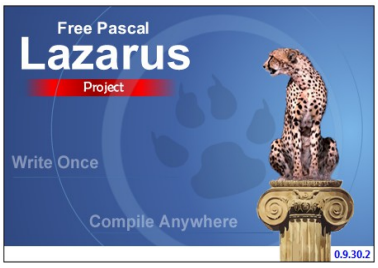
Messdatenerfassung

Das Handbuch zur Bedienung der Software (Version 1.2)

Jahr: 2011/2012
Programmierer: Mitja Stachowiak
Compiler: Freepascal (Lazarus)
Getestete Plattform (Lokal): Windows XP, Windows 7
Getestete Plattform (Server): Linux, Umschreiben für Windows möglich.



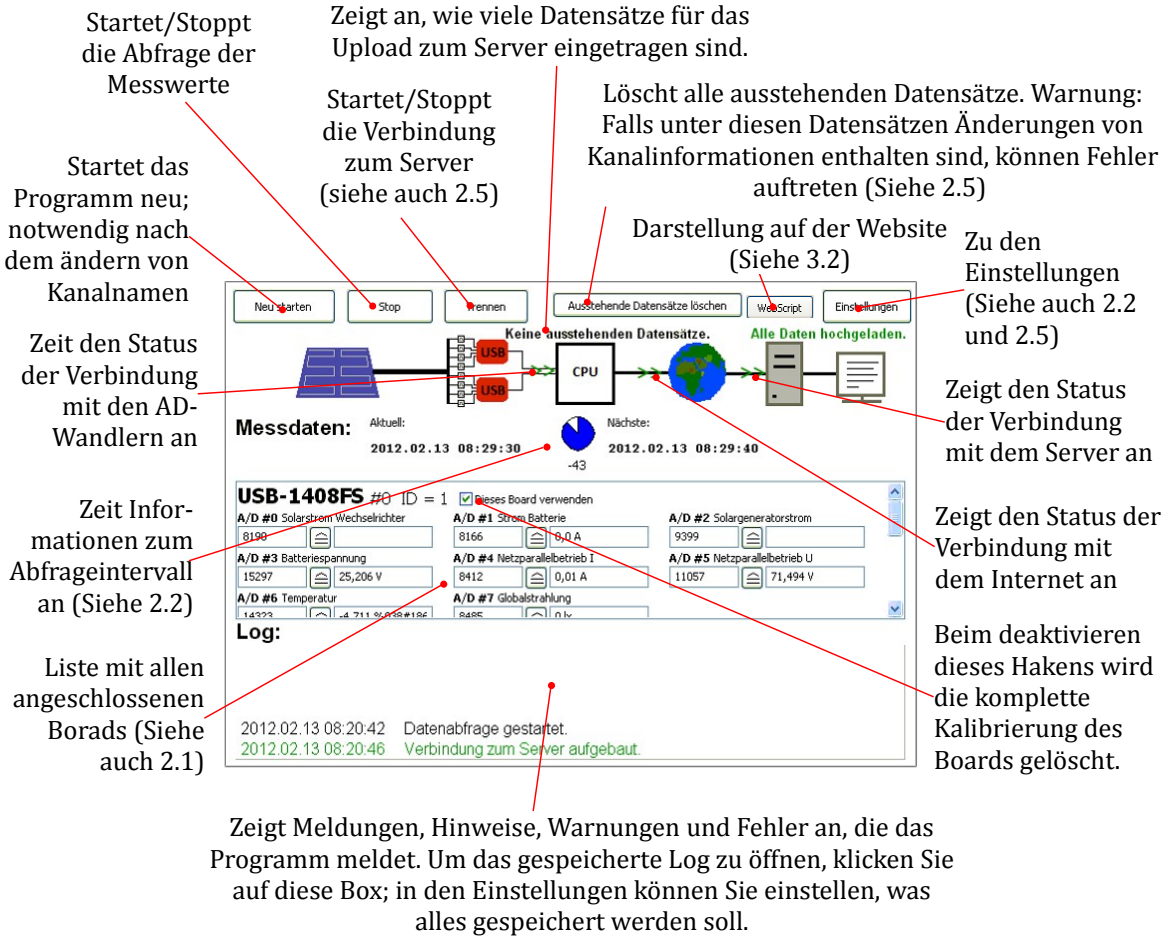
Die Übersicht:

Dieses Handbuch enthält alle anwenderseitig interessanten Informationen über die Software Messdatenupload.

1 Die Softwarekomponenten und ihre Funktionen

Die Software teilt sich in zwei Teile: Den lokalen - auf dem Laptop im Labor und den Serverteil auf der Schulhomepage. Der Serverteil wiederum ist ebenfalls geteilt: In das Perl-Script, das die Daten empfängt und wieder ausgibt und in ein HTML-Dokument, welches auf der Website angezeigt wird. Lokal können zusätzliche Kalibrierungskomponenten als DLLs erstellt werden.
Die Einstellungen für das Programm sind in Allgemeines, Upstream und Error unterteilt. In der Rubrik Allgemeines sind alle wichtigen Grundeinstellungen enthalten, auch die für das Abfrageintervall. Upstream umfasst die Einstellungen, welche für die Verbindung zum Server gebraucht werden. In der Rubrik Error kann man einstellen, welche Fehler und Warnungen protokolliert werden sollen. Die Einstellungen werden in den Anwendungsdaten gespeichert (Siehe 2.1). Mit dem Löschen dieser Einstellungen und des Autostarteintrages (Start → Alle Programme → Autostart → Datenupload) ist das Programm **deinstalliert**.

2 Bedienung der lokalen Software



2.1 Installation und Anschluss der AD-Wandler (Boards)

Das Programm DatenUpload.exe bedarf keiner Installation. Es kann an jedem beliebigen Ort gestartet werden. Jedoch muss vor dem ersten Start der Ordner mit den Anwendungsdaten angelegt werden; in diesem muss sich die Datei Einstellungen.ini befinden, diese braucht jedoch keinen Inhalt. Der Pfad dieser Datei ist dann:
*:/Dokumente und Einstellungen/ANGEMELDETER BENUTZER/Anwendungsdaten/DatenUpload/Einstellungen.ini. Nun sollte das Programm starten – die Installation des Treibers (mccdaq – Instacal und UniversalLib) ist vorausgesetzt. Standardgemäß wird das Programm keine angeschlossenen Boards erkennen. Diese müssen zuvor mit InstaCal konfiguriert werden. Es ist hierbei notwendig, eine Seriennummer einzugeben (Es darf nicht die gleiche Nummer doppelt verwendet werden).

2.1.1 Autostart

Seit wegen einem Kurzschluss das Ersatzlaptop eingerichtet wurde, gab es ein merkwürdiges Treiberproblem: Nach dem Hochfahren waren die angeschlossenen Boards einige Zeit nicht verfügbar. Um dies zu beheben, gibt es im Autostarteintrag für das Programm unter anderem die Möglichkeit, eine Startverzögerung einzustellen: *Einstellungen → Allgemein → Start/Stop → Programm beim Systemstart ausführen*. Geben Sie hier im Feld *Parameter* die Entsprechenden Einstellungen an (Trennung durch Leerzeichen):

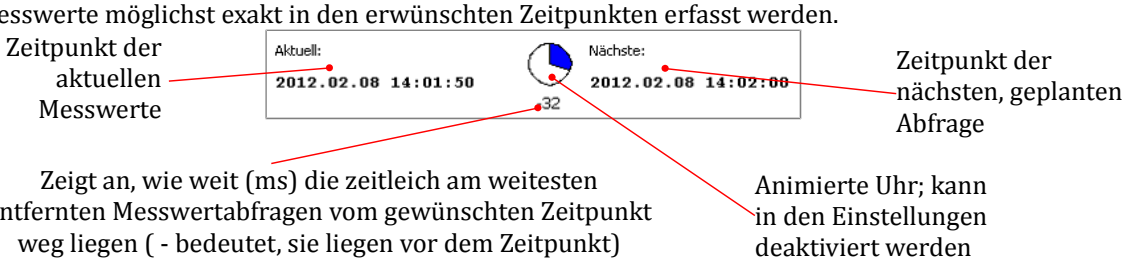
-start	Nach dem Programmstart wird die Messwertabfrage gestartet
-connect	Nach dem Programmstart wird die Verbindung zum Server aufgebaut
-hide	Das Programm wird nach dem Start versteckt
-restart	Das Programm startet so schnell, wie möglich, ohne den Ladebildschirm anzuzeigen
-timeout	Das Programm wartet vor dem Laden des Treibers für eine einstellbare Zeit (Beispiel: *-timeout 1000* für eine Sekunde warten)

2.1.2 Herunterfahren

Weil des Laptop nicht für einen Dauereinsatz konzipiert ist, ist es wichtig, dass von Zeit zu Zeit ein Neustart vorgenommen wird. Andernfalls können aufgrund von Speicherfehlern unerwünschte Fehler auftreten. Deswegen ist in der Software eine Funktion zum automatischen Herunterfahren enthalten: Öffnen Sie die Einstellungen und geben Sie in der Rubrik allgemein unter Start/Stop eine Uhrzeit (hh:mm) ein, zu welcher das System herunterfahren soll. Das Hochfahren muss über andere Wege gewährleistet werden; zum Beispiel über ein WakeOnLan-Server.

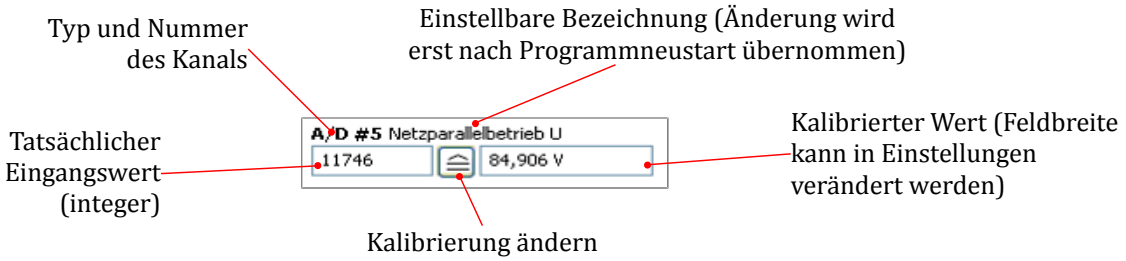
2.2 Intervall

Das Intervall bezeichnet das Abfrageintervall. Auch dieses kann in den Einstellungen festgelegt werden. Es darf jedoch nicht kürzer als 200ms sein. Mit jedem Intervall werden alle Messwerte abgefragt und (falls dies so eingestellt ist) für das Upload zum Server eingetragen. Der Intervall-Timer verfügt über eine spezielle Funktion, die alle Abfragen auf möglichst ganzzahlige Zeiten legt. Beispiel: Das Intervall ist auf zwei Minuten eingestellt. Die Abfrage der Messwerte soll also um 12:00 Uhr, 12:02 Uhr, 12:04 Uhr stattfinden, nicht aber um 12:00:35 Uhr, 12:02:35 Uhr, 12:04:35 Uhr oder um 11:59 Uhr, 12:01 Uhr, 12:03 Uhr. Des weiteren ist eine spezielle Optimierung eingebaut, die die Abfragedauer der letzten Messwerte misst und die Abfrage der Nächsten um die Hälfte davon nach vorne verlegt, sodass die Messwerte möglichst exakt in den erwünschten Zeitpunkten erfasst werden.

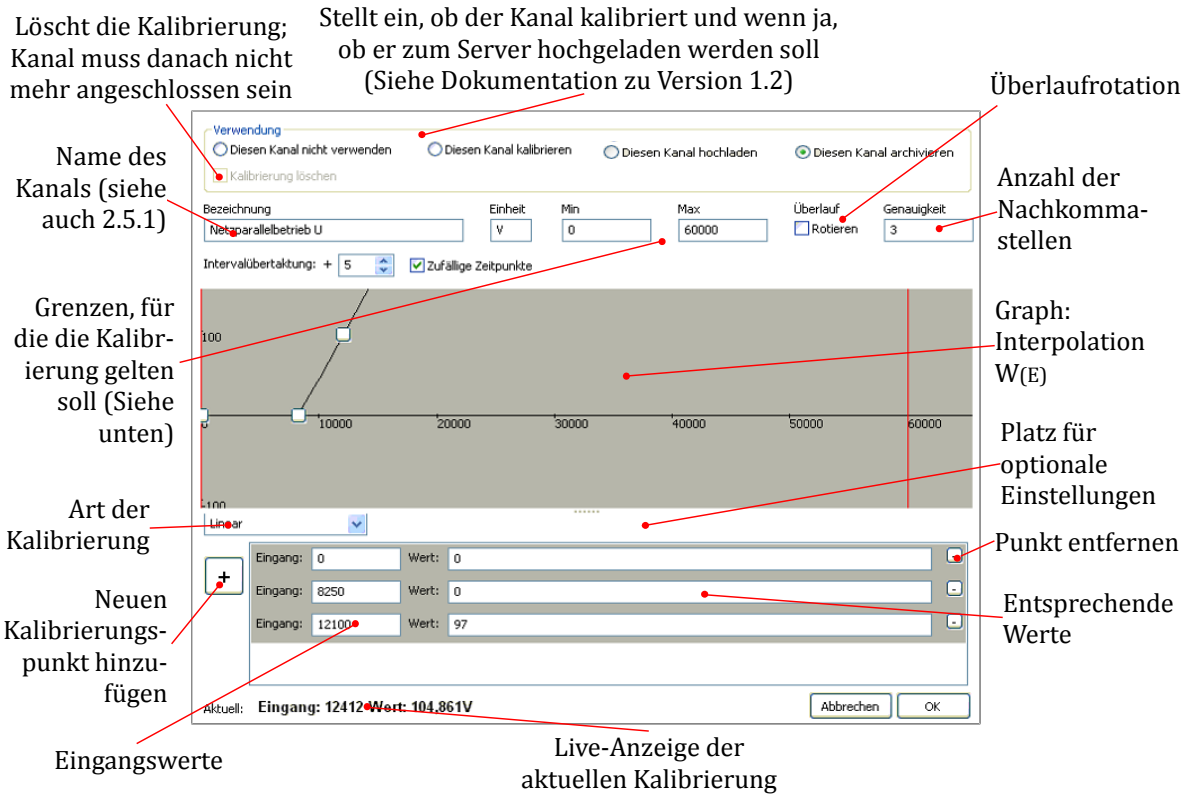


2.3 Kalibrierung

Die Kalibrierung ist der wohl umfangreichste Teil der Software. Die Werte, die die AD-Wandler liefern sind positive Ganzzahlen (integer). Diese entsprechen den Spannungen (+/- 10V), die an den Kanälen anliegen. Ein bei einem zwölf-bit-Wandler kann der Eingangswert maximal $2^{12}-1=4095$ sein. Null liegt in der Mitte, also bei 2048.



Mit einem Klick auf den Entspricht-Button eines Kanals kann dieser Kalibriert werden. Es öffnet sich das Kalibrierungsfenster (Sei W der kalibrierte Wert und E der Eingangswert des Wandlers (Integer)):



Die Idee der Kalibrierung ist die, dass man eine Funktion W(E) aufstellt. Diese verläuft durch unterschiedliche Kalibrierungspunkte. Diese stellen eine Verknüpfung zwischen dem Eingangswert und dem wirklichen Wert dar (Beispiel: 8940 ≙ 5V). Die Art der Kalibrierung gibt an, wie zwischen diesen Werten interpoliert werden soll. Da sich die Punkte auch im Graphen-Fenster verschieben lassen, wird diese Methode schnell klar, wenn man die unterschiedlichen Verfahren durchprobiert:

nächstliegend	Es wird immer der nächst kleinere, nächstliegende oder nächst größere Kalibrierungspunkt als Wert verwendet. Eine Auswahl zur Rundungsart wird angeboten, wenn Sie dieses Verfahren wählen. Geeignet für diskrete Werte (z.B. An/Aus, Schalterstellungen, etc.)
linear	Dies ist die wohl am häufigsten verwendete Kalibrierung. Die Kalibrierungspunkte werden linear verbunden. Tipp: Werte, die zum Beispiel nicht kleiner als Null werden können, können bei Null mit einem zusätzlichen Kalibrierungspunkt abgeschnitten werden. Dies verhindert, dass aufgrund von Messungenauigkeit diese Werte doch kleiner als Null werden können.
(Polynom-) Funktion	Bei Wahl dieser Option wird ein Eingabefeld für eine Funktion angeboten. Hier kann man unabhängig der Kalibrierungspunkte eine Funktion eingeben; diese muss jedoch ganzrational sein. Die Funktion kann auch automatisch per Gauss-Algorithmus bestimmt werden, sodass sie durch die Punkte geht. Hochgradige Funktionen schweifen jedoch oft zu sehr ab, um als Kalibrierung zu fungieren.
Splines	Die Splines sollen das Problem des Abschweifens umgehen, indem nur Teilfunktionen durch je zwei Punkte gebildet werden, die mit gleicher Steigung abschließen. Es handelt sich hierbei jedoch um keine echten, kubischen Splines
DLL	Alternativ kann auch eine eigene DLL zum kalibrieren verwendet werden (Siehe 2.3.1)

Der eigentlichen Kalibrierung voraus geht die Prüfung auf Überlauf. Bei dieser wird geprüft, ob sich die Messwerte innerhalb der angegebenen Grenzen (Min und Max) bewegen. Falls diese Prüfung unerwünscht ist, kann als Untergrenze Null und als Obergrenze der maximale Eingangswert des Wandlers eingegeben werden. Wenn ein Eingangswert eine Grenze überschreitet, wird er auf diese zurückgesetzt. Alternativ kann die Überlaufrotation eingeschaltet werden. In diesem Fall wird der Wert beim überschreiten der Obergrenze auf die Untergrenze gesetzt und beim unterschreiten der Untergrenze auf die Obergrenze. Dies kann zum Beispiel beim messen einer Windrichtung hilfreich sein (Für eine Richtung (Winkel) gilt: 370° = 10° = -350°).
Der Wert im Feld Intervallübertaktung gibt an, wie viele Messwertabfragen zusätzlich, zwischen den Intervallen, durchgeführt werden sollen und ob deren Zeitpunkte zufällig oder kontinuierlich sein sollen.

Mit dem setzen dieser Option wird die Zeitangabe der Messwerte ungenau, da sich diese nun aus dem arithmetischen Mittelwert vieler, zu unterschiedlichen Zeitpunkten gemessener Messwerte, berechnen. Allerdings kann dies die Genauigkeit der Messung steigern, wenn zum Beispiel Induktionen in den Leitungen das Ergebnis verfälschen.

2.3.1 Eigene DLL zur Kalibrierung schreiben

Wenn die angebotenen Methoden zur Messwertkalibrierung nicht ausreichen sollten, kann eine eigene DLL geschrieben werden. Diese erhält über ein spezielles Interface zugriff auf relevante Funktionen und Werte im Programm. Eine DLL kann zum Beispiel zum Messen der mittleren Änderungsrate genutzt werden, wie dies im Beispiel Wind der Fall ist, oder auch eine Brücke zwischen zwei Programmen darstellen. Wenn ein Kanal von einem anderen Programm abgefragt werden soll, kann dieses nicht selbst darauf zugreifen, weil die Treiber der AD-Wandler anscheinend nicht damit klar kommen, gleichzeitig von unterschiedlichen Programmen geladen zu werden. Deswegen sind weitere Interfaces angelegt (siehe 2.4).
Folgender Quellcode kann als Vorlage für die DLL (AD-Wandlung) genutzt werden:

```
library DieDLL;

{$mode objfpc}{$H+}

uses
  Classes, Callback;

var
 CB        : TCallback;
 OldMemMgr : TMemoryManager;
 ValueText : AnsiString = '';

{$R *.res}

procedure Create(Params : PChar; _Callback : TCallbackProc;
 Board,Channel : Word; MemMgr : TMemoryManager);
begin
 GetMemoryManager(OldMemMgr);
 SetMemoryManager(MemMgr);
 CB := TCallback.Create(_Callback,Board,Channel);
 // Start-Code hier
end;

function Calib(Flag : ShortInt; Value : Word) : PChar;
begin
 // Kalibrierung hier (Rückgabe: ValueText := 'Wert';)
 Result := PChar(ValueText);
end;

procedure Free;
begin
 // Exit-Code hier
 CB.Free;
 if (IsMemoryManagerSet) then SetMemoryManager(OldMemMgr);
end;

exports Create,Calib,Free;

end.
```

Die Unit Callback befindet sich im Codeverzeichnis der Hauptanwendung, welches als Quelltextverzeichnis hinzugefügt werden muss. In der Unit Callback befindet sich das Interface. Neben den drei exportierten Funktionen der DLL gibt es auch eine Eval-Funktion in der Anwendung, welche von den DLLs aufgerufen werden kann. Als Variablentransfer wird ein Pointer auf einen Array of Byte übergeben. Dieser kann von der Anwendung interpretiert werden und zur Übergabe von Werten aus der Anwendung an die DLL genutzt werden.
Da die Unit Callback von allen DLLs verwendet wird, dürfen die darin verwendeten Funktionen nicht verändert werden, jedoch können gefahrlos neue erstellt werden, die das Interface erweitern. Alle bisherigen sind folgender Maßen zu verwenden:

`procedure GetInterval`	Erwartet eine Methode vom Typ *procedure(NewInterval : Cardinal);* diese wird beim Aufruf von GetInterval und bei jeder Änderung des Abfrageintervalls aufgerufen und erhält das (neue) Intervall.
`procedure GetOverclockingAccess`	Erwartet eine Methode vom Typ *procedure(NewTime : TdateTime)*; und ein Pointer. Die erwartete Methode wird immer dann aufgerufen, wenn nach einer Messwertabfrage die eingeschobenen Zeitpunkte zwischen den Abfragen berechnet werden; NewTime ist hier der Zeitpunkt dieser letzten Messwertabfrage. Der Pointer wird auf einen *Array of TOverclockTimes* gesetzt. Dieser enthält die eingeschobenen Zeitpunkte (Null, wenn dieser Zeitpunkt nicht beachtet werden soll).
`property CalibOnRequest`	Wird diese Eigenschaft auf *true* gesetzt, werden alle Kalibrierungen, also auch die der eingeschobenen Zeitpunkte, sofort durchgeführt und nicht erst bei der Mittelwertberechnung vor dem Upload. Dies ist vor allem dann wichtig, wenn die eingeschobenen Zeitpunkte unabhängig vom Intervall sind. Jedoch muss sich die DLL so selbst um das anhalten der Abfrage kümmern (Siehe GetStarted)
`function GetStarted`	Gibt true zurück, wenn die Messwertabfrage gestartet ist
`function GetConnected`	Gibt true zurück, wenn die Verbindung zum Server gestarted ist
`procedure HTTPSend`	Trägt die übergebene Zeichenkette für das Upload ein. Siehe auch 2.5.1 und Dokumentation zu Version 1.0 .
`procedure ErrorReport`	Erwartet die Art und den Inhalt eines zu meldenden Fehlers bzw. Hinweises. Art = 0, für Meldung; 1 für Hinweis; 2 für Warnung; 3 für Fehler.
`function GetPrecision`	Gibt die für diesen Kanal eingestellte Rundungsgenauigkeit zurück.

Die Funktion Create wird nach dem laden der DLL aufgerufen, Free vor dem freigeben. Auch beim ändern der Kalibrierung wird die DLL neu geladen. Die Funktion Calib wird zum Kalibrieren der Messwerte aufgerufen. Flag ist hierbei die Nummer des entsprechenden, eingeschobenen Zeitpunktes (-1 für Hauptintervall). Das Result ist ein Pointer auf das erste Zeichen einer Zeichenkette (AnsiString). Diese wird unverändert in das Upload eingetragen. Deswegen dürfen darin nur für die URL zulässige Zeichen verwendet werden; ?, #, % und & sind ausgeschlossen, da diese schon für die Formatierung des Upload-Strings verwendet werden. Es gelten hier die Regeln zum Upload (Siehe 2.5.1).
Mit diesem Interface können die Abfragezeitpunkte auch komplett unabhängig vom Hauptintervall gewählt werden. Ignorieren Sie hierfür das Flag -1 (bzw. verwenden Sie es nur zur Erzeugung der Rückgabezeichenkette), setzen Sie CalibOnRequest auf *true* und Fügen Sie einen eigenen Abfragezeitpunkt ein, welcher mit jeder Abfrage dieses Punktes auf den nächsten, gewünschten Zeitpunkt gelegt wird.

2.4 Andere Kanaltypen nutzen

Das in 2.3.1 erklärte Interface steht nur für AD-Kanäle zur Verfügung. Je nach Board-Typ gibt es auch DA- oder Counter-Kanäle. Die Verwendung anderer Kanaltypen ist im Quelltext bereits angelegt und wurde mit *TDAChannel* exemplarisch durchgeführt.
Die Deklaration von *TADChannel* und *TDAChannel* befindet sich in der Datei Fenster.pas. Hier finden Sie auch Hinweise, wo im Quelltext neue Kanaltypen extra behandelt werden müssen. Wenn ein neuer Typ nur als Interface zur Verwendung in anderen Programmen verwendet werden soll, sollte dies reichen. Andernfalls müssen Funktionen, wie *Abrufen* oder *DoTimer* bearbeitet werden, damit die Abfragen der neuen Kanäle mit dem Intervall synchronisiert und für das Upload eingetragen werden können.
Ein einfaches Interface für DA-Kanäle ist bereits angelegt. Dieses besteht nur aus Create und Free, wobei Create folgender Maßen deklariert ist:

```
procedure(Params : PChar; Callback : Pointer; Board,Channel : word)
```

Callback ist ein Pointer auf eine Methode folgenden Typs:

```
procedure(Board,Channel : word; Value : integer)
```

Diese leitet den übergebenen Wert (Value) direkt an den entsprechenden DA-Kanal weiter. Das übernehmen des Memmorymanagers ist in diesem Interface nicht notwendig.

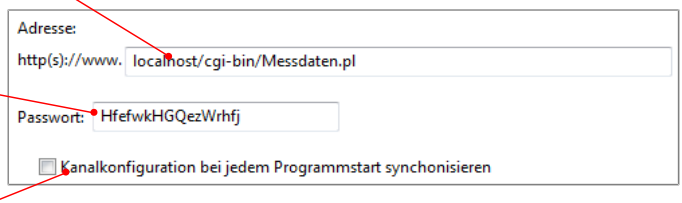
2.5 Das Upload

Das Upload ist ein rein HTTP-basiertes System. Auf dem Server befindet sich ein Script, das den URL-Anhang eines HTTP-Requests verarbeitet. Die Arbeitsweise dieses Systems ist noch die gleiche, wie in Version 1.0 und deswegen in der Dokumentation zu 1.0 näher erklärt.

Folgende Einstellungen sind für die Verbindung zum Server notwendig: Öffnen Sie die Einstellungen, Reiter *Upload*:

Die Adresse des Perl-Scripts auf dem Server.
Diese muss über HTTP erreichbar sein.

Das Passwort, welches vor der Installation des Server-Teils festgelegt wird (Siehe 3.1)



Bei Wahl dieser Option werden Nahmen und Einheiten aller Kanäle mit jedem Programmstart zum Server übertragen. So kann verhindert werden, dass zum Beispiel eine manuelle Änderung (In den XML-Daten) an einem Kanal nicht übernommen wird.

2.5.1 Formatierung des Upload-Contents

Weil das Upload über HTTP-Request abgewickelt wird, dürfen sich in den zu übertragenden Daten nur in URLs gültige Zeichen aus dem ASCII-Code befinden. ?,#,% und & werden bereits für die Formatierung der Zeichenketten verwendet und dürfen nicht, bzw. nur gezielt verwendet werden. Es steht jedoch eine Umsetzung aller ASCII-Zeichen in Dezimalwerten zur Verfügung: %xxx ist die Umschreibung. Das xxx ist eine immer dreistellige Dezimalzahl, welche die Nummer des an dieser Stelle einzusetzenden Zeichens im ASCII-Code repräsentiert. Ein- und zweistellige Nummern sind mit 00x, bzw. 0xx einzutragen. Der so formatierte Code wird auf dem Server wieder zurückverwandelt. Soll zum Beispiel ein Und-Zeichen bis zur Webseite übertragen werden, gibt es ein Problem, weil der Server die Umschreibung wieder zurück verwandelt und das Und-Zeichen dann die Formatierung zerstört. Also ist auch auf der Webseite ein Script aktiv, das die Zeichen umsetzt. Es muss also das %-Zeichen erneut umschrieben werden (Beispiel: Ä = Ä = %038Auml; = %037038Auml;).
Wann immer Sie Werte eintragen, die über das Upload gesendet werden, muss diese Umschreibung beachtet werden. Das gilt zum einen für Namen und Einheiten von Kanälen und zum anderen im DLL-Interface für den Rückgabewert von *Calib*. Für die Übertragung des Javascriptes wird diese Umformatierung automatisch gemacht.
Für weitere Informationen lesen Sie die Dokumentation zu Version 1.0 / 1.2. In Version 1.0 war die Doppelformatierung des %-Zeichens noch nicht enthalten.

3 Die Serversoftware

3.1 Installation der Serversoftware

Zum Installieren des Systems auf dem Server müssen alle Daten im Ordner Server hochgeladen werden. Dabei ist zu beachten, dass im Zierordner CGI (Perl) ausgeführt werden kann. Außerdem muss dieser über HTTP erreichbar sein. Sollte es notwendig sein, das Perl-Script in einem anderen Ordner als die HTML-Datei abzulegen, so kann dies getrennt werden: Messdaten.pl + Settings und index.html + Images + Script. Das Perl-Script ist für Linux-Server ausgelegt, kann aber leicht, durch ändern der ersten Zeile (Siehe Kommentar), für Windows-Server umgeschrieben werden.
Im Ordner Settings wird unter anderem auch das Passwort gespeichert. Dieses ist in der Datei Settings.dat in Textform gespeichert und muss vor dem Upload eingegeben werden. Beispiel:

```
Password=HfefwkHGQezWrhfj
```

Wichtig ist, dass in dieser Datei **KEIN Absatz** ist, auch nicht am Ende. Der Grund hierfür ist die unterschiedliche Absatzformatierung. Während Windows einen zwei-Byte-Absatz (#13#10) verwendet, besteht dieser auf Linux nur aus einem Byte (#13). Da auf dem Server Linux läuft, führt ein zwei-Byte-Absatz zu unerwünschten Fehlern beim verarbeiten dieser Datei.
Um zu verhindern, dass das Passwort im Internet gelesen werden kann, muss die **Dateiberechtigung** für alle Dateien im Ordner Settings auf 600 gesetzt werden. Dadurch wird der Zugriff von Außerhalb verweigert.
Die Datei index.html enthält einen Website-Ausschnitt, welcher nur die Stylesheets der Homepage importiert. Ansonsten ist diese HTML-Site vollkommen autonom und wird per IFRAME in die Website eingebunden. Es wäre prinzipiell auch möglich, den HTML-Code in Typo3 einzufügen, aber dort erhält dieser keinen Zugriff auf den HTML-Header. Es wird jedoch unbedingt empfohlen, den Aufruf der Funktionen, welche die Daten herunterladen, vom Header der HTML-Datei aus durchzuführen. Dies

bedingen unterschiedliche Sicherheitsbestimmungen für Javascript, welche unter anderem verhindern, dass ein Script Daten von fremden Domains herunterlädt. Es kann deswegen, gerade in älteren Browsern, auch vorkommen, dass diese HTML-Seite keine Verbindung zum Server aufbauen kann. Eingebunden im IFrame hat dies bisher jedoch immer funktioniert.

3.2 Aufbereitung der Messwerttabelle

Die Datei Script/Settings.js ist exemplarisch so aufgebaut:

```
function HookFn (Input) {
 return 'Der Wert ist: ' + Input;
}

function EvalFn () {
 return 4 * 6 / 3;
}

function LoadSettings () {
 MessdatenURL = 'URL_Messdaten.pl';
 AddVitualValue('Virtueller Wert','ValOf("Kanal 1") * EvalFn','Einheit');
 CreateGroup('Gruppe 1','Kanal 1,Kanal 2, Virtueller Wert');
 HookValueOutput('Kanal 1',HookFn);
 LinkChannel('Kanal 2','http://www.weitereinformationenzumkanal2.xx/');
}
```

In diesem Abschnitt können Sie alle wesentlichen Einstellungen zur Anzeige vornehmen. Dies sind eigene Befehle zur Aufbereitung der Tabelle. Grün markiert sind eigene Funktionen, welche später vom Messdatensystem aufgerufen werden. MessdatenURL ist die Adresse der Messdaten.pl. Diese sollte als absoluter Pfad angegeben werden. Es stehen (blau markiert) folgende Befehle zur Aufarbeitung der Tabelle zur Verfügung:

`AddVirtualValue`	Fügt einen virtuellen Kanal hinzu. Virtuelle Kanäle enthalten Messwerte, welche nicht hardwareseitig gemessen wurden, sondern softwareseitig berechnet werden. Erwartet wird der Name das Kanals, ein zu interpretierender Ausdruck, welcher dessen Wert liefert und die Einheit des Kanals.
`CreateGroup`	Fügt Kanäle zu einer Gruppe zusammen. Erwartet den Namen der Gruppe, sowie eine Liste der darin enthaltenen Kanäle. In der Liste werden die Kanäle mit Komma getrennt; es darf kein Leerzeichen nach dem Komma stehen.
`HookValueOutput`	Erwartet einen zu hookenden Kanal und eine Funktion. Die Funktion wird immer dann aufgerufen, wenn der Kanal einen neuen Wert erhält; der Wert wird der Funktion übergeben (siehe oben). Es wird dann der Rückgabewert der Funktion in die Tabelle geschrieben.
`LinkChannel`	Erwartet den Namen eines Kanals und eine URL. Verlinkt diesen Kanal mit der URL. Das Class-Attribut des Links ist *ChannelLink*.
`ValOf`	Die Funktion ValOf erwartet den Namen eines Kanals; gibt den Wert dieses Kanals zurück.
`StrToFloat`	Versucht, eine Zeichenkette als Zahl zu bearbeiten. Falls die Zeichenkette eine gültige Zahl ist, wird in der Variable LastPrecision die Anzahl signifikanter Stellen des Wertes gespeichert.
`LastPrecision`	enthält die Anzahl signifikanter Stellen des letzten Wertes, der mit StrToFloat umgewandelt wurde. Wenn eine Funktion von HookValueOutput aufgerufen wird, so ist in dieser Variable immer die Genauigkeit des entsprechenden Kanals gespeichert.

AddVirtualValue, CreateGroup, HookValueOutput und LinkChannel müssen vor dem Aufruf von Initialize (Rot markiert) aufgerufen werden.
Dieses Script kann auch lokal, im Programm bearbeitet werden. Dafür muss im Perl-Script (am Anfang) die Variable *$JavascriptURL* auf die richtige URL (relativ) der Datei Script/Settings.js gesetzt werden. Wenn Sie nicht möchten, dass das Script vom lokalen Programm aus verändert werden kann, geben Sie '' ein.