CUx-Daemon Dokumentation

Version 0.9

mit wireless M-Bus, EnOcean,
1-Wire, ArtDMX, WebCam, ...

Letzte Änderung: 24.01.2015

Uwe Langhammer (ulangham@gmx.de)

Inhaltsverzeichnis

1 Einleitung....4
2 Installation / Update / Deinstallation....7
3 Administrations-Interface....9
3.1 Status....9
3.2 Terminal....11
3.3 Setup....12
3.4 Info....13
3.5 Geräte....14
3.5.1 Gerät bearbeiten....15
3.6 Filebrowser....16
4 Anlegen von CUxD-Geräten....17
5 Verfügbare Geräte....20
5.1 Wettersensoren {CUX}, {WDE1}....22
5.1.1 (32) Temperatursensor....23
5.1.2 (01) Temperatur/Luftfeuchte-Sensor....24
5.1.3 (31) Kombisensor KS200/KS300....26
5.2 FS20-Geräte {CUX}....28
5.2.1 (03) FS20-Sensor (1-Kanal)....30
5.2.2 (02) FS20-Schaltaktor (1-Kanal)....34
5.2.3 (04) FS20-Dimmaktor (1-Kanal)....36
5.2.4 (05) FS20-Relais (1-Kanal)....38
5.3 Energie-Sensoren {CUX}....40
5.3.1 (06) EM1000 Energie-Sensoren....40
5.3.2 (27) ESA1000 / ESA2000 Energie-Sensoren....43
5.4 FHT-Heizungssteuerung {CUX}....45
5.4.1 (07) FHT8v Ventilantrieb....45
5.4.2 (08) FHT80b Wandthermostat....46
5.4.3 (09) Multiventil-Steuerung (8 Räume mit FHT8v-Stellantrieben)....52
5.4.4 (10) TF-2 Tür/Fensterkontakt (2-Kanal)....55
5.5 HMS-Sensoren und Gefahrenmelder {CUX}....56
5.5.1 (12) HMS 100 TF (Temperatur/Luftfeuchte-Sensor)....57
5.5.2 (13) HMS 100 T (Temperatursensor)....58
5.5.3 (14) HMS 100 W/WD (Wassermelder)....59
5.5.4 (15) HMS 100 RM / RM 100-2 (Gefahrenmelder)....60
5.5.5 (18) HMS 100 MG (Gefahrenmelder)....61
5.5.6 (20) HMS 100 CO (Gefahrenmelder)....62
5.5.7 (26) HMS 100 FIT (Gefahrenmelder)....63
5.5.8 (16) HMS 100 TFK (Tür-/Fensterkontakt)....64
5.6 (29) BidCos Geräte {CUX}....65
5.6.1 Füllstandsmesser KFM 100 S....65
5.7 (90) Universal Wrapper Devices....67
5.7.1 (1) Transform Device....67
5.7.2 (2) State-Monitor Device....70
5.7.3 (3) Thermostat Device....74
5.8 (28) System-Devices....83
5.8.1 System.Timer (16 Kanäle)....84
5.8.2 System.Exec (16 Kanäle)....88
5.8.3 System.Multi-Dim-Exec (1 + 16 Kanäle)....95
5.8.4 System.Ping (16 Kanäle)....98

5.9 (91) CloudMatic 100
5.9.1 CloudMatic.Mail....101
5.9.2 CloudMatic.SMS....102
5.9.3 CloudMatic.Push....103
5.9.4 CloudMatic.Cloud....104
5.9.5 CloudMatic.Webcam....105
5.10 Sonstige Geräte....110
5.10.1 (11) RS232-Füllstandsmesser {SONIC}....110
5.10.2 (40) 16 Kanal Universalsteuerung....111
6 Zusatzprogramme....117
6.1 artdmxdim (DMX: 512 Kanäle)....117
6.2 ccu_backup....120
6.3 dom_save....120
6.4 dom_backup....120
6.5 ether-wake v1.09....121
6.6 digitemp_DS9097U v3.5.0....121
6.7 export_ftp.sh....122
6.8 timer.tcl....122
7 Konfiguration....123
8 Daten-Logging....130
8.1 CUxD-HighCharts....133
9 DFU Firmware-Installation/Update über den CUx-Daemon....134
10 FAQ....137

1 Einleitung

Der CUxD ist eine universelle Schnittstelle zwischen der CCU-Logikschicht (ReGa HSS) und externen (auch virtuellen) Geräten. Um die CCU-Ressourcen (Speicher / Prozessor) optimal zu nutzen, wurde der CUx-Daemon (CUxD) als natives C-Programm implementiert. Er beinhaltet eine einfache Web-Oberfläche zur Administration und Verwaltung der CUxD-Geräte. Der Vorteil dieser Lösung besteht darin, dass sie im Gegensatz zu anderen verfügbaren Produkten, ausschließlich auf der HomeMatic-CCU läuft. Weil kein extra Rechner benötigt wird, halten sich die Betriebskosten und Investitionen in zusätzliche Hardware in Grenzen.

Dieses Projekt wurde bis zum Februar 2011 von Alex Krypthul als Schnittstelle für den CUL- bzw. CUN-Stick von Busware.de (daher der Name) entwickelt. Mittlerweile werden aber auch eine Vielzahl weiterer Protokolle und Funktionen mit und ohne Zusatzhardware unterstützt.

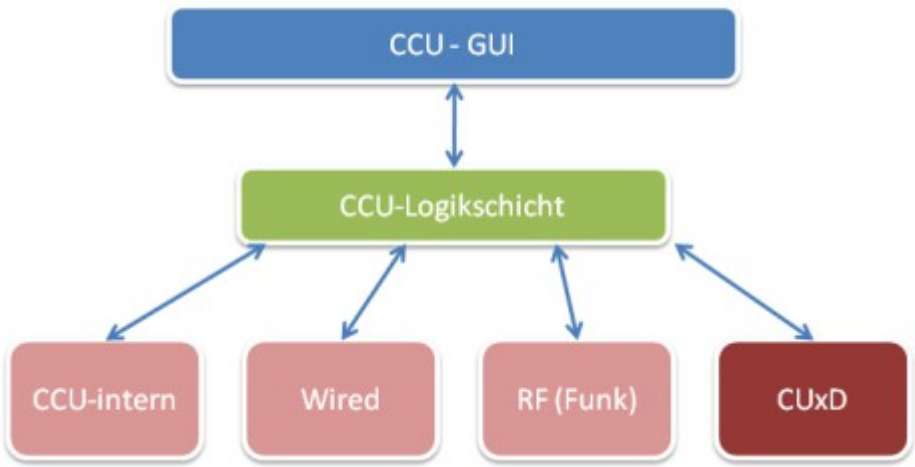
Die HomeMatic-CCU unterstützt standardmäßig drei Gerätetypen:

1. Die Zentraleinheit (CCU) selbst
2. Wired-Geräte (RS485-Bus mit HomeMatic-Protokoll)
3. Funkgesteuerte Geräte (HomeMatic-Protokoll)

Der CUxD erweitert zum einen die Funktionalität der CCU und mit entsprechender USB-Zusatzhardware werden auch viele weitere Protokolle (FS20, EnOcean, 1-Wire, ArtDMX, usw.) unterstützt.

Dabei erfolgt über die USB-Schnittstelle sowohl die Stromversorgung, als auch die Kommunikation zwischen dem CUxD und den angeschlossenen Geräten. Sollten die beiden USB-Ports der CCU nicht ausreichen, so können auch USB-Hubs (ggf. mit eigener Stromversorgung) zur Erweiterung eingefügt werden.

Der CUx-Daemon bildet eine (Software-) Schnittstelle zwischen der Zusatzhardware und der CCU. Um eine möglichst benutzerfreundliche Integration der Zusatzgeräte in die Benutzeroberfläche (WebUI) und Logikschicht (ReGa HSS) der CCU zu ermöglichen, wurde ein eigener **RPC-Server** implementiert, der beim Booten der CCU als weitere **Kommunikationsschnittstelle** in die CCU-Logikschicht eingebunden wird. Die grafische Darstellung der neuen Geräte auf der WebUI der CCU erfolgt dann über virtuelle „original“ HomeMatic-Geräte.



Die Logik für die Kommunikation und die Verarbeitung der Daten der angeschlossenen Geräte wird im CUxD durch das erkannte Gerät an der USB-Schnittstelle (automatisch oder manuell mittels **TTYASSIGN**) und den ausgewählten **CUxD-Gerätetyp** definiert.

Aktuell wird folgende Hardware mit den genannten CUxD-Gerätetypen unterstützt:

Hardware	getestete Geräte
ESP2/ESP3 EnOcean Gateway PioTek EnOcean Platine für CCU2	- EnOcean Taster (2-, 4-, 8-Kanal) - EnOcean Drehgriffkontakt - EnOcean Tür-/Fensterkontakt - PioTek EnOcean Tracker - EnOcean Temperatur- und Luftfeuchte-Sensoren - EnOcean Bewegungsmelder - EnOcean Schaltaktor, Dimmer, Jalousieaktor, Stellantrieb - EnOcean bidirektionaler Funktionsstecker mit Verbrauchsmessung - **RAW**-Datenverarbeitung über 16 Kanal Universalsteuerung
busware CUL/CUN/CUNO **über USB** http://www.busware.de/ http://culfw.de	- **ALLE** FS20 Geräte (Sensoren, Aktoren) - ELV-Wettersensoren (KS200, KS300, S300IA, S300TH, ASH2200, PS50) und dazu kompatible Geräte - ELV-EM1000 Energiemonitore - ESA1000/ESA2000 Energiemonitore - FHT80 (FHT80b Raumthermostat, FHT8v Ventilantrieb, FHT80 TF-2) - HMS100 Sensoren und Gefahrenmelder - KFM100S kapazitiver Füllstandsmesser - 1-Wire Temperatursensoren (DS18S20) mit HMS-Emulation über CUNO - 433MHz Lacrosse TX3 Temperatur und Luftfeuchte-Sensoren - **RAW**-Datenverarbeitung für **alle vom CUL unterstützten Geräte und Protokolle** über 16 Kanal Universalsteuerung
USB-WDE1 http://www.elv.de	- ELV-Wettersensoren (KS200/300, S300IA, S300TH, ASH2200, PS50) und dazu kompatible Geräte
RFXtrx433 über USB	- **RAW**-Datenverarbeitung über 16 Kanal Universalsteuerung
DS9097U USB-Adapter (z.B. LinkUSB, LinkUSBi)	- 1-Wire Sensoren mittels DigiTemp
Arduino (UNO, MEGA, ...)	- Emulation der Protokolle sämtlicher im CUxD implementierten Geräte
Vellemann 8090	- 8 Kanal USB-Relaiskarte über 16 Kanal Universalsteuerung
USB to RS232 Adapter - Prolific PL2303 - Moschip MOS7720 - Silabs CP210x - FTDI - CH341 (nur CCU2)	- RS232 Ultraschall Füllstandsmesser: http://www.icplan.de/seite25.htm - UM100, UM2102, UO2102 und UM-FT2232H - **RAW**-Datenverarbeitung für **alle Geräte und Protokolle** über 16 Kanal Universalsteuerung
ohne	- Universal-Wrapper - System-Funktionen (System.Timer, .Exec, .Ping) - CloudMatic Geräte (Mail, SMS, Push, Cloud) + Webcam - Device-Log
USB-Speichersticks SD-Karten	- Mount/Umount/Automount nach Neustart und auf der CCU1 bei Stromausfall (vor automatischer USB-Deaktivierung)

Werden Funk-Gateways in Form von USB-Sticks an der CCU angeschlossen, dann sollte dafür unbedingt eine USB-Verlängerung genutzt werden. Beim direkten Anschluss können Empfangsstörungen auf dem externen Gateway sowie der CCU auftreten und die Stabilität des Systems negativ beeinflussen.

Bei einem CUxD-Versions-Update werden alle bereits angelegten Geräte „aktualisiert“, d.h. bei Änderungen und Erweiterungen der Geräteeigenschaften in neueren CUxD-Versionen müssen nur in Ausnahmefällen einzelne CUxD-Geräte gelöscht und neu angelegt werden.

So funktioniert die Einbindung von CUxD-Geräten aus Sicht des Anwenders:

1. Nachdem die externe USB-Hardware konfiguriert wurde (automatisch oder mit Hilfe von TTY-Parametern) wird über die CUxD-Adminoberfläche der gewünschte Gerätetyp (z.B. FS20-Dimmer) ausgewählt und diesem eine eindeutige Seriennummer (am besten von 1 hoch zählen) zugewiesen. Optional kann hier auch ein Name für dieses Gerät vergeben werden. Den Namen kann man später jederzeit über die WebUI ändern. Zusätzlich ist ein der CCU bekanntes Geräte-Icon (z.B. „Zwischenstecker Dimmer“) auszuwählen. Dieses Icon dient nur zur Darstellung des Gerätes in der HomeMatic-WebUI.
2. Nach dem Anlegen erscheint das Gerät im „Posteingang“ der CCU und kann abschließend über die CCU Weboberfläche (WebUI) konfiguriert werden. Die weitere Vorgehensweise entspricht neu angelernten HomeMatic-Geräten.
3. Nach dem Abschluß der Konfiguration erscheint das Gerät in der WebUI und kann vom Anwender wie ein HomeMatic-Gerät bedient werden. Es kann von CCU-Programmen angesprochen und über das „Gerätemenü“ konfiguriert und gelöscht werden. „Direkte Geräteverknüpfungen“ werden dabei nicht unterstützt.

Da die WebUI nicht auf alle Funktionen der „neuen“ CUxD-Geräte ausgelegt ist, werden CUxD-Geräte nicht immer so „elegant“, z.B. mit deutschen Beschriftungen und angepassten Icons und Symbolen (dies ist leider fest in der WebUI hinterlegt) dargestellt. Um eine vollständige „elegante“ Integration zu ermöglichen, wäre nach jedem Firmware-Update ein Firmware-Patch notwendig, der aber der Prämisse eines möglichst geringen Einflusses auf die CCU widerspricht. Dies ist und wird nicht Bestandteil des CUxD-Projektes.

2 Installation / Update / Deinstallation

Die Installation auf der CCU erfolgt über das WebUI-Menü „Systemsteuerung → Zusatzsoftware“. Bei der Installation werden alle CUxD-Dateien auf der CCU im Verzeichnis „/usr/local/addons/cuxd/“ installiert.

Um den CUxD RPC-Server automatisch beim Hochfahren der CCU zu starten, wird zusätzlich die Startdatei **cuxdaemon** im Verzeichnis „/usr/local/etc/config/rc.d/“ abgelegt,.

Zusätzlich trägt sich der CUxD bei jedem Start als eigener RPC-Server in die Datei /etc/config/InterfacesList.xml ein. Diese Datei wird bei jedem CCU-Start neu geschrieben.

Die CCU überprüft beim Neustart zuerst alle bereits vorhandenen Schnittstellen (Wired, Funk). Das dauert je nach Anzahl der installierten HomeMatic-Geräte mehrere Minuten.

WebUI-Darstellung:

Während dieser Zeit ist der CUxD aber bereits gestartet und kann über das eigene Webinterface (**http://<AdresseDerCCU>/addons/cuxd/**) direkt angesprochen werden.

Allerdings erfolgt zu diesem Zeitpunkt noch keine Kommunikation zwischen der CCU-Logikschicht und dem CUxD. Auf der CUxD-Statusseite wird das durch entsprechend rot markierte Hinweise dargestellt:

- Nicht mit HomeMatic-CCU IP-Adresse:Port verbunden.
- Nicht als RPC-Server von der CCU angefordert.

Sobald ein Login auf der CCU-WebUI möglich ist, ändert sich der Text und diese Einträge werden grün. Die CUxD-Statusseite aktualisiert sich dabei nicht automatisch, sondern muss im Browser **manuell aktualisiert** werden!

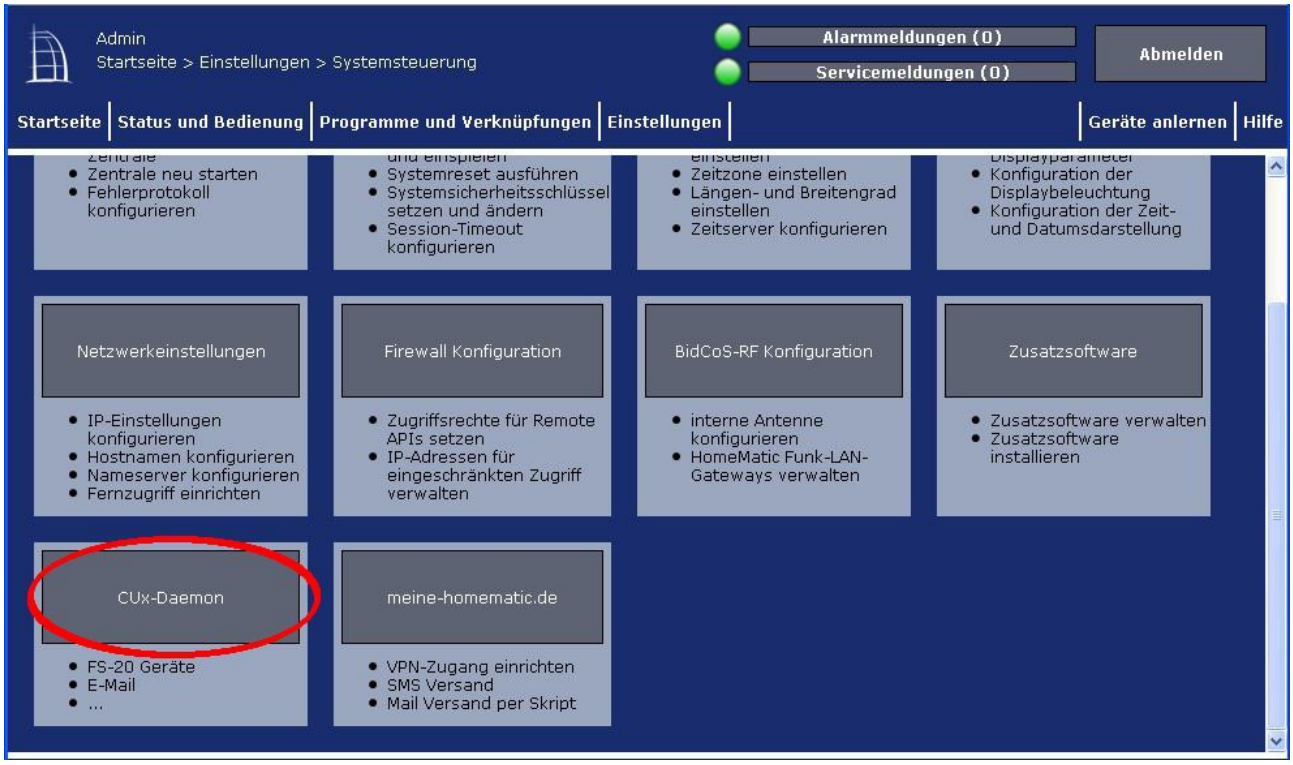
Wird der CUxD nach dem Start der WebUI nicht als RPC-Server von der CCU angefordert (2. roter Hinweis), so hilft nur ein Neustart der CCU.

Nach erfolgreicher Erstinstallation des CUxD muss die CCU nach dem Anlegen des ersten CUxD-Gerätes ein zweites Mal durchgestartet werden. Erst danach ist das CUxD-Interface vollständig betriebsbereit und kann auch Befehle von der CCU empfangen.

Erstinstallation:

1. CUxD installieren (automatischer Neustart der CCU)
2. CUxD Gerät(e) anlegen und im Posteingang der CCU bestätigen
3. CCU ein weiteres Mal durchstarten!
4. das CUxD-Interface ist jetzt funktionsbereit.

Der Zugriff auf das Administrations-Interface erfolgt entweder über die CCU-System-steuerung:



oder direkt über die CUxD-URL: **http://<AdresseDerCCU>/addons/cuxd/**.

Das **Update** auf eine neuere Version erfolgt einfach durch Neuinstallation über eine bestehende Installation im WebUI-Menü „Systemsteuerung → Zusatzsoftware“. Dafür muss die alte CUxD-Version **nicht deinstalliert** werden.

Nach einem Update sollten dann auch alle zuvor konfigurierten Geräte mit ihren Einstellungen vorhanden sein. Sie werden beim Update automatisch aktualisiert und müssen nur noch in Ausnahmefällen gelöscht und neu angelegt werden.

Bei jedem Versionsupdate wird die CUxD-Gerätekonfiguration zusätzlich in der Datei **cuxd.ps.old** gesichert. Im Fehlerfall kann man diese Datei zur **manuellen Wiederherstellung** „verlorengegangener“ Geräte(parameter) nutzen.

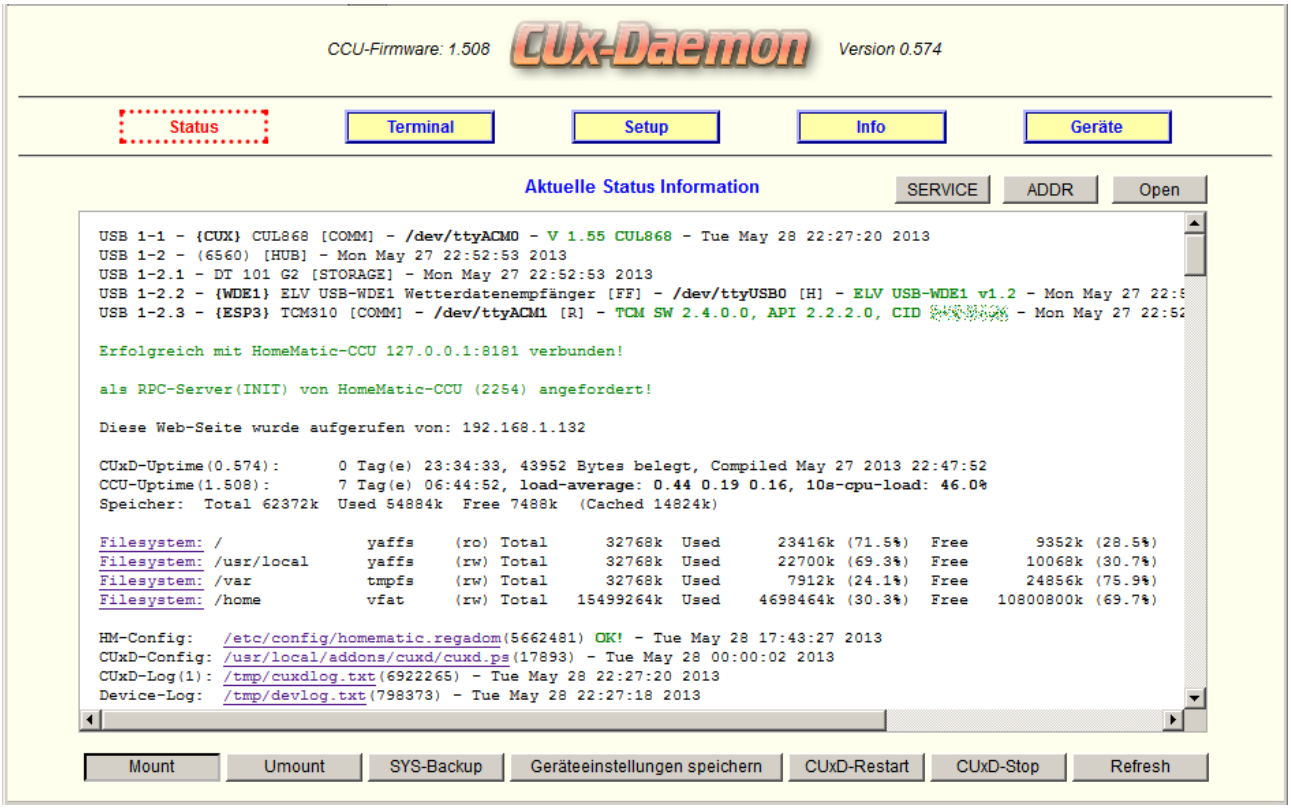
Die **Deinstallation** der Software erfolgt über die CCU-WebUI „Systemsteuerung → Zusatzsoftware“.

Achtung: Vor einer CUxD Deinstallation sollten alle CUxD-Geräte aus der CCU gelöscht werden, da die CCU zum Löschen der Geräte den RPC-Server des CUxD benötigt. Ein späteres Löschen ist erst nach erneuter CUxD-Installation wieder möglich.

Damit der CUxD nach einem Reboot der **CCU1** immer eine einigermaßen gültige Systemzeit bekommt und interne Intervall-Timer richtig initialisiert werden, empfiehlt sich die zusätzliche Installation des **settime**-AddOn's.

3 Administrations-Interface

3.1 Status



Auf der Statusseite erhält man einen Überblick über den Systemzustand der CCU und des CUxD. Ganz oben wird der Status aller vorhandenen USB-Geräte angezeigt.

Die folgenden beiden Zeilen beschreiben den Zustand der Schnittstelle zur CCU-Logik.

Bei einem Klick auf den Filesystem-Link wird der interne Filebrowser auf dem gewählten Volume gestartet.

In der HM-Config-Zeile wird hinter dem homematic.regadom File angezeigt, ob diese Systemdatei beim letzten Mal vollständig (**OK!**) oder unvollständig (**ERROR!**) gespeichert wurde. Bei einem Fehler gibt es beim nächsten CCU-Neustart sehr wahrscheinlich Probleme

Die Tasten haben folgende Bedeutung:

Mount

- der im Parameter **MOUNTCMD=** definierte Befehl wird ausgeführt. Diese Taste bleibt bis zum nächsten Umount gedrückt und der Status wird gespeichert.
- ist die Taste gedrückt, dann wird der im Parameter **MOUNTCMD=** definierte Befehl bei jedem CUxD-Start automatisch ausgeführt (Automount).

Umount

- der im Parameter **UMOUNTCMD=** definierte Befehl wird ausgeführt.

SYS-Backup

- der im Parameter **BACKUPCMD=** definierte Befehl wird ausgeführt.

Geräteeinstellungen speichern

- die aktuellen Geräteeinstellungen werden in das CUxD-Konfigurationsfile geschrieben. Bei aktiviertem „**AUTOSAVE=1**“ ist das nicht mehr notwendig.

CUxD-Restart

- der CUx-Daemon wird angehalten und automatisch neu gestartet. Dies ist in der Regel nicht notwendig.

CUxD-Stop

- der CUx-Daemon wird angehalten. Ein manueller Neustart ist danach nur aus der Systemsteuerung oder per Telnet möglich.

Refresh

- die Statusseite wird aktualisiert

SERVICE

- es öffnet sich eine neue Seite mit Zusatzfunktionen, wie z.B. Filebrowser, Prozessliste, CCU-Backup, CCU-Restart., Root-Passwort ändern, Shell-Befehle ausführen. Diese Seite (http://<ip_der_ccu>/addons/cuxd/maintenance.html) kann auch bei gestopptem CUxD aufgerufen werden.

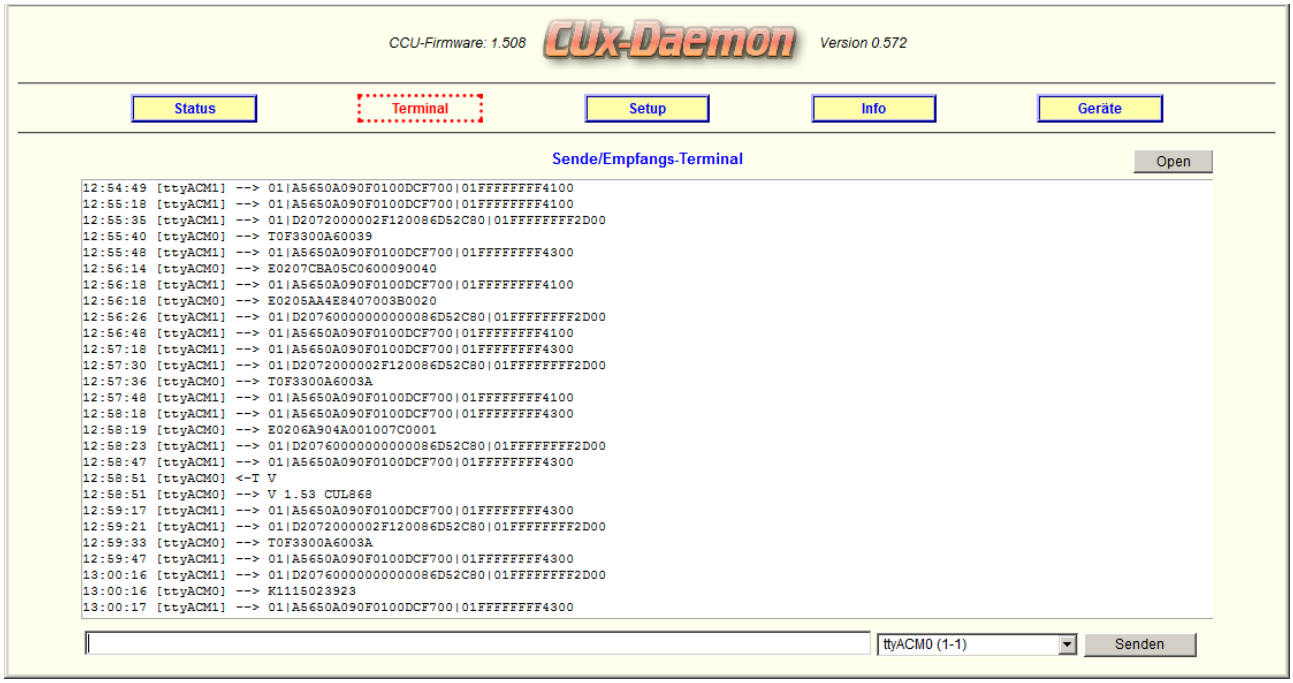
ADDR

- Anzeige der empf. Adressen (Seite 20) in einem neuen Fenster mit Aktualisierung

Open

- Anzeige der kompletten Statusseite in einem neuen Fenster

3.2 Terminal



Über die Terminal-Seite können Befehle an die ausgewählte USB-Schnittstelle gesendet werden. Es werden außerdem alle empfangenen Daten angezeigt. Mit dem Parameter **TTYHIDE** kann die Anzeige für ausgewählte Schnittstellen verhindert werden.

Ganz links steht in jeder Zeile das TTY, gefolgt von der Uhrzeit, dem Pfeil für die Datenübertragungsrichtung und den Daten. Wenn ein Befehl vom Terminal gesendet wurde, dann steht ein „T“ am Pfeil.

In der untersten Zeile können Daten (Befehle) eingegeben werden, die durch Drücken der „**Senden**“-Taste an das ausgewählte TTY gesendet werden. In dieser Zeile können auch mehrere {CUX} Befehle, durch Leerzeichen getrennt, hintereinander eingegeben werden. Bei anderen Geräten wird die komplette Zeile gesendet. Nach jeder Zeile bzw. jedem Befehl wird beim Senden normalerweise automatisch (außer im **TTYHEX**-Mode) ein CRLF hinzugefügt.

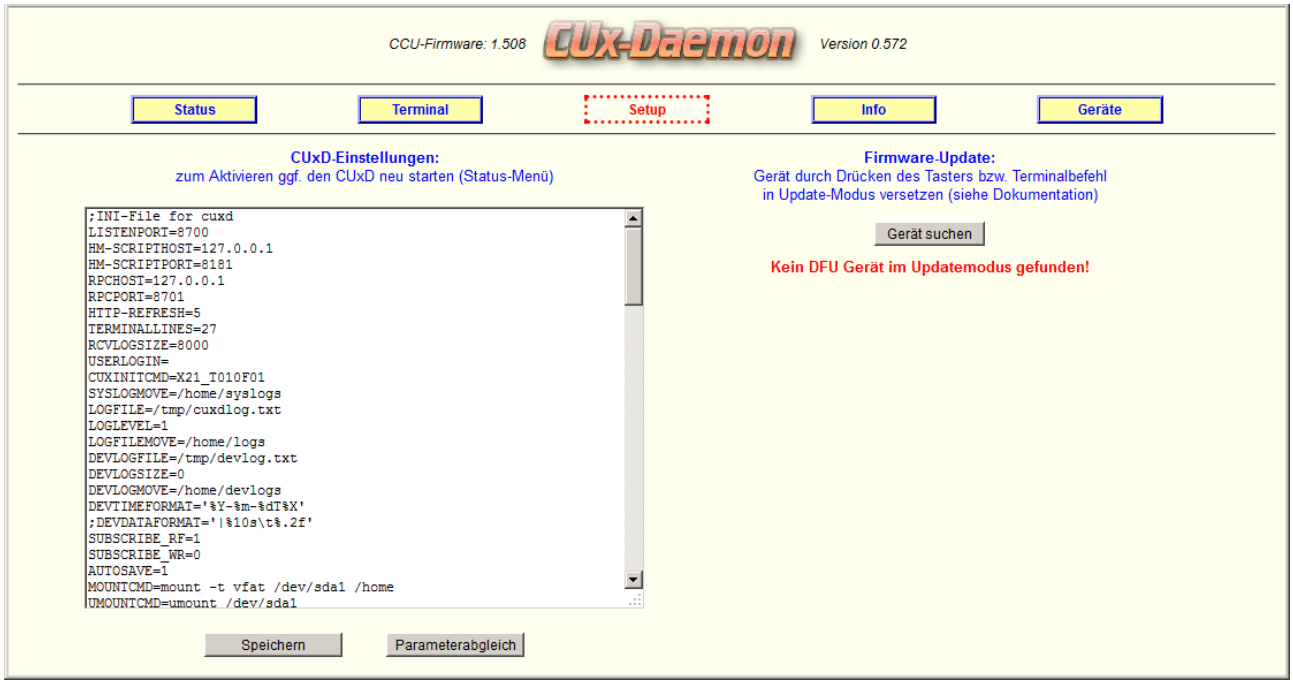
Die folgenden Sonderzeichen werden vor dem Senden nach C-Standard ersetzt: \\, \a, \b, \f, \t, \n, \r, \v, \xHH (HH ist eine 2-stellige Hexadezimalzahl). Endet die Zeile mit einem einzelnen Backslash \, dann wird am Zeilenende kein CRLF gesendet.

Weitere Tasten:

Open

- Anzeige des kompletten Terminal-Logs in einem neuen Fenster. Die Größe des Puffers kann mit dem CUxD-Parameter **RCVLOGSIZE=** definiert werden.

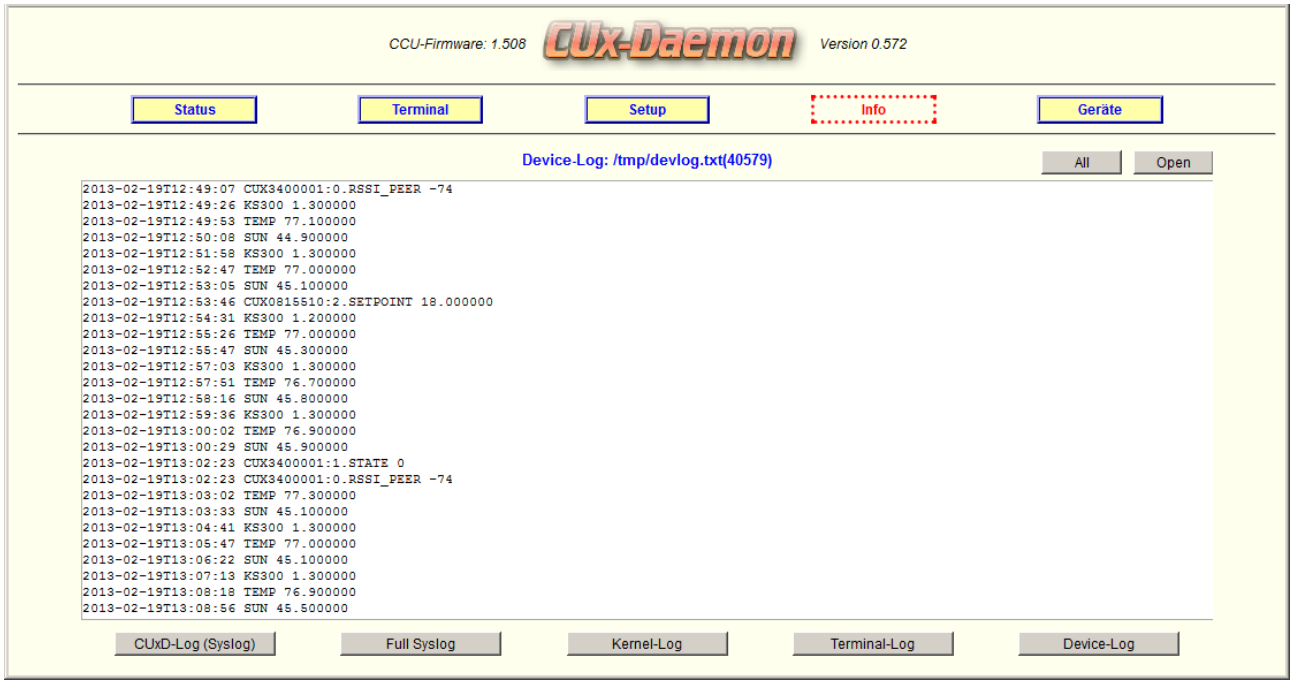
3.3 Setup



Auf der Setup-Seite können CUxD-Einstellungen geändert und die CUN/CUL-Firmware aktualisiert werden. Geänderte CUxD-Einstellungen sind (bis auf die ersten 5 Parameter) sofort nach dem „Speichern“ aktiv.

Zusätzlich kann das INI-File über die Taste „Parameterabgleich“ aktualisiert werden. Das ist immer dann sinnvoll, wenn nach einem Versionsupdate neue Parameter hinzugekommen sind.

3.4 Info



Auf der Info-Seite werden Log-Meldungen angezeigt.

Bei Problemen ist zuerst das CUxD-Log auf Fehlermeldungen zu prüfen.

Ist ein Device-Log aktiviert, so kann es über eine extra Taste ausgewählt werden.

Nach dem ersten Aufruf der Seite werden nur die letzten Meldungen angezeigt. Die Anzahl der Zeilen kann über den Parameter **TERMINALLINES=** konfiguriert werden.

Wenn das CUxD-Highcharts Addon installiert ist, dann sind auf der Device-Log Seite zusätzliche Formularfelder zum Aufruf des Addons eingeblendet.

Weitere Tasten:

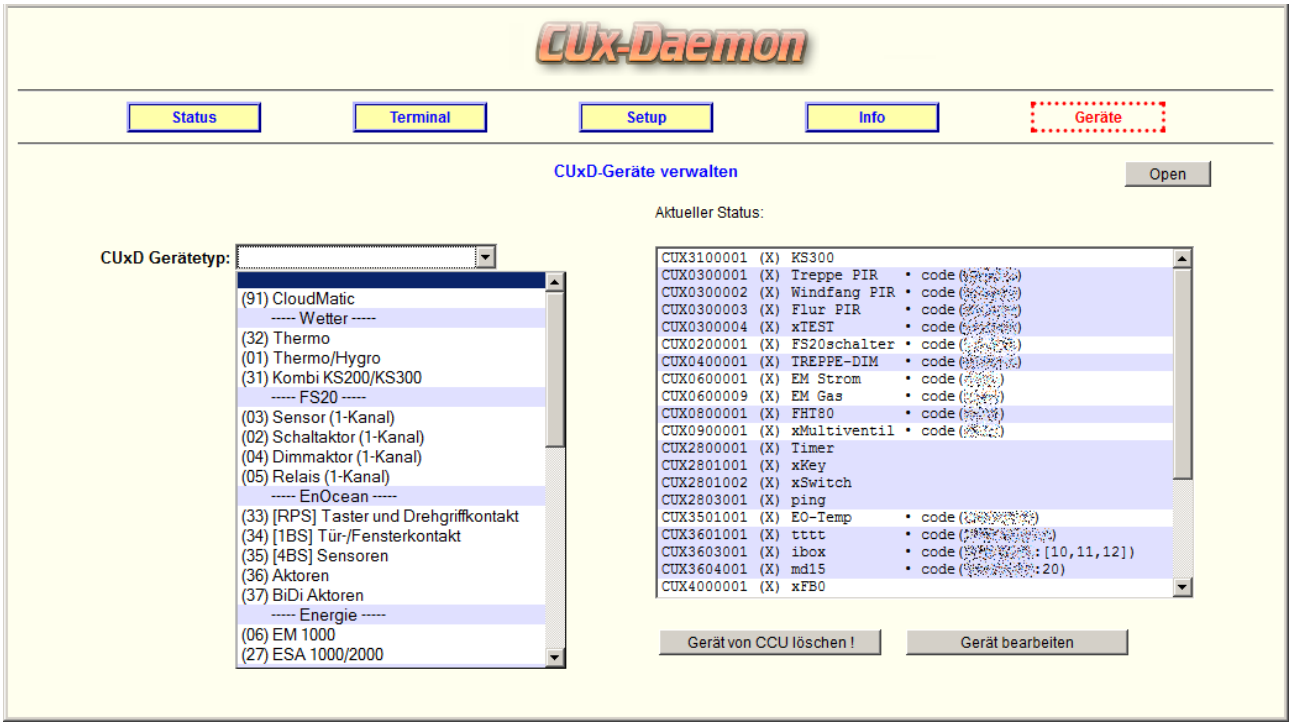
All

- das ausgewählte Log wird vollständig geladen.

Open

- das ausgewählte Log wird vollständig in einem neuen Fenster angezeigt.

3.5 Geräte



Auf der Geräte-Seite können CUxD Geräte gelöscht, neu angelegt und geändert werden. Außerdem werden zur Übersicht in der Auswahlliste alle angelegten Geräte mit deren CCU-Status (X)-konfiguriert, (?)-unkonfiguriert und den eingetragenen DEVICE- und CODE-Parametern angezeigt.

Weitere Tasten:

Open

- der Status aller Geräte wird zur Übersicht in einem neuen Fenster angezeigt

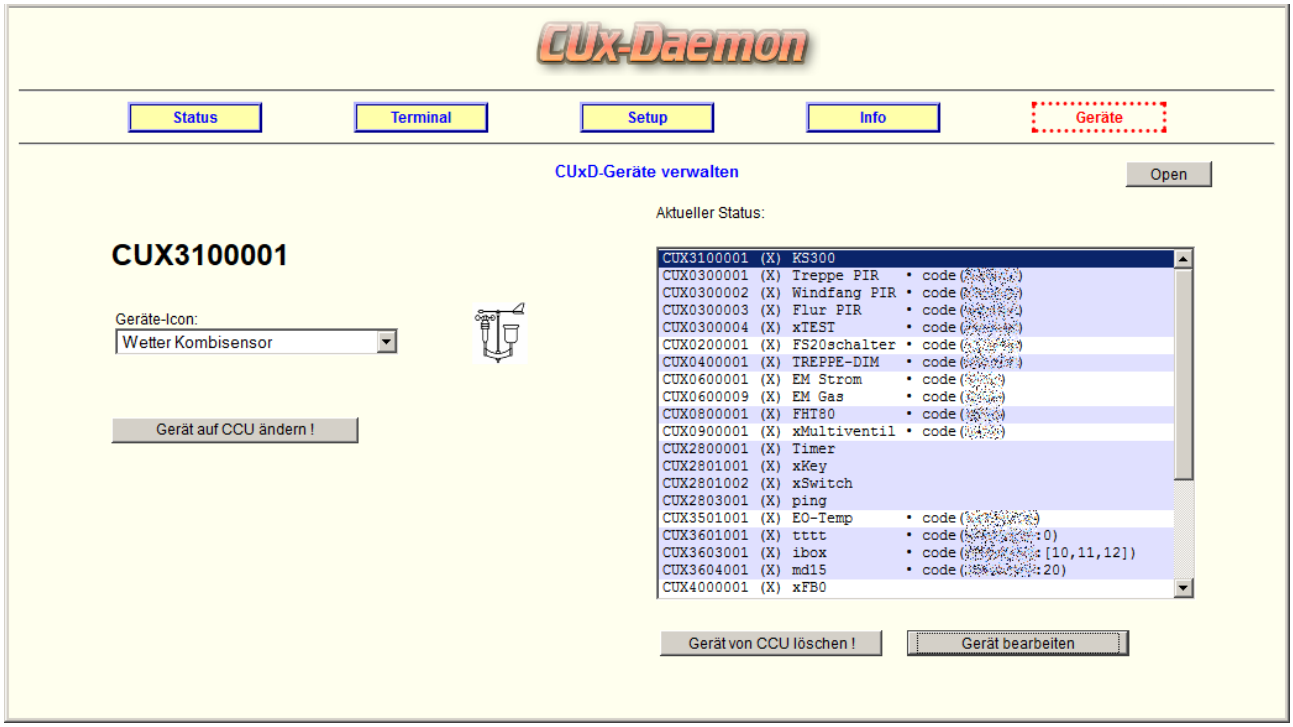
Gerät von CCU löschen

- das ausgewählte CUxD-Gerät wird von der CCU gelöscht

Gerät bearbeiten

- der Gerätetyp (das Icon) des ausgewählten CUxD-Gerätes kann geändert werden

3.5.1 Gerät bearbeiten

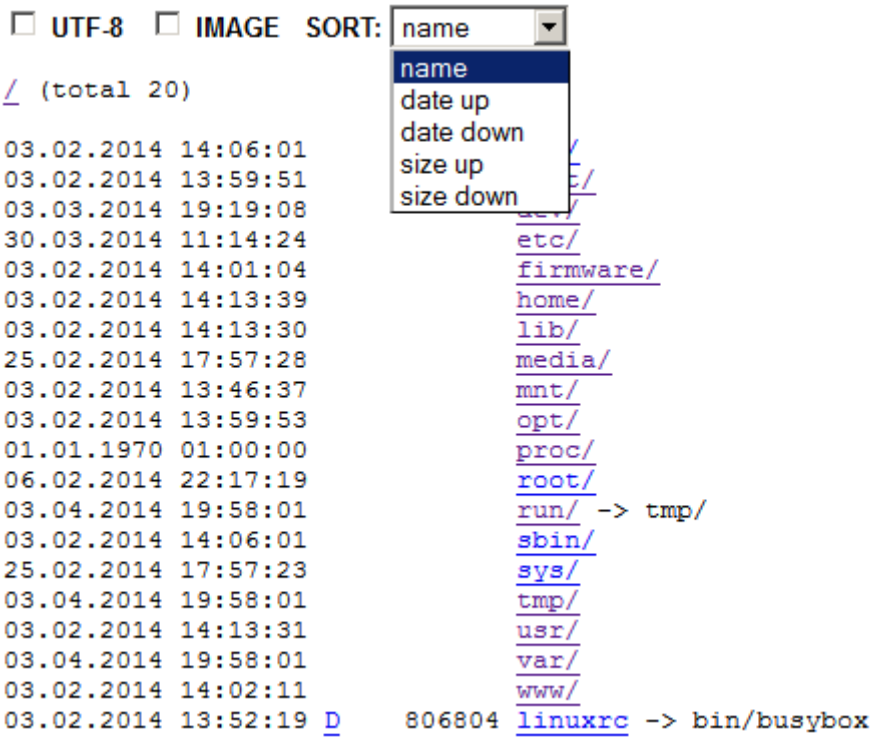


Hier kann der Gerätetyp (das Icon) von bereits konfigurierten CuxD-Geräten nachträglich geändert werden.

3.6 Filebrowser

Der CUxD besitzt einen integrierten HTML-Filebrowser. Darüber können Dateien ohne weitere Zusatzsoftware direkt im Webbrowser angesehen und heruntergeladen [D] werden. Wahlweise können auch die Zeichenkodierungen auf UTF-8 angepasst, und Bilder als Thumbnails im Browser dargestellt werden.

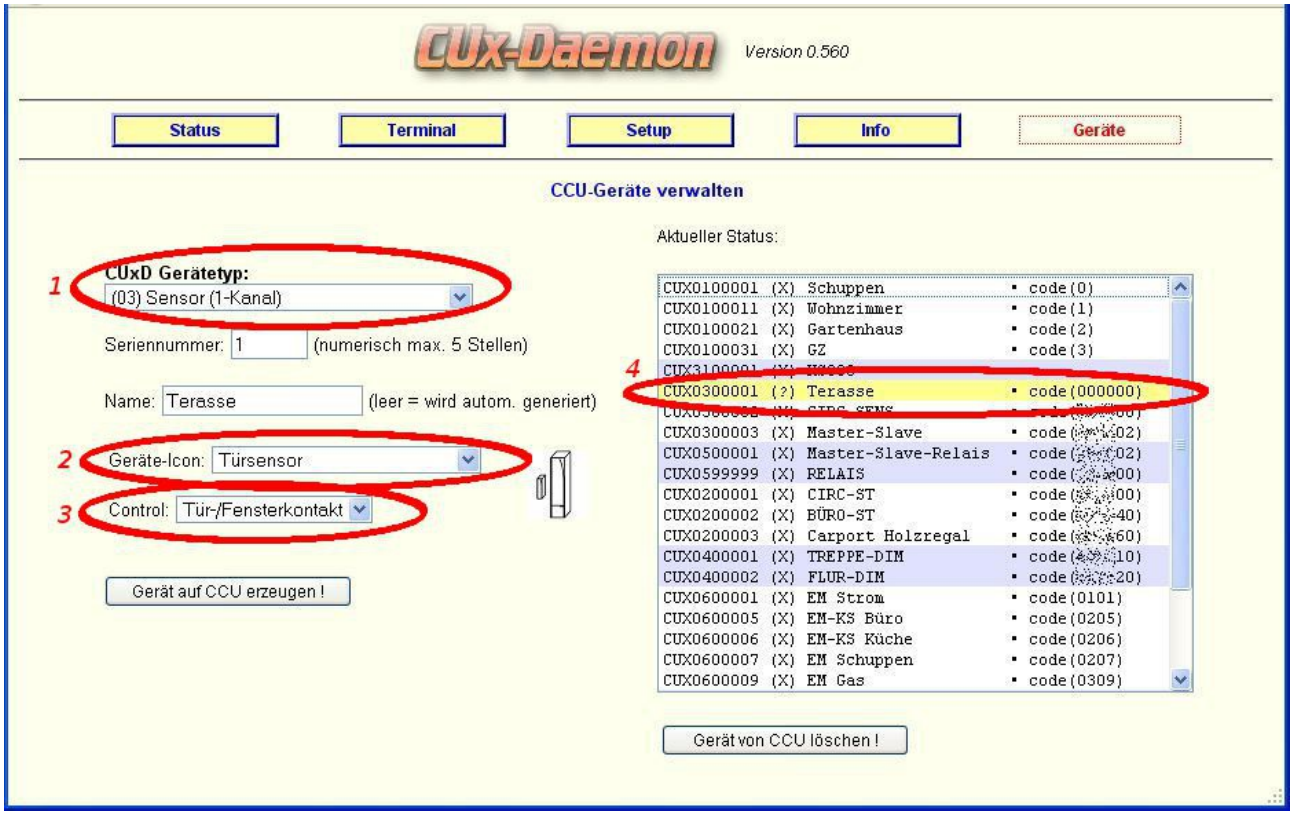
Die Sortierung der Verzeichnisansicht kann mittels einer Select-Box ausgewählt werden.



4 Anlegen von CUxD-Geräten

Zur Einbindung in die Verarbeitungslogik der CCU müssen die Geräte zuerst im CUxD angelegt werden. Danach befinden sie sich im Posteingang der CCU und werden von dort wie echte HomeMatic-Geräte in die Benutzeroberfläche (WebUI) übernommen.

Im folgenden wird das ausführlich am Beispiel eines FS20-Sensors beschrieben. Das Anlegen erfolgt über die Administrationsoberfläche „Geräte“:



Nach der Auswahl des **CUxD-Gerätetyps** **(1)**, der die ganze Funktionalität des Gerätes beinhaltet, erscheinen ergänzenden Eingabefelder.

Eine eindeutige **Seriennummer** ist für jedes Gerät zwingend erforderlich. Ab CUxD-Version 0.568 wird sie im Webbrowser automatisch gesetzt. Die Seriennummer besteht hier aus einer maximal 5-stelligen Dezimalzahl. Zusammen mit dem fest definierten Teil „**CUX**“ und dem gewählten Gerätetyp ergibt sich daraus dann die 10-stellige HM-Seriennummer des Gerätes auf der CCU (z.B. CUX0300001). Sie dient zur Identifikation jedes Gerätes und muss eindeutig sein. Bei der Eingabe wird das geprüft. Am einfachsten ist es, beim ersten Gerät mit 1 anzufangen und die Nummer bei jedem weiteren Gerät des gleichen Typs um 1 zu erhöhen. In der Liste auf der rechten Seite bekommt man einen Überblick, über die bereits vergebenen CUX-Seriennummern.

Die Angabe eines **Namens** (maximal 50 Zeichen) ist optional - er kann später über die CCU-Weboberfläche geändert werden. Die Eingabe in der CUxD-Administrationsoberfläche hat den Vorteil, dass gleichzeitig auch alle Kanäle des Gerätes diesen Namen erhalten (.z.B. „Terasse:1“).

Das gewählte **Geräte-Icon** **(2)** dient nur zur Darstellung des neuen Gerätes in der WebUI und hat ansonsten keinen weiteren Einfluss auf die Gerätefunktionalität. Zur Auswahl steht eine Liste von vordefinierten Icons der WebUI.

Abhängig vom CUxD-Gerätetyp können weitere Eingabefelder **(3)** angezeigt werden. In unserem Beispiel besteht die Möglichkeit, das WebUI-Control des Sensors zu definieren.

Bei den Wettersensoren besteht an dieser Stelle zum Beispiel die Möglichkeit, Statistiken hinzuzufügen.

Abschließend wird das Gerät über die Taste „**Gerät auf CCU erzeugen!**“ angelegt. Nun erscheint es mit dem Status **(?)** in der Listbox **(4)** auf der rechten Seite im Format:

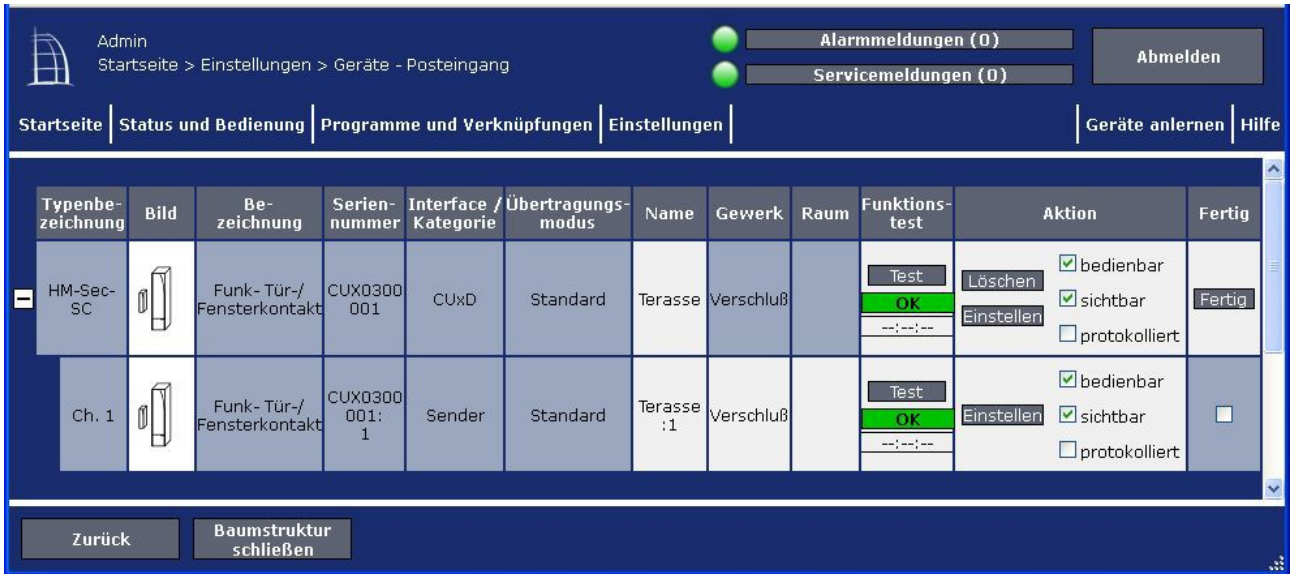
„Seriennummer (Status) Name ▪ extra“

Die Felder sind folgendermaßen definiert:

Seriennummer	CUXttnnnnn tt – Gerätetyp nnnnn – 5-stellige Seriennummer (zum Teil frei definierbar)
Status	(?) - unkonfiguriert (im Posteingang der WebUI) (X) - konfiguriert
Name	zuvor manuell vergeben oder automatisch generiert
extra	Gerätekonfiguration, z.B. dev(), code()

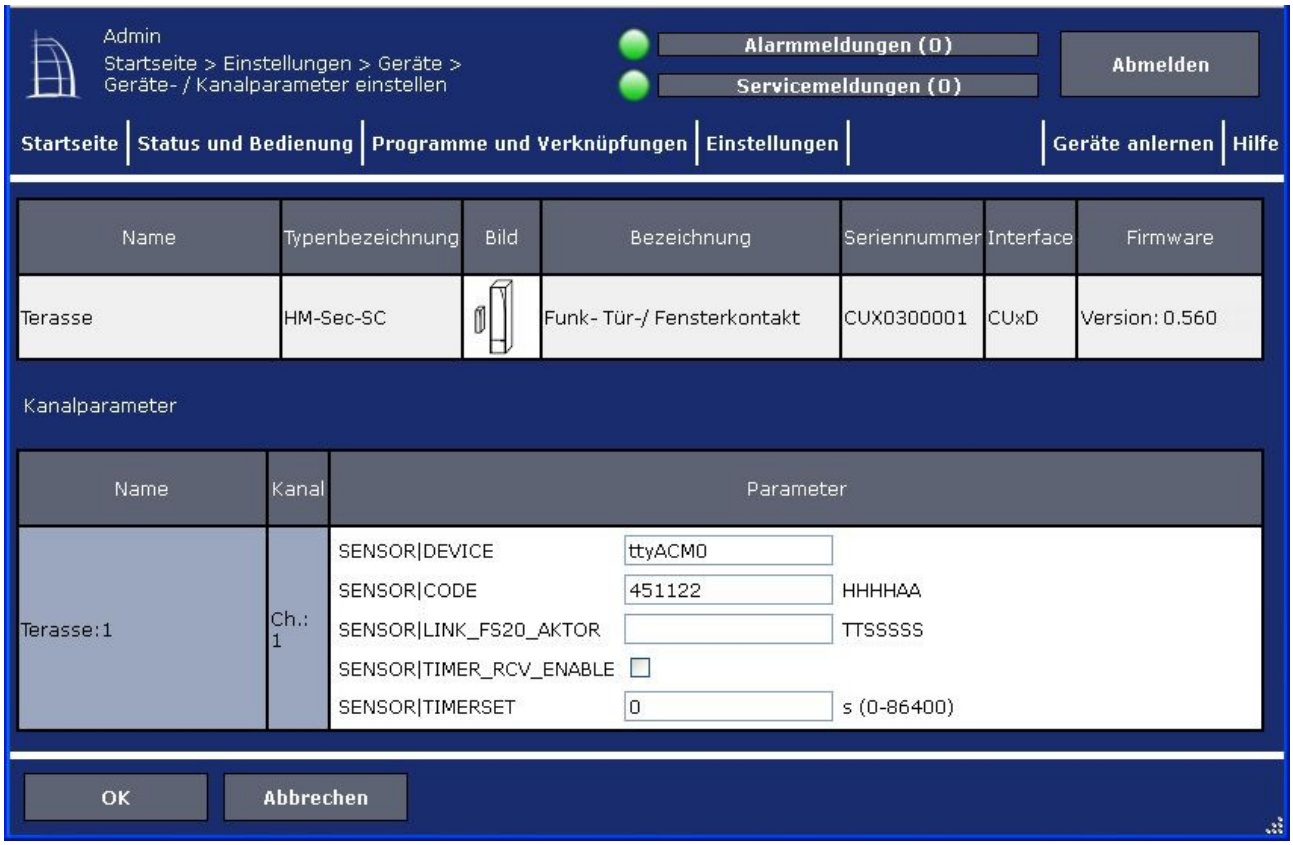
Das Löschen von CUxD-Geräten erfolgt nach dem Markieren einer Zeile auf der rechten Seite über den Button „**Gerät von CCU löschen!**“.

Die abschließende Konfiguration der neuen Geräte erfolgt über die CCU-Weboberfläche (WebUI). Hier erscheint jedes neu angelegte CUxD-Gerät sofort im „Posteingang“. Ein Anlernvorgang wie bei HM-Geräten ist nicht erforderlich. Von hier wird das Gerät dann wie ein HomeMatic-Gerät in die Logikschicht der CCU eingebunden.



An dieser Stelle kann (wie auch bei HM-Geräten) über den Button „**Einstellen**“ die abschließende Gerätekonfiguration erfolgen. Das betrifft bei den CUxD-Geräten neben optionalen Parametern auch die Konfiguration der Geräteadresse.

Nach dem Markieren der „**Fertig**“-Checkboxen für jeden Kanal wird das Gerät über den Button „**Fertig**“ endgültig übernommen und ist betriebsbereit.



Je nach Gerätetyp sind hier Konfigurationsparameter anzupassen. Bei unserem FS20-Sensor ist ein **SENSOR|CODE** (6-stellig) mit der FS20-Adresse des Gerätes notwendig. Bei den Wettersensoren wäre das eine 1-stellige Adresse von 0-7. Eine detaillierte Beschreibung erfolgt beim entsprechenden CUxD-Gerät in dieser Dokumentation.

Im Feld **SENSOR|DEVICE** kann dem Gerät, bei gleichzeitiger Verwendung mehrerer USB-Module des gleichen Typs, eines dieser Module für die Kommunikation zugewiesen werden. Bei einem leeren Feld ist automatisch das erste erkannte Gerät ausgewählt. Deshalb kann dieses Feld bei der Nutzung nur eines Moduls pro Gerätetyp auch leer gelassen werden.

Nach der Konfiguration müssen alle Kanäle des neuen Gerätes als „**Fertig**“-konfiguriert markiert werden. Nach der abschließenden Bestätigung verschwindet das Gerät aus dem Posteingang und ist nun unter „**Status und Bedienung > Geräte**“ bzw. „**Einstellungen > Geräte**“ zu finden.

Werden Datenpakete empfangen, zeigt das Control die Werte mit der letzten Aktualisierungszeit auf der CCU-Oberfläche (WebUI) an.

5 Verfügbare Geräte

Alle Geräte benötigen in der Regel die jeweilige Adresse des Funkpartners und das zu nutzende Device (Schnittstelle) für die Kommunikation. Diese variiert je nach Gerätetyp und wird bei jedem Gerät gesondert beschrieben. Kennt man die Adresse nicht, dann besteht entweder die Möglichkeit zum automatischen Anlernen oder es hilft ein Blick auf die Liste aller aktuell empfangenen Adressen der letzten Stunde ganz am Ende der CUxD-Statusseite. Diese Liste ist chronologisch sortiert, so dass die aktiven Adressen immer ganz oben stehen. Inaktive Adressen wandern an das Ende der Liste und verschwinden nach einer Stunde.

Neben dem Status [X] (im CUxD konfiguriert) und [?] (im CUxD noch nicht konfiguriert) werden für alle bekannten Geräte der Gerätename mit der am Gerät eingestellten Adresse, der im CUxD einzutragende CODE und ggf. die Empfangsfeldstärke angezeigt.

Beispiel:

```
gefundene Adressen (aktuelle zuerst 19:27:42):

    Letzte    Status  Gerät                    'CODE'
    19:27:38  [X]     EnOcean-RPS(2007590)     '001EA226'      (-42dBm)
    19:27:33  [X]     WEATHER-KS
    19:27:33  [X]     WEATHER-T/H(3)           '2'
    19:27:33  [X]     WEATHER-T/H(2)           '1'
    19:27:33  [X]     WEATHER-T/H(1)           '0'
    19:27:27  [X]     EnOcean-VLD(8837196)     '0086D84C'      (-45dBm)
    19:26:44  [X]     FHT80b(015,056)          '0F38'          (-46dBm)
    19:26:18  [X]     EM1000-EM(5)             '0205'          (-61dBm)
    19:26:14  [?]     EM1000-EM(7)             '0207'          (-42dBm)
    19:23:19  [X]     EM1000-EM(6)             '0206'          (-71dBm)
    19:15:39  [X]     EnOcean-1BS(107321)      '0001A339'      (-74dBm)
    19:15:04  [X]     FS20(3334 4142 - 1112)   'ABCD01'        (-77dBm)
    19:14:43  [X]     EnOcean-4BS(26435782)    '019360C6'      (-74dBm)
```

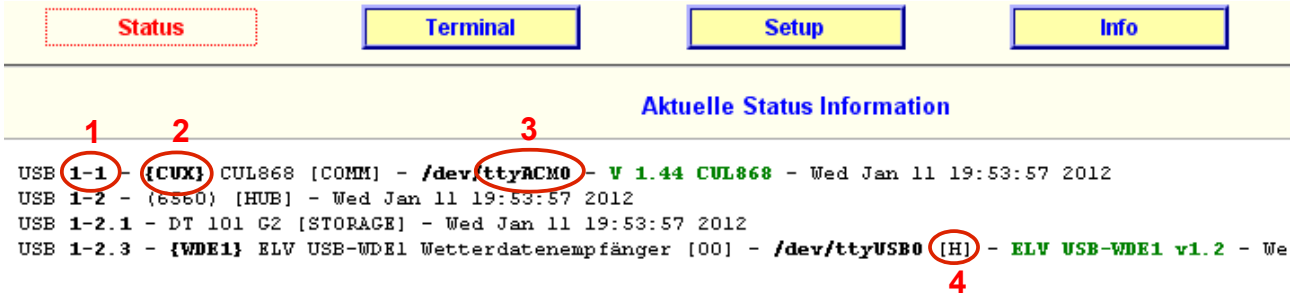
Die Geräteadresse wird in der Regel in der Gerätekonfiguration unter „**CODE**“ eingetragen. Da dieses Feld intern als String definiert ist, sollte bei der Eingabe von Hexadezimal-Adressen unbedingt auf die Groß-/Kleinschreibung geachtet werden.

Eine weitere Möglichkeit für das Heraussuchen der empfangenen Geräte/Adressen besteht mittels Terminal-Funktion der Administrations-Weboberfläche.

Alternativ können die Geräte natürlich auch nach Hersteller-Beschreibung auf neue Adressen angelernt werden. In diesem Fall wird in der Gerätekonfiguration vom CUxD-Gerät ein beliebiger „**CODE**“ eingetragen, das Gerät (*nicht die CCU!*) in den Anlernmodus versetzt und ein beliebiger Schaltbefehl zum Gerät abgesendet. Das ist aber nur zu empfehlen, falls noch keine direkten Verknüpfungen zwischen den Geräten (z.B. FS20-Fernbedienung zu FS20-Aktor) vorhanden sind.

Zusätzlich kann bei den meisten Geräten im Feld „**DEVICE**“ die CCU-Schnittstelle eintragen werden, über welche die Kommunikation mit diesem Gerät erfolgen soll. Als Wert sind hier die USB-ID **(1)** oder das TTY **(3)** erlaubt. Wird das Feld leer gelassen, so nutzt das konfigurierte Gerät automatisch das erste bzw. alle mit der CCU verbundenen Geräte dieses Typs **(2)** für die Kommunikation. Der Typ **(2)** jedes verbundenen Gerätes wird automatisch bestimmt, kann aber mit Hilfe des CUxD-Konfigurationsparameters **TTYASSIGN=** überschrieben werden. Das ist zum Beispiel beim Einsatz eines CUNO und bei einigen EnOcean ESP2- bzw. ESP3-Gateways notwendig.

Hinter dem TTY werden in eckigen Klammern **(4)** die aktiven TTY-Flags angezeigt. Dabei bedeuten: H..Hide, R..Raw und X..Hex.



Vor einer Weiterverarbeitung werden alle empfangenen Daten automatisch anhand bestimmter Merkmale auf deren Plausibilität geprüft.

Ab Version 0.563 liefern neben ESP3-Geräten auch CUX-Geräte mit jedem empfangenen Signal bei zuvor aktiviertem RSSI-Flag (CUXINITCMD=X21 bzw. TTYINIT=ttyACM0:X21) zusätzlich auch die Empfangsfeldstärke in dBm (Kanal: **0**, Datenpunkt: **RSSI_PEER**) zurück.

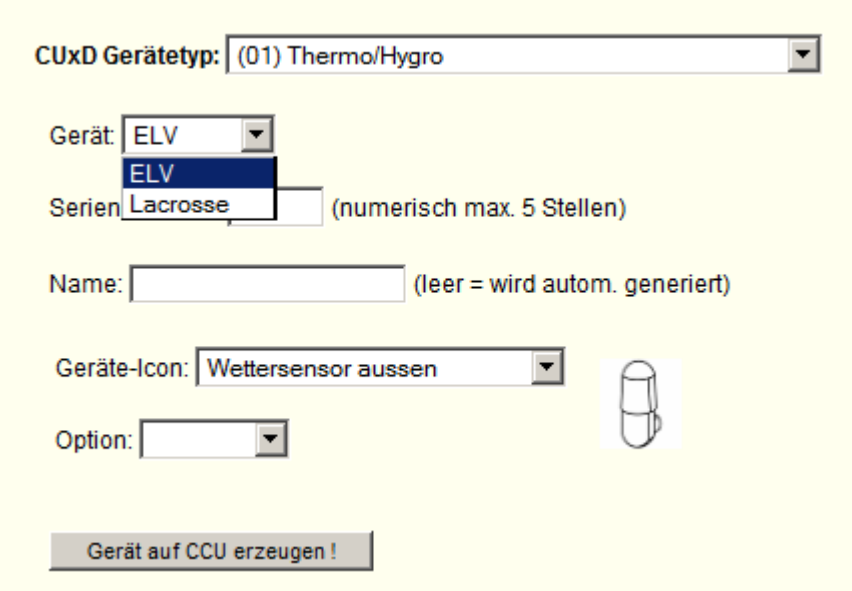
5.1 Wettersensoren {CUX}, {WDE1}

Die Datenpakete der ELV-Wettersensoren beginnen mit „**K**“ und sind in verschiedene Gerätetypen aufgeteilt. Diese Sensoren können ohne Konfigurationsänderung mit dem CUL/CUN und/oder dem USB-WDE1 empfangen werden (z.B. CUL V3, CUL V4 oder USB-WDE1). Bei gleichzeitigem Empfang von beiden Geräten werden Dubletten automatisch herausgefiltert.

Die Datenpakete der 433 MHz Lacrosse TX3 Temperatur- und Luftfeuchte-Sensoren beginnen mit „**t**“ und können ebenfalls empfangen werden, wenn ein CUL433 genutzt wird.

Zusätzlich ist eine Überwachung der zyklischen Statusmeldungen möglich. Wenn der Sensor sich nicht mindestens einmal innerhalb von 60 Minuten meldet, dann erfolgt auf der CCU eine eine **UNREACH**-Servicemeldung.

Zu diesen Sensoren zählen nicht die Wettersensoren der HMS100-Serie. Sie sind im Abschnitt 5.5 beschrieben.



5.1.1 (32) Temperatursensor

Einbindung von ELV und Lacrosse TX3 Temperatursensoren.

Konfigurationsparameter:

Kanal	Parameter		
Ch.: 1	WEATHER|CODE	3	
	WEATHER|TEMP_OFFSET	0.00	K (-50.00-50.00)
	WEATHER|STATISTIC	☑	
	WEATHER|RESET	☐	
	Zyklische Statusmeldung	☑	

CODE - Adresse des Sensors (im Sensor eingestellter Wert minus 1). Im Terminal ist es die erste Stelle nach der „K“-Kennung des Datenpaketes.

TEMP_OFFSET - Temperatur-Offset zur Kalibrierung des Sensors

STATISTIC - [x] aktivieren der Tagesstatistik

RESET - [x] Rücksetzen der Tagesstatistik (wenn **STATISTIC** aktiviert ist)

CYCLIC_INFO_MSG - [x] zyklische Statusmeldung des Sensors überwachen

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung	
Filter	Filter	Filter			
Thermo:1		Wetter	29.10.2011 10:56:03	Lufttemperatur 20.90 °C	[MISS_24H] 1
				[TEMP_MIN_24H] 20.40 °C	[TEMP_MAX_24H] 21.60 °C

Kanaltypen:

Kanaltyp	Kanalnummer
WEATHER	1

Kanaltyp WEATHER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
TEMPERATURE	float	°C	lesend	Temperatur
folgende Datenpunkte sind nur bei aktivierter Statistikfunktion verfügbar				
MISS_24H	integer		lesend	fehlende Datenpakete in den letzten 24 Stunden (maximal: 491)
TEMP_MIN_24H	float	°C	lesend	min. Temperatur (24 Stunden)
TEMP_MAX_24H	float	°C	lesend	max. Temperatur (24 Stunden)

5.1.2 (01) Temperatur/Luftfeuchte-Sensor

Einbindung von ELV und Lacrosse TX3 Temperatur/Luftfeuchte-Sensoren.

Zusätzlich zu den gemessenen Temperatur- und Luftfeuchte-Daten werden neben einer Statistik auch der Taupunkt und die absolute Luftfeuchte nach den unter der URL http://www.wettermail.de/wetter/feuchte.html beschriebenen Formeln berechnet.

Konfigurationsparameter:

Kanal	Parameter		
Ch.: 1	WEATHER|CODE	4	
	WEATHER|TEMP_OFFSET	0.00	K (-50.00-50.00)
	WEATHER|HUM_OFFSET	0.00	% (-50.00-50.00)
	WEATHER|STATISTIC	☑	
	WEATHER|RESET	☐	
	Zyklische Statusmeldung	☑	

CODE - Adresse des Sensors (im Sensor eingestellter Wert minus 1). Im Terminal ist es die erste Stelle nach der „K“-Kennung des Datenpaketes.

TEMP_OFFSET - Temperatur-Offset zur Kalibrierung des Sensors

HUM_OFFSET - Luftfeuchte-Offset zur Kalibrierung des Sensors

STATISTIC - [x] aktivieren der Tagesstatistik

RESET - [x] Rücksetzen der Tagesstatistik (wenn **STATISTIC** aktiviert ist)

CYCLIC_INFO_MSG - [x] zyklische Statusmeldung des Sensors überwachen

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung	
Filter	Filter	Filter			
Schuppen:1		Wetter	29.11.2012 21:36:54	Lufttemperatur 7.00 °C	Relative Luftfeuchte 77%
				[DEW_POINT] 3.30 °C	[ABS_HUMIDITY] 6.00 g/m³
				[TEMP_MIN_24H] 6.90 °C	[TEMP_MAX_24H] 7.00 °C
				[HUM_MIN_24H] 77.00%	[HUM_MAX_24H] 77.30%
				[MISS_24H] 0	

Kanaltypen:

Kanaltyp	Kanalnummer
WEATHER	1

Kanaltyp WEATHER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
TEMPERATURE	float	°C	lesend	Temperatur
HUMIDITY	integer	%	lesend	Relative Luftfeuchte (gerundet)
HUMIDITYF	float	%	lesend	Relative Luftfeuchte
DEW_POINT	float	°C	lesend	Taupunkt
ABS_HUMIDITY	float	g/m³	lesend	Absolute Luftfeuchte
folgende Datenpunkte sind nur bei aktivierter Statistikfunktion verfügbar				
MISS_24H	integer		lesend	fehlende Datenpakete in den letzten 24 Stunden (maximal: 491)
TEMP_MIN_24H	float	°C	lesend	min. Temperatur (24 Stunden)
TEMP_MAX_24H	float	°C	lesend	max. Temperatur (24 Stunden)
HUM_MIN_24H	float	%	lesend	min. Luftfeuchte (24 Stunden)
HUM_MAX_24H	float	%	lesend	max. Luftfeuchte (24 Stunden)

5.1.3 (31) Kombisensor KS200/KS300

Da man an diesem Sensor keine ID einstellen kann, darf er nur einmal im Empfangsbereich vorhanden sein.

Die sofortige Regenerkennung wird entweder über den KS300-Sensor direkt oder bei Änderungen des Wippenzählers ausgelöst.

Konfigurationsparameter:

Kanal	Parameter		
Ch.: 1	WEATHER|RAINFKT	295.00	ml (100.00-500.00)
	WEATHER|TEMP_OFFSET	0.00	K (-50.00-50.00)
	WEATHER|HUM_OFFSET	0.00	% (-50.00-50.00)
	WEATHER|STATISTIC	☑	
	WEATHER|RESET	☐	
	Zyklische Statusmeldung	☑	

RAINFKT - wird beim Neuanlegen mit 295 ml/m² pro Wippenschlag initialisiert.

TEMP_OFFSET - Temperatur-Offset zur Kalibrierung des Sensors

HUM_OFFSET - Luftfeuchte-Offset zur Kalibrierung des Sensors

STATISTIC - [x] aktivieren der Tagesstatistik

RESET - [x] Rücksetzen Tagesstatistik (wenn **STATISTIC** aktiviert ist)

CYCLIC_INFO_MSG - [x] zyklische Statusmeldung des Sensors überwachen

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung	
Filter	Filter	Filter			
KS300:1		Wetter	29.11.2012 21:43:01	Lufttemperatur 3.10 °C	Relative Luftfeuchte 92%
				Windgeschwindigkeit 7.70 km/h	aktuell kein Regen
				[DEW_POINT] 1.90 °C	[ABS_HUMIDITY] 5.50 g/m³
				[RAIN_CTR] 436.60 mm	[WIND_MAX_24H] 7.70 km/h
				[TEMP_MIN_24H] 3.10 °C	[TEMP_MAX_24H] 3.10 °C
				[HUM_MIN_24H] 92.00%	[HUM_MAX_24H] 92.00%
				[RAIN_CTR_24H] 0.00 mm	[MISS_24H] 0

Kanaltypen:

Kanaltyp	Kanalnummer
WEATHER	1

Kanaltyp WEATHER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
TEMPERATURE	float	°C	lesend	Temperatur
HUMIDITY	integer	%	lesend	Relative Luftfeuchte
RAINING	boolean	%	lesend	sofortige Regenerkennung
RAIN_CTR	float	mm	lesend	Regenmenge (Absolutwert)
WIND_SPEED	float	km/h	lesend	Windgeschwindigkeit
DEW_POINT	float	°C	lesend	Taupunkt
ABS_HUMIDITY	float	g/m³	lesend	Absolute Luftfeuchte
folgende Datenpunkte sind nur bei aktivierter Statistikfunktion verfügbar				
MISS_24H	integer		lesend	fehlende Datenpakete in den letzten 24 Stunden (maximal: 565)
TEMP_MIN_24H	float	°C	lesend	min. Temperatur (24 Stunden)
TEMP_MAX_24H	float	°C	lesend	max. Temperatur (24 Stunden)
HUM_MIN_24H	float	%	lesend	min. Luftfeuchte (24 Stunden)
HUM_MAX_24H	float	%	lesend	max. Luftfeuchte (24 Stunden)
WIND_MAX_24H	float	km/h	lesend	max. Windgeschwindigkeit (24 Stunden)
RAIN_CTR_24H	float	mm	lesend	Regenmenge (24 Stunden)

5.2 FS20-Geräte {CUX}

Für die Kommunikation mit FS20-Geräten ist ein CUL, CUN oder CUNO notwendig (z.B. CUL V3, CUL V4).

Protokollbeschreibung hier: http://fhz4linux.info/tiki-index.php?page=FS20%20Protocol

Befehlsaufbau:

FHHHHAABBTTRR

HHHH.....FS20-Hauscode

AA.....FS20-Adresse

BB.....FS20-Befehl

TT.....FS20-Timer (optional und abhängig vom Befehl)

RR.....RSSI-Wert vom Empfang (optional)

Die Datenpakete der FS20-Geräte beginnen im CUxD-Terminal immer mit „**F**“. Die nächsten 6 Zeichen beschreiben den Hauscode und die Adresse. Diese 6 Zeichen (A-F in Großbuchstaben!) müssen als **CODE** in der Geräte-Konfiguration eingetragen werden. Das **DEVICE**-Feld bleibt normalerweise leer und wird nur bei Verwendung mehrerer CULs genutzt.

Zum besseren Verständnis der FS20-Adressierung empfiehlt sich ein Blick in die Bedienungsanleitung der entsprechenden FS20-Geräte. Im Anhang (FAQ) ist beschrieben, wie man die ELV-FS20-Codes in hexadezimale Code-Werte für den CUxD umrechnet.

Als Faustregel für neue FS20-Geräte kann man sich folgendes merken:

- Für alle FS20-Sender/Sensoren usw. empfiehlt sich der FS20-Sensor oder FS20-Relaisaktor.
- Für alle FS20-Empfänger/Aktoren usw. empfiehlt sich der FS20-Schaltaktor oder FS20-Dimmaktor.

Ansonsten ist der gewählte CUxD-Gerätetyp abhängig vom speziellen Anwendungsfall und hat keinen direkten Bezug zur eingesetzten Hardware (Sensor, Aktor) sondern nur auf die Datenaufbereitung im CUxD.

Tabelle mit FS20-Befehlen:

Hex	FS20-Befehl	Beschreibung
00	Do.Off	Aus
01..0F	Do.DimXX%	Helligkeitsstufe einstellen (in 6.25% Schritten)
10	Do.On (100%)	Helligkeitsstufe 100%
11	Do.PreviousValue	Auf letztem Helligkeitswert einschalten
12	Do.Toggle	Wechsel zwischen Do.Off und Do.PreviousValue
13	Do.DimUp	Eine Helligkeitsstufe (6.25%) heller
14	Do.DimDown	Eine Helligkeitsstufe (6.25%) dunkler
15	Do.DimUpAndDown	Wechsel zwischen Do.DimUp und Do.DimDown
1B	Program.Reset	In Werkszustand zurücksetzen
>>> Die folgenden Befehle erfordern die Angabe einer Timer-Zeit <<<		
interne Geräte-Timer setzen		
36	Program.Time	Einschaltdauer setzen
3C	Program.DimUpTime	Zeit für Heraufdimmen setzen
3D	Program.DimDownTime	Zeit für Herabdimmen setzen
Befehle mit zeitgesteuerter Ausführung (Timer hat Vorrang vor den Geräte-Timern!)		
38	Do.Off + (timer) Do.PreviousValue	
39	Do.On + (timer) Do.Off	
3A	Do.PreviousValue + (timer) Do.Off	
3E	Do.On + (timer) Do.PreviousValue	
3F	Do.PreviousValue + (timer) Do.PreviousState	

Bei der FS20-Übertragung werden alle TIMER-Werte (Einschaltdauer usw.) abhängig von ihrer Dauer auf folgende Werte gerundet:

Intervall	Zeitschritte
0,25 s → 4 s	0,25 s
4 s → 8 s	0,5 s
8 s → 16 s	1 s
16 s → 32 s	2 s
32 s → 64 s	4 s
(1:04 min) 64 s → 128 s (2:08 min)	8 s
(2:08 min) 128 s → 256 s (4:16 min)	16 s
(4:16 min) 256 s → 512 s (8:32 min)	32 s
(8:32 min) 512 s → 1024 s (17:04 min)	64 s (1:04 min)
(0:17:04 h) 1024 s → 2048 s (0:34:08 h)	128 s (2:08 min)
(0:34:08 h) 2048 s → 4096 s (1:08:16 h)	256 s (4:16 min)
(1:08:16 h) 4096 s → 8192 s (2:16:32 h)	512 s (8:32 min)
(2:16:32 h) 8192 s → 15360 s (4:16:00 h)	1024 s (17:04 min)

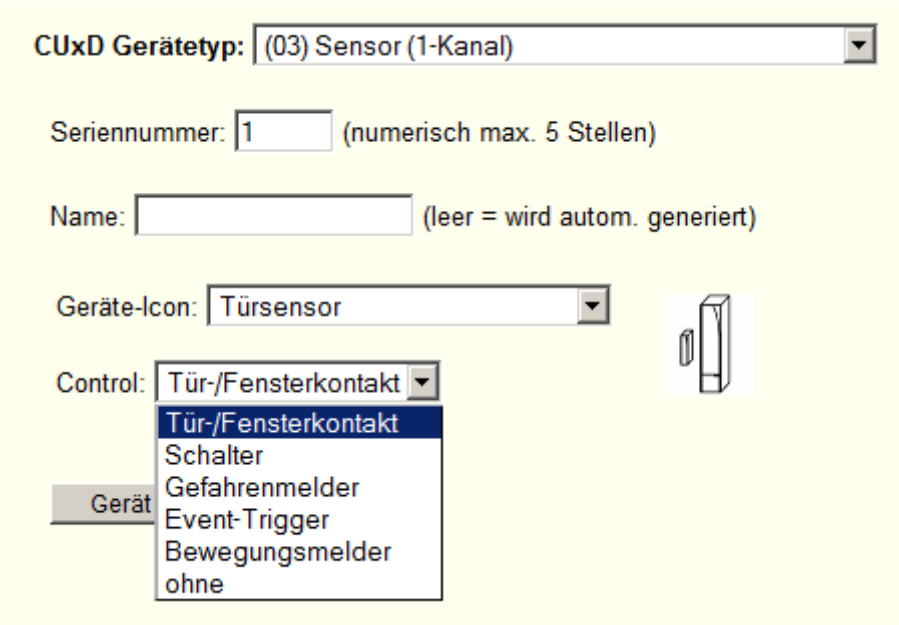
5.2.1 (03) FS20-Sensor (1-Kanal)

Dieses Gerät stellt Ein/Aus-Zustände dar, die von FS20-Sendern/Sensoren gesendet wurden. Neben dem einfachen Ein/Aus-Zustand werden auch FS20-Befehlswerte und ggf. FS20-Timerwerte empfangen und können in Programmverknüpfungen oder Scripts verarbeitet werden. Mittels einer frei definierbaren Filterzeit kann die Weiterleitung von empfangenen Befehlen zur CCU-Logikschicht für eine bestimmte Zeitdauer deaktiviert werden.

Durch das Setzen eines Befehls- und Timer-Wertes ist es sogar möglich, beliebige FS20-Befehle an die zuvor definierte Adresse des „Sensors“ zu senden. Da FS20-Sensoren in der Regel keine Empfangsmodule besitzen, macht diese Anwendung nur bei Ansteuerung von FS20-Hardware mit Empfangsmodulen (FS20-Aktoren) Sinn. Sie stellt eine Erweiterung der Gerätefunktionalität auf CCU-Seite dar. Weiterhin können FS20-Aktoren über dieses Gerät weitere Adressen (z.B. Funktionsgruppen) zugeordnet werden.

Da einige Sensoren (z.B. Bewegungsmelder bei Bewegungserkennung) nur einen Ein-Zustand senden, besteht hier die Möglichkeit, diesen automatisch (per Timer) oder manuell (per Programmverknüpfung oder WebUI) zurückzusetzen.

Beim Anlegen eines neuen Sensors kann als Darstellung (Control) je nach Anwendung „Tür-/Fensterkontakt“, „Schalter“, „Gefahrenmelder“, „Event-Trigger“, „Bewegungsmelder“ oder „ohne“ gewählt werden. Abhängig vom ausgewählten Control verhalten sich die Datenpunkte geringfügig anders.



Darstellung (Zustand „false“):

Tür-/Fensterkontakt:

Kanaltyp: SHUTTER_CONTACT
STATE: „BOOLEAN“

Schalter:

Kanaltyp: SWITCH
STATE: „BOOLEAN“

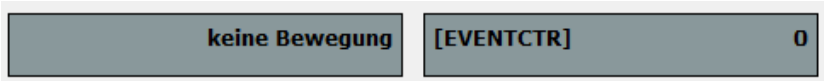
Gefahrenmelder:

Kanaltyp: DANGER
STATE: „BOOLEAN“

Event-Trigger:

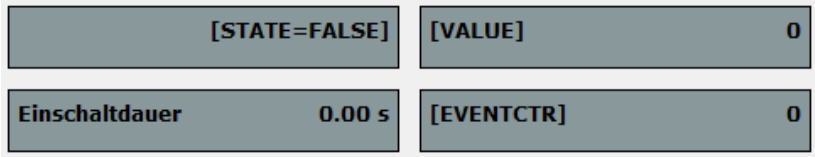
Kanaltyp: EVENT_INTERFACE
STATE: „ACTION“

Bewegungsmelder:



Kanaltyp: MOTION_DETECTOR
MOTION: „BOOLEAN“

ohne:



Kanaltyp: SENSOR

bei diesem Control ist die Funktionalität zum Senden von „VALUE“ und „ON_TIME“ deaktiviert!

Konfigurationsparameter:

Name	Kanal	Parameter		
TREPPE-LED:1	Ch.: 1	SENSOR|DEVICE		
		SENSOR|CODE	123411	HHHHAA
		SENSOR|LINK_FS20_AKTOR	0400001	TTSSSSS
		SENSOR|TIMER_RCV_ENABLE	☐	
		SENSOR|TIMER_SET	0	s (0-86400)
		SENSOR|FILTER_SET	0	s (0-86400)

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - FS20 Adresse des Gerätes

LINK_FS20_AKTOR - Weiterleiten der empfangenen Befehle an einen FS20-Aktor (Schalter, Dimmer) auf der CCU.
Es muss die Seriennummer eines CUx-Aktors ohne die ersten 3 Zeichen („CUX“) eingetragen werden. Diese Funktion kann zum einen dafür genutzt werden, um einen Aktor auf der CCU abzubilden der über verschiedene FS20-Adressen und Funktionsgruppen angesprochen wird, oder zum Verarbeiten der Rückmeldung eines mit FS20KSE erweiterten F20-Aktors. Auf der CCU wird dabei lediglich der Status des FS20-Aktors aktualisiert, so dass Programmverknüpfungen darauf reagieren können.

TIMER_RCV_ENABLE - [x] empfangene FS20-Timerbefehle werden für das Halten des empfangenen Status genutzt.

TIMER_SET - Timerwert in Sekunden (1..86400), die der Status „STATE=TRUE“ gehalten werden soll (0.. Deaktiviert)

FILTER_SET - Filterzeit in Sekunden(1..86400), die keine neuen Ereignisse mit dem selben Status weitergemeldet werden (0.. Deaktiviert)

Sind TIMER_RCV_ENABLE und TIMERSET aktiv, dann wird TIMERSET nur für FS20-Befehle ohne Timer ausgeführt. Bei Timerbefehlen wird der empfangene Timerwert verarbeitet.

Der Kanaltyp ist abhängig vom gewählten Control:

Kanaltyp	Kanalnummer
...	1

Kanaltyp ...:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
STATE	boolean, action		lesend schreibend	Zustand, auf der CCU ist eine manuelle Änderung *ohne Sendefunktion* möglich.
VALUE	integer		lesend schreibend	FS20-Befehlswert (Sendefunktionalität!)
ON_TIME	float	s	lesend schreibend	„Einschaltdauer“ - FS20-Erweiterungs-Byte (Timer) (Sendefunktionalität!)
WORKING	boolean		lesend	kennzeichnet aktive Zeitabläufe (Timer)
EVENTCTR	integer		lesend schreibend	Zähler der empfangenen Befehle (0..255)
CONTROL	integer		lesend	konfiguriertes Control-Element: 0..Tür-/Fensterkontakt 1..Schalter 2..Gefahrenmelder 3..Event-Trigger 4..ohne 5..Bewegungsmelder

Beim Ändern des Zustandes (STATE) eines Sensors (manuell per WebUI oder per Script) werden **keine FS20-Befehle gesendet!**

Der Ereigniszähler (EVENTCTR) wird bei jedem empfangenen Befehl erhöht und ist per WebUI/Script setzbar (Wertebereich: 0..255). Aus diesem Grund wird die letzte Aktualisierungszeit des Sensors auf der WebUI auch dann angezeigt, wenn sich sein Zustand nicht ändert.

VALUE von ausgewählten FS20-Befehlen:

VALUE	FS20-Befehl
0	Aus
1 bis 15	Dimmwert in 6.25% Stufen, (8 entspricht 50%)
16	Ein bzw. Dimmwert = 100%
17	Ein (alter Wert)
18	Toggle
19	Eine Helligkeitsstufe heller (+6.25%)
20	Eine Helligkeitsstufe dunkler (-6.25%)
21	Eine Helligkeitsstufe heller bzw. dunkler (Toggle)
24	Aus für Ausschaltdauer (ON_TIME)
25	Ein (100%) für Einschaltdauer (ON_TIME)
26	Ein (alter Wert) für Einschaltdauer (ON_TIME)

5.2.2 (02) FS20-Schaltaktor (1-Kanal)

Der FS20-Schaltaktor dient zum einfachen Ein-/Ausschalten eines FS20-Aktors unter Berücksichtigung der verfügbaren Geräte-Timer.

Konfigurationsparameter:

Ch.: 1	SWITCH|DEVICE		
	SWITCH|CODE	432140	HHHHAA
	SWITCH|CODE1	43214F	HHHHAA
	SWITCH|CODE2		HHHHAA
	SWITCH|CODE3		HHHHAA
	SWITCH|REPEAT	0	(0-2)
	SWITCH|INVERT	☐	
	SWITCH|DEV_TIMER	YES	
	SWITCH|DEVICE_TIMER	15360.00	s (0.00-86400.00)

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - FS20 Adresse des Gerätes (wird zum Senden von Befehlen verwendet!)

CODE1..3 - alternative FS20-Adresse (Funktionsgruppe, lokaler Master, globaler Master) zum Aktualisieren des Aktors auf der CCU.

REPEAT - Anzahl der Sendewiederholungen für schlecht erreichbare Aktoren (0 ist der Defaultwert und bedeutet KEINE Wiederholung). Bei Verwendung des TOGGLE-Befehls muss dieser Wert 0 sein!

INVERT - [x] vertauschen der Schaltzustände (z.B. bei vertauschten Anschlüssen am FS20WS1)

DEV_TIMER - YES das Gerät unterstützt interne Timer
NO das Gerät unterstützt keine internen Timer
EMU Timer werden vom CUxD emuliert (bei Geräten ohne Timer!)

DEVICE_TIMER - aktueller Wert des internen Timers eines FS20-Aktors. Bei Änderungen wird dieser Wert auch zum Gerät gesendet.

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
BÜRO-ST:1	Büro	Licht	23.10.2011 23:01:04	Aus Ein

Kanaltypen:

Kanaltyp	Kanalnummer
SWITCH	1

Kanaltyp SWITCH:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
STATE	boolean		lesend schreibend	Schaltzustand des Aktors
ON_TIME	float	s	schreibend	„Einschaltdauer“ - Ein- bzw. Ausschaltdauer des folgenden Timer-Befehls
TOGGLE	action		schreibend	ein FS20-TOGGLE-Befehl wird gesendet und damit der aktuelle Zustand gewechselt
WORKING	boolean		lesend	kennzeichnet aktive Zeitabläufe (Timer)
PROG_TIMER	float	s	lesend schreibend	Zugriff auf den internen Geräte-Timer des FS20-Aktors. Dieser Wert entspricht dem Konfigurationsparameter DEVICE_TIMER.

Bei der Verwendung von Timer-Befehlen müssen diese den Schaltvorgängen in Programmverknüpfungen oder Scripts vorangestellt werden.

Beispiel:

Es soll ein Schaltaktor mit der Adresse „123456“ und der Funktionsgruppenadresse „12345F“ auf der CCU abgebildet werden.

Dafür kann im CUxD entweder die Funktionsgruppenadresse beim Parameter **CODE1** eingetragen werden **oder** es wird ein FS20-Sensor mit einer Verknüpfung auf den Aktor angelegt (alter Weg):

- CUxD FS20-Schaltaktor mit (Basis-)Adresse „123456“
- CUxD FS20-Sensor mit Funktionsgruppenadresse „12345F“und Weiterleitung (LINK_FS20_AKTOR-Parameter) auf den zuvor definierten FS20-Schaltaktor

Der Status des Aktors auf der CCU reagiert jetzt auf beide Adressen.

Nach dem gleichen Muster können auch mehrere Aktoren mit unterschiedlichen Adressen und der gleichen Funktionsgruppe angelegt werden.

Mittels der „Senden“ Zusatzfunktionalität des **FS20-Sensors** (VALUE- und ON_TIME-Datenpunkte) können von der CCU Befehle (auch Timerbefehle) an die Funktionsgruppe gesendet werden. Dabei aktualisiert sich der Schaltzustand aller zugeordneten Aktoren gleichzeitig.

5.2.3 (04) FS20-Dimmaktor (1-Kanal)

Der FS20-Dimmaktor erlaubt das Schalten/Dimmen eines FS20-Dimmaktors in 16 Schritten unter Berücksichtigung der verfügbaren Geräte-Timer.

Konfigurationsparameter:

Kanal	Parameter		
Ch.: 1	DIMMER|DEVICE		
	DIMMER|CODE	432110	HHHHAA
	DIMMER|CODE1	4321FF	HHHHAA
	DIMMER|CODE2		HHHHAA
	DIMMER|CODE3		HHHHAA
	DIMMER|REPEAT	0	(0-2)
	DIMMER|DEV_TIMER	EMU (YES / NO / EMU)	
	DIMMER|DEVICE_TIMER		s (0.00-86400.00)
	DIMMER|DEVICE_DIM_UP		s (0.00-86400.00)
	DIMMER|DEVICE_DIM_DOWN	88.00	s (0.00-86400.00)

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - FS20 Adresse des Gerätes (wird zum Senden von Befehlen verwendet!)

CODE1..3 - alternative FS20-Adresse (Funktionsgruppe, lokaler Master, globaler Master) zum Aktualisieren des Aktors auf der CCU.

REPEAT - Anzahl der Sendewiederholungen für schlecht erreichbare Aktoren (0 ist der Defaultwert und bedeutet KEINE Wiederholung).
Bei Verwendung des TOGGLE-Befehls muss dieser Wert 0 sein!
Weiterhin durch den mehrfachen Empfang des gleichen Befehls die internen Soft-On- und Soft-Off-Timer beeinflusst.

DEV_TIMER - YES das Gerät unterstützt interne Timer
NO das Gerät unterstützt keine internen Timer (z.B. FS20LD)
EMU Timer werden vom CUxD emuliert (bei Geräten ohne Timer!)

DEVICE_TIMER - aktueller Wert des internen Timers eines FS20-Aktors.
Bei Änderungen wird dieser Wert auch zum Gerät gesendet.

DEVICE_DIM_UP - aktueller Wert des internen Soft-On-Timers eines FS20-Aktors.
Bei Änderungen wird dieser Wert auch zum Gerät gesendet.

DEVICE_DIM_DOWN - aktueller Wert des internen Soft-Off-Timers eines FS20-Aktors.
Bei Änderungen wird dieser Wert auch zum Gerät gesendet.

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
FLUR-DIM:1		Licht	27.10.2011 22:38:30	0 100% 69 % Ein Aus

Kanaltypen:

Kanaltyp	Kanalnummer
DIMMER	1

Kanaltyp DIMMER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
LEVEL	float	100%	lesend schreibend	Dimmwert des Aktors (Helligkeitslevel)
OLD_LEVEL	action		schreibend	Letzter Dimmwert des Aktors wird wiederhergestellt
RAMP_TIME	float	s	schreibend	Dimmzeit für das Dimmen zum angegebenen Helligkeitslevel
ON_TIME	float	s	schreibend	„Einschaltdauer“ - Ein- bzw. Ausschaltdauer des folgenden Befehls (Helligkeitslevel)
TOGGLE	action		schreibend	ein FS20-TOGGLE-Befehl wird gesendet und damit der aktuelle Zustand (Level) gewechselt
WORKING	boolean		lesend	kennzeichnet aktive Zeitabläufe (Timer)
PROG_TIMER	float	s	lesend schreibend	Zugriff auf den internen Geräte-Timer des FS20-Aktors. Dieser Wert entspricht dem Konfigurations-parameter DEVICE_TIMER.
PROG_DIM_UP	float	s	lesend schreibend	Zugriff auf den internen Soft-On-Timer des FS20-Aktors. Dieser Wert entspricht dem Konfigurations-parameter DEVICE_DIM_UP.
PROG_DIM_DOWN	float	s	lesend schreibend	Zugriff auf den internen Soft-Off-Timer des FS20-Aktors. Dieser Wert entspricht dem Konfigurations-parameter DEVICE_DIM_DOWN.

Bei der Verwendung von Timer-Befehlen müssen diese den Schaltvorgängen (LEVEL-Änderung) in Programmverknüpfungen oder Scripts vorangestellt werden.

5.2.4 (05) FS20-Relais (1-Kanal)

Beim Relais werden FS20-Befehle an einen beliebigen anderen Aktor weitergeleitet und dort ausgeführt. Dazu muss in der Gerätekonfiguration zusätzlich zur FS20-Adresse die Seriennummer eines konfigurierten Aktor-Kanals eingetragen werden. Es können auch CUxD-Aktoren verwendet werden.

Der verbundene Aktor wird synchron zum empfangenen FS20-Befehl gesteuert.

Der Zustand des FS20-Relais steuert nicht den verbundenen Aktor, sondern dient zum Aktivieren/Deaktivieren der Befehlsweiterleitung an diesen. Damit der FS20-Relais-Aktor funktioniert, muss er zuvor manuell, per Programmverknüpfung oder per Script aktiviert (**eingeschaltet**) werden. Somit sind über einfache Programmverknüpfungen z.B. auch zeitabhängige Weiterleitungen möglich.

Konfigurationsparameter:

Kanal	Parameter		
Ch.: 1	RELAIS|DEVICE		
	RELAIS|CODE	12ABCD	HHHHAA
	RELAIS|CODE1		HHHHAA
	RELAIS|CODE2		HHHHAA
	RELAIS|CODE3		HHHHAA
	RELAIS|HMSERIAL	HEQ0271443:1	SERIAL:X
	RELAIS|HSS_TYPE	DIMMER	
	RELAIS|TRANS1	17 18	old new[:timer]
	RELAIS|TRANS2		old new[:timer]
	RELAIS|TRANS3		old new[:timer]
	RELAIS|TRANS4		old new[:timer]
	RELAIS|TRANS5		old new[:timer]

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - FS20 Adresse des Gerätes

CODE1..3 - alternative FS20-Adressen

HMSERIAL - HomeMatic Seriennummer des Gerätes mit Kanalnummer

HSS_TYPE - Anzeige des gefundenen HomeMatic-Gerätetyps (SWITCH/DIMMER/BLIND/...)

TRANS1..5 - Zeichenkette mit Befehlskonvertierung
Beispiele:
„17 8“ - 17. (Do.On) wird zu 8. (Dim50%) umgesetzt
„17 18“ - 17. (Do.On) wird zu 18. (Do.Toggle) umgesetzt
„17 26:11.0“ - 17. (Do.On) wird zu 26. (Ein für Zeitdauer: 11s)

Kanaltypen:

Kanaltyp	Kanalnummer
RELAIS	1

Kanaltyp RELAIS:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
STATE	boolean		lesend schreibend	Zustand des Relais. Das Relais kann aktiviert (Ein) und deaktiviert (Aus) werden. So sind über Programmverknüpfungen zeitlich begrenzte Befehlsweiterleitungen möglich.
VALUE	integer		lesend	FS20-Befehlswert
TRANS	string		lesend	ausgeführte Befehlskonvertierung
ON_TIME	float	s	lesend	empfangene Einschaltdauer
RAMP_TIME	float	s	lesend	empfangene Dimmzeit

5.3 Energie-Sensoren {CUX}

Für den Empfang der Daten ist ein CUL, CUN oder CUNO notwendig (z.B. CUL V3, CUL V4).

5.3.1 (06) EM1000 Energie-Sensoren

Protokollbeschreibung hier: http://fhz4linux.info/tiki-index.php?page=EM+Protocol

Paketaufbau (hexadezimal kodiert):

ETTAACCSSSSLLLLMMMMRR

TT.....Typ

AA.....Adresse

CC.....8bit Zähler für Datentelegramme

SSSS.....16bit aufsummierter Verbrauch seit dem Einschalten

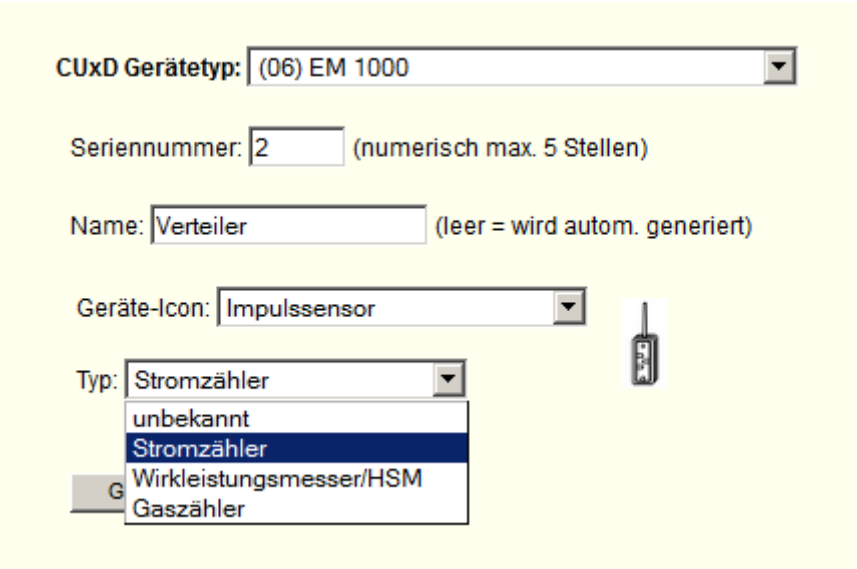
LLLL.....16bit Durchschnittsverbrauch in den letzten 5 Minuten oder 0

MMMM.....16bit Spitzenverbrauch der letzten 5 Minuten oder 0

RR.....RSSI-Wert vom Empfang (optional)

Die CUx-Datenpakete der ELV EM-Energiesensoren beginnen immer mit „**E**“. Die folgenden 4 Zeichen beinhalten den Typ und die Adresse des Gerätes. Jeder Sensor sendet seine Messwerte im 5 Minuten Intervall.

Wird beim Neuanlegen der genutzte Zählertyp definiert, dann werden auf der WebUI die richtigen Einheiten (Strom: W, Wh, kW, kWh und Gas: m³, m³/h) angezeigt und die Defaultwerte entsprechend gesetzt.



Defaultwerte nach dem Anlegen neuer Sensoren:

Wechselstromzähler (**EM-WZ**)	CODE = 0101,	TURNPUNIT = 75
Wirkleistungsmesser (**EM-EM, EM-HSM**)	CODE = 0205,	TURNPUNIT = 1
Gaszähler (**EM-GZ**)	CODE = 0309,	TURNPUNIT = 100

Achtung:

Einige EM1000-HSM Sensoren senden unter bestimmten Umständen im Leerlauf fehlerhafte Daten. Für den Fall, dass der Verbrauch [**SUM**] bei EM1000-EM und EM1000-HSM Sensoren zwischen 2 aufeinanderfolgenden Messungen über der Durchschnittsleistung [**MEAN5MINUTES**] bzw. der Maximalleistung [**MAX5MINUTES**] liegt, ist im CUxD bereits ein Workaround implementiert, der den Verbrauchswert bei der Aufsummierung [**METER**] korrigiert.

Konfigurationsparameter:

Name	Kanal	Parameter
EM-KS Büro:1	Ch.: 1	SENSOR|DEVICE SENSOR|CODE 0205 SENSOR|TURNPUNIT 1 SENSOR|RESET ☐ SENSOR|SETMETER 284736.00 Wh (0.00-9999999.98) Zyklische Statusmeldung ☑

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - Typ + Adresse des EM-Gerätes. Aufgrund von Firmwarebeschränkungen in den Sensoren sind nur 4 Zähler pro Typ möglich. Das ergibt folgende Codes:

Zähler#	**1**	**2**	**3**	**4**	
EM-WZ	0101	0102	0103	0104	*Wechselstromzähler (optisch)*
EM-EM/HSM	0205	0206	0207	0208	*Zwischenstecker/Hutschienenmodul*
EM-GZ	0309	030A	030B	030C	*Gaszähler (optisch)*

TURNPUNIT - Umdrehungen (Impulse) pro kWh bzw. pro m³. Bei Stromzählern ist hier der aufgedruckte Wert einzutragen (z.B. 75 oder 120 oder ...)

RESET - [x] Rücksetzen der Statistikdaten

SETMETER - setzen des absoluten Zählerstandes des Gas-/Stromzählers (METER)

CYCLIC_INFO_MSG - [x] Zyklische Statusmeldung des Sensors überwachen. Wenn der Sensor sich nicht mindestens einmal innerhalb von 60 Minuten meldet, erfolgt eine **UNREACH**-Servicemeldung auf der CCU.

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
EM-KS Büro:1	Büro	Energiemanagement	27.10.2011 23:01:26	[COUNTER] 100 / [SUM] 4856.00 Wh / [MEAN5MINUTES] 70.00 W / [MAX5MINUTES] 640.00 W / [SUM_1H] 106.00 Wh / [MAX_1H] 640.00 W / [SUM_24H] 656.00 Wh / [MAX_24H] 860.00 W / [MISS_24H] 0 / [METER] 70392.00 Wh

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
COUNTER	integer	lesend	Nummer des empfangenen Datensatzes (0..255)
SUM	float	lesend	kumulierter Zählerwert seit Gerätestart des Sensors. Es handelt sich um einen 16-bit Zähler, der regelmäßig überläuft und dann wieder bei 0 beginnt.
MEAN5MINUTES	float	lesend	Durchschnittsleistung/Verbrauch der letzten 5 Minuten
MAX5MINUTES	float	lesend	Maximalleistung/Verbrauch in den letzten 5 Minuten
SUM_1H	float	lesend	Verbrauch der letzten Stunde
MAX_1H	float	lesend	Maximalleistung/Verbrauch in der letzten Stunde
SUM_24H	float	lesend	Verbrauch der letzten 24 Stunden (Aktualisierung nach Wechsel der akt. Stunde)
MAX_24H	float	lesend	Maximalleistung/Verbrauch der letzten 24 Stunden (Aktualisierung nach Wechsel der akt. Stunde)
MISS_24H	integer	lesend	fehlende Datenpakete in den letzten 24 Stunden (maximal: 288, Aktualisierung nach Stundenwechsel)
METER	float	lesend	absoluter Zählerstand des Gas-/Stromzählers (kann über die Gerätekonfiguration gesetzt werden)
CONTROL	integer	lesend	ausgewählter Zählertyp: 0..unbekannt 1..Wechselstromzähler 2..Wirkleistungsmesser 3..Gaszähler

5.3.2 (27) ESA1000 / ESA2000 Energie-Sensoren

Paketaufbau (hexadezimal kodiert):

SCCDDDDTTTTSSSSSSSSAAAALLLLLLTTTTRR

CC.....7bit Zähler für Datentelegramme

DDDD.....16bit Adresse

TTTT.....16bit Typ

SSSSSSSS..32bit aufsummierte Impulse seit dem Einschalten

AAAA.....16bit aktuelle Impulse im letzten Messintervall

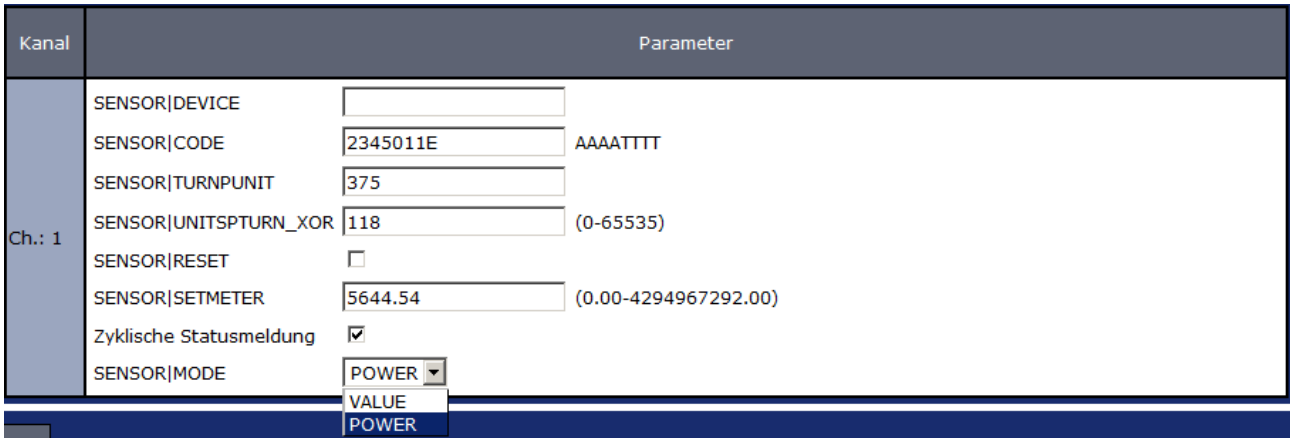
LLLLLL.....24bit Zeitstempel in 10s?

TTTT.....16bit konfigurierte Impulse / kWh vom Sensor

RR.....RSSI-Wert vom Empfang (optional)

Die CUx-Datenpakete der ELV ESA1000/2000-Energiesensoren beginnen mit „**S**“. Die folgenden 2 Zeichen sind ein Zähler und die nächsten 8 Zeichen beinhalten die Adresse und den Typ des Gerätes. Jeder Sensor sendet seine Messwerte im Abstand von 2 bis 3 Minuten. Der Funk-Empfang ist leider nicht sehr zuverlässig, so das regelmäßig Daten-pakete verloren gehen.

Konfigurationsparameter:



DEVICE	- CUx Geräte-ID oder TTY oder leer
CODE	- Adresse + Typ des ESA-Gerätes.
TURNPUNIT	- Umdrehungen (Impulse) pro Einheitswert. Beim Gaszähler ist hier 0 einzutragen, da der Wert bereits am Sensor errechnet wird.
UNITSPTURN_XOR	- XOR-Korrekturwert für fehlerhafte **UNITSPTURN**-Werte. In dieses Feld ist das Ergebnis der XOR-Verknüpfung zwischen dem am Sensor eingestellten Wert und dem empfangenen unkorrigierten **UNITSPTURN**-Wert einzutragen. Der unkorrigierte **UNITSPTURN**-Wert wird ausgeben, wenn dieser Parameter auf 0 gesetzt ist!
RESET	- [x] Rücksetzen der Statistikdaten
SETMETER	- setzen des absoluten Zählerstandes des Zählers (METER)
CYCLIC_INFO_MSG	- [x] Zyklische Statusmeldung des Sensors überwachen. Wenn der Sensor sich nicht mindestens einmal innerhalb von 60 Minuten meldet, erfolgt eine **UNREACH**-Servicemeldung auf der CCU.
MODE	- Verarbeitung der Empfangsdaten: **VALUE** Empfangsdaten nur ausgeben **POWER** Empfangsdaten als Stromzähler aufbereiten

VALUE-Mode:

ESA1000:1		Energiemanagement	04.01.2013 13:19:39	[COUNTER]	27	[SUM]	81.74
				[LAST_VALUE]	0.00	[LAST_TICKS]	85672.00
				[UNITSPTURN]	375.00	[METER]	5644.54
				[SUM_1H]	0.03	[SUM_24H]	0.03

POWER-Mode:

ESA1000:1		Energiemanagement	04.01.2013 13:26:42	[COUNTER]	0	[SUM]	0.00 kWh
				[LAST_VALUE]	0.00 W	[METER]	5644.54 kWh
				[SUM_1H]	0.00 kWh	[SUM_24H]	0.00 kWh

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
COUNTER	integer	lesend	Nummer des aktuellen Datensatzes (0..127)
SUM	float	lesend	kumulierter Zählerwert seit Gerätestart des Sensors. Dieser Zähler läuft regelmäßig über und beginnt dann wieder bei 0.
LAST_VALUE	float	lesend	Durchschnittsverbrauch im letzten Messintervall
LAST_TICKS	float	lesend	Letztes Messintervall in Sekunden mal 10 (GZ)
UNITSPTURN	float	lesend	Am Sensor eingestellter Umrechnungsfaktor (kann mittels **UNITSPTURN_XOR**-Parameter korrigiert werden) Beim Gaszähler-Sensor ist der am Sensor eingestellte Wert mit 1000 zu multiplizieren.
METER	float	lesend	absoluter Zählerstand des Zählers (kann über die Gerätekonfiguration gesetzt werden)
SUM_1H	float	lesend	Energieverbrauch der letzten vollen Stunde
SUM_24H	float	lesend	Energieverbrauch der letzten 24 Stunden (Aktualisierung jede Stunde)
CONTROL	integer	lesend	ausgewählter Mode: 0..VALUE 1..POWER

5.4 FHT-Heizungssteuerung {CUX}

Für die Kommunikation ist ein CUL, CUN oder CUNO notwendig (z.B. CUL V3, CUL V4).

Die CUx-Datenpakete der FHT-Geräte beginnen immer mit „**T**“. Die folgenden 4 Zeichen beinhalten den Hauscode des Gerätes.

5.4.1 (07) FHT8v Ventilantrieb

Dieses Gerät empfängt die vom FHT80b gesendeten Einstellungen an die FHT8v-Ventilantriebe und dient als reine Statusanzeige für jeden vorhandenen Ventilantrieb. Nach den 4 Zeichen (CCCC) für den Hauscode folgen 2 Zeichen (AA) mit der Nummer des Ventilantriebes.

Der 4-stellige Hauscode (CCCC) entspricht dem eingestellten Hauscode des steuernden FHT80b Wandthermostaten. Die folgende 2-stellige Nummer (AA) entspricht der Nummer des konfigurierten Ventilantriebes von „01“ bis „08“.

Aufgrund von zeitlichen Beschränkungen im FHT-Protokoll, werden vom FHT80b normalerweise immer alle konfigurierten Ventilantriebe gleichzeitig über die Adresse „00“ angesprochen. Dieser Sonderfall wird vom CUxD erkannt und automatisch an alle konfigurierten Ventilantriebe mit dem gleichen 4-stelligen Hauscode (CCCC) weitergeleitet.

Konfigurationsparameter:

Name	Kanal	Parameter
FHT-V1:1	Ch.: 1	CLIMATECONTROL_VENT_DRIVE|DEVICE CLIMATECONTROL_VENT_DRIVE|CODE 0F3301 CCCCAA

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - Hauscode + Nummer vom Ventilantrieb (folgende 2 Zeichen)

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
FHT-V1:1		Heizung Klima	27.10.2011 21:55:12	Ventilantrieb Status 38% / Ventilantrieb Offsetstellung 0.00%

Kanaltypen:

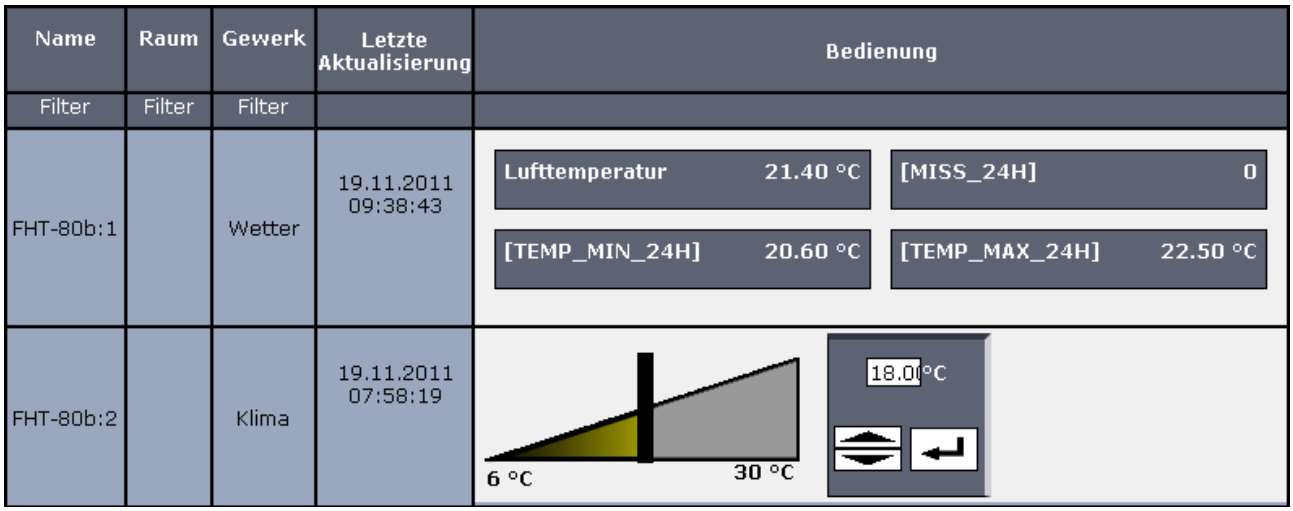
Kanaltyp	Kanalnummer
CLIMATECONTROL_VENT_DRIVE	1

Kanaltyp CLIMATECONTROL_VENT_DRIVE:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
VALVE_STATE	integer	%	lesend	Ventilantrieb Status
VALVE_OFFSET_VALUE	float	%	lesend	Ventilantrieb Offsetstellung

5.4.2 (08) FHT80b Wandthermostat

Der FHT80b-Wandthermostat wird über 2 Kanäle abgebildet. Ein Kanal dient zum Auslesen der gemessene Temperatur und einer zum Einstellen der Parameter.



Die FHT-Kommunikation ist sehr träge, da die Datenpakete nur ca. alle 116 Sekunden übertragen werden können. Auch kann es vorkommen, dass Daten (abhängig von der Empfangssituation) manchmal nicht komplett, sondern in mehreren Zyklen übertragen werden.

Um die FHT-Kommunikation zu aktivieren, muss auf dem CUx-Gerät ein FHT-Hauscode gesetzt werden. Dieser Hauscode besteht aus 4 hexadezimalen Ziffern. Die ersten beiden Ziffern stellen HC1 und die 3. und 4. Ziffer HC2 dar.

HC1 und HC2 müssen jeweils im Bereich von 00h-63h liegen. Werden HC1 und HC2 auf 00h gesetzt, dann ist die FHT-Kommunikation deaktiviert. (Im FHT80b-Menü werden Code1 und Code2 dezimal von 0-99 angezeigt, müssen also umgerechnet werden).

Bei der Steuerung von FHT80b-Wandthermostaten müssen alle HC1 (auf FHT80b und CUL/CUN) identisch und alle HC2 verschieden sein.

<u>Beispiel für zwei FHT80b:</u>

FHT80b HC1=**12** HC2=**11** → dezimal: Code1=**18**, Code2=**17**
FHT80b HC1=**12** HC2=**12** → dezimal: Code1=**18**, Code2=**18**
CUL HC1=**12** HC2=**34**

Auf dem CUL/CUN kann der eigene Hauscode über das CUxD-Terminal mit dem Befehl „T01“ ausgelesen und „T01XXXX“ gesetzt werden. Zum Beispiel setzt der Befehl „T011234“ den Hauscode auf HC1=12 und HC2=34 (alle Werte hexadezimal!).

Am sichersten ist es, wenn man den Hauscode im **TTYINIT**-Parameter speichert (TTYINIT=ttyACM0:X21\nT011234). Damit werden bei einem CUxD-Neustart auch gleich alle FHT-Befehlspuffer im CUL/CUN gelöscht.

Die FHT80b-Hauscodes werden im jeweiligen CUxD-Gerät als Parameter eingetragen.

CUL Pairing mit FHT80b-Wandthermostaten

Zum Pairing muss mindestens ein Ventilantrieb am Wandthermostaten konfiguriert sein. Danach kann der FHT80b in den „Anlernmodus“ gesetzt werden (**Menü „CEnt“ auf „nA“ setzen**).

Jetzt sollte sofort ein Befehl vom CUL zum Thermostaten gesendet werden (z.B. aus dem CUxD-Terminal oder RESYNC [x] aus der Gerätekonfiguration). Wenn ca. 2 Minuten später im FHT80b-Menü **„CEnt“ auf „On“** steht, war das Pairing erfolgreich und es können Daten zum CUL/CUN übertragen werden (siehe auch http://fhemwiki.de/index.php/FHT80b). Bei mehreren Wandthermostaten sollte immer nur **ein FHT80b nach dem anderen** angelernt werden.

Aufgrund der trägen Kommunikation kann es unter Umständen eine Weile dauern, bis nach dem Anlernen alle Daten übertragen wurden.

Treten bei der Nutzung von FHT80b-Wandthermostaten in Zusammenhang mit dem CUL/CUN-Modul verstärkt Kommunikationsprobleme und LOVF-Meldungen auf, so kann ein Ablernen (Menü „Cent“ auf „Off“) aller Wandthermostate und Deaktivieren (T010000) der FHT80-Kommunikation für einige Stunden mit einem darauffolgenden erneuten Anlernen der Wandthermostate helfen.

Batteriewarnungen der Wandthermostate werden als LOWBAT-Servicemeldung auf der CCU dargestellt und löschen sich automatisch nach einem Batteriewechsel.

Bei empfangenen SYNC-Meldungen vom Thermostaten wird der FHT80-Puffer im CUx gelöscht, um LOVF-Meldungen aufgrund erfolgloser Sendeversuche zu verhindern.

Bei aktiviertem CLOCKSYNC [x] wird nach dem Neu-Einrichten, einem Batteriewechsel oder einer Synchronisation mit den Ventilantrieben die aktuelle Uhrzeit innerhalb der nächsten 30 Minuten zum FHT80b übertragen. Danach erfolgt der Uhrzeitabgleich ca. einmal täglich. Um die Auslastung des 868MHz Frequenzbandes etwas zu verteilen (*LOVF-Meldungen*), besteht zusätzlich zwischen der Synchronisation von unterschiedlichen FHT80b-Thermostaten ein Mindestabstand von 60 Minuten.

Konfigurationsparameter:

Parameter		
DEVICE		
CODE	0F33	CCCC
RESYNC	☐	
CLOCKSYNC	☑	
STATISTIC	☑	
RESET	☐	
Zyklische Statusmeldung	☑	

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - FHT80b-Hauscode

RESYNC - [x] pairing vom FHT80b mit dem CUL/CUN. Es wird ein Befehl zum FHT80b gesendet. Zuvor muss der FHT80b in den Anlernmodus gesetzt werden. (Menü „CEnt“ auf „nA“ setzen) Nach erfolgreichem Pairing steht das Menü „CEnt“ auf „On“.

CLOCKSYNC - [x] täglicher Uhrzeitabgleich von der CCU

STATISTIC - [x] aktivieren der Statistik-Option

RESET - [x] Rücksetzen aller Statistikdaten (wenn STATISTIC aktiviert ist)

CYCLIC_INFO_MSG - [x] Zyklische Statusmeldung des Sensors überwachen. Wenn der Sensor sich nicht mindestens einmal innerhalb von 2 Stunden meldet, erfolgt eine **UNREACH**-Servicemeldung auf der CCU.

Kanaltypen:

Kanaltyp	Kanalnummer
WEATHER	1
CLIMATECONTROL_REGULATOR	2

Kanaltyp WEATHER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
TEMPERATURE	float	°C	lesend	Temperatur
folgende Datenpunkte sind nur bei aktivierter Statistikfunktion verfügbar				
MISS_24H	integer		lesend	fehlende Temperatur-Datenpakete in den letzten 24 Stunden (maximal: 96)
TEMP_MIN_24H	float	°C	lesend	min. Temperatur (24 Stunden)
TEMP_MAX_24H	float	°C	lesend	max. Temperatur (24 Stunden)

Kanaltyp CLIMATECONTROL_REGULATOR:

DP-Name	Typ	Einh.	Zugriff	Beschreibung
SETPOINT	float	°C	lesend schreibend	Solltemperatur
STATE	boolean		schreibend	Ventil öffnen, Ventil schließen
MODE_TEMPERATUR_REGULATOR	integer		lesend schreibend	Temperaturreglermodus 0.. automatisch 1.. manuell 2.. Party (nur lesend!)
TEMPERATUR_COMFORT_VALUE	float	°C	lesend schreibend	Programmierte Komfort-temperatur am FHT80b
TEMPERATUR_LOWERING_VALUE	float	°C	lesend schreibend	Programmierte Absenk-temperatur am FHT80b
TEMPERATUR_WINDOW_OPEN_VALUE	float	°C	lesend schreibend	Programmierte Fenster-Auf-Temperatur am FHT80b
PARTY_END_TIME	string		lesend schreibend	Party/ Urlaub-Endzeit am FHT80b. Dieses Feld kann auf „HH:MM“ oder „DD.MM“ oder „+MIN“ gesetzt werden.

Per Script können Temperaturwerte von 0 bis 99,5°C (5,5°C = OFF und 30,5°C = ON) gesetzt werden.

Party/Urlaubs-Endzeit (PARTY_END_TIME)

Durch das Setzten dieses Datenpunktes kann auf dem FHT80b-Wantthermostaten der Party- oder Urlaubs-Mode aktiviert werden.

„HH:MM“ - Party-Ende

Dieser Wert wird auf volle 10 Minuten gerundet und legt das Ende des Party-Modus fest. Bereits vergangene Uhrzeiten werden auf den nächsten Tag gesetzt. Werte größer als „23:50“ werden immer automatisch auf den nächsten Tag umgerechnet und gesetzt.

Programmbeispiel:

„DD.MM“ - Urlaubs-Ende

Datum des Tages, an dem um 0:00 Uhr der Urlaubs-Modus verlassen werden soll.

Programmbeispiel:

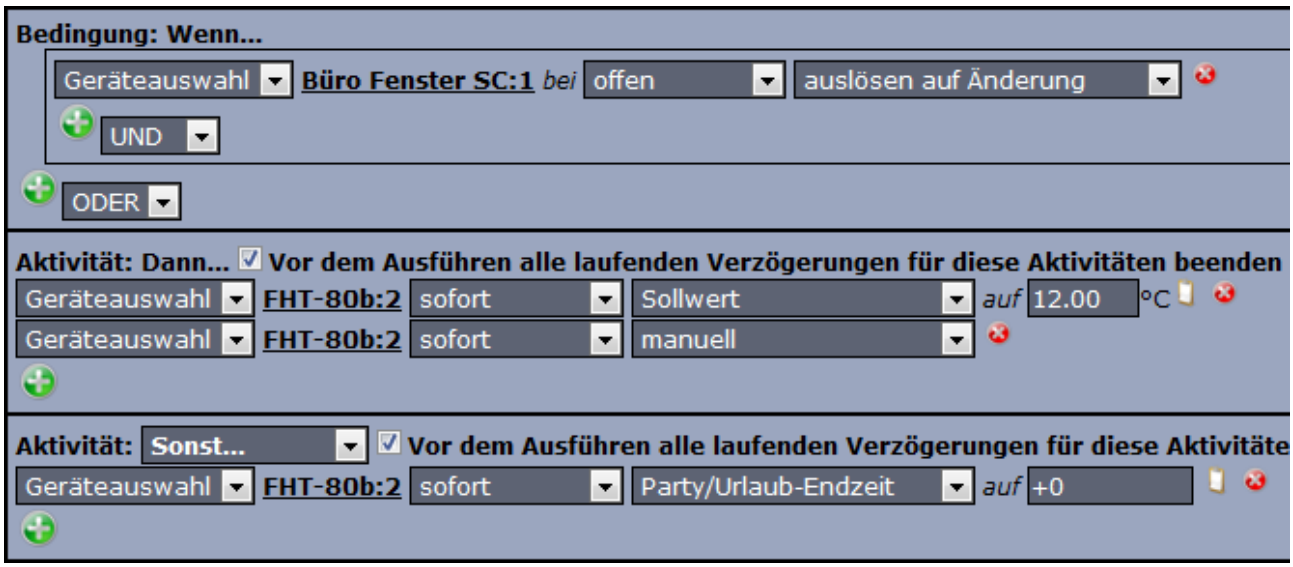
„+MIN“ - Party-Ende als Offset in Minuten

Dieser Wert wird zur aktuellen Zeit addiert und legt dann auf 10 Minuten gerundet das Ende des Party-Modus fest. Ist der übergebene Wert größer als der Maximalwert (folgender Tag 23:50 Uhr), dann wird er automatisch auf den Maximalwert gesetzt. Ist der übergebene Wert 0, dann wird er auf den nächstmöglichen Wert gesetzt.

Programmbeispiel:

Mit diesem Parameter ist es auch möglich, die Funktion eines Tür/Fensterkontaktes mit der CCU zu emulieren.

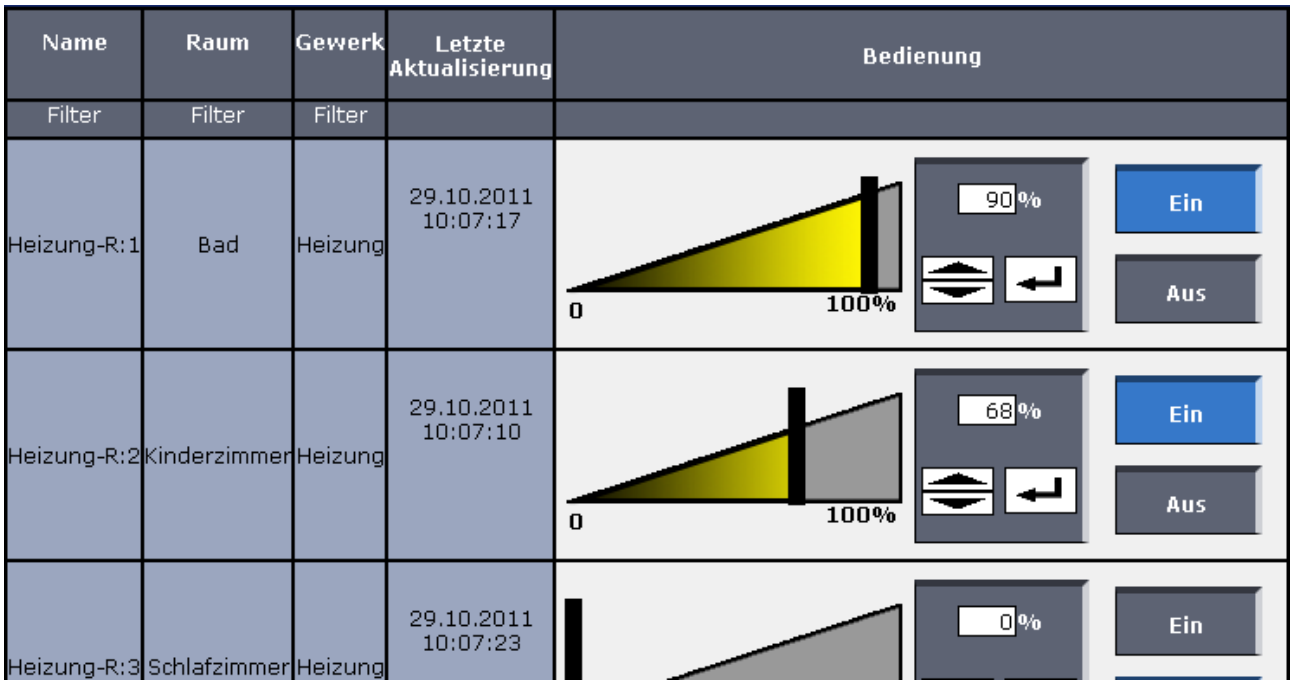
Programmbeispiel für TFK-Emulation:



5.4.3 (09) Multiventil-Steuerung (8 Räume mit FHT8v-Stellantrieben)

Die Multiventilsteuerung erlaubt die direkte Ansteuerung von FHT8v-Stellantrieben in maximal 8 verschiedenen Räumen ohne FHT80b-Wandthermostat. Für jeden angeschlossenen CUL kann nur ein solches Gerät angelegt werden, da die culfw maximal 8 verschiedene FHT8v-Empfangsadressen ansteuern kann. Wenn man jedem Ventilantrieb im selben Raum die gleiche Adresse gibt, dann kann man auf diese Weise die Heizungsventile in 8 verschiedenen Räume stellen.

Die Multiventilsteuerung wird auf der CCU durch ein 8-Kanal Gerät mit den entsprechenden Reglern dargestellt:



Damit die direkte Kommunikation des CUL/CUN mit den Ventilantrieben funktioniert, müssen der konfigurierte Hauscode und der auf dem CUL/CUN gesetzte Hauscode gleich sein.

Beim Einsatz der Multiventilsteuerung und der CUx-Thermostate muss die erste Zahl des Hauscodes HC1 (1. und 2. Ziffer) für die Multiventilsteuerung und die zu steuernden FHT80b-Thermostate identisch sein. Die zweite Zahl des Hauscodes (HC2) (3. und 4. Ziffer) muss für alle Geräte verschieden sein.

Zum Beispiel wäre folgendes möglich:

Hauscode der Multiventilsteuerung: 1234

Hauscodes der FHT80b-Wandthermostate: 1232, 1233, 1235, 1236, ...

Der CUL-Hauscode kann im Terminal mit dem Befehl „T01XXXX“ gesetzt und mit dem Befehl „T01“ ausgelesen und kontrolliert werden. Mit „T010000“ wird die FHT-Funktionalität am CUL/CUN ausgeschaltet.

Es empfiehlt sich, den Hauscode im **TTYINIT**-Parameter zu speichern (TTYINIT= ttyACM0:X21\nT011234). Damit werden bei jedem CUxD-Neustart auch gleich alle FHT-Befehlspuffer gelöscht.

Um die Kommunikation zu beschleunigen, bekommt bei der Multiventilsteuerung jedes Ventil einen eigenen Hauscode, indem die erste Zahl des Hauscodes HC1 für jedes folgende Ventil in der Liste erhöht wird.

Für unser Beispiel würde das bedeuten:

1. Ventil = 1234
2. Ventil = 1334
3. Ventil = 1434
4. Ventil = 1534

Sollten die Ventilantriebe die Synchronisation verlieren (z.B. beim Neustart der CCU - ein Neustart des CUxD ist ohne Bedeutung), so dauert es ein paar Stunden und sie sind automatisch wieder synchron.

Wenn es nach einem Verlust der Synchronisation schneller gehen soll: Die Taste am Ventilantrieb so lange drücken, bis ein akustisches Signal ertönt und AC im Display erscheint - etwas warten - wieder lange drücken bis ein Signal ertönt und das Antennensymbol blinkt (das Ventil ist jetzt im Empfangsmodus und synchronisiert sich beim nächsten empfangenen Signal).

Konfigurationsparameter:

Parameter		
DEVICE		
CODE	5040	CCCC
CUX_HOUSE_CODE	5040	

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - Hauscode (muss auf den Wert von CUX_HOUSE_CODE gesetzt werden, damit die Steuerung funktioniert)

CUX_HOUSE_CODE - aktueller FHT-Hauscode vom CUL-Modul

Name	Kanal	Parameter
Heizung-R:1	Ch.: 1	DIMMER|PAIR ☐
Heizung-R:2	Ch.: 2	DIMMER|PAIR ☐
Heizung-R:3	Ch.: 3	DIMMER|PAIR ☐
Heizung-R:4	Ch.: 4	DIMMER|PAIR ☐

PAIR - [x] Anlernen an Ventilantriebe: Dazu ist zuvor die Taste am Ventilantrieb so lange zu drücken, bis ein Signalton ertönt und **AC** im Display erscheint.
Jetzt den PAIR [x] Parameter vom entsprechenden Ventilantrieb setzen und mit „OK“ bestätigen.
Das erfolgreiche Anlernen bestätigt der Ventilantrieb mit einer Tonfolge und einem blinkenden Antennensymbol. Nach dem ersten empfangenen Datenpaket ertönt ebenfalls ein Signalton und das Antennensymbol bleibt dauerhaft aktiv.

Kanaltypen:

Kanaltyp	Kanalnummer
DIMMER	1..8

Kanaltyp DIMMER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
LEVEL	float	100%	lesend schreibend	Ventilöffnung
OLD_LEVEL	action		schreibend	letzten Level wiederherstellen

5.4.4 (10) TF-2 Tür/Fensterkontakt (2-Kanal)

Dieses Gerät dient zum Anzeigen vom Status der FHT80 TF-2 Tür/Fensterkontakte. Die Adresse des TF-2 muss zuerst aus dem CUxD-Terminal ausgelesen werden. Es sind die ersten 6 Zeichen nach dem „T“.

Der Tür/Fenster-Kontakt der FHT80-Serie bietet je nach Konfiguration die Erkennung von 2 getrennten Kanälen (intern/extern). Wenn beide Kanäle überwacht werden, ist der 1. Kanal der interne Kontakt und der 2. Kanal der externe Kontakt.

Ist am Sensor nur ein Kontakt konfiguriert, dann werden auf dem CUxD beide Kontakte synchron angezeigt. Es besteht leider keine Möglichkeit, anhand der Empfangsdaten die Anzahl der konfigurierten Kanäle zu ermitteln.

Achtung: Der TF-2 meldet sich nicht sofort nach Zustandsänderung sondern nur alle 60 Sekunden. Bei aktuellen TF-2 Sensoren ist leider keine getrennte Auswertung beider Kontakte mehr möglich.

Darstellung:

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung	
Filter	Filter	Filter			
TF-2:1		Verschluß		Offen	Verschlossen
TF-2:2		Verschluß		Offen	Verschlossen

Konfigurationsparameter:

Parameter		
DEVICE		
CODE	423DAE	CCCCAA

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - Adresse des TF-2

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1..2

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Zustand des Sensors

5.5 HMS-Sensoren und Gefahrenmelder {CUX}

Für den Datenempfang ist ein CUL, CUN oder CUNO notwendig (z.B. CUL V3). Der Funk-Empfang ist leider nicht sehr zuverlässig, so dass viele Daten verloren gehen. Die CUx-Datenpakete der HMS-Sensoren beginnen immer mit „**H**“. Als Gerätecode müssen die ersten 4 Zeichen nach dem „H“ (aus dem CUxD-Terminal) eingetragen werden.

Die Batteriewarnungen der HMS-Sensoren werden als LOWBAT-Servicemeldung auf der CCU dargestellt und löschen sich automatisch nach einem Batteriewechsel.

Nach einem Batteriewechsel ändert sich normalerweise die HMS-Adresse des betreffenden Sensors. Die neue HMS-Adresse ist mit dem konfigurierten Code in der Gerätekonfiguration entweder manuell oder automatisch (LEARN) abzugleichen.

Beim Einsatz eines einzigen Sensors im Empfangsbereich kann für den entsprechenden Gerätetyp eine automatische Erkennung aller Sensoren dieses Gerätetyps aktiviert werden (ALL_CODES). Damit wird der CODE-Parameter ignoriert.

Konfigurationsparameter:

Parameter	
DEVICE	
CODE	1CC4
ALL_CODES	☐
LEARN	☐
STATISTIC	☑
RESET	☐

DEVICE - CUx Geräte-ID oder TTY oder leer

CODE - HMS-Adresse

ALL_CODES - [x] bei Aktivierung werden alle Sensoren **dieses Gerätetyps** unabhängig von ihrer HMS-Adresse erkannt. Der Parameter sollte nur aktiviert werden, wenn sich nicht mehr als 1 Sensor des entsprechenden Gerätetyps in der Empfangsreichweite befindet. Dadurch entfällt die Eingabe der HMS-Adresse und deren eventueller Abgleich nach einem Batteriewechsel. Die unter CODE eingegebene Adresse wird ignoriert.

LEARN - [x] über diesen Schalter kann der CODE eines neu eingeschalteten Sensors automatisch dem CUxD-Gerät zugewiesen werden. Der Schalter ist unmittelbar vor dem Einsetzen der neuen Batterien zu aktivieren und deaktiviert sich automatisch, sobald ein Sensor gefunden wurde. Diese Funktion arbeitet nur innerhalb von 1 Minute nach dem Einsetzen der Batterien. Gegebenenfalls ist der Batteriewechsel zu wiederholen.

STATISTIC - [x] (nur bei HMS 100 T und HMS 100 TF) aktivieren der Statistik-Option

RESET - [x] Rücksetzen aller Statistikdaten (wenn STATISTIC aktiviert ist)

5.5.1 (12) HMS 100 TF (Temperatur/Luftfeuchte-Sensor)

Paketaufbau (hexadezimal kodiert):

HAAAAF**0**$T_1T_2H_1T_3H_2H_3$RR

AAAA.....Adresse

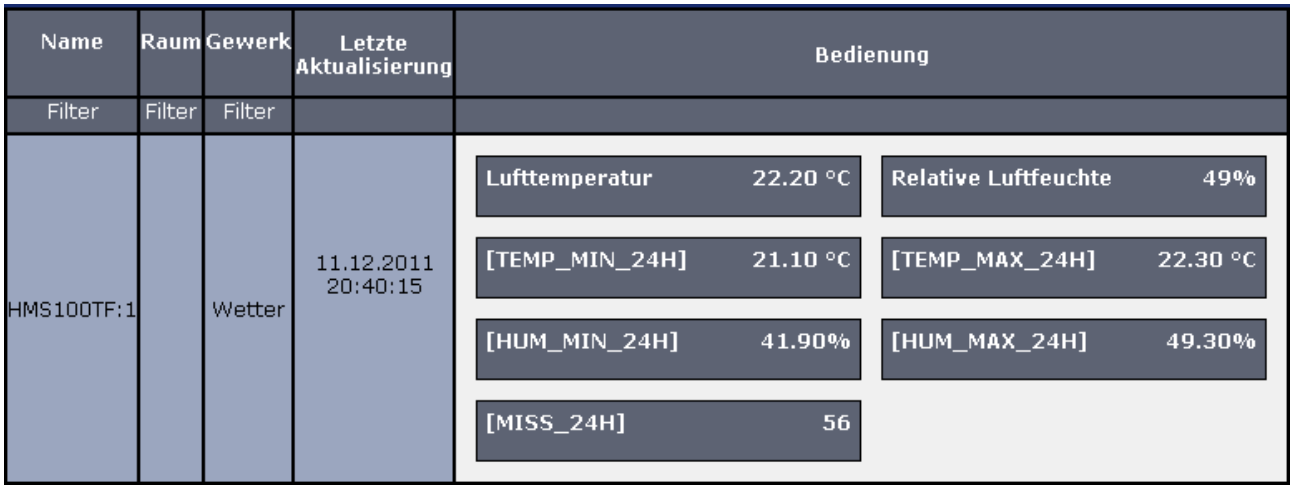
F.....Flags und Temperatur-Vorzeichen

$T_1T_2T_3$.....Temperatur = $(10*T_3 + T_1 + T_2/10)$ * Vorzeichen

$H_1H_2H_3$.....Luftfeuchte = $(10*H_2 + H_3 + H_1/10)$

RR.....RSSI-Wert vom Empfang (optional)

Zusätzlich zu den gemessenen Temperatur- und Luftfeuchte-Daten werden neben einer Statistik auch der Taupunkt und die absolute Luftfeuchte nach den unter der URL http://www.wettermail.de/wetter/feuchte.html beschriebenen Formeln berechnet.



Kanaltypen:

Kanaltyp	Kanalnummer
WEATHER	1

Kanaltyp WEATHER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
TEMPERATURE	float	°C	lesend	Temperatur
HUMIDITY	integer	%	lesend	Relative Luftfeuchte (gerundet)
HUMIDITYF	float	%	lesend	Relative Luftfeuchte
DEW_POINT	float	°C	lesend	Taupunkt
ABS_HUMIDITY	float	g/m³	lesend	Absolute Luftfeuchte
folgende Datenpunkte sind nur bei aktivierter Statistikfunktion verfügbar				
TEMP_MIN_24H	float	°C	lesend	min. Temperatur (24 Stunden)
TEMP_MAX_24H	float	°C	lesend	max. Temperatur (24 Stunden)
HUM_MIN_24H	float	%	lesend	min. Luftfeuchte (24 Stunden)
HUM_MAX_24H	float	%	lesend	max. Luftfeuchte (24 Stunden)
MISS_24H	integer		lesend	fehlende Datenpakete in den letzten 24 Stunden (maximal: 276)

5.5.2 (13) HMS 100 T (Temperatursensor)

Paketaufbau (hexadezimal kodiert):

HAAAA**F**1$T_1T_2$0$T_3$00RR

AAAA.....Adresse

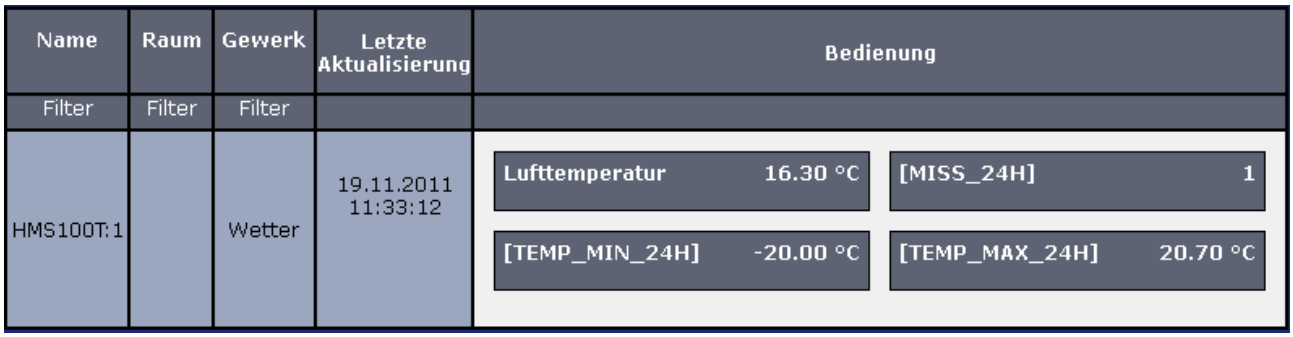
F.....Flags und Temperatur-Vorzeichen

$T_1T_2T_3$.....Temperatur = $(10*T_3 + T_1 + T_2/10)$ * Vorzeichen

RR.....RSSI-Wert vom Empfang (optional)

Mit einem CUNO und entsprechender Verkabelung können über dieses Gerät zusätzlich 1-Wire Temperatursensoren (DS18S20) eingebunden werden. Die Initialisierung des CUNO und Aktivierung der HMS-Emulation mit einem Abfrageintervall von 30s kann dabei aus dem CUxD-Terminal mit den folgenden Befehlen erfolgen:

```
Oi
OHt30
OHo
```



Kanaltypen:

Kanaltyp	Kanalnummer
WEATHER	1

Kanaltyp WEATHER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
TEMPERATURE	float	°C	lesend	Temperatur
folgende Datenpunkte sind nur bei aktivierter Statistikfunktion verfügbar				
MISS_24H	integer		lesend	fehlende Datenpakete in den letzten 24 Stunden (maximal: 270)
TEMP_MIN_24H	float	°C	lesend	min. Temperatur (24 Stunden)
TEMP_MAX_24H	float	°C	lesend	max. Temperatur (24 Stunden)

5.5.3 (14) HMS 100 W/WD (Wassermelder)

Wassermelder mit und ohne abgesetzten Sensor.

Paketaufbau (hexadezimal kodiert):

HAAAAF2SSCCRR

AAAA.....Adresse

F.....Flags

SS.....Status

CC.....Zähler für Datentelegramme

RR.....RSSI-Wert vom Empfang (optional)

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Zustand des Sensors
COUNTER	integer	lesend	Nummer des aktuellen Datensatzes (0..255)

5.5.4 (15) HMS 100 RM / RM 100-2 (Gefahrenmelder)

Rauchmelder

Paketaufbau (hexadezimal kodiert):

HAAAAF3SSCCRR

AAAA.....Adresse

F.....Flags

SS.....Status

CC.....Zähler für Datentelegramme

RR.....RSSI-Wert vom Empfang (optional)

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
HMS100RM:1		Sicherheit		O.K. Gefahr

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Zustand des Sensors
COUNTER	integer	lesend	Nummer des aktuellen Datensatzes (0..255)

5.5.5 (18) HMS 100 MG (Gefahrenmelder)

Gasmelder (Methan/Erdgas)

Paketaufbau (hexadezimal kodiert):

HAAAAF6SSCCRR

AAAA.....Adresse

F.....Flags

SS.....Status

CC.....Zähler für Datentelegramme

RR.....RSSI-Wert vom Empfang (optional)

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
HMS100MG:1		Sicherheit		

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Zustand des Sensors
COUNTER	integer	lesend	Nummer des aktuellen Datensatzes (0..255)

5.5.6 (20) HMS 100 CO (Gefahrenmelder)

Gasmelder (Kohlenmonoxid)

Paketaufbau (hexadezimal kodiert):

HAAAAF8SSCCRR

AAAA.....Adresse

F.....Flags

SS.....Status

CC.....Zähler für Datentelegramme

RR.....RSSI-Wert vom Empfang (optional)

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
HMS100CO:1		Sicherheit		O.K. Gefahr

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Zustand des Sensors
COUNTER	integer	lesend	Nummer des aktuellen Datensatzes (0..255)

5.5.7 (26) HMS 100 FIT (Gefahrenmelder)

Beim HMS100 FI-Trenner wird nur der Status angezeigt.

Paketaufbau (hexadezimal kodiert):

HAAAAFESSCCRR

AAAA.....Adresse

F.....Flags

SS.....Status

CC.....Zähler für Datentelegramme

RR.....RSSI-Wert vom Empfang (optional)

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
FI-Trenner:1		Verschluß		O.K. Gefahr

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Zustand des Sensors
COUNTER	integer	lesend	Nummer des aktuellen Datensatzes (0..255)

5.5.8 (16) HMS 100 TFK (Tür-/Fensterkontakt)

Tür-/Fensterkontakt.

Paketaufbau (hexadezimal kodiert):
HAAAAF4SSCCRR *(normally closed)*
HAAAAF5SSCCRR *(normally open)*
AAAA.....Adresse
F.....Flags
SS.....Status
CC.....Zähler für Datentelegramme
RR.....RSSI-Wert vom Empfang (optional)

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

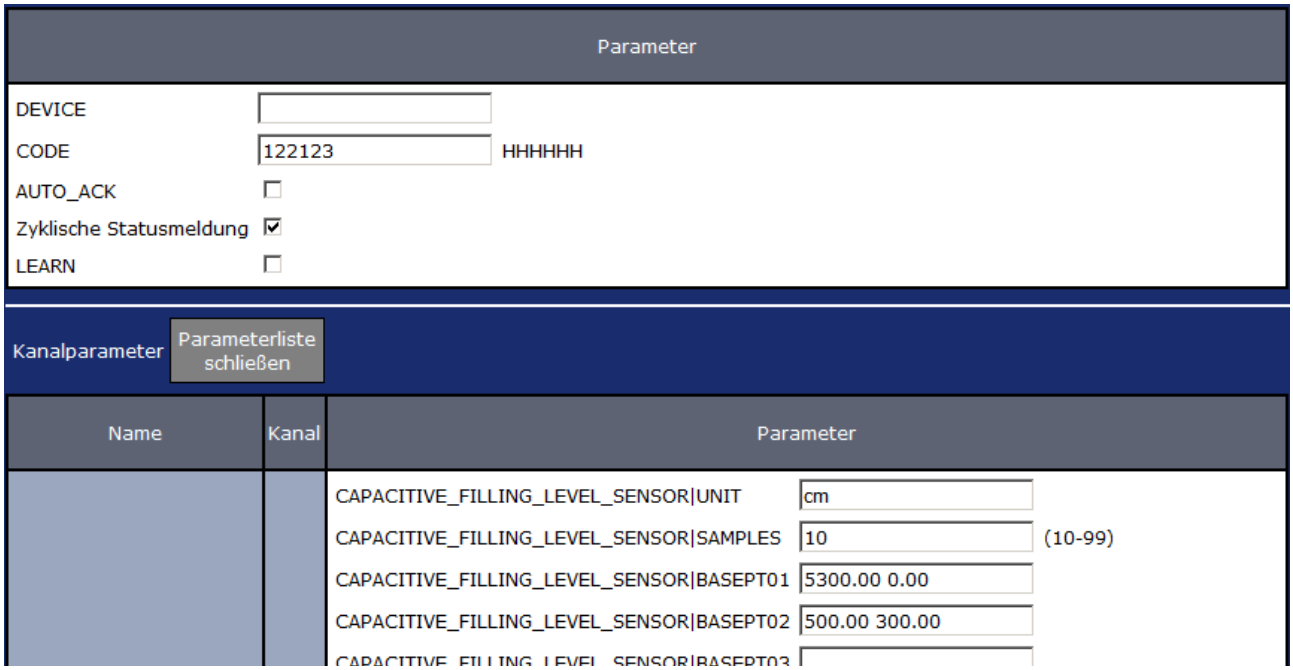
DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Zustand des Sensors
COUNTER	integer	lesend	Nummer des aktuellen Datensatzes (0..255)

5.6 (29) BidCos Geräte {CUX}

Da dieses Protokoll ein anderes Datenübertragungsverfahren nutzt, ist der Empfang nur über einen eigenen CUL möglich. Bei gleichzeitigem Empfang von FS20-Geräten (1kHz, AM) und BidCos-Geräten (20kHz, FM) müssen zwei CULs an die CCU angeschlossen werden. Die Initialisierung des BidCos CULs kann dann mit **TTYINIT** erfolgen (z.B: TTYINIT=ttyACM1:X21\nAr). Danach werden vom CUL auch alle HomeMatic-Datenpakete empfangen.

5.6.1 Füllstandsmesser KFM 100 S

Der kapazitive Füllstandsmesser liefert abhängig vom Flüssigkeitsstand eine Frequenz als Ausgangswert. Die Frequenz lässt sich mit diesem CUxD-Gerät anhand einer Tabelle von Messwerten auf andere Werte (cm, Liter, ...) abbilden. Zwischen den Tabellenwerten erfolgt eine lineare Interpolation. Auf diese einfache Art und Weise lässt sich jede mögliche Behälterform im CUxD abbilden.



DEVICE	- USB-ID oder TTY oder leer
CODE	- Hexadezimale Adresse des KFM100S
AUTO_ACK	- [x] automatische Empfangsbestätigung des Messwertes an den Sensor. Das ist nur notwendig, wenn der Sensor vorher an das Anzeigegerät KFM100E angelernt wurde und das Anzeigegerät danach ausgeschaltet wird.
CYCLIC_INFO_MSG	- [x] zyklische Statusmeldung des Sensors überwachen. Wenn der Sensor sich nicht mindestens einmal innerhalb von 3 Stunden meldet, erfolgt eine **UNREACH**-Servicemeldung auf der CCU.
LEARN	- [x] das Anlernen der Adresse erfolgt beim Empfang des nächsten Messwertes
UNIT	- Einheit des Messwertes (cm, Liter, ...)
SAMPLES	- Anzahl der Stützwerte für die Interpolationsfunktion
BASEPT01..99	- Stützwerte: Frequenz und entsprechender Messwert (durch Leerzeichen getrennt), werden automatisch sortiert

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
KFM100S:1			14.08.2012 22:44:41	[FREQ] 5434.00 Hz / Aktueller Füllstand 0.000 cm

Kanaltypen:

Kanaltyp	Kanalnummer
CAPACITIVE_FILLING_LEVEL_SENSOR	1

Kanaltyp CAPACITIVE_FILLING_LEVEL_SENSOR:

DP-Name	Typ	Zugriff	Beschreibung
FREQ	float	lesend	Frequenzwert des Sensors
FILLING_LEVEL	float	lesend	Füllstand nach Interpolation der Werte

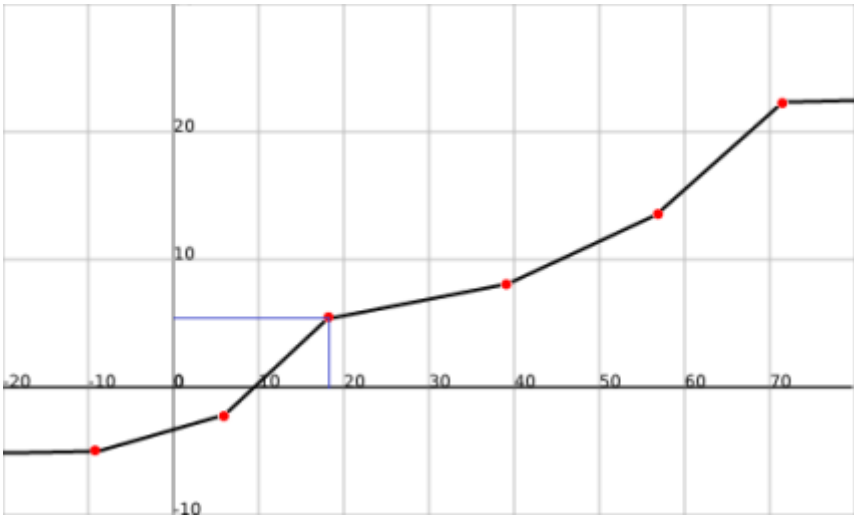
5.7 (90) Universal Wrapper Devices

Für die Funktion dieser Geräte müssen bei Überwachung von Nicht-CUxD-Geräten die CUxD-Parameter **SUBSCRIBE_RF=1** und/oder **SUBSCRIBE_WR=1** gesetzt sein.

5.7.1 (1) Transform Device

Wenn HM-Sensoren umgebaut werden (z.B. Temperatursensoren als Helligkeitssensoren siehe hier: HomeMatic-Forum) ist es oft störend, dass weiterhin z.B. Temperaturwerte angezeigt werden, die dem entsprechenden Widerstand eines Temperatursensors, aber nicht dem Helligkeitswert entsprechen.

Das Transform-Device kann gemessene Werte beliebiger Sensoren anhand definierter Stützpunkte in neue Werte umrechnen. Die Kennlinie wird über maximal 99 eingegebene Stützpunkte (x,y) linear interpoliert. Durch die gleichzeitige Berechnung von Zentral- und Durchschnittswerten, können Störungen herausgefiltert werden.



Beim Empfang eines Wertes (x) wird der entsprechende Datenpunkt-Wert (y) ausgegeben.

Zusätzlich erfolgt mit den Ergebnissen eine Zentralwert- und Durchschnittsberechnung über eine zuvor definierte Anzahl (**HISTORY_BUFFER**) von gespeicherten Werten.

Auf der CCU wird das Gerät z.B. folgendermaßen dargestellt:

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		
Helligkeit:1	Garten	Wetter	28.10.2011 08:02:19	[LIGHT] 66.30% [MEDIAN] 61.50% [MEAN] 59.56%

Über den Geräteparameter „**HIDE_DPS**“ können alle Datenpunkte ausgeblendet werden, um das Gerät z.B. nur als Container für eigene Systemvariablen zu nutzen.

Name	Raum	Gewerk	Letzte Aktualisierung	Bedienung
Filter	Filter	Filter		

Konfigurationsparameter:

USE_HMDATAPT	☑	
HMSERIAL	IEQ0166432:1	SERIAL:X
HMDATAPT	TEMPERATURE	
HSS_TYPE	WEATHER	
Fehler	OK!	

USE_HMDATAPT - [x] HM-Gerät überwachen (ggf. **SUBSCRIBE_RF=1** und/oder **SUBSCRIBE_WR=1**). Zum Beschreiben des Datenpunktes per HM-Script muss dieser Parameter deaktiviert sein!

HMSERIAL - HM-Serien- und Kanalnummer des zu überwachenden Gerätes (kann beliebiges HomeMatic oder CUxD-Gerät sein)

HMDATAPT - auszuwertender Datenpunkt des zu überwachenden Gerätes. Nach der Konfiguration werden im CUxD-Syslog die vorhandenen Datenpunkte des Gerätes angezeigt.

HSS_TYPE - Anzeige des gefundenen HomeMatic-Kanaltyps (zur Kontrolle)

Ch.: 1	WRAPPER|HIDE_DPS	☐	
	WRAPPER|DATAPT	BRIGHTNESS	
	WRAPPER|UNIT	%	
	WRAPPER|HISTORY_BUFFER	3	(1-30)
	WRAPPER|HYSTERESIS	0.00	(0.00-1000.00)
	WRAPPER|SAMPLES	2	(2-99)
	WRAPPER|BASEPT01	-20.00 0.00	
	WRAPPER|BASEPT02	80.00 100.00	

HIDE_DPS - Ausblenden der Datenpunkte dieses Gerätes

DATAPT - neuer Datenpunkt-Name des Wrapper-Device. Er wird im Wrapper-Device als [Name] ausgegeben. Dies ist auch der Name des Datenpunktes auf den über Programmverknüpfungen zugegriffen werden kann. (Groß- / Kleinschreibung beachten).

UNIT - Einheit für **DATAPT**

HISTORY_BUFFER - Speicherwerte für die Zentralwert- und Durchschnittsberechnung.

HYSTERESIS - Erst bei Änderung des Daten-, Mittel- oder Zentralwertes um den angegebenen absoluten Wert, werden Updates zur CCU gesendet. So können zum Beispiel Schwankungen der Messwerte heraus-gefiltert werden.

SAMPLES - Anzahl der Stützpunkte

BASEPT01..99 - Stützpunkte für die Interpolationsfunktion auf die Datenwerte (jeweils „x y“ mit Leerzeichen getrennt).
Nach der Eingabe werden die Stützpunkte automatisch aufsteigend in der X-Dimension sortiert. Nicht genutzte Stützpunkte sind leer zu lassen. X-Werte, die kleiner als der kleinste Stützpunkt (x) sind, werden als Y-Wert des kleinsten Stützpunktes und X-Werte, die größer als der größte Stützpunkt sind, als Y-Wert des größten Stützpunktes ausgegeben.

Kanaltypen:

Kanaltyp	Kanalnummer
WRAPPER	1

Kanaltyp WRAPPER:

DP-Name	Typ	Zugriff	Beschreibung
[*DATAPT*]	float	lesend	definierter DP-Name
MEDIAN	float	lesend	Zentralwert
MEAN	float	lesend	Mittelwert (Durchschnitt)
SET_STATE	float	scheibend	neuen Eingabewert schreiben (z.B. per HM-Script, **USE_HMDATAPT** muss deaktiviert sein!)

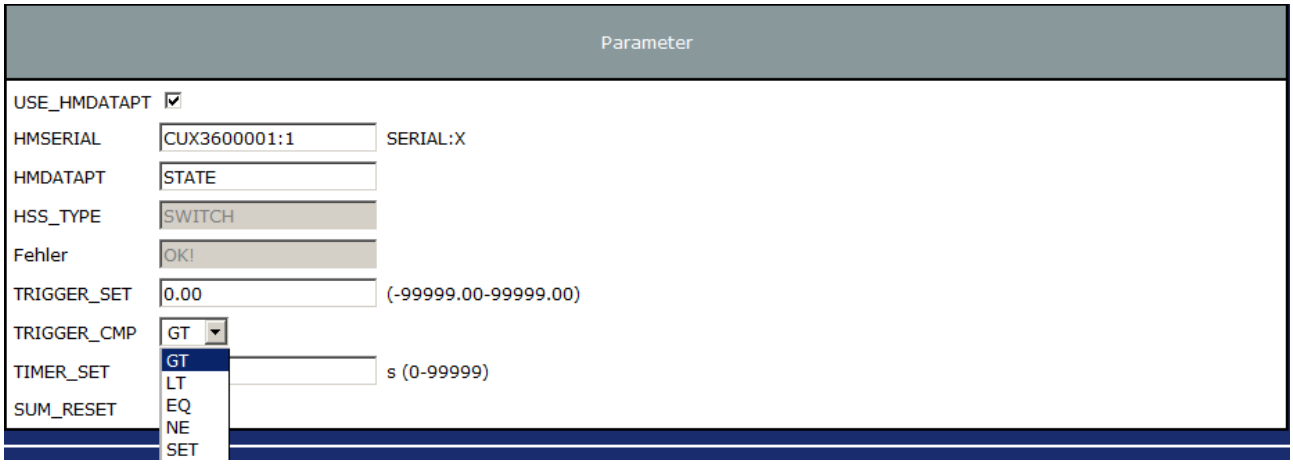
5.7.2 (2) State-Monitor Device

Die kontinuierliche und zeitnahe Überwachung von Betriebszeiten und Gerätezuständen ist mit Hilfe von periodisch aufgerufenen HM-Scripts und Systemvariablen auf der CCU leider nur mit relativ hohem Einsatz von Systemressourcen und somit auf Kosten der Performance des Gesamtsystems möglich.

Dieses Gerät ermöglicht es auf einfache Weise und ressourcenschonend, An- und Aus-Zeiten eines beliebigen Gerätes bzw. Datenpunktes zu überwachen. Diese Werte können dann wiederum in Programmverknüpfungen und Scripts genutzt werden. Es werden neben der letzten bzw. aktuellen Ein-/Aus-Zeit auch die aufsummierten Zeiten und Zustandswechsel der letzten 60 Minuten, 24-Stunden und 168-Stunden (1 Woche) ausgegeben. Zusätzlich können Events generiert und ein weiterer Zeitbereich (Kanal 3) zwischen 1 und 168 Stunden frei definiert werden.

Auch nach einem CCU-Neustart bleiben bis dahin aufgezeichnete Statistik-Daten erhalten.

Konfigurationsparameter:



USE_HMDATAPT - [x] HM-Gerät überwachen (ggf. **SUBSCRIBE_RF=1** und/oder **SUBSCRIBE_WR=1**). Zum Beschreiben des Datenpunktes per HM-Script muss dieser Parameter deaktiviert sein!

HMSERIAL - HM-Serien- und Kanalnummer des zu überwachenden Gerätes (kann beliebiges HomeMatic oder CUxD-Gerät sein)

HMDATAPT - auszuwertender Datenpunkt des zu überwachenden Gerätes. Nach der Konfiguration werden im CUxD-Syslog die vorhandenen Datenpunkte des Gerätes angezeigt.

HSS_TYPE - Anzeige des gefundenen HomeMatic-Kanalyps (zur Kontrolle)

TRIGGER_SET - Trigger für empfangenen Datenwert (s. **TRIGGER_CMP**)

TRIGGER_CMP - Vergleichsoperation des überwachten Datenwertes mit dem **TRIGGER_SET**-Wert. Der Gerätestatus (TRUE/FALSE) ergibt sich aus dem Ergebnis dieser Operation. Dabei bedeuten:

GT	*größer als* **TRIGGER_SET**
LT	*kleiner als* **TRIGGER_SET**
EQ	*gleich* **TRIGGER_SET**
NE	*ungleich* **TRIGGER_SET**
SET	ist immer **TRUE**

TIMER_SET - ist dieser Wert größer als 0, dann wird der Gerätestatus nach Ablauf der Sekunden automatisch auf FALSE zurückgesetzt.

SUM_RESET - **TIME_ON_SUM** und **SWITCH_SUM** auf 0 setzen!

Ch.: 1	WRAPPER|TIME_ON_EVENT_SET	20	s (0-99999)
	WRAPPER|TIME_OFF_EVENT_SET	10	s (0-99999)
	WRAPPER|CMD_EXEC_TRUE		
	WRAPPER|CMD_EXEC_FALSE		
Ch.: 2	Keine Parameter einstellbar		
Ch.: 3	WRAPPER|DEFINE_HOURS	0	h (0-168)

TIME_ON_EVENT_SET - nach den Sekunden im TRUE-Status (ohne Unterbrechung) wird **TIME_STATE** auf TRUE gesetzt und **TIME_ON_EVENT** zur CCU gesendet.

TIME_OFF_EVENT_SET - nach den Sekunden im FALSE-Status (ohne Unterbrechung) wird **TIME_STATE** auf FALSE gesetzt und **TIME_OFF_EVENT** zur CCU gesendet.

CMD_EXEC_TRUE - Leer oder Befehlszeile, die bei **TIME_ON_EVENT** aufgerufen wird

CMD_EXEC_FALSE - Leer oder Befehlszeile, die bei **TIME_OFF_EVENT** aufgerufen wird

DEFINE_HOURS - Anzahl der Stunden für das frei definierbare Intervall auf Kanal 3

Die 1h-Werte gelten für 60 Minuten und werden alle 2 Minuten aktualisiert.

Die **TIME_ON_SUM** und **SWITCH_SUM** Werte werden alle 2 Minuten aktualisiert.

Die 24h-Werte gelten immer für 24h ab der letzten vollen Stunde. Wenn es z.B. 21:40 Uhr ist, dann zählt der Wert von 22:00 Uhr am Vortag bis 21:00 Uhr am aktuellen Tag. Neue 24h-Werte gibt es immer nur zur vollen Stunde.

Bei Änderungen der Parameter **USE_HMDATAPT**, **HMSERIAL** oder **HMDATAPT** werden alle intern aufgezeichneten Statistik-Daten des Gerätes zurückgesetzt.

state:1			17.10.2013 19:18:00	[STATE=TRUE]		[SWITCH_1H]	0.00
				[TIME_ON_1H]	60.00 min	[TIME_OFF_1H]	0.00 min
				[TIME_ON]	558.95 min	[TIME_OFF]	1.62 min
				[TIME_ON_SUM]	1378.48 min	[SWITCH_SUM]	8.00
				[TIME_ON_EVENT]	0	[TIME_OFF_EVENT]	0
state:2			17.10.2013 19:00:00	[TIME_ON_24H]	1318.38 min	[TIME_OFF_24H]	1.62 min
				[SWITCH_24H]	1.00	[PERCENT_ON_24H]	100%
				[TIME_ON_168H]	1318.38 min	[TIME_OFF_168H]	1.62 min
				[SWITCH_168H]	1.00	[PERCENT_ON_168H]	100%
state:3				[TIME_ON_HHH]	0.00 min	[TIME_OFF_HHH]	0.00 min
				[SWITCH_HHH]	0.00	[PERCENT_ON_HHH]	0%

Kanaltypen:

Kanaltyp	Kanalnummer
SWITCH	1
WRAPPER	2,3

Kanaltyp SWITCH (1) (Aktualisierung alle 2 Minuten und bei Status-Updates):

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	**TRUE** => eingeschaltet, **FALSE** => ausgeschaltet
SWITCH_1H	float	lesend	Einschaltvorgänge (**TRUE**) in der letzten Stunde
TIME_ON_1H	float	lesend	Zeitdauer im Status **TRUE** in der letzten Stunde
TIME_OFF_1H	float	lesend	Zeitdauer im Status **FALSE** in der letzten Stunde
TIME_ON	float	lesend	letzte/aktuelle Zeitdauer im Status **TRUE**
TIME_OFF	float	lesend	letzte/aktuelle Zeitdauer im Status **FALSE**
TIME_ON_SUM	float	lesend	Zeitdauer im Status **TRUE** seit dem letzten **SUM_RESET** Event (upd. 2 Minuten)
SWITCH_SUM	float	lesend	Anzahl der Einschaltvorgänge seit dem letzten **SUM_RESET** Event (upd. 2 Minuten)
TIME_ON_EVENT	action	event	Ereignis wird nach der im Parameter **TIME_ON_EVENT_SET** eingestellten Zeit ausgelöst
TIME_OFF_EVENT	action	event	Ereignis wird nach der im Parameter **TIME_OFF_EVENT_SET** eingestellten Zeit ausgelöst
TIME_STATE	boolean	lesend	Status wird nach der konfigurierten Verzögerung beim Auslösen von **TIME_ON_EVENT** (**TRUE**) und **TIME_OFF_EVENT** (**FALSE**) aktualisiert.
SUM_RESET	action	schreibend	**TIME_ON_SUM** und **SWITCH_SUM** auf 0 zurücksetzen
SET_STATE	float	scheibend	neuen Eingabewert schreiben (z.B. per HM-Script, **USE_HMDATAPT** muss deaktiviert sein!)

Kanaltyp WRAPPER (2) (Aktualisierung jede Stunde für die vergangenen Stunden):

DP-Name	Typ	Zugriff	Beschreibung
TIME_ON_24H	float	lesend	Zeitdauer im Status TRUE in letzten 24 Stunden
TIME_OFF_24H	float	lesend	Zeitdauer im Status FALSE in letzten 24 Stunde
SWITCH_24H	float	lesend	Anzahl der Einschaltvorgänge in den letzten 24 Stunden
PERCENT_ON_24H	integer	lesend	% im Status TRUE in den letzten 24 Stunden
TIME_ON_168H	float	lesend	Zeitdauer im Status TRUE in letzten 7 Tagen
TIME_OFF_168H	float	lesend	Zeitdauer im Status FALSE in letzten 7 Tagen
SWITCH_168H	float	lesend	Anzahl der Einschaltvorgänge in den letzten 7 Tagen
PERCENT_ON_168H	integer	lesend	% im Status TRUE in den letzten 7 Tagen

Kanaltyp WRAPPER (3) (Aktualisierung jede Stunde für die vergangenen Stunden):

DP-Name	Typ	Zugriff	Beschreibung
TIME_ON_HHH	float	lesend	Zeitdauer im Status TRUE im Intervall
TIME_OFF_HHH	float	lesend	Zeitdauer im Status FALSE im Intervall
SWITCH_HHH	float	lesend	Anzahl der Einschaltvorgänge im Intervall
PERCENT_ON_HHH	integer	lesend	% im Status TRUE im Intervall

Bei jedem Befehlsaufruf (**CMD_EXEC_TRUE, CMD_EXEC_FALSE**) werden zusätzliche **Umgebungsvariablen** gesetzt:

CUXD_DEVICE aktuelles CUxD-Gerät: CUX9002xxx

CUXD_STATE Ein (1), Aus (0) Schaltzustand des Gerätes (**STATE**)

In der **Befehlszeile** können dabei folgende Platzhalter genutzt werden:

$DEVICE$ entspricht ***CUXD_DEVICE***

$STATE$ entspricht ***CUXD_STATE***

HM-Scriptbeispiel zum Löschen der SUM-Werte:

```
dom.GetObject("CUxD.CUX9001xxx:1.SUM_RESET").State(1);
```

Beispiel zum sofortigen Ein- und verzögerten Ausschalten eines HM-Funk-Schaltaktors mit der Seriennummer *JEQ0123456* ohne Programmverknüpfung:

Zuerst sind die Geräteparameter zum Triggern z.B. auf einen Tür-/Fensterkontakt zu setzen. Danach dann Kanal 1 folgendermaßen konfigurieren:

TIME_ON_EVENT_SET= 0s

TIME_OFF_EVENT_SET= 30s

CMD_EXEC_TRUE= /usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.*JEQ0123456*:1.STATE 1

CMD_EXEC_FALSE= /usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.*JEQ0123456*:1.STATE 0

5.7.3 (3) Thermostat Device

Mit diesem Wrapper-Device kann man systemfremde Temperatur/Luftfeuchte Sensoren auf einfache Weise und ohne den Umweg über Systemvariablen auf der CCU abbilden. Es kann auch an einen bereits im System vorhandenen Wetter-Sensor Kanal angekoppelt, oder per HM-Script gesetzt werden.

Die Ankopplung an das CUxD-interne CUX-THFILE-Gerät (siehe TH-DIR=) ermöglicht auf einfache Weise z.B. mittels digitemp ausgelesene 1-Wire Sensoren in die WebUI einzubinden.

Um auch mit unkalibrierten Sensoren vernünftige Ergebnisse zu erhalten, besteht die Möglichkeit einen Temperatur bzw. Luftfeuchte Offset zu konfigurieren.

Neben der Berechnung von Taupunkt und absoluter Luftfeuchte in g/kg oder g/m³ nach den Formeln unter http://www.wettermail.de/wetter/feuchte.html bzw. der Beschreibung unter http://www.thermoguard.ch/download/Theorie_der_Feuchte.pdf werden auch Statistikdaten ausgegeben.

Zusätzlich ist ein Zweipunkt- und Universal-PID-Regler mit konfigurierbarer Hysterese für Heizungs- oder Kühlungs- bzw. Befeuchtungs- oder Entfeuchtungsanwendungen implementiert. Auch hier kann der Sollwert entweder durch die direkte Kopplung mit einem vorhandenen bereits im System konfigurierten Thermostaten synchronisiert oder per HM-Script bzw. WebUI gesetzt werden.

Bei Be- und Entfeuchtung sollte die absolute Luftfeuchte in g/kg als Regelgröße dienen, da diese die geringste zeitliche Schwankung aufweist.

Ein **OFFSET**-Parameter ermöglicht die Verschiebung des Einstellbereiches eines angekoppelten HM-Wandthermostaten, um Temperaturen außerhalb des Bereiches von 6 bis 30°C regeln zu können. Bei Verwendung z.B. als umschaltbarer Heiz-/Kühlregler kann (wenn bei der Konfiguration **INVERT** deaktiviert ist) durch Eingabe eines negativen **OFFSET**-Wertes (z.B. -0.5) ein Nullenergieband (im Beispiel 1.0 Grad) erzeugt werden.

Mit einem Thermostaten (z.B. HomeMatic-Wandthermostat oder FHT80b) und einem Schaltaktor (z.B. HomeMatic, FS20, EnOcean, usw.) kann somit ganz ohne HM-Script, nur über eine einfache Programmverknüpfung oder eine konfigurierbare **Befehlszeile**, mit der CCU eine elektrische Heizung geregelt werden.

Zusätzlich kann der Stellwert des PID-Reglers in ein PWM-Signal gewandelt werden.

Nach einem CCU-Neustart bleiben alle zuletzt empfangenen Werte einschließlich der Statistiken erhalten.

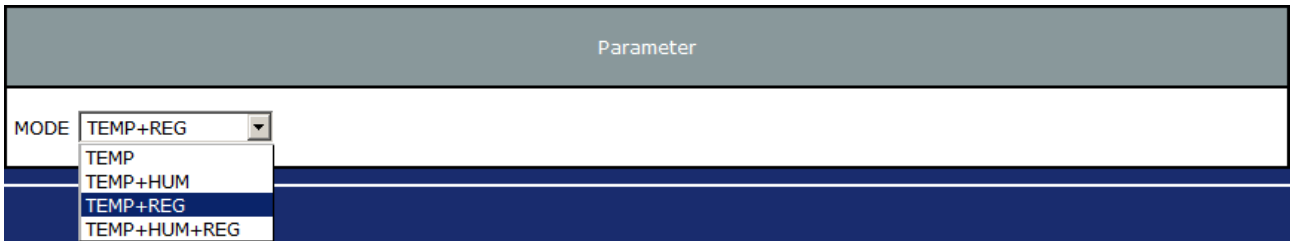
Bei jedem Befehlsaufruf (**CMD_EXEC**) werden zusätzliche **Umgebungsvariablen** gesetzt:

CUXD_DEVICE	aktuelles CUxD-Gerät: CUX9002xxx
CUXD_VALUE	Ein (1), Aus (0) Zustand bzw. Stellwert vom PID-Regler
CUXD_DIFF	Differenz: Sollwert - Istwert
CUXD_INVERT	1 wenn Regelung invertiert (INVERT=1), sonst 0
CUXD_INV0VAL	***CUXD_VALUE*** wenn Regelung nicht invertiert (INVERT=0), sonst 0
CUXD_INV1VAL	***CUXD_VALUE*** wenn Regelung invertiert (INVERT=1), sonst 0

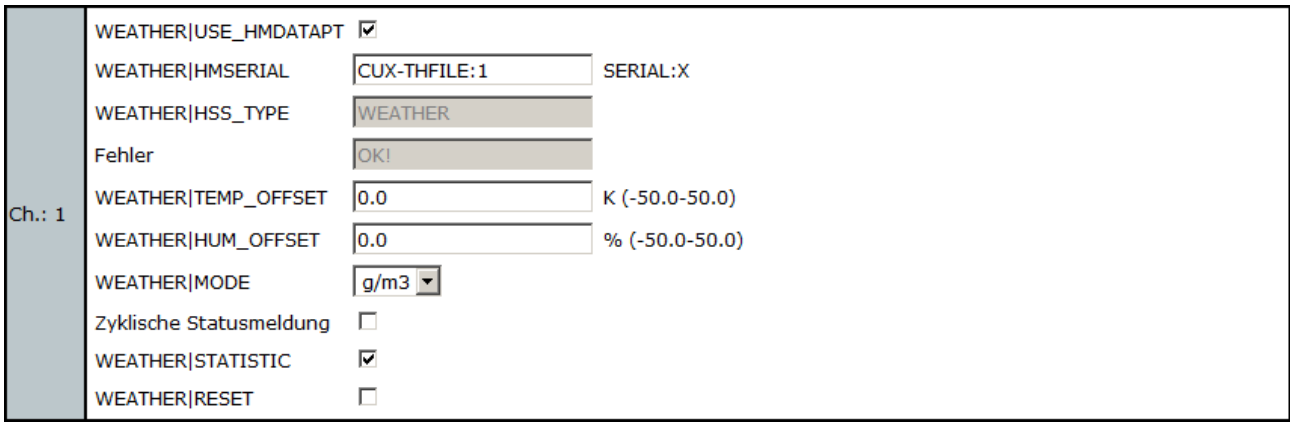
In der **Befehlszeile** können dabei folgende Platzhalter genutzt werden:

$DEVICE$	entspricht ***CUXD_DEVICE***
$VALUE$	entspricht ***CUXD_VALUE***
$DIFF$	entspricht ***CUXD_DIFF***
$INVERT$	entspricht ***CUXD_INVERT***
$INV0VAL$	entspricht ***CUXD_INV0VAL***
$INV1VAL$	entspricht ***CUXD_INV1VAL***

Konfigurationsparameter:



MODE - Auswahl der bereitgestellten Datenpunkte des Wrapper-Gerätes (Temperatur, Luftfeuchte, Regulator)



USE_HMDATAPT - [x] HM-Gerät überwachen (ggf. **SUBSCRIBE_RF=1** und/oder **SUBSCRIBE_WR=1**). Zum direkten Beschreiben des Datenpunktes per HM-Script muss dieser Parameter deaktiviert sein! (die beiden Kanäle werden unabhängig voneinander konfiguriert)

HMSERIAL - HM-Serien- und Kanalnummer des zu überwachenden Gerätes (kann beliebiger HomeMatic oder CUxD-Kanal mit *TEMPERATURE / ACTUAL_TEMPERATURE* und *HUMIDITY / ACTUAL_HUMIDITY* Datenpunkten sein)

TEMP_OFFSET - fester Temperatur-Offset zur Korrektur von Sensorabweichungen

HUM_OFFSET - fester Luftfeuchte-Offset zur Korrektur von Sensorabweichungen

MODE - Berechnung der absoluten Luftfeuchte in $\frac{g}{m^3}$ oder $\frac{g}{kg}$

CYCLIC_INFO_MSG - [x] Aktualisierung der WEATHER-Datenpunkte überwachen. Wenn bei aktivierter Funktion innerhalb von 60 Minuten keine Aktualisierung erfolgt, dann wird eine **UNREACH**-Servicemeldung zur CCU gesendet.

STATISTIC - [x] aktivieren der Tagesstatistik Datenpunkte

RESET - [x] Rücksetzen der Tagesstatistik

Ch.: 2	CLIMATECONTROL_REGULATOR|MODE	temperature	
	CLIMATECONTROL_REGULATOR|USE_HMDATAPT	☑	
	CLIMATECONTROL_REGULATOR|HMSERIAL	CUX0800001:2	SERIAL:X
	CLIMATECONTROL_REGULATOR|HSS_TYPE	CLIMATECONTROL_REGU	
	Fehler	OK!	
	CLIMATECONTROL_REGULATOR|INVERT_SETPOINT	☐	
	CLIMATECONTROL_REGULATOR|OFFSET	0.0	(-50.0-50.0)
	CLIMATECONTROL_REGULATOR|MIN	6	(-100-300)
	CLIMATECONTROL_REGULATOR|MAX	30	(-100-300)
	CLIMATECONTROL_REGULATOR|AUTO_INVERT	☐	
	CLIMATECONTROL_REGULATOR|INVERT	☐	
	CLIMATECONTROL_REGULATOR|HYSTERESIS	5.0	(0.0-1000.0)
	CLIMATECONTROL_REGULATOR|CMD_EXEC	/usr/local/addons/therm	
	CLIMATECONTROL_REGULATOR|CONTROLLER	☑	
	CLIMATECONTROL_REGULATOR|XP	4.0	(0.0-2000.0)
	CLIMATECONTROL_REGULATOR|TN	600	s (0-10800)
	CLIMATECONTROL_REGULATOR|TV	120	s (0-5400)
	CLIMATECONTROL_REGULATOR|TZ	10	s (0-1800)
	CLIMATECONTROL_REGULATOR|MAX_VAL	100	(1-100000)

MODE - Auswahl des zu regelnden Wertes (Temperatur, relative Luftfeuchte, absolute Luftfeuchte)

USE_HMDATAPT - [x] Gerät überwachen (ggf. **SUBSCRIBE_RF=1** und/oder **SUBSCRIBE_WR=1**) oder DP per HM-Script beschreiben (die beiden Kanäle werden unabhängig voneinander konfiguriert)

HMSERIAL - HM-Serien- und Kanalnummer des zu überwachenden Gerätes (kann beliebiger HomeMatic oder CUxD-Kanal mit *SETPOINT / SET_TEMPERATURE* Datenpunkten sein)

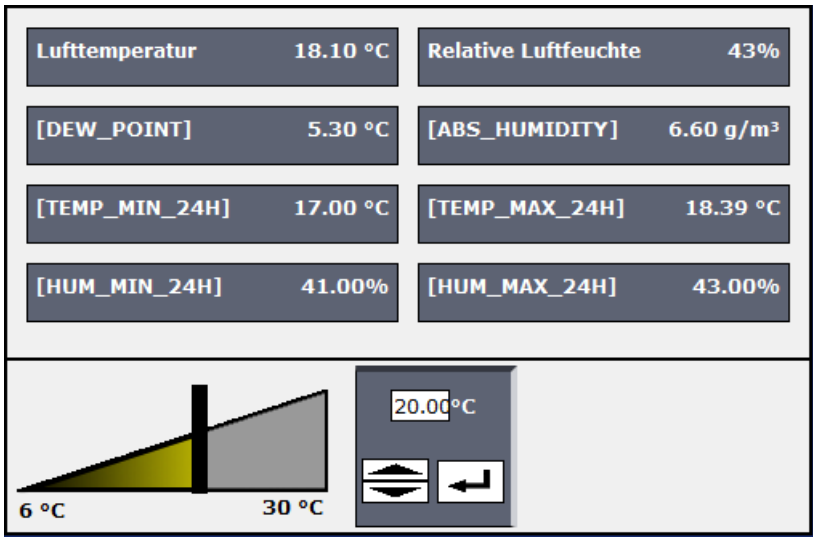
INVERT_SETPOINT - empfangenen SOLL-Wert bei direkter Kopplung an einen HM-Thermostaten invertieren. (aus 20°C wird -20°C und aus 10°C wird -10°C). Auch falls die Darstellung bei Stellwerten < 0°C in der WebUI nicht stimmt, werden die Werte intern trotzdem richtig umgerechnet (siehe Systemprotokoll)!

OFFSET - Offset (±50.00, Auflösung: 0,01) auf den SOLL-Wert zur Verschiebung des Reglerbereiches (kann auch zur Erzeugung eines Nullenergiebandes verwendet werden!)

MIN - Mindestwert für virtuellen Sollwert-Regler

MAX - Maximalwert für virtuellen Sollwert-Regler

Zweipunkt- bzw. Universal-PID-Regler	
AUTO_INVERT	- die Invertierung der Regelung erfolgt automatisch in Abhängigkeit vom Soll- und Istwert unter Einberechnung des **OFFSET**s für das Nullenergie-band. *((Sollwert - Istwert) < Offset → INVERT=1)* Der aktuelle Zustand kann über den Datenpunkt **SET_INVERT** ausgelesen werden.
INVERT	- Regelung invertieren (kühlen bzw. entfeuchten) für **Zweipunkt-** und **PID-Regler**
HYSTERESIS	- **Zweipunkt-Regler:** Schalt-Hysterese (Auflösung: 0,02) **PID-Regler:** Hysterese bei der Aktualisierung des Stellwertes (Auflösung: 0,01)
CMD_EXEC	- Befehlszeile, die bei Status- bzw. Stellwertänderung aufgerufen wird
CONTROLLER	- [x] im CUxD implementierten **Universal-PID-Regler** aktivieren (siehe FHZ-Forum). Ansonsten ist der **Zweipunkt-Regler** aktiviert. Die folgenden 5 Parameter dienen zur Konfiguration des integrierten **Universal-PID-Reglers**, der die errechneten Stellwerte im LEVEL-Datenpunkt bereitstellt.
XP	- **Proportional-Band** zur Berechnung der Regelverstärkung (bei 0 ist die Regelverstärkung unendlich und der Stellwert wechselt somit nur zwischen 0 und MAX_VAL)
TN	- **Nachstellzeit** in s (bei 0 ist der **I-Anteil** abgeschaltet!)
TV	- **Vorhaltezeit** in s (bei 0 ist der **D-Anteil** abgeschaltet!)
TZ	- **Zykluszeit** für die Berechnung der **I- und D-Anteile** in Sekunden (bei 0 arbeitet der Regler „event driven“ und die Berechnung erfolgt nur nach Änderung des Soll-Wertes, nach Aktualisierung des Ist-Wertes oder bei direkter Abfrage des Datenpunktes aus HM-Script heraus: dom.GetObject("CUxD.CUX9002xxx:2.LEVEL").State())
MAX_VAL	- Maximalwert des LEVEL-Datenpunktes (Stellwert) zur Anpassung an verschiedene Aktoren



Der **PWM-Kanal** (3) dient zur Wandlung des Stellsignals vom PID-Regler in ein PWM-Signal zur Ansteuerung von beliebigen Schaltaktoren.

Ch.: 3	SWITCH|TZ	600	s (0-7200)
	SWITCH|MIN	30	s (0-3600)
	Einschaltdauer	☑	
	SWITCH|CMD_EXEC_TRUE	/usr/local/addons/cuxd/e	
	SWITCH|CMD_EXEC_FALSE		

TZ - **Zykluszeit** des PWM-Wandlers (Periodendauer) in Sekunden (10...7200). Bei 0 ist der PWM-Wandler deaktiviert!

MIN - Minimale Ein- und Ausschaltzeit in Sekunden (**MIN** darf maximal die Hälfte von **TZ** sein!) Ergibt die lineare Umrechnung des Stellwertes eine Einschaltdauer **< MIN/2**, ist das Signal (**STATE**) ständig aus; bei einer rechnerischen Einschaltdauer **>** (**TZ - MIN/2**) ständig ein.

ON_TIME - [x] (**Einschaltdauer**) muss aktiviert werden, wenn man den Aktor zusätzlich über eine Einschaltdauer (**ON_TIME**) steuern möchte. Dann wird der **STATE**-Datenpunkt auch aktualisiert, wenn sich der Status in der Zykluszeit nicht ändert.

CMD_EXEC_TRUE - Leer oder Befehlszeile, die bei der Aktualisierung vom PWM-Signal mit **STATE**=TRUE aufgerufen wird

CMD_EXEC_FALSE - Leer oder Befehlszeile, die bei der Aktualisierung vom PWM-Signal mit **STATE**=FALSE aufgerufen wird

Bei jedem Befehlsaufruf (**CMD_EXEC_TRUE, CMD_EXEC_FALSE**) werden zusätzliche **Umgebungsvariablen** gesetzt:

CUXD_DEVICE aktuelles CUxD-Gerät: CUX9002xxx

CUXD_STATE Ein (1), Aus (0) Schaltzustand des PWM-Signals (**STATE**)

CUXD_ONTIME Einschaltdauer bei **CMD_EXEC_TRUE** bzw. Ausschaltdauer bei **CMD_EXEC_FALSE** in Sekunden.

In der **Befehlszeile** können dabei folgende Platzhalter genutzt werden:

$STATE$ entspricht ***CUXD_STATE***

$ONTIME$ entspricht ***CUXD_ONTIME***

Beispiel zur Ansteuerung eines HM-Schaltaktors (HEQ0504751) mittels Befehlszeile (timer.tcl-Script) und Einschaltdauer (**ON_TIME** ist aktiviert und **CMD_EXEC_FALSE** leer):

CMD_EXEC_TRUE =

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.HEQ0504751:1.STATE 1 0 0 $ONTIME$
```

Beispiel zur Ansteuerung eines HM-Schaltaktors (HEQ0504751) mittels Befehlszeile (timer.tcl-Script) ohne Nutzung der Einschaltdauer (**ON_TIME** ist deaktiviert):

CMD_EXEC_TRUE =

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.HEQ0504751:1.STATE 1
```

CMD_EXEC_FALSE =

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.HEQ0504751:1.STATE 0
```

Kanaltypen:

Kanaltyp	Kanalnummer
WEATHER	1
CLIMATECONTROL_REGULATOR	2
SWITCH	3

Kanaltyp WEATHER:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
TEMPERATURE	float	°C	lesend	Temperatur
HUMIDITY	integer	%	lesend	Relative Luftfeuchte (gerundet)
DEW_POINT	float	°C	lesend	Taupunkt
ABS_HUMIDITY	float	g/m³ g/kg	lesend	Absolute Luftfeuchte (siehe MODE-Parameter) in g/m³ bzw. g/kg
SET_TEMPERATURE	float	°C	schreibend	Temperatur manuell setzen
SET_HUMIDITY	float	%	schreibend	Luftfeuchte manuell setzen
folgende Datenpunkte sind nur bei aktivierter Statistikfunktion verfügbar				
TEMP_MIN_24H	float	°C	lesend	min. Temperatur (24 Stunden)
TEMP_MAX_24H	float	°C	lesend	max. Temperatur (24 Stunden)
HUM_MIN_24H	float	%	lesend	min. Luftfeuchte (24 Stunden)
HUM_MAX_24H	float	%	lesend	max. Luftfeuchte (24 Stunden)

Kanaltyp CLIMATECONTROL_REGULATOR:

DP-Name	Typ	Zugriff	Beschreibung
SETPOINT	float	lesend schreibend	Sollwert (bei Thermostat-Kopplung nur lesend!)
STATE	boolean	lesend	Schaltzustand der Zweipunktregelung abhängig von Soll/Ist-Werten (WebUI-Bezeichnung: Ventil schließen (false) / Ventil öffnen (true)) (wird nur bei Änderung aktualisiert)
LEVEL	float	lesend	Stellwert des Universal-PID-Reglers (wird nur bei Änderung aktualisiert)
SET_INVERT	boolean	lesend schreibend	Regelung invertieren (Kühlbetrieb, Entfeuchtung) für Zweipunkt- und PID-Regler. Neben dem Aktualisieren des **INVERT**-Parameters wird zusätzlich der aktuelle **OFFSET** invertiert, der I-Wert des Reglers zurückgesetzt und der Stellwert (**STATE** bzw. **LEVEL**) aktualisiert. Dieser Datenpunkt wird mit jeder Aktualisierung der Stellwerte ausgegeben.

Kanaltyp SWITCH (PWM-Wandler):

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend	Ein (1), Aus (0) Schaltzustand des PWM-Signals

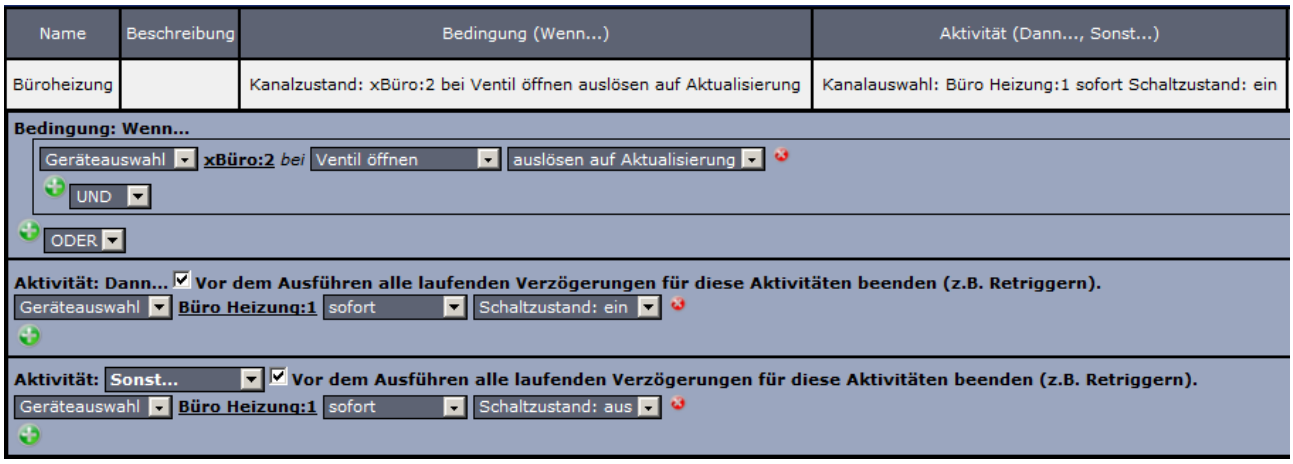
HM-Scriptbeispiel zur Berechnung der absoluten Feuchte mit eigenen Werten:

```
dom.GetObject("CUxD.CUX9002001:1.SET_TEMPERATURE").State("20.5");
dom.GetObject("CUxD.CUX9002001:1.SET_HUMIDITY").State("62.5");
var abs_hum = dom.GetObject("CUxD.CUX9002001:1.ABS_HUMIDITY").State();
```

Da die gesetzten Werte im CUxD-Gerät zwischengespeichert werden, müssen im folgenden nur noch die geänderten Werte gesetzt werden. Zum Beispiel:

```
dom.GetObject("CUxD.CUX9002001:1.SET_HUMIDITY").State("68");
var abs_hum = dom.GetObject("CUxD.CUX9002001:1.ABS_HUMIDITY").State();
```

Beispiel einer Programmverknüpfung zur Heizungssteuerung über einen Schaltaktor:



Beispiel für Zweipunktregler zum automatischen Heizen/Kühlen mit Offset und Hysterese:
AUTO_INVERT = 1, OFFSET = 1.0, HYSTERESE = 1.0

Danach folgendes Verhalten (Anfangstemperatur = 20°C, SETPOINT = 22°C):

	aktuelle Temperatur	INVERT	STATE	INV0VAL (Heizung)	INV1VAL (Kühlung)	Funktion
	Temperatur steigend					
	< 21,5°C	0	1	**1**	0	heizen
	< 23,5°C	0	0	0	0	aus
	>= 23,5°C	1	1	0	**1**	kühlen
	Temperatur fallend					
	> 22,5°C	1	1	0	**1**	kühlen
	> 20,5 °C	1	0	0	0	aus
	<= 20,5°C	0	1	**1**	0	heizen

Im Beispiel wird bis 21,5°C geheizt und dann ab 23,5°C gekühlt. Die Kühlung wird bei 22,5°C abgeschaltet und die Heizung ab 20,5°C wieder eingeschaltet. Somit wird ein häufiges Um- bzw. Ein-/Ausschalten von Heizung und Kühlung vermieden.

Beispiel (Ist-Temperatur vom Wandthermostaten):

Parameter	
MODE	TEMP+REG

Kanal	Parameter		
Ch.: 1	WEATHER|USE_HMDATAPT	☑	
	WEATHER|HMSERIAL	LEQ0017543:1	SERIAL:X
	WEATHER|HSS_TYPE	WEATHER_TRANSMIT	
	Fehler	OK!	
	WEATHER|TEMP_OFFSET	0.0	K (-50.0-50.0)
	WEATHER|HUM_OFFSET	0.0	% (-50.0-50.0)
	WEATHER|MODE	g/m3	
	Zyklische Statusmeldung	☐	
	WEATHER|STATISTIC	☑	
	WEATHER|RESET	☐	

Kanaladresse des Temperatursensors

PWM Fußbodenheizungsregler (Sollwert vom Wandthermostaten):

Ch.: 2	CLIMATECONTROL_REGULATOR|MODE	temperature	
	CLIMATECONTROL_REGULATOR|USE_HMDATAPT	☑	
	CLIMATECONTROL_REGULATOR|HMSERIAL	LEQ0017543:2	SERIAL:X
	CLIMATECONTROL_REGULATOR|HSS_TYPE	THERMALCONTROL_TRAN	
	Fehler	OK!	
	CLIMATECONTROL_REGULATOR|INVERT_SETPOINT	☐	
	CLIMATECONTROL_REGULATOR|OFFSET	0.0	(-50.0-50.0)
	CLIMATECONTROL_REGULATOR|MIN	6	(-100-300)
	CLIMATECONTROL_REGULATOR|MAX	30	(-100-300)
	CLIMATECONTROL_REGULATOR|AUTO_INVERT	☐	
	CLIMATECONTROL_REGULATOR|INVERT	☐	
	CLIMATECONTROL_REGULATOR|HYSTERESIS	1.0	(0.0-1000.0)
	CLIMATECONTROL_REGULATOR|CMD_EXEC		
	CLIMATECONTROL_REGULATOR|CONTROLLER	☑	
	CLIMATECONTROL_REGULATOR|XP	4.0	(0.0-2000.0)
	CLIMATECONTROL_REGULATOR|TN	7200	s (0-10800)
	CLIMATECONTROL_REGULATOR|TV	0	s (0-5400)
	CLIMATECONTROL_REGULATOR|TZ	120	s (0-1800)
	CLIMATECONTROL_REGULATOR|MAX_VAL	100	(1-100000)

Kanaladresse des Thermostaten

Ch.: 3	SWITCH|TZ	3600	s (0-7200)	
	SWITCH|MIN	60	s (0-3600)	
	Einschaltdauer	☐		
	SWITCH|CMD_EXEC_TRUE	/usr/local/addons/cuxd/extra/timer.tcl	CUxD.CUX0200001:1.STATE	1
	SWITCH|CMD_EXEC_FALSE	/usr/local/addons/cuxd/extra/timer.tcl	CUxD.CUX0200001:1.STATE	0

Kanaladresse des Schaltaktors

2-Punkt Regler (Sollwert vom Wandthermostaten):

Ch.: 2	CLIMATECONTROL_REGULATOR|MODE	temperature	
	CLIMATECONTROL_REGULATOR|USE_HMDATAPT	☑	
	CLIMATECONTROL_REGULATOR|HMSERIAL	LEQ0017543:2	SERIAL:X
	CLIMATECONTROL_REGULATOR|HSS_TYPE	THERMALCONTROL_TRAN	
	Fehler	OK!	
	CLIMATECONTROL_REGULATOR|INVERT_SETPOINT	☐	
	CLIMATECONTROL_REGULATOR|OFFSET	0.0	(-50.0-50.0)
	CLIMATECONTROL_REGULATOR|MIN	6	(-100-300)
	CLIMATECONTROL_REGULATOR|MAX	30	(-100-300)
	CLIMATECONTROL_REGULATOR|AUTO_INVERT	☐	
	CLIMATECONTROL_REGULATOR|INVERT	☐	
	CLIMATECONTROL_REGULATOR|HYSTERESIS	0.2	(0.0-1000.0)
	CLIMATECONTROL_REGULATOR|CMD_EXEC	/usr/local/addons/cuxd/e	
	CLIMATECONTROL_REGULATOR|CONTROLLER	☐	
	CLIMATECONTROL_REGULATOR|XP	4.0	(0.0-2000.0)
	CLIMATECONTROL_REGULATOR|TN	600	s (0-10800)
	CLIMATECONTROL_REGULATOR|TV	120	s (0-5400)
	CLIMATECONTROL_REGULATOR|TZ	10	s (0-1800)
	CLIMATECONTROL_REGULATOR|MAX_VAL	1	(1-100000)
Ch.: 3	SWITCH|TZ	0	s (0-7200)
	SWITCH|MIN	0	s (0-3600)
	Einschaltdauer	☐	
	SWITCH|CMD_EXEC_TRUE		
	SWITCH|CMD_EXEC_FALSE		

DEAKTIVIERT!

CMD_EXEC (Schaltaktor)

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.JEQ0205721:1.STATE $VALUE$
```

5.8 (28) System-Devices

Die folgenden Umgebungsvariablen sind allen aufgerufenen **CMD_**-Scripts gesetzt und entsprechen den gleichnamigen INI-Parametern:

CUXD_ADDRESS_BUFFER	*CUXD_HM_SCRIPTHOST*	*CUXD_RCVLOGSIZE*
CUXD_AUTOSAVE	*CUXD_HM_SCRIPTPORT*	*CUXD_RPCHOST*
CUXD_BACKUPCMD	*CUXD_HTTP_REFRESH*	*CUXD_RPCPORT*
CUXD_CUXINITCMD	*CUXD_LISTENPORT*	*CUXD_SUBSCRIBE_RF*
CUXD_DEVDATAFORMAT	*CUXD_LOGFILE*	*CUXD_SUBSCRIBE_WR*
CUXD_DEVLOGFILE	*CUXD_LOGFILEMOVE*	*CUXD_SYSLOGMOVE*
CUXD_DEVLOGMOVE	*CUXD_LOGLEVEL*	*CUXD_TERMINALLINES*
CUXD_DEVLOGSIZE	*CUXD_LOGSIZE*	*CUXD_TH_DIR*
CUXD_DEVTIMEFORMAT	*CUXD_MOUNTCMD*	*CUXD_UMOUNTCMD*

5.8.1 System.Timer (16 Kanäle)

Mit diesem Gerät wird nach Ablauf einer vorgegebenen Zeit ein Event ausgelöst. So sind flexiblere Zeitsteuerungen als mit dem Zeitmodul der WebUI möglich. Pro CUxD-Gerät können bis zu 16 unabhängige Timer angelegt werden. Die Steuerung der einzelnen Timer erfolgt über die entsprechenden Geräte-Datenpunkte.

Neben der Möglichkeit des Abbruchs und Retriggerns läuft ein einmal gestarteter Timer auch nach einem CCU- bzw. CUxD-Neustart weiter.

Anstelle eines Events kann zum Ablauf des Timers auch sofort eine beliebige Befehlszeile auf der CCU ausgeführt werden. Wird dieses Feature mit dem automatischen Timer-Neustart kombiniert, erfolgt der Aufruf periodisch im gesetzten Intervall. Danach kann dann der exit()-Code oder die Standardausgabe des Befehls weiter verarbeitet werden.

Parameter		
CHANNELS	16	(1-16)

CHANNELS - Anzahl der Timer-Kanäle (maximal 16). Sollte die Darstellung nicht aktualisiert werden, dann hilft ein Reload im Webbrowser.

Ch.: 1	SYSTEM|TIMER_PRESET		
	SYSTEM|REPEAT	☐	
	SYSTEM|CMD_EXEC		
	SYSTEM|EXEC_FUNC	system()	
	SYSTEM|EXEC_TIMEOUT	1	Min (1-999)

TIMER_PRESET - manuelles Setzen des Timers (siehe **TIMER_SET** Datenpunkt)

REPEAT - [x] automatischer Neustart des Timers nach Ablauf der Zeit

CMD_EXEC - Befehlszeile, die nach dem Ablauf der Zeit ausgeführt werden soll

EXEC_FUNC - interner Befehl zum Ausführen der unter **CMD_EXEC** definierten Befehlszeile:

- system() - in **CMD_RET** steht der exit()-Code des Befehls
- popen() - in **CMD_RET** steht die Standardausgabe des Befehls
- process - in **CMD_RET** steht der exit()-Code des Befehls und über die Standardausgabe können CUxD Datenpunkte auf der CCU beliebig geändert werden.

EXEC_TIMEOUT - maximale Laufzeit in Minuten bevor der Prozess durch den CUxD automatisch beendet wird

Bei jedem Befehlsaufruf wird die zusätzliche **Umgebungsvariable** gesetzt:

CUXD_CHANNEL aufgerufener Kanal des System.Exec-Gerätes: CUX2800xxx:x

In der **Kommandozeile** kann dabei folgender Platzhalter genutzt werden:

$CHANNEL$ entspricht ***CUXD_CHANNEL***

$TS**<*format*>**$ Zufallszahl/-zeit im **TIMER_SET**-Format definieren

Beispielscript zum Ändern vom Status der Geräte CUX2801002:1 und CUX2801002:2, ausgeführt mit **EXEC_FUNC=process**

```
#!/bin/sh
echo "CUX2801002:1.STATE 0"
echo "CUX2801002:2.STATE 1"
```

Timer:1			30.11.2012 19:33:48	

Kanaltypen:

Kanaltyp	Kanalnummer
SYSTEM	1...16

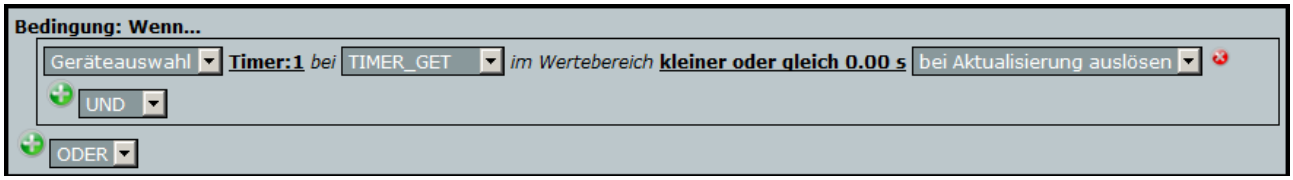
Kanaltyp SYSTEM:

DP-Name	Typ	Zugriff	Beschreibung
TIMER_STOP	action	schreibend	Abbruch des Timers → Tastendruck auf WebUI Dabei wird kein Timer-Event ausgelöst!
TIMER_SET	string	lesend schreibend	Setzen des Timers: „0“ - entspricht **TIMER_STOP** „*sss*“ - auslösen **in** *sss* Sekunden (relativ) „:*ss*“ - auslösen **um** XX:*XX*:*ss* (absolut) „*mm*:*ss*“ - auslösen **um** XX:*mm*:*ss* (absolut) „*hh*:*mm*:*ss*“ - auslösen **um** *hh*:*mm*:*ss* (absolut) „d:*hh*:*mm*:*ss*“ - auslösen **am** Wochentag *d* [0-6] **um** *hh*:*mm*:*ss* (absolut) „+*sss*“ - akt. Timer um *sss* Sekunden **erhöhen** „-*sss*“ - akt. Timer um *sss* Sekunden **verkürzen** zusätzliche Erweiterung: „... **r***nnn***zzz*“ - **zufälliges** Auslösen zwischen *<timer_set>* und *<timer_set>+nnn*zzz* Sekunden. *nnn* ist die zufällige Anzahl der Zeitschritte mit der Länge *zzz*.
TIMER_EVENT	action	event	Timer-Event wird beim Ablauf des Timers ausgelöst, wenn **CMD_EXEC** nicht gesetzt ist. Es kann zum Triggern von Programmverknüpfungen auf der CCU genutzt werden. (*siehe folgende Beispiele!*)
STATE	boolean	lesend schreibend	Dieser Datenpunkt kann beliebig initialisiert werden und wird mit jedem **TIMER_EVENT** auf **TRUE** gesetzt. Kann zum Triggern genutzt werden. (*siehe folgende Beispiele!*)
TIMER_GET	float	lesend	Auslesen der verbleibenden Zeit (in Sekunden) bis zum Ablauf des Timers. Kann zum Triggern genutzt werden. (*siehe folgende Beispiele!*)
CMD_RET	string	lesend	Nach dem Ausführen von **CMD_EXEC** wird in diesem Datenpunkt der exit()-Code (**EXEC_FUNC**=system) bzw. der STDOUT-Rückgabewert (**EXEC_FUNC**=popen) des ausgeführten Befehls übergeben.
WORKING	boolean	lesend	kennzeichnet einen aktiven Timer

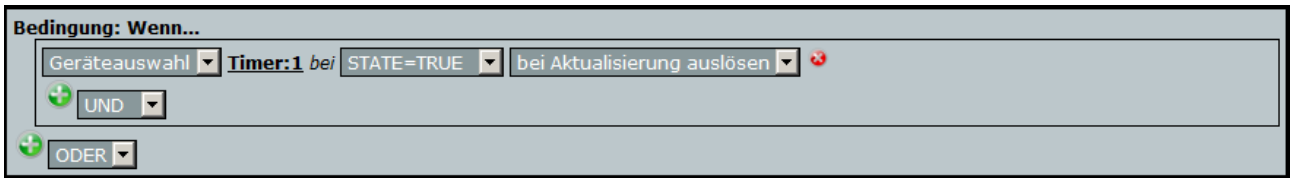
Programm auslösen, indem auf **TIMER_EVENT** getriggert wird:

Wird das Programm nicht ausgelöst, dann gibt es zwei weitere Möglichkeiten, das gleiche zu erreichen.

Programm auslösen, indem auf **TIMER_GET kleiner oder gleich 0** getriggert wird:

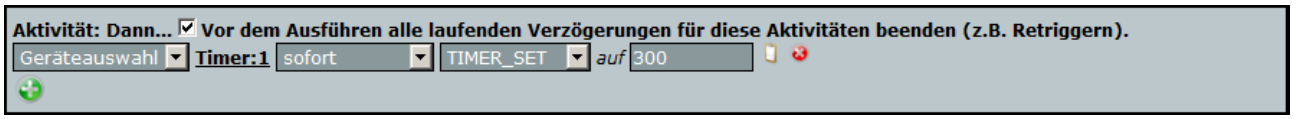


Programm auslösen, indem auf **STATE=TRUE** getriggert wird:



Es gibt 2 verschiedene Möglichkeiten, den Timer zu setzen. Entweder in Sekunden relativ zur aktuellen Uhrzeit, oder absolut zur Stunde bzw. zum Tag. Zusätzlich kann jedem Wert noch eine zufällige Zeit hinzugefügt werden.

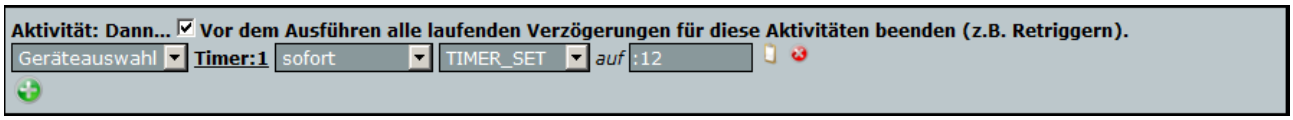
Event in 300s auslösen:



HM-Scriptbeispiel (CUX2800001 wurde zuvor angelegt!):

```
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State(300);
!-----------------------------------------------------------------------------
!zufällig im Bereich von 300s bis 420s auslösen ([0..120] * 1s step):
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State("300 r120");
!zufällig nach 300s, 360s, 420s oder 480s auslösen ([0..3] * 60s step):
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State("300 r3*60");
!60s zum aktuellen Timer addieren:
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State("+60");
!30s vom aktuellen Timer abziehen:
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State("-30");
```

Event um XX:XX:12 Uhr auslösen:

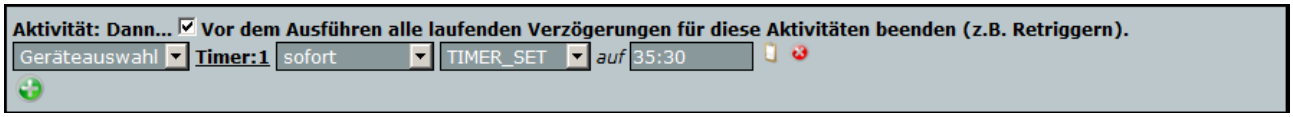


HM-Scriptbeispiel (CUX2800001 wurde zuvor angelegt!):

```
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State(":12");
```

Angenommen, es ist beim Setzen 15:30:00 Uhr, dann wird um 15:31:12 Uhr ausgelöst.
Ist es aber bereits nach 15:31:12 Uhr, dann wird um 15:32:12 Uhr ausgelöst.

Event um XX:35:30 Uhr auslösen:

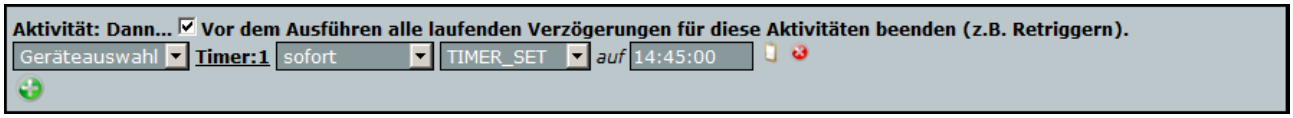


HM-Scriptbeispiel (CUX2800001 wurde zuvor angelegt!):

```
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State("35:30");
```

Angenommen, es ist beim Setzen 15:30:00 Uhr, dann wird um 15:35:30 Uhr ausgelöst.
Ist es aber bereits nach 15:35:30 Uhr, dann wird um 16:35:30 Uhr ausgelöst.

Event um 14:45:00 Uhr auslösen:

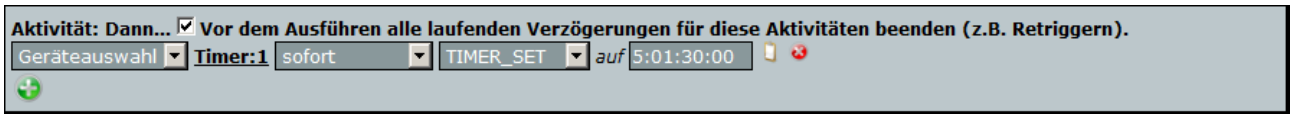


HM-Scriptbeispiel (CUX2800001 wurde zuvor angelegt!):

```
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State("14:45:00");
```

Ist es beim Setzen bereits nach 14:45:00, dann wird das Event erst am nächsten Tag ausgelöst.

Event am Freitag um 01:30:00 Uhr auslösen:



HM-Scriptbeispiel (CUX2800001 wurde zuvor angelegt!):

```
dom.GetObject("CUxD.CUX2800001:1.TIMER_SET").State("5:01:30:00");
```

Ist es beim Setzen bereits Freitag um 01:30:00 oder später, dann wird das Event erst eine Woche später am Freitag um 01:30:00 ausgelöst. Für die Wochentage können 0=Sonntag, 1=Montag, 2=Dienstag, 3=Mittwoch, 4=Donnerstag, 5=Freitag und 6=Samstag verwendet werden.

5.8.2 System.Exec (16 Kanäle)

Dieses Gerät dient als Ersatz der undokumentierten und fehleranfälligen system.exec() Funktion auf der CCU. Die Funktionalität wird über das Lesen bzw. Schreiben von Datenpunkten eines HM-Gerätes abgebildet. Damit existiert eine einfache Möglichkeit zum direkten Ausführen von Systembefehlen aus der WebUI bzw. der CCU-Logikschicht. Die konfigurierte Befehlszeile wird mittels C-Funktion system() bzw. popen() als überwachter **Hintergrundprozess** ausgeführt. Die Überwachung kann verhindert werden, wenn am Ende der Befehlszeile ein **&** Zeichen steht.

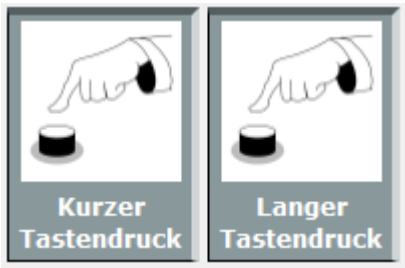
Weiterhin können bis zu 9 Geräteparameter, 5 Kanalparameter und 99 Parameter-Datenpunkte definiert werden. Diese Parameter können als Platzhalter in die Befehlszeile eingebaut werden. Alle Parameter können vor der Ersetzung in der Befehlszeile auch automatisch URL-Encoded werden.

Eine zusätzliche **CMD_TIMER** Befehlszeile pro Kanal ermöglicht die periodische Statusabfrage der Kanäle innerhalb des Gerätes unter Verwendung der gleichen konfigurierten Geräte- und Kanalparameter. Um Ressourcen zu schonen, werden sich überschneidende Timer-Prozesse pro Gerät nicht parallel, sondern nacheinander abgearbeitet.

Um unbeabsichtigte Programmaufrufe bei der Abfrage der Datenpunkte **CMD_RETS** und **CMD_RETL** zu vermeiden, muss vor der Abfrage dieser Datenpunkte eine '1' an den Datenpunkt **CMD_QUERY_RET** gesendet werden. Erst danach ist der Programmaufruf für die folgenden 10s freigeschaltet.

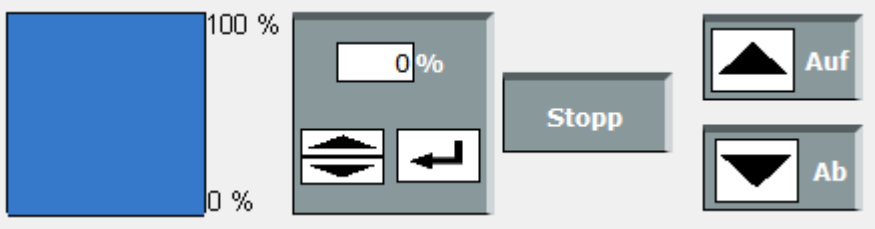
Die folgenden Controls müssen beim Anlegen des CUxD-Gerätes ausgewählt werden und ermöglichen den direkten Aufruf der mittels **CMD_SHORT** und **CMD_LONG** konfigurierten Kommandozeilen bei Änderung des Gerätezustands:

Taster (zustandslos)

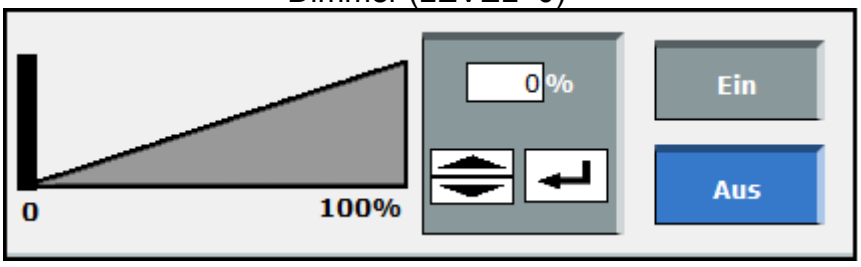


Schalter (STATE=false)

Jalousie (geschlossen, LEVEL=0)



Dimmer (LEVEL=0)



Konfigurationsparameter:

Parameter	
CHANNELS	3 (1-16)

CHANNELS - Anzahl der Geräte-Kanäle (maximal 16). Sollte die Darstellung nicht aktualisiert werden, dann hilft ein Reload im Webbrowser.

PARAM1..9 - Geräteparameter zur Ersetzung in der Befehlszeile

Kanal	Parameter		
Ch.: 1	SYSTEM|CMD_SHORT	/usr/local/addons/cuxd/ex	
	SYSTEM|CMD_LONG	/usr/local/ccu_backup 1	
	SYSTEM|EXEC_TIMEOUT	60	Min (1-999)
	SYSTEM|PARAMETER	2	(0-99)

CMD_SHORT - Befehlszeile mit Platzhaltern für Parameter (kurzer Tastendruck oder AUS oder OLD_LEVEL oder STOP)

CMD_LONG - Befehlszeile mit Platzhaltern für Parameter (langer Tastendruck oder EIN oder LEVEL-Wert)

EXEC_TIMEOUT - maximale Laufzeit in Minuten bevor der Prozess durch den CUxD automatisch beendet wird

CH_PARAM1..5 - Kanalparameter zur Ersetzung in der Befehlszeile

TIMER_PRESET - setzen des Timers (0=AUS, siehe System.Timer)

CMD_TIMER - Befehlszeile, die periodisch nach Ablauf des Timers aufgerufen wird

PARAMETER - Anzahl der optionalen Parameter-Datenpunkte (maximal 99) für diesen Kanal (Die Parameter werden vor dem Ersetzen der Platzhalter 1...99 in der Befehlszeile URL-Encoded). Es wird nur die Anzahl der in diesem Parameter definierten optionalen Parameter angelegt.

MAX_VAL - Maximalwert (1 bis 65535) bei Level=100% des Dimmer- bzw. Jalousie-Kanals. Zum Beispiel werden bei **MAX_VAL**=1000 die %-Werte in Werte zwischen 0 und 1000 umgerechnet.

*Werden als Parameter Anführungs- oder Sonderzeichen (z.B. '=', '&', ...) verwendet, dann kann es im Eingabefeld der WebUI Probleme geben! In diesem Fall ist das direkte Setzen über den Datenpunkt **CMD_SETS** bzw. **CMD_SETL** möglich.*

*Sind **CMD_SHORT** bzw. **CMD_LONG** leer, dann ändert sich nur der Status dieses Gerätes auf der CCU ohne Aufruf einer Systemfunktion! Darüber kann das Gerät als Dummy-Gerät auf der CCU für eigene Anwendungen genutzt werden.*

Kanaltypen:

Kanaltyp	Kanalnummer
KEY / SWITCH / BLIND / DIMMER	1..16

Kanaltyp KEY / SWITCH / BLIND / DIMMER:

DP-Name	Typ	Zugriff	Beschreibung
LEVEL	float	lesend schreibend	Dimmer + Jalousieaktor → **CMD_RUNL** negative Werte werden invertiert und direkt als $VALUE$ übergeben.
OLD_LEVEL	action	schreibend	Dimmer → **CMD_RUNS**
STOP	action	schreibend	Jalousieaktor → **CMD_RUNS**
STATE	boolean	lesend schreibend	Schalter: Beim Schaltzustand *false* (Aus) wird intern ein kurzer Tastendruck **CMD_RUNS** und beim Schaltzustand *true* (Ein) ein langer Tastendruck **CMD_RUNL** ausgeführt.
CMD_RUNS	action	schreibend	Taster: kurzer Tastendruck (Befehl **CMD_SHORT** ausführen, Rückgabe des *system()* Exit-Status im DP **CMD_RETS**)
CMD_RUNL	action	schreibend	Taster: langer Tastendruck (Befehl **CMD_LONG** ausführen, Rückgabe des *system()* Exit-Status im DP **CMD_RETL**)
CMD_SETS	string	lesend schreibend	Befehlszeile setzen (kurzer Tastendruck) – entspricht dem Geräteparameter **CMD_SHORT**
CMD_SETL	string	lesend schreibend	Befehlszeile setzen (langer Tastendruck) – entspricht dem Geräteparameter **CMD_LONG**
CMD_RETS	string	lesend	Befehl ausführen (kurzer Tastendruck) mit Rückgabe von STDOUT, *popen()*. Es werden alle '<' und '>'-Zeichen durch Leerzeichen ersetzt.
CMD_RETL	string	lesend	Befehl ausführen (langer Tastendruck) mit Rückgabe von STDOUT, *popen()*. Es werden alle '<' und '>'-Zeichen durch Leerzeichen ersetzt.
CMD_QUERY_RET	action	schreibend	Abfrage von **CMD_RETS** und **CMD_RETL** des Gerätes für die folgenden 10 Sekunden aktivieren
CMD_EXEC	string	schreibend	übergebenen Befehl sofort ausführen ohne Rückgabewerte, *system()*-Aufruf
LOGIT	string	schreibend	String „*Name*;*Wert*“ mit **DEVTIMEFORMAT** und **DEVDATAFORMAT** in **DEVLOGFILE** schreiben.
SYSLOG	string	schreibend	INFO-Meldung ins Syslog der CCU schreiben
WORKING	boolean	lesend	Abarbeitung von **CMD_RUNS** bzw. **CMD_RUNL**
CONTROL	integer	lesend	konfiguriertes Control-Element: 0..Taster (**KEY**), 1..Schalter (**SWITCH**), 2..Jalousie (**BLIND**), 3..Dimmer (**DIMMER**)
SET_STATE	float	schreibend	**LEVEL** bzw. **STATE** Datenpunkt setzen, ohne eine Aktion auszuführen. Dabei werden **negative LEVEL**-Werte invertiert und als Ausgabewert anhand von **MAX_VAL** in den entsprechenden %-Wert rückgerechnet. Beispiel: MAX_VAL = 200 SET_STATE(**0.6**) → LEVEL= 0.6 → **60%** SET_STATE(**-100**) → LEVEL= 0.5 → **50%**
RAND	string	lesend schreibend	Integer Zufallszahl 0 <= **RAND** <= *max* erzeugen. *max* kann auf den Datenpunkt geschrieben werden und zwischen 1 und 2147483647 liegen. Per Default ist max auf 65535 gesetzt. Nach Änderung bleibt der Maximalwert dauerhaft für diesen Kanal (auch nach einem Neustart) gesetzt.

Optionale Datenpunkte (siehe SYSTEM|PARAMETER)			
PARAMETER_S_1	string	schreibend	1. Parameter **1** (kurzer Tastendruck o. Aus)
PARAMETER_L_1	string	schreibend	1. Parameter **1** (langer Tastendruck o. Ein)
...			
PARAMETER_S_99	string	schreibend	99. Parameter **99** (kurzer Tastendruck o. Aus)
PARAMETER_L_99	string	schreibend	99. Parameter **99** (langer Tastendruck o. Ein)

Bei jedem Befehlsaufruf werden zusätzliche **Umgebungsvariablen** gesetzt:

CUXD_CHANNEL aufgerufener Kanal des System.Exec-Gerätes: CUX2801xxx:x

CUXD_VALUE kurzer (0) oder langer (1) Tastendruck bzw. Ein (1), Aus (0) Zustand bzw. errechneter LEVEL beim Jalousie- und Dimm-Aktor

CUXD_OLDVALUE VALUE-Wert vom letzten Aufruf

CUXD_MAXVALUE Wert des **MAX_VAL** Geräteparameters zur VALUE-Berechnung

In der **Kommandozeile** können dabei folgende Platzhalter genutzt werden:

$CHANNEL$ entspricht ***CUXD_CHANNEL***

1...99 Inhalt der entsprechenden Parameter-Datenpunkte (URL-Encoded)

$_1$...$_99$ Inhalt der entsprechenden Parameter-Datenpunkte (wie eingegeben)

$P1$...$P9$ Inhalt der Geräteparameter (URL-Encoded)

$_P1$...$_P9$ Inhalt der Geräteparameter (wie eingegeben)

$C1$...$C5$ Inhalt der Kanalparameter (URL-Encoded)

$_C1$...$_C5$ Inhalt der Kanalparameter (wie eingegeben)

$TS**<*format*>**$ Zufallszahl/-zeit im **TIMER_SET**-Format definieren

$VALUE$ entspricht ***CUXD_VALUE***

$OLDVALUE$ entspricht ***CUXD_OLDVALUE***

$MAXVALUE$ entspricht ***CUXD_MAXVALUE***

HM-Script mit Rückgabe von STDOUT (Beispiel):

```
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("ping -c 5 192.168.1.1");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var v = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
WriteLine(v);
```

Es ist zu beachten, dass die Verarbeitung des HM-Scripts erst fortgesetzt wird, nachdem das aufgerufene Programm beendet wurde. Während dieser Zeit werden auch keine anderen HM-Scripts ausgeführt!

HM-Script ohne Rückgabe von STDOUT (Beispiel):

```
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -q -O /dev/null
'http://192.168.0.99:50000/track=neue_email.mp3'");
dom.GetObject("CUxD.CUX2801001:1.CMD_RUNS").State(1);
```

Die Verarbeitung des HM-Scripts wird sofort und unabhängig von der Laufzeit des aufgerufenen Programms fortgesetzt!

3 Zufallszahlen im Bereich von 0 bis 255 erzeugen:

```
dom.GetObject("CUxD.CUX2801001:1.RAND").State(255);
integer rand1 = dom.GetObject("CUxD.CUX2801001:1.RAND").State().ToInteger();
integer rand2 = dom.GetObject("CUxD.CUX2801001:1.RAND").State().ToInteger();
integer rand3 = dom.GetObject("CUxD.CUX2801001:1.RAND").State().ToInteger();
```

Beispiel für die Nutzung von Parameter-Datenpunkten ohne Rückgabe von STDOUT:

```
dom.GetObject("CUxD.CUX2801001:1.CMD_SETL").State("wget -q -O /dev/null
'http://192.168.0.5/info?text=$1$&priority=$2$'");
dom.GetObject("CUxD.CUX2801001:1.PARAMETER_L_1").State("Wasser im Büro!");
dom.GetObject("CUxD.CUX2801001:1.PARAMETER_L_2").State("5");
dom.GetObject("CUxD.CUX2801001:1.CMD_RUNL").State(1);
```

Ändert sich die Befehlszeile nicht, dann reicht es, **CMD_SETS** **einmalig** per Script oder über die Gerätekonfiguration (**CMD_SHORT**) zu setzen.
Im Script ist danach nur noch der Aufruf von:

```
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var v = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
WriteLine(v);
```

bzw.

```
dom.GetObject("CUxD.CUX2801001:1.CMD_RUNS").State(1);
```

notwendig. Der zuletzt genannte Befehl wird auch beim Auslösen eines kurzen Tastendruckes ausgeführt.

Soll der Befehl nicht in der Konfiguration gespeichert werden, dann besteht auch die Möglichkeit ihn mittels **CMD_EXEC** sofort auszuführen. Dabei werden die gespeicherten Einstellungen nicht verändert:

```
dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("/usr/local/geturl.sh 42");
```

Die Verarbeitung des HM-Scripts wird sofort und unabhängig von der Laufzeit des aufgerufenen Programms fortgesetzt!

Einfaches Beispiel zum Ersatz von undokumentierten system.Exec()-Aufrufen:

vorher:

```
string stderr;
string stdout;
string url="'http://192.168.0.1/web/message?text=Hello_World&type=3&tmout=10'";
system.Exec("wget -q -O - "#url, &stdout, &stderr);
```

nachher:

- **zuerst muss für dieses Beispiel im CUxD ein „(28) System-Exec“ Gerät mit der Seriennummer 1 angelegt werden, dann...**

```
string url="'http://192.168.0.1/web/message?text=Hello_World&type=3&tmout=10'";
dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#url);
```

Soll nach einem Befehl mit längerer Laufzeit eine weitere Aktion ausgeführt werden, dann würde bei Anwendung des **CMD_RETS** bzw. **CMD_RETL** Datenpunktes zum Programmaufruf die CCU für die gesamte Programmlaufzeit blockieren.

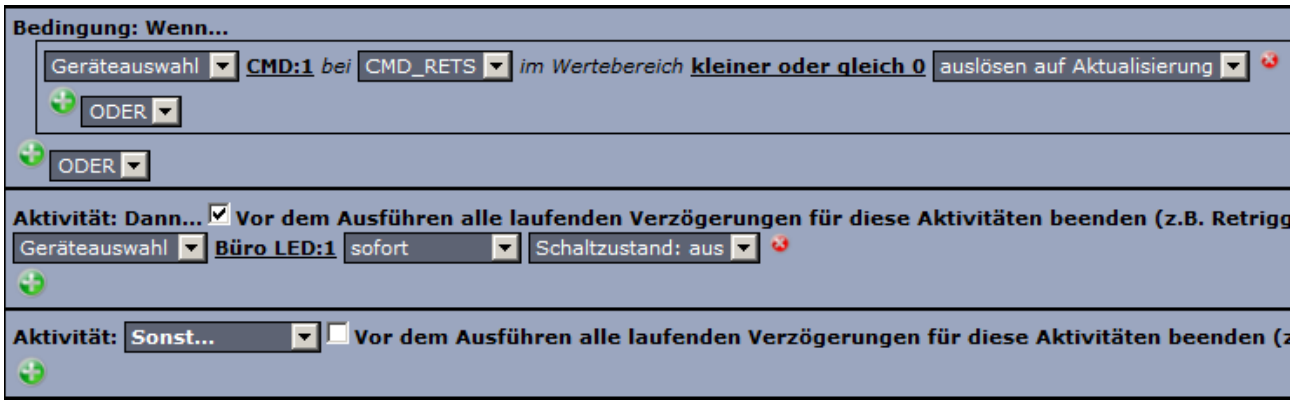
Aus diesem Grund gibt es die Möglichkeit, den Exit-Code eines per **CMD_RUNS** bzw. **CMD_RUNL** aufgerufenen Befehls als Ereignis in einer Programmverknüpfung auszuwerten und so eine Aktion nach Beendigung des Befehls auszuführen, ohne dabei andere Programme für die gesamte Laufzeit dieses Befehls zu blockieren.

Der Aufruf des Befehls erfolgt entweder per HM-Script:

```
dom.GetObject("CUxD.CUX2801001:1.CMD_RUNS").State(1);
```

oder direkt per kurzem Tastendruck in der WebUI.

Nach der Abarbeitung wird der Wert des Exit-Codes im Datenpunkt **CMD_RETS** an die CCU Logikschicht gesendet und kann asynchron mittels einer einfachen Programm-verknüpfung abgefragt werden:



Die folgenden beiden Befehle bewirken dasselbe:

```
dom.GetObject("CUxD.CUX2801001:1.STATE").State(0);
dom.GetObject("CUxD.CUX2801001:1.CMD_RUNS").State(1);
```

Und auch diese beiden Befehle bewirken dasselbe:

```
dom.GetObject("CUxD.CUX2801001:1.STATE").State(1);
dom.GetObject("CUxD.CUX2801001:1.CMD_RUNL").State(1);
```

Aufruf von **CMD_EXEC** in einer Programmverknüpfung ohne HM-Script:

Name	Beschreibung	Bedingung (Wenn...)	Aktivität (Dann..., Sonst...)	Aktion
Backup CCU	Backup CCU	Zeit: Wöchentlich um 02:00 Uhr beginnend am 06.10.2012 auslösen zu Zeitpunkten	Kanalauswahl: CMD:1 sofort CMD_EXEC auf /usr/local/addons/cuxd/extra/ccu_backup /home/backup	system intern

Die Aktualisierung vom **LEVEL** bzw. **STATE** Datenpunkt eines Gerätes (z.B. Rückmeldung eines gesteuerten Gerätes) kann durch den Aufruf einer URL mittels **SET_STATE** erfolgen. Dabei wird nur der gespeicherte Wert aktualisiert!

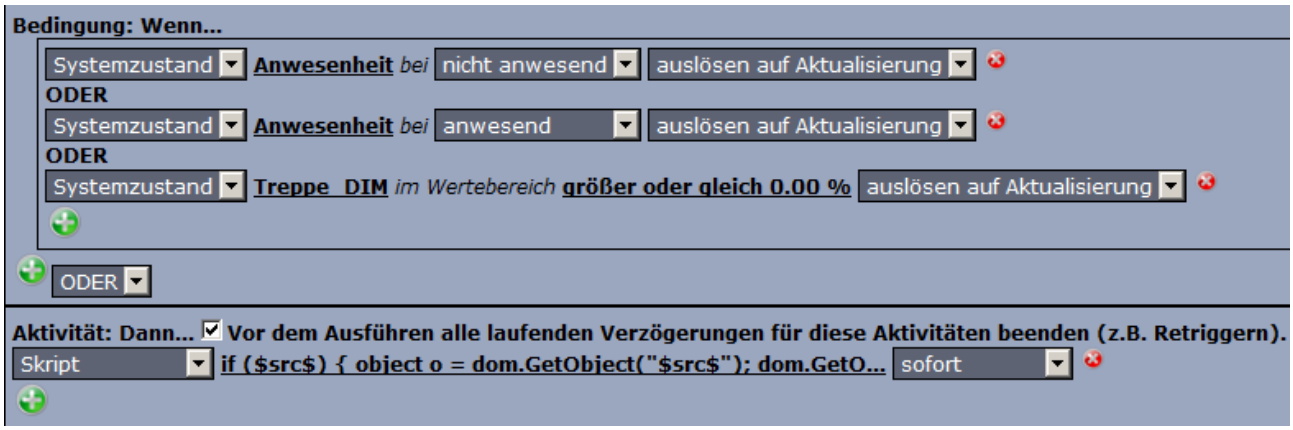
Aktualisierung mit 100%-Wert (von 0.000 bis 1.000):

```
http://IP/cuxd.exe?x=dom.GetObject("CUxD.CUX2801001:2.SET_STATE").State(0.66);
```

bzw. mit dem absoluten Ausgabewert (von 0 bis MAX_VAL) als negativer Parameter:

```
http://IP/cuxd.exe?x=dom.GetObject("CuxD.CUX2801001:2.SET_STATE").State(-153);
```

Beispiel zum Loggen von Systemvariablen mit den CUxD-Einstellungen im **DEVLOGFILE**:



Script (Logging bei Aktualisierung):

```
object o = dom.GetObject("$src$");
if (o) {
  dom.GetObject("CUxD.CUX2801001:1.LOGIT").State(o.Name()#";"#o.Value());
}
```

Script (Logging bei Änderung):

```
object o = dom.GetObject("$src$");
if (o) {
  if (o.Value() <> o.LastValue()) {
    dom.GetObject("CUxD.CUX2801001:1.LOGIT").State(o.Name()#";"#o.Value());
  }
}
```

5.8.3 System.Multi-Dim-Exec (1 + 16 Kanäle)

Mit diesem Gerät können z.B. RGB-Dimmer, deren Werte mittels Aufruf einer frei konfigurierbaren Kommandozeile (z.B. über den Aufruf eines Tools oder einer URL) gesetzt werden können, in die WebUI integriert werden.

Der konfigurierte Befehl wird mittels einer Taste oder auch bei Änderung der Dim-Werte mittels C system() Befehl auf der CCU ausgeführt. So besteht die Möglichkeit, zuerst die Helligkeitswerte mehrerer Kanäle eines Gerätes zu setzen, um sie danach vollständig an den/die Dimmer zu übertragen.

Um die Kommandozeile so variabel wie möglich zu gestalten, existieren für die Helligkeits- und Maximal-Werte Platzhalter, die beim Aufruf durch aktuelle Werte ersetzt werden.

Konfigurationsparameter:

Parameter	
CHANNELS	3 (1-16)
CMD_EXEC	/usr/local/addons/extra/

CHANNELS - Anzahl der steuerbaren Dimmer-Kanäle (maximal 16). Sollte die Darstellung nicht aktualisiert werden, dann hilft ein Reload im Webbrowser.

CMD_EXEC - Befehlszeile
Für die Dim-Level können die Platzhalter **1**, **2**, …, **16** verwendet werden. Über die Parameter **MAX_VAL** (Platzhalter **$m1$**, **$m2$**, … **$m16$**) ist deren Wertebereich pro Kanal frei definierbar. Die maximale Laufzeit des aufgerufenen Scripts kann 180s betragen.

*Werden im **CMD_EXEC**-Parameter Anführungs- oder Sonderzeichen (z.B. '=') verwendet, dann kann es im Eingabefeld der WebUI Probleme geben!*

Name	Kanal	Parameter
RGB-DIM:1	Ch.: 1	Keine Parameter einstellbar
R-Dim:2	Ch.: 2	DIMMER|EXEC_ON_CHANGE ☑ DIMMER|MAX_VAL 255 (1-65535)
G-DIM:3	Ch.: 3	DIMMER|EXEC_ON_CHANGE ☑ DIMMER|MAX_VAL 255 (1-65535)
B-DIM:4	Ch.: 4	DIMMER|EXEC_ON_CHANGE ☑

EXEC_ON_CHANGE - ausführen von **CMD_EXEC** sofort bei Level-Änderung des Kanals.

MAX_VAL - Maximalwert (1 bis 65535) bei Level=100% des Dimmer-Kanals. Zum Beispiel werden bei **MAX_VAL**=255 die %-Werte in Werte zwischen 0 und 255 umgerechnet.

Beispiel für **CMD_EXEC** zur Ansteuerung von DMX-Kanal 1 bis 3 über artdmxdim mit Soft-Dim (1s) bei Wert-Änderungen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.168.0.90 0 7 1000 1:$1$ 2:$2$ 3:$3$
```

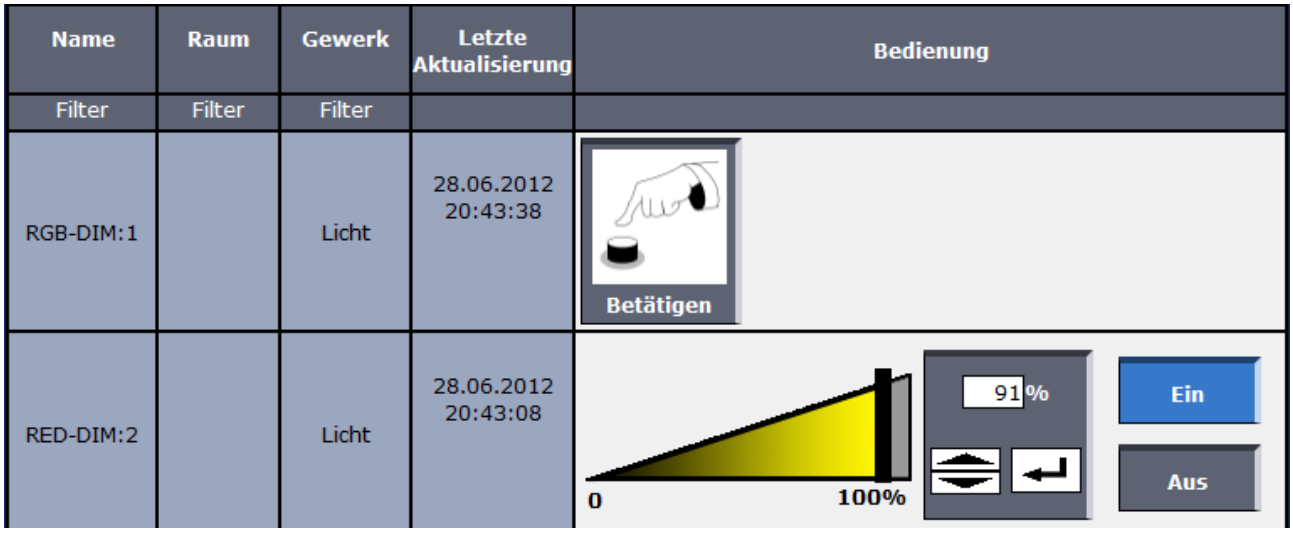
Bei Aufteilung auf mehrere CUxD-Geräte können alle 512 DMX-Kanäle eines DMX-Universums angesteuert werden.

Beispiel für **CMD_EXEC** zur Ansteuerung von DMX-Kanal 16 bis 18 über artdmxdim

```
/usr/local/addons/cuxd/extra/artdmxdim 192.168.0.90 0 7 0 16:$1$ 17:$2$ 18:$3$
```

Kanaltypen:

Kanaltyp	Kanalnummer
SYSTEM	1
DIMMER	2...17



Kanaltyp SYSTEM:

DP-Name	Typ	Zugriff	Beschreibung
STATE	action	schreibend event	Tastendruck (**CMD_EXEC**-Befehl ausführen) Nachdem die Abarbeitung des **CMD_EXEC**-Befehls beendet wurde, wird ein Event ausgelöst.
HOLD	action	schreibend	deaktiviert **EXEC_ON_CHANGE** bis zum nächsten **STATE**-Aufruf

Kanaltyp DIMMER:

DP-Name	Typ	Zugriff	Beschreibung
LEVEL	float	lesend schreibend	Dimmwert des Aktors (Helligkeitswert) abhängig vom Parameter **EXEC_ON_CHANGE** wird die **CMD_EXEC**-Befehlszeile bei Änderung sofort ausgeführt. Negative Werte werden invertiert und direkt als ***CUXD_VALUE****xx* gesetzt.
OLD_LEVEL	action	schreibend	Letzten Dimmwert des Aktors wiederherstellen
SET_STATE	float	schreibend	**LEVEL** Datenpunkt setzen, ohne eine Aktion auszuführen. Dabei werden **negative LEVEL**-Werte invertiert und als Ausgabewert anhand von **MAX_VAL** in den entsprechenden %-Wert rückgerechnet. Beispiel: MAX_VAL = 300 SET_STATE(**0.6**) → LEVEL= 0.6 → **60%** SET_STATE(**-90**) → LEVEL= 0.3 → **30%**

Bei jedem Befehlsaufruf werden folgende zusätzliche Umgebungsvariablen gesetzt:

CUXD_DEVICE aktuelles CUxD-Gerät: CUX2802xxx

CUXD_TRIGGER_CH auslösender Kanal des CUxD-Gerätes:
1 – Tastendruck (STATE)
2..n+1 – Dimmwert-Änderung (LEVEL)

CUXD_VALUE2 VALUE des 1. Dimmer-Kanals

...

CUXD_VALUE*n+1* VALUE des n-ten Kanals

CUXD_MAXVALUE2 MAXVALUE des 1. Dimmer-Kanals

...

CUXD_MAXVALUE*n+1* MAXVALUE des n-ten Kanals

Beispiel zum Setzen aller RGB-Werte eines Gerätes auf (0%,0%,0%) ohne Flackern per HM-Script:

```
dom.GetObject("CUxD.CUX2802001:1.HOLD").State(1);
dom.GetObject("CUxD.CUX2802001:2.LEVEL").State(0);
dom.GetObject("CUxD.CUX2802001:3.LEVEL").State(0);
dom.GetObject("CUxD.CUX2802001:4.LEVEL").State(0);
dom.GetObject("CUxD.CUX2802001:1.STATE").State(1);
```

Die Aktualisierung vom **LEVEL** Datenpunkt eines Gerätes (z.B. Rückmeldung eines gesteuerten Gerätes) kann durch den Aufruf einer URL mittels **SET_STATE** erfolgen. Dabei wird im Gegensatz zur Aktualisierung durch den **LEVEL**-Datenpunkt nur der gespeicherte Wert aktualisiert und keine Befehlszeile ausgeführt!

Aktualisierung mit 100%-Wert (von 0.000 bis 1.000):

```
http://IP/cuxd.exe?x=dom.GetObject("CUxD.CUX2802001:3.SET_STATE").State(0.77);
```

bzw. mit dem absoluten Ausgabewert (von 0 bis MAX_VAL) als negativer Parameter:

```
http://IP/cuxd.exe?x=dom.GetObject("CuxD.CUX2802001:3.SET_STATE").State(-196);
```

5.8.4 System.Ping (16 Kanäle)

Dieses Gerät dient zum Prüfen der Erreichbarkeit von maximal 16 verschiedenen Hosts anhand von ICMP-Paketen (Ping) oder Verbindungsversuchen auf konfigurierte TCP-Ports. Anhand der TCP-Ports kann man beliebige Server-Dienste überwachen. Sollen mehr als 16 Hosts überwacht werden, dann ist es möglich, im CUxD mehrere dieser Geräte anzulegen.

Alle Geräte arbeiteten bidirektional, d.H. neben dem Aussenden von Netzwerk-Ping's werden auch ankommende ICMP-Ping's von den konfigurierten Adressen verarbeitet und starten den Timer für das jeweilige Sende-Intervall neu.

Wenn man z.B. das Intervall auf 90s setzt, und alle 60s vom konfigurierten Host ein Ping an die CCU sendet, dann erkennt die CCU den Host als ALIVE, ohne jemals einen Ping zu dem Host zu senden.

Zusätzlich können die Sende-Intervalle bei Erreichbarkeit und Nicht-Erreichbarkeit getrennt voneinander konfiguriert werden, um z.B. Strom beim Anpingen eines Smartphones mit aktiviertem WLAN zu sparen.

Konfigurationsparameter:

Kanal	Parameter		
Ch.: 1	SWITCH|ACTIVE	☑	
	SWITCH|IP_DNS_ADR	fritz.box	
	SWITCH|PORT	80	(0-65535)
	SWITCH|INTERVAL_ALIVE	60	s (15-3600)
	SWITCH|INTERVAL_FAIL	60	s (15-3600)
	SWITCH|MAX_RETRY	0	(0-5)
	SWITCH|THRESHOLD	1	(1-255)
	SWITCH|CMD_EXEC_TRUE	/usr/local/addons/cuxd/e	
	SWITCH|CMD_EXEC_FALSE	/usr/local/addons/cuxd/e	

ACTIVE - [x] Kanal ist Aktiv. Deaktivierte Kanäle werden in der WebUI ausgeblendet. Sollte die Darstellung nicht aktualisiert werden, dann hilft ein Reload vom Webbrowser.

IP_DNS_ADR - IP- bzw. DNS-Adresse des Hosts

PORT - Ist dieser Parameter 0, dann wird ein ICMP ECHO REQUEST zum konfigurierten Host (**IP_DNS_ADR**) gesendet. Ansonsten wird versucht, einen TCP-Connect auf den angegebenen Port auszuführen. Auf diese Weise kann z.B. ein Netzwerk-Dienst (HTTP, FTP, usw.) auf dem Zielhost überwacht werden.

INTERVAL_ALIVE - Ping Intervall nach Erreichbarkeit des Hosts in Sekunden

INTERVAL_FAIL - Ping Intervall nach Nicht-Erreichbarkeit des Hosts in Sekunden

MAX_RETRY - Maximale Anzahl der Pings, die unmittelbar nach Erkennen der Nicht-erreichbarkeit des Hosts ausgesendet werden, bevor der Status an die CCU gemeldet wird.

THRESHOLD - erst nach der konfigurierten Anzahl an fehlerhaften Pings wird der Gerätestatus **verzögert** auf **FALSE** gesetzt. Die **MAX_RETRY**-Pings werden nicht mitgezählt.

CMD_EXEC_FALSE - wenn vorhanden, dann wird die Befehlszeile bei Änderung vom Gerätestatus von **TRUE** auf **FALSE** aufgerufen.

CMD_EXEC_TRUE - wenn vorhanden, dann wird die Befehlszeile bei Änderung vom Gerätestatus von **FALSE** auf **TRUE** aufgerufen.

Bei jedem Befehlsaufruf werden zusätzliche **Umgebungsvariablen** gesetzt:

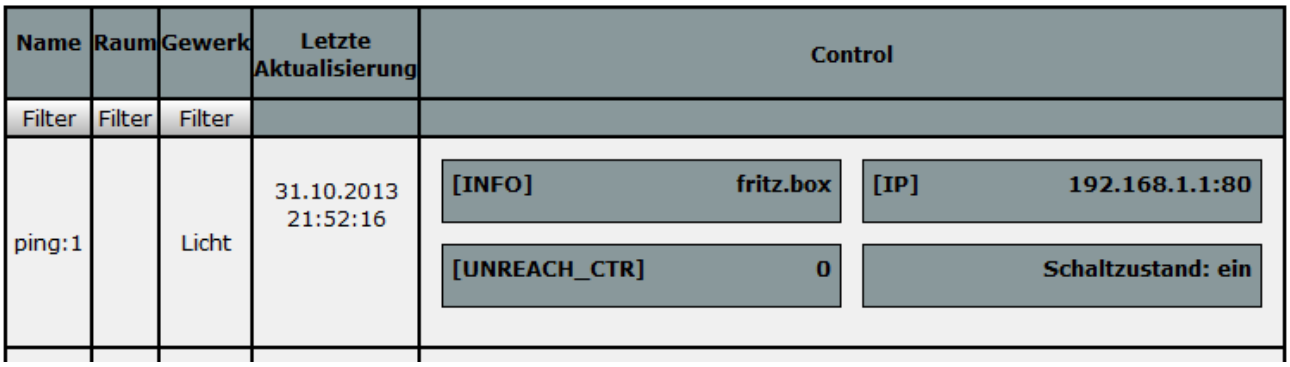
CUXD_CHANNEL auslösender Kanal des System.Ping-Gerätes: CUX2803xxx:x

CUXD_VALUE aktueller Wert von **UNREACH_CTR**

In der **Kommandozeile** können dabei folgende Platzhalter genutzt werden:

$CHANNEL$ entspricht ***CUXD_CHANNEL***

$VALUE$ entspricht ***CUXD_VALUE***



Kanaltypen:

Kanaltyp	Kanalnummer
SWITCH	1..16

Kanaltyp SWITCH:

DP-Name	Typ	Zugriff	Beschreibung
INFO	string	lesend	eingetragene **IP_DNS_ADR**
IP	string	lesend	*IP-Adresse* des Hosts nach DNS-Auflösung. Ist ein Port konfiguriert, dann steht in diesem Datenpunkt *IP-Adresse:Port*
UNREACH_CTR	integer	lesend	Zähler für Fehlversuche (0..255) Die **MAX_RETRY**-Pings werden nicht mitgezählt
STATE	boolean	lesend	Aktueller Status des letzten Ping's wird nur bei Änderung aktualisiert

5.9 (91) CloudMatic …

Diese Geräte wurden in Kooperation mit der Easy SmartHome GmbH implementiert und um den Zugriff auf die CloudMatic-Dienste von meine-homematic.de zu vereinfachen. Dafür wurden die CloudMatic-Funktionen als virtuelle Geräte auf der CCU abgebildet.

Die Schnittstellenanpassung zwischen den virtuellen CUxD-Geräten und den CloudMatic-Funktionen erfolgt über das Programm /etc/config/addons/mh/cloudmatic. Es ist per Default im CUxD-Parameter **CLOUDMATIC_CMD=** gesetzt und wird beim Hinzufügen neuer Geräte (CloudMatic.Mail, CloudMatic.SMS, CloudMatic.Push, CloudMatic.Cloud) im Geräteparameter **CMD_EXEC** abgespeichert. Diese Parameter können angepasst werden und z.B. auf eigene Scripts verweisen.

Sollen jedoch CloudMatic-Funkionen genutzt werden, dann ist zuvor der CloudMatic-Dienst auf der CCU zu installieren. Nach der Installation sollte eine gültige CloudMatic-ID in der Datei /etc/config/addons/mh/cmid vorhanden sein und der Geräteparameter **CMD_EXEC** muss auf das Programm /etc/config/addons/mh/cloudmatic verweisen.

Der CUxD übergibt bei jedem Aufruf vom **SEND**-Datenpunkt der Geräte CloudMatic.Mail, CloudMatic.SMS und CloudMatic.Push alle gesetzten Datenpunkte über die Befehlszeile an das im Geräteparameter **CMD_EXEC** konfigurierte Programm. Dieses Programm bereitet die Daten dann entsprechend auf und führt die eigentliche Funktion aus.

Der Aufruf des unter **CMD_EXEC** definierten Scripts erfolgt mit folgenden Parametern:

<**cmd**> <*device-type*> *itp*=<**I**D,**T**YPE,**P**RIO> *p1*=<...> … *p4*=<...> *o1*=<...> … *o5*=<...>

device-type	= **1**..Mail, **2**..SMS, **3**..Push, **4**..Cloud
itp	= Template**I**D, **T**ype, **P**riority
p1..4	= Parameter 1..4 (abhängig vom Gerät)
o1..5	= Option 1..5 (optional)

Die Werte der Übergabeparameter sind URL-Encoded. Leere Parameter werden nicht übergeben.

5.9.1 CloudMatic.Mail

Kanal	Parameter		
Ch.: 1	SYSTEM|P_MAILTO		
	SYSTEM|P_MAILCC		
	SYSTEM|P_SUBJECT		
	SYSTEM|P_TYPE	STANDARD	
	SYSTEM|P_TEXT		
	SYSTEM|P_TEMPLATEID	0	(0-99)
	SYSTEM|P_OPTION_1		
	SYSTEM|P_OPTION_2		
	SYSTEM|P_OPTION_3		
	SYSTEM|P_OPTION_4		
	SYSTEM|P_OPTION_5		
	SYSTEM|CMD_EXEC	/etc/config/addons/mh/c	

Kanaltypen:

Kanaltyp	Kanalnummer
SYSTEM	1

Kanaltyp SYSTEM:

DP-Name	Typ	Zugriff	Beschreibung
SEND	action	schreibend	Tastendruck auf WebUI
MAILTO	string	schreibend	Email-Empfänger
MAILCC	string	schreibend	Email-CC
SUBJECT	string	schreibend	Email-Betreff
TYPE	enum	schreibend	(0..STANDARD, 1..HTML)
TEXT	string	schreibend	Email-Text (bei Template-ID = 0)
TEMPLATEID	integer	schreibend	0 oder ID des vordefinierten Templates
OPTION_1	string	schreibend	Daten für vordefiniertes Template
OPTION_2	string	schreibend	Daten für vordefiniertes Template
OPTION_3	string	schreibend	Daten für vordefiniertes Template
OPTION_4	string	schreibend	Daten für vordefiniertes Template
OPTION_5	string	schreibend	Daten für vordefiniertes Template
RETURN	integer	lesend	Rückgabewert nach Funktionsaufruf
WORKING	boolean	lesend	Daten werden übertragen

5.9.2 CloudMatic.SMS

Kanal	Parameter		
Ch.: 1	SYSTEM|P_PHONE	01711234567	
	SYSTEM|P_TYPE	STANDARD	
	SYSTEM|P_TEXT	Hier ist die CCU	
	SYSTEM|P_TEMPLATEID	0	(0-99)
	SYSTEM|P_OPTION_1		
	SYSTEM|P_OPTION_2		
	SYSTEM|P_OPTION_3		
	SYSTEM|P_OPTION_4		
	SYSTEM|P_OPTION_5		
	SYSTEM|CMD_EXEC	/etc/config/addons/mh/c	

Kanaltypen:

Kanaltyp	Kanalnummer
SYSTEM	1

Kanaltyp SYSTEM:

DP-Name	Typ	Zugriff	Beschreibung
SEND	action	schreibend	Tastendruck auf WebUI
PHONE	string	schreibend	SMS-Telefonnummern
TYPE	enum	schreibend	(0..STANDARD, 1..PREMIUM, 2..ALARM)
TEXT	string	schreibend	SMS-Text (bei Template-ID = 0)
TEMPLATEID	integer	schreibend	0 oder ID des vordefinierten Templates
OPTION_1	string	schreibend	Daten für vordefiniertes Template
OPTION_2	string	schreibend	Daten für vordefiniertes Template
OPTION_3	string	schreibend	Daten für vordefiniertes Template
OPTION_4	string	schreibend	Daten für vordefiniertes Template
OPTION_5	string	schreibend	Daten für vordefiniertes Template
RETURN	integer	lesend	Rückgabewert nach Funktionsaufruf
WORKING	boolean	lesend	Daten werden übertragen

5.9.3 CloudMatic.Push

Kanal	Parameter		
Ch.: 1	SYSTEM|P_PRIORITY	NORMAL	
	SYSTEM|P_TYPE	ANDROID	
	SYSTEM|P_APIKEY		
	SYSTEM|P_TEXT		
	SYSTEM|P_TEMPLATEID	0	(0-99)
	SYSTEM|P_OPTION_1		
	SYSTEM|P_OPTION_2		
	SYSTEM|P_OPTION_3		
	SYSTEM|P_OPTION_4		
	SYSTEM|P_OPTION_5		
	SYSTEM|CMD_EXEC	/etc/config/addons/mh/c	

Kanaltypen:

Kanaltyp	Kanalnummer
SYSTEM	1

Kanaltyp SYSTEM:

DP-Name	Typ	Zugriff	Beschreibung
SEND	action	schreibend	Tastendruck auf WebUI
PRIORITY	enum	schreibend	(0..LOW, 1..NORMAL, 2..IMPORTANT, 3..HIGH, 4..CRITICAL)
TYPE	enum	schreibend	(0..GROWL, 1..ANDROID, 2..TEST)
APIKEY	string	schreibend	
TEXT	string	schreibend	Nachrichtentext (bei Template-ID = 0)
TEMPLATEID	integer	schreibend	0 oder ID des vordefinierten Templates
OPTION_1	string	schreibend	Daten für vordefiniertes Template
OPTION_2	string	schreibend	Daten für vordefiniertes Template
OPTION_3	string	schreibend	Daten für vordefiniertes Template
OPTION_4	string	schreibend	Daten für vordefiniertes Template
OPTION_5	string	schreibend	Daten für vordefiniertes Template
RETURN	integer	lesend	Rückgabewert nach Funktionsaufruf
WORKING	boolean	lesend	Daten werden übertragen

5.9.4 CloudMatic.Cloud

Parameter		
CHANNELS	1	(1-16)
CMD_EXEC	/etc/config/addons/mh/c	

Kanal	Parameter		
Ch.: 1	SYSTEM|P_TYPE	0	(0-99)
	SYSTEM|P_TEMPLATEID	0	(0-99)
	SYSTEM|P_OPTION_1		
	SYSTEM|P_OPTION_2		
	SYSTEM|P_OPTION_3		
	SYSTEM|P_OPTION_4		
	SYSTEM|P_OPTION_5		
	SYSTEM|P_TYPE	0	(0-99)
	SYSTEM|P_TEMPLATEID	0	(0-99)

Kanaltypen:

Kanaltyp	Kanalnummer
SYSTEM	1..16

Kanaltyp SYSTEM:

DP-Name	Typ	Zugriff	Beschreibung
SEND	action	schreibend	Tastendruck auf WebUI
TYPE	integer	schreibend	
TEMPLATEID	integer	schreibend	ID des vordefinierten Templates
OPTION_1	string	schreibend	Daten für vordefiniertes Template
OPTION_2	string	schreibend	Daten für vordefiniertes Template
OPTION_3	string	schreibend	Daten für vordefiniertes Template
OPTION_4	string	schreibend	Daten für vordefiniertes Template
OPTION_5	string	schreibend	Daten für vordefiniertes Template
RETURN	integer	lesend	Rückgabewert nach Funktionsaufruf
WORKING	boolean	lesend	Daten werden übertragen

5.9.5 CloudMatic.Webcam

Über die folgenden CUxD-Konfigurationsparameter kann die Webcam-Konfiguration auf der CCU angepasst werden. Dieses Gerät funktioniert auch ohne CloudMatic-Anmeldung.

Die Parameter haben nach der Installation Default-Werte:

WEBCAMCONFIG=/usr/local/addons/cuxd/webcamconfig.ini

- In diesem Parameter steht das Webcam-Konfigurationsfile
- Die Suche nach einem konfigurierten <TYP> beginnt zuerst in diesem und erst danach im ausgelieferten /usr/local/addons/cuxd/webcamconfig-default.ini File.
- Mit jedem CUxD-Update wird webcamconfig-default.ini überschrieben. Also sollten eigene Anpassungen nur im webcamconfig.ini File vorgenommen werden!

WEBCAMSNAPSHOT=/tmp/snapshots

- In diesem Parameter steht das Wurzelverzeichnis für die gespeicherten Webcam-Snapshots. Die Snapshots werden in einem Unterverzeichnis mit der Seriennummer des CUxD-Gerätes abgelegt. Existiert das Verzeichnis nicht, dann wird es automatisch angelegt.

WEBCAMCACHE=/tmp/webcams

- In diesem Parameter steht das Wurzelverzeichnis für die zwischengespeicherten Webcam-Snapshots im ***CACHE***-Mode (nur die zwischengespeicherten Bilder). Es sollte auf der RAM-Disk liegen! Die Snapshots werden in einem Unterverzeichnis mit der Seriennummer des CUxD-Gerätes abgelegt. Existiert das Verzeichnis nicht, dann wird es automatisch angelegt.

Im **WEBCAMCONFIG**-File können URLs für verschiedene Webcam-Modelle (<TYP>) vordefiniert werden. Über den **TYPE**-Parameter können diese URLs dann in der Gerätekonfiguration des CUxD-Gerätes eingetragen werden.

Die Zeilen haben den folgenden Aufbau (als Trennung der Werte dienen ein oder mehrere Leerzeichen!)

<TYP> <Parameter> <Nummer> <Daten>

Beginnt die Zeile mit 0 als <TYP>, dann wird sie als Kommentarzeile ignoriert!

Parameter:

TYP	Par	Num	Beschreibung
1..99	info	0	Name der Webcam für **INFO**-DP
1..99	img	0	Snapshot **URL** für **IMAGE**-DP
1..99	img	1	Stream **URL** für **IMAGE**-DP
1..99	img	2	Snapshot **URL** für **SEND**-DP und **CACHE**-Mode
1..99	ptz	0..99	**URL** für PTZ-Steuerung oder sonstige Funktionen
1..99	ir	0	**URL** für IR Beleuchtung OFF
1..99	ir	1	**URL** für IR Beleuchtung AUTO
1..99	ir	2	**URL** für IR Beleuchtung ON
1..99	sw	0	**URL** zum Ausschalten des Alarmkontaktes der Webcam
1..99	sw	1	**URL** zum Einschalten des Alarmkontaktes der Webcam
1..99	reset	0	Reboot **URL**

In den **URL**s werden die Platzhalter <ip>, <port>, <user>, <pass> und <opt1>...<opt5> durch den Inhalt der konfigurierten Geräteparameter und <ts> durch den aktuellen Unix-Zeitstempel ersetzt.

Mehrere URLs können entweder durch Leerzeichen getrennt in einer Zeile oder als Wiederholung des gleichen Parameters in mehreren aufeinanderfolgenden Zeilen eingetragen werden. Sie werden dann nacheinander in der konfigurierten Reihenfolge aufgerufen.

Sind ganze Blöcke von Parametern (z.B. „ir“, „ptz“ oder „sw“) nicht vorhanden, so werden die entsprechenden Datenpunkte/Kanäle des Webcam-Gerätes in der WebUI ausgeblendet.

Um Fehlauslösungen durch Nutzung der WebUI zu vermeiden, lösen die Datenpunkte **PTZ_CMD** und **IR** nur bei Änderung ihres Wertes einen Aufruf der konfigurierten URL aus. Dazu wird der zuletzt gesetzte Wert intern im CUxD zwischengespeichert. Soll aus irgendeinem Grund die gleiche URL erneut aufgerufen werden, dann muss der intern gespeicherte letzte Wert des Datenpunktes durch Übergabe von **-1** unmittelbar vor dem eigentlichen Aufruf zurückgesetzt werden. Dabei wird keine URL aufgerufen!

Ein Snapshot wird nur beim Betätigen der Taste (Kanal 2) bzw. Auslösen des **SEND**-Datenpunktes auf der CCU abgespeichert!

Parameter		
TYPE	1	(1-99)
IP_DNS_ADR	192.168.72.17	
PORT	88	(0-65535)
USER	admin	
PASS	*****	
P_OPTION_1		
P_OPTION_2		
P_OPTION_3		
P_OPTION_4		
P_OPTION_5		

TYPE - vordefinierter Typ aus WEBCAMCONFIG-File, für eigene Typen wird dafür zuerst das WEBCAMCONFIG-File gescannt und danach das mit jeder Version aktualisierte webcamconfig-default.ini File im CUxD-Verzeichnis.

IP_DNS_ADR - IP- bzw. DNS-Adresse der Webcam **<ip>**

PORT - TCP-Port der Webcam **<port>**

USER - Webcam-Nutzer **<user>**

PASS - Webcam-Passwort **<pass>**

P_OPTION_1..5 - optionale Parameter **<opt1>** bis **<opt5>** zur Verwendung als Platzhalter in den URLs

Kanal	Parameter
Ch.: 1	WEBCAM|STREAM CACHE WEBCAM|WIDTH 480 px (0-640) WEBCAM|HEIGHT 240 px (0-480) WEBCAM|RELOAD 30 s (0-3600)
Ch.: 2	KEY|CMD_EXEC KEY|SNAPSHOTS 50 (0-999) KEY|CLEANUP ☐
Ch.: 3	MOTION_DETECTOR|ACTIVE ☑

STREAM - Steuerung der vom **IMAGE**-Datenpunkt ausgegebenen URL:
NO - Thumbnail und Bild sind Snapshot der Webcam
THUMB - Thumbnail ist Stream, Bild ist Snapshot
LARGE - Thumbnail ist Snapshot, Bild ist Stream
BOTH - Thumbnail und Bild sind Stream
CACHE - CCU als Proxy für Thumbnail und Bild Snapshots wie ***NO***, aber hier zeigen die URLs direkt auf den im WEBCAMCACHE-Verzeichnis zwischengespeicherten Snapshot

WIDTH - Maximalbreite des dargestellten Bildes in der WebUI

HEIGHT - Maximalhöhe des dargestellten Bildes in der WebUI

RELOAD - Aktualisierungsintervall des **IMAGE**-Datenpunktes in der WebUI bzw. der auf der CCU im WEBCAMCACHE-Verzeichnis zwischengespeicherten Webcam-Snapshots im ***CACHE***-Mode

CMD_EXEC - Befehlszeile, die nach dem erfolgreichen Ausführen eines Snapshots ausgeführt werden soll. Um den Snapshot dann weiterverarbeiten zu können (z.B. Email-Versand o. ä.), werden beim Aufruf neben den üblichen CUxD-Umgebungsvariablen (siehe Seite 83) zusätzlich die Umgebungsvariablen **CUXD_SNAPDIR** auf das aktuelle Snapshot-Verzeichnis und **CUXD_SNAPFILE** auf den aktuellen Snapshot-Dateinamen gesetzt.

<u>Beispiel:</u>
CUXD_SNAPDIR=/tmp/snapshots/CUX9104001/
CUXD_SNAPFILE=CUX9104001_026.jpg

SNAPSHOTS - maximale Anzahl der gespeicherten Snapshots im WEBCAMSNAPSHOT-Verzeichnis für die aktuelle Webcam (Speicherplatz beachten!)

CLEANUP - [x] alle *.jpg Snapshots aus dem WEBCAMSNAPSHOT-Verzeichnis der aktuellen Webcam löschen. Ist das Unterverzeichnis mit dem Gerätenamen danach leer, dann wird es auch gelöscht.

ACTIVE - [x] Bewegungsmelder Kanal aktivieren

Name	Raum	Gewerk	Letzte Aktualisierung	Control
Filter	Filter	Filter		
Webcam:1			05.04.2014 12:07:11	[INFO] FOSCAM [IMAGE] stream/image Snapshots [PTZ_CMD] 9 [IR=OFF] 0
Webcam:2		Taster	05.04.2014 12:00:00	Betätigen SNAPSHOT-Taste
Webcam:3				Helligkeit 0 keine Bewegung

Thumbnail-Vorschau der Snapshots:

Kanaltypen:

Kanaltyp	Kanalnummer
WEBCAM	1
KEY	2
MOTION_DETECTOR (CloudMatic)	3
SWITCH (Alarmkontakt)	4

Kanaltyp WEBCAM:

DP-Name	Typ	Zugriff	Beschreibung
INFO	string	lesend	Info-Text aus WEBCAMCONFIG-File: *<type> info 0 <Webcam-name>*
IMAGE	string	lesend	HTML-Code mit URLs für die Darstellung des Bildes usw. auf der WebUI. Die URLs werden im WEBCAMCONFIG-File definiert: *<type> img 0 <Snapshot-URL>* *<type> img 1 <Stream-URL>* *<type> img 2 <Snapshot-URL für Cache-Mode>* Über einen Link gelangt man auf eine neue Seite mit den Thumbnails aller gespeicherten Snapshots.
PTZ_CMD	integer	schreibend	Vordefinierte Funktion aus WEBCAMCONFIG-File aufrufen: *<type> ptz 0..99 <URL>*
IR	enum	schreibend	Infrarot-LED Steuerung (0=OFF, 1=AUTO, 2=ON) aus WEBCAMCONFIG-File aufrufen: *<type> ir 0 <URL für OFF>* *<type> ir 1 <URL für AUTO>* *<type> ir 2 <URL für ON>*

Kanaltyp KEY:

DP-Name	Typ	Zugriff	Beschreibung
SEND	action	schreibend	Snapshot in **WEBCAMSNAPSHOT=** Verzeichnis speichern. Die Snapshot-URL ist im WEBCAMCONFIG-File definiert: *<type> img 2 <Snapshot-URL>*
RESET	action	schreibend	Webcam rebooten. Die Funktion ist im WEBCAMCONFIG-File definiert: *<type> reset 0 <Reboot-URL>*

Kanaltyp MOTION_DETECTOR (Dummy-Kanal wird von CloudMatic gesetzt):

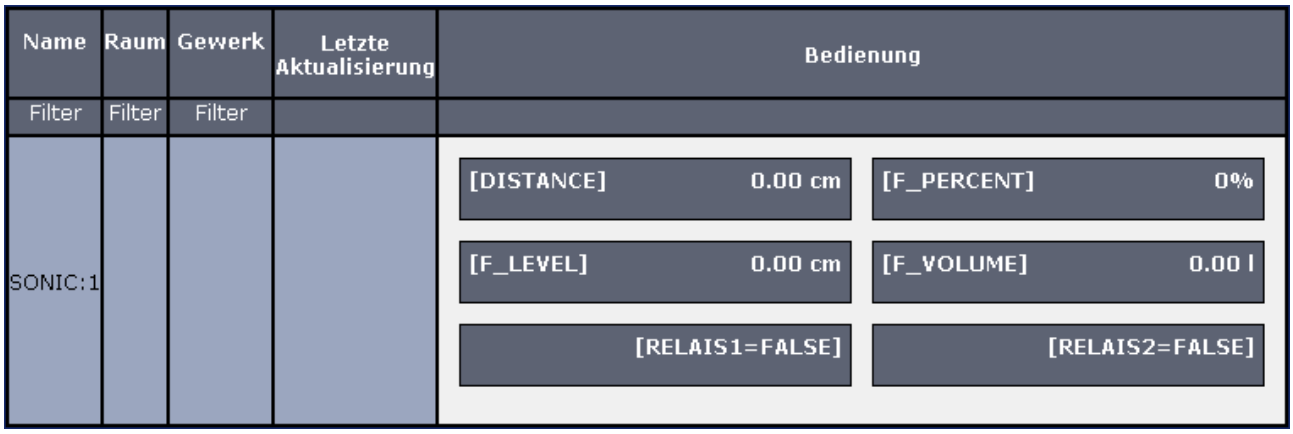
DP-Name	Typ	Zugriff	Beschreibung
BRIGHTNESS	integer	lesend	Helligkeit
MOTION	boolean	lesend	Bewegung
SET_BRIGHTNESS	integer	schreibend	Helligkeit setzen
SET_MOTION	bolean	schreibend	Bewegung setzen

5.10 Sonstige Geräte

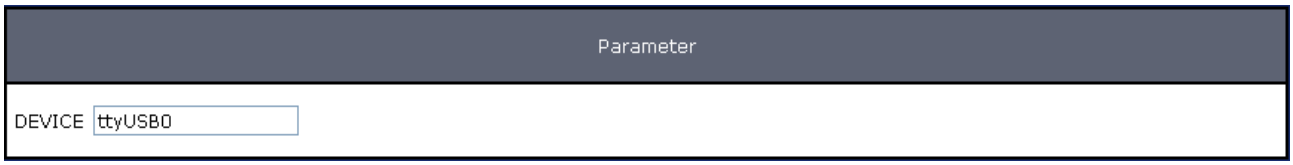
Die folgenden Geräte können keiner anderen Gruppe direkt zugeordnet werden.

5.10.1 (11) RS232-Füllstandsmesser {SONIC}

Mit dem ICPLAN Ultraschall-Füllstandsmesser (http://www.icplan.de/seite25.htm) kann der Füllstand einer Zisterne ermittelt werden. Die Ankopplung an die CCU erfolgt über einen USB-Serial Converter. Unterstützt werden USB-Serial Adapter mit Prolific PL2303, Moschip MOS7720, CH341 (nur CCU2), Silabs CP210x und FTDI Chipsatz. Für andere Adapter muss vorher manuell das passende Kernel-Modul auf der CCU geladen werden.



Konfigurationsparameter:



DEVICE - Geräte-ID oder TTY oder leer (Default: TTYASSIGN=ttyUSBx:SONIC)

Kanaltypen:

Kanaltyp	Kanalnummer
SENSOR	1

Kanaltyp SENSOR:

DP-Name	Typ	Einheit	Zugriff	Beschreibung
DISTANCE	float	cm	lesend	Abstand Sensor Wasser
F_PERCENT	integer	%	lesend	Füllprozente
F_LEVEL	float	cm	lesend	Füllhöhe
F_VOLUME	float	l	lesend	Füllmenge nach Geometrie des Behälters in Litern
RELAIS1	boolean		lesend	Zustand Relais1
RELAIS2	boolean		lesend	Zustand Relais2

5.10.2 (40) 16 Kanal Universalsteuerung

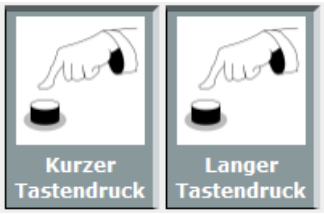
Dieses Gerät ist universell für viele Aufgaben einsetzbar. Es können beliebige frei definierbare Befehle (auch RAW-Befehle) über eine beliebige serielle Schnittstelle gesendet und empfangen werden.

Auf zuvor definierte Empfangsbefehle (RCV_...) wird vom CUxD auf der CCU ein kurzer bzw. langer Tastendruck generiert oder Status gesetzt. Gleichzeitig kann darauf vom CUxD automatisch mit dem Aussenden einer Antwort (CMD_...) auf einem frei definierbaren Gerät, reagiert werden. Diese Steuerung ist nur aktiv, wenn als Control der Taster ausgewählt wurde.

Sind CMD_SHORT und CMD_LONG leer, dann ändert sich beim Tastendruck bzw. Schaltvorgang nur der Status des Kanals (Schalter, Tür-/Fensterkontakt), ohne dass dabei Daten gesendet werden.

Die folgenden Controls können beim Anlegen des CUxD-Gerätes gewählt werden:

Taster (zustandslos)



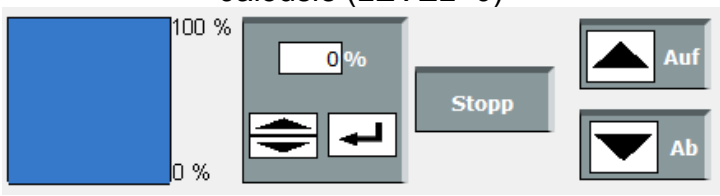
Schalter (STATE=false)

Tür-/Fensterkontakt (STATE=false)

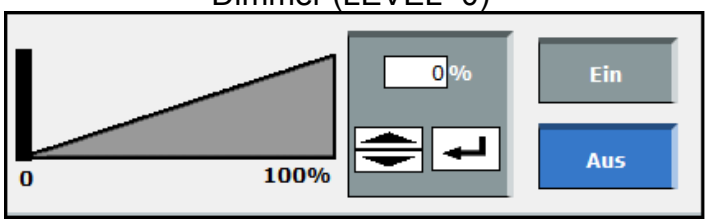
Bewegungsmelder (MOTION=false)

Gefahrenmelder (STATE=false)

Jalousie (LEVEL=0)



Dimmer (LEVEL=0)



Konfigurationsparameter:

Parameter
DEVICE 1-1

DEVICE - USB-ID oder TTY oder leer (**für RCV_ Parameter, bei allen Kanälen gleich!**)

Kanal	Parameter
Ch.: 1	KEY|ACTIVE ☑ KEY|REPEAT 0 (0-2) KEY|DEVICE KEY|REG_MATCH ☐ KEY|RCV_SHORT KEY|RCV_LONG KEY|CMD_EXEC ☐ KEY|CMD_SHORT KEY|CMD_LONG

ACTIVE - [x] Channel ist Aktiv. Alle deaktivierten Channels werden in der WebUI ausgeblendet. Sollte die Darstellung nicht aktualisiert werden, dann hilft ein Reload der WebUI im Webbrowser.

REPEAT - Anzahl der Sendewiederholungen für schlecht erreichbare Aktoren (nur für CUL, 0 ist der Defaultwert und bedeutet KEINE Wiederholung)

DEVICE - USB-ID oder TTY (**für alle Sendebefehle über diesen Kanal!**) Wenn dieser Parameter leer ist, dann wird auf dem DEVICE gesendet, das für den Empfang (**RCV_** Parameter) definiert wurde. (siehe oben!)

REG_MATCH - POSIX Regular Expressions zum Vergleich der Empfangszeichenkette (**RCV_SHORT** und **RCV_LONG**) nutzen

RCV_SHORT - Empfangszeichenkette (kurzer Tastendruck) zum Vergleich oder leer

RCV_LONG - Empfangszeichenkette (langer Tastendruck) zum Vergleich oder leer

CMD_EXEC - [x] Kommandozeile nach Empfang und beim Senden ausführen!

CMD_SHORT - Sendebefehle (kurzer Tastendruck oder Aus oder AB) oder leer

CMD_LONG - Sendebefehle (langer Tastendruck oder Ein oder AUF) oder leer

CMD_STOP - Sendebefehle für ***Jalousie Aktor*** (STOP-Kommando) oder leer

Wenn **REG_MATCH deaktiviert** ist, dann wird die konfigurierte Empfangszeichenkette mit dem Anfang der empfangenen Datenzeile verglichen. Stimmt sie überein, dann werden (wenn vorhanden und das **Gerät als Taster definiert** ist) automatisch bei **RCV_SHORT** die **CMD_SHORT**-Befehle und bei **RCV_LONG** die **CMD_LONG**-Befehle gesendet. Der Sendebefehl kann dabei auch auf einem anderen DEVICE (s. DEVICE-Parameter im Kanal) ausgesendet werden. Ein Fragezeichen **?** in der Empfangszeichenkette dient als Platzhalter für ein beliebiges Zeichen.

Es können mehrere {CUX}-Sendebefehle durch Leerzeichen getrennt eingegeben werden. Bei nicht-CUX-Geräten wird die gesamte Zeichenkette ohne Änderung gesendet. Die Zeichen **=** und **"** müssen durch den entsprechenden Hex-Code **\x3D** bzw. **\x22** ersetzt werden, ansonsten gibt es im WebUI-Formular Probleme.

Bei manuellem oder programmiertem Auslösen eines kurzen bzw. langen Tastendrucks **und** definierten Sendebefehlen (CMD_...), werden diese gesendet und die letzte Aktualisierung bekommt den aktuellen Zeitstempel. Gleichzeitig wird beim Datenempfang ein kurzer (..._SHORT) bzw. langer (..._LONG) Tastendruck generiert.

Um das Gerät zur universellen Ankopplung fremder Aktoren und Sensoren nutzen zu können, besteht die Möglichkeit, anstelle von vordefinierten Sendebefehlen, ein eigenes Script aufzurufen, das dann eine individuelle Verarbeitung der Daten übernehmen kann.

Dazu muss **CMD_EXEC** aktiviert und danach anstelle von Sendebefehlen ein Befehlsaufruf (z.B. TCL-Script) in die **CMD_**-Parameter eingetragen werden. Dieses Script kann Daten mittels **SEND_CMD** auf das TTY senden. LEVEL-Werte (Jalousie und Dimmer) werden für die Übergabe in Integer-Werte zwischen 0 (0%) und 1000 (100.0%) umgerechnet. Rückmeldungen sind vom Script direkt auf den Datenpunkt **SET_STATE** des Kanals zu schreiben (siehe nächster Absatz).

Beim Empfang wird immer nach erfolgreichem Vergleich der Empfangszeichenkette mit **RCV_SHORT** die Befehlszeile in **CMD_SHORT** und mit **RCV_LONG** die Befehlszeile in **CMD_LONG** aufgerufen. Der Status des CUxD-Gerätes muss aus dem Script heraus über den Datenpunkt **SET_STATE** des Kanals gesetzt werden. Die LEVEL-Werte (Jalousie und Dimmer) können dabei entweder als positive Float-Werte 0.00 = 0% und 1.00 = 100% bzw. negative Integer-Werte 0 = 0% und -1000 = 100.0% übergeben werden.

Bei jedem Befehlsaufruf (**CMD_SHORT, CMD_LONG, CMD_STOP**) werden zusätzliche **Umgebungsvariablen** gesetzt:

CUXD_CHANNEL	aufgerufener Kanal des aktuellen CUxD-Gerätes: CUX40xxxxx:1
CUXD_VALUE	Senden (TRX=1): kurzer (0) oder langer (1) Tastendruck bzw. Ein (1), Aus (0) Zustand bzw. LEVEL (0..1000) beim Jalousie- und Dimm-Aktor Empfang (TRX=0): vollständige Empfangszeichenkette
CUXD_OLDVALUE	VALUE-Wert vom letzten Aufruf (nur bei TRX=1)
CUXD_TRX	(0) Empfang, (1) Senden

In der **Befehlszeile** können dabei folgende Platzhalter genutzt werden:

$CHANNEL$	entspricht ***CUXD_CHANNEL***
$VALUE$	entspricht ***CUXD_VALUE***
$OLDVALUE$	entspricht ***CUXD_OLDVALUE***
TRX	entspricht ***CUXD_TRX***

Kanaltypen:

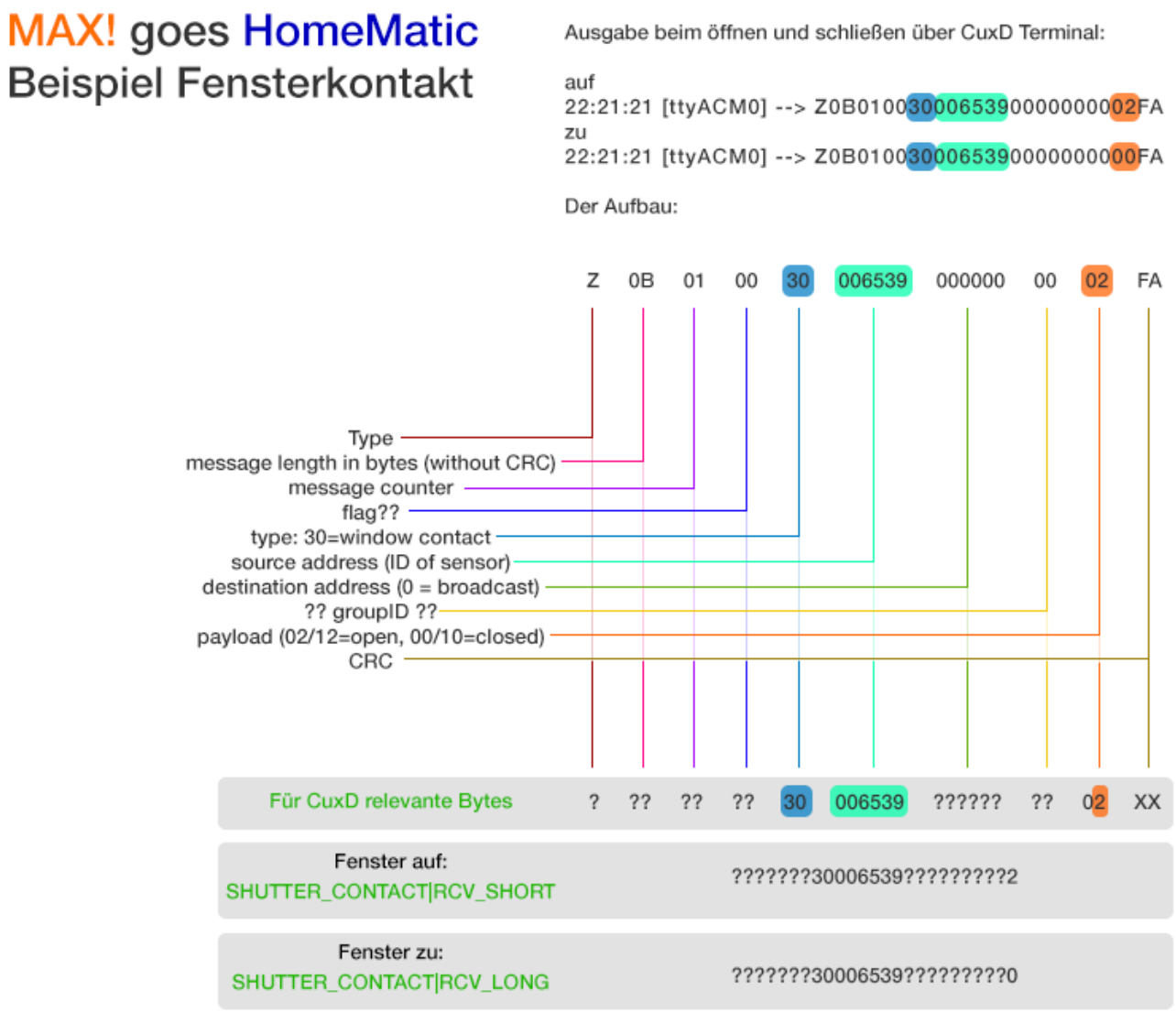
Kanaltyp	Kanalnummer
KEY / SWITCH / DIMMER / SHUTTER / MOTION_DETECTOR / DANGER / BLIND	1..16

Kanaltyp KEY / SWITCH / DIMMER / SHUTTER / MOTION_DETECTOR / DANGER/ BLIND:

DP-Name	Typ	Zugriff	Beschreibung
STATE	boolean	lesend schreibend	Schalter/Fensterkontakt: Beim Schaltzustand *false* (Aus) wird **CMD_SHORT** und beim Schaltzustand *true* (Ein) **CMD_LONG** gesendet. Beim Empfang vom **RCV_SHORT** wird der Schaltzustand *false* (Aus) und beim Empfang von **RCV_LONG** der Schaltzustand *true* (Ein) gesetzt.
PRESS_SHORT	action	event schreibend	Taster: kurzer Tastendruck
PRESS_LONG	action	event schreibend	Taster: langer Tastendruck
LEVEL	float	lesend schreibend	Jalousieaktor: öffnen bzw. schließen Dimmer: Dimmwert setzen
STOP	action	schreibend	Jalousieaktor: Bewegung anhalten
OLD_LEVEL	action	schreibend	Dimmer: letzten Dimmwert setzen
MOTION	boolean	lesend	keine Bewegung = **RCV_SHORT** empfangen Bewegung erkannt = **RCV_LONG** empfangen
SEND_CMD	string	schreibend	Über diesen Datenpunkt können beliebige Sendebefehle sofort zum entsprechenden DEVICE gesendet werden.
RCVS	string	lesend	komplette Empfangszeile nach einem erfolgreichen Vergleich mit dem **RCV_SHORT**-Parameter
RCVL	string	lesend	komplette Empfangszeile nach einem erfolgreichen Vergleich mit dem **RCV_LONG**-Parameter
CONTROL	integer	lesend	konfiguriertes Control-Element: 0..Taster (**KEY**) 1..Schalter (**SWITCH**) 2..Tür-/Fensterkontakt (**SHUTTER**) 3..Bewegungsmelder (**MOTION_DETECTOR**) 4..Gefahrenmelder (**DANGER**) 5..Jalousie (**BLIND**) 6..Dimmer (**DIMMER**)
SET_STATE	float	schreibend	Gerätestatus (STATE, LEVEL, MOTION) manuell setzen, ohne eine Aktion auszuführen.

Empfang von MAX! Fensterkontakten und Tastern mittels Universalsteuerung

Für den Empfang von MAX! Datentelegrammen ist auf dem CUL zuerst der Moritz-Mode mittels **Zr** Befehl im CUxD-Terminal bzw. fest über den **TTYINIT=** Parameter zu aktivieren!



MAX! Fensterkontakt (Typ 30) als CUxD-Tür-/Fensterkontakt

Fenster auf:

SHUTTER_CONTACT|RCV_SHORT=Z??????30**YYYYYY**??????????0

Fenster zu:

SHUTTER_CONTACT|RCV_LONG=Z??????30**YYYYYY**??????????2

MAX! ECO-Taster (Typ 50) als CUxD-Taster oder CUxD-Schalter

Taste oben (bzw. Aus):

*|RCV_SHORT=Z??????50YYYYYY??????????????1

Taste unten (bzw. Ein):

*|RCV_LONG=Z??????50YYYYYY??????????????0

MAX! Zwischenstecker (Typ 40) als CUxD-Schalter

Der MAX! Zwischenstecker muss zuvor an einen MAX! Cube angelernt werden. Danach kann man ihn mit Statusrückmeldung vom MAX! Cube und vom CUxD schalten.

Aus:

SWITCH|RCV_SHORT=Z??????40??????YYYYYY???E

SWITCH|CMD_SHORT=Zs0BNN0440XXXXXXYYYYYY01VE

Ein:

SWITCH|RCV_LONG=Z??????40??????YYYYYY???2

SWITCH|CMD_LONG=Zs0BNN0440XXXXXXYYYYYY01V2

```
NN     - Zähler für Funk-Telegramme von 00 bis FF
XXXXXX - Funk-Adresse vom MAX! Cube
YYYYYY - Funk-Adresse des Zwischensteckers
V      - Wert zwischen 0 und F
```

6 Zusatzprogramme

Im Verzeichnis /usr/local/addons/cuxd/extra/ sind einige Tools zur Nutzung mit dem CUxD abgelegt. Sie können auch direkt von der CCU-Konsole aufgerufen werden.

6.1 artdmxdim (DMX: 512 Kanäle)

Mit diesem Tool können Dim-Werte und Lichteffekte über das ArtNet-DMX Protokoll an ArtNet-Node's (z.B. http://www.ulrichradig.de/home/index.php/dmx/alias-2) übertragen werden.

Durch den gleichzeitigen Aufruf mit unterschiedlichen Parametern ist es möglich, Fading-Effekte auf mehreren Kanälen gleichzeitig und unabhängig voneinander ablaufen zu lassen, so lange sich die Kanalnummern unterscheiden.

In der Datei /tmp/artdmx_<ip>_<universe>.dat wird der aktuelle Zustand aller 512 Kanäle des DMX-Universums abgespeichert. Zur Kontrolle kann der Zustand einzelner Kanäle oder des gesamten DMX-Universums mit dem r-Parameter ausgelesen werden.

Aufruf:

```
./artdmxdim <HOST> <UNIV> <FN>[:<NUM>] <WAIT[:MAXW]> [<playlist.txt>|<C:L>...]
```

HOST Hostname oder IP-Adresse der ArtNet-Node

UNIV DMX-Universum

FN auszuführende Funktion:
- r……Status des Kanals auslesen (ch [ch …])
- w…..gespeicherten Status des Universums zur DMX-ArtNetNode senden
- 1.....DimTo (ch:Wert [ch:Wert …])
- 2:n...setzen der Dim-Level anhand einer vordefinierten Playlist mit optionaler Angabe der Wiederholungen
- 3.....RGB Sonnenuntergang vom aktuellen Dimmwert auf 0
- 4.....RGB Sonnenaufgang bis zum Wert (255,255,0)
- 5:n...RGB Farbwechsel mit optionaler Angabe der Wiederholungen
- 6:n...Zufalls-Farbwechsel mit zufälliger Wartezeit (20ms Schritte) und optionaler Angabe der Wiederholungen
- 7.....DimTime (ch:Wert [ch:Wert …]) Soft-Dimmwertänderung in der vorgegebenen (zufällig bestimmten) Zeitspanne (*MINWAIT:MAXWAIT*) in ms.
- 8.....Funktion der Kanäle stoppen.

WAIT Wartezeit nach jeder einzelnen Dimmwert Änderung in Millisekunden (abhängig von ausgewählter Funktion). Bei 0 werden die Ziel-Werte sofort und ohne Soft-Übergang gesetzt. Bei Funktion 6 kann für die Zufallsberechnung eine minimale und eine maximale Wartezeit vorgegeben werden. Bei Funktion 7 kann die Dimm-Zeit und Mindestschrittweite in ms angegeben werden

C:L Kanalnummer **C** (1..512) mit optionalem Dimm-Wert **L** (0..255). Bei Funktion 6 wird optional der maximale Dimm-Wert angegeben.

Beispiele:

aktuellen Wert von Kanal 1-3 sofort auf 0 setzen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 1 0 1:0 2:0 3:0
```

Kanal 4-6 mit 30ms Pause zwischen den Befehlen auf den angegebenen Wert dimmen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 1 30 4:250 5:250 6:50
```

RGB Sonnenuntergang vom aktuellen Wert mit 300ms Pause zwischen den Befehlen (Kanal 7,8,9):

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 3 300 7
```

RGB Sonnenaufgang vom aktuellen Wert mit 300ms Pause zwischen den Befehlen (Kanal 7,8,9):

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 4 300 7
```

RGB Farbwechsel vom aktuellen Wert mit 300ms Pause und 10 Wiederholungen (Kanal 7,8,9):

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 5:10 300 7
```

Zufalls-Farbwechsel (Kanal 1,2,3):

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 6 0 1 2 3
```

Zufalls-Farbwechsel mit 20 Wiederholungen (Kanal 1-3), zufälliger Pause (1s-5s) und Soft-Überblendung:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 6:20 1000:5000:d 1 2 3
```

Zufalls-Farbwechsel mit 99 Wiederholungen (Kanal 1-3), einer zufälligen Pause zwischen 20 und 80ms und einer Begrenzung des maximalen Dim-Wertes auf 120 (Kanal 2) und 70 (Kanal 3):

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 6:99 20:80 1 2:120 3:70
```

TV-Simulation (RGB Kanal 1,2,3):

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 6:999 200:8000 1:120 2:120 3:70
```

Werte für die Kanäle 1, 2 und 3 auslesen (Ergebnisse durch Tabulator getrennt):

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 r 0 1 2 3
```

Werte für alle Kanäle auslesen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 r 0 all
```

Kanal 1-3 in 5000ms = 5 Sekunden soft vom aktuellen auf den angegebenen Wert dimmen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 7 5000 1:250 2:250 3:50
```

Kanal 1-3 in zufälliger Zeit zwischen 1 und 9 Sekunden vom aktuellen auf den angegebenen Wert dimmen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 7 1000:9000 1:90 2:50 3:70
```

eine aktive DIM-Funktion auf Kanal 2 stoppen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 8 0 2
```

vordefinierte Playlist mit einer Start-Verzögerung von 1s abspielen:

```
/usr/local/addons/cuxd/extra/artdmxdim 192.1.0.90 0 2 1000 /tmp/playlist.txt
```

kommentierte Beispiele für die Datei *playlist.txt*:

```
1:150 2:0 3:0 w:500 1:0 2:150 3:0 w:500
1:0 2:0
3:150:255 # Kanal auf Zufallswert zwischen 150 und 255 setzen
w:1000 # jetzt senden und 1s (1000ms) warten
1:10 2:10 3:10 d:1000:2000 # dimmen zufällig innerhalb von 1-2s
p:200:300 # Kanäle 1,2,3 zufällig auf 200-300% hochrechnen
d:2000 # innerhalb von 2s auf den neuen Wert dimmen
w:1000:3000 # zufällig 1-3s warten
# und nun alles ausschalten!
1:0 2:0 3:0
```

(Eine Zeile darf maximal 250 Zeichen enthalten. Die Dim-Werte haben das Format **<Kanal>:<Min>[:<Max>]** und können frei über mehrere Zeilen definiert werden. Bei Angabe eines Max-Wertes wird zur Laufzeit ein Zufallswert zwischen Min und Max bestimmt.

Steht als Kanalnummer **w** (wait), dann werden alle bis dahin definierten Werte zur ArtNet-Node gesendet. Weiter geht es nach der konfigurierten Wartezeit in Millisekunden.

Bei **d** (dim) wird in der angegebenen Zeit (siehe Funktion 7) auf den zuvor definierten Wert gedimmt. Werden hinter **d** und **w** zwei Werte eingegeben (**d:min:max** bzw. **w:min:max**), dann wird eine zufällige Wartezeit zwischen min und max berechnet.

Steht als Kanalnummer **p** (percent), dann werden die Werte aller zuletzt angegebenen Kanäle anhand des Prozentwertes umgerechnet. 100% entsprechen dem Originalwert, 50% der Hälfte und 200% dem doppelten. Auch hier ist anhand von 2 Werten die Angabe eines Zufallsbereiches möglich. Für die Ausgabe ist ein **w** bzw. **d** erforderlich.

Steht in einer Zeile ein **#**-Zeichen (Kommentarzeichen), dann werden alle nachfolgenden Eingaben in dieser Zeile übersprungen)

6.2 ccu_backup

Dieses Script erzeugt ein gültiges Systembackup (SBK-File), mit dem ein Restore über die WebUI möglich ist. Das Backup-File kann in einem beliebigen Verzeichnis (auch NFS oder auf einem externen USB-Stick) abgelegt werden. Durch einen zeitgesteuerten Aufruf über eine Programmverknüpfung lassen sich so automatisch z.B. tägliche oder wöchentliche Backups anlegen.

Aufruf:

```
/usr/local/addons/cuxd/extra/ccu_backup [<Zielverzeichnis> [<Filename>]]
```

Beispiel:

```
/usr/local/addons/cuxd/extra/ccu_backup /home/backup
```

Alle Parameter sind optional. Defaultwert für den Filenamen ist der Name, den die CCU auch in der WebUI vergeben würde mit einem um Stunde und Minute erweiterten Zeitstempel. Das Default-Zielverzeichnis ist: /tmp

6.3 dom_save

Dieses Script aktualisiert die zentrale Konfigurationsdatei (/etc/config/homematic.regadom) der CCU und prüft die gespeicherte Datei auf Vollständigkeit. Es bietet sich an, dieses Script vor einem manuellen Reboot der CCU aufzurufen. Bei einem Fehler wird ein Fehlercode zurückgeliefert.

Aufruf:

```
/usr/local/addons/cuxd/extra/dom_save
```

Beispiel:

```
/usr/local/addons/cuxd/extra/dom_save
```

6.4 dom_backup

Dieses Script prüft die Konfigurationsdatei (/etc/config/homematic.regadom) und kopiert diese einschließlich der aktuellen Gerätedateien und der cuxd.ini und cuxd.ps Files in einem tgz-Archiv in das angegebene Zielverzeichnis (z.B. auf einem externen USB-Stick).

Durch den zeitgesteuerten Aufruf über eine Programmverknüpfung lassen sich so z.B. täglich automatische Sicherungen anlegen. Bei einem Fehler wird ein Fehlercode zurückgeliefert.

Aufruf:

```
/usr/local/addons/cuxd/extra/dom_backup [<Zielverzeichnis> [<Filename>]]
```

Beispiel:

```
/usr/local/addons/cuxd/extra/dom_backup /home/dombackup
```

Die Parameter sind optional. Defaultwert für den Filenamen ist der Name, den die CCU einem Backup in der WebUI vergeben würde mit einem um Stunde und Minute erweiterten Zeitstempel. Wird kein Zielverzeichnis angegeben, dann erfolgt lediglich eine Prüfung der aktuellen Konfigurationsdatei.

6.5 ether-wake v1.09

Dieses kleine Tool von Donald Becker (http://www.scyld.com/), dient zum Versenden von Wake-on-LAN (WOL) Paketen. Damit können WOL fähige Computer aufgeweckt (eingeschaltet) werden. Weitere Infos finden sich hier: http://wiki.tuxbox.org/Etherwake

Aufruf:

```
/usr/local/addons/cuxd/extra/ether-wake <MAC-Adresse>
```

Beispiel:

```
/usr/local/addons/cuxd/extra/ether-wake 00:11:22:33:44:55
```

6.6 digitemp_DS9097U v3.5.0

Dieses Tool von Brian C. Lane (http://www.digitemp.com/software.shtml) dient zur Abfrage von 1-Wire Temperatursensoren (DS18S20, DS1820, DS18B20, DS1822 usw.) über einen per USB angeschlossenen DS9097U Adapter.

Werden die Messwerte der Temperatursensoren in einem vordefinierten Format abgespeichert, dann können sie mittels **TH-DIR=** Parameter eingelesen und mittels CUxD-Wrapper-Device auf der CCU verarbeitet werden.

Aufruf:

```
/usr/local/addons/cuxd/extra/digitemp_DS9097U <options>
```

Beispiel für die Integration von 1-Wire Temperatursensoren:

Testen ob Adapter und Sensoren erkannt werden (Beispiel: ttyUSB0):

```
/usr/local/addons/cuxd/extra/digitemp_DS9097U -q -s /dev/ttyUSB0 -w
```

DigiTemp Konfigurationsfile erzeugen (absoluter Pfad eingekürzt):

```
.../digitemp_DS9097U -i -s /dev/ttyUSB0 -q -c .../digitemp.conf
```

*Im Konfigurationsfile den DigiTemp Parameter „**LOG_FORMAT %s %.2C**“ setzen!*

DigiTemp periodisch mittels Zeitsteuerung in einer WebUI Programmverknüpfung aufrufen:

```
/usr/local/addons/cuxd/extra/digitemp_DS9097U -a -q -c
/usr/local/addons/cuxd/extra/digitemp.conf -l /tmp/digitemp.log -n 3
```

CUxD-Wrapper „**(3) Thermostat Device**“ Gerät anlegen und als Temperatursensor mit einer Sensornummer aus der Datei digitemp.conf bzw. digitemp.log konfigurieren.

CUxD-Konfigurationsparameter **TH-DIR=**/tmp/digitemp.log setzen.

6.7 export_ftp.sh

Dieses Shell-Script nutzt Curl um Dateien aus einem Spool-Verzeichnis per FTP zu einem externen Host zu verschieben. Während der Übertragung wird die Datei im FTP-Zielverzeichnis als <file>.part angelegt und nach erfolgreicher Übertragung dann in <file> umbenannt und aus dem Spool-Verzeichnis gelöscht.

Aufruf:

```
/usr/local/addons/cuxd/extra/export_ftp.sh user:pass <ip> <ftp-dir> <spool-dir>
```

Im Zusammenhang mit dem **DEVLOGEXPORT=** Parameter können die Device-Logfiles auf diese Weise ganz einfach von der CCU auf einen externen Server übertragen werden.

Beispiel zum Export nach ftp://192.168.1.2/import/ccu:

```
STARTUPCMD=mkdir /tmp/export
DEVLOGFILE=/tmp/devlog.dat
DEVLOGMOVE=/tmp/export
DEVLOGEXPORT=/usr/local/addons/cuxd/extra/export_ftp.sh username:password
192.168.1.2 import/ccu
```

6.8 timer.tcl

Den Wert eines Datenpunktes nach einer vorgegebenen Zeit in Sekunden ändern. Die Sekunden sind optional und können beim Aufruf auch weggelassen werden. Da der Aufruf des Scripts nicht retriggerbar ist, sollte für längere Intervalle der **System.Timer (16 Kanäle)** genutzt werden. Dieses Script kann auch genutzt werden, um einen Datenpunkt über die Befehlszeile (z.B. mittels **CMD_EXEC**) zu ändern. Um die Funk-Kommunikation zu minimieren kann mit dem <Vergleich> Parameter verhindert werden, dass ein Aktor erneut auf den Zustand gesetzt wird, in dem er sich gerade befindet.

Aufruf:

```
/usr/local/addons/cuxd/extra/timer.tcl <DP> <Wert> [<Sek>] [<CMP>] [<ONTIME>]
```

Beispiel:

Einschalten eines Aktors nach 12.3 Sekunden:

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.IEQ0514341:1.STATE 1 12.3
```

Abschalten des Aktors sofort:

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.IEQ0514341:1.STATE 0
```

sofortiges Senden des Einschaltbefehls, wenn der Aktor noch nicht eingeschaltet ist:

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.IEQ0514341:1.STATE 1 0 1
```

sofortiges Senden des Einschaltbefehls und Ausschalten nach 60 Sekunden:

```
/usr/local/addons/cuxd/extra/timer.tcl BidCos-RF.IEQ0514341:1.STATE 1 0 0 60
```

7 Konfiguration

Die Konfigurationsdaten sind auf der CCU in der Datei „cuxd.ini“ im Programmverzeichnis des CUx-Daemon gespeichert.

Alle Einstellungen der Parameter sollten bei Bedarf über die Administrationsoberfläche erfolgen. (Eine externe Modifikation mit einem Texteditor ist auch möglich, erfordert dann abschließend aber immer einen CUxD-Restart).

Nach Änderung von Parametern über die Administrationsoberfläche werden diese in der Regel sofort übernommen. Eine Ausnahme stellen bestimmte, auf der Statusseite extra gekennzeichnete, Parameter dar. Diese erfordern nach Änderung einen Neustart über die „Restart“ Taste.

Nach der Erstinstallation werden Default-Parameter angelegt. Über die Taste „Parameterabgleich“ können (z.B. nach einem Versionsupdate) Defaultwerte für bestimmte neue Parameter hinzugefügt werden. Es können auch Leer- und Kommentarzeilen (mit einem Semikolon beginnend) eingefügt werden.

Die Änderung der folgenden Parameter ist nicht notwendig und erfordert immer einen CUxD-Restart.

LISTENPORT=8700

- Port für die interne HTTP-Kommunikation des CGI-Proxy-Scripts mit dem CUxD.

HM-SCRIPTHOST=127.0.0.1
HM-SCRIPTPORT=8181

- Adresse und Port für den HM-Script Zugriff zum ReGaHSS (**nicht ändern!**)

RPCHOST=127.0.0.1
RPCPORT=8701

- Adresse und Port des CUxD RPC-Servers (**nicht ändern!**)

Bei Änderung nachfolgender Parameter ist <u>kein</u> CUxD-Restart erforderlich. Sie werden sofort übernommen.

HTTP-REFRESH=5

- Refresh-Intervall in Sekunden für die „Terminal-Seite“ der Administrationsoberfläche zum Senden bzw. Empfangen von Datenpaketen

TERMINALLINES=25

- Anzahl der Zeilen, die Im Terminalfenster angezeigt werden sollen (Browser-abhängig)

RCVLOGSIZE=8000

- Größe des Ringpuffers (in Bytes) für das Logging von empfangenen und gesendeten Datenpaketen

ADDRESS-BUFFER=60

- Mindestzeit in Minuten, für die empfangene Adressen gespeichert werden sollen. Diese Adressen werden auf der CUxD-Statusseite unter „*gefundene Adressen*“ angezeigt.

USERLOGIN=

- Die CUxD-Administrationsoberfläche kann mit „basic authentication“ geschützt werden. Format: user:password, z.B. USERLOGIN=root:pass

AUTOSAVE=1

- 0.. kein automatisches Speichern der Gerätekonfiguration
- 1.. automatisches Speichern der Gerätekonfiguration bei jedem Stop/Restart/Reboot des Daemons bzw. der CCU und jeden Tag um 0:00 Uhr.

DEVLOGFILE=

- Zum Aktivieren des Device-Logging ist hier ein Dateiname mit dem vollständigen lokalen Pfad auf der CCU einzutragen. Bei leerem Parameter ist das Logging deaktiviert, ansonsten kann das Logfile unter „Info → Device-Log“ angezeigt werden. Ein direktes Loggen auf den USB-Stick bzw.die SD-Karte kann Stabilitätsprobleme der CCU verursachen. Hier ist die Verwendung von **DEVLOGMOVE=** zu empfehlen.

DEVLOGSIZE=+100000

- Dieser Parameter bestimmt die maximale Größe des Device-Logfiles in Bytes. Das Logfile wird bei Überschreiten der Größe vom Anfang her gekürzt. Somit bleiben immer die letzten aktuellen Daten erhalten. Ist der Parameter auf 0 gesetzt, dann wird die Datei nicht gekürzt.
- Ist zusätzlich der Parameter DEVLOGMOVE= definiert, dann wird das Device-Logfile beim überschreiten der Größe entsprechend verschoben. Bei '+' zusätzlich auch täglich um 0:00 Uhr.

DEVLOGMOVE=

- Dieser Parameter ist optional und enthält das Zielverzeichnis in welches DEVLOGFILE täglich um 0:00 Uhr oder/und nach Überschreiten der Dateigröße und vor jedem CCU-Reboot verschoben wird. Ein täglicher Export um 0:00 Uhr erfolgt bei DEVLOGSIZE=0, ein Export bei Überschreiten der Dateigröße bei DEVLOGSIZE=<size> und beides bei DEVLOGSIZE=+<size>. Das Verschieben erfolgt in folgender Reihenfolge:
  1. die Datei wird in <name>.0 umbenannt.
  2. Ein neuer Hintergrundprozess wird gestartet
     1. die Datei wird von <name>.0 in <name>.YYYYMMDD-HHMM umbenannt, wenn die Zieldatei noch nicht existiert.
     2. Prüfen, ob das Zielverzeichnis existiert.
     3. die Datei wird unter dem Namen <name>.YYYYMMDD-HHMM.$ ins Zielverzeichnis kopiert, wenn sie dort noch nicht existiert.
     4. die Ziel-Datei wird in <name>.YYYYMMDD-HHMM umbenannt
     5. die Quell-Datei wird gelöscht

DEVTIMEFORMAT=%Y.%m.%dT%X

- Format für die Datumsausgabe im Logfile (siehe **Daten-Logging**)

DEVDATAFORMAT=

- Format für die Daten im Logfile (siehe **Daten-Logging**)

LOGIT=

- Die Auswahl der zu „loggenden“ Geräte erfolgt über diesen Parameter. Er kann beliebig oft wiederholt werden und definiert normalerweise je Eintrag ein CCU-Gerät bzw. einen Datenpunkt (siehe **Daten-Logging**).

DEVLOGEXPORT=

- Dieser Parameter ist optional und definiert einen Kommandozeilenbefehl, der periodisch (alle 5 Minuten) nach dem Umbenennen des aktuellen Device-Logfiles in YYYYMMDD-HHMM.<name> aufgerufen wird. Zusätzlich wird der Kommandozeile das unter **DEVLOGMOVE=** definierte Verzeichnis und der Name des aktuellen Device-Logfiles übergeben. Das unter **DEVLOGMOVE=** definierte Verzeichnis dient dabei als Spool-Verzeichnis (auf der RAM-Disk der CCU!) für den Export. Es muss auf dem gleichen Volume (RAM-Disk!) wie **DEVLOGFILE=** liegen!

Beispiel für Export nach /mnt/export:
STARTUPCMD=mkdir /tmp/export
DEVLOGFILE=/tmp/devlog.dat
DEVLOGMOVE=/tmp/export
DEVLOGEXPORT=/usr/local/addons/export.sh /mnt/export

alle 5 Minuten wird folgendes ausgeführt:

- *rename /tmp/devlog.dat → /tmp/export/YYYYMMDD-HHMM.devlog.dat*
- *fork()*
  - *system(<DEVLOGEXPORT> <DEVLOGMOVE> YYYMMDD-HHMM.devlog.dat)*

export_file.sh (sofortiges Verschieben des aktuellen Files):

```
#!/bin/sh
mv $2/$3 $1/$3.part
mv $1/$3.part $1/$3
```

export_dir.sh (verschieben aller Files aus dem Spool-Verzeichnis):

```
#!/bin/sh
cd $2
for file in *
do
  if [ -f $file ]
  then
    mv $file $1/$file.part
    mv $1/$file.part $1/$file
  fi
done
```

SUBSCRIBE_RF=1
SUBSCRIBE_WR=1

- Diese beiden Parameter akzeptieren 0 oder 1 als Wert. Wird der entsprechende Parameter auf '1' gestellt, so abonniert der CUxD die Events die von der CCU bei Datenänderungen an die entsprechenden Geräte (RF = Funk, WR = Wired) gesendet werden.
  Sollen nur CUxD-Geräte geloggt werden, so können beide Parameter auf '0' gestellt werden. Möchten man nur Funk-Geräte loggen, so reicht es SUBSCRIBE-RF=1 zu setzen. Diese Parameter sind auch für die Funktion des Universal-Wrapper-Device notwendig.

MOUNTCMD=

- ist dieser Parameter gesetzt, dann kann der eingetragene Befehl mittels CUxD auf der Statusseite ausgeführt werden (Mount-Taste). Es können auch mehrere Befehle mit Semikolon getrennt eingegeben werden.

  Beispiele: *MOUNTCMD=mount -t vfat /dev/sda1 /home*
  MOUNTCMD=mount -t nfs -o nolock 192.168.5.67:/Public/ccu /home

  Auf der Statusseite besteht jetzt die Möglichkeit, das Verzeichnis /home zu mounten. Der aktuelle Status wird abgespeichert und bei einem Neustart des CUxD wiederhergestellt. Das bedeutet, ein gemounteter USB-Stick wird sofort nach einem Reboot der CCU beim CUxD-Start erneut gemountet. Der USB-Stick wird auch gemountet, wenn die USB-Schnittstelle nach einem Stromausfall automatisch aktiviert wird.

UMOUNTCMD=

- ist dieser Parameter gesetzt, dann kann der eingetragene Befehl mittels CUxD auf der Statusseite ausgeführt werden (Umount-Taste). Es können auch mehrere Befehle mit Semikolon getrennt eingegeben werden. Bei einem Stromausfall wird dieser Befehl vor dem automatischen Deaktivieren der USB-Schnittstelle ausgeführt.

  Beispiel: *UMOUNTCMD=umount /dev/sda1*

BACKUPCMD=

- ist dieser Parameter gesetzt, dann können die eingetragenen Befehle mittels CUxD auf der Statusseite gestartet werden (SYS-Backup-Taste). Die Befehle werden als **neuer Prozess im Hintergrund** ausgeführt. Mehrere Befehle werden mit einem Semikolon voneinander getrennt.

  Beispiel:
  BACKUPCMD=cd /;tar cf /home/backup/ccuTS.tar usr/local var;sync
  oder für ein CCU-SBK-Backupfile:
  BACKUPCMD=/usr/local/addons/cuxd/extra/ccu_backup /home/backup

  Der Platzhalter „**TS**“ wird durch den aktuellen Zeitstempel ersetzt.

STARTUPCMD=

- Parameter mit Systembefehl, der bei jedem CUxD-Start ausgeführt wird. Dieser Parameter kann mehrfach vorhanden sein und z.B. genutzt werden, um neue (bisher unbekannte) Kernel-Module beim CUxD-Start zu laden oder /tmp Verzeichnisse für spezielle Anwendungen anzulegen,
  Beispiel: *STARTUPCMD=insmod ext2.ko*

TH-DIR=

- Dieser Parameter ist optional und zeigt auf ein Verzeichnis bzw. eine Datei auf der CCU zum Import von Dateien mit Temperatur/Luftfeuchte-Werten (z.B. durch eigene Scripts oder einen periodischen Aufruf von digitemp zum Auslesen von 1-Wire Temperatursensoren). Das Verzeichnis wird jede Sekunde ausgelesen. Gefundene Dateien werden verarbeitet und danach gelöscht. Steht am Ende des Parameters ein */*, dann handelt es sich um ein Verzeichnis. Ansonsten um eine Datei. Der Inhalt der Datei ist so definiert:

  «*ID*» «*Temperaturwert in °C*» «*Luftfeuchtewert in %*»

  Der Luftfeuchte-Wert ist optional. Beispieldatei für digitemp (z.B. *DS18S20*):

  ```
  0 22.35
  1 20.10
  2 21.00
  ```

  Die eingelesenen Werte werden im CUxD-internen Gerät (**CUX-THFILE:«*id*»**) mit der eingelesenen **ID** als Kanalnummer und dem Kanaltyp **WEATHER** bereitgestellt. Sie können über ein CUxD-Wrapper-Device zur CCU weitergeleitet werden. Werte mit gleicher **ID** werden beim Einlesen gefiltert, wobei immer nur der aktuellste Wert vom Gerät übertragen wird.

TH-DIR-FILTER=

- Dieser Parameter ist optional und definiert eine Liste mit durch Leerzeichen getrennten ungültigen Werten (maximal 5), die bei der Auswertung der durch TH-DIR= eingelesenen Daten ignoriert werden. Ein **T** vor dem Wert bedeutet Temperatur und ein **H** Luftfeuchte.
  Zum Beispiel bei digitemp 1-Wire: **TH-DIR-FILTER=**T85.00

SYSLOGMOVE=

- Zum automatischen Syslog-Backup kann mit diesem optionalen Parameter das File /var/log/messages.0 automatisch in das definierte Zielverzeichnis verschoben werden. Zusätzlich wird vor jedem CCU-Reboot das File /var/log/messages in dieses Verzeichnis verschoben. Das Verschieben erfolgt dabei in folgender Reihenfolge:
  1. Umbenennen der Datei in <name>.YYYYMMDD-HHMM, wenn die Zieldatei noch nicht existiert.
  2. Prüfen, ob das Zielverzeichnis existiert.
  3. Verschieben der Datei unter dem neuen Namen ins Zielverzeichnis, wenn sie dort noch nicht existiert.

SYSLOGMOVEDAILY=1

- dieser optionale Parameter erzwingt das tägliche Verschieben des CCU-Syslogs von /var/log/messages in das mit SYSLOGMOVE= definierte Verzeichnis unter dem Dateinamen *messages.YYYYMMDD-HHMM*.

LOGFILE=

- CUxD-Logfile für Programmanalyse und Debugging mit dem vollständigen lokalen Pfad auf der CCU. Vor jeder Zeile wird ein Zeitstempel ausgegeben.
  Achtung: Dieser Parameter sollte aus Performance-Gründen normal deaktiviert sein. Ein direktes Loggen auf den USB-Stick bzw. die SD-Karte kann Stabilitätsprobleme der CCU verursachen und sollte vermieden werden. Hier ist die Verwendung von **LOGFILEMOVE=** zu empfehlen.

LOGLEVEL=0

- dieser Parameter geht von 0 bis 9 und beschreibt den Umfang der Log-Einträge. Je größer der Wert, desto detaillierter und umfangreicher wird das Log.

LOGSIZE=+1000000

- erreicht das CUxD-Logfile die vorgegebene Größe, dann wird es in <name>.0 umbenannt und ein neues Logfile geschrieben. Dabei wird ein eventuell vorhandenes älteres Logfile überschrieben.
- Ist zusätzlich der Parameter LOGFILEMOVE= definiert, dann wird das Logfile bei überschreiten der Größe entsprechend verschoben. Bei '+' zusätzlich auch täglich um 0:00 Uhr.

LOGFILEMOVE=

- Dieser Parameter ist optional und enthält das Zielverzeichnis in welches LOGFILE täglich um 0:00 Uhr bzw. nach Überschreiten der Dateigröße und vor jedem CCU-Reboot verschoben wird. Das Verschieben erfolgt in folgender Reihenfolge:
  1. die Datei wird in <name>.0 umbenannt.
  2. Ein neuer Hintergrundprozess wird gestartet
     1. die Datei wird von <name>.0 in <name>.YYYYMMDD-HHMM umbenannt, wenn die Zieldatei noch nicht existiert.
     2. Prüfen, ob das Zielverzeichnis existiert.
     3. die Datei wird unter dem Namen <name>.YYYYMMDD-HHMM.$ ins Zielverzeichnis kopiert, wenn sie dort noch nicht existiert.
     4. die Ziel-Datei wird in <name>.YYYYMMDD-HHMM umbenannt
     5. die Quell-Datei wird gelöscht

TTY-Schnittstellenparameter

Diese Parameter dienen zur Konfiguration der USB-Schnittstelle und Verarbeitung der empfangenen Daten durch den CUx-Daemon. Als Parameter für die Schnittstelle kann sowohl das TTY (z.B. „ttyACM0“) als auch die USB-ID (z.B. „1-2.1“) genutzt werden.

CUXINITCMD=X21_

- Dieser Initialisierungsstring wird beim Verbinden an **alle** CUx-Geräte gesendet. Der Eintrag „X21“(mit RSSI-Daten) bzw. „X01“ (ohne RSSI-Daten) am Anfang ist Bedingung, damit die empfangenen Datenpakete richtig ausgewertet werden können. Es können auch zusätzliche Parameter angegeben werden, um z.B. den FHT-Hauscode zu setzten und CUx-Empfangseinstellungen zu verändern.
  Das „_“-Zeichen dient als Befehls-Trennung!
  Den FHT-Hauscode definiert man mit dem Befehl „T01XXXX“. Mit „T010000“ wird die FHT-Funktionalität auf dem CUL/CUN abgeschaltet.
  *Dieser Parameter wird vom Parameter **TTYINIT** überschrieben!*

TTYADD=

- unterstützt die Hardware neben den USB-Schnittstellen noch weitere serielle Schnittstellen, dann können sie über diesen Parameter manuell dem CUxD hinzugefügt werden. Dieser Parameter muss für jedes zusätzliche TTY angegeben werden (also ggf. mehrfach!).
  Beispiel: *TTYADD=ttyAMA0*

TTYPARAM=

- mit diesem Parameter können die Verbindungseinstellungen mit den TTY's angepasst werden.
- ist dieser **Parameter nicht vorhanden**, so werden per Default automatisch **alle** vorhandenen seriellen USB-Schnittstellen verbunden.
- ist dieser **Parameter vorhanden**, so werden vom CUxD nur die Schnittstellen verbunden, für die TTY.....-Parameter existieren. Dabei reicht es bereits aus, für **jede genutzte serielle Schnittstelle** einen TTYPARAM-Parameter anzugeben. Schnittstellen ohne TTY.....-Parameter werden nicht mit dem CUxD verbunden! Um **keine** Schnittstelle zu verbinden, kann auch ein beliebiges, nicht vorhandenes, TTY angegeben werden (z.B. TTYPARAM=NONE).
  Beispiel (manuelle Konfiguration):
  TTYPARAM=ttyACM0 ; Baudrate: Default
  TTYPARAM=ttyUSB0:9600:8N1 ; Baudrate: 9600 8N1
  TTYPARAM=ttyUSB0:9600:7E1 ; Baudrate: 9600 7E1
  TTYPARAM=ttyUSB0:9600:7O2 ; Baudrate: 9600 7O2
  *TTYPARAM=ttyUSB1:38400:8N1C ; Baudrate: 38400 8N1 (RTS**C**TS)*
  *TTYPARAM=ttyUSB1:19200:8N1X ; Baudrate: 19200 8N1 (**X**ON/XOFF)*
- die folgenden Baudraten werden unterstützt:
  50,150,300,600,1200,2400,4800,9600,19200,38400,57600,115200,230400

TTYHIDE=

- Parameter zum Ausblenden von TTYs im CUxD-Terminal. Die Daten werden trotzdem weiter empfangen und verarbeitet. Dieser Parameter kann mehrfach vorhanden sein.
  Beispiel: *TTYHIDE=ttyUSB0*

TTYHEX=

- Parameter zur Hexadezimal-Umwandlung der TTY-Daten. Für Geräte vom Typ CUX und WDE1 ist dieser Parameter deaktiviert. Dieser Parameter kann mehrfach vorhanden sein.
  Beispiel: *TTYHEX=ttyUSB0*

TTYRAW=

- Parameter um das TTY in den RAW-Mode zu setzen. Dieser Parameter kann mehrfach vorhanden sein und sollte normalerweise zusammen mit dem Parameter TTYHEX verwendet werden. Für Geräte vom Typ CUX und WDE1 ist dieser Parameter deaktiviert.
  Beispiel: *TTYRAW=ttyUSB0*

TTYASSIGN=

- Parameter zum Überschreiben der automatischen Geräteerkennung. Auf der Statusseite wird bei jedem USB-Gerät der erkannte bzw. zugewiesene Gerätetyp in geschweiften Klammern angezeigt (verfügbare Gerätetypen sind CUX, WDE1, SONIC, ESP2, ESP3, ARDU, NONE). Dieser Parameter kann mehrfach vorhanden sein.
  Beispiel: *TTYASSIGN=ttyUSB0:CUX*

TTYINIT=

- Dieser Initialisierungsstring wird beim Verbinden des Gerätes an das definierte Gerät gesendet. Bei CUx-Geräten ist der Eintrag „X21“(mit RSSI-Daten) bzw „X01“ (ohne RSSI-Daten) zur Initialisierung notwendig, damit Datenpakete richtig empfangen werden können. Folgende Sonderzeichen werden vor dem Senden nach C-Standard ersetzt: \\, \a, \b, \f, \t, \n, \r, \v, \xHH (HH ist eine 2-stellige Hexadezimalzahl). Endet die Zeile mit einem einzelnen Backslash \, dann wird am Zeilenende kein CRLF gesendet. Mehrere Parameter (Zeilen) sind auch möglich.
  Beispiel: *TTYINIT=ttyUSB0:X21\nT010000*

TTYMAXIDLE=

- Mit diesem Parameter können die Minuten der Inaktivität angegeben werden, bevor die serielle Schnittstelle automatisch getrennt und wieder verbunden wird. Zusätzlich wird beim disconnect die DTR-Leitung auf LOW gesetzt um z.B. bei einem Arduino einen Reset auszulösen. (z.B. sinnvoll beim USB-WDE1)
  Beispiel: *TTYMAXIDLE:ttyUSB0:10*

8 Daten-Logging

Das Logging von beliebigen CCU-Geräten stellt eine Alternative zu dem leider sehr ressourcenintensiven und unsicheren Logging über HomeMatic-Scriptbefehle dar.

Beim Logging werden die Daten vom CUxD in eine Datei geschrieben.

Zum Aktivieren des Logs muss der Parameter „**DEVLOGFILE=**“ auf einen Dateinamen mit vollständigem Pfad gesetzt werden. Ist dieser Parameter leer, so ist das Logging deaktiviert.

Die Log-Datei kann über die CUxD-Adminoberfläche unter „**Info** → **Device-Log**“ ausgelesen werden und mit dem CUxD-Highcharts Addon können die geloggten Datenpunkte grafisch dargestellt werden.

Es wird nicht geprüft, ob genügend Platz auf dem Zielverzeichnis verfügbar ist. Auf der RAM-Disk der CCU1 (/var/ bzw. /tmp/) stehen maximal 32MB zur Verfügung!

Über den Parameter „**DEVLOGSIZE=**“ kann die Maximalgröße des Logfiles in Bytes festgelegt werden, ab der sie automatisch gekürzt wird. Bei 0 wird die Datei nicht gekürzt.

Mit dem optionalen Parameter „**DEVLOGMOVE=**“ kann das Logfile täglich um 0:00 Uhr umbenannt bzw. umbenannt und verschoben werden. Dabei wird ans Ende der Datei immer ein Zeitstempel im Format „.YYYYMMDD-HHMM“ angefügt. Soll die Datei dabei von der CCU in ein gemountetes Verzeichnis (USB-Stick o.ä.) verschoben werden, dann sollte man dafür nicht das Wurzelverzeichnis des Ziel-Volumes, sondern ein Unterverzeichnis wählen. Ist der USB-Stick einmal nicht gemountet wird so verhindert, dass die Datei zum Mount-Point verschoben wird.

Der periodische Export (NFS, FTP, …) von Log-Daten auf einen externen Host zur weiteren Verarbeitung ist über den Parameter **DEVLOGEXPORT=** über ein frei definierbares Script möglich.

CCU1-interne Systemmeldungen (Batterie, Netzteil, Sabotage usw.) werden auch an den CUxD weitergeleitet und können geloggt werden. Will man HM-Geräte loggen, so sind zusätzlich die Parameter „**SUBSCRIBE_RF=1**“ (Funk) und/oder „**SUBSCRIBE_WR=1**“ (Wired) zu setzen. Erst danach werden die entsprechenden Meldungen zum CUxD übertragen und stehen dem Log-Filter zur Verfügung.

Das Filtern der zu „loggenden“ Geräte erfolgt über den Parameter „**LOGIT=**“. Dieser Parameter kann beliebig oft wiederholt werden und definiert die zu loggenden Datenpunkte der Geräte mit einem optionalen Alias. Die Reihenfolge der Parameter ist wichtig. Sobald ein Filter übereinstimmt, werden alle folgenden Parameter übersprungen. Um das Logging für bestimmte Datenpunkte auszuschließen, kann als DP oder ALIAS ein „**!**“ eingegeben werden.

Zum Loggen von Datenpunkten, die nur bei Wertänderungen und relativ selten übertragen werden, wie z.B. Thermostat-Solltemperaturen (SETPOINT) eignet sich ein periodischer (z.B. jeder Stunde) Aufruf der folgenden HM-Befehle:

```
string s = "HEQxxxxxxx:2.SETPOINT";
var v = dom.GetObject("BidCos-RF."#s).Value();
dom.GetObject("CUxD.CUX2801001:1.LOGIT").State(s#";"#v);
```

Im Beispiel ist die Seriennummer des zu loggenden Gerätes entsprechend zu ergänzen und das CUxD-System.Exec() Gerät mit der Nummer 1 muss angelegt sein.

Jeder **LOGIT**-Eintrag kann 3 Elemente enthalten:

1. die Seriennummer oder einen Teil davon aus der CCU-Gerätekonfiguration mit abschließendem Channel.
2. Den internen Namen des zu loggenden Datenpunktes. Der Name muss genau übereinstimmen. (Groß-/Kleinschreibung beachten!)
3. Einen Alias, der in das Logfile geschrieben wird (der Alias darf keine Leerzeichen enthalten)

Die Elemente müssen durch mindestens 1 Leerzeichen voneinander getrennt werden.

Alle korrekt erkannten Log-Einträge werden am Ende der CUxD-Statusseite angezeigt.

Das Format ist so definiert: **LOGIT=***[DEVICE][:[CHANNEL]] [DP [ALIAS]]*

Beispiele:

LOGIT=CUX3200001:1 TEMPERATURE T1
- es wird der Wert für CUX3200001:1.TEMPERATURE unter dem Alias T1 geloggt

LOGIT=GEQ0051630:1 HUMIDITY
- es wird der Wert für GEQ0051630:1.HUMIDITY geloggt

LOGIT=GEQ0051630:2
- es werden alle DPs von Gerät GEQ0051630:2 geloggt

LOGIT=GEQ0127760
- es werden alle DPs von Gerät GEQ0127760 geloggt

LOGIT=GEQ0041764 STATE
- es werden alle STATE DPs von Gerät GEQ0041764 geloggt

LOGIT=CUX TEMPERATURE
- es werden alle TEMPERATURE DPs von den Geräten CUX..... geloggt

LOGIT=:0 RSSI_PEER !
- ab hier werden alle RSSI_PEER-DPs von Channel 0 für weitere LOGIT-Parameter ignoriert

LOGIT=:0
- es werden alle DPs von Channel 0 aller Geräte geloggt

LOGIT=:0 UNREACH
- es werden alle UNREACH DPs von Channel 0 aller Geräte geloggt

LOGIT=:0 !
- ab hier werden alle DPs von Channel 0 für weitere LOGIT-Parameter ignoriert

LOGIT=CUX
- es werden alle DPs von allen Geräten mit CUX am Anfang der Seriennummer geloggt

LOGIT=:
- es werden alle DPs von allen Geräten geloggt (zum Debugging!)

Im Logfile besteht jeder Log-Eintrag immer aus 3 Elementen:

1. Datum/Uhrzeit (siehe **DEVTIMEFORMAT**),
2. vollständiger Name des Datenpunktes bzw. Alias,
3. Wert des Datenpunktes (siehe **DEVDATAFORMAT**).

Damit die Log-Einträge einfach für eigene Auswertungen nutzbar sind (z.B. klassische CSV-Datei mit Semikolon oder Tabulator als Spaltentrenner) besteht die Möglichkeit, die Formatierung der Datenzeile entsprechend anzupassen.

Der Parameter **DEVTIMEFORMAT** muss in einfache ' (Anführungszeichen) eingeschlossen werden.

Folgende Platzhalter werden unterstützt:

spec.	Replaced by	Example
%a	Abbreviated weekday name *	Thu
%A	Full weekday name *	Thursday
%b	Abbreviated month name *	Aug
%B	Full month name *	August
%c	Date and time representation *	Thu Aug 22 14:23:56 2013
%d	Day of the month (01-31)	22
%H	Hour in 24h format (00-23)	14
%I	Hour in 12h format (01-12)	02
%j	Day of the year (001-366)	234
%m	Month as a decimal number (01-12)	08
%M	Minute (00-59)	23
%p	AM or PM designation	PM
%S	Second (00-61)	56
%U	Week number with the first Sundayas the first day of week one (00-53)	33
%w	Weekday as a decimal number with Sunday as 0 (0-6)	4
%W	Week number with the first Monday as the first day of week one (00-53)	34
%x	Date representation *	08/22/13
%X	Time representation *	14:23:56
%y	Year, last two digits (00-99)	13
%Y	Year	2013
%Z	Timezone name or abbreviation	CDT
%%	A % sign	%

Wird der Parameter leer gelassen, so wird automatisch der UNIX-Timestamp (Sekunden seit 1.1.1970) verwendet.

Der Parameter **DEVDATAFORMAT** muss in einfache ' (Anführungszeichen) eingeschlossen werden und definiert die Formatierung der beiden Datenwerte. Er kann auch leer sein.

2 Platzhalter dürfen verwendet werden:

1. Der DP-Name bzw. Alias aus der LOGIT-Einstellung wird durch den Platzhalter **%s** dargestellt. Hier gibt es noch die Möglichkeit die minimale Länge des Textfeldes zu definieren, z.B. bedeutet %10s - solange der Alias kleiner als 10 Zeichen ist wird er mit 10 Zeichen rechtsbündig dargestellt (auffüllen mit Leerzeichen).
2. Der Datenwert muss als Fließkommazahl formatiert werden z.B. bedeutet **%.1f** eine Stelle Genauigkeit (nach dem Dezimalpunkt). Zwischen den Platzhaltern können Zeichen eingefügt werden (z.B. „;“ oder „\t“), um entsprechend formatierte CSV-Dateien zu erhalten.

8.1 CUxD-HighCharts

Zur grafischen Darstellung der Logdaten kann direkt auf der CCU das CUxD-HighCharts-Addon genutzt werden.

Beispielkonfiguration mit Ablage der Logfiles auf der **CCU2** SD-Karte:

1. CUxD- und HighCharts-AddOn installieren
2. SD-Karte per WebUI initialisieren
3. auf CUxD-Service-Seite
   -> Shell Command: **`mkdir -p /media/sd-mmcblk0/cuxd/devlog`**
   -> Ausführen
4. im CUxD-Setup die folgenden Parameter setzen (dürfen nur 1x vorhanden sein!):
   `DEVLOGFILE=/tmp/devlog.txt`
   `DEVLOGSIZE=`
   `DEVLOGMOVE=/media/sd-mmcblk0/cuxd/devlog`
5. die gesetzten Parameter auf der CUxD-Statusseite überprüfen
6. Aufruf der Diagramme über CUxD -> Info -> Device-Log -> Chart

9 DFU Firmware-Installation/Update über den CUx-Daemon

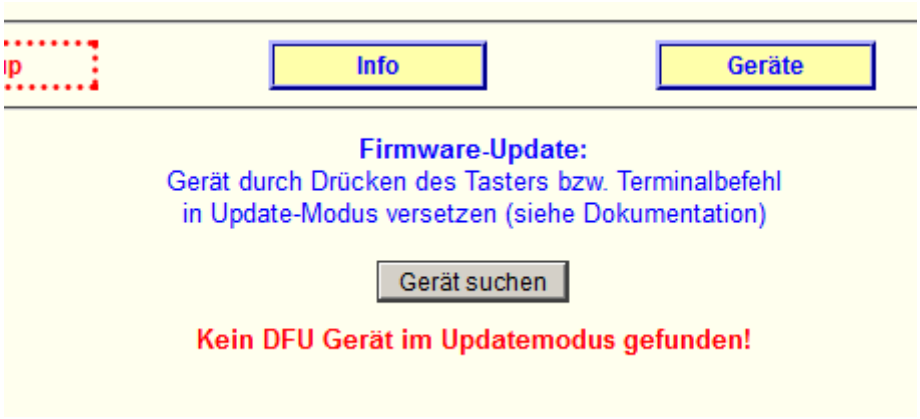
Original werden die CUN/CUL/EUL-Module (http://busware.de) ohne Firmware ausgeliefert. Um diese Module mit dem CUxD nutzen zu können, muss zuerst die culfw-Firmware (http://culfw.de) aufgespielt werden. Ein neues „leeres“ Modul befindet sich nach dem Verbinden automatisch im Update-Mode.
Der CUx-Daemon bietet die Möglichkeit, die Firmware von DFU-Geräten mittels dfu-programmer direkt aus dem Web-Browser heraus über die CCU zu flashen. Dafür ist kein weiterer Rechner (Windows / Linux) erforderlich.

Für das Flashen der Firmware wird eine modifizierte Version des dfu-programmers (0.5.2) http://dfu-programmer.sourceforge.net/ mit atmega32u2 Unterstützung (für CUL V4) verwendet. Dieses Programm ist ist ebenfalls im CUxD-Paket enthalten.

Der CUx-Daemon bietet über die „Setup“-Seite der Administrations-Weboberfläche ein einfaches Interface zur Bedienung des dfu-programmers.

Die „Setup“-Seite ist in 2 Bereiche geteilt:

- die linke Seite dient zur Einstellung der Parameter des CUx-Daemon (ini-Datei),
- die rechte Seite ist das Interface zum Firmwareupdate für DFU-Geräte.



Der CUx-Daemon erkennt die angeschlossenen Geräte und deren Betriebsmodus. Im „normalen“ Betrieb kann die Firmware nicht aktualisiert werden. Dazu muss das entsprechende Modul zuerst in den „**Updatemode**“ versetzt werden.

Das CUN/CUL/EUL-Modul startet z.B. im Update-Mode, wenn der Mikroschalter mit einem spitzen Gegenstand gedrückt wird, **während** das Modul mit dem USB-Anschluss verbunden wird. Sollte das nicht sofort funktionieren, dann ist der Mikroschalter auch nach dem Verbinden noch etwas gedrückt zu halten.

Ist bereits eine ältere culfw-Version auf dem CUL/CUN-Modul installiert, dann kann das Modul auch per Terminalbefehl (z.B. „BBB“) in den Update-Mode versetzt werden.

Erst danach erscheint nach manuellem Refresh mittels „**Gerät suchen**“ im CUx-Daemon die Anzeige:

Je nach angeschlossenem Gerät wird ein anderer Mikrocontroller (in grün) mit passender Firmware-Auswahlliste angezeigt. In der Auswahlliste werden alle Dateien angezeigt, die sich im entsprechenden ZIP-File befinden.

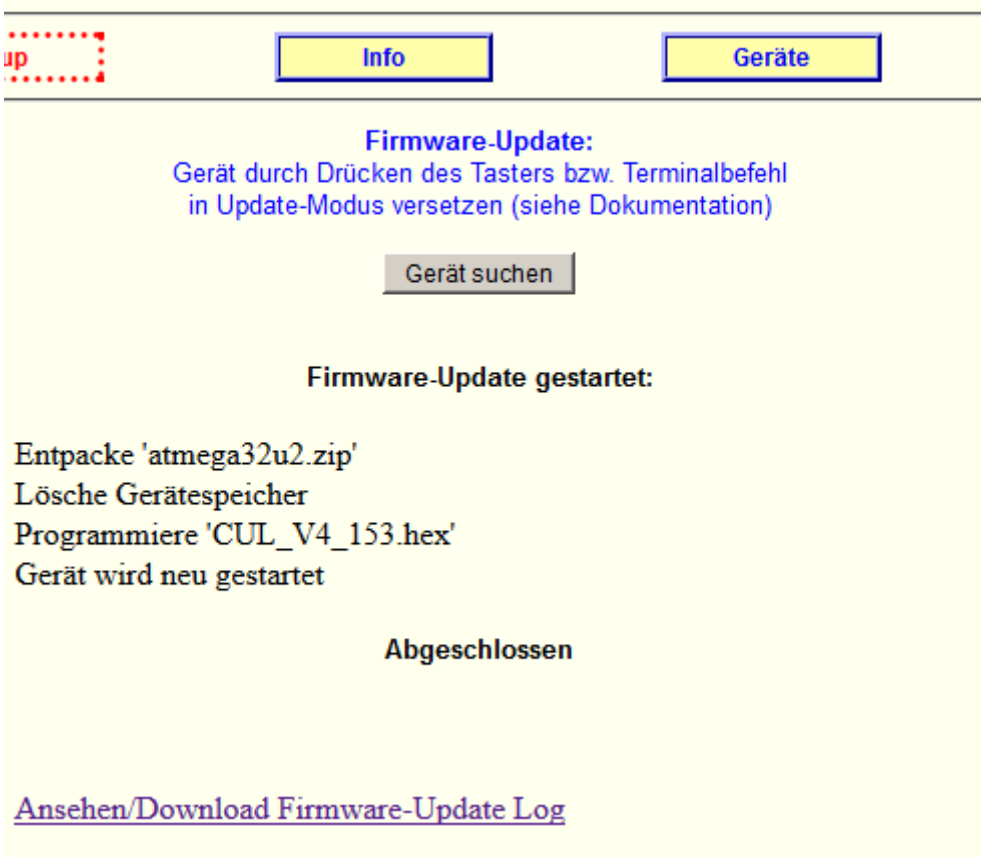
Eine Auswahl von ZIP-Files mit Firmware-Dateien befinden sich auf der CCU im Verzeichnis /usr/local/addons/cuxd/dfu. Sie können bei Bedarf z.B. per FTP unabhängig vom CUxD aktualisiert und erweitert werden.

Neben der aktuellen CUL-Firmware befindet sich auch die Firmware für den busware EUL-Stick im Archiv. Diese EUL-Firmware ermöglicht den Betrieb des EUL-Sticks als EnOcean-Repeater, wenn er lediglich in ein standalone USB-Netzteil gesteckt wird.

Aktuell werden vom CUxD die folgenden Mikrocontroller unterstützt: at90usb162, at90usb646, at90usb1286, atmega32u4 und atmega32u2.

Nach dem Starten über den Firmwareupdate-Button werden automatisch folgende Schritte abgearbeitet:

- Entpacken der ausgewählten Firmware aus der Archiv-Datei
- Löschen des Flash-Speichers vom Modul
- Programmieren des Flash-Speichers vom Modul
- Neuinitialisierung des Moduls



Das Logfile des Firmware-Updates kann über den angezeigten Link kontrolliert werden. Bis auf die Meldung „**... could not release interface ...**“ dürfen keine weiteren Fehlermeldungen auftauchen.

Nach dem Update ist das Modul normalerweise sofort wieder betriebsbereit und ohne Neustart im CUx-Daemon verfügbar. Sollte das nicht der Fall sein, dann hilft ein kurzzeitiges trennen des Moduls vom USB-Anschluss um einen Neustart der Firmware zu erzwingen.

10 FAQ

1. **Was bedeuten LOVF-Meldungen vom CUL/CUN/CUNO?**
   Um Störeinflüsse zu minimieren, ist auf dem 868,3MHz Frequenzband eine Begrenzung der Sendezeit auf maximal 1% vorgeschrieben. Das entspricht ungefähr 160 FS20-Befehlen pro Stunde. Die culfw hält sich an dieses Limit. Sollen mehr Daten in kurzer Zeit gesendet werden, dann begrenzt das die culfw und liefert einen LOVF-Fehler (Limit OVerFlow). Dieses Limit kann auch unbeabsichtigt, z.B. durch interne FHT-Kommunikation im Hintergrund, erreicht werden.
   Nach einiger Zeit geht's dann wieder. Die verfügbare Sendezeit kann man im CUxD-Terminal mit dem „X"-Befehl (2. Zahl multipliziert mit 10 ms) ansehen. Eine FS20 Übertragung benötigt zum Beispiel ca. 210ms.
   Siehe auch: http://fhemwiki.de/wiki/1%25_Regel
2. **Mein FS20-Relais-Aktor funktioniert nicht, obwohl er richtig konfiguriert ist.**
   Damit der FS20-Relais-Aktor funktioniert, muss er zuvor manuell (WebUI: „Status und Bedienung" → Geräte → Aktor → Ein), per Programmverknüpfung oder per Script aktiviert (eingeschaltet) werden.
3. **Wie werden die ELV-FS20-Codes hexadezimal umgerechnet?**
   *Man teilt den ELV-Code von links nach rechts in Gruppen mit jeweils **zwei** Zeichen und übersetzt diese dann nach folgender Tabelle in jeweils **ein** hexadezimales Zeichen (funktioniert in beide Richtungen):*

ELV-FS20-Code (2 Zeichen)	Hexadezimal (1 Zeichen)
11	0
12	1
13	2
14	3
21	4
22	5
23	6
24	7
31	8
32	9
33	A
34	B
41	C
42	D
43	E
44	F

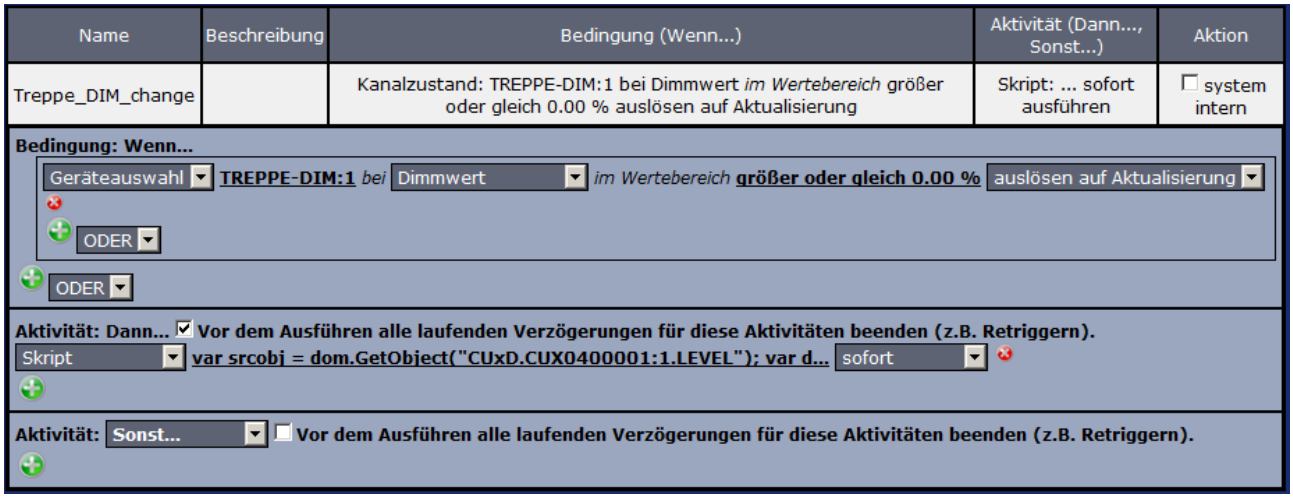
4. **Das Update der CCU-Firmware schlägt fehl.**
   Wurde über den CUxD ein USB-Stick gemountet, dann kann es vorkommen, das das CCU-Firmware Update fehlschlägt.In diesem Fall könnte es helfen, den CUxD einfach vor dem Firmware-Update über die Adminoberfläche zu beenden. Dabei werden auch alle konfigurierten USB-Sticks „abgehängt".
   Eine weitere Möglichkeit ist das Update nach dem Neustart der CCU im abgesicherten Modus.

5. Synchronisation von CUxD-Datenpunkten mit Systemvariablen

Ein Datenpunkt kann über Programmverknüpfungen mit einer Systemvariable synchronisiert werden.

Beispiel: Der Dimmwert des FS20-Aktors **TREPPE-DIM** (CUX0400001) soll mit der Systemvariable **Treppe_DIM** synchronisiert werden...

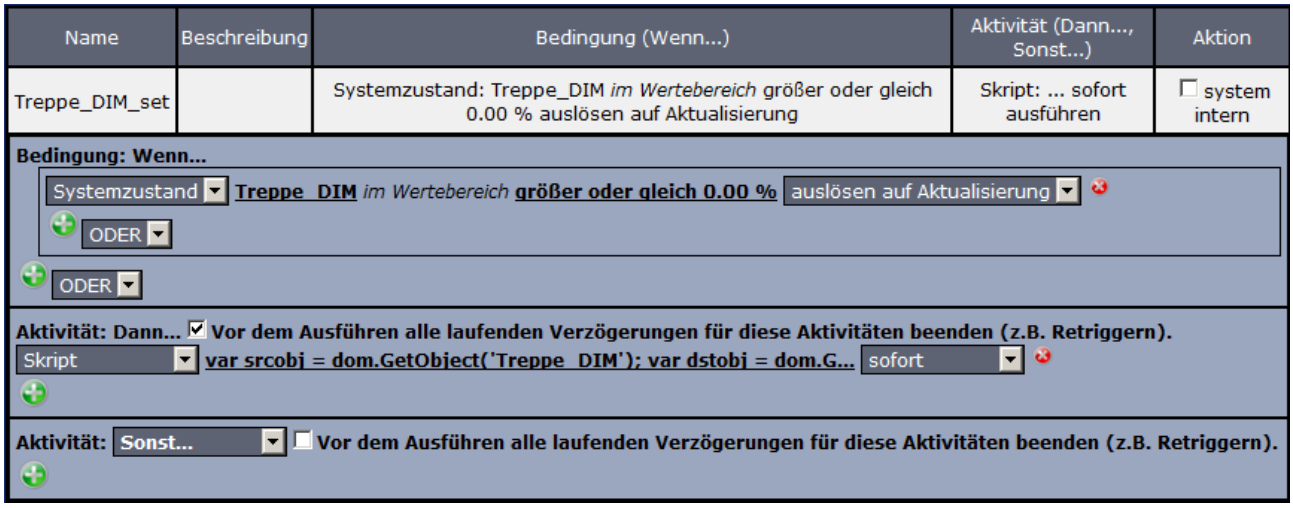
1. Programmverknüpfung (**Gerät → Systemvariable**):



HM-Script:

```
var srcobj = dom.GetObject("CUxD.CUX0400001:1.LEVEL");
var dstobj = dom.GetObject("Treppe_DIM");
var srcval = srcobj.Value() * 100;
if (srcval != dstobj.Value()) {
  dstobj.State(srcval);
}
```

2. Programmverknüpfung (**Systemvariable → Gerät**):



HM-Script:

```
var srcobj = dom.GetObject("Treppe_DIM");
var dstobj = dom.GetObject("CUxD.CUX0400001:1.LEVEL");
var srcval = srcobj.Value() / 100;
if (srcval != dstobj.Value()) {
  dstobj.State(srcval);
}
```

6. **Muss bei jedem Start vom CUxD ein culfw-Update durchgeführt werden?**
   Nein! Die culfw wird bei einem Update fest in den Flash-Speicher des CUL bzw. CUN geschrieben (http://culfw.de/culfw.html).
7. **Anstelle der Anzeige der CUxD-Weboberfläche wird beim Aufruf der URL zum Abspeichern einer Datei aufgefordert.**
   Die CUxD-Weboberfläche wird dynamisch durch ein CGI-Script auf der CCU generiert. Ist der lighthttpd-Server auf der CCU fehlerhaft konfiguriert, dann wird dieses CGI-Script mit der Dateiendung '.ccc' unter Umständen nicht ausgeführt, sondern dem Client zum Download angeboten.
   Ursache dieses Fehlers könnte die Installation eines alten bzw. fehlerhaften AddOns gewesen sein.
   Ein Upgrade oder das erneute Aufspielen der installierten CCU-Firmware (ab Version 1.503) beseitigt dieses Problem.
8. **Wie kann ein CUNO per USB an z.B. ttyUSB0 angebunden werden? Die automatische Erkennung funktioniert nicht.**
   Dazu sind im CUxD-Setup die folgenden Parameter zu setzen:
   TTYPARAM=ttyUSB0:38400:8N1
   TTYASSIGN=ttyUSB0:CUX
9. **Im CUxD-Syslog steht die Info „USB(.../ttyUSB0) connect device disabled!“ und das Gerät wird zwar richtig erkannt aber nicht automatisch connected.**
   Es wurden TTY-Parameter für andere USB-Geräte definiert. Danach werden nicht mehr alle angeschlossenen, sondern nur noch die im Setup konfigurierten Geräte connected. Das Hinzufügen eines TTYPARAM-Parameters (im Beispiel ***TTYPARAM=ttyUSB0)*** *für das betreffende Gerät behebt das Problem.*
10. **Die WebUI ist abgestürzt und CUxD ist installiert. Besteht die Möglichkeit, die CCU bzw. CUxD irgendwie neu zu starten?**
    Ja,über die URL (http://<ip_der_ccu>/addons/cuxd/maintenance.html) ist jederzeit ein CCU-Neustart sowie auch ein CUxD Neustart möglich.
11. **Die WebUI startet nicht und CUxD ist installiert. Besteht die Möglichkeit, Telnet zu aktivieren um die Konfiguration manuell zu reparieren?**
    Ja,über die URL (http://<ip_der_ccu>/addons/cuxd/maintenance.html) können beliebige Shell-Befehle auf der CCU ausgeführt werden. Auf der CCU1 kann Telnet über den folgenden Befehl manuell gestartet werden:
    /usr/sbin/inetd
12. **Die letzte Aktualisierung in der WebUI wird nicht richtig angezeigt.**
    Die Spalte in der WebUI müsste richtig „letzte Änderung“ heißen. In der WebUI ändert sich der Zeitstempel der letzten Aktualisierung eines Kanals nur, wenn sich die Werte der zugehörenden Datenpunkte geändert haben. Wurden jedoch die gleichen Werte erneut gesendet (wie z.B. bei Temperatur oder Energiesensoren) dann aktualisiert die WebUI diesen Zeitstempel nicht.
13. **Zugriff auf die SD-Karte mit CCU2-Firmware ab Version 2.7.8.**
    Um auf die automatisch gemountete SD-Karte auch über einen anderen Pfad (z.B. /mnt) zugreifen zu können, kann folgendes als mount-Befehl im CUxD-Setup eingetragen werden:
    MOUNTCMD=mount -o bind /media/sd-mmcblk0 /mnt
    UMOUNTCMD=umount /mnt
    Der Zugriff über einen symbolischen Link auf den Pfad /tmp/sd/ wäre z.B. so:
    MOUNTCMD=ln -s /media/sd-mmcblk0 /tmp/sd
    UMOUNTCMD=rm -f /tmp/sd