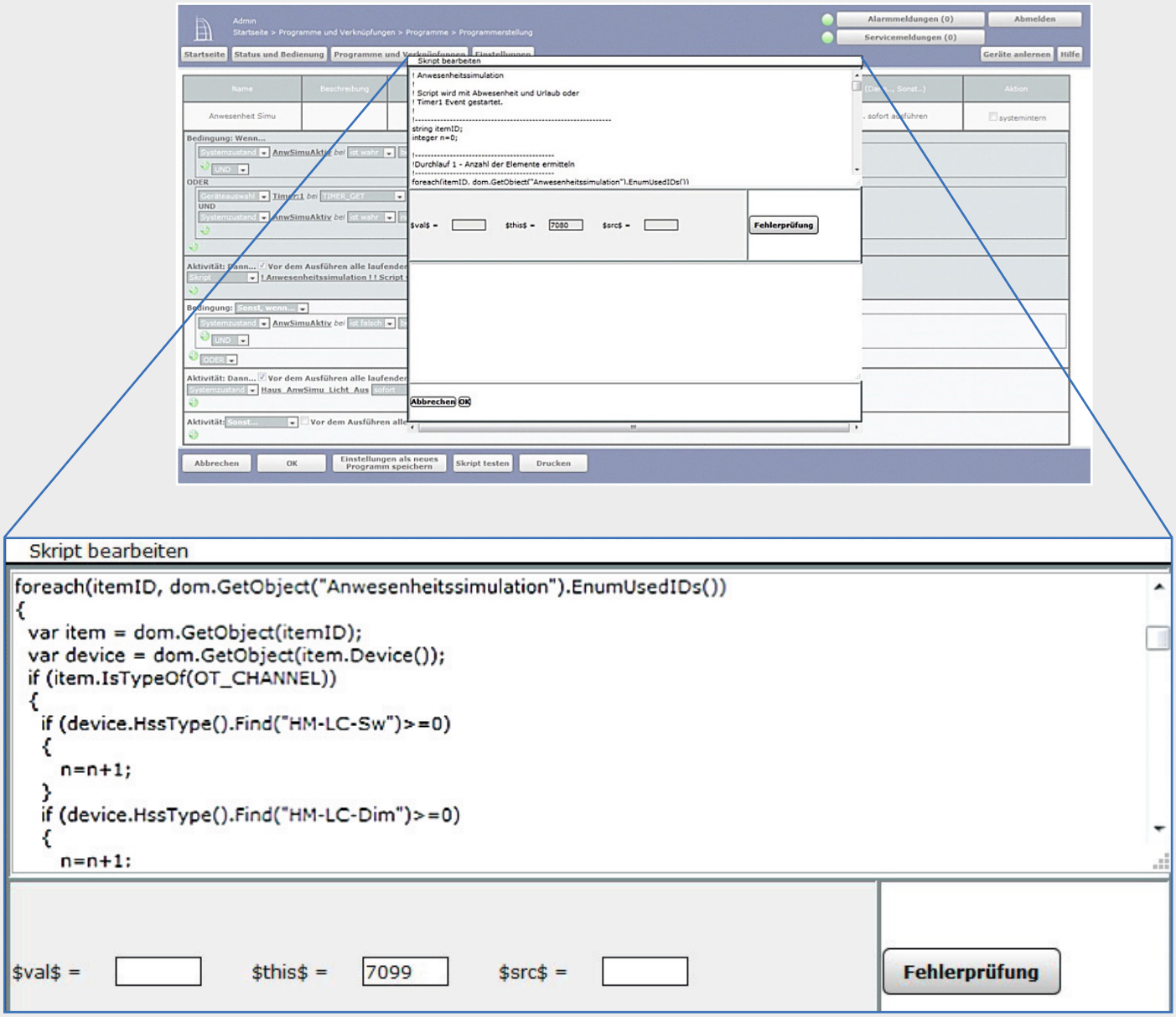
Homematic Scriptprogrammierung

Teil 8 – Infodarstellung auf mediola Interface und dazugehörige Scripte



Im achten Teil der Artikelserie beschäftigen wir uns nochmals mit der Scriptprogrammierung im Zusammenhang mit dem mediola Gateway und dem mediola Creator und wir schreiben weitere Scripte.

Nützliche und notwendige Informationen

Mit der mediola Oberfläche ist es möglich, eine eigene selbst designte Bedienungs- und Informationsoberfläche für ein Pad oder ein Smartphone zu „bauen".

Um das Ganze übersichtlich und anwenderfreundlich zu machen, ist eine gute Planung bezüglich der Inhalte und Darstellungen unumgänglich. Besonders wichtig sind dabei übersichtliche Informationsseiten, mit deren Hilfe man sich sehr schnell und einfach einen Überblick über Zustände oder Fehler der Anlage verschaffen kann.

Als aktiver „Anlagenbauer" und „Anlagenprogrammierer" hat man einen anderen Blick denn als reiner Anwender, und die Akzeptanz und die Beurteilung eines solchen Systems steht und fällt mit der Übersichtlichkeit und Klarheit der Darstellung.

Da in einer Homematic Hausinstallation sehr viele Sensoren mit Batteriebetrieb verbaut sind, wollen wir in diesem Artikel exemplarisch auf das Thema der Visualisierung von Batteriefehlern näher eingehen.

Batteriefehler

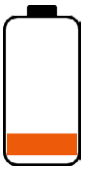
Eine ganze Reihe von Homematic Geräten arbeitet mit Batterien. Die meisten dieser Geräte erlauben die Abfrage eines Parameters „Batteriefehler" bzw. „LowBat", aber leider nicht alle. So gibt es beispielsweise Geräte, deren Batteriezustand über den Datenpunkt „Fault Report" abzufragen ist. Genauso findet man in der mediola Oberfläche nicht immer den Parameter „Batteriefehler" oder „Batterie leer".

Abhilfe: Wenn für die Homematic Geräte mit Batterien Systemvariablen (Zentralenvariable) für den Batteriezustand definiert werden, können diese mit dem folgendem Script gesetzt bzw. zurückgesetzt werden – je nach Batteriezustand.

Damit kann sehr einfach eine Übersicht aufgebaut werden:

Volle Batterien werden mit diesem Symbol dargestellt.

Bei einer (demnächst) leeren Batterie wird dieser rote Füllstandsbalken dargestellt.



Damit ist sofort ersichtlich, welche Batterie bald leer ist, bzw. über die Darstellung im Gesamtbild, um welches Gerät es sich handelt.

Alle grafischen Elemente dieser Seite können mit den unterschiedlichsten Programmen (Paint, Gimp, PowerPoint ...) erstellt bzw. gezeichnet werden. Der Hintergrund ist im Beispiel einfach ein png-File des Grundrisses.

Die Homematic Geräte lassen sich von der Homepage kopieren und in den mediola Creator importieren. Hier nun das Script (der erste Teil des Scripts):

```
01  var myAssembly = dom.GetObject("Batteriebetrieb");
02  string itemID;
03  string text = "";
04  boolean condition = false;
05
```

In der Variablen *myAssembly* steht der Name des Gewerkes für den Batteriebetrieb.

Die anderen (Script-) Variablen sind zur Scriptlaufzeit benötigte Variablen. *Condition* wird verwendet, um innerhalb des Scriptes den Batteriezustand zwischenzuspeichern.

condition = true → Batteriefehler
condition = false → kein Batteriefehler

```
06  foreach(itemID, myAssembly.EnumUsedIDs()) {
07      var item = dom.GetObject(itemID);
08      var device = dom.GetObject(item.Device());
09      var interface = dom.GetObject(item.Interface());
10      string interface_name = interface.Name();
11      string device_address = device.Address();
12      device_address = device_address.StrValueByIndex(":", 0);
13      if (device.HssType().Find("HM-CC-RT-DN")>=0)
14      {
15         string channel_name = interface_name # "." # device_address #
16         var channel = dom.GetObject(channel_name);
17         if ( (channel.State() == 6) )
18         {
19             condition = true;
20         }
21         else
22         {
23             condition = false;
24         }
25      }
26      else
27      {
28         string channel_name = interface_name # "." # device_address # ":0.LOWBAT";
29         var channel = dom.GetObject(channel_name);
30         if ( (channel.State() == true) && (channel.Value() == true) )
31         {
32            condition = true;
33         }
34         else
35         {
36            condition = false;
37         }
38      }
```

Nach der Definition der für das Script benötigten Variablen (Zeile 1 bis 4) wird in einer Schleife (Zeile 6) das Gewerk „Batteriebetrieb" Gerät für Gerät abgefragt.

Im Bereich der Zeilen 15 bis 24 werden die Heizungsventile auf den Zustand 6 des Datenpunktes „fault reporting" und im Bereich der Zeilen 28 bis 38 alle anderen Geräte auf Batteriefehler überprüft und die Variable *condition* dementsprechend gesetzt.

Im folgenden Scriptteil, der immer noch innerhalb der Schleife über das Gewerk „Batteriebetrieb" abläuft, wird je nach Namen des aktuell in der Schleife befindlichen Gerätes eine Systemvariable (Zentralenvariable) gesetzt oder rückgesetzt, die dann schlussendlich im Frontend mediola über das Batteriesymbol visualisiert wird:

```
39
40      text = item.Name().Substr(0,item.Name().Find(":"));
41      !----------------
42      ! 2-fach Taster
43      !----------------
44      if (text ==
"Wohnzimmer*2_fach_Taster*04"){dom.GetObject("Batt_Fehl_WZ_2T_04").State(condition
);}
45      if (text ==
"Wohnzimmer*2_fach_Taster*01"){dom.GetObject("Batt_Fehl_WZ_2T_01").State(condition
);}
46      if (text ==
"Wohnzimmer*2_fach_Taster*02"){dom.GetObject("Batt_Fehl_WZ_2T_02").State(condition
);}
47      !----------------
48      ! 6-fach Taster
49      !----------------
50      if (text ==
"Wohnzimmer*6_fach_Taster*01"){dom.GetObject("Batt_Fehl_WZ_6T_01").State(condition
);}
```

```
39
40      text = item.Name().Substr(0,item.Name().Find(":"));
41      !----------------
42      ! 2-fach Taster
43      !----------------
44      if (text == "Wohnzimmer*2_fach_Taster*04"){dom.GetObject("Batt_Fehl_WZ_2T_04").State(condition);}
45      if (text == "Wohnzimmer*2_fach_Taster*01"){dom.GetObject("Batt_Fehl_WZ_2T_01").State(condition);}
46      if (text == "Wohnzimmer*2_fach_Taster*02"){dom.GetObject("Batt_Fehl_WZ_2T_02").State(condition);}
47      !----------------
48      ! 6-fach Taster
49      !----------------
50      if (text == "Wohnzimmer*6_fach_Taster*01"){dom.GetObject("Batt_Fehl_WZ_6T_01").State(condition);}
51      if (text == "Wohnzimmer*6_fach_Taster*02"){dom.GetObject("Batt_Fehl_WZ_6T_02").State(condition);}
52      if (text == "Wohnzimmer*6_fach_Taster*03"){dom.GetObject("Batt_Fehl_WZ_6T_03").State(condition);}
53      if (text == "Wohnzimmer*6_fach_Taster*04"){dom.GetObject("Batt_Fehl_WZ_6T_04").State(condition);}
54      if (text == "Buero*6_fach_Taster*01"){dom.GetObject("Batt_Fehl_Buero_6T_01").State(condition);
        }
55
56      !-----------------------------
57      ! Innen-Bewegungsmelder
58      !-----------------------------
59      if (text == "Wohnzimmer*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_WZ_BewMldg_1").State(con
        dition);}
60      if (text == "Wohnzimmer*Bewegungsmelder*02"){dom.GetObject("Batt_Fehl_WZ_BewMldg_2").State(con
        dition);}
61      if (text == "WC_oben*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_WCoben_Bew_1").State(condit
        ion);}
62      if (text == "Treppe_unten*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_TreUnten_Bew_1").State
        (condition);}
63      if (text == "1.OG*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_OG1_Bew_1").State(condition);}
64      if (text == "2.OG*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_OG2_Bew_1").State(condition);}
65      if (text == "Flur_Eingang*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_FlurEing_Bew_1").State
        (condition);}
66      if (text == "Garten*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_Garten_Bew_1").State(conditi
        on);}
67      if (text == "Garten*Bewegungsmelder*02"){dom.GetObject("Batt_Fehl_Garten_Bew_2").State(conditi
        on);}
68      if (text == "Schlafzimmer*Bewegungsmelder*01"){dom.GetObject("Batt_Fehl_SchlZim_Bew_1").State(
        condition);}
69      }
```

Phasen, Bereiche, Jahreszeiten etc.

Es gibt zahlreiche Phasen über den Zeitraum eines Jahres hinweg, die Dinge in unserem Smart Home steuern. Zu nennen wären beispielsweise:

die Jahreszeiten → steuern beispielsweise unterschiedliche Einstellungen für die Heizung
Bewässerungszeit → zu dieser Zeit ist die automatische Gartenbewässerung aktiv
Weihnachtszeit → in dieser Zeit wird die Weihnachtsbeleuchtung automatisch ein- und ausgeschaltet

Das folgende Script, das zyklisch einmal in der Nacht nach 0:00 Uhr aufgerufen wird, setzt diese Phasen, die als logische Systemvariablen (Zentralenvariablen) in der CCU geführt werden.

Die Jahreszeiten besitzen festgelegte Anfangs- und Endzeiten, andere Phasen sollten über individuelle Anfangs- und Endzeiten bestimmbar sein.

Am besten entwirft man sich vorher eine einheitliche Nomenklatur, zum einen, um das Ganze übersichtlich zu halten, und zum anderen, um später Programmvereinfachungen durchführen zu können. Dies gilt insbesondere für Erweiterungen. Aber dazu später mehr.

Vereinbaren wir also beispielsweise:
Die Systemvariable, die eine Phase beschreibt, bekommt einen beschreibenden Namen, z. B. Weihnachtszeit. Soll die Phase in Tag und Monat für den Beginn und das Ende frei definierbar sein, dann werden 4 weitere Systemvariablen (Zentralenvariablen) benötigt, die aus dem Namen der Variablen bestehen, die den Zustand beschreiben (hier eben Weihnachtszeit), erweitert mit z. B. „_B_Tag", „_E_Tag", „_B_Mon", „_E_Mon" für die zeitlichen Definitionen. Zusammenfassend also in diesem Beispiel:

Systemvariable (Zentralenvariable)	Funktion
Weihnachtszeit	Wird in der Weihnachtszeit true, ansonsten false
Weihnachtszeit_B_Tag	An diesem Tag beginnt die Weihnachtszeit
Weihnachtszeit_E_Tag	An diesem Tag endet die Weihnachtszeit
Weihnachtszeit_B_Mon	In diesem Monat beginnt die Weihnachtszeit
Weihnachtszeit_E_Mon	In diesem Monat endet die Weihnachtszeit

Die Variable für die Jahreszeit ist eine Werteliste mit den Daten:

Variablenwert	Bedeutung
0	Frühling
1	Sommer
2	Herbst
3	Winter

Weiterhin wird eine weitere Systemvariable vom Typ Zahl benötigt:

MonTagAkt = aktuelles Datum, Monat und Tag.

Zur Berechnung der Jahreszeiten wird die Rechnung für die meteorologischen Jahreszeiten (auf der Nordhalbkugel) verwendet:

Frühling 01.03. bis 31.05.
Sommer 01.06. bis 31.08.
Herbst 01.09. bis 30.11.
Winter 01.12. bis 28./29.02.

Nun zum Script:

```
01 !Aktuellen Tag und Monat in vergleichbare Systemvariable umsetzen
02 !-------------------------------------------------------------------------------
03 dom.GetObject("MonTagAkt").State(system.Date("%d").ToInteger() +
   (system.Date("%m").ToInteger()*100));
```

In diesem ersten Scriptabschnitt wird der aktuelle Monat und der aktuelle Tag in die Systemvariable MonTagAkt geschrieben. Um die Datumsvergleiche übersichtlich und einfacher zu halten, wird aus den beiden Daten eine Zahl nach der Formel MonTagAkt = <aktueller Monat> * 100 + <aktueller Tag> berechnet. Damit bedeutet die Zahl 601 zum Beispiel Monat Juni und Tag 1.

```
!-------------------------------------------------------
!Fruehling Maerz,April,Mai
!-------------------------------------------------------
if ((dom.GetObject("MonTagAkt").State()<601) &&
    dom.GetObject("Jahreszeit").State(0);
    dom.GetObject("Bewässerungszeit").State(true);
}
```

Beachten Sie, dass bei Phasen über den Jahreswechsel die beiden Bedingungen durch ODER (||) verknüpft sind, bei allen anderen hingegen durch UND (&&).

Der meteorologische Frühling geht vom 01.03. bis zum 31.05., mit unserer Umrechnung von 301 bis 531 (also kleiner als 601, was dem 01.06. entspricht).

```
!-------------------------------------------------------
!Sommer Juni,Juli,August
!-------------------------------------------------------
if ((dom.GetObject("MonTagAkt").State()<901) &&
    dom.GetObject("Jahreszeit").State(1);
    dom.GetObject("Bewässerungszeit").State(true);
}
!-------------------------------------------------------------------
!Herbst September,Oktober,November
!-------------------------------------------------------------------
if ((dom.GetObject("MonTagAkt").State()<1201) &&
    dom.GetObject("Jahreszeit").State(2);
}
!-----------------------------------------------
!Bewaesserungsjahreszeit verlaengert
!-----------------------------------------------
if ((dom.GetObject("MonTagAkt").State()<1101) &&
(dom.GetObject("Monat.Tag").State()>=901)) {
    dom.GetObject("Bewässerungsjahreszeit").State(true);
}
!-------------------------------------------------------------
!Winter Dezember.Januar,Februar
!-------------------------------------------------------------
if ((dom.GetObject("MonTagAkt").State()<301) ||
(dom.GetObject("Monat.Tag").State()>=1201)) {
    dom.GetObject("Jahreszeit").State(3);
    dom.GetObject("Bewässerungsjahreszeit").State(false);
}
```

Beim Winter muss, wie oben erwähnt, die ODER-Verknüpfung verwendet werden (die Periode geht über den Jahreswechsel), für den Winter gilt die Zeit 01.12. (>1201) bis 28./29.02. (<301).

```
!---------------------------------
!Weihnachtszeit
!---------------------------------
var start_periode;
var ende_periode;
start_periode=(dom.GetObject(Weihnachtszeit_B_Mon").State()*100) +
dom.GetObject(Weihnachtszeit_B_Tag").State();
ende_periode=(dom.GetObject(Weihnachtszeit _E_Mon").State()*100) +
dom.GetObject(Weihnachtszeit_E_Tag").State();
if ((dom.GetObject("MonTagAkt").State() >= start_periode) ||
(dom.GetObject("MonTagAkt").State() < ende_periode))
{
    dom.GetObject("Weihnachtszeit").State(true);
}
else
{
    dom.GetObject("Weihnachtszeit").State(false);
}
```

Beim Script wird davon ausgegangen, dass das Ende der Weihnachtszeit immer im Folgejahr vom Anfang der Weihnachtszeit liegt.

Die Zeile 44:

```
if ((dom.GetObject("MonTagAkt").State() >= start_periode) ||
(dom.GetObject("MonTagAkt").State() <= ende_periode))
```

bedeutet übersetzt Folgendes:

wenn das aktuelle Datum auf oder nach dem Anfang der definierten Weihnachtszeit liegt
oder
wenn das aktuelle Datum vor dem Ende der definierten Weihnachtszeit liegt.

Spätere Erweiterungen ohne Änderung des Scripts:
Im Folgenden eine einfache Möglichkeit, auch später weitere Perioden hinzuzufügen, ohne das Script ändern zu müssen:

Wenn man vorhat, später weitere Perioden einzufügen, dann kann man die Namen der Perioden auch als String mit Separatoren in eine Systemvariable schreiben. Es ist aber darauf zu achten, dass beim Hinzufügen einer Periode auch die 4 Systemvariablen (Zentralenvariablen) für den Tag des Beginns, den Monat des Beginns, den Tag des Endes und den Monat des Endes der Periode erstellt werden, ansonsten wird das Script an der betreffenden Abfragestelle (mit einem Fehler) abbrechen.

Nehmen wir als Liste (Systemvariable *Periodenliste*) beispielsweise an:
Die Systemvariable *Periodenliste* vom Typ Zeichenkette hat den Wert (Inhalt):
„Bewässerungszeit,Weihnachtszeit,Urlaubszeit“

Dann lässt sich diese Liste in einem Script durchsuchen:

```
string sPeriodenname;

foreach (sPeriodenname, dom.GetObject("Periodenliste").State().Split(","))
{
   ! Die lokale Scriptvariable sPeriodenname hat nun beim
   ! ersten Durchlauf der Schleife den Inhalt Bewässerungszeit,
   ! zweiten Durchlauf der Schleife den Inhalt Weihnachtszeit,
   ! dritten Durchlauf der Schleife den Inhalt Urlaubszeit.
   !
   ! hier kann dann die Abfrage für den Beginn und das Ende gemacht
   ! werden.
}
```

Die Namen der 4 Systemvariablen (Zentralenvariablen) für den Tag des Beginns, den Monat des Beginns, den Tag des Endes und den Monat des Endes der Periode kann man sich im Script durch Ketten von Strings ganz einfach selbst bilden:

Im ersten Durchlauf hat die Stringvariable sPeriodenname den Inhalt: „Bewässerungszeit“.

Der Tag des Beginns ist nach unserer oben gewählten Namenskonvention: „Bewässerungszeit_B_Tag“
Als Programmcode ausgedrückt: `sPeriodenname#"_B_Tag"`

Bei der Datumsabfrage (größer Startdatum/kleiner Endedatum) haben wir noch das Problem, dass bei frei definierbaren Start- und Endedaten von vorneherein nicht klar ist, ob eine UND-Verknüpfung der Abfragen (Start und Ende im gleichen Jahr) oder eine ODER-Verknüpfung (Periode geht über den Jahreswechsel) verwendet werden muss. Anders ausgedrückt: Beides ist möglich.

Wir müssen also zuerst feststellen, welche logische Verknüpfung der beiden Abfragen verwendet werden muss. Einfach feststellen kann man dies folgendermaßen:

- Liegt das Endedatum der Periode nach dem Startdatum (Betrachtung ohne Jahreszahl), dann wird UND verknüpft, da beide Perioden im gleichen Jahr liegen.
- Liegt das Endedatum der Periode vor dem Startdatum (Betrachtung ohne Jahreszahl), dann wird ODER verknüpft, da beide Perioden in unterschiedlichen Jahren liegen.

Die zugehörige Abfrage im Programm sieht folgendermaßen aus:

Endedatum nach Anfangsdatum, also UND-Verknüpfung:

```
var start_periode;
var ende_periode;
start_periode=(dom.GetObject(sPeriodenname#"_B_Mon").State()*100)
+dom.GetObject(sPeriodenname# "_B_Tag").State();
ende_periode=(dom.GetObject(sPeriodenname#"_E_Mon").State()*100)
+dom.GetObject(sPeriodenname# "_E_Tag").State();

If (ende_periode > start_periode)
{
   !Datumsabfrage mit UND
}
```

Endedatum vor Anfangsdatum, also ODER-Verknüpfung:

```
var start_periode;
var ende_periode;
start_periode=(dom.GetObject(sPeriodenname#"_B_Mon").State()*100)+dom.GetObject(sPeriodenname#
"_B_Tag").State();
ende_periode=(dom.GetObject(sPeriodenname#"_E_Mon").State()*100)+dom.GetObject(sPeriodenname#
"_E_Tag").State();

If (ende_periode < start_periode)
{
   !Datumsabfrage mit ODER
}
```

In einem fertigen Script sähe das folgendermaßen aus (Ersatz der Scriptzeilen 37 bis 59 oben):

```
37  string sPeriodenname;
38  var start_periode;
39  var ende_periode;
40  foreach (sPeriodenname, dom.GetObject("Periodenliste").State().Split(","))
    {
41     start_periode=(dom.GetObject(sPeriodenname#"_B_Mon").State()*100)+dom.GetObject(
       sPeriodenname# "_B_Tag").State();
42     ende_periode=(dom.GetObject(sPeriodenname#"_E_Mon").State()*100)+dom.GetObject(s
       Periodenname# "_E_Tag").State();
43     If (ende_periode > start_periode)
44     {
45        if ((dom.GetObject("MonTagAkt").State() > start_periode) ||
          (dom.GetObject("MonTagAkt").State() <= ende_periode))
46        {
47           dom.GetObject(sPeriodenname).State(true);
48        }
49        else
50        {
51           dom.GetObject(sPeriodenname).State(false);
52        }
53     }
54     If (ende_periode < start_periode)
55     {
56        if ((dom.GetObject("MonTagAkt").State() > start_periode) &&
          (dom.GetObject("MonTagAkt").State() <= ende_periode))
57        {
58           dom.GetObject(sPeriodenname).State(true);
59        }
60        else
61        {
62           dom.GetObject(sPeriodenname).State(false);
63        }
64     }
65  }
```

Was uns jetzt noch fehlt, ist der Zusammenhang zum mediola Creator:

Grundsätzlich lassen sich solche Daten (Perioden) auch sehr schön visualisieren. Ein Beispiel ist die Jahreszeit im Zusammenhang mit der Heizungsregelung:

Die Daten für Beginn und Ende einer Periode lassen sich mithilfe der im letzten Artikel beschriebenen Datumseinstellung sehr gut definieren.

Bei der oben beschriebenen flexiblen Programmierung, die ohne Scriptänderung Erweiterungen zulässt, ist es sinnvoll, in der Bedieneroberfläche eine Eingabemöglichkeit für Texte zu schaffen, um die ebenfalls oben beschriebene Systemvariable *Periodenliste* zu beschreiben.
Aber dies ist ein Thema für eine der nächsten Folgen.

Im nächsten Artikel geht es weiter mit den Homematic Script mediola Themen.

Sehr geehrter Leser,
bei diesem Artikel zur Scriptprogrammierung handelt es sich um einen Fachbeitrag eines erfahrenen Homematic Users und Autors. Die ELV/eQ-3 Unternehmensgruppe selbst nutzt die Möglichkeiten dieser Schnittstelle nicht, möchte aber den Anwendern der CCU2 den Zugang zu dieser Schnittstelle nicht verwehren.
Sollten Sie Schwierigkeiten bei der Verwendung dieser zusätzlichen Programmiermöglichkeit der CCU2 haben, so haben Sie bitte Verständnis dafür, dass wir Ihnen hierzu leider keinen Support geben können. In den entsprechenden Foren und Internet-Plattformen rund um das Thema „Programmierung Homematic CCU" finden Sie jedoch sicherlich im Bedarfsfall die notwendigen Anregungen und Hilfestellungen für Ihr Projekt.

Mögliche Quellen im Internet:
https://www.homematic-inside.de/software/download/item/homematic-skript
https://homematic-forum.de/forum/viewtopic.php?f=19&t=18692