Contents

1 Einführung

Kurze Hilfestellung zur Einrichtung der Datenkonvertierung von Datendateien diverser Wechselrichter in Solarlog-Dateien zum Darstellen auf einer Solarlog-Homepage.

2 Eigenschaften

InverterData2SolarLog unterstützt folgende Wechselrichter

- SolarExplorer von SMA
- Danfoss Triply Lynx

3 Installation

1. Download
2. Installkit auspacken

3. Konfigrationsdateien anpassen
4. Kit lokal austesten
5. Kit auf Werbserver kopieren

3.1 Download

Die Datei ***links.html*** aus dem Homepagekit von Solare Datensysteme bitte durch die Datei ***links.html*** ersetzen, die im Paket im Verzeichnis ***SolarLog*** enthalten ist.

Das Homepagekit kann bei SolarLog herunter geladen werden. Bitte die Version 2.5.0 verwenden!

Das Paket mit den Scripten kann [hier][InverterData2SolarLog-Download] heruntergeladen werden.

Hinweise in letzter Minute, die noch nicht in diese Seite eingearbeitet wurden, gibt es hier.

3.2 Installkit auspacken

Vor der Inbetriebnahme der Homepage ist die Datei `base_vars.js` zu erstellen (falls sie noch nicht vorhanden ist) und anzupassen!!!

3.3 Konfigurationsdateien anpassen

3.3.1 base_vars.js

Wenn die Scripte einmal fehlerfrei durchgelaufen sind, wird eine Datei `base_vars.js` erstellt falls sie noch nicht vorhanden. Sie enthält dann aber nur Einträge, die durch die Scripte verändert und aktualisiert werden. Ist diese Datei schon vorhanden, werden nur die Einträge angepasst.

Die folgenden Einträge in der `base_vars.js` müssen deshalb nicht beachtet, verändert oder eingefügt werden:

```
var SLTyp =
var DATALOGGER_NAME =
var Firmware =
var isOnline =
var SLDatum =
var SLUhrzeit =
```

Sie werden von den PHP-Scripten geschrieben!!!

Die folgenden Einträge sind unbedingt anzupassen:

- `var AnlagenKWP = 7000`

  Gesamtleistung der Anlage in Watt (Summe aller angeschlossenen Modulleistungen)

- `var AnzahlWR = 1`

  Anzahl Wechselrichter.

- `MaxWRP[0] = new Array(xxxx,xxxxx,xxxxxx,xxxxxx)`

  Für jeden WR ist so eine Zeile erforderlich. Bei zwei WR gibt es deshalb z.B. noch eine Zeile

  `MaxWRP[1] = ...`

  Die Bedeutung der Werte bitte ddd entnehmen.

- `WRInfo[0] = new Array("Danfoss TLX PRO 6k", ...)`

  Für jeden WR ist so eine Zeile erforderlich. Bei zwei WR gibt es deshalb z.B. noch eine Zeile

  ```
  WRInfo[0] = new Array("Danfoss TLX PRO 6k", "1111111111", 7000,
                        0, "Danfoss", 2, null, null, 6000,
                        null, 12, 0, 1, 1000)

  WRInfo[1] = ........
  ```

  Die Bedeutung der Werte bitte Solarlog-Datenformat entnehmen.

- `WRInfo[0][6] = new Array("String1","String2")`

  Wenn an den WR mehrere Strings angeschlossen sind, ist für jeden WR eine solche Zeile erforderlich.

  Die Bedeutung der Werte bitte Solarlog-Datenformat entnehmen.

- `WRInfo[0][7] = new Array(1,1)`

  Wenn an den WR mehrere Strings angeschlossen sind, ist für jeden WR eine solche Zeile erforderlich.

  Die Bedeutung der Werte bitte Solarlog-Datenformat entnehmen.

- `WRInfo[0][9] = new Array(3500,3500)`

  Wenn an den WR mehrere Strings angeschlossen sind, ist für jeden WR eine solche Zeile erforderlich.

  Die Bedeutung der Werte bitte Solarlog-Datenformat entnehmen.

- `var HPTitel = "Name meiner Hoepage"`

  In Anführungszeichen eingeschlossen.

- `var HPBetreiber = "Name des Anlagenbetreibers"`

  In Anführungszeichen eingeschlossen.

- 'var HPEmail = "Email des Betreibers"'

  In Anführungszeichen eingeschlossen.

- `var HPStandort = "Standort der Anlage"`

  In Anführungszeichen eingeschlossen.

- `var HPModul = "Anzahl und Typ der Solarmodule"`

  In Anführungszeichen eingeschlossen.

- `var HPWR = "WR-Typ und Bezeichnung"`

  In Anführungszeichen eingeschlossen.

- `var HPLeistung = "Anlagenleistung"`

  In Anführungszeichen eingeschlossen.

- `HPInbetrieb = "Inbetriebnahmedatum der Anlage"`

  In Anführungszeichen eingeschlossen.

- `var HPAusricht = "Ausrichtung der Anlage"`

  In Anführungszeichen eingeschlossen.

- `var BannerZeile1 = "beliebiger Wert für Banner"`

  In Anführungszeichen eingeschlossen.

- `var BannerZeile2 = "beliebiger Wert für Banner"`

  In Anführungszeichen eingeschlossen.

- `var BannerZeile3 = "beliebiger Wert für Banner"`

  In Anführungszeichen eingeschlossen.

- `var BannerLink = "http://www...../banner.html"`

  URL auf die Datei banner.html, die sich im Solarlog-Verzeichnis befindet.

  In Anführungszeichen eingeschlossen.

- `StatusCodes[0] = "Off|MPP"`

  Hier werden die zwei möglichen Status-Zustände des jeweiligen WR eingetragen (Zustand Off und MPP).

  Die Zeile muss für jeden WR enthalten sein (analog *MaxWRP[x]* und *WRInfo[x]*, siehe weiter oben in dieser Tabelle).

Wer die Werte “übersetzen” möchte, kann hier auch gern “Aus,An” eintragen.

In Anführungszeichen eingeschlossen.

- `FehlerCodes[0] = " "`

  Der Inhalt dieser Werte ist gleichgültig weil Danfoss-WR keinen Fehlercode ausgeben. Die Zeile muss lediglich für jeden WR enthalten sein (analog *MaxWRP[x]* oder *WRInfo[x]*, siehe weiter oben in dieser Tabelle).

  In Anführungszeichen eingeschlossen.

- `var Verguetung =`

  Vergütung pro kWh in €Cent * 100

  Beispiel: 21,54 Cent/kWh => Eintrag `2154`

- `var WRTyp = "Danfoss"`

  Beliebiger Eintrag des WR-Types.

  In Anführungszeichen eingeschlossen.

- `var SLVer = 2`

  Hier muss 2 eingetragen sein!

- `var Intervall = 600`

  Hier ist unbedingt das Intervall in Sekunden einzutragen, in dem der WR seine Daten liefert!

  Es handelt sich dabei nicht um das Upload-Intervall neuer Daten-Dateien, sondern das Intervall, mit dem die Ertragswerte in den Datendateien gespeichert werden.

  Beispiel:

  ```
  10 Minuten => 600
   5 Minuten => 300
   1 Minute => 60
  ```

- `var isTemp = true | false`

  true wenn die WR-Temperatur in den gelieferten Daten vorhanden ist (bei Danfoss WR der Fall).

  false wen die WR-Temperatur nicht vorhanden ist.

- `var eventsHP = 0`

  Ist bei Danfoss-WR auf 0 zu stellen !!!

- `var Lang = "DE"`

  Sprache, in der die SL-Seite angezeigt werden soll (siehe im Solarlog-Verzeichnis Dateien `lang_XX.js`).

- `var Lang = "DE"` für die Deutsche Seite.
- `var FirmwareDate = "23.03.2012"`

  Dieser Eintrag muss ein gültiges Datum enthalten, es darf nicht zu alt sein !!!

  Der Wert ist ist dabei fast beliebig wählbar, ich empfehle, den 23.03.2012 oder später zu übernehmen.

Beispiele zur base_vars.js:

- Ein WR mit 2 Strings ohne WR-Temperatur (Standard Solarlog 100e)

  http://photonensammler.homedns.org/SLDaten/base_vars.js
- Emulation (auch von mir geschrieben) 2 WR, einer mit 5 Strings, einer mit 4 Strings ohne WR-Temperatur

  http://helios64.dyndns.org/base_vars.js

  Hier kann man gut sehen, wie mehrere Strings in der `Base_vars.js` zu definieren sind.
- Ein Danfoss-WR mit 2 Strings und WR-Temperatur (Emulation aus diesem Paket):

  http://www.lykkegaarden.eu/lykkegaarden/base_vars.js

  Eine Beschreibung des Solarlog-Datenformats findet sich Solarlog-Datenformat.

  Die Datei base_vars.js hat keinen Einfluss auf das Funktionieren der Konvertierungsscripte.

  Sollen erst einmal testweise die Solarlog-Dateien erstellt werden, ist diese Datei nicht erforderlich.

  Sie ist aber erforderlich wenn die erzeugten Daten später auf der Internetseite angezeigt werden sollen.

3.4 Hinweise zur Verzeichnisstruktur und den Pfadangaben in den config-Dateien

- Unter UNIX-Systemen ist bei den Pfadangaben und Dateinamen auf Groß- und Kleinschreibung zu achten!!!

  Ich schlage die Verwendung einer Verzeichnisstruktur vor, an die man nicht gebunden ist, die aber die erste Einrichtung der Scripte erleichtern kann:

  //[Verzeichnisstruktur] (index-Dateien/image003.jpg)

- Das Verzeichnis xx dient nur der besseren Veranschaulichung wie die Hierarchie der Verzeichnisse aussieht, es hat keinen Einfluss auf die Funktion, kann auch weggelassen oder umbenannt werden.

- In "InverterData" legt der Wechselrichter seine Datendateien ab, die später in Solarlog-Daten konvertiert werden sollen.

- In "SolarLog" werden die heruntergeladenen Dateien der Solarlog-Homepage gespeichert. Auf dieses Verzeichnis (auf `index.html`) verweist auch die URL, über die später die Seite angezeigt wird. In diesem Verzeichnis werden auch die erzeugten SL-Dateien abgelegt.

- In Danfoss2SolarLog muss sich das Verzeichnis "Classes" mit den darin befindlichen PHP-Dateien befinden!!! Weiterhin müssen sich dort die 3 "config.xxxx.php" Dateien und das Script "update.php" befinden.

  Das Script "update.php" kann zum Testen der Funktion der Konvertierung von Hand gestartet werden. Später wird es durch einen eingerichteten Cronjob gestartet, der abhängig von der Zeit zu der der WR neue Daten liefert, automatisch ausgeführt wird.

- Ich empfehle, das Verzeichnis "errors" an seinem vorgeschlagenen Platz zu belassen.

  Änderungen sind durch Editieren der "config.general.php" möglich. In diesem Verzeichnis wird bei Scriptfehlern eine Datei "errors.txt" erzeugt, in der aufgetretene Fehlermeldungen geloggt werden.

  Diese Datei kann gelöscht werden, sie wird bei Scriptfehlern automatisch neu erzeugt.

- Die Voreinstellungen in den "config.xxxx.php" Dateien beziehen sich auf die oben angegebene Verzeichnisstruktur. Wird diese Struktur verwendet, sind damit nur wenige Anpassungen in den Konfigurationsdateien erforderlich.

3.4.1 Einstellungen in der Datei `config.danfoss.php`

Die Einstellungen in der Datei sind kommentiert. Zeilen, die mit `//` beginnen, oder Abschnitte, die in `/\* \*/` eingeschlossen sind, sind Kommentare.

`define-Einträge`, die geändert werden und schon im Beispiel durch Anführungszeichen eingeschlossen sind, müssen wieder durch Anführungszeichen eingeschlossen werden!!!

- `define('USE_FTP',false);`

  false: wenn die Scripte die zu konvertierenden Datendateien des WR aus einem *lokalen Verzeichnis* beziehen.

  true: WR-Datendateien sollen von einem *FTP-Server* geholt werden

- `define('FTP_SERVER', 'ftp.myftpserver.com');`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***true*** steht:

  Hier wird der FTP-Server eingetragen, von dem die Scripte die WR-Datendateien laden sollen.

- `define('FTP_USERNAME', 'myFTPUsername');`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***true*** steht:

  Hier wird der Username zum Zugriff auf den FTP-Server eingetragen.

- `define('FTP_PASSWORD', 'MyFTPPassword');`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***true*** steht:

  Hier wird das Passwort zum Zugriff auf den FTP-Server eingetragen.

- `define('FTP_PORT', 21);`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***true*** steht:

  FTP-Port – in der Regel 21, Eintrag muss normalerweise nicht verändert werden.

- `*define('FTP_TIMEOUT', 90);`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***true*** steht:

  Timeout der FTP-Verbindung in Sekunden.

  Standardwert ist 90 Sekunden. Ist die FTP-Verbindung sehr langsam wodurch es öfter zu Abbrüchen kommt, kann dieser Wert erhöht werden.

- `define('FTP_INVERTER_DATA_PATH', '.');`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***true*** steht:

  Befinden sich die WR-Datendateien nicht im Root-Verzeichnis des FTP-Servers, kann hier das FTP-Verzeichnis eingetragen werden, in dem sich die Dateien befinden.

  Für das FTP-Root-Verzeichnis ist der Eintrag

  define('FTP_INVERTER_DATA_PATH', '')

- `define('LOCAL_TEMP_DIR', '.');`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***true*** steht:

  Die WR-Datendateien müssen beim Download per FTP irgendwo lokal zwischengespeichert werden. Hier ist ein Verzeichnis einzutragen, auf das die Scripte Schreibrechte haben und wo die Daten gespeichert werden können.

  Bei der Einstellung `define('LOCAL_TEMP_DIR', null);` wird das Temp-Verzeichnis des Systems ermittelt und die Daten dort zwischengespeichert.

  Nach der Verarbeitung der Daten werden die Dateien wieder gelöscht.

- `define('LOCAL_TEMP_DIR', '.');`

  Eintrag ist nur relevant wenn ***USE_FTP*** auf ***false*** steht:

  Hier ist das Verzeichnis einzutragen, in dem sich die WR-Datendateien befinden, wenn sie auf dem lokalen Rechner vorliegen und nicht per FTP geholt werden.

  Der anzugebende Verzeichnispfad ist relativ zum Verzeichnis, in dem sich update.php befindet, anzugeben.

- `define('USED_STRINGS_1', '1,2');`

  Dieser Wert ist immer zu konfigurieren!!!

  Für jeden WR ist dieser Eintrag einmal vorhanden. Es ist anzugeben, welcher String des WR belegt ist.

  Beispiele:

  ```
  define('USED_STRINGS_1', '1,2'); WR1 benutzt String 1 und 2
  define('USED_STRINGS_1', '1,2,3'); WR1 benutzt String 1,2 und 3
  define('USED_STRINGS_2', '1,3'); WR2 benutzt String 1 und 3
  ```

- `define('DANFOSS_PLANT_NAME', 'u43381201-1');`

  Dieser Wert ist immer zu konfigurieren!!!

  Hier ist der Name der WR-Datendateien einzutragen. Dabei wird er ohne den letzten "-" vor dem Timestamp, der den Dateinamen abschließt, einzutragen.

  Im Beispiel heißen die WR-Dateien *u43381201-1-xxxxxxxxxxx*

- `define('DELIMITER', ';');`

  Dieser Wert muss in der Regel nicht verändert werden.

3.4.2 Einstellungen in der Datei "config.solarlog.php"

- `define('SLFILE_DATA_PATH', '../SolarLog');`

  Hier ist das Verzeichnis einzutragen, in dem die Scripte die konvertierten Solarlog-Dateien ablegen sollen.

  Diese Eintragung zeigt auf das Verzeichnis, in dem sich die Homepage aus dem Homepage-Kit befindet.

  Das Benutzer unter dem das Script läuft, benötigt für dieses Verzeichnis Schreibrechte.

4 Generelle Anmerkungen

- Alle Verzeichnisangaben beziehen sich relativ auf das Verzeichnis, in dem sich update.php befindet.

  Wer genau weiß was er tut, kann selbstverständlich auch absolute Verzeichnispfade angeben.

- Alle Verzeichnisnamen sind ohne abschließende Schrägstriche anzugeben.

- Trennungen innerhalb der Pfade sind durch den Schrägstrich "/" vorzunehmen.

- Um die Solarlog-Dateien zu erzeugen, ist das Script `update.php` zu starten.

- Dies geschieht am besten mittels Cronjob, dessen Startintervall an das Uploadintervall der WR angepasst wird.

- **Achtung:** Beim allerersten Start (zum Testen von natürlich Hand) kann die Zeit der Abarbeitung etwas länger sein weil erst einmal sämtliche vorhandenen Altdaten der WR verarbeitet werden. Während der Zeit der Abarbeitung kann im Solarlog-Verzeichnis das Entstehen der einzelnen Solarlog-Dateien beobachtet werden.

- Wenn die Solarlog-Dateien erst einmal erzeugt sind, geht jede weitere Abarbeitung schneller weil nur noch geänderte Daten gelesen werden müssen.

- Das Script update.php nur einmal starten!!!! Mehrfaches gleichzeitiges Abarbeiten führt unweigerlich zu Fehlern wenn das zuerst gestartete Script noch läuft!!!!

- Sollten auf Grund falscher Einstellungen in den config-Dateien fehlerhafte Solarlog-Dateien erzeugt werden oder die Scriptabarbeitung wegen anderer Fehler abgebrochen werden, ist vor einem erneuten Starten zu prüfen, ob schon SL-Dateien erzeugt wurden.

  Diese sind vor dem erneuten Starten zu löschen!!!!! Die SL-Dateien sind erst vollständig und fehlerfrei wenn update.php einmal fehlerfrei abgearbeitet wurde.

  Es werden folgende SL-Dateien im Solarlog-Verzeichnis erzeugt, die vor erneutem Scriptstart gelöscht werden müssen:

  - `min_day.js` - diese Datei unbedingt löschen, sie steuert den gesamten Scriptablauf
  - `days.js`
  - `days_hist.js`
  - `min_cur.js`

  - `months.js`
  - `years.js`
  - `minYYMMDD.js` mit `YY`=Jahr `MM`=Monat `DD`=Tag

- Je nachdem, wie weit das Script beim Abarbeiten war bis ein Fehler auftrat, kann es sein, dass noch nicht alle o.g. SL-Dateien erzeugt wurden.
- Die Scripte befinden sich noch im Beta-Stadium - ich übernehme keine Gewährleistung, dass sie schon auf jedem System fehlerfrei funktionieren!!!
- Sie wurden bisher nur mit einem WR getestet. Ich übernehme keine Gewähr, dass sie auch mit mehreren WR funktionieren (obwohl das durchaus möglich sein könnte und bei der Programmierung – soweit möglich – auch schon berücksichtigt wurde).

%

Viel Erfolg! Jörg

[InverterData2SolarLog-Download]: [http://photonensammler.homedns.org/Danfoss2solarlog/DownloadPaket/]