufrufen einzuhalten.

Auf vielfachen Wunsch gibt es noch eine spezielle Option der GETSITE-Anweisung zur Ansteuerung spezieller Geräte über http-Requests. Bei dieser Option wird nur die http-Anfrage gesendet und unmittelbar danach die Netzwerkverbindung zum Server geschlossen - ohne auf eine Antwort zu warten.
Diese Option entspricht nicht den Spezifikationen zur Kommunikation mit einem Webserver und kann beim Server zu Fehlfunktionen führen !!
Diese Option ist nur als Ausnahme zum Ansprechen spezieller Geräte gedacht, bei denen der Webserver keine korrekte Antwort generiert oder diese zu lange dauert, und sollte nur in speziellen

Ausnahmefällen verwendet werden. Im Normalfall führt die Verwendung dieser Option zu Fehlfunktionen beim Server.
Um eine solche spezielle http-Anfrage abzusetzen wird einfach der Parameter "Zielvariable" weggelassen, da ja nicht auf eine Antwort gewartet wird.
Bei dieser Option muss der Port als Parameter 2 angegeben werden, die Angabe des Ports mit Doppelpunkt hinter dem Servernamen funktioniert hier nicht.
Die Option kann nur mit einfachen http://-Abfragen verwendet werden - https-Abfragen sind nicht möglich.

Beispiel:
```
GetSite("MeinSpeziellesGeraet/Pfad/Seite?Parameter",81)
```

1.11.5.13. GRUPPENZUWEISUNG

GRUPPENZUWEISUNG

Syntax :
GRUPPENZUWEISUNG(*Typ, Gruppe, Wert***)**

Mit dieser Anweisung kann mehreren Objekten gleichzeitig ein neuer Wert zugewiesen werden. Die Gruppe kann eine benutzerdefinierte Gruppe von Objekten ein oder ein Gerätecode. Bei Verwendung eines Gerätecodes wird allen Geräten mit diesem Code der angegebene Wert zugewiesen.
Die einzelnen Parameter haben folgende Bedeutung:
Typ
Typ der Gruppenzuweisung als Buchstabe. Möglich sind "G" für benutzerdefinierte Gruppen und "D" für Zuweisungen nach Gerätecode (Devicecode).
Gruppe
für den Typ "G" der Name der Gruppe als Zeichenkonstante oder Zeichenvariable.
für den Typ "D" der Gerätecode als Zahlenkonstante oder Variable.
Folgende Gerätecodes sind möglich:
1100 : Schaltaktoren
1200 : Dimmer des HomeMatic-Systems
1300 : Rollladenaktoren
10010 : Wandthermostate

Wert
Der Wert muss jeweils kompatibel zur angegebenen Gruppe sein. Das heisst für Binäraktoren muss der Wert 0 oder 1 sein oder der genaue Zustandstext der Aktoren.

Beispiele:
für Typ "G":
```
Gruppenzuweisung("G", "LichtEG", 1)
```
hat die gleiche Wirkung wie
```
Gruppenzuweisung("G", "LichtEG", "an")
Gruppenzuweisung("G", "AlleDimmer", 45)
Gruppenzuweisung("G", "RollladenEG", "halb")
```
für Typ "D":
```
Gruppenzuweisung("D", 1100, 0)

Gruppenzuweisung("D", 10010, 0)
```
kann benutzt werden um alle Wandthermostate auf "OFF" zu stellen

1.11.5.14. LESEWERTEDATEI

LESEWERTEDATEI

Syntax :
LESEWERTEDATEI(*"Speicherort/Dateiname"***)**

Mit dieser Anweisung können Objektwerte, die zuvor mit SCHREIBEDATEI in eine Datei geschrieben wurden wieder ausgelesen und den Objekten zugewiesen werden.
Eine Datei, die mit LESEWERTEDATEI gelesen wird muss im Linux-Format geschrieben worden sein, d.h. jede Zeile darf nur mit einem LF abgeschlossen werden. Dateien, die im Windows-Format vorliegen können nicht verwendet werden.
Eine Werte-Datei wird normalerweise mit der normalen SCHREIBEDATEI-Anweisung erstellt, es muss darauf geachtet werden als Zeilenende-Parameter 1 zu verwenden, damit die Datei im Linux-Format erstellt wird.
Der oder die Variablenwerte werden in die Datei geschrieben, wobei jeder Wert in einer separaten Zeile stehen muss.
Beispiel (die Variable WertText muss als Variable vom Typ Zeichen definiert sein):

```
Löschedatei("werte.txt")
WertText:="Schaltsteckdose=" + Schaltsteckdose
Schreibedatei("werte.txt",WertText,1)
WertText:="LichtDiele.Zeittabelle=" + LichtDiele.Zeittabelle
Schreibedatei("werte.txt",WertText,1)
```

in der Datei steht dann z.B.
Schaltsteckdose=an
LichtDiele.Zeittabelle=aus

Die Anweisung SCHREIBEDATEI hängt die neuen Zeilen normalerweise immer an die schon vorhandenen Zeilen an, es können beliebig viele Zeilen in die Datei geschrieben werden.
Wenn eine Datei nur einen Wert enthält sollte dise Datei auch nur eine Zeile enthalten, damit nicht unnötigerweise derselbe Wert mehrmals eingelesen wird. Um nicht vorher jedesmal die Anweisung LÖSCHEDATEI ausführen zu müssen gibt es die Option bei SCHREIBEDATEI ein * hinter den Dateinamen zu setzen, um die Datei neu zu erstellen, so dass diese nur eine Zeile beinhaltet.
Ansonsten muss vor der Erstellung einer neuen Werte-Datei eine alte Datei mit gleichem Namen mit der Anweisung LÖSCHEDATEI gelöscht werden! Wenn das nicht getan wird, kommt es zwar nicht zu einer Fehlfunktion, da immer der letzte gelesene Wert aktuell ist, aber die die Datei wird immer grösser und beim Lesen werden viele unnötige Zeilen verarbeitet.

Es können beliebig viele unterschiedliche Wertedateien erstellt werden, so ist es auch möglich für bestimmte Objekte/Variablen jeweils separate Dateien zu benutzen.

Mit der Anweisung LESEWERTEDATEI kann diese Datei gelesen werden, die gespeicherten Werte werden dann direkt in die Objekte bzw. Variablen eingesetzt. Die Zeile
Schaltsteckdose=an
in der Datei würde also bewirken, dass das Objekt "Schaltsteckdose" den Wert "an" bekommt.

Beispiel für die Anweisung:

```
Lesewertedatei("werte.txt")
```

1.11.5.15. LÖSCHEANZEIGE

LÖSCHEANZEIGE

Syntax :
LÖSCHEANZEIGE(*OBJEKT***)**

Mit dieser Anweisung werden die Zusatzvariablen eines speziellen Anzeigemoduls (z.B. 19-Tasten-Fernbedienung) gelöscht.
Diese Anweisung bewirkt keine Aktualisierung des Hardwaremoduls, falls dieses aktualisiert werden soll, muss zusätzlich die Anweisung ANZEIGEN benutzt werden.

1.11.5.16. LÖSCHEDATEI

LÖSCHEDATEI

Syntax :
LÖSCHEDATEI(*"Unterverzeichnis/Dateiname"***)**

Mit dieser Anweisung können Dateien gelöscht werden. Das können selbst erstellte Textdateien, aber auch beliebige andere Dateien wie die Historydatei history.txt oder die Systemlog-Datei syslog.txt sein.

Bitte beachten Sie unbedingt:
Durch das beabsichtigte oder unbeabsichtigte Löschen von Dateien, die nicht selbst erstellt wurden, kann die Systemintegrität zerstört und die Funktionen der Zentrale beeinträchtigt werden.
Löschen Sie daher keinesfalls Dateien, von denen Sie nicht wissen wofür diese benutzt werden .
Achten Sie unbedingt darauf, dass das auch nicht versehentlich, etwa durch Angabe eines falschen Dateinamens, geschieht.
Aus Sicherheitsgründen ist das Löschen von Dateien nur innerhalb des festgelegten Standardverzeichnisses möglich, daher sollte aus Sicherheitsgründen immer ein Standardverzweichnis angegeben werden.

Beispiel:
```
LÖSCHEDATEI("BEWEGUNGEN.TXT")
```

1.11.5.17. SCHLIESSEDATEIEN

SCHLIESSEDATEIEN

Syntax :
SCHLIESSEDATEIEN

Mit dieser Anweisung werden alle geöffneten Dateien, die mit der Anweisung SCHREIBEDATEI erstellt worden sind geschlossen.
Diese Anweisung muss verwendet werden bevor auf die Datei zugegriffen wird (z.B. zum email-Versand) und bevor die Zentrale ausgeschaltet wird oder der USB-Stick entfernt wird, andernfalls könnten Daten verloren gehen.

Beispiel:

```
SCHLIESSEDATEIEN
```

1.11.5.18. SCHREIBEDATEI

SCHREIBEDATEI

Syntax :
SCHREIBEDATEI(*"Pfad/Dateiname","Text",Zeilenvorschubcode***)**

Mit dieser Anweisung können Textdateien erstellt werden.
Je nach Typ der verwendeten Zentrale werden unterschiedliche Verzeichnisse verwendet.

Als erster Parameter in der Klammer wird der Dateiname angegeben, Dieser Parameter ist vom Typ Zeichen und kann eine Konstanten oder eine Variable sein. Der Dateiname kann auch einen gültigen Pfad beinhalten, dieser muss dann aber schon vorhandenen sein, es wird kein neues Verzeichnis angelegt falls ein nicht existierendes Verzeichnis angegeben wird.
Die neue Zeile in der Datei wird an den vorhandenen Inhalt angehängt. Oftmals ist es erforderlich, dass die neue Zeile nicht angehängt wird, sondern als einziger Wert in der Datei stehen soll. Das ist z.B. der Fall wenn die Datei später mit LESEWERTEDATEI benutzt wird.
Um die Datei neu zu schreiben und den vorherigen Inhalt zu überschreiben, kann ein * als letztes Zeichen an den Dateinamen angehängt, beispielsweise:
SCHREIBEDATEI(*"Alterwert.txt*", Zeile, 1)*

Als zweiter Parameter die Zeile, die in die Datei geschrieben werden soll, dieser Parameter ist vom Typ Zeichen und kann eine Konstamte in Hochkomma ober eine Variable sein,

Als dritter Parameter wird angegeben welcher Zeilenvorschub für die ausgegebene Zeile verwendet werden soll. Dieser Parameter ist vom Typ Zahl und kann als Konstante oder Variable angegeben werden. Es gibt folgende Codes für den Zeilenvorschub:
0 = kein Zeilenvorschub, die nächste Ausgabe beginnt unmittelbar hinter dieser Zeile
1 = Linux Zeilenvorschub (LF), es wird nur ein Linefeed ausgegeben, mit einem Editor unter Windows wird normalerweise kein Zeilenvorschub dargestellt. Wenn die Datei mit LESEWERTEDATEI gelesen werden soll, muss dieser Code für den Zeilenvorschub verwendet werden.
2 = Windows-Zeilenvorschub (CR/LF), es werden die für Windows üblichen zwei Codes für den Zeilenvorschub ausgegeben, so dass die Datei nach einem Download unter Windows mit einem Texteditor betrachtet werden kann. Im Normalfall sollte diese Option benutzt werden.

Bei der CCU1 werden die Textdateien auf einem USB-Stick, der in der Zentrale eingesteckt sein muss, geschrieben.
Bei der CCU2 kann sowohl auf die SD-Karte wie auch auf einen USB-Stick geschrieben werden. Wenn auf die SD-Karte der CCU2 geschrieben werden soll muss diese über die WEB-UI initialisiert und eingebunden worden sein. Das passiert im Menüpunkt
Einstellungen->Systemsteuerung->Speichereinstellungen.

Auf welches Medium bzw. in welches Verzeichnis geschrieben wird, kann im Reiter Zentrale des Hardwarefensters festgelegt werden. Bei der CCU1 kann das Verzeichnis nicht verändert werden, da nur das vorgegebene Verzeichnis verfügbar ist. Bei der CCU2 kann ein beliebiges Verzeichnis als Standardverzeichnis angegeben werden. Voreingestellt ist der Standardpfad für die SD-Karte. Falls standardmässig auf einen USB-Stick geschrieben werden soll, muss der Pfad **/var/datadisk/** angegeben und der USB-Stick eingebunden werden.

Bei der CCU3 wird auf einen eingesteckten USB-Stick geschrieben, das Standardverzeichnis ist hier
/media/usb0/ .

Wenn in der Anweisung nur der Dateiname ohne Pfadangabe benutzt wird wird in das festgelegte Standardverzeichnis geschrieben. Wenn vor dem Dateinamen ein Pfad angegeben ist, der mit / beginnt, ist das eine absolute Pfadangabe und die Datei wird an den angegebenen Ort geschrieben. Wenn in ein Unterverzeichnis des Standardverzeichnisses geschrieben werden soll darf die Pfadangabe nicht mit / sondern muss mit dem Namen des Unterverzeichnisses beginnen. Das Unterverzeichnis muss existieren, also vorher angelegt worden sein.
Als erster Parameter in der Klammer wird der Dateiname (ggfs. mit Pfad) angegeben, als zweiter

Parameter die Zeile, die in die Datei geschrieben werden soll. Beide Parameter sind vom Typ Zeichen und können Konstanten oder Variablen sein. Der Dateiname kann auch einen gültigen Pfad beinhalten, dieser muss dann aber auf dem USB-Stick vorhandenen sein erstellt werden.
Als dritter Parameter wird angegeben welcher Zeilenvorschub für die ausgegebene Zeile verwendet werden soll. Dieser Parameter ist vom Typ Zahl und kann als Konstante oder Variable angegeben werden. Es gibt folgende Codes für den Zeilenvorschub:
0 = kein Zeilenvorschub, die nächste Ausgabe beginnt unmittelbar hinter dieser Zeile
1 = Linux Zeilenvorschub (LF), es wird nur ein Linefeed ausgegeben, mit einem Editor unter Windows wird normalerweise kein Zeilenvorschub dargestellt.
2 = Windows-Zeilenvorschub (CR/LF), es werden die für Windows üblichen zwei Codes für den Zeilenvorschub ausgegeben, so dass die Datei nach einem Download unter Windows mit einem Texteditor betrachtet werden kann. Im Normalfall sollte diese Option benutzt werden.

Bevor ein USB-Stick entfernt wird muss die Anwendung SCHLIESSEDATEIEN ausgeführt werden, sonst gehen die zuletzt geschriebenen Daten bzw. Dateien verloren.

Bitte beachten Sie bei der CCU2 unbedingt:
Wenn ein USB-Stick verwendet wird muss dieser formatiert sein und **vor** dem Start der Zentrale eingesteckt werden. Nach dem Einstecken eines USB-Sticks muss die Zentrale ggfs. neu gebootet werden oder der USB-Stick muss über den entsprechenden Punkt des Webservers in der WEB-UI eingebunden werden.
Menüpunkt Einstellungen->Systemsteuerung->CL-Studio->Dateien bearbeiten.

Über diese WEB-Server-Seite können die Dateien bearbeitet werden und es kann ein Download der Dateien durchgeführt werden.
Bevor Dateien bearbeitet oder downgeloadet werden muss die Anweisung SCHLIESSEDATEIEN ausgeführt werden, damit alle geschriebenen Daten in den Dateien vorhanden sind. Ein USB-Stick sollte nicht entfernt werden bevor die Dateien durch Download gesichert wurden.
Bei der Angabe eines Pfads, bei dem in andere Verzeichnisse als die Standardverzeichnisse für SD-Karte oder USB-Stick geschrieben wird kann das Betriebssystem der Zentrale und die Systemstabilität beschädigt werden. Auf andere Verzeichnisse als die angegebenen Standardverzeichnisse kann über das Webserverprogramm nicht zugegriffen werden, d.h. Dateien an anderen Orten können nicht über das Browserprogramm gewartet werden. Benutzen Sie andere Verzeichnisse als die Standardverzeichnisse nur wenn Sie genau wissen was Sie tun und über entsprechende Fachkenntnisse verfügen.

Wenn das Standardverzeichnis für USB-Stick der SD-Karte angegeben wird und diese nicht eingebunden worden sind, wird in den RAM der Zentrale geschrieben.
Wenn die Dateien dort zu viel Platz beanspruchen wird die Funktionalität der Zentrale beeinträchtigt und es kommt zu Fehlfunktionen oder Absturz !
Die Anweisung SCHREIBEDATEI sollte also möglichst immer nur Dateien auf SD-Karte oder USB-Stick schreiben. Andernfalls müssen die Dateien regelmässig kontrolliert und ggfs. gelöscht werden. Andernfalls kann es passieren, dass zu viel Speicher der Zentrale belegt wird und ein normaler Betrieb nicht mehr möglich ist.
Beispiel:
(Zeile ist im Script als Variable vom Typ Zeichen definiert)

```
Zeile:="Uhrzeit " + Uhrzeit + ":  Bewegungsmelder Haustür"
SCHREIBEDATEI("BEWEGUNGEN.TXT",Zeile,2)
```

Beschreibung der Verwendung dieser Anweisung wenn der PC als Zentrale benutzt wird:
In welches Verzeichnis standardmässig geschrieben wird, kann im Reiter Zentrale des Hardwarefensters festgelegt werden.
Es kann auch Dateiname angegeben werden, der einen Pfad beinhaltet.

Beispiel:
```
SCHREIBEDATEI("c:\daten\CL\MeineDatei.txt","Das ist der Text",2)
```

1.11.5.19. SENDE

SENDE

Syntax :
SENDE(*Objektname***)**

Mit dieser Anweisung kann der aktuelle Zustand eines Objekts an den zugeordneten Empfänger gesendet werden, ohne dass der Zustand des Objekts sich geändert hat.
Gesendet wird nicht bei Ausführung des Befehls sondern immer erst nach Beendigung des aktuellen Scripts oder bei einer WARTE-Anweisung.
Diese Anweisung sollte auf keinen Fall in Scripts verwendet werden, die in einem Zeitintervall aufgerufen werden.

Die Anweidung SENDE funktioniert nicht bei bei Geräten, bei denen die Einstellung nicht aufgrund des aktuellen Objektzustands vom Geräte selbst vorgenommen werden kann.
Die Anweisung kann z.B. nicht für FS20-Rolllalden/Markisten-Aktoren benutzt werden, da hierbei die Einstellung über eine Prozedur in der Zentrale mit mehreren Funkmeldungen vorgenommen wird.

Bitte beachten Sie unbedingt:
Diese Anweisung generiert bei jeder Ausführung eine Meldung an das Gerät.
Wenn es sich um ein Funkmodul handelt kann die hemmungslose und unsachgemässe Benutzung dieser Anweisung viele unnötige Funkmeldungen generieren, was dazu führen kann, dass das Duty-Cycle-Konto überlauft und alle Funkmeldungen an Aktoren zeitweise blockiert werden.

1.11.5.20. SENDEMAIL

SENDEMAIL

Syntax :
SENDEMAIL(*"Zieladresse","Betrefftext","Erste Zeile<<zweite Zeile<<dritte Zeile"<,"Anhang1;Anhang2">***)**

Mit dieser Anweisung können Mails versendet werden. Voraussetzung ist, dass im Netzwerk, in dem die Zentrale angeschlossen ist, eine Verbindung zum Internet besteht oder automatisch aufgebaut wird sobald die Zentrale eine Verbindungsanforderung zu einem Mail-Server schickt.
Als erster Parameter der Anweisung wird die Zieladresse angegeben, als zweiter Parameter folgt der Betrefftext, als dritter Parameter der Text der Mail. Wenn mehrere Zeilen Text gesendet werden sollen, so können die <<-Zeichen als Zeilenvorschub-Zeichen verwendet werden, der Text darf maximal 8000 Zeichen haben.
Diese ersten drei Parameter müssen angegeben werden, jeder Parameter muss gültige Daten enthalten und darf nicht leer sein.
Der vierte Parameter ist optional, es können mehrere Dateien getrennt durch Semikolon angegeben werden, die als Anhang mit der mail geschickt werden.
Wenn zu einer Datei kein Pfad angegeben wird, wird der Standardpfad benutzt, der im Hardwarefenster im Reiter Zentrale als Pfad für eigene Dateien auf der Zentrale angegeben ist.
Mit der Funktion MAILLOG, kann der Status der jeweils letzten fünf versendeten Mails überprüft werden.
Wenn Mail-Server, Absenderadresse oder einer der ersten drei Parameter nicht angegeben oder leer ist wird die Mail nicht versendet. Im Maillog erscheint dann der Fehler 999.

Im Rahmen neuer Sicherheitsmaßnahmen müssen bei vielen Providern Absenderadressen und Kontonamen gleich sein um Täuschungen durch falschen Absender zu verhindern. Wenn Mails nicht versendet werden kann das eine Ursache sein, die geprüft werden sollte.

Bitte beachten Sie, dass jedesmal wenn die Anweisung ausgeführt wird eine Mail gesendet wird. Die Programmierung muss so aufgebaut sein, dass nicht ungewollt viele Mails versendet werden. Werden Mails z.B. als SMS auf das Handy gesendet, können Fehler in der Programmierung, die ungewollt viele Mails erzeugen, zu erheblichen Kosten führen. Es müssen also Vorkehrungen getroffen werden, um ein zu häufiges Versenden von Mails zu verhindern.
Im folgenden Beispiel verhindert die Abfrage nach MaxTemp, dass bei konstanter Temperatur z.B. über einem Grenzwert dauernd gesendet würde.
Beispiel:

```
sendemail("meineAdresse@Mein-Provider.com","Temperaturprotokoll","Temperatu
ren von heute","Temperaturen.txt")

wenn TempASH550 > MaxTemp dann
  Mailtext:="Info Treibhaus:<<Temperatur ist " + TempASH550 + " Grad"

sendemail("Meine.BueroAdresse@Mein-Provider.com","Temperaturinfo",Mailtext)
  MaxTemp:=TempASH550 + 0.5
endewenn
```

Es wird jetzt immer nur dann gesendet, wenn die Temperatur um mehr als 0,5 Grad gestiegen ist. Würde die Abfrage dagegen z.B. lauten

```
wenn TempASH550 > 25 dann
```

so würde bei jeder Scriptausführung, bei der eine höhere Temperatur als 25 Grad erreicht ist eine Mail gesendet, also z.B. alle paar Minuten wenn ein Temperaturwert empfangen wird, oder gar alle 5 Sekunden, wenn das Script in diesem Zeitintervall ausgeführt würde.

Eine andere Möglichkeit ist, das Versenden von Mails z.B. über "virtuelle" Objekte zu steuern. Im folgenden Beispiel würde z.B. nur jede halbe Stunde eine Mail verschickt:

```
wenn mailSchalter eingeschaltet und Schaltdauer(mailSchalter) > "00:30:00"
```

```
dann
  mailSchalter ausschalten
endewenn
wenn mailSchalter ausgeschaltet dann
  sendemail("Meine.BueroAdresse@Mein-Provider.com","Wasseralarm","Wasser
im Keller !")
  mailSchalter einschalten
endewenn
```

Beachten Sie bitte, dass die Funktion SCHALTDAUER nur für Objekte, nicht aber für Variablen verwendet werden kann.
Eine weitere Möglichkeit ist z.B. die Zeiten zwischen zwei Mails über STARTUHR und STOPPUHR zu begrenzen.

```
wenn STOPPUHR(LetzteMail) > "00:30:00" dann
  sendemail("Meine.BueroAdresse@Mein-Provider.com","Wasseralarm","Wasser
im Keller !")
  STARTUHR(LetzteMail)
endewenn
```

Bitte unbedingt beachten:
Testen Sie Ihre Programmierung und überprüfen ob nicht unbeabsichtigt viele Mails versendet werden. Beachten Sie, dass Mails Kosten verursachen können, z.B. durch Verbindungsentgelte oder SMS auf ein Handy.

1.11.5.21. SENDESMS

SENDESMS

Syntax :
SENDESMS(*"TelefonnummerEmpfänger", "Text der SMS", "Absendername", Statusvariable***)**

Mit dieser Anweisung können SMS versendet werden.

Zum Versand von SMS ist ein SMS-Konto mit Guthaben erforderlich.
Zu diesem Konto gibt es einen SMS-Code, der im Reiter *Sicherheit* des Einstellungsfensters angegeben wird.
Eine SMS kann maximal 160 Zeichen haben.
Das SMS-Guthaben kann im CL-Shop unter diesem Link bestellt werden:
SMS für CL-Software

Beschreibung der Anweisung SENDESMS:
Als erster Parameter der Anweisung wird die Mobilfunknummer des Handys angegeben, an das die SMS gesendet werden soll.
Wenn diese Nummer eine Festnetznummer im Inland ist wird dort angerufen und die SMS vorgelesen.
Je nach Telefonanbieter des Festnetzanschlusses kann es sein, dass dieses Featurue nicht zur Verfügung steht steht.

Als zweiter Parameter wird der Text der SMS angegeben. Der Text muss aus Buchstaben, Ziffern und Leerezichen bestehen, zulässige Sonderzeichen sind ! ? *, Komma, Punkt, Doppelpunkt, Semikolon, Klammern und einfaches Anführungszeichen.
Die Zeichen % & + # § werden in Leerzeichen umgewandelt.
Umlaute werden in eine andere Schreibweise umgewandlet ä zu ae, Ö zu Oe usw.
Bei Verwendung anderer als der aufgeführten Sonderzeichen werden diese in ? umgewandelt, können jedoch auch dazu führen, dass die SMS nicht versendet werden kann.

Dritter Parameter ist der Absendername, unter diesem Namen erscheint die SMS auf dem Ziel-Handy.

Vierter Parameter der Anweisung ist eine Variable oder ein Objekt vom Typ Zeichen, welches selbst erstellt werden muss. In diese Variablen wird eine Statusmeldung geschrieben sobald die SMS an den SMS-Server übergeben wurde.
Es werden 3 Parameter, getrennt durch Komma in die Statusvariable geschrieben.
Erster Parameter ist eine dreistellige Zahl.
100 bedeutet, dass die SMS ohne Fehler SMS-Server übertragen wurde, eine andere Zahl oder Text zeigen an, dass ein Fehler aufgetreten ist.
Der zweite Parameter gibt an wieviele SMS-Credits verbraucht wurden und ist abhängig von der Länge der gesendete SMS.
Wenn die Nachricht kleiner als 160 Zeichen ist wird nur 1 SMS-Credit berechnet, über 160 Zeichen 2 Credits.
Eine SMS kann maximal 320 Zeichen haben, wenn der Text mehr Zeichen hat werden diese abgeschnitten.
Wenn aufgrund eines Fehlers in den Parametern eine SMS nicht versendet werden konnte, erfolgt keine Belastung der Credits, das gilt jedoch nicht wenn ungültige Mobilfunknummern verwendet werden, da dann trotzdem ein (erfolgloser) Versand erfolgt.
Den aktuellen Kontostand Ihrer SMS-Credits können Sie sich auf folgender Seite anzeigen lassen:
https://cl-control.de/sms/smskonto.php

Allgemeine Hinweise:
Eine SMS wird nicht bei Ausführung des Befehls versendet um Verzögerungen der Ausführung durch Internetzugriffe zu vermeiden. Das Versenden passiert nach Ausführung des Befehls im Hintergrund, daher steht der Status des SMS-Versands immer erst einige Sekunden nach Ausführung des Befehls in der Variablen zur Verfügung.

Wann die SMS empfangen wird ist vom Mobilfunknetz bzw. der Mobilfunkverbindung des Empfängers abhängig.

Bitte beachten Sie:

Wenn versucht wird eine SMS an eine ungültige Mobilfunknummer versendet wird erscheint keine Fehlermeldung, weil die Unzustellbarkeit der Mail erst bei der späteren Verarbeitung im Mobilfunknetz festgestellt wird. Da es bei einem solchen Fehler keine Rückantwort gibt ist eine entsprechende Gutschrift auf dem SMS-Konto nicht möglich.

```
wenn TempTiefkuehl > -10 und SMSTiefkuehlMerker ausgeschaltet dann
  SMSText:="Achtung Temperatur Tiefkühlschrank ist " + TempTiefkuehl + "
Grad"
  sendesms("01712345678", SMSText, "MeinHaus", SMSStatus)
 SMSTiefkuehlMerker einschalten
endewenn
```

Bitte unbedingt beachten:

Sie müssen durch geeignete Programmierung sicherstellen, dass nicht ungewollt zu viele SMS verschickt werden, also z.B. einen Zähler oder Merker wie im zweiten Beispiel verwenden. Ein solcher Merker kann dann manuell oder zeitgesteuert wieder ausgeschaltet bzw. auf 0 gesetzt werden.

Sie sollten Ihre Programmierung testen und überprüfen ob nicht unbeabsichtigt viele SMS versendet werden.

1.11.5.22. SETCCUSYSVAR

SETCCUSYSVAR

Syntax :
SETCCUSYSVAR(*Systemvariablenname,Wert,IP_der_CCU***)**

Mit dieser Anweisung können Systemvariablen der CCU verändert werden. Der Type des angegebenen Werts muss dem Typ der Systemvariablen entsprechen.
Bei Systemvariablen des Typs "Logikwert" kann eine Variable des Typs Zahl ohne Komma verwendet werden, der Wert 0 entspricht "falsch" der Wert 1 entspricht "wahr".
Der Wert kann eine Konstante oder eine Variable sein.
Als IP der CCU wird für die aktuelle Zentrale normalerweise "127.0.0.1" angegeben, es kann allerdings auch die IP-Adresse einer beliebigen CCU im Netzwerk angegeben werden. Die Angabe "localhost" funktioniert bei der CCU nicht zuverlässig, daher besser die "localhost"-IP "127.0.0.1" verwenden.
Die Ausführung dieser Anweisung wird im Hintergrund durchgeführt. Während das passiert werden andere Scripts weiter ausgeführt. Nachdem die CCU-Logikschicht geantwortet hat wird die Ausführung des Scripts fortgesetzt. Die Ausführung der Anweisung kann je nach Auslastung der CCU und individueller Umgebung einige Sekunden dauern.
Daher muss unbedingt darauf geachtet werden, dass die Abfragen nicht ständig in einem kürzeren Zeitintervall durchgeführt werden. Wenn das Zeitintervall bei permanenten Abfragen unter Antwortzeit des Requests liegt kommt es nach einiger Zeit zu einem Überlauf des Buffers, was zur Folge hat, dass Speicher blockiert wird und keine weiteren HTTP-Requests mehr möglich sind. Daher sollte nicht mehr als eine Abfrage innerhalb von 5 Sekunden gemacht werden.

Beispiele:

```
SetCCUSysVar("Anwesenheit",1,"127.0.0.1")

SetCCUSysVar("MeinText","Das ist der Text","192.168.0.55")
```

1.11.5.23. SETZEHISTORYDIFFERENZ

SETZEHISTORYDIFFERENZ

Syntax :
SETZEHISTORYDIFFERENZ(*"Variablenname",Differenzwert***)**

Wenn für ein Objekt oder eine Variable die Historyfunktion aktiviert ist, wird jeder neue Wert, der sich vom vorigen Wert unterscheidet, in die History-Datei geschrieben, Bei Temperaturen z.B. auch wenn diese sich nur um 0,1 Grad schwanken. Dadurch entstehen sehr viele im Normalfall nicht benötigte Einträge, die die Historydatei unnötig aufblähen. Oftmals sind es gerade bei Temperaturen über einen längeren Zeitraum nur minimale Schwankungen von 0,1 Grad nach oben und unten, die viele einzelne Einträge verursachen.

Daher werden bei Werten mit Gleitkommazahlen (z.B. Temperaturen) nur neue Einträge in die History geschrieben, wenn ein Wert sich um 0,2 oder mehr vom letzten Historywert unterscheidet. Bei ganzen Zahlen (z.B. Luftfeuchtigkeit) wird nur ein neuer Historyeintrag geschrieben, wenn er sich um mehr als 1 vom vorigen Wert unterscheidet. Dadurch wird die Grösser der Historydatei erheblich vermindert.
Mit der Anweisung SETZEHISTORYDIFFERENZ kann in einem Script (vorzugsweise in einem INIT_ Script) der für das Schreiben von Historydaten relevante Differenzwert individuell festgelegt werden. Wenn ein Wert 0 gesetzt wird, wird jede Veränderung in die Historydatei geschrieben, bei grösseren Werten nur wenn der mit der Anweisung festgelegte Wert erreicht oder überschritten wird. Bei Temperaturwerten kann es durchaus sinnvoll sein den Differenzwert auf 0,5 zu setzen, bei der Luftfeuchtigkeit auf einen Wert grösser 2.

Bitte unbedingt beachten:
Der erste Parameter ist eine Zeichenkonstante, diese muss in Hochkomma gesetzt werden!

Beispiel:

```
SetzeHistoryDifferenz("Raumthermostat1.Temperatur", 0.5)
SetzeHistoryDifferenz("Raumthermostat1.Luftfeuchtigkeit", 3)
SetzeHistoryDifferenz("TemperaturAquarium", 0.1)
```

1.11.5.24. SETZEKONFIG

SETZEKONFIG

Syntax :
SETZEKONFIG(*Parameter***)**

Mit dieser Anweisung können spezielle Konfigurationseinstellungen und Aktionen während der Laufzeit vorgenommen werden.
Diese Anweisung funktioniert nicht auf einer CCU1.

Dazu gibt es folgende Parameter:

SETZEDT zur Einstellung welches Dezimaltrennzeichen verwendet werden soll bei der Zuweisung von Fliesskommzahlen an Textvariablen bzw. bei der Ausgabe von Zahlenwerten.
Die Zeichenfolge
SETZEDT=.
setzt den Punkt als Dezimaltrennzeichen, bei
SETZEDT=,
wird das Komma sls Dezimlatrennzeichen festgelegt.
Manchmal kann es sinnvoll sein das Dezimaltrennzeichen nur kurz für bestimmte Zuweisungen innerhalb eine Scripts umzustellen.
Beispiel:

```
SETZEKONFIG("SETZEDT=.")
MeinTextPar:=NeueTemperatur + ", XY, 3"
SETZEKONFIG("SETZEDT=,")
```

Die Parameter BEENDEN oder ENDPROGRAM bewirken eine Beendigung der ExecEngine nach Ausführung der Anweisung.
Bei einer Linux-Zentrale wird die ExecEngine durch einen Watchdog automatisch neu gestartet.
Bei einem PC als Zentrale erfolgt kein automatischer Neustart.
Beispiel:

```
SETZEKONFIG("BEENDEN")
```

1.11.5.25. SETZEWERT

SETZEWERT

Syntax :
SETZEWERT(*Objektname,Wertecode,Wert***)**

Mit dieser Anweisung können verschiedene Einstellungen in den Hardwaremodulen des HomeMatic-Systems vorgenommen werden.
Für das FHZ-System gibt es auch einige Module, für die SETZEWERT verwendet werden kann, weitere Infos dazu finden Sie unter FHZ-System.
Diese Anweisung erzeugt immer einen Sendebefehl und sollte nur sparsam verwendet werden, Sie darf keinesfalls in Scripts erwendet werden, die in kurzen Zeitintervallen ausgeführt werden, das könnte den DUTY-CYCLE an seine Grenzen bringen und so das ganze System blockieren.
Es muss auch unbedingt beachtet werden, dass bei Einstellungen über diese Anweisung nur ein Sendebefehl erzeugt wird. Die Werte und Zustände von Objekten und Variablen werden nicht direkt beeinflusst, sondern erst durch eventuelle Rückmeldungen von den Hardwaremodulen aktualisiert. Das kann zur Folge haben, dass Hardware und Visualisierung nicht oder nicht zeitgleich synchron sind und Zustandsabfragen in wenn-Bedingungen nicht aktuell sind. Daher sollte diese Möglichkeit den Zustand von Hardwaremodulen zu ändern nur in Ausnahmefällen benutzt werden.
Die Anweisung sollte nur von erfahrenen Benutzern verwendet werden, die unsachgemässe Verwendung dieser Anweisung kann zu Fehlfunktionen führen.

Die Wertecodes können Konstanten oder auch Variablen vom Typ Zeichen sein.
Die Wertecodes müssen in Grossbuchstaben geschrieben sein, es kann auch die englische Bezeichnung (in Klammern) verwendet werden.
Mit dieser Anweisung wird die Methode setValue aus dem XMLRPC-Protokoll des BidCoS-Services aufgerufen. Das funktioniert nicht nür für die im unteren Teil angegebenen Wertecodes, sondern auch für andere, es kann aber setValue-Kombinationwen geben, die nicht funktionieren z.B. weil ein anderer Kanal erfoderlich ist als der im Objekt hinterlegte.

Es liegt in der Verantwortung des Benutzers zu prüfen ob die verwendeten Wertecodes und Werte zu dem Gerät passen, an das gesendet wird.
Der Compiler kann nicht prüfen ob die benutzten Wertecodes und Werte plausibel und richtig sind!

Beipsiel zum einschalten den BOOST-Modus bei Thermostaten (wird von Thermostaten der ersten Serie nicht unterstützt):
```
SETZEWERT(ThermostatName,"BOOST_MODE",1)
```

Bitte unbedingt beachten:
Bis auf den ersten Parameter müssen alle in Hochkomma stehen und Grossbuchstaben haben. Nur bei ganzen Zahlen muss der letzte Parameter nicht in Hochkomma stehen.
Diese Option funkltioniert nur mit dem HMIP-Rauchmelder, nicht mit dem normalen HomeMatic--Rauchmelder.

Für folgende Wertecodes können momentan deutsche Schlüsselwörter verwendet werden:
EINSCHALTZEIT (ON_TIME), Einheit ist Sekunden
Beispiel:
```
Setzewert(Stehlampe,"EINSCHALTZEIT",30)
Stehlampe einschalten
```

Mit diesem Schlüsselwert kann z.B. für Schaltaktoren und Dimmer die Einschaltzeit eingestellt werden werden. Es ist keine Dauer-Einstellung, sie ist nur gültig für den nächsten Einschaltbefehl an das Modul. Bei dieser Möglichkeit ein Gerät für eine bestimmte Zeit einzuschalten, wird der interne Timer des Moduls benutzt, d.h. das Gerät schaltet nach der angegebenen Anzahl von Sekunden aus, auch wenn die Zentrale nicht mehr aktiv ist. Die Verwendung dieser Methode kann bei sicherheitskritischen Funktionen vorteilhaft sein.

VERZÖGERUNG (RAMP_TIME), Einheit ist Sekunden
Beispiel:

```
Setzewert(Dimmer,"VERZÖGERUNG",8)
Dimmer:=70
```

Mit diesem Schlüsselwert kann z.B. die Dimmgeschwindigkeit von Dimmern gesteuert werden.
ALTE STUFE (OLD_LEVEL)
Beispiel:
```
Setzewert(Dimmer,"ALTE STUFE",1)
```
Mit diesem Schlüsselwert kann ein Dimmer auf seine letzte Dimmstufe vor dem Ausschalten gesetzt werden. Die 1 ist als fester Wert erforderlich.

STUFE (LEVEL)
Beispiel:
```
Setzewert(Dimmer,"STUFE","0.6")
```
Mit diesem Schlüsselwert kann die Helligkeit eines Dimmers eingestellt werden. Der Wert ist ein Dezimalwert zwischen 0 und 1, als Parameter in Form einer Konstante oder Variable vom Typ Zeichen.
Diese Möglichkeit der Dimmersteuerung sollte nur in Ausnahmefällen benutzt werden, besser ist es den Dimmer über eine Zuweisung in einem Script (z.B. Dimmer:=60) einzustellen.

ZUSTAND (STATE)
Beispiel:
```
Setzewert(Stehlampe,"ZUSTAND",1)
```

Mit diesem Schlüsselwert kann ein Binäraktor geschaltet werden. Bei Wert 1 wird eingeschaltet, bei Wert 0 wird ausgeschaltet. Diese Steuerungsmöglichkeit sollte nur in Ausnahmefällen verwendet werden, besser ist es Binäraktoren über die Anweisungen "einschalten" und "ausschalten" zu steuern.

Wichtiger Hinweis zur besonderen Behandlung von Werten des Typs "bool"
Es gibt viele Werte vom Typ "bool". Damit diese korrekt gesetzt werden muss als Wert "TRUE" oder "FALSE" (jeweils in doppeltem Hochkomma) angegeben werden. Durch diese Wertangabe wird erkannt, dass es sich uim einen Wert vom Typ "bool" handelt und für die Übertragung an die Zentrale der korrekte Werttyp eingestellt.So kann dann z.B. auch ein Schaltaktor gesteuert werden.
Beispiel für einen Schataktor:
```
Setzewert(Stehlampe,"STATE","TRUE")
```

Beispiel zum Ausschalten der Bewegungserkennung eines Bewegungsmelders:
```
Setzewert(Bewegungsmelder,"MOTION_DETECTION_ACTIVE","FALSE")
```

Mit SETZEWERT ist es auch möglich die Sirene von HMIP-Rauchmeldern zu schalten.
Zum Einschalten:
```
SETZEWERT(MeinHMIPRauchmelder, "SMOKE_DETECTOR_COMMAND", "INTRUSION_ALARM")
```
Zum Ausschalten:
```
SETZEWERT(MeinHMIPRauchmelder, "SMOKE_DETECTOR_COMMAND",
"INTRUSION_ALARM_OFF")
```

Für neuere Thermostate können über diese Anweisung mehrere Einstellungen vorgenommen werden (gilt nicht für ältere Baureihen).

Der BOOST-Modus wird eingeschaltet mit
```
SETZEWERT(ThermostatName,"BOOST_MODE",1)
```

Der AUTO-Modus wird eingeschaltet mit
```
SETZEWERT(ThermostatName,"AUTO_MODE",1)
```

Der MANU-Modus wird eingeschaltet mit
```
SETZEWERT(ThermostatName,"MANU_MODE",Temperatur)
```
wobei die Temperatur eine Konstante mit einer Kommastelle oder Variable vom Typ Zahl mit einer Kommastelle sein kann.
Die Angabe der Kommastelle ist zwingend erforderlich, also z.B.:
```
SETZEWERT(Wohnzimmer,"MANU_MODE",22.0)
```
Um die Absenk- bzw. Komforttemperatur einzustellen können folgende Anweisungen benutzt werden:

```
SETZEWERT(ThermostatName,"LOWERING_MODE",1)
```
bzw.
```
SETZEWERT(ThermostatName,"COMFORT_MODE",1)
```

Bitte beachten Sie unbedingt:
Diese Anweisung generiert bei jeder Ausführung eine Meldung an das Gerät.
Wenn es sich um ein Funkmodul handelt kann die hemmungslose und unsachgemässe Benutzung dieser Anweisung viele unnötige Funkmeldungen generieren, was dazu führen kann, dass das Duty-Cycle-Konto überlauft und alle Funkmeldungen an Aktoren zeitweise blockiert werden.

1.11.5.26. SETZEZEITTABELLENWERT

SETZEITTABELLENWERT

Syntax :
SETZEZEITTABELLENWERT(*Objektname***)**

Mit dieser Anweisung kann für ein Objekt der Wert aus der Zeittabelle des Objekts eingestellt werden, der nach der Zeittabelle zum letzten Zeitpunkt der Zeittabelle vor der aktuellen Zeit eingestellt worden ist bzw. eingestellt worden wäre falls die Zeittabelle aktiv gewesen wäre.

Diese Anweisung ist z.B. sinnvoll wenn die Zeittabelle eines Objekt ausgeschaltet war und zu einem beliebigen Zeitpunkt wieder eingeschaltet wird. Bei Ausführung dieser Anweisung wird dann der Wert der Zeittabelle im Objekt eingestell, der eingestellt worden wäre wenn die Zeittabelle eingeschaltet gewesen wäre.
Beispiel:

```
ThermostatWohnraum.Zeittabelle einschalten
SETZEZEITTABELLENWERT(ThermnostatWohnraum)
```

1.11.5.27. STARTE

STARTE

Syntax :
STARTE(*OBJEKT***)**

Diese Anweisung stellt das Script des das angegebenen Objekts in die Ausführungswarteschlange. Die Programme der Objekte in der Ausführungswarteschlange werden nach dem FIFO (First In First Out) -Prinzip abgearbeitet. Nach jeder Ausführung eines Programms innerhalb des normalen Ausführungsintervalls werden alle Programme in der Ausführungswarteschlange ausgeführt.
Die Ausführung des Skripts wird also erst gestartet nachdem das aktuelle Script beendet worden ist.

1.11.5.28. STARTPROGRAMM

STARTPROGRAMM

Syntax :
STARTPROGRAMM(*"Programmname Parameter"***)**

Mit dieser Anweisung können Programme oder Betriebssystembefehle der jeweiligen Hardwareplattform der Zentrale aufgerufen werden.
Die Parameter für das Programm müssen dabei je nach Hardwareplattform und Betriebssystem in einfache Hochkomma gesetzt werden.

Beispiel Programm mit Parameter aufrufen:

```
Startprogramm("/usr/local/myprograms/audiostart 'liste=2 mode=7'")
```

1.11.5.29. STARTUHR

STARTUHR

Syntax :
STARTUHR(*Zeitvariable***)**

Die aktuelle Zeit wird in der Zeitvariablen gespeichert. Dies ermöglicht eine Stoppuhrfunktion indem mit der Funktion STOPPUHR(Zeitvariable) die Differenz zwischen gespeicherter und aktueller Zeit ermittelt wird.
Beispiel:
Die Variable Uhr1 ist als Variable vom Typ Uhrzeit definiert.
STARTUHR(Uhr1**)**
Die seit der Ausführung der obigen Anweisung vergangene Zeit kann mit der Anweisung STOPPUHR(Uhr1) ermittelt werden.
siehe auch STOPPUHR

1.11.5.30. STOPPE

STOPPE

Syntax :
STOPPE(*Objektname***)**

Mit dieser Anweisung kann der aktive Verstellvorgang von Rollladen- und Markisensteuerungen beendet werden.

Beispiel:

```
STOPPE(RolloWohnen)
```

1.11.5.31. TEMPERATURABSENKUNG

TEMPERATURABSENKUNG

Syntax :
TEMPERATURABSENKUNG(*Wandthermostatname,Absenktemperatur***)**

Diese Anweisung kann nur im Skript eines Tür/Fensterkontakts oder Drehgriffkontakts verwendet werden.
Damit kann ein Wandthermostat direkt in Abhängigkeit von Tür/Fensterkontakten (oder Drehgriffkontakten) über Skripts sehr einfach gesteuert werden, ohne dass man bei der Erstellung der Anwendung in den Skripts die unterschiedlichen Zustände von mehreren Fensterkontakten für einen Wandthermostat berücksichtigen muss. Mit dieser Anweisung können bis zu 8 Fensterkontakte für einen Wandthermostat verwendet werden. Beim Öffnen eines Fensters wird die in der Anweisung angegebene Absenktemperatur für den Wandthermostat eingestellt. Erst wenn alle Fenster wieder geschlossen sind, wird die zuletzt am Wandthermostat eingestellt Temperatur wieder aktiviert.
Wenn diese Anweisung benutzt wird, darf der Fensterkontakt nicht durch eine direkte Verknüpfung mit dem Wandthermostat verbunden sein und der Modus des Wandthermostats muss auf "manuell" stehen. Andernfalls ist eine Steuerung des Wandthermostats über die Skripts der Fensterkontakte nicht möglich.
Es ist zu beachten, dass der Abgleich von Temperaturen einige Minuten dauern kann.
Wenn die Absenktemperatur eine Kommastelle hat, muss die Temperatur in Hochkommas gesetzt werden.
Diese Anweisung wird einfach als einzelne Zeile im Skript des des Sensors geschrieben (ohne wenn-Bedingung bezüglich des Zustands des Tür/Fensterkontakts !)

Beispiel:

```
TEMPERATURABSENKUNG(Wandthermostat1,"17,5")
```

Bitte beachten Sie:
Bei Fensterkontakten ist meistens die Option "Statusmeldungen senden" in den Einstellungen des Geräts aktiviert.
Wenn "Ausführen bei Empfang" für das Skript des Kontakts aktiviert ist, würde das Skript auch ausgeführt wenn der Zustand des Fensterkontakts sich nicht verändert hat. Das führt dann zu vielen unnötigen Temperaturmeldungen an das Thermostat.
Aus diesem Grunde sollte für das Skript eines Fensterkontakts in dem diese Anweisung verwendet wird die Option "Ausführen bei Änderung" statt "Ausführen bei Empfang" verwendet werden.
Diese Anweisung funktioniert nur korrekt wenn es sich beim Raumthermostat und den Fensterkontakten um HomeMatic-Geräte handelt.

1.11.5.32. VARDEF

VARDEF

Syntax :
VARDEF *Type:Variablenname=Startwert*

Variablendefinition für das aktuelle Objekt.
Mögliche Typen für Variablen sind Zahl, Zeichen, Zeit, Uhrzeit, Datum oder ein Typ mit objektspezifischen Zuständen wie Licht, Schalter usw.
Variablenname dürfen nicht identisch sein mit Anweisungen, Funktionen oder Typen.
Beim Typ Zahl muss ein Startwert angegeben werden, damit wird die Zahl der Kommastellen angegeben. Wenn der Startwert kein Komma beinhaltet wird die Variable als Integer-Zahl ohne Kommastellen definiert und es sind damit keine Berechnungen mit Kommastellen möglich. Wenn kein Startwert angegeben wird, wird als Standardwert 0 eingesetzt, die Variable kann also nur Zahlenwerte ohne Kommastelle beinhalten..
In Visualisierungen wird die Zahl der Nachkommastellen angezeigt, die im Startwert angegeben sind

Beispiel:

VARDEF Zahl:Summe=0,00
VARDEF Zeichen:Zeile1=Ergebnis

In Scripten von anderen Objekten kann auf die Variable zugegriffen werden, indem der Name des Objekts in dem die Variable definiert wurde der Variable getrennt durch einen Punkt vorangestellt wird

Beispiel:

Summe:=Summe + **Objektxy.Wert1**

1.11.5.33. VERLASSEN

VERLASSEN

Syntax :
VERLASSEN

Das aktuelle Skript wird sofort beendet. Die Ausführung wird mit dem nächsten anstehenden Skript fortgesetzt.

Beispiel:

```
WENN Uhrzeit groesser "23:00:00" DANN
VERLASSEN
ENDEWENN
```

1.11.5.34. WARTE

WARTE

Syntax :
WARTE ("*00:00:10*"**)**

Als Parameter für diese Anweisung kann die Zeit in Hochkommas oder eine Zeichen- oder Uhrzeit-Variable verwendet werden.

Beispiel :
Wartezeit:="01:30:00"
WARTE(Wartezeit)
Wartezeit ist im Beispiel eine Variable vom Typ Zahl.

Es kann auch ein konstanter Wert in ganzen Sekunden oder Minuten verwendet werden, dieser Wert kann aber nicht in einer Variablen stehen.
Beipiel:
```
warte 2 Minuten
```

Es kann auch eine Variable vom Typ Zahl als Parameter hinter der WARTE-Anweisung angegebene werden. In dem Fall muss diese die Sekunden beihalten die gewartet werden soll und es darf keine Zeiteinheit angegeben werden.
Beipiel:
```
MeineSekundenZahl:=5
warte MeineSekundenZahl
```
MeineSekundenZahl ist im Beispiel eine Variable vom Typ Zahl

Mit der Anweisung WARTE kann die Ausführung eines Skripts für die angegebene Zeit unterbrochen werden, das Skript wird dabei temporär beendet bis zum Ablauf der Wartezeit. Diese Anweisung hat keine Auswirkung auf andere Skripts und Funktionen, diese laufen weiter. Nach der angegebene Zeit wird die Ausführung mit der nächsten Anweisung hinter der WARTE-Anweisung fortgesetzt.
Wenn ein Skript mit der Anweisung AUFRUFEN als Unterprogramm aus einem anderen Skript aufgerufen wurde, wird das aufrufende Skript fortgesetzt wenn die WARTE-Anweisung ausgeführt wird.

Bitte unbedingt beachten:
Eine WARTE-Anweisung und damit die aktuelle Ausführung des Skripts wird abgebrochen, wenn das Makro vor Ablauf der Wartezeit erneut aufgerufen wird.

1.11.5.35. WENN

WENN

Syntax :
WENN <**NICHT**> *Bedingung* <**UND/ODER**> *Bedingung* **DANN**
Anweisungen
<**SONST**>
Anweisungen
ENDEWENN

Die wenn-Anweisung ist die einzige Anweisung, die sich über mehrere Zeilen erstrecken kann. Der **wenn**-Teil kann in mehreren Zeilen stehen während ***sonst*** und ***endewenn*** –Zeilen jeweils einzelne Zeilen sein müssen.
Mit der WENN-Anweisung ist es möglich den weiteren Programmablauf von einer oder mehreren Bedingungen abhängig zu machen. Wenn-Anweisungen können auch verschachtelt werden, d.h. zwischen dem **wenn** und dem **endewenn** (bzw. sonst) können weitere wenn-Anweisungen stehen. Jede wenn-Anweisung muss mit einer endewenn-Anweisung beendet werden, ansonsten wird bei der Code-Generierung eine entsprechende Fehlermeldung ausgegeben.
Beispiele:

```
WENN LichtBad ausgeschaltet UND
     SCHALTDAUER(LichtBad) groesser "00:05:00" DANN
  Ventilator ausschalten
ENDEWENN

WENN Temperatur < 20.5 dann
  Heizluefter einschalten
SONST
  Heizluefter ausschalten
ENDEWENN

WENN Taste2 = "lang" dann
  Stehlampe ausschalten
SONST
  Stehlampe einschalten
ENDEWENN
```

Vor jeder Bedingung kann ein **NICHT** gesetzt werden, dann wird die Anweisung hinter dann ausgeführt wenn die Bedingung **nicht** zutrifft.
Bitte beachten Sie:
Das Wort NICHT muss **vor** der eigentlichen Bedingung stehen, es darf **nicht in der Bedingung** stehen.

Beispiel:
Falsch wäre die umgangsprachliche Formulierung:
`wenn LichtBad NICHT ausgeschaltet oder Wochentag NICHT = "Montag" dann`
es würde ein Syntaxfehler angezeigt.

Richtig ist:
`wenn NICHT LichtBad ausgeschaltet oder NICHT Wochentag="Montag" dann`

Bitte beachten Sie:
Häufige Syntaxfehler in wenn-Anweisungen sind, dass das Wort ***dann*** vergessen wird und dass die Anweisung nicht mit einem ***endewenn*** abgeschlossen wird.
Da wenn-Anweisungen sich immer über mehrere Zeilen erstrecken, kann die Zeile für einen Syntaxfehler oft nicht bestimmt werden. Wenn in einem Skript mit einer wenn-Anweisung ein Syntaxfehler ohne Fehlerbeschrebung auftritt, prüfen Sie alle Elemente der wenn-Anweisung um den Fehler zu finden.
Zur besseren Übersichtlichkeit sollten SONST und ENDEWENN immer in separaten Zeilen stehen

1.11.6. Funktionen

1.11.6.1. AKTIVIERT

AKTIVIERT

Syntax :
AKTIVIERT(*Objekt***)**

Mit dieser Funktion kann festgestellt werden, ob ein Objekt aktiviert ist. Diese Funktion ist z.B. nützlich wenn ein Programm für besondere Aktionen nur zeitweise aktiviert wird.

Beispiel:

```
wenn AKTIVIERT(Alarmanlage) dann
  Starte(AlleLichterAn)
endewenn
```

1.11.6.2. BATTERIELEER

BATTERIELEER

Syntax :
BATTERIELEER(*Objekt***)**

Mit dieser Funktion kann in Bedingungen geprüft werden, ob ein Modul eine schwache Batterie gemeldet hat.

Die Funktion ist WAHR, wenn eine Batteriewarnung von dem Modul empfangen wurde.
Beispiel:

```
wenn BATTERIELEER(TEMPERATURSENSOR) dann
  Anzeige:="Batterie Temperatursensor austauschen"
endewenn
```

Hinweis:
Um auf Batteriewarnungen zu reagieren kann ein spezielles "Fehler"-Objekt vom Typ Zeichen definiert werden, nähere Infos dazu unter Spezielle Objekte. Jedesmal wenn eine Batteriewarnung empfangen wird, enthält dieses Fehler-Objekt den Namen des Moduls/Objekts, bei dem die Batteriewarnung aufgetreten ist und das Skript des Fehler-Objekts wird ausgeführt.

1.11.6.3. COMERROR

COMERROR

Syntax :
COMERROR(*Objekt***)**

Mit dieser Funktion kann in Bedingungen geprüft werden, ob für ein Modul aktuell eine Kommunikationsstörung gemeldet wurde.

Die Funktion ist WAHR, wenn aktuell eine Kommunikationsstörung existiert, d.h. wenn eine Meldung über eine Kommunikationstörung von der Zentrale empfangen wurde und danach keine weitere Meldung von dem Modul gekommen ist. Sobald eine normale Meldung von dem Modul empfangen wird, wird der Fehlerstatus zurückgesetzt.

Beispiel:

```
wenn COMERROR(Aussensteckdose) dann
  Anzeige:="Kommunikation mit Aussensteckdose immer noch gestört"
endewenn
```

Hinweis:
Um auf Kommunikationsstörungen direkt zu reagieren kann ein spezielles "Fehler"-Objekt vom Typ Zeichen definiert werden, nähere Infos dazu unter Spezielle Objekte. Jedesmal wenn eine Meldung über eine Kommunikationsstörung empfangen wird, enthält dieses Fehler-Objekt den Namen des Objekts für das Modul, für das die Kommunikation gestört ist und das Skript des Fehler-Objekts wird ausgeführt.

1.11.6.4. DATEIVORHANDEN

DATEIVORHANDEN

Syntax :
DATEIVORHANDEN

Mit dieser Funktion kann geprüft werden ob eine Datei existiert.

Beispiel:

```
wenn DATEIVORHANDEN("BEWEGUNGEN.TXT") dann
  Anzeige:="Meldungen vom Bewegungsmelder vorhanden"
endewenn
```

1.11.6.5. DATUM

DATUM

Syntax :
DATUM

Diese Funktion gibt das aktuelle Datum im Format TT.MM.JJJJ zurück.

Beispiel:

```
WENN DATUM =* "15.02.****" UND UHRZEIT=“08:00:00“ DANN
PLAY(”c:\AudioDaten\homeputer\HappyBirthday.wav”)
ENDEWENN
```

1.11.6.6. FEIERTAG

FEIERTAG

Syntax :
FEIERTAG

Diese Funktion hat keine weiteren Parameter und ist in WENN-Bedingungen wahr wenn der aktuelle Tag in der Konfiguration als Feiertag oder als freier Tag definiert ist.

Beispiel:

```
WENN FEIERTAG ODER WOCHENTAG = "Sonntag" DANN
  WECKZEIT:="09:00:00"
ENDEWENN
```

1.11.6.7. GESCHALTET

GESCHALTET

Syntax :
GESCHALTET(*Objekt***)**

Diese Funktion wird ausschließlich in Bedingungen verwendet, um zu prüfen ob ein Objektzustand sich geändert hat.

Beispiel:

```
WENN Alarmsensor eingeschaltet und GESCHALTET(Alarmsensor) DANN
  SENDEMAIL(.........)
  ERLEDIGT(Alarmsensor)
ENDEWENN
```

Sobald sie die Anweisung ERLEDIGT für ein Objekt ausgeführt wurde ergeben die weiteren GESCHALTET-Abfragen das Ergebnis unwahr (also falsch) bis der Zustand des angegebenen Objekts sich erneut verändert. Die Bedingungsabfrage im obigen Skript würde beispielsweise verhindern, dass eine Alarm-mail mehrmals gesendet wird, da die GESCHALTET-Bedingung nur bei der ersten Ausführung des Skripts zutrifft.

siehe auch ERLEDIGT

1.11.6.8. JAHR

JAHR

Syntax :
JAHR

Diese Funktion gibt das aktuelle Jahr als Zahl zurück.

1.11.6.9. MAILLOG

MAILLOG

Syntax :
MAILLOG(*Index***)**

Mit dieser Funktion können die Logs der letzten 5 e-mail-Sendungen abgerufen werden. Der Index 1 der Tabelle enthält immer den Status der zuletzt gesendeten e-mail, der Index 5 den Status der ältesten der letzten fünf mails.
Die Status-Infos für eine mail werden als Text mit drei Zeilen gespeichert. Die Zeilen werden jeweils durch << getrennt und können daher direkt auf ein Anzeigeobjekt vom Typ Zeichen ausgegeben werden.
In der ersten Zeile stehen Datum, Uhrzeit und der Status der Sendung OK=korrekt versandt, bei einem Fehler eine Fehlerkennung und/oder ein Fehlertext.
In der zweiten Zeile steht die Zieladresse der mail.
In der dritten Zeile steht der Betrefftext der mail.
Die Fehlerkennungen haben folgende Bedeutungen:
999 : Einer der folgenden Werte ist ungültig oder leer: Mail-Server, Absenderadresse, Zieladresse, Betreff, Mailtext.
andere : Ursache im Text angegeben

Beispiel:

```
Anzeige1 := MAILLOG(1)
```

Anzeige1 hätte danach folgenden z.B. Textinhalt:
02.06.2008 12:23:20 OK<<Meine.BueroAdresse@Mein-Provider.com<<Temperaturinfo

Auf dem Anzeigepanel des Objekt Anzeige1 würde dann Meldung erscheinen:
02.06.2008 12:23:20 OK
Meine.Bueroadresse@Mein-Provider.com
Temperaturinfo

1.11.6.10. MONAT

MONAT

Syntax :
MONAT

Diese Funktion gibt den aktuellen Monat als Zahl zurück.

Beispiel:

```
WENN Monat = 5 DANN
  Anzeige1:="Der Mai ist gekommen.."
ENDEWENN
```

1.11.6.11. MONATSTAG

MONATSTAG

Syntax :
MONATSTAG

Diese Funktion gibt den aktuellen Tag des Monats als Zahl zurück.

Beispiel:

```
WENN MONATSTAG=1 DANN
  Anzeige1:=“Ein neuer Monat hat begonnen“
ENDEWENN
```

1.11.6.12. OBJEKTBEZ

OBJEKTBEZ

Syntax :
OBJEKTBEZ(*Objekt***)**

Diese Funktion gibt die Bezeichnung des angegebenen Objekts als Text zurück und dient dazu diese anzeigen zu können oder z.B. als Mail verschicken zu können.

Beispiel:

```
Sendezeile:="Das ist eine mail von Objekt  " + OBJEKTBEZ(selbst)
SENDEMAIL("Status", Sendezeile, 2)
```

1.11.6.13. OBJEKTNAME

OBJEKTNAME

Syntax :
OBJEKTNAME(*Objekt***)**

Diese Funktion gibt den Namen des angegebenen Objekts als Text zurück und dient dazu diesen anzeigen zu können oder als Mail verschicken zu können.

Beispiel:

Sendezeile:="Das ist eine mail von Objekt " + OBJEKTNAME(selbst)
```
SENDEMAIL("Status", Sendezeile, 2)
```

1.11.6.14. LÖSCHEDATEI

LÖSCHEDATEI

Syntax :
LÖSCHEDATEI(*"Dateiname"***)**

Mit dieser Anweisung können Dateien gelöscht werden. Das können selbst erstellte Textdateien, aber auch beliebige andere Dateien wie die Historydatei **history.txt** oder die Systemlog-Datei **syslog.txt** sein.

Bitte beachten Sie unbedingt:
Durch das beabsichtigte oder unbeabsichtigte Löschen von Dateien, die nicht selbst erstellt wurden, kann die Systemintegrität zerstört und die Funktionen der Zentrale beeinträchtigt werden.
Löschen Sie daher keinesfalls Dateien, von denen Sie nicht wissen wofür diese benutzt werden.
Achten Sie unbedingt darauf, dass das auch nicht versehentlich, etwa durch Angabe eines falschen Dateinamens, geschieht.

Beispiel:

```
LÖSCHEDATEI("BEWEGUNGEN.TXT")
```

1.11.6.15. SCHALTDAUER

SCHALTDAUER

Syntax :
SCHALTDAUER(*Objekt***)**

Diese Funktion gibt die Zeit zurück, in der das Objekt sich in seinem aktuellen Zustand befindet. Es wird die reine Uhrzeitdifferenz zwischen letzter Schaltzeit und aktueller Uhrzeit ermittelt ermittelt, Tage werden nicht berücksichtigt, d.h. als Ergebnis kann es keinen Wert grösser als 23:59:59 geben.

1.11.6.16. SELBST

SELBST

Syntax :
SELBST

Diese Funktion gibt das Objekt des Skripts zurück, in dem sie verwendet wird.
Damit ist es möglich das aktuelle Objekt im Skript zu verwenden ohne den Objektnamen schreiben zu müssen.
Einer der Vorteile der Funktion SELBST ist, dass das Skript einfach in ein anderes Objekt kopiert werden kann, ohne das Änderungen bezüglich des Objektnamens im Skript erforderlich sind.

Beispiel:

```
SELBST umschalten
Anzeige:=Objektbez(SELBST) + " ist jetzt " + SELBST
```

Im Objekt Anzeige vom Typ Zeichen würde dann z.B. Folgendes angezeigt:
Stehlampe Wohnzimmer ist jetzt an

1.11.6.17. SONNENAUFGANG

SONNENAUFGANG

Syntax :
SONNENAUFGANG

Mit dieser Funktion kann die Uhrzeit des Sonnenaufgangs für den aktuellen Tag ermittelt werden. Da diese Zeit vom geografischen Standort abhängig ist, kann auf der Seite *Einstellungen* das Postleitzahlgebiet angegeben werden. Durch Mausklick auf die Schaltfläche [Eingabe Koordinaten] können auch die exakten Koordinaten des Standorts eingegeben werden. Die Berechnung erfolgt durch astronomische Näherungsformeln, es wird die sogenannte „bürgerliche Dämmerung“ berechnet. Je nach Jahreszeit und geographischem Standort kann es zu einer Abweichung von einigen Minuten kommen.

Beispiel:

```
wenn Uhrzeit = SONNENAUFGANG dann
  RollladenSchlafzimmer hochfahren
endewenn
```

Hinweis:
Das Skript im Beispiel muss mindestens 1 x in der Minute ausgeführt werden, da die Zeit des Sonnenuntergangs immer volle Minuten hat. Das muss bei der Einstellung des Ausführungsintervalls beachtet werden für Skripts in denen wenn-Bedingungen mit Sonnenzeiten benutzt werden.
siehe auch SONNENUNTERGANG

1.11.6.18. SONNENUNTERGANG

SONNENUNTERGANG

Syntax :
SONNENUNTERGANG

Mit dieser Funktion kann die Uhrzeit des Sonnenuntergangs für den aktuellen Tag ermittelt werden. Da diese Zeit vom geografischen Standort abhängig ist, kann auf der Seite *Einstellungen* das Postleitzahlgebiet angegeben werden. Durch Mausklick auf die Schaltfläche [Eingabe Koordinaten] können auch die exakten Koordinaten des Standorts eingegeben werden. Die Berechnung erfolgt durch astronomische Näherungsformeln, es wird die sogenannte „bürgerliche Dämmerung“ berechnet. Je nach Jahreszeit und geographischem Standort kann es zu einer Abweichung von einigen Minuten kommen.

Beispiel:

```
wenn Uhrzeit = SONNENUNTERGANG dann
  RollladenSchlafzimmer runterfahren
endewenn
```

Hinweis:
Das Skript im Beispiel muss mindestens 1 x in der Minute ausgeführt werden, da die Zeit des Sonnenuntergangs immer volle Minuten hat. Das muss bei der Einstellung des Ausführungsintervalls beachtet werden für Skripts in denen wenn-Bedingungen mit Sonnenzeiten benutzt werden.
siehe auch SONNENAUFGANG

1.11.6.19. STOPPUHR

STOPPUHR

Syntax:
STOPPUHR(Zeitvariable**)**

Diese Funktion gibt die Zeitdauerdifferenz zwischen der aktuellen Uhrzeit und der in der Zeitvariablen hinterlegten Zeit im Format HH:MM:SS zurück. Es wird die reine Uhrzeitdifferenz ermittelt, Tage werden nicht berücksichtigt, d.h. als Ergebnis kann es keinen Wert grösser als 23:59:59 geben.

Beispiel:
Das Treppenhauslicht soll ausgeschaltet werden, wenn es länger als 2 Minuten an ist und der Taster zum Einschalten länger als 1 Minute nicht mehr betätigt wurde. Beachten Sie, daß als Parameter bei der Ermittlung der Stoppzeiten die Variable CT des Objekts und nicht das Objekt selbst verwendet wird.

```
wenn LiTreppenh eingeschaltet dann
  wenn Stoppuhr(LiTreppenh.ct) groesser "00:02:00" und
       Stoppuhr(TastTreppenh.ct) groesser "00:01:00" dann
    LiTreppenh aussschalten
  endewenn
endewenn
```

Bitte beachten Sie:
Als Parameter darf nicht der Objektname, sondern es muss eine Zeitvariable oder die für jedes Objekt vorhandenen Zeitvariable CT (also *Objektname***.CT**) angegeben werden. Es gibt auch noch eine andere Funktion um festzustellen wie lange ein Objekt sich in seinem aktuellen Zustand befindet, das ist die Funktion SCHALTDAUER(Objekt), bei dieser Funktion wird das Objekt (ohne den Zusatz.CT) als Parameter angegeben.

1.11.6.20. TAUPUNKT

TAUPUNKT

Syntax:
TAUPUNKT(Temperatur, Luftfeuchte**)**

Mit dieser Funktion kann die Taupunkttemperatur ermittelt werden.
Wenn die Taupunkttemperatur unterschritten wird, verflüssigt sich der in der Luft enthaltene Wasserdampf.
Es kommt an Wänden, Fenstern usw. zur Kondenswasserbildung, wenn dort die Taupunkt-Temperatur unterschritten wird.
Die entstehende Feuchtigkeit kann dann z.B. zu Schimmelbildung führen. Um das zu vermeiden kann durch Änderung der Temperatur bzw. der Luftfeuchtigkeit (z.B. durch Lüften) der Taupunkt verändert werden, damit er unter der Temperatur z.B. der Wände eines Raums liegt.

Beispiel zur Ermittlung des Taupunkt mit den Temperatur- und Luftfeuchtewerten eines Wandthermostats:

```
TaupunktKeller1:=Taupunkt(WTKeller1.Temperatur, WTKeller1.Luftfeuchtigkeit)
```

1.11.6.21. TRIGGER

TRIGGER

Syntax:
TRIGGER(Objekt**)**

Mit dieser Funktion kann die Ursache der letzten Wertänderung des Objekts bestimmt werden.
Der Rückgabewert ist ein Zahlencode mit folgender Bedeutung:
1=Skript
2=Hardware
3=Hardware-Status nach Abfrage
4=Zeittabelle
5=TimeControl (nur c-comatic mit Zeitsteuerungsmodul)
6=Fensterkontakt über Anweisung TEMPERATURABSENKUNG
7,8=extern über App, Webserver, Web-Schnitttselle, XMLRPC-Schnittstelle usw.
9=LESEWERTEDATEI
10=binäre Temperatursteuerung (für beim Thermostat eingetragene Schaltakaktoren)
ab 11 weitere Visualisierungen

Mit dieser Funktion ist es möglich in einem Skript unterschiedliche Aktionen in Abhängigkeit von der Ursache der aktuellen Wertänderung eines Objekts auszuführen.
Wenn die Option "Ausführen bei Änderung" aktiviert ist, kann so ermittelt werden, was die aktuelle Skriptausführung verursacht hat um ggfs. unterschiedliche Programmierungen je nach Ursache der Wertänderung vorzunehmen.

Beispiel:

```
wenn TRIGGER(selbst) = 4 dann
  Anzeige:="Wert-Änderung durch Zeittabelle"
endewenn
```

Bitte beachten Sie:
Wenn der Objektwert eines Hardware-Objekts geändert wird, melden die meisten Hardwaremodule den aktuellen Status zurück, nachdem sie auf den neuen Wert umgeschaltet haben. Da eine solche Hardwaremeldung nicht von einer Meldung durch tatsächliche Bedienung der Hardware zu unterscheiden ist, wird der aktuelle Triggercode je nach Hardware eventuell nur Sekundenbruchteile nachdem eine Änderung z.B. durch eine Skriptanweisung erfolgte auf "Hardware" (also 2) geändert.
Bis der Triggercode durch diese Rückmeldung von der Hardware wieder geändert wird, gibt die Funktion den Triggercode zurück, der die Hardwareschaltung verursacht hat, kann also z.B. im Skript des geschalteten Objekts benutzt werden um die Ursache der aktuellen Wertänderung zu ermitteln.

1.11.6.22. UHRZEIT

UHRZEIT

Syntax :
UHRZEIT

Diese Funktion gibt die aktuelle Uhrzeit im Format HH:MM:SS zurück. Die Uhrzeit wird im 5-Sekunden-Takt aktualisiert, die Sekunden der Uhrzeit sind also immer durch 5 teilbar.

Beispiel:

```
WENN UHRZEIT groesser "23:00:00" DANN
AKTZEIT:=UHRZEIT
ENDEWENN
```

1.11.6.23. UHRZEITSEKUNDE

UHRZEITSEKUNDE

Syntax :
UHRZEITSEKUNDE

Diese Funktion gibt die aktuelle Uhrzeit im Format HH:MM:SS zurück. Die Sekunden sind aktuell und nicht wie in der Funktion UHRZEIT im 5-Sekundenintervall.

1.11.6.24. WOCHENTAG

WOCHENTAG

Syntax :
WOCHENTAG

Diese Funktion gibt den aktuellen Wochentag als Text zurück. Bei Vergleichsbedingungen mit dieser Funktion muss die Groß/Kleinschreibung beachtet werden.

Beispiel:

```
WENN WOCHENTAG = "Montag" DANN
AKTTAG:=WOCHENTAG
ENDEWENN
```

1.11.6.25. ZEIT

ZEIT

Syntax :
ZEIT

Diese Funktion ergibt die Zeit in einem internen Format. Dieses Format wird z.B. verwendet um die Stoppzeit zu ermitteln.

Beispiel:
Eine Zeitvariable kann mit der Funktion Zeit gefüllt werden. Die Variable UHR1 muss vom Typ *Zeit* sein.
Nachdem die Anweisung

```
UHR1:=ZEIT
```

ausgeführt wurde kann später mit der Funktion

```
STOPPUHR(UHR1)
```

die seit der Ausführung der ersten Anweisung vergangene Zeit ermittelt werden.
Die Anweisung UHR1:=ZEIT hat die gleiche Wirkung
wie die Anweisung STARTUHR(UHR1).
Weiterhin ist der Typ Zeit nützlich um die komplette Zeit (also Datum und Uhrzeit zu merken und auszugeben. Wenn eine Variable dieses Typs einem Zeichen-Objekt zugewiesen wird, werden Datum und Uhrzeit als Text ausgegeben.

1.11.6.26. ZUFALLSZEIT

ZUFALLSZEIT

Syntax :
ZUFALLSZEIT(Zeitvariable**)**

Generiert mit einem Zufallsgenerator eine Zeit bis maximal zum in der Zeitvariablen angegebenen Wert. Die Zeitvariable muss den Typ Uhrzeit haben (nicht den Typ Zeit, da dieser auch das Datum beinhaltet). Der maximal zulässige Wert ist ”23:59:59”. Es wird immer eine durch 5 teilbare Sekundenzahl zurückgegeben.
Beispiel:
Ein Programm zur Anwesenheitssimulation während des Urlaub soll das Licht zwischen 20 und 21 Uhr einschalten und zwischen 22 und 23:30 Uhr ausschalten.
Die Ein- und Ausschaltzeiten werden am besten von einem Programm, welches bei Tageswechsel läuft festgelegt.
Dies könnte so aussehen:

```
Einschaltzeit:="20:00:00"+Zufallszeit("01:00:00")
```
** *(generiert eine Zeit zwischen 20 und 21 Uhr)*

```
Ausschaltzeit:= "22:00:00"+Zufallszeit("01:30:00")
```
** *(generiert eine Zeit zwischen 22 und 23:30 Uhr)*

Verwendet werden können diese Zeiten dann in einem Skript um z.B. Beleuchtungen zu schalten.
Ein solches Skript könnte so aussehen:

```
wenn Uhrzeit=Einschaltzeit dann
  Licht einschalten
endewenn
wenn Uhrzeit=Ausschaltzeit dann
  Licht ausschalten
endewenn
```

1.11.6.27. Funktionen zur Textbearbeitung

1.11.6.27.1. ERSETZEN

ERSETZEN

Syntax :
ERSETZEN(*Zeile,Alt,Neu***)**

Diese Funktion ersetzt den Text des Parameters "Alt" in der Zeile durch den Text des Parameters "Neu".
"Zeile" ist ein Objekt/eine Variable vom Typ Zeichen, "Alt" und "Neu" sind Variablen/Objekte vom Typ Zeichen oder Konstanten in Hochkommas.

Beispiel:

```
ZEILE:="Es ist jetzt dunkel im Zimmer"
ZEILE:=ERSETZEN(ZEILE,"dunkel","hell")
```

Nach Ausführung hat sich die Zeile geändert auf :"Es ist jetzt hell im Zimmer".

1.11.6.27.2. GROSSBUCHSTABEN

GROSSBUCHSTABEN

Syntax :
GROSSBUCHSTABEN(*Text***)**

Diese Funktion wandelt alle Kleinbuchstaben des Parameters Text in Grossbuchstaben um. Text kann eine Variable oder Konstante vom Typ Zeichen sein.

Beispiel:

```
GROSSERTEXT:=GROSSBUCHSTABEN(ZEILE)
```

1.11.6.27.3. KLEINBUCHSTABEN

KLEINBUCHSTABEN

Syntax :
KLEINBUCHSTABEN(*Text***)**

Diese Funktion wandelt alle Grossbuchstaben des Parameters Text in Kleinbuchstaben um. Text kann eine Variable oder Konstante vom Typ Zeichen sein.

Beispiel:

```
KLEINERTEXT:=KLEINBUCHSTABEN(ZEILE)
```

1.11.6.27.4. LESETEXTPAR

LESETEXTPAR

Syntax :
LESETEXTPAR(*Text,Index,Trennzeichen***)**

Diese Funktion gibt den Textteil zurück, der im durch Index angegebenen Teil des Textes steht wobei die einzelnen Textteile durch das im Parameter *Trennzeichen* angegebene einzelne Zeichen getrennt werden.

Beispiel:

```
ZEILE:="Teil1/Teil2/Teil3/Teil4"
PAR3:=LESETEXTPAR(ZEILE,3,"/")
```

In der Variablen PAR3 würde nach der Ausführung "TEIL3" stehen.

1.11.6.27.5. LINKERTEIL

LINKERTEIL

Syntax :
LINKERTEIL(*Text,Anzahl***)**

Diese Funktion gibt die angegebene Anzahl Buchstaben der linken Seite des Textes zurück.
Text ist eine Variable des Typs Zeichen, Anzahl ist eine Variable des Typs Zahl ohne Kommastellen oder eine Konstante.

Beispiel:

```
ZEILE:="Das ist eine Zeile"
LINKS:=LINKERTEIL(ZEILE,3)
```

In der Textvariablen LINKS würde nach der Ausführung nur das Wort "Das" stehen.

1.11.6.27.6. RECHTERTEIL

RECHTERTEIL

Syntax :
RECHTERTEIL(*Text,Anzahl***)**

Diese Funktion gibt die angegebene Anzahl Buchstaben der rechten Seite des Textes zurück.
Text ist eine Variable des Typs Zeichen, Anzahl ist eine Variable des Typs Zahl ohne Kommastellen oder eine Konstante.

Beispiel:

```
ZEILE:="Das ist eine Zeile"
RECHTS:=RECHTERTEIL(ZEILE,5)
```

In der Textvariablen RECHTS würde nach der Ausführung nur das Wort "Zeile" stehen.

1.11.6.27.7. TEXTLÄNGE

TEXTLÄNGE

Syntax :
TEXTLÄNGE(*Text***)**

Diese Funktion gibt die Anzahl der Zeichen zurück, die in der angegebenen Textzeile enthalten sind.

Beispiel:

ZEILE:="abcde"
ANZAHL:=**TEXTLÄNGE(ZEILE)**

In der Variablen ANZAHL vom Typ Zahl würde nach der Ausführung die Zahl 5 stehen weil der Text 5 Zeichen lang ist.

1.11.6.27.8. TEXTPOSITION

TEXTPOSITION

Syntax :
TEXTPOSITION(*Textteil,Text***)**

Diese Funktion gibt die Position eines Textteils innerhalb einer Textzeile zurück. Das Ergebnis ist vom Typ Zahl ohne Kommastellen.
Textteil ist ein Objekt/eine Variable vom Typ Zeichen oder eine Konstante in Hochkommas, Text eine Variable oder ein Objekt vom Typ Zeichen.

Beispiel:

```
ZEILE:="Wo ist denn das XY in dieser Zeile?"
XYPOS:=TEXTPOSITION(ZEILE,"XY")
```

In der Variablen XYPOS würde nach der Ausführung die Zahl 16 stehen, weil das "X" von "XY" an der 16. Stelle der Zeile steht.

1.11.6.27.9. TEXTTEIL

TEXTTEIL

Syntax :
TEXTTEIL(*Zeile,Pos,Anzahl***)**

Diese Funktion ermittelt einen Textabschnitt aus der Zeile, beginnend an der Stelle "Pos" mit der Anzahl im Parameter "Anzahl" angegebenen Buchstaben.
Zeile ein Objekt/eine Variable vom Typ Zeichen, "Pos" und "Anzahl" sind Variablen/Objekte vom Typ Zahl ohne Kommastellen oder Konstanten.

Beispiel:

```
ZEILE:="Das xyz wir herausgeholt"
NEUERTEXT:=TEXTTEIL(ZEILE,5,3)
```

In der Variablen NEUERTEXT würde nach der Ausführung "xyz" stehen.

1.11.6.27.10. TEXTZWISCHEN

TEXTZWISCHEN

Syntax :
TEXTZWISCHEN(*Text,Textteil1,Textteil2***)**

Diese Funktion gibt den Textteil zurück, der zwischen den in den Parametern Textteil1 und Textteil2 angegebenen Zeichen steht.

Beispiel:

```
ZEILE:="Breite=200cm,Farbe=rosarot;Gewicht=53kg;"
Farbe:=TEXTZWISCHEN(ZEILE,"Farbe=",";")
```

In der Variablen Farbe würde nach der Ausführung "rosarot" stehen. Mit dieser Anweisung ist es z.B. möglich auf einfache Art bestimmte Werte einer zuvor mit GETSITE eingelesenen Internetseite auszuwerten.

1.12. Hinweise zur Hardware

Hinweise zur Hardware

In diesem Abschnitt werden Besonderheiten zur Behandlung von System und speziellen Modulen beschrieben.
Es werden nicht alle Module aufgeführt, sondern nur die, bei denen die Ansteuerung und bzw. das Verhalten vom Standard abweicht.

1.12.1. Datenpunkte von Geräten ändern

Datenpunkte von Geräten anlegen / ändern

Manchmal kann es erforderlich werden die Art wie Meldungen von der Hardware kommen bzw. wie diese an die Hardware-Geräte gesendet werden zu ändern.
Das ist z.B. bei HomeMatic erforderlich die Eigenschaften des Hardware-Geräts über die EB-UI geändert werden, z.B. von Taster auf Schalter.
Bei manchen Geräten wird diese Änderung automatisch vorgenommen wenn der Typ des Kanals oder der Objekttyp geändert wird aber es gibt auch einige Geräte bei denen das durch manuelle Änderung gemacht werden muss.

Das passiert dann bei der Objektverwaltung im unteren Abschnitt, der mit der Auswahlbox *Bus* beginnt.
Diese Anpassungen sind nur möglich, wenn es sich um ein Hardwareobjekt handelt, zu dem entsprechende Daten verwaltet werden können, andernfalls gibt es auf der Objektseite keine entsprechenden Eingabefelder.
Änderungen in diesem Bereich sind nicht ungefährlich, wenn dort falsche Einstellungen gemacht werden, kann es passieren, dass der Zugriff auf die Hardware nicht mehr möglich ist und das entsprechende Modul gelöscht und neu importiert werden muss.
Damit nicht versehentlich Änderungen in diesem Bereich gemacht werden muss der Zugriff über den Button *Freigabe* freigeschaltet werden.

Um z.B. beim HomeMatic-System den Typ eines Eingangs von *Taster* auf *Schalter* zu ändern muss der Name des Datenpunkts von **PRESS** auf **STATE** geändert werden.

Bei Austausch eines Moduls des HomeMatic-Systems mit einem Modul desselben Typs kann die Adresse des Moduls hier geändert werden.

Wenn Module des HomeMatic-Systems nicht importiert, sondern manuell angelegt wurden kann die Adresse hier nachträglich eingetragen werden.

1.12.2. HomeMatic

HomeMatic

HomeMatic-Geräte erkennt man daran, dass die Typbezeichnung mit HM-.. beginnt.
Hier werden Hinweise zu speziellen HomeMatic-Geräten beschrieben.

Bei den HomeMatic und HomeMatic IP - Geräten, bei denen die Eingänge umkonfigurierbar sind ist folgendes zu beachten:

Die aktuelle Konfiguration des Eingangstyps wird beim Import nicht automatisch eingestellt, es muss im Programm jeweils der passende Objekttyp ausgewählt werden.

Der Eingangstyp *Schalter* des HomeMatic-Geräts entspricht nicht wirklich einem Schalter, sondern einem Taster. Bei jeder Aktion, egal ob der Kontakt geschlossen oder geöffnet wird, wird jeweils die Meldung *PRESS_SHORT* gesendet. In der Software muss daher in dem Fall der Objekttyp *Taster* ausgewählt werden.
Wenn an den Eingang des Geräts ein Schalter angeschlossen wird, bei dem der aktuelle Zustand gesendet werden soll, muss am Gerät in der WEB-UI unter *Kanalverhalten* bzw. *Eingangstyp* **Tür/Fensterkontakt** oder **Schliesserkontakt** ausgewählt werden.

1.12.2.1. Rollladen/Jalousiesteuerungen

Rollladen/Jalousiesteuerungen

Die Laufzeiten für Rollladen-, Jalousie- und Markisensteuerungen werden in der Konfiguration über die WEB-Oberfläche der Zentrale angegeben. Es sollte niemals eine kleinere Motorumschaltzeit als 0,5 Sekunden (besser 1 Sekunde) angegeben werden, da sonst die Gefahr besteht, dass die Relais des Aktors beschädigt werden.

Im Folgenden wird für Rollladen- Jalousie und Markisensteuerungen nur der Begriff Rolllade verwendet.

Beim Import bzw. der Erstellung wird für diese Steuerungen der Objekttyp *Jalousie* verwendet. Die Stellungen der Rollladen und Jalousien (mit Lamellen falls vorhanden) wird dann in Prozenten von 0 bis 100 angegeben, wobei der Wert 0 für komplett geschlossen und der Wert 100 für komplett geöffnet verwendet wird.
Der Objekttyp Jalousie ist der am besten geeigntete Objekttyp für diese Steuerungen.

Um eine Rolllade mit dem Objekttyp **Jalosie** zu öffnen würde im Skript die beispielsweise folgende Anweisung verwendet:
```
RollladeKueche:=100
```
Um eine Rolllade zur Hälfte zu öffnen würde beipielsweise folgende Anweisung verwendet:
```
RollladeKueche:=50
```

Um den Zustand einer Rolllade abzufragen wird die Prozentzahl verwendet, also z. B.:
```
wenn RollladeDiele > 90 dann
  Anzeige:="Rolllade Diele ist offen"
endewenn
```

Wenn es sich um einne Jalousieaktor handelt, der auch Lamellen steuern kann, gibt es in dem Objekt eine zusätzliche Variable *Lamellen*, die zur Steuerung der Lamellenstellung benutzt wird. Auch hier werden Prozentzahlen von 0 - 100 benutzt.

Alternativ zum Objekttyp *Jalousie* kann für Rollladen auch der Objekttyp Rolllade oder Rolllade1 benutzt werden.
Bei diesen Objekttypen werden nicht Zahlen, sondern Zustände verwendet, die aber keine so genaue Steuerung wie der Typ Jalousie ermöglichen.
Die Steuerung von Rollladen mit den Rollladentypen kann im Skript mit den Schlüsselwörtern **runterfahren** und **rauffahren** geschehen.

also z.B.
RollladeWohnen runterfahren
Um in Zwischenstellungen zu fahren, wird die jeweilige Zwischenstellung direkt als Wert zugewiesen.
also z.B.
RollladeWohnen:="halb"
Beachten Sie dabei bitte, dass die gewünschten Zwischenstellung in Hochkomma gesetzt werden und exakt dem Wort des Zustands entsprechen muss (auch Gross-/Kleinschreibung beachten). Die genauen Wörter für die jeweiligen Positionen werden auf der Seite "Visualisierung" der Objektdefinitionen angezeigt.
Der Zustand einer Rolllade kann mit den Zustandstexten abgefragt werden. Schlüsselwörter (z.B. geöffnet, geschlossen) können für Rollladen **nicht** verwendet werden.

Richtig wäre also z.B.

```
wenn RollladeKueche = "oben" dann
  AnzeigeKueche:="offen"
endewenn
oder
wenn RollladeKueche = "unten" dann
```

```
  AnzeigeKueche:="geschlossen"
endewenn
```

oder

```
wenn RollladeKueche = "dreiviertel" dann
   AnzeigeKueche:="3/4 geschlossen"
endewenn
```

Für Rollladen gibt es eine zusätzliche spezielle Anweisung zum Stoppen eines Verstellvorgangs. Dies passiert mit der Anweisung
STOPPE(*Rolladexx*).
Bitte beachten Sie:
Nach einem Fahrbefehl an eine Rolllade wird der neue Zustand in der Visualisierung für einen Sekundenbruchteil angezeigt, danach wird jeweils der Zustand angezeigt, der von dem Steuerungsmodul aktuell gesendet wird. Das ist normalerweise erst einmal der ursprüngliche Zustand, erst nach Abschluss der Verstellvorgangs wird der neue Zustand empfangen und angezeigt.

1.12.2.2. HM-Thermostate

HM-Thermostate

Über eine kurze Betätigung der Menü-Taste des Thermostats kann der Modus eingestellt werden.Empfehlenswert ist der Modus "Manu". In diesem Modus kann das Thermostat über die Zentrale eingestellt werden, aber es sind auch manuelle Veränderungen der Temperatur am Stellrad möglich.
Damit ein Thermostat ausschliesslich über die Zentrale gesteuert wird, kann der Modus auf "Cent" gesetzt werden.
Weitere Infos dazu finden Sie in der Bedienungsanleitung des Thermostats.
Zur Visualisierung des Thermostats wird ein spezielles Panel benutzt.
Die Temperatur auf der linken Seite des Panels ist die Solltemperatur, die über die Visualisierung verändert werden kann. Die Temperatur auf der rechten Seite des Panels ist die aktuelle Ist-Temperatur, die vom Thermostat empfangen wurde. In der Mitte wird die aktuelle gemessene Luftfeuchtigkeit angezeigt, bei Heizkörperthermostaten die aktuelle Ventilstellung.
Die Darstellung von Luftfeuchtigkeit und Ist-Temperatur kann in der Definition des Thermostat-Panels ausgeschaltet werden.

Die Solltemperatur wird in den Skripts direkt über den Namen des Thermostats eingestellt bzw. abgefragt.
Beispiel:

```
WandthermostatWohnen:=21,5
```

oder

```
WandthermostatWohnen setzen auf 21,5
```

bzw.

```
wenn WandthermostatWohnen > 23 dann
```

Bei Zuweisungen der Soll-Temperatur ist zu beachten, dass diese grundsätzlich in 0,5 Grad Schritten anzugeben ist und im min/max-Bereich des jeweiligen Geräts liegt, dieser kann je nach Thermostat unterschiedlich sein. Wenn Format oder Wertebereich nicht stimmen wird der Wert vom Gerät nicht akzeptiert.

Es können auch die Werte Off und On über Skripts im Raumthermostat gesetzt werden, indem für Off der Wert 0, für On der Wert 100 zugewiesen wird.
also z.B.

```
WandthermostatWohnen:=100
```

Der Wert **Off** schliesst das Ventil vollständig, der Wert **On** öffnet das Ventil vollständig. Bei beiden Modi wird die interne Zeittabelle des Thermostats deaktiviert.

Die Ist-Temperatur steht in Skripts als Variable *Temperatur* des Raumthermostats zur Verfügung, die Luftfeuchtigkeit als Variable *Luftfeuchtigkeit*,
also z.B.

```
wenn Wandthermostat.Temperatur > 22.5 dann
....
```

Wenn das Wandthermostat-Panel angeklickt wird, öffnet sich ein Fenster in dem die Solltemperatur verändert werden kann.
Bitte beachten Sie:
Es gibt Thermostate älterer Baureihen, die nur im Abstand von einigen Minuten kommunikationsbereit sind, bei diesen Thermostaten kann es Verzögerungen von einigen Minuten geben bis geänderte Werte übertragen und angezeigt werden.

Variablen eines Wandthermostats
Ein Wandthermostat hat einige Variablen, diese können je nach Typ des Wandthermostat unterschiedlich sein, bei manchen Typen stehen nicht alle Variablen zur Verfügung.
Hier eine Liste der Variablen:

Variablennname	Verwendung	Typ
Temperatur	Ist-Temperatur	Zahl

```
Luftfeuchtigkeit   Luftfeuchtigkeit              Zahl
Batteriestatus     aktuelle Batteriespannung     Zahl
Modus              Betriebsmodus                 Zahl 0=AUTO, 1=MANUELL
Der Modus kann nicht verändert werden, dazu muss SETZEWERT benutzt werden
(s. unten)
Zeittabelle        Zeittabelle (de)aktivieren    Schalter ein/aus
Heizung            Steuerung Heizkaktor ein/aus  Schalter
Kuehlung           Steuerung Kühlaktor ein/aus   Schalter
TCTemparatur       nur zur internen Verwendung
```

Verwendung der Anweisung **SETZEWERT** bei HM-Thermostaten (gilt nicht für HmIP-Thermostate):
Für das Heizkörperthermostat HM-CC-RT-DN und das Wandthermostat HMTC-IT-WM-W-EU können über die Anweisung SETZEWERT mehrere Einstellungen vorgenommen werden.
Der BOOST-Modus wird eingeschaltet mit

```
SETZEWERT(ThermostatName,"BOOST_MODE",1)
```

Der AUTO-Modus wird eingeschaltet mit

```
SETZEWERT(ThermostatName,"AUTO_MODE",1)
```

Der MANU-Modus wird bei HM-Thermostaten mit

```
SETZEWERT(RaumthermostatName,"MANU_MODE",17.0)
```

gesetzt, wobei die Temperatur eine Konstante mit Kommastelle oder Variable vom Typ Zahl mit Kommastelle sein kann.
(Gilt nicht für HMIP, dazu siehe siehe Hinweise unter HmIP-Thermostaten!)
Um die Absenk- bzw. Komforttemperatur einzustellen können folgende Anweisungen benutzt werden:

```
SETZEWERT(ThermostatName,"LOWERING_MODE",1)
```

bzw.

```
SETZEWERT(ThermostatName"COMFORT_MODE",1)
```

Temperatursteuerung mit Wandthermostaten über Schaltaktoren:
Im Verwaltungsprogramm für Wandthermostate können binäre Aktoren zur Steuerung von Heiz- und Kühl-Geräten angegeben werden. Die über die Aktoren gesteuerten Geräte werden in Abhängigkeit der Soll- und Ist-Temperaturen direkt geschaltet. Dabei können über spezielle Profile die Temperatur-Grenzwerte zum Schalten der Aktoren den individuellen Gegebenheiten angepasst werden. Die Einstellungen der Profile kann mit dem Button unterhalb der Auswahlfelder für die Aktoren aufgerufen werden.
Mit diese Methode der Steuerung und der Möglichkeit eigene Heizprofile zu erstellen können auch träge reagierende Fussbodenheizungen individuell gesteuert werden.
Bitte beachten Sie:
Die Steuerung von Fussbodenheizungen über die direkte Verknüpfung von Wandthermostat und Schaltaktor ist normalerweise nicht möglich, da der hierbei verwendete Regelalgorithmus für Fussbodenheizungen nicht geeignet ist.
Es können unterschiedliche Aktoren zur Heizung und Kühlung verwendet werden, je nach System kann für Heizen und Kühlen aber auch derselbe Schaltaktor benutzt werden.
Dabei ist folgendes zu beachten:
Die Funktionen zur Heizung und Kühlung können standardmässig nicht gleichzeitig aktiv sein.
Die Steuerung welche Funktion aktiv sein soll erfolgt über die Variablen Heizung und Kühlung des Thermostats oder über die "Hauptschalter" für Heizen und Kühlen im Fenster Spezielle Objekte.
Um die Kühlung über Variable zu aktivieren muss die Variable Heizung ausgeschaltet werden und die Variable Kühlung des Thermostats eingeschaltet werden.
Das wir über Skriptanweisungen realisiert, z.B.:

```
WTWohnen.Heizung ausschalten
WTWohnen.Kuehlung einschalten
```

Die Variable Heizung hat Vorrang, wenn diese eingeschaltet ist, wird die Kühlung unabhängig vom Stand der Variablen Kuehlung deaktiviert.

Eine andere Möglichkeit ist die jeweiligen „Hauptschalter“ zu benutzen. Diese Schalter können Hardwareschalter, aber auch virtuelle Schalter sein.
Die Schalter müssen jeweils im Fenster Spezielle Objekte (Menüpunkt Konfigurieren->Einstellungen->Reiter Objekte->Button [Spezielle Objekte]) unter "Hauptschalter für Heizen" bzw. "Hauptschalter für Kühlen" ausgewählt werden sein.
Heizung und Kühlung sind dann jeweils nur aktiv wenn der jeweilige Schalter eingeschaltet ist.
Im Gegensatz zur Steuerung über Variablen können mit dieser Methode Heizung und Kühlung auch

gleichzeitig aktiviert werden, das ist jedoch keinesfalls zu empfehlen, da je nach Einstellung der Hysterese in den Heizprofilen beide Aktoren wechselseitig aktiviert werden können.
Die Software überprüft ob gemäss der vorgenommen Einstellungen beide Aktoren gleichzeitig eingeschaltet würden, wenn das der Fall ist wird keiner der Aktoren eingeschaltet.

1.12.2.3. 19-Tasten-Fernbedienung

19-Tasten-Fernbedienung

Die zwei oberen Tasten des Dreier-Blocks auf der rechten Seite können nicht durch das Programm benutzt werden, diese sind für interne Funktionen der Fernbedienung vorgesehen (siehe Bedienungsanleitung der Fernbedienung).
Das letzte Objekt der Fernbedienung (Standardname nach Modulimport .._18) ist ein Objekt vom Typ Zeichen, über das Text auf der Fernbedienung ausgegeben werden kann. Zur Ausgabe von Text wird dieser einfach dem Objekt zugewiesen, also z.B:
`AnzeigeRC19:="Alarm"`
Weiterhin können die einzelnen Icons der Fernbedienung über Variablen dieses Objekts aktiviert werden.
Folgende Variablen für die einzelnen Icons stehen zur Verfügung:
PfeilUnten, PfeilOben, Glocke, Rollo, Licht, Uhr, Tür, Telefon, Szene, Schalter, Fenster.
Um die entsprechenden Icons anzuzeigen werden die Variablen "eingeschaltet". Also z.B.:
`AnzeigeRC19.Glocke einschalten`
`AnzeigeRC19.Licht einschalten`
bewirken, dass die Icons "Glocke" und "Glühbirne" angezeigt werden.

Ausser den Variablen für die Icons gibt es noch weitere Variablen um Informationen auf der Fernbedienung auszugeben.

Die Variable **Einheit** kann mit Werten von 0 bis 4 belegt werden und bewirkt die Anzeige einer der folgenden Einheiten:
0 = Keine Einheit
1 = Prozent
2 = Watt
3 = Celsius
4 = Fahrenheit

Mit der Variablen **Beleuchtung** kann die Hintergrundbeleuchtung gesteuert werden.
0 = Keine Hintergrundbeleuchtung
1 = Hintergrundbeleuchtung einschalten
2 = Langsames Blinken
3 = Schnelles Blinken

Mit der Variablen **Beep** kann die Ausgabe eines akustischen Signals aktiviert werden.
0 = Keine Ton
1 = Ton 1
2 = Ton 2
3 = Ton 3

Die Ausgabe auf die Fernbedienung erfolgt nicht automatisch nach Zuweisung von Werten sondern mit einem speziellen Ausgabebefehl nachdem alle gewünschten Werte gesetzt wurden. Dieser Befehl lautet:
`Anzeigen(Objektname)`
Die Ausgabe wird dann einige Zeit auf der Fernbedienung angezeigt. Soll die Ausgabe nochmals angezeigt werden, so muss die Ausgabeanweisung erneut ausgeführt werden.
Alle Variablen für die Icon-Darstellung auf der Fernbedienung können gelöscht werden mit der Anweisung
`LöscheAnzeige(Objektname)`
Dies erspart das Löschen der einzelnen Variablen, wenn die Ausgabe an die Fernbedienung geändert werden soll. Diese Anweisung bewirkt keine Aktualisierung der Anzeige des Moduls, die Aktualisierung muss jeweils mit der Anweisung **Anzeigen(...)** vorgenommen werden.

Beispiel:
Das Wort "LICHT" soll als Text erscheinen und das Glockensymbol der Fernbedienung soll eingeschaltet werden:

```
FB19Tasten_18.Glocke einschalten
FB19Tasten_18:="WORT"
Anzeigen(FB19Tasten_18)
```

1.12.2.4. MP3 Funkgong mit Signalleuchte HM-OU-CFM-...

MP3 Funkgong mit Signalleuchte HM-OU-CFM-...

Das erste Objekt (der erste Kanal) dieses Moduls ist die Leuchte des MP3-Players. Der Typ des Objekts ist "Zeichen" (also Text). Zur Ansteuerung des MP3-Players werden mehrere Parameter benutzt, die durch Komma getrennt werden.

Parameter sind ein Start/Stopp-Code, die Anzahl der Wiederholungen, die Dauer der Ausgabe und bis zu 10 Codes zur Ausgabe von bis zu 10 unterschiedlichen Lichtsignalen.
Diese Parameter werden in einer Textzeile übergeben und jeweils durch ein Komma getrennt.
Im Folgenden die Beschreibung der einzelnen Parameter, in der Reihenfolge in der sie in der Zeile stehen müssen:

Aktivierungskennung: 1=Starten (normal), 0=Ausgabe beenden
Wiederholungen : Anzahl der Wiederholungen als ganze Zahl zwischen 1 und 255
maximale Gesamtdauer der Ausgabe in Sekunden : ganze Zahl zwischen 1 und 108000, 108000=keine Begrenzung, Ausgabe bis alle Licht-Parameter abgearbeitet sind.
Die nächsten 10 Parameter legen fest, was die Leuchte anzeigt:

	kurz	lang
Rot	17	18
Grün	33	34
Orange	49	50
Blau	65	66
Violett	81	82
Cyan	97	98
Weiss	113	114
Pause	2	
Aus	0	

Beispiel:
Die Leuchte soll dreimal folgende Sequenz anzeigen:
Grün kurz, Rot lang, Pause, Orange kurz
Dazu wird in einem Skript folgende Zuweisung gemacht:

```
MP3Leuchte:="1,3,108000,33,18,2,49"
```

Das zweite Objekt (der zweite Kanal) ist der MP3-Player. Der Typ des Objekts ist "Zeichen" (also Text). Zur Ansteuerung des MP3-Players werden mehrere Parameter benutzt, die durch Komma getrennt werden.
Parameter sind die Lautstärke, die Anzahl der Wiederholungen, die Gesamtspieldauer und der Index der abzuspielenden MP3-Datei bzw. eine Playlist von bis zu 10 MP3-Dateien.

Diese Parameter werden in einer Textzeile übergeben und jeweils durch ein Komma getrennt.
Im Folgenden die Beschreibung der einzelnen Parameter, in der Reihenfolge in der sie in der Zeile stehen müssen:

Lautstärke : Zahl zwischen 0 und 1 (0.1 bis 1.0), Dezimaltrennzeichen ist ein Punkt.
Wiederholungen : Anzahl der Wiederholungen als ganze Zahl zwischen 1 und 255
maximale Gesamtdauer der Soundausgabe in Sekunden : ganze Zahl zwischen 1 und 108000, 108000=Dateilänge der MP3-Datei
MP3-Datei bzw. Playlist : der Index der abzuspielenden MP3-Datei(en) jeweils getrennt durch Komma, maximal 10 Dateien.

MP3-Datei bzw. Playlist : der Index der abzuspielenden MP3-Datei(en) jeweils getrennt durch Komma, maximal 10 Dateien.

Beispiel:

Die Dateien 001_Sound.mp3, 003_Song und 006_Ansage2.mp3 sollen mit halber Lautstärke einmal abgespielt werden.
Dazu wird in einem Skript folgende Zuweisung gemacht:

```
MP3Player:="0.5,1,108000,1,3,6"
```

Als Dezimaltrenner für die Lautstärke muss ein Punkt verwendet werden, da das Komma zur Trennung der Parameter verwendet wird.

Bitte beachten Sie unbedingt folgende Besonderheit bei diesem Modul:
Es wird immer nach einer Zuweisung gesendet wird und nicht wie bei anderen Modulen üblich nur wenn der Wert der Objekte sich ändert. Das wurde so eingerichtet, da die abzuspielende Sequenz oftmals gleich ist und bei erneuter Zuweisung auch erneut abgespielt werden soll. Es ist daher wichtig bei der Programmierung zu beachten, dass jede Zuwesiung zu einer Ausgabe an das Modul führt auch wenn der Wert sich nicht ändert.

1.12.2.5. MP3-Player HM-OU-CM-PCB

MP3-Player HM-OU-CM-PCB

Der Typ des Objekts ist "Zeichen" (also Text). Zur Ansteuerung des MP3-Players werden mehrere Parameter benutzt, die durch Komma getrennt werden.
Parameter sind die Lautstärke, die Anzahl der Wiederholungen, die Gesamtspieldauer und der Index der abzuspielenden MP3-Datei bzw. eine Playlist von bis zu 10 MP3-Dateien.

Diese Parameter werden in einer Textzeile übergeben und jeweils durch ein Komma getrennt.
Im Folgenden die Beschreibung der einzelnen Parameter, in der Reihenfolge in der sie in der Zeile stehen müssen:

Lautstärke : Zahl zwischen 0 und 1 (0.1 bis 1.0), Dezimaltrennzeichen ist ein Punkt.
Wiederholungen : Anzahl der Wiederholungen als ganze Zahl zwischen 1 und 255
maximale Gesamtdauer der Soundausgabe in Sekunden : ganze Zahl zwischen 1 und 108000, 108000=Dateilänge der MP3-Datei
MP3-Datei bzw. Playlist : der Index der abzuspielenden MP3-Datei(en) jeweils getrennt durch Komma, maximal 10 Dateien.

MP3-Datei bzw. Playlist : der Index der abzuspielenden MP3-Datei(en) jeweils getrennt durch Komma, maximal 10 Dateien.

Beispiel:
Die Dateien 001_Sound.mp3, 003_Song und 006_Ansage2.mp3 sollen mit halber Lautstärke einmal abgespielt werden.
Dazu wird in einem Skript folgende Zuweisung gemacht:

```
MP3Player:="0.5,1,108000,1,3,6"
```

Als Dezimaltrenner für die Lautstärke muss ein Punkt verwendet werden, da das Komma zur Trennung der Parameter verwendet wird.

Bitte beachten Sie unbedingt folgende Besonderheit bei diesem Modul:
Es wird immer nach einer Zuweisung gesendet wird und nicht wie bei anderen Modulen üblich nur wenn der Wert der Objekte sich ändert. Das wurde so eingerichtet, da die abzuspielende Sequenz oftmals gleich ist und bei erneuter Zuweisung auch erneut abgespielt werden soll. Es ist daher wichtig bei der Programmierung zu beachten, dass jede Zuwesiung zu einer Ausgabe an das Modul führt auch wenn der Wert sich nicht ändert.

1.12.2.6. Statusanzeige HM-Dis-WM55

Statusanzeige HM-Dis-WM55

Mit der Statusanzeige können über das Drücken auf den oberen oder unteren Rand aktuelle Informationen abgerufen werden.
Die Statusanzeige hat zwei Taster, durch die Betätigung wird jeweils das Skript eines Tasters aufgerufen. In diesem Skript kann dann mit der Anweisung SETZEWERT der Text erstellt werden, der für einige Sekunden auf dem Display der Statusanzeige angezeigt wird. Die Ausgabe an die Statusanzeige kann nur im Skript eines Tasters der Statusanzeige mit der Anweisung **Setzewert** mit Wertecode "StatusDisplay" (siehe Beispiel) erzeugt werden. Die Statusanzeige erwartet die Zuweisung eines Textes Sekundenbruchteile nach Betätigung der Taste, daher dürfen in dem Skript keine Verzögerungen auftreten, also keine WARTE-Anweisungen oder Schleifen benutzt werden.
Das Schlüsselwort "selbst" kann innerhalb eines Skripts immer benutzt werden um das "eigene" Objekt des Skripts zu bezeichnen, das ist oftmals einfacher und schneller als jedes mal im Skript den "eigenen" Objektnamen zu schreiben.
Die Anweisungen des folgenden Beispiels müssen also im Skript eines Tasters der Statusanzeige stehen.

```
wenn selbst = "kurz" dann
  Setzewert(selbst,"StatusDisplay","Kurz<<gedrückt")
sonst
  Setzewert(selbst,"StatusDisplay","Lang<<gedrückt")
endewenn
```

Ein Zeilenvorschub wird mit zwei kleiner-Zeichen << hintereinander erzeugt
Der Text kann farbig dargestellt werden, am Ende einer Zeile kann ein Icon angezeigt werden.
Um die Textfarbe zu setzen wird ein Backslash ("\") gefolgt vom Buchstaben "F" (für Farbe) und einer Zahl oder einem Schlüsselwort, abgeschlossen wird der Code immer mit einem Semikolon (unbedingt erforderlich!).

Es stehen folgende Farben zur Verfügung:
Weiss -> Schlüsselwort WEISS oder Zahl 1
Rot -> Schlüsselwort ROT oder Zahl 2
Orange -> Schlüsselwort ORANGE oder Zahl 3
Gelb -> Schlüsselwort GELB oder Zahl 4
Grün -> Schlüsselwort GRÜN oder Zahl 5
Blau -> Schlüsselwort BLAU oder Zahl 6

Um ein Icon zu setzen wird ein Backslash ("\") gefolgt vom Buchstaben "I" (für Icon) und einer Zahl oder einem Schlüsselwort, abgeschlossen wird der Code immer mit einem Semikolon (unbedingt erforderlich!).
Es stehen folgende Icons zur Verfügung:

Lampe aus -> Schlüsselwort AUS oder Zahl 1
Lampe an -> Schlüsselwort AN oder Zahl 2
Schloss offen -> Schlüsselwort OFFEN oder Zahl 3
Schloss geschlossen -> Schlüsselwort GESCHLOSSEN oder Zahl 4
Fehler -> Schlüsselwort FEHLER oder Zahl 5
OK -> Schlüsselwort OK oder Zahl 6
Info -> Schlüsselwort INFO oder Zahl 7
Nachricht -> Schlüsselwort NACHRICHT oder Zahl 8
Servicemeldung -> Schlüsselwort SERVICE oder Zahl 9
Signal grün -> Schlüsselwort GRÜN oder Zahl 10
Signal gelb -> Schlüsselwort GELB oder Zahl 11
Signal rot -> Schlüsselwort ROT oder Zahl 12

Als erstes muss der Text stehen, Icon- oder Farbcodes dürfen nicht am Anfang stehen.
Der Text um zwei Zeilen zu erzeugen, wobei die erste Zeile gelb und die zweite Zeile blau dargestellt wird, am Ende der ersten Zeile ein offenes Schloss, am Ende der zweiten Zeile ein Info

Zeichen sieht beispielsweise folgendermassen aus:

"Erste Zeile\FGELB;\IOPEN;<<zweite Zeile\F6;\I7;"

Um z.B. offene Fenster in der Anzeige anzuzeigen könnte man folgendes Skript schreiben:

```
Text:="Offene Fenster:<<"
wenn FensterWohnen geöffnet dann
  Text:="Wohnen\FGELB;<<"
endewenn
wenn FensterKueche geöffnet dann
  Text:=Text + "Küche\FGELB;<<"
endewenn
wenn FensterSchhlafen geöffnet dann
  Text:=Text + "Schlafen\FROT;"
endewenn
Setzewert(selbst,"StatusDisplay",Text)
```

1.12.2.7. EPaper-Display HM-Dis-EP-WM55

EPaper-Display HM-Dis-EP-WM55

Dieses Gerät hat eine E-Paper Displayanzeige und zwei Tasten. Es kann jederzeit eine Ausgabe auf das Display erfolgen, wobei es auch möglich ist eine Tonfolge auszugeben und die LED aufblinken zu lassen. Im Display können bis zu drei Zeilen angezeigt werden.

Wenn eine der beiden Tasten betätigt wird, wird die Tastermeldung empfangen und im Display erscheint der Standardtext, der in der WEB-UI für die beiden Tasten hinterlegt ist.
Eine neuer beliebiger Anzeigetext kann erst nach einer Wartezeit von mindestens 5 Sekunden an das Display geschickt werden.

Beispiel für ein Skript:

```
wenn Terrassentuer geschlossen dann
  AktuellerStatus:="Die Türe ist<<geschlossen\IGESCHLOSSEN;"
sonst
  AktuellerStatus:="Die Türe ist<<geöffnet\IOFFEN;"
EDisplay:=AktuellerStatus
```

wobei AktuellerStatus hier eine Variable Typ Zeichen ist.

Ein Zeilenvorschub wird mit zwei kleiner-Zeichen << hintereinander erzeugt
Am Ende einer Zeile kann ein Icon angezeigt werden.
Um ein Icon zu setzen wird ein Backslash ("\") gefolgt vom Buchstaben "I" (für Icon) und einem Schlüsselwort, abgeschlossen wird der Code immer mit einem Semikolon (unbedingt erforderlich!).
Es stehen folgende Icons zur Verfügung:

Lampe aus -> Schlüsselwort AUS oder Zahl 1
Lampe an -> Schlüsselwort AN oder Zahl 2
Schloss offen -> Schlüsselwort OFFEN oder Zahl 3
Schloss geschlossen -> Schlüsselwort GESCHLOSSEN oder Zahl 4
Fehler -> Schlüsselwort FEHLER oder Zahl 5
OK -> Schlüsselwort OK oder Zahl 6
Info -> Schlüsselwort INFO oder Zahl 7
Nachricht -> Schlüsselwort NACHRICHT oder Zahl 8
Servicemeldung -> Schlüsselwort SERVICE oder Zahl 9

Mit der Ausgabe auf das Display kann zusätzlich eine Tonfolge bestimmt werden.
Um eine Tonfolge auszugeben wird ein Backslash ("\") gefolgt vom Buchstaben "S" (für Sound) und einer Ziffer für die Tonfolge, abgeschlossen wird der Code immer mit einem Semikolon.
Folgende Tonfolgen sind möglich:
1 -> lang lang
2 -> lang kurz
3 -> lang kurz kurz
4 -> kurz
5 -> kurz kurz
6 -> lang

Weiterhin kann die Anzahl der Wiederholungen für die Tonfolge bestimmt werden, das ist der zweite Wert innerhalb des Sound-Parameters.

Als dritter Wert des Sound-Parameters kann der Abstand der zu wiederholenden Tonfolge angegeben werden, wobei die Einheit ein 10-Sekunden-Intervall ist, der Wert 3 würde also einen Abstand von 30 Sekunden zwischen den Tonfolgen bedeuten.

Für die Tonfolge gibt es also drei Werte, die durch Komma getrennt werden;
\S<Tonfolge>,<Wiederholungen>,<Zeitabstand in 10 Sekundenintervall>;
Wenn der Wert für Wiederholungen und Zeitabstand weggelassen wird, wird der Ton nur einmal

ausgegeben.
Am Ende des Codes ist ein Semikolon erforderlich!

Weiterhin ist es möglich bei Ausgabe auf das Display die LED kurz aufblinken zu lassen, wobei unterschiedliche Farben benutzt werden können.
Um die LED aufblinken zu lassen wird ein Backslash ("\") gefolgt vom Buchstaben "L" (für LED) und einer Ziffer für die Farbe, abgeschlossen wird der Code immer mit einem
Semikolon.
Folgende Farben können durch die Ziffer im Code bestimmt werden:
1 -> rot
2 -> grün
3 -> orange

Als erstes muss der Text stehen, Codes für Icons, Sound und LED dürfen nicht am Anfang stehen.

Hier ein Beispiel mit folgenden Eigenschaften:
Es werden drei Zeilen angezeigt, wobei die erste Zeile als Icon eine Glühbirne bekommt, die zweite Zeile ein geschlossenes Schloss und die dritte Zeile das Infozeichen bekommt. Als Soundausgabe wird ein langes Tonsignal mit einer Wiederholung im Abstand von 10 Sekunden festgelegt und ein Blinken der LED in Orange.
Dazu muss die Ausgabe wie folgt aufgebaut werden:

EDisplay:="Erste Zeile\IAN;<<Zweite Zeile\IGESCHLOSSEN;<<Dritte Zeile\I7;\S6,1,1;\L3;"

1.12.2.8. WinMatic

WinMatic

Die Steuerung der WinMatic ist etwas speziell, intern wird die Position mit Kommazahlen ziwischen 0 und 1 gesteuert, verriegelt wird mit -0,005.
Da die Steuerung des Geräts prinzipiell über Zahlen mit nur einem Datenpunkt erfolgt, lässt sich die Steuerung nicht wie bei anderen Geräten auch über Zustände realisieren.
Um dieses spezielle Gerät möglichst einfach abbilden zu können wird der interne Wert umgerechnet auf Prozentzahlen zwischen 0 und 100, als "Verriegelungswert" wird -1 benutzt.

Eine weitere Besonderheit dieses Geräts ist, dass man nicht einfach den Wert setzen kann, wenn der aktuelle Zustand "verriegelt", also -1 ist. Zuerst muss "geöffnet" werden, indem der Wert auf 0 gesetzt wird. Erst wenn dieser Verstellvorgang am Gerät abgeschlossen ist, kann der Wert für den Öffnungswinkel zum Gerät übertragen werden.

Hier ein Beispielskript zur Ansteuerung einer WinMatic, in dem ein Fenster zum Lüften für 90 Minuten geöffnet wird:

```
wenn WAktor < 0 dann  // wenn verriegelt
  WAktor:=0           // Fenstergriff öffnen
  warte 20 Sekunden   // warten bis der Öffnungsvorgang abgeschlossen ist
endewenn
WAktor:=50            // Fenster kippen auf 50 Prozent
warte 90 Minuten      // 90 Minuten geöffnet lassen
WAktor:=-1            // und zum Schluss schliessen und verriegeln
```

1.12.3. HomeMatic IP

HomeMatic IP

Hier werden Hinweise zu speziellen HomeMatic-IP-Geräten beschrieben.
HomeMatic-Geräte erkennt man daran, dass die Typbezeichnung mit HmIP-.. beginnt.

Bei den HomeMatic und HomeMatic IP - Geräten, bei denen die Eingänge umkonfigurierbar sind ist folgendes zu beachten:
Die aktuelle Konfiguration des Eingangstyps wird beim Import nicht automatisch eingestellt, es muss im Programm jeweils der passende Objekttyp ausgewählt werden.

Das Verhalten bei Tastern hat sich gegenüber HomeMatic-Geräten leider geändert. Bei Geräten mit Tastern, die auch einen langen Tastendruck erlauben, wird dieser mit den Standardeinstellungen nicht mehr nur einmal, sondern ununterbrochen gesendet. Um das zu vermeiden muss in der Konfiguration des Tasters in der WEB-UI der Wert ***Timeout für langen Tastendruck*** auf 100ms gesetzt werden.

1.12.3.1. HmIP-Thermostate

HmIP-Thermostate

Zur Visualisierung des Thermostats wird ein spezielles Panel benutzt.
Die Temperatur auf der linken Seite des Panels ist die Solltemperatur, die über die Visualisierung verändert werden kann. Die Temperatur auf der rechten Seite des Panels ist die aktuelle Ist-Temperatur, die vom Thermostat empfangen wurde. In der Mitte wird die aktuelle gemessene Luftfeuchtigkeit angezeigt, bei Heizkörperthermostaten die aktuelle Ventilstellung.
Die Darstellung von Luftfeuchtigkeit und Ist-Temperatur kann in der Definition des Thermostat-Panels ausgeschaltet werden.

Die Solltemperatur wird in den Skripts direkt über den Namen des Thermostats eingestellt bzw. abgefragt.
Beispiel:

```
WandthermostatWohnen:=21,5
```

oder

```
WandthermostatWohnen setzen auf 21,5
bzw.
wenn WandthermostatWohnen > 23 dann
```

Es können auch die Werte Off und On über Skripts im Raumthermostat gesetzt werden, indem für Off der Wert 0, für On der Wert 100 zugewiesen wird.
also z.B.

```
WandthermostatWohnen:=100
```

Der Wert **Off** schliesst das Ventil vollständig, der Wert **On** öffnet das Ventil vollständig. Bei beiden Modi wird die interne Zeittabelle des Thermostats deaktiviert.

Bei Zuweisungen der Soll-Temperatur ist zu beachten, dass diese grundsätzlich in 0,5 Grad Schritten anzugeben ist und im min/max-Bereich des jeweiligen Geräts liegt, dieser kann je nach Thermostat unterschiedlich sein. Wenn Format oder Wertebereich nicht stimmen wird der Wert vom Gerät nicht akzeptiert.

Die Ist-Temperatur steht in Skripts als Variable *Temperatur* des Raumthermostats zur Verfügung, die Luftfeuchtigkeit als Variable *Luftfeuchtigkeit*,
also z.B.

```
wenn Wandthermostat.Temperatur > 22.5 dann
....
```

Wenn das Wandthermostat-Panel angeklickt wird, öffnet sich ein Fenster in dem die Soll-Temperatur verändert werden kann.

Variablen eines Wandthermostats
Ein Wandthermostat hat einige Variablen, diese können je nach Typ des Wandthermostat unterschiedlich sein, bei manchen Typen stehen nicht alle Variablen zur Verfügung.
Hier eine Liste der Variablen:

Variablennname	Verwendung	Typ
Temperatur	Ist-Temperatur	Zahl
Luftfeuchtigkeit	Luftfeuchtigkeit	Zahl (nicht bei Heizkörperthermostaten)
VentilPos	Ventilposition in %	Zahl (nur bei Heizkörperthermostaten)
Batteriestatus	aktuelle Batteriespannung	Zahl (nicht bei der Wired-Version)
Modus	Betriebsmodus	Zahl 0=AUTO, 1=MANUELL Der Modus kann nicht verändert werden, dazu muss SETZEWERT benutzt werden (s. unten)
Zeittabelle	Zeittabelle (de)aktivieren	Schalter ein/aus (nur wenn für das Thermostat eine Zeittabelle angelegt wird)

```
Heizung            Steuerung Heizkaktor ein/aus  Schalter (nicht bei
Heizkörperthermostaten)
Kuehlung           Steuerung Kühlaktor ein/aus   Schalter (nicht bei
Heizkörperthermostaten)
TCTemparatur       nur zur internen Verwendung   (nicht bei allen
Thermostaten)
```

Hinweise zur Verwendung der Anweisung **SETZEWERT** bei HmIP-Thermostaten (gilt nicht für HM-Thermostate):

Der BOOST-Modus wird eingeschaltet mit
```
SETZEWERT(ThermostatName,"BOOST_MODE",1)
```

Der AUTO-Modus wird eingeschaltet mit
```
SETZEWERT(ThermostatName,"CONTROL_MODE",0)
```

Der MANU-Modus wird eingeschaltet mit
```
SETZEWERT(ThermostatName,"CONTROL_MODE",1)
```

Das Setzen der Solltemperatur erfolgt bei HmIP-Thermostaten mit dem Schlüsselwort SET_POINT_TEMPERATUR
```
SETZEWERT(ThermostatName,"SET_POINT_TEMPARATURE","22,5")
```

Temperatursteuerung mit Wandthermostaten über Schaltaktoren:
Im Verwaltungsprogramm für Wandthermostate können binäre Aktoren zur Steuerung von Heiz- und Kühl-Geräten angegeben werden. Die über die Aktoren gesteuerten Geräte werden in Abhängigkeit der Soll- und Ist-Temperaturen direkt geschaltet. Dabei können über spezielle Profile die Temperatur-Grenzwerte zum Schalten der Aktoren den individuellen Gegebenheiten angepasst werden. Die Einstellungen der Profile kann mit dem Button unterhalb der Auswahlfelder für die Aktoren aufgerufen werden.
Mit diese Methode der Steuerung und der Möglichkeit eigene Heizprofile zu erstellen können auch träge reagierende Fussbodenheizungen individuell gesteuert werden.

Es können unterschiedliche Aktoren zur Heizung und Kühlung verwendet werden.
Dabei ist folgendes zu beachten:
Die Funktionen zur Heizung und Kühlung können standardmässig nicht gleichzeitig aktiv sein.
Die Steuerung welche Funktion aktiv sein soll erfolgt über die Variablen Heizung und Kühlung des Thermostats oder über die "Hauptschalter" für Heizen und Kühlen im Fenster Spezielle Objekte.
Um die Kühlung über Variable zu aktivieren muss die Variable Heizung ausgeschaltet werden und die Variable Kühlung des Thermostats eingeschaltet werden.
Das wir über Skriptanweisungen realisiert, z.B.:
```
WTWohnen.Heizung ausschalten
WTWohnen.Kuehlung einschalten
```

Die Variable Heizung hat Vorrang, wenn diese eingeschaltet ist, wird die Kühlung unabhängig vom Stand der Variablen Kuehlung deaktiviert.
Eine andere Möglichkeit ist die jeweiligen „Hauptschalter“ zu benutzen. Diese Schalter können Hardwareschalter, aber auch virtuelle Schalter sein.
Die Schalter müssen jeweils im Fenster Spezielle Objekte (Menüpunkt Konfigurieren->Einstellungen->Reiter Objekte->Button [Spezielle Objekte]) unter "Hauptschalter für Heizen" bzw. "Hauptschalter für Kühlen" ausgewählt werden sein.
Heizung und Kühlung sind dann jeweils nur aktiv wenn der jeweilige Schalter eingeschaltet ist.
Im Gegensatz zur Steuerung über Variablen können mit dieser Methode Heizung und Kühlung auch gleichzeitig aktiviert werden, das ist jedoch keinesfalls zu empfehlen, da je nach Einstellung der Hysterese in den Heizprofilen beide Aktoren wechselseitig aktiviert werden können.
Die Software überprüft ob gemäss der vorgenommen Einstellungen beide Aktoren gleichzeitig eingeschaltet würden, wenn das der Fall ist wird keiner der Aktoren eingeschaltet.

1.12.3.2. HmIP-Rauchmelder

HmIP-Rauchmelder

Bei den HmIP-Rauchmeldern kann die Sirene über die Zentrale geschaltet werden.

Das passiert mit der Anweisung SETZEWERT.

Zum Einschalten wird folgende Anweisung benutzt:
```
SETZEWERT(MeinHMIPRauchmelder, "SMOKE_DETECTOR_COMMAND", "INTRUSION_ALARM")
```

Zum Ausschalten wird folgende Anweisung benutzt:
```
SETZEWERT(MeinHMIPRauchmelder, "SMOKE_DETECTOR_COMMAND",
"INTRUSION_ALARM_OFF")
```

1.12.3.3. HmIP-ASIR - Alarmsirenen

HmIP-ASIR - Alarmsirenen

Diese Alarmsirenen erlaubt die Ausgabe optischer und akustischer Alarmsignale.
Die Aktivierung des Alarms erfolgt durch Ausgabe eines codierten Textes mit drei Parametern an die Alarmsirene.

Der erste Parameter bestimmt welches akustische Signal ausgegeben werden soll, der zweite Patameter bestimmt das optische Signal, der dritte Parameter die Zeit wie lange die Alarmsignale ausgegegben werden sollen.
Die Angabe der Signale erfolgt als Index einer Tabelle von Auswahlmöglichkeiten.

Folgende akustischen Signale stehen zur Verfügung (Parameter 1):

```
0 = kein akustisches Signal
1 = Frequenz steigend
2 = Frequenz fallend
3 = Frequenz steigend/fallend
4 = Frequenz tief/hoch
5 = Frequenz tief/mittel/hoch
6 = Frequenz hoch ein/aus
7 = Frequenz hoch ein, lang aus
8 = Frequenz tief ein/aus, hoch ein/aus
9 = Frequenz tief ein - lang aus, hoch ein - lang aus
```

Folgende optischen Signale stehen zur Verfügung (Parameter 2):

```
0 = kein optisches Signal
1 = Abwechselndes langsames Blinken
2 = Gleichzeitiges langsames Blinken
3 = Gleichzeitiges schnelles Blinken
4 = Gleichzeitiges kurzes Blinken
5 = Bestätigungssignal 0 - lang lang
6 = Bestätigungssignal 1 - lang kurz
7 = Bestätigungssignal 2 - lang kurz kurz
```

Die Zeit (Parameter 3) wird wie folgt angegeben:
Zeitwert direkt gefolgt von der Zeitheinheit.
Als Zeitwert kann eine Zahl zwischen 1 und 60 angegeben werden, als Zeiteinheit S für Sekunden oder M für Minuten.

Beispiel für eine Ausgabe innerhalb eines Skripts:

```
Alarmsirene:="3,1,15S"
```

Diese Sequenz bewirkt, dass für 15 Sekunden ein akustisches Signal mit steigender und fallender Frequenz ausgegeben wird und die LEDs dabei abwechsenld langsam blinken.

Bitte beachten Sie unbedingt folgende Besonderheiten bei diesem Modul:
Zur Ansteuerung der Alarmsirene wird vom BidCoS ein anderes Verfahren benutzt als für Aktoren normalerweise üblich. Dadurch bedingt können die Reaktionszeiten schwanken und es kann über 5 Sekunden dauern bis die Alarmsirene reagiert.

Es wird immer nach einer Zuweisung gesendet wird und nicht wie bei anderen Modulen üblich nur wenn der Wert der Objekte sich ändert. Das wurde so eingerichtet, da die abzuspielende Sequenz oftmals gleich ist und bei erneuter Zuweisung auch erneut abgespielt werden soll. Es ist daher wichtig bei der Programmierung zu beachten, dass jede Zuwesiung zu einer Ausgabe an das Modul führt auch wenn der Wert sich nicht ändert.

1.12.3.4. HmIP-BSL - Schaltaktor mit Signalleuchte

HmIP-BSL - Schaltaktor mit Signalleuchte

Dieser Schaltaktor verfügt neben dem Schaltausgang über zwei Signalleuchten die jeweils in unterschiedlicher Farbe und einer Helligkeit zwischen 0 und 100 aktiviert werden können.
Beim Import haben diese Signallleuchten jeweils den Namen *Farbsignal...* .
Der Name kann natürlich wie bei anderen Objekten geändert werden.

Jedes Objekt zu den Signalleuchten hat zusätzlich eine Variable mit Namen *Farbe*, die mit Zahlen zwischen 0 und 7 belegt werden kann.
Folgenden Zahlen und zugehörige Farben können verwendet werden:
0 = aus
1 = blau
2 = grün
3 = türkis
4 = rot
5 = violett
6 = gelb
7 = weiss

Zu diesem Gerät werden bei Import 5 Objekte erzeugt:
Zwei Taster (Kanäle 1 und 2) ,

der eigentliche Schaltaktor (Index 1, Kanal 4) zum Schalten des Schaltaktors

und
Signalleuchte 1 (Index 2, Kanal 8)
Signalleuchte 2 (Index 3, Kanal 12)
jeweils als Dimmer mit Variable *Farbe* zum Setzen der Farbe

Beispiel:
```
AktorSig einschalten
Signalleuchte1.Farbe:=4
Signalleuchte1:=80
```

Schaltet den Aktor 1 und setzt die Farbe einer Leuchte auf rot und die Helligkeit auf 80%.

1.12.3.5. HmIP-DLD - Türschlossantrieb

HmIP-DLD - Türschlossantrieb

Dieser HmIP-Türschlossantrieb hat leider ein vollkommen anderes Protokoll als die KeyMatic der HomeMatic-Serie.

Der Türschlossantrieb hat zwei separate Objekte mit unterschiedlichen Zuständen für Status und Aktionen, die Zustände des Aktionen-Objekts werden zur Steuerung des Antriebs benutzt, das Status-Objekt wird nur zur Anzeige des aktuellen Zustands benutzt, eine Zustandsänderung im Status-Objekt hat keinen Einfluss auf den Antrieb und macht daher keinen Sinn.

Zur Steuerung werden die Zustände des Aktionen-Objekts benutzt, die Skriptanweisungen sehen also z.B. so aus:

Türaktion:=“verschliessen“
Türaktion:=“entriegeln“
Türaktion:=“öffnen“

Normalerweise wird immer nur an die Hardware gesendet, wenn der Zustand eines Objekts sich geändert hat. Das ist bei dem Aktinen-Objekt dieses Moduls anders. Da der Zustand des Objekts sich durch die die Rückmeldung des Geräts nicht ändert (weil diese an das Status-Objekt geht), wird bei diesem Objekt immer eine Meldung an die Hardware generiert wenn eine Zuweisung erfolgt. Daher ist bei der Programmierung darauf zu achten, dass nicht zu viele unnötige Zuweisungen gemacht werden.

1.12.3.6. HmIP-MP3P - Kombisignalgeber

HmIP-MP3P - Kombisignalgeber

Mit diesem Gerät können auf einer SD-Karte gespeicherte Sound-Dateien abgespielt werden und die integrierte Signalleuchte kann in verschiedenen Farben und Helligkeitsstufen gesteuert werden.

Steuerung des Sounds (Index 1, Kanal 2).
Die Lautstärke wird durch Zuweisung einer Zahl zwischen 0 und 100 an den ersten Aktor des Geräts (Index 1, Kanal 2) definiert. Über die Variable *Soundfile* des Aktors wird mit einer Zahl zwischen 1 und 252 angegeben welche Sounddatei abgespielt werden soll.
Sobald die Lautstärke des Geräts verändert wird, wird die Datei abgespielt.
Wenn die Variable den Wert 0 hat wird der interne Sound abgespielt.
Beispiel für Skriptanweisungen um Datei 7 mit 50% Lautärke auszugeben:

```
Sound.Soundfile:=7
Sound:=50
```

Steuerung der Signalleuchte (Index 2, Kanal 6)
Die Helligkeit wird durch Zuweisung einer Zahl zwischen 0 und 100 bestimmt. Über die Variable *Farbe* des Aktors wird mit einer Zahl zwischen 0 und 7 die Farbe der Signalleuchte festgelegt.
Sobald die Helligkeit verändert wird, wird die Lampe eingeschaltet.
Folgenden Zahlen und zugehörige Farben können verwendet werden:
0 = aus
1 = blau
2 = grün
3 = türkis
4 = rot
5 = violett
6 = gelb
7 = weiss

Beispiel für Skriptanweisungen um die LED rot leuchten zu lassen mit 80% Helligkeit:

```
Farbleuchte.Farbe:=4
Farbleuchte:=80
```

1.12.3.7. HmIP-SWD Wassersensor

HmIP-SWD Wassersensor

Der HmIP-Wassersensor hat leider eine etwas seltsame Logik.
Üblicherweise gab es bei Wassersensoren immer einen Datenpunkt mit drei möglichen Zuständen OK, Feuchtigkeit, Wasser.
Das ist bei diesem HmIP-Wassersensor leider anders.

Es gibt insgesamt 3 Datenpunkte, einmal den "Hauptdatenpunkt" ALARMSTATE mit den Zuständen OK und Alarm und für Feuchtigkeit und Wasser jeweils einen separaten Datenpunkt, ebenfalls mit den Zuständen OK und Alarm.
Für den Wassersensor wird daher ein Objekt für den Datenpunkt ALARMSTATE erstellt. Um nicht drei unterschiedliche Objekt für das Modul zu haben und es zu unübersichtlich werden zu lassen sind die Werte für Feuchtigkeit und Wasser in zwei Variablen des Objekt untergebracht. Um diese in einer Visualisierung anzuzeigen können virtuelle Objekte erstellt werden, denen im Skript des Objekts zum Wassersensor die Variablenwerte zugewiesen werden.
Dazu wird die Option *Ausführung bei Empfang* aktiviert und z.B. folgende Anweisungen im Skript geschrieben:

```
ObjWasser:=Wassersensor.Wasser
ObjFeuchtigkeit:=Wassersensor.Feuchtigkeit
```

wobei ObjWasser und ObjFeuchtigkeit in diesem Beispiel virtuelle Objekt mit dem Objekttyp *ASchalter* sind.

Wenn das Objekt des Wassersensors den Zustand Alarm bekommt, hat immer auch die Variable *Feuchtigket* den Zustand Alarm.
Wenn Wasser detektiert wird hat auch die Variable *Wasser* den Zustand Alarm.

1.12.3.8. HmIP-WHS2 - Heizungsaktor

HmIP-WHS2 - Heizungsaktor

Dieses Gerät ist für spezielle Anwendungsfunktionen vorgesehen und arbeitet etwas anders als normale Schaltaktoren, bei denen bei denen der Ausgang jeweils über einen Kanal, in der Software also ein Objekt, gesteuert wird.
Dieser Heizungsaktor hat 2 Aktoren (Relais), die über jeweils 3 unterschiedliche Kanäle angesteuert werden.
Dabei wird der erste Ausgang eingeschaltet, sobald einer der Kanäle 2-4 (erkennbar an der Ziffer hinter dem Doppelpunkt der Adresse) eingeschaltet wird.
Ausgeschaltet wird der Ausgang nur wenn alle Kanäle ausgeschaltet sind, solange ein zugehöriges Objekt noch eingeschaltet bleibt der Ausgang eingeschaltet.
Der zweite Ausgang funktioniert genauso und wird über die Kanäle 6-8 gesteuert.

Es ist also kein Fehler wenn der Ausgang nicht ausgeschaltet wird wenn nur eines der zugehörigen Objekte ausgeschaltet wird, es müssen alle Objekte eines Ausgangs ausgeschaltet werden, damit das Relais abschaltet.

1.12.3.9. EPaper-Display HmIP-WRCD

EPaper-Display HmIP-WRCD

Dieses Gerät hat eine E-Paper Displayanzeige und zwei Tasten. Es kann eine Ausgabe in das Display geschrieben werden und es können verschiedene Tonsignale ausgegeben werden.

Im Display können 5 einzelne Zeilen angezeigt werden, Zeile 1 ist die oberste Zeile, in der normalerweise der Text für Taster 1 steht, Zeile 5 ist die unterste Zeile, in der normalerweise der Text für Taster 2 steht.

Taster 1 und Taster 2 können als normale Taster behandelt werden, es ist eine Registrierung im Fenster Schaltsensoren erforderlich damit Tastendrücke empfangen werden können.

Zur Ausgabe in das Display können mehrere Parameter gesetzt werden, alle Parameter ausser dem Text sind optional.

Die Parameter werden durch Komma getrennt, die Reihenfolge der Parameter sieht folgendermassen aus:
`Text,Zeile,Ausrichtung,Hintergrundfarbe,Textfarbe,Icon,Tonsignal,Zeitabstand,Wiederholungen`

Beschreibung der Parameter:

Text : Der auszugebende Text, maximal 14 Zeichen, wenn zusätzlich ein Icon augegeben werden soll darf der Text nur 11 Zeichen haben damit das Icon vollständig zu sehen ist.

Alle weiteren Parameter sind optional, wenn keine Zeile und Ausrichtung angegeben wird, wird mittig in Zeile 3 geschrieben.

Zeile : Es können die Zeilen 1-5 benutzt werden, wobei 1 und 5 jeweils die oberste bzw. unterste Zeile ist, in der der Text für den Taster steht.

Ausrichtung : 0=links, 1=mittig, 2=rechts

Hintergrundfarbe : 0 = WEISS
1 = SCHWARZ

Textfarbe : 0 = WEISS
1 = SCHWARZ

Icon : 0 = kein Icon
1 = LAMP_OFF
2 = LAMP_ON
3 = PADLOCK_OPEN
4 = PADLOCK_CLOSED
5 = ERROR
6 = EVERYTHING_OKAY
7 = INFORMATION
8 = NEW_MESSAGE
9 = SERVICE_MESSAGE
10 = SUN
11 = MOON
12 = WIND
13 = CLOUD
14 = THUNDERSTORM
15 = DRIZZLE
16 = CLOUD_AND_MOOON
17 = RAIN

18 = SNOW
19 = CLOUD_AND_SUN
20 = CLOUD_SUN_AND_RAIN
21 = SNOWFLAKE
22 = RAINDROP
23 = FLAME
24 = WINDOW_OPEN
25 = SHUTTERS
26 = ECO
27 = PROTECTION_DEACTIVATED
28 = EXTERNAL_PROTECTION
29 = INTERNAL_PROTECTION
30 = BELL
31 = CLOCK

Wenn kein Tonsignal ausgegeben werden soll, werden die folgenden Parameter nicht angegeben.

Tonsignal : 0 = LOW_BATTERY
1 = DISARMED
2 = INTERNALLY_ARMED
3 = EXTERNALLY_ARMED
4 = DELAYED_INTERNALLY_ARMED
5 = DELAYED_EXTERNALLY_ARMED
6 = EVENT
7 = ERROR

Wenn das Tonsignal nicht wiederholt werden soll, werden die folgenden Parameter nicht angegeben.

Zeitabstand : Abstand zwischen den Wiederholungen in Sekunden

Wiederholungen : Anzahl der Wiederholungen zusätzlich zur ersten Ausgabe des Tonsignals

Beispiel für eine Skriptanweisung:

```
WRCDDisplay:="Mein Text,2,0,0,1,24,6,3,2"
```

Der Text "Mein Text" wird in Zeile 2 linksbündig in das Display geschrieben mit Hintergrundfarbe weiss und Schriftfarbe schwarz.
Als Icon wird ein offenes Fenster angezeigt (24).
Das Tonsignal EVENT (6) wird ausgegeben und im Abstand von 3 Sekdunden zweimal wiederholt

1.12.3.10. HmIP-WRCR - Drehtaster

HmIP-WRCR - Drehtaster

Der HmIP-WRCR-Drehtaster ist in erster Linie zur Verwendung in direkten Verknüpfungen zusammen mit Dimmern geeignet.

Beim Drehen des Knopfes werden keine Zahlenwerte zur Drehbewegung gemeldet, sondern je nach Dauer der Drehbewegung mehrere Ereignisse "PRESS_LONG" hintereinander.
Diese Ereignisse werden dann an die Logikschichten, also in diesem Fall an die ExecEngine, gemeldet.

Beim Import des Geräts werden 3 Taster-Objekte erstellt.
Das erste Taster-Objekt bildet die Druckfunktion des Drehrads ab, funktioniert wie ein normaler Taster und kann die Zustände "kurz" und "lang" melden.
Das zweite Taster-Objekt des Geräts wechselt in den Zustand "lang" wenn der Drehknopf mit dem Uhrzeigersinns gedreht wird.
Das dritte Taster-Objekt des Geräts wechselt in den Zustand "lang" wenn der Drehknopf entgegen des Uhrzeigersinns gedreht wird.
Beim Import wird standardmässig für die Dreh-Taster eine Reset-Zeit in den Ruhezustand von 2 Sekunden eingetragen, zwei Sekunden nach der letzten "lang"-Meldung einer Drehbewegung wird also der Zustand des Taster-Objekts auf "aus" gesetzt.
Wenn eine Reset-Zeit von nur 1 Sekunde benutzt wird, kann es passieren, dass während der Drehbewegung zwischendurch kurz der Zustand "aus" gesetzt wird.

Für die Nutzung in der Programmierung ist das Gerät aufgrund der beschriebenen Funktionsweise nur bedingt geeignet, es wird vorzugsweise für direkte Verknüpfungen verwendet.

1.12.3.11. HmIP-USBSM

HmIP-USMSM

Bei diesen USB-Aktor ist das Verhalten zum Zeitpunkt der Implementierung nicht entsprechend der Geräte- und Protokoll-Beschreibung.
Beim Drücken der Taste auf Kanal 1 wird der Aktor umgeschaltet, aber es werden keine Tastermeldungen generiert.

Eventuell wird das mit einem zukünftigen Firmwareupdate für das Gerät behoben.

1.12.3.12. HmIP-MIOB

HmIP-MIOB

Bei diesen Gerät ist das Verhalten zum Zeitpunkt der Implementierung nicht entsprechend der Geräte- und Protokoll-Beschreibung.

Bei Änderung des Status der Eingänge werden keine Meldungen an die Zentrale geschickt.

1.12.3.13. HmIP-WUA/ELV-SH-WUA

HmIP-WUA / ELV-SH-WUA

Bei diesem Gerät wurde zusätzlich zum in der Beschreibung und WEB-UI vorhandenen Ausgang 1 (OUT1) auf Kanal 2 auch die Ansteuerung des Ausgangs 2 (OUT2) auf Kanal 4 implementiert. weiterführende Infos zum zweiten Ausgang können der Bedienungsanleitung des Geräts entnommen werden.

1.12.3.14. ELV-SH-WSC

ELV-SH-WSC - Servosteuerung

Bei der Servosteuerung wird die Position des Servomotors in Prozentzahlen von 0-100 angegeben. Diese Position wird dann sofort ohne Verzögerung der Fahrzeit dauerhaft eingestellt.

Optional ist es auch möglich neben der Position die Fahrzeit zur Verstellung und die Zeit in der der Motor in der neuen Position verbleibt bis er sich wieder zurückstellt anzugeben.
Das geschieht über die Anweisung SETTZEWERT mit entsprechenden Parameter unter dem Datenpunjt "COBINED_PARAMETER".
Eine solche Anweisung sieht dann z.B. so aus:

```
Setzewert(Servo1,"COMBINED_PARAMETER","L=60,OT=20,RT=5")
```

Der erste Parameter der Anweisung SETZEWERT ist das Objekt welches angesteuert werden soll, der zweite Parameter ist immer "COMBINED_PARAMETER". Der dritte Parameter ist eine Variable vom Typ Zeichen oder eine Konstante in Hochkomma, die drei weitere Parameter zur Einstellung des Motors beinhalten.

- Parameter L ist der Level für den Servomotor zwischen 0 und 100%.

- Parameter OT steht für "ON_TIME" und ist die Zeit in Sekunde, in der die neue Position verbleiben soll wobei der höchste Wert 8580000 für unendliche Verweildauer steht.

- Parameter RT steht für "RAMP_TIME" und ist die Fahrzeit in Sekunden für die Verstellung zur aktuellen Position.

Im oben angegebenen Beispiel fährt der Motor also auf die Position 60% innerhalb von 5 Sekunden und bleibt dort für 20 Sekunden.

1.12.3.15. HmIP-RGBW

HmIP-RGBW

Der Betriebsmodus des Geräts muss in der WEB-UI im Modul unter "Einstellen" im Abschnitt "Modus" aus einer Auswahlbox ausgewählt eingestellt werden. Der Modus ist aber leider nicht wie üblich in den beim Import übertragenen Daten enthalten. Daher muss wenn nicht der RGBW- bzw. RGB-Modus benutzt wird, der Modus "Tunable White" bzw. "Einzelkanäle" durch manuelles Anlegen des Moduls eingestellt werden.

Dazu wird das durch den Import generierte Gerät HmIP-RGBW gelöscht und eines der in der Geräteliste aufgeführten Module HmIP-RGBW-TW für "Tunable White" bzw. HmIP-RGBW-DIM für vier einzelne Dimmer (also Auswahl "Einzelkanäle) aus der Modulliste angelegt.
Nach dem Anlegen muss die Adresse des Moduls noch manuell eingetragen werden. Die Kanalnummer ist schon vorbelegt, die Platzhalter xxx im Adressfeld müssen durch die Adresse des Geräts ersetzt werden.

1.12.3.16. HmIP-WKP Keypad

HmIP-WKP Keypad

Das Keypad wurde anscheinend in erster Linie zur Verwendung mit Direktverknüpfungen konzipiert und ist zur Verwendung an einer Zentrale leider nur eingeschränkt geeignet.

Wenn ein korrekter Code eingegeben wird und eine der die Schlosstasten "Verriegeln" oder "Entriegeln" gedrückt wird, werden Meldungen nur auf Kanal 0 an die Zentrale geschickt. Der Datenpunkt CODE_ID enthält die Benutzernummer des eingegebenen Codes. Wenn ein ungültiger Code eingegeben wurde wird als CODE-ID 32 empfangen.
Es kommen aber leider keinerlei Infos über die gedrückte Schlosstaste , so dass unterschiedliche gewünschte Funktionen nur über unterschiedliche Codes realisiert werden können.

In der Software wird beim Import ein Objekt vom Typ Zahl angelegt, das bei Empfang vom Keypad die empfangene Benutzernummer zugewiesen bekommt. 1 bis 8 sind gültige Werte, bei 32 wurde ein falscher Code eingegeben.

1.12.3.17. HmIP-Wired

HmIP-Wired

Bei den HmIP-Wired-Geräten ist zu beachten, dass hier im Gegensatz zu den HM-Wired-Geräten bei den meisten Tastern/Schaltern eine Registrierung der einzelnen Kanäle (wie bei Funk-Modulen) erforderlich ist.

1.12.3.18. HmIPW-WRC6

HmIPW-WRC6

Dieses Gerät hat für jede der 6 Tasten eine LED, die mit verschiedenen Optionen gesteuert werden kann.
Die Objekte für die LEDs haben den Objekttyp *Dimmer* und können mit Helligkeiten zwischen 0 (=aus) und 100 (=volle Helligkeit) gesteuert werden. Die erste LED ist dem ersten Taster zugeordnet, die 6. LED dem 6.Taster, das letzte LED-Objekt wird zur Steuerung aller LEDs gemeinsam benutzt.
Die Optionen zur Darstellung werden über Variablen gesteuert, dazu werden folgende Variablen benutzt:

Farbe: diese Variable kann die Zahlen 0 bis 7 zugewiesen bekommen, die Zahlen haben folgende Bedeutung:
0 = aus (bei 0 bleibt die LED dunkel, es muss also immer ein Wert zugewiesen werden damit die LED leuchtet.)
1 = Blau
2 = Grün
3 = Türkis
4 = Rot
5 = Violet
6 = Gelb
7 = Weiss
8 = alter wert

Modus: diese Variable bestimmt das Verhalten der LED und kann die Zahlen 0 bis 11 zugewiesen bekommen. Die Zahlen haben folgende Bedeutung:
0 = aus (bei 0 bleibt die LED dunkel, es muss also immer ein Wert zugewiesen werden damit die LED leuchtet.)
1 = ununterbrochen an
2 = langsames Blinken
3 = normales Blinken
4 = schnelles Blinken
5 = langsames Blitzen
6 = mittelmässiges Blitzen
7 = schnelles Blitzen
8 = langsames auf/abdimmen
9 = mittleres auf/abdimmen
10 = schnelles auf/abdimmen
11 = alter Wert

Dauer: Ein Wert zwischen 0 und 16343, der die Dauer in Sekunden angibt, für die die LED eingeschaltet bleiben soll, nach Ablauf der Dauer wird die LED ausgeschaltet indem die Helligkeit auf 0 gesetzt wird. Die Dauer muss als letzter Parameter zugewiesen werden.

Beispiel von Anweisungen in einem Makro:

```
Taste2LED:=70        // Helligkeit auf 70%
Taste2LED.Farbe:=4   // Farbe auf rot setzen
Taste2LED.Modus:=3   // normales Blinken
warte 1 Sekunde      // damit die gesetzten Werte sofort und damit auf
jeden Fall vor der Dauer gesendet werden
Taste2LED.Dauer:=4   // nach 4 Sekunden Helligkeit auf 0 setzen
```

1.12.4. Shelly-System

Shelly-System

Die Shelly-Geräte kommunizieren über WLAN direkt mit der Zentrale, ohne dass ein zusätzliches Gateway erforderlich ist.
Hersteller der auf dem bekannten ESP8266-Prozessor basierenden Geräte ist Allterco Robotics aus Bulgarien.

Erhältlich sind die Geräte in vielen deutschen Online-Shops wie https://www.technikhaus.de

Zur Konfiguration stellen die Geräte jeweils ein eigenes WLAN zu Verfügung und sind über die jeweils in der Anleitung angegebene IP (normalerweise 192.168.33.1) erreichbar.
Bei der Konfiguration über einen Browser kann dann die Integration in das eigene lokale WLAN vorgenommen werden.

Momentan werden folgende Shelly-Geräte unterstützt:

Shelly1 und **Shelly1PM** inkl. der verschiedenen Addons zu den Geräten
Shelly2.5 als 2-fach-Aktor und als Rollladenaktor
ShellyPlug - Schaltsteckdose
ShellyDuo - Lampe
ShellyVintage - Vintage-Glühbirne
ShellyDimmer - Dimmer UP
ShellyDW2 - Tür/Fensterkontakt
Shellyi3 - Eingabe f. Taster, Schalter, Kontakte
ShellyHT - Temperatur/Luftfeuchtigkeitssensor
ShellyMotion - Bewegungsmelder
ShellyFlood - Wassermelder
ShellyButton1 - Button

1.12.4.1. Allgemeine Gerätekonfiguration

Shelly Gerätekonfiguration

Das zu installierende Shelly-Gerät wird in der linken Listbox im Fenster Modulauswahl ausgewählt und mit dem Roten-Pfeil-Button in die Listbox der verwendeten Geräte übertragen.

Shelly-Aktoren werden über ihre IP-Adresse angesprochen, diese Eingabe wird für den jeweiligen Aktor auf der Shelly-Konfigurationsseite im Browser aktiviert (5) und auf der der Geräteseite der CL-Software eingegeben (1). Es sollte eine feste IP-Adresse verwendet werden, damit diese nicht über DHCP durch den Router geändert wird, was zur Folge hätte, dass das Gerät nicht mehr gesteuert werden kann.
Die Werte und Ereignisse von Shelly-Sensoren werden an den MQTT-Server der CL-Software geschickt, der diese aufbereitet und an die ExecEngine weiterleitet. Im Shelly-Gerät werden die MQTT-Einstellungen (6) und die Eingabe eines Gerätenamens aktiviert (7), es darf kein Benutzername (Username) und Kennwort (Password) vergeben werden. In der Konfigurationsseite des Browsers wird ein Gerätename (2) festgelegt, der in der CL-Software auf der Geräteseite der Sensoren angegeben wird (2). Der Gerätename entspricht dabei der Geräte-Adresse in anderen Systemen.
Zur Eingabe des Gerätenamens wird der Knopf "Gerätename" (3) angeklickt, dieser wird dann in alle Geräte eines Moduls eingesetzt. Je nach Gerät kann der Knopf zur Eingabe des Gerätenamens an verschiedenen Stellen sein (3).
Die IP-Adresse der Zentrale wird unter "Advanced-Developer Settings" im Eingabefeld "Server" der Shelly-Konfigurationsseite im Browser angegeben (4). Der Port, der hinter der IP-Adresse angegeben wird, ist immer 1883.

Nach den Änderungen der Einstellungen des Shelly-Geräts muss dieses neu gestartet werden damit die neuen Einstellungen benutzt werden!
Nach dem Start der Zentrale bzw. der ExecEngine kann es je nach Geräten eventuell mehrere Minuten dauern bis diese sich am MQTT-Server angemeldet haben und Daten übermitteln.

Beim Anlegen der Geräte werden jeweils die passenden Objekte automatisch generiert.
Bei Sensoreingängen muss der Objekttyp entsprechend der im Shelly-Gerät eingestellten Funktion (Taster/Schalter/Kontakt) geändert werden.

1.12.4.2. Shelly1(PM) mit Addons

Shelly1 und Shelly1PM mit Addons

Diese beiden Aktoren haben eine Schnittstelle, an die externe Sensoren angeschlossen werden können.
Daher gibt es bei der in der Liste der zur Verfügung stehenden Geräte für diese Aktoren unterschiedliche erweiterte Gerätebezeichnungen..Dabei wird an die normale Gerätebezeichnung ein Zusatz angehängt, aus dem die jeweils verwendeten Sensoren hervorgehen.

Es gibt folgende Zusätze:

Add-ExtSwitch für einen externen Kontakt oder Taster

Add-T3 für bis zu 3 Temperatursensoren

Add-TH für einen Temperatur/Luftfeuchtigkeitssensor

Je nach ausgewähltem Gerät werden bei der Erstellung des Geräts unterschiedliche Objekte für das Gerät erstellt.

1.12.4.3. Shelly2.5 als Rollladenaktor

Shelly2.5

Dieser Aktor beinhaltet zwei Relais, die für zwei unterschiedliche Geräte verwendet werden können.
Alternativ kann dieser Aktor auch als Rollladenaktor konfiguriert werden.

Es gibt insgesamt drei unterschiedliche Auswahlmöglichkeiten in der Liste der Geräte.

Shelly2 bei Verwendung der Ausgänge für zwei unterschiedliche Verbraucher.

ShellyRollOhnePos bei Verwendung als Rollladenaktor ohne Kalibrierung.
Dabei wird ein Objekt mit den Zuständen oben, stop, unten erstellt.
Beim Zustand stop wird in der Visualisierung eine halb geschlossene Rolllade dargestellt. Damit die Software ermitteln kann ob die Rolllade komplett geöffnet bzw. geschlossen ist oder in einer Zwischenstellung steht, muss die Gesamtlaufzeit der Rolllade im Konfigurationsfenster des Aktors in der CL-Software angegeben werden.

ShellyRollMitPos bei Verwendung als Rollladenaktor mit Kalibrierung.
Diese Option kann nur verwendet werden nachdem die Kalibrierungsfunktion des Aktors erfolgreich durchgeführt wurde, sonst kann der Aktor nicht über die Prozentzahlen zur Angabe der Position gesteuert werden.
Nachdem die Kalibrierung durchgeführt wurde wird auf der Webseite des Aktors ein Schieber zur Positionssteuerung angezeigt und der Aktor kann über einen Zahlenwert für die prozentuale Öffnung gesteuert werden und meldet seine aktuelle Stellung als Prozentzahl.

1.12.5. FHZ-System

FHZ-System

Das FHZ-System wird schon lange nicht mehr produziert und vertrieben und nur noch sehr selten verwendet.
Daher werden in der neuen Version 5.0 nur noch FS20-Module als 1-Kanal-Sender und 1-Kanal-Empfänger unterstützt.

In der Modulauswahl werden keine kompletten Module mehr aufgeführt, sondern nur noch 1-Kanal-Sender und 1-Kanal-Empfänger als Schaltaktor und Dimmer. Damit können dann einzelne noch betriebsfähige FS20-Geräte eingebunden werden
Module mit mehr als einem Kanal können verwendet werden, indem jeder Kanal als einzelnes Modul angelegt wird.

Module werden angelegt im Menüpunkt ***Programmierung->Module->Neues Modul***.

Bevor ein Modul fertig konfiguriert werden kann muss mindestens ein FHZ-Gateway angelegt werden. Als Gateways können FHZ2000 und an einer CCU3 auch die USB-Schnittstelle FHZ1xxxPC benutzt werden.

Beim Anlegen einer FHZ2000 muss die Seriennummer angegeben werden, bei einer FHZ1xxxPC der virtuelle COM-Port der CCU3. Im Normalfall ist das immer COMUSB0, wenn auch andere USB-Gateways benutzt werden kann es auch COMUSB1 sein.
Die Gateway werden unter dem Menüpunkt ***Hardware->Gateways*** angelegt.

Bei einem Aktor-Modul (1-Kanal-Empfänger als Schaltaktor oder Dimmer) muss angegeben werden über welches Gateway gesendet wird.
Die anzugebende FS20-Adresse besteht aus 3 Blöcken mit je 4 Ziffern, die grösste Ziffer die benutzt werden darf ist 4.
In den ersten beiden Blöcken steht der FS20-Hauscode, im letzten die Adresse des Moduls, Grundsätzlich kann auch der Hauscode bei jedem Modul unterschiedlich sein.

Um eine Adresse an einen Empfänger (Aktor) zu vergeben muss dieser in den Anlernmodus versetzt werden und dann an diesen Aktor gesendet werden (z.B. durch Schalten des Aktors in der Visualisierungsansicht).

Um die Adresse eines FS20-Senders herauszufinden kann das PC.-Programm der Version 4.1 benutzt werden, der dort ermittelte Wert kann dann als Adresse eingetragen werden. In Version 5 gibt es keine Option zur Anzeige der empfangenen Adresse eines FS20-Moduls mehr.

1.12.5.1. FS20-Zustandsmelder

FS20-Zustandsmelder

Bitte beachten Sie, dass bei FS20-Zustandsmeldern normalerweise nur eine Meldung kommt, wenn der Einschalt-/Alarmzustand dieses Melders erreicht ist, aber keine Meldung wenn dieser Zustand wieder in den Normalzustand wechselt.
Ein Regensensor sendet z.B. nur eine Meldung wenn es beginnt zu regnen, nicht aber wenn es aufhört. Ein Bewegungsmelder sendet nur eine Meldung in dem Moment wo er eine Bewegung erkennt.
Eine eventuell in dem Zustandsmelder einzustellende Einschaltzeit wird nicht berücksichtigt, da die Funktionen von der Programmierung bestimmt werden sollen.
Das Zurücksetzen des Sensorzustands auf den Normalwert muss vom Programm vorgenommen werden.

Am sinnvollsten geschieht das in dem Skript des Zustandsmelders. Dabei kann die Anweisung WARTE benutzt werden, damit der Zustand einige Zeit am Bildschirm angezeigt wird.

Beispiel bei einem Bewegungsmelder:

```
Anweisungen....
warte 10 Sekunden
Bewegungsmelder ausschalten
```

1.13. Allgemeine Hinweise

1.13.1. Aktoren

Aktoren

Aktoren sind Schaltmodule (also Empfänger), an die Endgeräte (Rollläden, Beleuchtung und andere beliebige Geräte) angeschlossen werden.

1.13.2. Balkendarstellung

Balkendarstellung

Die Balkendarstellung eignet sich für Objekte des Typs Zahl, also z.B. Temperatur, Luftfeuchtigkeit, Windgeschwindigkeit usw.

1.13.3. Dateiverwaltung

Dateiverwaltung

Dateien, die von der CL-Software erzeugt werden, können über den Webserver der Zentrale verwaltet werden.
Das gilt für die History-Datei, die Syslog-Datei und alle mit der Anweisung SCHREIBEDATEI erzeugten Dateien.

Bei der c-comatic wird die Seite zur Dateiverwaltung über den Konfigurations-Webserver aufgerufen.

Bei einer CCU wird diese Seite aufgerufen durch Anklicken des Buttons **[CL-Studio]** unter *Einstellungen->Systemsteuerung* oder unter *Einstellungen->Systemsteuerung->Zusatzsoftware* mit dem Button **[Einstellen]** im Rahmen der Zusatzsoftware.
Auf der Seite die dann erscheint, stehen einige Infos zur aktuellen Programmausführung. Im unteren Bereich gibt es denn Button **[Dateien bearbeiten]** mit dem eine neue Seite zur Dateiverwaltung geöffnet wird.

Auf dieser Seite kann der Zugriff nur auf die vorgegebenen Verzeichnisse der jeweiligen Zentrale erfolgen. Die Verwaltung von anderen Verzeichnissen oder Unterverzeichnissen ist nicht möglich.

1.13.4. Dimmersteuerung

Dimmersteuerung

Besonderheiten bei Dimmern

Dimmer werden vorzugsweise als Schieberegler mit ein/aus Tasten oder nur als Schieberegler dargestellt. Der Wert dieser Schieberegler kann z.B. mit der Maus am Bildschirm oder über eine Skriptanweisung verändert werden.
Um eine bestimmte Helligkeit an einem Dimmer einzustellen wird einfach dem Dimmerobjekt eine Zahl zwischen 0 und 100 zugewiesen. 0 schaltet den Dimme aus, 100 auf volle Helligkeit.
Beispiel:
DimmerDiele:=50

Bitte beachten Sie:
Nachdem am Bildschirm ein neuer Wert für einen Dimmer eingestellt wurde, zeigt dieser die Werte an, die während des Einstellvorgangs gemeldet werden. Er springt dabei meistens auf den alten Wert oder eine geringfügig davon abweichende Einstellung zurück. Erst nachdem der Verstellvorgang abgeschlossen ist, wird der neue aktuelle Wert von der Hardware gesendet und eingestellt.

1.13.5. Duty-Cycle

Duty-Cycle

Zu Funkmodulen im Bereich 868 MHz gibt es gesetzliche Vorschriften, nach denen ein Gerät maximal 1% seiner Betriebszeit senden darf. Im täglichen Betrieb führt das normalerweise zu keiner Einschränkung, aber man muss bei der Programmierung beachten, dass die CCU nicht unnötigerweise zu häufig sendet. Das würde im Extremfall dazu führen, dass das Duty-Cycle-Zeitkonto überläuft und das Senden für ein Zeitlang nicht mehr möglich ist.
Die CL-Software vermeidet unnötige Funksendungen automatisch, indem nur gesendet wird wenn der Zustand eines Aktors sich ändert.
Es gibt aber einige Anweisungen, mit denen der Benutzer selbst Funkmeldungen generieren und dadurch den Duty-Cycle belasten kann.
Das sind folgende Anweisungen:
SENDE
SETZEWERT
ABFRAGE
ALLEABFRAGEN

Diese Anweisungen werden nur in Ausnahmefällen benötigt, die Benutzung sollte dann nur sparsam und mit Bedacht erfolgen um das Duty-Cycle-Konto nicht unnötig zu belasten.
Detaillierte Informationen zu den Anweisungen sind in der Hilfe beschrieben.

1.13.6. ExecEngine

Exec-Engine

Die Exec-Engine ist das Programm, das auf der Zentrale läuft und für die komplette Logik- und Zeitsteuerung der Anwendung zuständig ist. Die Informationen zu den programmierten Anwendungsfunktionen erhält die Exec-Engine über die Ausführungsdatei. Die Ausführungsdatei beinhaltet die in einen speziellen optimierten Binärcode übersetzten Skripts, Zeittabellen und Definitionen, die mit dem Konfigurationsprogramm erstellt worden sind. Auch die Daten für Visualisierungen werden von der Exec-Engine bereitgestellt, wobei die Anzahl der Endgeräte mit denen die Exec-Engine gleichzeitig kommuniziert von der Version abhängig ist.

1.13.7. Freigabe des Programms

Freigabe des Programms

Die Freigabe zur zeitlich unbegrenzten Nutzung des Programms erfolgt durch den Menüpunkt Software->Lizenz durch Eingabe der persönlichen Lizenznummer (PLN).

Die Freigabe wird nur einmal durchgeführt und für die weitere Verwendung gespeichert.

Bitte bewahren Sie Ihre Lizenznummer unbedingt gut auf, es kann sein, dass Sie diese benötigen um künftige Updates zu erhalten oder nach einem Austausch der Zentrale eine erneute Freigabe zu machen.

1.13.8. Gruppen

Gruppen

Über das Anlegen von Gruppen ist es möglich mehreren Objekten gleichzeitig neue Werte zuzuweisen.
Zudem gibt es in Apps die Option Ansichten in Gruppen zu organisieren.

Die zur Verfügung stehenden Gruppen werden unter Einstellungen->Objektgruppen verwaltet.

Auf der Seite zur Verwaltung der Objekte können eine oder mehrere Gruppen ausgewält werden, denen das Objekt angehören soll.

Einem Licht im Erdgeschoss kann also z.B. die Gruppe *Erdgeschoss* und die Gruppe *Licht* zugeordnet werden.

Die Zuweisung von Werten an Gruppen wird durch die Anweisung GRUPPENZUWEISUNG vorgenommen.

1.13.9. History und Log-Dateien

History und Log-Dateien

Bitte beachten Sie:
Diese Features stehen nicht in allen Versionen zur Verfügung.

Um das Schreiben von Historydaten zu aktivieren muss die Checkbox *Historydaten schreiben* auf der Seite Hardware->Zentrale aktiviert werden. Zusätzlich muss für diejedes Objekt für das Historydaten geschrieben werden sollen die Checkbox *Historydaten schreiben* auf der Objektseite aktiviert werden.

Das Schreiben einer Log-Datei mit allgemeinen Informationen zur Ausführung wird über die Checkbox *Syslog schreiben* auf der Seite Hardware->Zentrale aktiviert.

Je nach Art der Zentrale werden die Dateien in unterschiedliche Verzeichnisse geschrieben. Das jeweilige Verzeichnis kann im Reiter *Zentrale* des Hardwarefensters festgelegt werden. Bei der CCU1 kann das Verzeichnis nicht verändert werden, da nur das vorgegebene Verzeichnis verfügbar ist.

Bitte unbedingt beachten:
Wenn eine CCU1 oder CCU2 verwendet wird muss ein USB-Stick bzw. eine SD-Karte eingebunden werden, wenn in den internen Speicher der CCU geschrieben wird kommt es nach einiger Zeit zu Fehlfunktionen weil der interne Speicher überläuft.
Hinweise zum Einbinden eines USB-Sticks bei der CCU1 bzw. einer SD-Karte bei der CCU2 finden Sie im Kapitel SCHREIBEDATEI.
History und Log-Datei können mit einem Browser angezeigt, downgeloadet und gelöscht werden.

Bei der CCU1 und CCU2 muss dazu folgende Eingabe in der Adresszeile des Browser gemacht werden:
http://<IP der Zentrale>/addons/contronics/files/workfiles.cgi
also zum Beispiel
http://192.168.0.136/addons/contronics/files/workfiles.cgi

Bei der CCU3 wird ein anders Verzeichnis benutzt:
http://<IP der Zentrale>/addons/cl-control/workfiles.cgi
also zum Beispiel
http://192.168.0.53/addons/cl-control/workfiles.cgi

1.13.10. Objekte

Objekte

Die Objekte sind die zentralen Elemente für alle Steuerungsfunktionen.

Zunächst die Erklärung zu einigen Begriffen, die verwendet werden um ein Objekt zu beschreiben:

Aktoren sind Schaltmodule, also Empfänger, die von der Zentrale gesteuert werden. Wenn ein Modul mehrere Ausgänge hat, so ist jeder Ausgang ein separater Aktor.
Sensoren sind Sendemodule, die Meldungen an die Zentrale senden, wobei jede Taste einer Fernbedienung bzw. jeder Kanal eines Sensors ein separater Sensor ist.

Der Begriff **Objekt** umfasst einen Sensor bzw. einen Aktor inklusive der gesamten Definition, die für diesen in der Konfiguration gemacht wurde (Darstellungsart, Skripts Schaltzeiten usw.)
Wenn der Objektzustand eines Aktor- Objekts durch Visualisierung oder Skript geändert wird, so wird automatisch an den zugehörigen Aktor eine Meldung gesendet und dieser schaltet auf den Zustand des Objekts.
Wenn ein Sensor-Objekt eine Meldung seines Sensors empfängt, ändert sich der Zustand des Objektes entsprechend und das Skript des Objekts wird ausgeführt.
Objekte können während der Ausführung eines Projektes in verschiedenen Ansichten am Bildschirm angezeigt werden. Die Zustände von Objekten können normalerweise durch Mausklick verändert werden.
Es kann auch Objekte geben, denen kein Sensor oder Aktor zugeordnet ist, diese werden *virtuelle Objekte* genannt. Virtuelle Objekt können auf der Seite *Programmierung->Objekte* angelegt werden.

Es gibt auch ein automatisch erzeugtes virtuelles Objekt, das Objekt Anwesenheitssimulation, das automatisch immer dann erzeugt wird, wenn auf der Seite Anwesenheitssimullation etwas eingetragen wurde. Diese Objekt ist vom Typ Schalter, hat aber kein zugeordnetes Hardware-Modul, sondern dient nur dazu die Anwesenheitssimulation zu aktivieren bzw. zu deaktivieren.
Ein weiteres virtuelles Objekte ist das zu jedem Projekt automatisch erzeugten Objekte *Uhr.* Das Objekt *Uhr* ist ein Anzeigefenster, in dem immer die aktuelle Uhrzeit (im 5-Sekunden-Rhythmus angezeigt wird).
In der Standard-Version wird automatisch ein Objekt vom Typ Zeichen mit dem Namen Anzeige erzeugt. Dieses Objekt kann zur Ausgabe von beliebiger werte benutzt werden.
In der Studio-Version können beliebig viele "virtuelle" Objekte mit beliebigem Typ angelegt werden, also auch beliebig viele Anzeige-Objekte, daher wird hier kein Anzeige-Objekt automatisch erstellt.

1.13.11. Objektdarstellung

Objektdarstellung

Für ein Objekt gibt es verschiedene Darstellungsmöglichkeiten. Es kann als Objektrahmen, Textrahmen oder als kleines Bild-Symbol (Bitmap) dargestellt werden. Welche Darstellungsart aktuell verwendet wird kann auf der Seite Visualisierung der Objektdefinitionen festgelegt werden.

Für Objekt mit mehreren möglichen Zuständen kann für die unterschiedlichen Zustände jeweils ein anderes Bild-Symbol definiert werden. Wenn nur ein Bild zur Visualisierung für ein Objekt ausgewählt wird, kann man aufgrund der Position des Objekts im Hintergrundbild erkennen um welches Objekt es sich handelt.

1.13.12. Objektrahmen und Textrahmen

Objektrahmen und Textrahmen

Objektrahmen und Textrahmen dienen dazu ein Objekt und seinen Zustand oder Wert auf dem Bildschirm darzustellen. In einem Objektrahmen wird neben der Ausgabe der Objektbezeichnung und dem aktuelle Zustand als Text im rechten Teil auch ein Symbol angezeigt, wenn für dieses Objekt ein Symbol ausgewählt wurde. Durch Anklicken eines Objektrahmens oder Textrahmens kann der Zustand des Objekt verändert werden.
Die Grösse und Position für Objektrahmen und Textrahmen der einzelnen Objekte kann im Entwurfsfenster festgelegt werden. Dieses Entwurfsfenster wird mit dem Knopf **[Ansicht bearbeiten]** auf der Seite Ansichten des Einstellungsfensters geöffnet.

1.13.13. Programmierung

Programmierung

Die Programmierung des Systems erfolgt über eine eigene Scriptsprache mit vielen leistungsfähigen Anweisungen.

Die Anweisungen werden auf der Seite Script der Objektdefinitionen eingegeben.

1.13.14. Projekt

Projekt

Ein Projekt ist eine individuell erstellte Anwendung mit den ausgewählten Modulen, den Skripts, programmierten Funktionen und erstellten Definitionen.

Das Projekt wird in einer Datei mit der Endung SPG gespeichert.

Bitte beachte Sie unbedingt:
Die Projektdatei muss regelmässig gesichert werden, da bei einem Festplattenfehler alle Konfigurationen und Programmierungen verloren gehen.
Die Konfigurationen und Programmierungen werden nicht in der Zentrale gespeichert!

Keinesfalls sollte der Projektname öfters geändert werden, weil sowohl auf der Zentrale, in Apps und auch auf dem PC Dateien und Daten in Abhängigkeit vom Projektnamen erzeugt und gespeichert werden.
Für die Sicherungen muss natürlich ein anderer Name vergeben werden, aber der Projektname, der auf die Zentrale übertragen wird sollte möglichst nicht verändert werden. Bei Änderungen des Namens kommt es dann z.B. bei Apps die Einstellungen unter dem Projektnamen speichern zu scheinbaren Datenverlusten und es entstehen unnötige Dateien auf der Zentrale..
Mit dem Menüpunkt Projekt->Schnellsicherung wird eine Sicherungsdatei erzeugt, bei der Datum und Uhrzeit an den Projektnamen angehängt werden. Speichern Sie diese Dateien nicht nur auf der Festplatte, sondern auch auf einem Datenträger, z.B. einem USB-Stick.

1.13.15. Schieberegler

Schieberegler

Schieberegler werden benutzt um Zahlenwerte einzustellen. Ein Anwendungsbeispiel ist die Einstellung der Helligkeit eines Dimmers. Der aktuelle Wert des Moduls wird in einem Schieberegler oben rechts angezeigt. Der Wert des Schiebereglers kann manuell am Bildschirm mit der Maus oder durch ein Skript verändert werden.

1.13.16. Sensoren

Sensoren

Sensoren sind Eingabeelemente wie z.B. Taster von Fernbedienungen, Bewegungsmelder, Temperaturfühler, Wassermelder, Rauchmelder usw.

1.13.17. Sonnenzeiten

Sonnenzeiten

In einigen Zeitfenstern kann als Zeitpunkt zur Ausführung einer Aktion die Zeit des Sonnenaufgangs bzw. des Sonnenuntergangs für das aktuelle Datum angegeben werden. In wenn-Bedingungen können diese Zeiten zum Vergleich mit der aktuellen Uhrzeit verwendet werden. Zur Eingabe des Zeitpunkt *Sonnenaufgang* geben Sie anstelle der Uhrzeit die Buchstaben **SA** ein, für den *Sonnenuntergang* **SU**.
Es wird der Zeitpunkt der sogenannten „bürgerlichen Dämmerung“ berechnet. Es gibt noch andere Definitionen zu Sonnenaufgangs- und –untergangszeiten (z.B. astronomische Sonnenzeiten, nautische Sonnenzeiten), die teilweise in Abhängigkeit der Jahreszeiten stark voneinander abweichen. Daher sind diese Zeiten eventuell nicht identisch mit Angaben in Tabellen, in denen andere Sonnenzeiten verwendet werden. Die berechneten Zeiten sind bis auf wenige Minuten genau – Vorausetzung ist die Angabe des geografischen Standorts nach Postleitzahl oder Längen- und Breitengrad.

Die Zeiten für Sonnenaufgang und Sonnenuntergang können im Einstellungsfenster im Reiter **Allgemein** mit dem Button "weitere Einstellungen" individuell angepasst werden.

1.13.18. Spezielle Objekte

Spezielle Objekte

Es können spezielle Objekte angelegt werden, die dazu dienen besondere Situationen, Warnungen oder Fehlermeldungen zu bearbeiten oder bestimmte Systemwerte einzustellen.
Spezielle Objekte sind normalerweise virtuelle Objekte, die je nach Art der Verwendung als Objekte vom Typ Zeichen, Schalter oder Zahl angelegt werden.
Existieren keine passenden Objekte für eine Funktion, so bleibt die Auswahlliste für diese Objektart leer.
Details zur Verwendung der einzelnen speziellen Objekte sind jeweils im zugehörigen Abschnitt beschrieben.

Level für History
Hier kann ein vorher definiertes Objekt vom Typ Zahl angegeben werden. Dieser Wert kann über Skripts während der Programmausführung geändert werden.
Die Zahl kann folgende Werte haben:
0 : Es werden hat werden keine Historydaten geschrieben, auch wenn diese Option für ein Objekt eingestellt ist.
1 : Es werden Historydaten geschrieben für alle Objekte, für die diese Option aktiviert ist.
2 : Es werden Historydaten für alle Hardware-Objekte und für alle Objekte, für die die Historyoption aktiviert ist, geschrieben.
3 : Es werden Historydaten für alle Objekte geschrieben, auch für Objekte für die die Historyoption nicht aktiviert ist.
Bitte beachten Sie unbedingt: Bei Level 2 und 3 können grosse Datenmengen anfallen. Insbesondere bei Verwendung einer CCU muss darauf geachtet werden, dass die Historydatei nicht in den normalen Speicher, sondern auf die SD-Karte oder einen USB-Stick geschrieben, wird um Fehlfunktionen aufgrund eines Speicherüberlaufs zu vermeiden.

Einstellung für Systemlog
Hier kann ein vorher definiertes Objekt vom Schalter oder Zahl angegeben werden. Beim Schalterzustand "ein" werden die Historydaten der Stufe 1 geschrieben, ist das Objekt vom Typ Zahl werden Systemmeldungen der in der Zahl angegebenen Stufe geschrieben. 0=kein Systemlog, 1=nur wichtige Systemmeldungen wie Ausführungsstart, Ausführungsende und Fehler, 2=zusätzlich werden Warnungen in die Log-Datei geschrieben, 3=es werden Fehler, Warnmeldungen und Informationen geschrieben. Die Stufe 3 kann grössere Datenmengen erzeugen und sollte im Normalfall nicht verwendet werden. Beachten Sie, dass es bei einem Speicherüberlauf zu gravierenden Fehlfunktionen und einem Systemabsturz kommen kann. Das gilt insbesondere bei Verwendung einer CCU ohne DS-Karte oder USB-Stick.

Objekt für Verbindungsfehler
Hier kann ein vorher definiertes Objekt vom Typ Zeichen angegeben werden. Jedesmal wenn eine Meldung über eine Kommunikationsstörung empfangen wird, wird der Name des Objekts, bei dem die Kommunikationsstörung aufgetreten ist als Wert für das *Objekt für Verbindungsfehler* gesetzt und das Script wird ausgeführt.

Allgemeines Fehlerobjekt
Hier kann ein vorher definiertes Objekt vom Typ Zeichen angegeben werden. Wenn im System ein Fehler erkannt wird, werden Informationen dazu in dieses Objekt geschrieben und das Skript des Objekts wird ausgeführt.

Objekt für Sabotagemeldungen
Hier kann ein vorher definiertes Objekt vom Typ Zeichen angegeben werden. Jedesmal wenn eine Sabotagemeldung empfangen wird, wird der Name des Objekts das die Sabotagemeldung gesendet hat als Wert für das *Objekt für Sabotagemeldungen* gesetzt und sein Skript wird

ausgeführt.

Objekt für Batteriemeldungen
Hier kann ein vorher definiertes Objekt vom Typ Zeichen angegeben werden. Jedesmal wenn eine Batteriewarnung empfangen wird, wird der Name des Objekts, das die Batteriewarnung gesendet hat als Wert für das *Objekt für Batteriemeldungen* gesetzt und das Skript wird ausgeführt.

Hauptschalter für Heizen
Hier kann ein vorher definiertes virtuelles Objekt vom Typ Schalter angegeben werden. Beim Schalterzustand "aus" wird die Temperatursteuerung der Thermostate für die Binäraktoren zum Heizen deaktiviert. Mit diesem Schalter ist es möglich in Abhängigkeit von der Jahreszeit oder der aktuell gemessenen Aussentemperatur die Heizfunktion über den Binäraktor komplett zu deaktivieren oder zu aktivieren.

Hauptschalter für Kühlen
Hier kann ein vorher definiertes virtuelles Objekt vom Typ Schalter angegeben werden. Beim Schalterzustand "aus" wird die Temperatursteuerung für den Binäraktoren zum Kühlen deaktiviert. Mit diesem Schalter ist es möglich in Abhängigkeit von der Jahreszeit oder der aktuell gemessenen Aussentemperatur die Kühlfunktion über den Binäraktor komplett zu deaktivieren oder zu aktivieren.

1.13.19. Visualisierung

Visualisierung

Als Visualisierung wird die Darstellung der einzelnen Objekten und deren aktueller Zustände bzw. Werte bezeichnet.

Die Visualisierung erfolgt über einen beliebigen Browser auf einem beliebigem Endgerät.

Die Seiten zur Visualisierung werden mit dem Menüpunkt Programmierung->Visualisierung erstellt. Dabei können die Objekte an beliebiger Stelle auf der jeweiligen Visualisierungsseite positioniert werden.

1.13.20. Zeilenvorschub

Zeilenvorschub

Wenn bei der Ausgabe von Texten über ein Objekt vom Typ Zeichen kann ein Zeilenvorschub mit zwei<<-Zeichen eingefügt werden.

Beispiel:

```
Textausgabe:="Das ist die erste Zeile<<und das ist die zweite Zeile."
```

1.13.21. Zustandsmelder

Zustandsmelder

Bitte beachten Sie, dass bei Zustandsmeldern normalerweise nur eine Meldung kommt wenn der besondere Zustand dieses Melders erreicht ist, aber keine Meldung wenn dieser Zustand wieder in den Normalzustand wechselt.
Ein Regensensor sendet z.B. nur eine Meldung wenn es beginnt zu regnen, nicht aber wenn es aufhört, ein Bewegungsmelder sendet nur eine Meldung in dem Moment wo er eine Bewegung erkennt.

Eine eventuell in dem Zustandsmelder einzustellende Einschaltzeit wird nicht berücksichtigt, da die Aktionen auf die Meldung von der Programmierung bestimmt werden.
Auch das Zurücksetzen des Sensorzustands auf den Normalwert muss vom Programm vorgenommen werden, damit die Visualisierung korrekt ist.

Im Reiter Allgemein der Objektdefinitionen kann im Eingabefeld "Resetzeit in Sekunden" eine Zeit angegeben werden nach der der Zustand des Objekts automatisch wieder auf den AUS-Zustand gebracht wird.

Eine andere Option ist, dieses Zurücksetzen in den Normalzustand im Skript des Objekts zu machen.
Dabei kann die Anweisung warten oder WARTE benutzt werden, damit der Zustand einige Zeit am Bildschirm angezeigt wird.

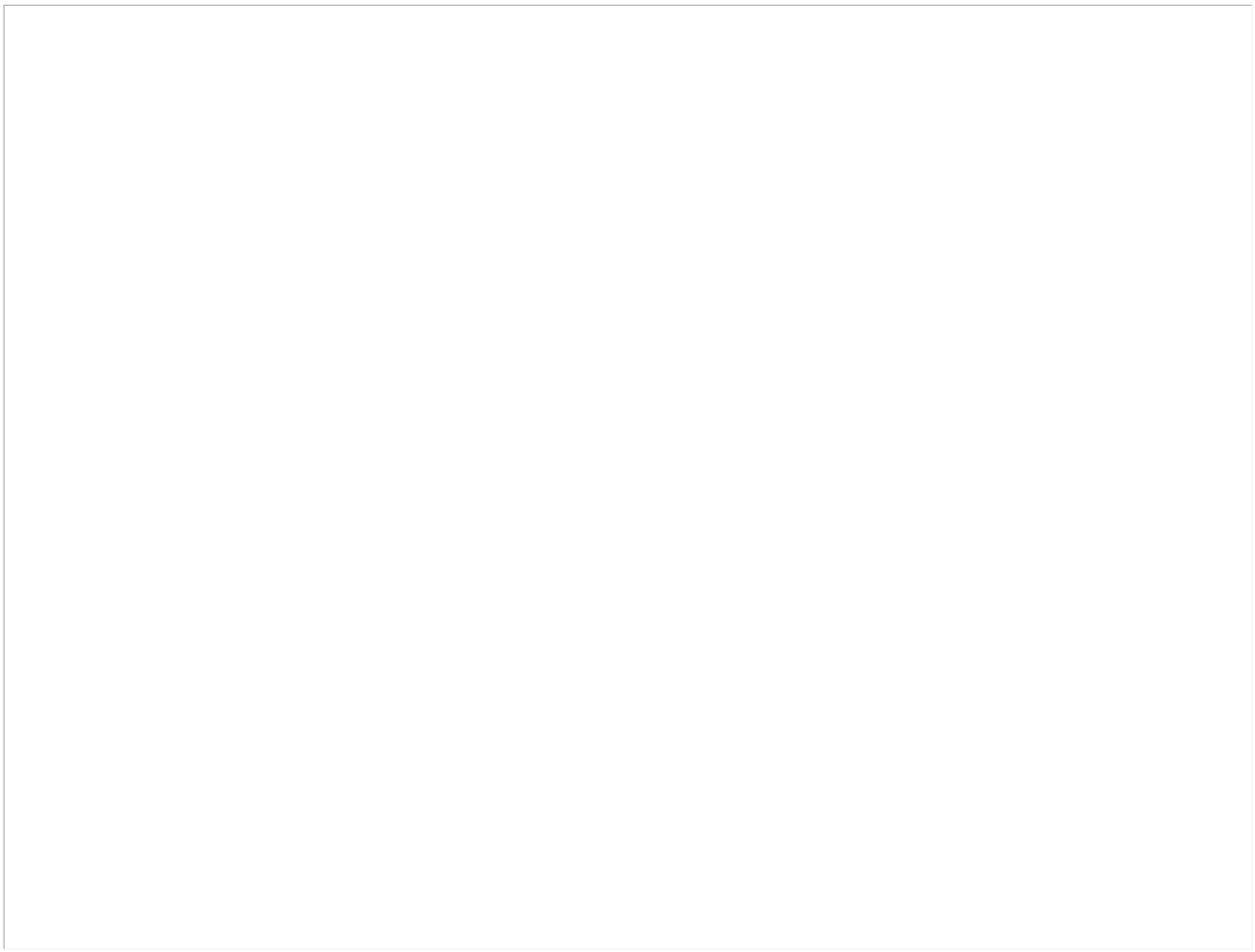
Beispiel bei einem Bewegungsmelder:

```
Anweisungen....
warte 10 Sekunden
Bewegungsmelder ausschalten
```

1.13.22. Variablen über virtuelle Objekte setzen

Variablen über virtuelle Objekte setzen

Bei einigen Geräten (z.B. ZigBee-Geräten) werden spezielle Eigenschaften über Variablen gesteuert. Nicht in allen Visualisierungsoptionen stehen spezielle Funktionen zur Steuerung solcher Geräte zur Verfügung. In dem Fall können einfach virtuelle Objekte erstellt werden, deren Werte dann die Variablen des Geräts setzen.
Der Objektwert einer ZigBee-Farbleuchte ist der Ein/Aus-Zustand des Objektes. Um die Farbe über eine Visualisierung zu setzen, in der kein spezielles Objekt für die Farbleuchte zur Verfügung steht wird dann einfach ein virtuelles Objekt von Typ Zahl mit dem Wertebereich 0-65535 angelegt und im Skript des Objekts die Farb-Variable des Geräts gesetzt, wie im Screenshot gezeigt.



1.14. FAQs

Allgemein
Wenn scheinbar grundlos plötzlich diffuse Probleme auftreten oder normale Funktionen nicht mehr funktionieren hilft es oftmals die Zentrale (CCU) neu zu starten.

Wenn die Kommunikation mit der Zentrale nicht funktioniert lässt sich das in den meisten Fällen beheben, indem die CCU länger als 20 Sekunden stromlos gemacht wird damit sie komplett neu startet.

Problem:
Das PC-Programm hängt beim Start, am unteren Rand des Fensters wird die Meldung "Initialisierung und Verbindungsaufbau" angezeigt.
Das passiert wenn keine Verbindung zur Zentrale mit der angegebenen gespeicherten IP-Adresse hergestellt werden kann.

Lösung:
Drücken Sie unmittelbar nach dem Programmstart irgendeine Taste auf der Tastatur (solange "Verbindungsaufbau deaktivieren mit Tastendruck..." angezeigt wird).
Dann wird nicht versucht eine Verbindung zur Zentrale aufzubauen und Sie können die Verbindungseinstellungen prüfen.
Oftmals ist die Ursache, dass die IP-Adresse der Zentrale durch DHCP geändert wurde.

Wenn es nicht an einer falschen IP-Adresse liegt kann das Problem meistens durch einen Neustart der Zentrale behoben werden, hilft ein einfacher Neustart nicht kann es sein, dass die Zentrale länger als 20 Sekunden stromlos gemacht werden muss.

Eine andere Ursache kann sein, dass die Verbindung nach einem Update des Virenscanners blockiert wird, in dem Fall muss die Freigabe der Verbindung
im Virenscanner freigegeben werden.

Problem:
Es wird eine neue Freigabe mit der Lizenznummer verlangt, aber diese funktioniert nicht.

Lösung:
Meistens liegt es daran, dass nicht die reguläre PLN benutzt wird, sondern eine temporäre PLN für einen Testzeitraum.
Eine reguläre PLN hat ein "W" an 5. Stelle, eine temporäre PLN hat ein "T" an 5. Stelle.
Unter diesem Link finden Sie eine Anleitung zur Freigabe:
http://www.cl-control.de/downloads/pdf/FreigabeCLStudio.pdf

Sollte die Freigabe trotz erfolgreicher neuer Freigabe erneut angefordert werden, so wurde diese wahrscheinlich nicht
korrekt in der Zentrale gespeichert. In diesem Fall hilft es normalerweise die Zentrale neu zu starten und die Freigabe dann zu wiederholen.

1.15. Online-Informationen

Online-Informationen

Aktuelle Informationen und Updates zum Programm finden Sie im Internet unter:

http://www.cl-control.de/software.html

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

O

P

R

S

T

U

V

W

Z