Betriebliche Praxis

Anwendung als Datenschnittstelle

Zur Datenübertragung von Phoenix Contact Steuerung zu der Automatisierungsoftware IPSymcon

Halil Karagöde
31.07.2013

Fachhochschule Dortmund
University of Applied Sciences and Arts

0. Inhaltsverzeichnis

1. Aufgabenstellung

1.1. Aufgabe

Im Rahmen des Modulfaches „Betriebliche Praxis“ sollen die Aufgaben der Wertverarbeitung von analogen Signalen und die Datenübertragung zwischen der speicherprogrammierbaren Steuerung (SPS) von Phoenix Contact und der Automatisierungssoftware IP-Symcon erarbeitet werden. Dabei werden die Steuerung und die benötigten Software bereitgestellt.

1.2. Vorgehensweise

Die Werteverarbeitung der analogen Signale erfolgt in der Steuerung. Durch mathematisches Vorgehen kann der analoge Eingangswert zu dem erwünschten Ausgangswert verarbeitet werden. Diese Verarbeitung wird dem Benutzer nachhinein als eine Funktion zur Verfügung gestellt.
Die Datenübertragung zwischen der Steuerung und der Software erfolgt manuell, d.h. durch eine Anwendung, die zwischen diesen beiden Elementen aktiviert wird, erfolgt der Datenaustausch. Deshalb muss hierbei an den drei Stellen programmiert werden. Die Steuerung und die Software IP-Symcon werden hierbei als Sender und Empfänger zugleich dienen können.

1.3. Zielsetzung

1.3.1. Microsoft Visual Studio

Die zu programmierende Anwendung, die als eine Datenschnittstelle zwischen der Steuerung und der Software arbeiten soll, wird mithilfe der Programmiersprache Visual Basic .NET programmiert. Diese Programmiersprache gehört zu der Entwicklungsebene der Microsoft Visual Studio. Hierbei wird zur Programmierung der Anwendung die Software Visual Basic .NET Professional Version 2010 benutzt. Dieses Programm soll abhängig von der SPS bzw. der Software IP-Symcon arbeiten.

1.3.2. Phoenix Contact

Da die SPS von Phoenix Contact als Sender und Empfänger von Mess- und Nutzdaten arbeiten soll, muss dieser Datenverkehr der Steuerung kenntlich gemacht werden. Hierbei erfolgt das Schreiben und Lesen von Daten auf dem Datenserver FTP der Steuerung. Wenn die Steuerung kein FTP anbietet, so kann kein Datenaustausch erfolgen. Die hierbei benutzte Steuerung ILC 150ETH bietet jedoch FTP an, genauso wie die anderen Steuerungen der ILC Reihe.

1.3.3. IP-Symcon

Bei der Software IP-Symcon werden die Daten aus der Steuerung (via Anwendungsschnittstelle) entnommen und auf der benutzerfreundlichen Weboberfläche Webfront dargestellt. Andersherum können Daten von dieser Software an die Steuerung gesendet werden. Hierbei dienen zyklisch aufrufbare PHP Skripte zur Lesung und Schreibung von Daten.

2. Phoenix Contact

2.1. Unternehmen

Abbildung 1: Firmenlogo von Phoenix Contact [1]

Phoenix Contact ist ein deutsches Unternehmen in der Branche Automatisierungstechnik mit mehr als 12.800 Mitarbeitern. Dieser Marktführer bietet Produkte in der industriellen Verbindungstechnik, Automatisierungstechnik und Interfacetechnik.
Das Feldbussystem Interbus ist eine Erarbeitung von Phoenix Contact und hat sehr weite Verbreitung in der Automobilindustrie.

2.2. Relevante Produkte & Funktionen

Das Unternehmen bietet für den Kleinanwender Kleinsteuerungen, die für wenig Geld viel Automatisierung versprechen. Hierbei gibt es die Steuerungen der Axiocontrol-Serie (AXC) für Achsenkontrollen, Steuerungen der Inline-Serie (ILC) für Automatisierungsprozesse, Steuerungen der speziellen Inline-Serie (ME) für Maschinenbauprozess und die Steuerungen der 100er Klasse für die Wasserwirtschaft.
Bei dieser Ausarbeitung wurde die Steuerung ILC 150ETH (Art.Nr.: 2985330) benutzt. Hierbei kamen passende digitale und analoge Eingangs-/Ausgangskarten zur Einsatz.

2.3. Steuerung ILC 150ETH

2.3.1. Beschreibung und Funktion

Abbildung 2: SPS ILC 150 ETH von Phoenix Contact [2]

Die speicherprogrammierbare Steuerung ILC 150 ETH (Inline Controller 150 Ethernet) ist eine Kleinsteuerung für kleinere Anwendungen. Diese Steuerung kann via Ethernet mit anderen Teilnehmern oder mit dem Programmiergerät verbunden werden. Die Programmierung bzw.

Parametrierung erfolgt durch die Automationssoftware PC-WorX, die sich an die Norm IEC 61131 hält. Die Steuerung arbeitet auf einem Flash-File-System und enthält einen FTP-Server. Zahlreiche Protokolle wie HTTP, FTP, SMTP, SQL und viele weitere Protokolle werden von der Steuerung unterstützt.

2.3.2. Beanspruchung

Der Computer wird als Programmiergerät benutzt und wird über einen Netzwerkkabel mit der Steuerung verbunden. Neben der Steuerung werden noch eine Eingangskarte mit 16 digitalen Eingängen, eine Ausgangskarte mit 16 digitalen Ausgängen und eine Analogeingangskarte mit 8 analogen Eingängen beschaltet. Die Spannungsversorgung der Steuerung erfolgt durch die Netzspannung über einen 230V/20V Transformator von Siemens. Ein Kippschalter wird zu Aktivierungszwecken an einem digitalen Eingang geschaltet.

2.4. Programmiersoftware PC-Worx

2.4.1. Beschreibung und Funktion

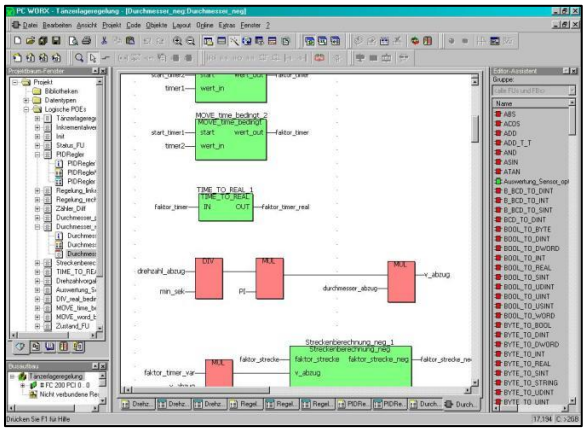


Abbildung 3: Benutzeroberfläche von PC-Worx [3]

PC-WorX ist eine Automatisierungssoftware, die das Parametrieren bzw. Programmieren der Steuerungen von Phoenix Contact ermöglicht. Hierbei sind Debugging, Einzelschritte bei Diagnose, Programm-/Wertänderung bei laufender Steuerung und das Programmieren in allen Programmiersprachen der IEC 61131 möglich.

2.4.2. Beanspruchung

Die Software wird als Programmiersoftware für die Steuerung benutzt. Bei laufender Steuerungen werden hierbei die aktuellen Werte beobachtet und entsprechend ausgewertet.

3. IP-Symcon

3.1. Unternehmen

Abbildung 4: Firmenlogo von IP-Symcon [4]

Die Symcon GmbH ist ein deutsches mittelständiges Unternehmen, die Produkte für die Gebäudeautomatisierung entwickelt. Das Unternehmen wird von diversen nationalen Initiativen und Forschungsinstituten gefördert.

3.2. Relevante Produkte und Funktionen

Das Hauptprodukt der Symcon GmbH ist die Software IP-Symcon. Dieses ermöglicht den Benutzer die Regelung und Steuerung von softwareprogrammierbaren Steuerungen über eine Weboberfläche, die online zugegriffen werden kann. Das Programm verbindet sich durch das entsprechende Interface mit der Steuerung und wechselt somit Daten aus. In der Basic Version von IP-Symcon werden dem Benutzer die üblichen Funktionen und Eigenschaften der Web-Visualisierung angeboten. Jedoch erweiterte Anpassungsmöglichkeiten der Webvisualisierung oder eigen-erstellte Windows Visualisierungen (Dashboard) werden nicht angeboten. Bei der Professional Version sind diese Funktionen vorhanden und der Nutzer kann gleichzeitig fünf solcher Webfronts nutzen. Bei der Unlimited Version gibt es bei der Variableanzahl keine Begrenzung und die Anzahl von Webfronts beträgt zehn und mehr.

3.3. Kompatibilität

IP-Symcon kann in der Funktechnik mit Komponenten verbunden werden, die mit dem Protokollen FS20, HMS, FHT, KS300 oder FS10 arbeiten. Funkgeräte wie Homematic oder Eaton können dabei mit entsprechenden Adapter oder Software ebenfalls mit der Software IP-Symcon verbunden werden.
Kabelgebundene Geräte, welche die Protokolle LCN, KNX/EIB, digitalSTROM, Modbus TCP/RTU oder M-Bus unterstützen, können ebenfalls mit der Software IP-Symcon kommunizieren. Somit ist es möglich, Steuerungen von Firmen wie Siemens, Vipa, Wago, Beckhoff oder ABB zu nutzen.

3.4. IP-Symcon Software

3.4.1. Beschreibung und Funktion

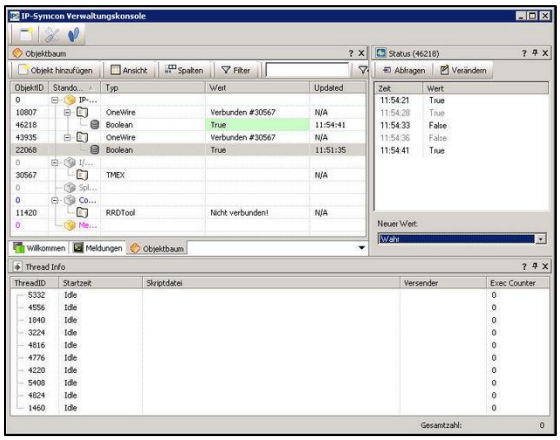


Abbildung 5: Benutzeroberfläche der Software IP-Symcon [5]

Mit der Software IP-Symcon wird dem Nutzer ein Visualisierungstool angeboten. Damit ist die Herstellung der Kommunikation mit der Steuerung, Regelung und Steuerung der Werte in der speicherprogrammierbaren Steuerung, Darstellung der Werte auf grafischer Ebene und Entwerfen von grafischen Oberflächen zur Darstellung bzw. Änderung der Werte (Dashboard) möglich.

3.4.2. Beanspruchung

Dieses Programm wird bei dieser Ausarbeitung zur Darstellung von Werten aus der Phoenix Contact SPS. Zugleich kann die Änderung dieser Werte ebenfalls erfolgen. Da die Steuerung keine direkte Kommunikation mit der Software IP-Symcon herstellen kann, da die entsprechenden Protokolle nicht vorhanden sind, wird hierbei die selbstprogrammierte Anwendung als Schnittstelle dienen.

3.5. Webfront

3.5.1. Beschreibung und Funktion

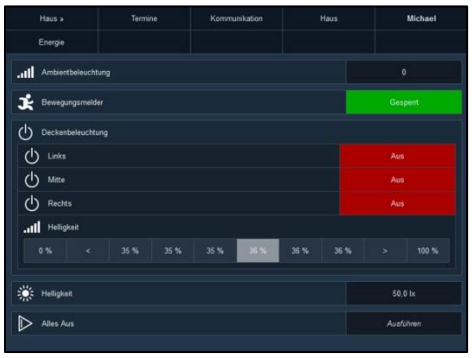


Abbildung 6: Benutzerberfläche von Webfront [6]

Die Webfront ist die Webvisiualisierung der Anwendung IP-Symcon. Dort werden alle Variablen und Skripte dargestellt. Je nach Steuerung bzw. Variable kann die Darstellung dieser Elemente geändert werden. Es sind bereits vordefinierte Ansichten vorhanden. Dem Nutzer wird auch angeboten, selber eine Darstellungsansicht der Werte zu erstellen.

3.5.2. Beanspruchung

Auf der Weboberfläche werden die Werte aus der Steuerung dargestellt. Diese Werte kommen ursprünglich aus der Phoenix Contact Steuerung. Jedoch die Verarbeitung und Anpassung erfolgt in der Schnittstellen –Anwendung. Ein PHP-Skript wird das Lesen und Schreiben der Aktualwerte übernehmen. Ein zyklisches Ereignis sorgt für die Aktualisierung der Werte und in eine Variable wird der Wert aus der Steuerung geschrieben.

4. Microsoft Visual Studio

4.1. Unternehmen

Abbildung 7: Firmenlogo von Microsoft [7]

Die Microsoft Corporation ist amerikanischer Software- und Hardwarehersteller mit ca. 94.000 Mitarbeitern. Eine seiner bekannten Produkte ist das Betriebssystem Windows und Office.
Das Unternehmen bietet diverse Produkte aus den Bereichen Betriebssysteme, Anwendungsprogramme, Serverprodukte, Entwicklungsumgebungen, Fernsehplattform, Spiele, Mobiltechnik und Visualisierungen.

4.2. Relevante Produkte und Funktionen

Die Entwicklungsumgebung Microsoft Visual Studio bietet verschiedene Hochsprachen wie Visual Basic .NET, C, C++, C#, HTML, Javascript und CSS.

4.3. Visual Basic .NET

4.3.1. Beschreibung und Funktionen

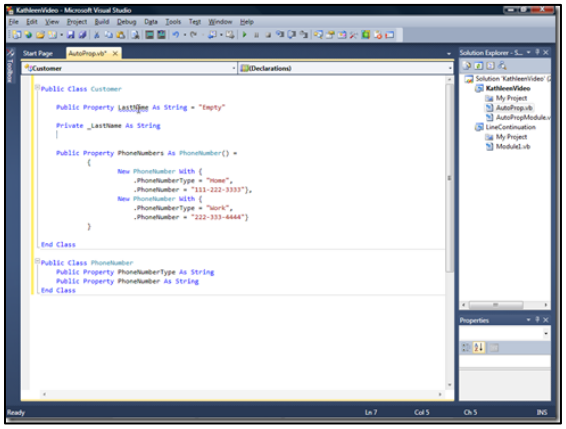


Abbildung 8: Programmieroberfläche von Visual Basic .NET [8]

Visual Basic .NET enthält die Bibliotheken und Befehle aus dem .NET Framework und ist der Vorgänger der Sprache Visual Basic 6. Diese Sprache unterstützt die objektorientierte Programmierung und die grafische Oberfläche. Ohne Objekte, Klassen und Instanzen ist die objektorientierte Programmierung in Visual Basic .NET nicht möglich.

4.3.2. Beanspruchung

Die Anwendung, die als Schnittstelle zwischen der Steuerung und der Software IP-Symcon dienen soll, wird in der Hochsprache Visual Basic .NET Professional 2010 programmiert. Hierbei wird eine grafische Oberfläche entworfen, die dem Nutzer die Benutzung der Anwendung erleichtert. Dahinter steckt die eigentliche Verarbeitung. Dabei werden der Steuerung via FTP die aktuellen Daten entnommen und der Software IP-Symcon übergeben. Somit sind folgende Funktionen für die Anwendung relevant: Verbindung mit dem FTP Server, Datenübertragung, Schreiben/Lesen und Darstellung von Daten.

5. Realisierung I (Analogwertverabeitung)

5.1. Vorgehensweise

Die Analogkarte liefert ein Wert von 2 Bytes (Word). Dieser Wert muss entsprechend skaliert und umgerechnet werden, sodass am Ende der Berechnung ein nützlicher Messwert vorhanden ist. Die Art des Messwertes spielt keine Rolle. Diese kann eine Temperatur, Distanz oder Gewicht sein.
In der Verarbeitung wird der analoge Messwert auf eine Messgröße rangiert und dimensioniert und entsprechend umgerechnet.

5.2. Verarbeitung

Der maximale Analogmesswert beträgt theoretisch 27648. Dieses entspricht ein Stromwert von 20mA am Analogeingang. Der Messwert beträgt 0 bei dem Stromwert 4mA. Das heißt, dass nur dieser Bereich als gültig gilt. Da am Eingang in der Realität ein größerer Strom von 20mA bzw. kleinerer Strom von 4mA liegen kann, müssen diese Fälle beachtet werden. Diese Bereiche dürfen wegen deren Unlinearität nicht als ein Messwert ausgegeben werden.

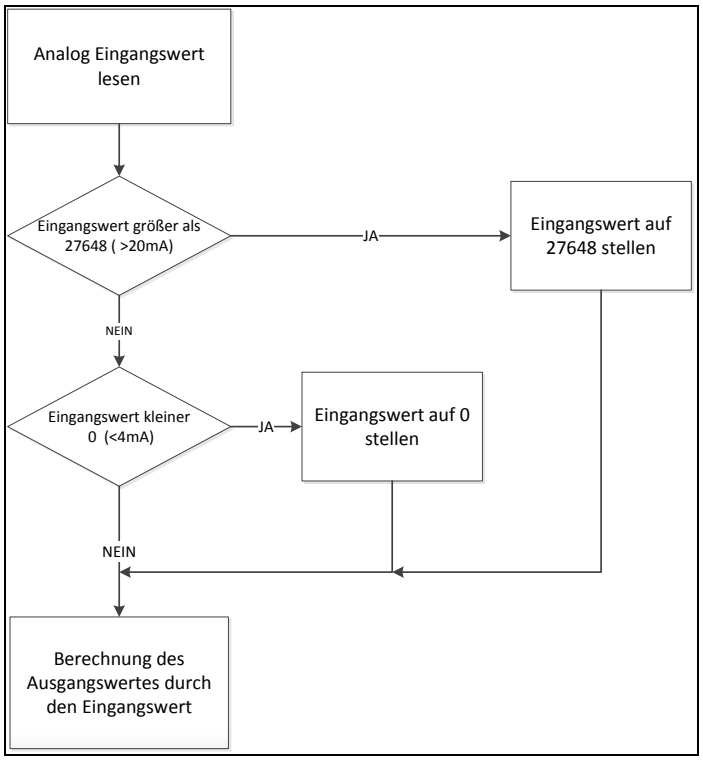


Abbildung 9: Programmablauf der Analogwertverarbeitung

Hierbei wird als erstes abgefragt, ob der gelesene analoge Eingangswert größer als der maximal messbarer Wert von 27648 (20mA) ist. Falls das der Fall ist, so wird stetig dieser maximale Wert angenommen. Falls der Eingangswert jedoch kleiner als 0 (4mA) ist, so wird dieser minimale Wert angenommen. Falls keiner der beiden Fälle zutrifft, so wird der analoge Messwert übernommen. Am Ende der Verzweigungen erfolgt durch mathematische Hilfe die Berechnung des Ausgangswertes.

Für die Berechnung des Messwertes ist der untere Schrankenwert und der obere Schrankenwert notwendig. Der untere Schrankenwert sagt aus, welcher physikalische Wert (z.B. Temperatur) bei dem Eingangsstrom 0mA anliegt.

Die mathematische Formel sieht wie folgt aus:

$$\textbf{Ausgangswert} = \frac{\text{Max. Wert} - \text{Min. Wert}}{27648} * \text{Eingangswert} + \text{Min. Wert}$$

5.3. Programmierung

5.3.1. Variable

```
VAR_OUTPUT
        Value_OUTPUT :   REAL;
END_VAR
```

Die Ausgangsvariable ist vom Typ Real (d.h. dass am Ausgang ist eine Fließkommazahl).

```
VAR_INPUT
        Trigger :        BOOL;
        Measure_min :    REAL;
        Measure_max :    REAL;
        Analog_INPUT :   WORD;
END_VAR
```

Zur Triggerung dient die Eingangs-Bool-Variable. Als untere/obere Messwertschranken sind die Eingangsvariablen Measure_min/Measure_max, die als Fließkommazahl eingegeben werden. Der gemessene Analogwert wird üblicherweise vom Typ Word eingeschrieben.

```
VAR
        temp_messwert :  REAL;
        temp_mess_2 :    REAL;
        temp_mess_1 :    REAL;
        new_messmax :    REAL;
END_VAR
```

Diese Variablen dienen nur als temporäre Variable. Da dieser Ablauf in AWL programmiert wird, muss bei jeder Operation eine Zuweisung bestehen.

5.3.2. Programmcode

```
(* Umwandlung nur bei Triggersignal *)
LD TRIGGER
jmpcn nomeas
```

Das Triggersignal entscheidet darüber, ob die Wertverarbeitung aktiviert werden soll. Erst wenn ein positives Triggersignal anliegt, erfolgt die Wertverarbeitung.

```
(* Analogwert in Temp laden *)
LD      Analog_INPUT
WORD_TO_REAL
ST      temp_messwert
```

Als erstes wird das analoge Signal, welche im Format WORD ist, ins Format REAL umgewandelt.

```
(* Analogwert zu groß *)
LD      27648.0
LT      temp_messwert
jmpcn   small

LD      27648.0
ST      temp_messwert
```

Wenn der gelesene Analogwert größer als der maximal zulässige Wert (27648) ist, so wird dieser maximale Wert angenommen. Wenn dieser Fall nicht zutrifft, dann wird zu dem nächsten Fall gesprungen.

```
(* Analogwert zu klein *)
small:
LD      temp_messwert
LT      0.0
jmpcn   calcu

LD      0.0
ST      temp_messwert
```

Wenn hierbei der gelesene Analogwert kleiner als der minimal zulässige Wert (0) ist, so wird dieser minimale Wert angenommen. Ansonsten wird zu der Berechnung gesprungen.

```
(* Berechnung der Messgröße *)
calcu:
LD      Measure_max
SUB     Measure_min
ST      new_messmax

LD      new_messmax
DIV     27648.0
ST      temp_mess_1

LD      temp_messwert
MUL     temp_mess_1
ST      temp_mess_2

LD      measure_min
ADD     temp_mess_2
ST      Value_OUTPUT
```

Hierbei wird die obengenannte Rechnungsformel in AWL umgesetzt.

```
(* Sprung für keine Aktion *)
nomeas:
```

Falls das Triggersignal nicht anliegt, so wird hierher gesprungen und es folgen keine weiteren Befehle.

5.4. Modul

5.4.1. Modul

Dieses Modul wird von dem Nutzer letztendlich benutzt. Hierbei wurde alles sauber zusammengefasst, sodass die Komplexität des Programmablaufs hinter diesem Modul für den Benutzer nicht sichtbar wird. Lediglich müssen wenige Eingänge und Ausgänge geschaltet werden.

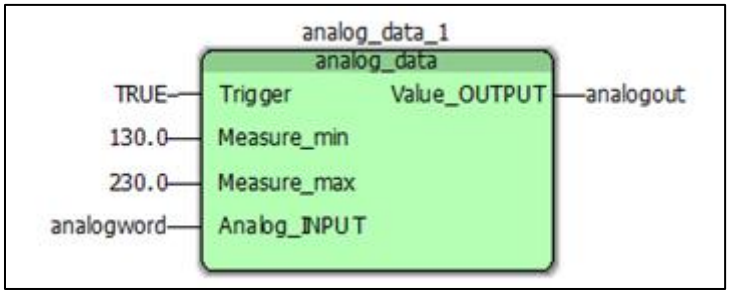


Abbildung 10: Modul zur Analogwertverarbeitung (Baustein: analog_data)

5.4.2. Eingänge/Ausgänge

Eingänge:

Trigger	BOOL	Triggersignal zur Auslösung der Umwandlung des Wertes
Measure_min	REAL	Untere Schranke des Messwertes bei 4mA bzw. 1V
Measure_max	REAL	Obere Schranke des Messwertes bei 20mA bzw. 10,5V
Analog_INPUT	WORD	Eingangswort von der Analogkarte

Ausgänge:

Value_OUTPUT	REAL	Berechneter Ausgangswert als Fließkommazahl

5.5. Test

5.5.1. Testwerte

Hierbei werden dem Eingang des Moduls folgenden Analogeingangswerte zugewiesen. Dementsprechend ergeben sich die Ausgangswerte.

Obere Schranke (Max.): *230 (bei 20mA)*
Untere Schranke (Min.): *100 (bei 4mA)*

Analogeingangs-wert	**Analogeingangs-strom** [I/mA]	**Ausgangswert** [REAL]
-1500	3,13	100,00
-1000	3,42	100,00
-500	3,71	100,00
0	4,00	100,00
500	4,29	102,35
1000	4,58	104,70
1500	4,87	107,05
2000	5,16	109,40
2500	5,45	111,75
3000	5,74	114,11
3500	6,03	116,46
4000	6,31	118,81
4500	6,60	121,16
5000	6,89	123,51
5500	7,18	125,86
6000	7,47	128,21
6500	7,76	130,56
7000	8,05	132,91
7500	8,34	135,26
8000	8,63	137,62
8500	8,92	139,97
9000	9,21	142,32
9500	9,50	144,67
10000	9,79	147,02
10500	10,08	149,37
11000	10,37	151,72
11500	10,66	154,07
12000	10,94	156,42
12500	11,23	158,77
13000	11,52	161,13
13500	11,81	163,48
14000	12,10	165,83

Analogeingangs-wert	**Analogeingangs-strom** [I/mA]	**Ausgangswert** [REAL]
15000	12,68	170,53
15500	12,97	172,88
16000	13,26	175,23
16500	13,55	177,58
17000	13,84	179,93
17500	14,13	182,28
18000	14,42	184,64
18500	14,71	186,99
19000	15,00	189,34
19500	15,28	191,69
20000	15,57	194,04
20500	15,86	196,39
21000	16,15	198,74
21500	16,44	201,09
22000	16,73	203,44
22500	17,02	205,79
23000	17,31	208,15
23500	17,60	210,50
24000	17,89	212,85
24500	18,18	215,20
25000	18,47	217,55
25500	18,76	219,90
26000	19,05	222,25
26500	19,34	224,60
27000	19,63	226,95
27500	19,91	229,30
27648	20,00	230,00
28000	20,20	230,00
28500	20,49	230,00
29000	20,78	230,00
29500	21,07	230,00
30000	21,36	230,00

5.5.2. Diagramm

Das typische Verhalten des Analogwertes ist das Linearitätsverhalten zwischen der unteren und der oberen Schranke (Messbereich). Bei 20mA bzw. 10,5V am Analogeingang liefert diese Karte der Steuerung den Wert 27648 im Format Word.

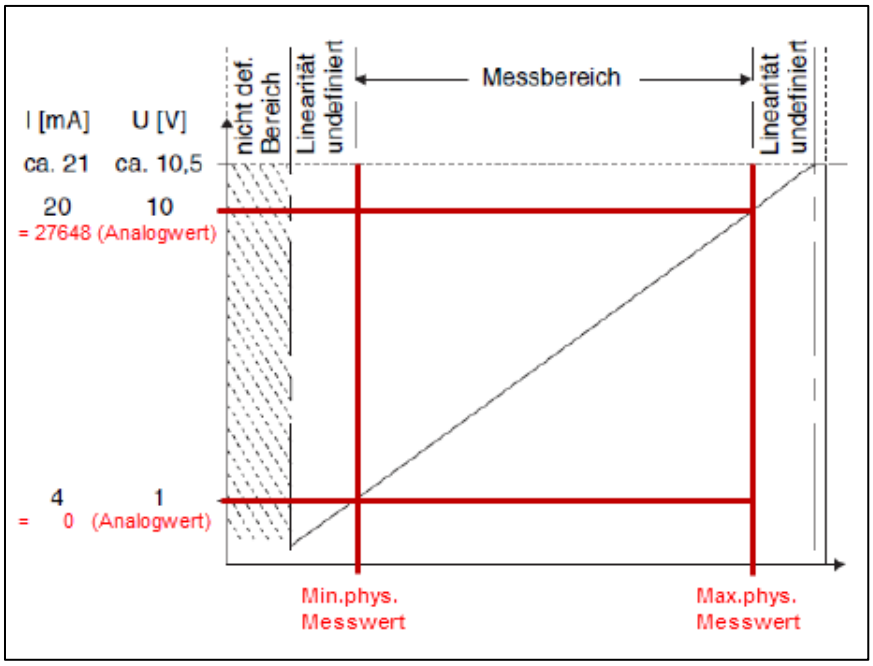


Abbildung 11: Verlauf physikalischer Messwert / Analogwert

Im realen Verlauf (nach den Testwerten von 5.5.1.) erkennt man ebenfalls die Linearität im Messbereich. Da wir außerhalb des Messbereiches das Minimum bzw. Maximum annehmen, unterscheidet sich der reale Verlauf vom theoretischen Verlauf.

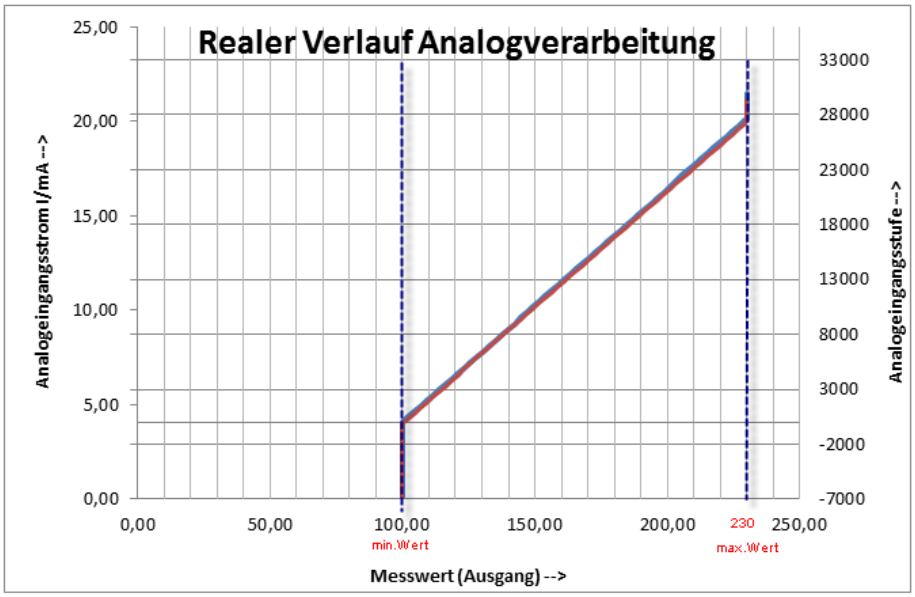


Abbildung 12: Realer Verlauf der Analogwertverarbeitung

6. Realisierung II (Phoenix Contact SPS)

6.1. Vorgehensweise

In der SPS von Phoenix Contact wird das zyklische Schreiben und Lesen von Textdateien erfolgen. Dabei wird als Speicherort der FTP-Server dieser SPS benutzt. Die Umwandlung der Werte mit lesbaren Formaten für die Schnittstellenanwendung erfolgt ebenfalls in der Steuerung. Durch Aufruf des bestimmten Moduls erfolgt das Schreiben oder Lesen von Daten.

6.2. Schreibvorgang

6.2.1. Schreibvorgang

Die Messwerte als Fließkommazahl werden dem Modul „SEND_TO_IPSYMCON“ für die Übertragung übergeben. Dabei wird die Fließkommazahl in eine Zeichenfolge umgewandelt und anschließend in die Textdatei auf dem FTP-Server gespeichert, die von der Schnittstelle bzw. IP-Symcon abgelesen werden.

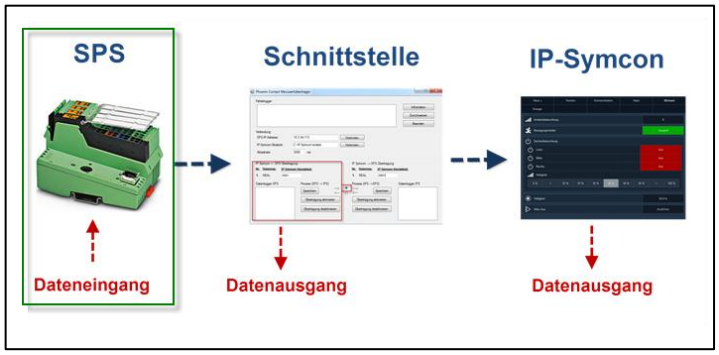


Abbildung 13: Richtung der Datenübertragung (SPS->IPS)

6.2.2. Ablauf des Schreibvorgangs

Da das Schreiben von Textdateien in einer SPS nicht direkt erfolgt, muss ein bestimmter Programmablauf programmiert werden. Hierbei wird erst dem Modul „SEND_TO_IPSYMCON“ ein Realwert zugewiesen. Dieser wird in einer Unterfunktion in das String-Format umgewandelt. Dabei wird ab der zweiten Nachkommastelle abgeschnitten. Dieser Wert wird dann gebuffert bzw. zwischengespeichert. Danach wird die Textdatei geöffnet, der Inhalt des Buffers in die Datei geschrieben und anschließend wird die Datei geschlossen. Danach fängt der gleiche Prozess von Anfang an, solange die Datenübertragung aktiviert und gültig ist.

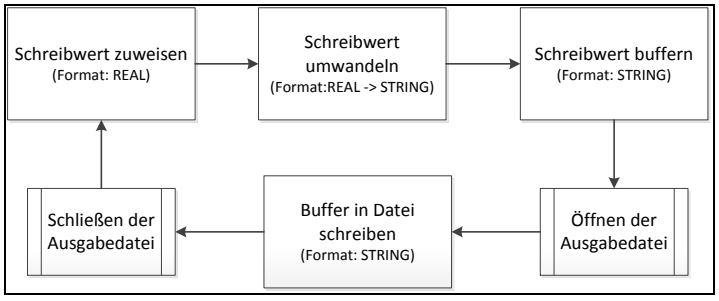


Abbildung 14: Ablauf Buffern und Schreiben von Daten beim Schreibvorgang

6.3. Lesevorgang

6.3.1. Lesevorgang

Beim Lesevorgang wird eine Textdatei (die von der Schnittstelle bzw. IP-Symcon kommt) auf dem FTP-Server geöffnet und der Inhalt aufgerufen. Die Ausgabe ist eine Zeichenfolge, die wiederum in eine Fließkommazahl umgewandelt wird und anschließend in dem Modul „RECEIVE_FROM_IPSYMCON“ eine Variable zugewiesen wird.

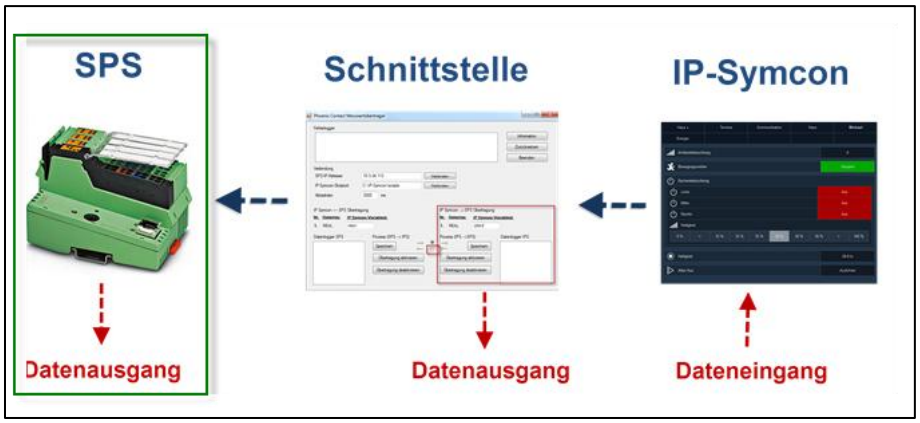


Abbildung 15: Richtung der Datenübertragung (IPS->SPS)

6.3.2. Ablauf des Lesevorgangs

Das Buffering muss auch beim Lesen von Dateien erfolgen. Als erstes wird hierbei die Textdatei geöffnet und der Inhalt wird in den Buffer geschrieben. Das Format ist hierbei üblicherweise String. Nach dem Buffern wird die Textdatei geschlossen und der Lesewert in einer Unterfunktion in das Real-Format umgewandelt. Bei der Umwandlung werden alle Zeichen des Wertes auf Korrektheit überprüft. Nur wenn Zahlen oder Trennzeichen in dieser Zeichenfolge enthalten sind, so wird dieser String direkt in das Real-Format umgewandelt. Am Ende wird dieser umwandelte Real-Wert in einem Modul eine Variable zugewiesen.

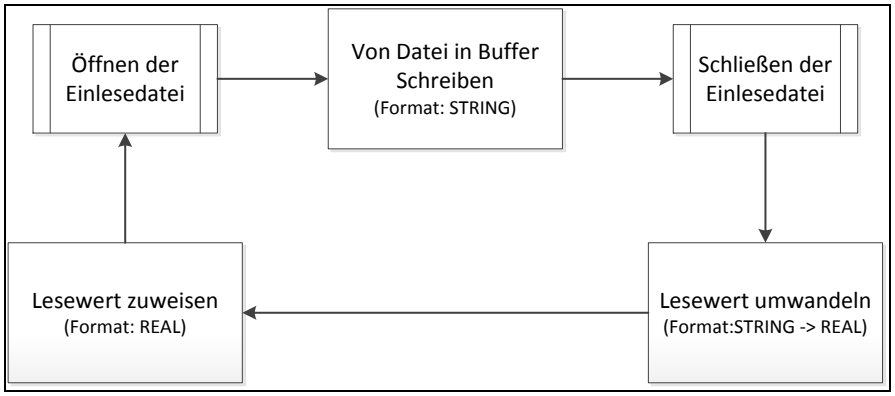


Abbildung 16: Ablauf Buffern und Lesen von Daten beim Lesevorgang

6.3.3. Modul zur Verarbeitung vom Textinhalt

Da der Inhalt der einzulesenden Textdatei eine Zeichenfolge ist, kann es in der Steuerung nicht direkt benutzt werden. Diese Zeichenfolge muss auf die Bedingungen zur Umwandlung in eine Fließkommazahl (Real) überprüft und dann erst umgewandelt werden.

```
VAR_INPUT
        _string :        STRING;
END_VAR
```

Als Eingang ist die Zeichenfolge aus der Textdatei definiert.

```
VAR_OUTPUT
        var0 :   BOOL;
        _real :  REAL;
END_VAR
```

Als Ausgabe des Moduls gibt es die „var0“, die angibt dass eine Umwandlung in eine Fließkommazahl möglich ist und die Variable „_real“, die den Wert aus dem Textinhalt als Fließkommazahl enthält.

```
VAR
        vgl_string :     STRING;
        Pos        :     INT;
        _stemp     :     STRING;
        string_length :  INT;
        zaehler :        INT;
        _stemp2 :        STRING;
END_VAR
```

Die Zwischenvariablen dienen zu internen Zwischenspeicherungen im Modul.

```
if EQ_STRING('0',_string)
THEN ;
ELSE
        var0:=FALSE;
        _stemp:=_string;
        _stemp2:=_string;
        string_length:=LEN(_string);  (* 6 *)
```

Als erstes wird überprüft, ob der Inhalt der Datei leer ist. Wenn dieses nicht zutrifft, so wird der Textinhalt in zwei Zwischenvariablen gespeichert. Außerdem wird die Anzahl der Zeichen dieser Zeichenfolge ermittelt.

```
        for zaehler:=1 TO string_length BY 1 DO
                pos:=string_length-zaehler;
                vgl_string := LEFT(_stemp , 1);
                _stemp := RIGHT(_string, pos);
```

Eine Schleife durchläuft sovielmal wie die Anzahl der Zeichen der Zeichenfolge. In der Schleife wird von der Zeichenfolge (Textdateiinhalt) das erste Zeichen jeweils abgeschnitten und zwischengespeichert.

```
                if EQ_STRING(vgl_string,'0') OR EQ_STRING(vgl_string,'1') OR EQ_STRING(vgl_string,'2') OR
                EQ_STRING(vgl_string,'3') OR EQ_STRING(vgl_string,'4') OR EQ_STRING(vgl_string,'5') OR
                EQ_STRING(vgl_string,'6') OR EQ_STRING(vgl_string,'7') OR EQ_STRING(vgl_string,'8') OR
                EQ_STRING(vgl_string,'9') OR EQ_STRING(vgl_string,'.')  OR EQ_STRING(vgl_string,' ')
                THEN ;
                ELSE var0:=TRUE;
                END_IF;
        END_FOR;
```

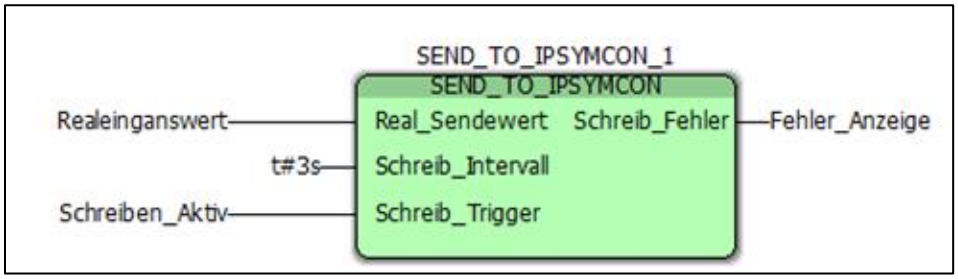
Das abgeschnittene Zeichen wird darauf überprüft, ob es eine Zahl oder ein Trennzeichen ist. Durch die Schleife werden dadurch alle Zeichen überprüft. Falls einer der Zeichen dieses Kriterium nicht erfüllt, so wird das Bit „var0“ gesetzt.

```
        if var0=TRUE
        THEN _real:=0.0;
        ELSE
                _real:=STRING_TO_REAL(_string);
        END_IF;
END_IF;
```

Falls das Bit „var0“ gesetzt wurde, weil einer der Zeichen aus dem Textinhalt keine Zahl oder Trennzeichen ist, so wird der Ausgang „_real“ auf 0 gesetzt. Falls das Bit nicht gesetzt wurde, so wird die gesamte Zeichenfolge direkt in das Real-Format umgewandelt und dem Ausgang „_real“ zugewiesen.

6.4. Module

6.4.1. Schreibmodul



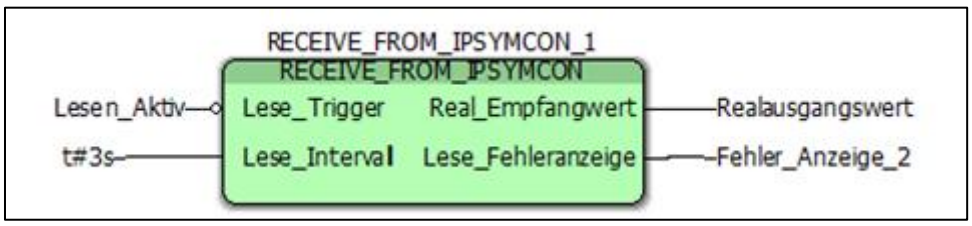
Eingänge:

Realeingangswert	REAL	Wert, welches IP-Symcon übergeben werden soll
Schreib_Intervall	TIME	Intervall zwischen den Schreibzugriffen (Zeit wie bei IPS und Schnit.)
Schreib_Trigger	BOOL	Trigger zur Auslösung des Schreibprozesses (Dauersignal möglich)

Ausgänge:

Schreib_Fehler	BOOL	Fehlerstatus (FALSE=Keine Fehler, TRUE=Schreibfehler vorhanden)

6.4.2. Lesemodul



Eingänge:

Lese_Trigger	BOOL	Trigger zur Auslösung des Leseprozesses (Dauersignal möglich)
Lese_Intervall	TIME	Intervall zwischen den Lesezugriffen (Zeit wie bei IPS und Schnit.)

Ausgänge:

Real_Empfangswert	REAL	Wert, welches von IP-Symcon kommt
Lese_Fehleranzeige	BOOL	Fehlerstatus (FALSE=Keine Fehler, TRUE=Lesefehler vorhanden)

7. Realisierung II (IP-Symcon)

7.1. Vorgehensweise

Ein Skript, zum Abrufen/Speichern der Textdatei mit dem aktuellen Messwert, wird zyklisch ablaufen. Dabei werden die Werte von der Variable (beim Lesevorgang) von IP-Symcon gelesen und in die Textdatei mit vordefinierten Zeitintervallen geschrieben. Beim Schreibvorgang werden die Werte aus der Textdatei (die von der SPS kommt) gelesen und in die Variable von IP-Symcon geschrieben. Die Variablennummer werden dabei in der Schnittstellenanwendung angegeben.

7.2. Skript

7.2.1. Lesevorgang

Beim Lesevorgang wird die Datei mit dem Wertinhalt (aus der Schnittstelle bzw. SPS) per PHP-Skript in eine Variable geschrieben. Dabei wird der Wert in das passende Zahlenformat von IP-Symcon angepasst. Die Aktualisierung des Wertes erfolgt in dem Schreib-Zeitintervall der SPS und der Schnittstelle. Hierbei müssen vom Benutzer keine Eingaben gemacht werden. Lediglich muss die Variable existieren und vom Typ Real, Integer oder Bool sein. Die Variablennummer wird in der Schnittstellen-Anwendung in eine Konfigurationsdatei im PHP-Skript Ordner von IP-Symcon gespeichert. Der Lesevorgang wird auch nur dann aktiviert, wenn bei der Schnittstelle die Übertragung aktiviert wird. Dabei wird ein Status-Flag von der Schnittstelle per Konfigurationsdatei zur IP-Symcon übermittelt.

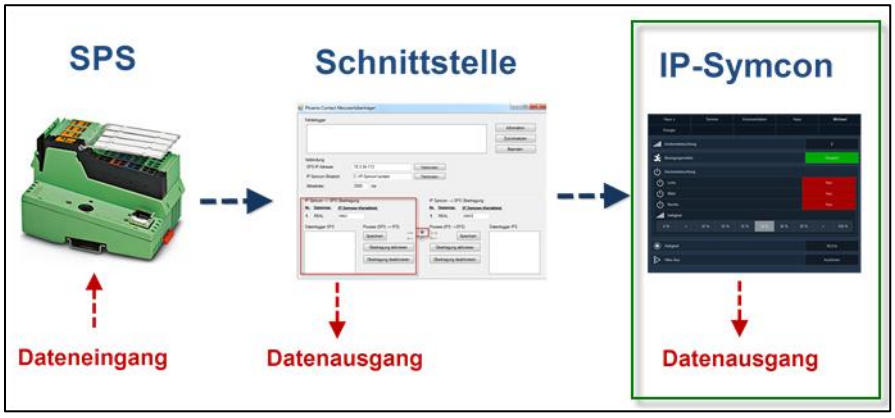


Abbildung 17: Richtung der Datenübertragung (SPS->IPS)

7.2.2. Schreibvorgang

Beim Schreibvorgang wird der Aktualwert der Variable von IP-Symcon in eine Textdatei geschrieben. Diese Variablennummer liegt ebenfalls in einer Konfigurationsdatei im Skript-Ordner und wird von der Schnittstellen-Anwendung beschrieben.

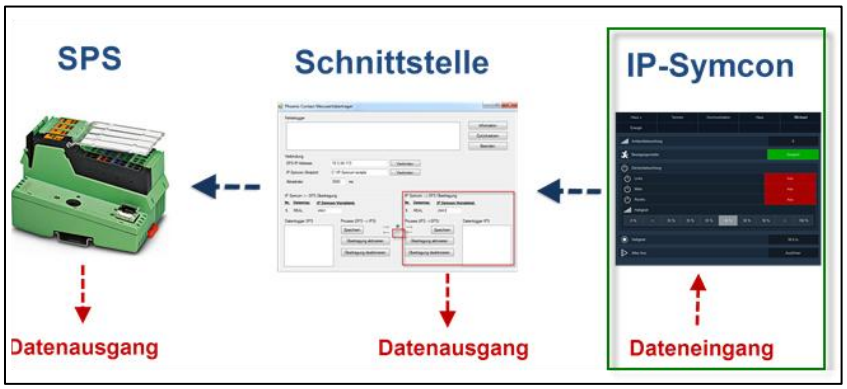


Abbildung 18: Richtung der Datenübertragung (IPS->SPS)

7.3. Zyklisches Ereignis

In einem bestimmten Zyklus wird der Lese-/Schreibprozess wiederholt. Dabei wird die Aktualität der Messwerte garantiert. Zuerst wird im PHP-Skript abgefragt, ob die Übertragung von SPS zur IP-Symcon (also bei der SPS Lesevorgang und bei IP-Symcon Schreibvorgang) aktiviert ist. Die Abfrage erfolgt durch den Status-Flag von der Anwendung zur IP-Symcon welche per Konfigurationsdatei übermittelt wird. Falls dieser aktiv ist, so erfolgt das Lesen der Textdatei (mit dem Wertinhalt aus der SPS) und das Schreiben dieses Wertes in eine Variable von IP-Symcon. Falls diese Übertragung nicht aktiviert ist oder nach dem Lesevorgang, so erfolgt die Abfrage ob die Übertragung in die andere Richtung aktiviert ist. Die Abfrage erfolgt ebenfalls durch einen Status-Flag per Konfigurationsdatei. Falls dieses der Fall ist, so wird der aktuelle Wert einer Variablen in die Textdatei geschrieben. Falls die Übertragung nicht aktiv ist oder der Schreibprozess zu Ende ist wird der ganze Prozess wiederholt.

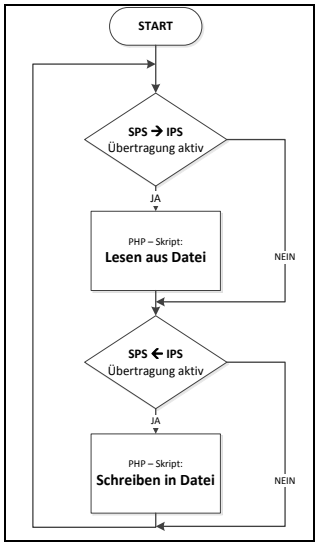


Abbildung 19: Zyklusablauf der Schreib-/Leseprozesse bei IP-Symcon

8. Realisierung II (Microsoft Visual Studio)

8.1. Vorgehensweise

Diese Anwendung dient zur Verbindung der Software IP-Symcon mit der Steuerung, Darstellung der Übertragungswerte und der allgemeinen Einstellungen. Die grafische Benutzeroberfläche erleichtert hierbei die Nutzung dieser Anwendung. Die Anwendung regelt dann den einwandfreien Datenfluss zwischen IP-Symcon und der SPS.

8.2. Grafische Benutzeroberfläche (GUI)

Die grafische Benutzeroberfläche bietet Vielzahl von Einstellungs- und Darstellungsmöglichkeiten. Viele der Funktionen können jedoch nur in Abhängigkeit der SPS und der Software IP-Symcon aktiviert werden.

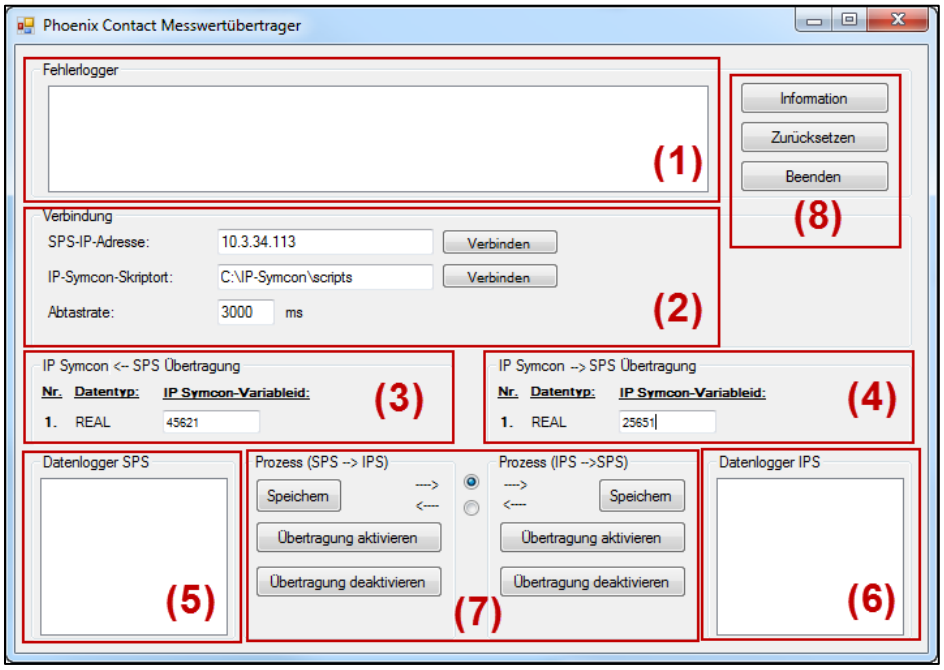


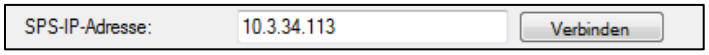
Abbildung 20: Benutzeroberfläche der Schnittstellenanwendung

(1)	Fehlerlogger	Listet und protokolliert die Fehler und Zustände
(2)	Verbindung	Verbindungseinstellungen zur Steuerung und IP-Symcon
(3)	Übertragungseinstellung	Variablenummer von IP-Symcon zum Lesen der Daten
(4)	Übertragungseinstellung	Variablenummer von IP-Symcon zum Schreiben der Daten
(5)	Datenlogger SPS	Listet die aktuellen Werte aus der Phoenix Contact SPS
(6)	Datenlogger IPS	Listet die aktuellen Werte aus IP-Symcon
(7)	Übertragung	Aktivierung der Übertragung zwischen SPS und IP-Symcon
(8)	Schaltelemente	Zur Information, Zurücksetzung und Beenden des Programms

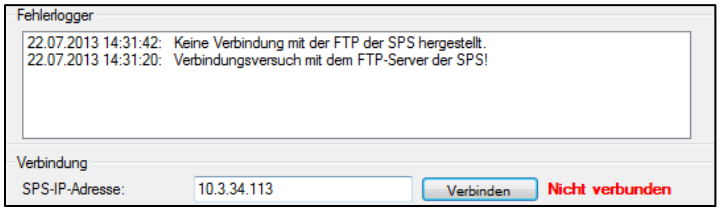
8.3. Verbindung

8.3.1. Verbindung mit der SPS

Als erstes muss die Verbindung mit der SPS von Phoenix Contact hergestellt werden. Dabei muss die IP-Adresse der Station (SPS) eingetragen werden. Dabei wird eine Anfrage auf den FTP Server der SPS geschickt. Wenn der Server erreichbar ist, so wird der Server verbunden.



Durch den Klick des Buttons „Verbinden“ erfolgt die Anfrage auf den FTP-Server „ftp://10.3.34.113/“ Wenn die Verbindung (nicht) erfolgreich hergestellt wurde, so wird dieser Zustand neben dem Button und in dem Fehlerlogger vermerkt.

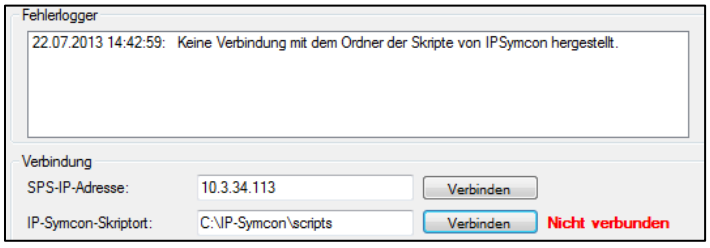


Ohne die Verbindung, kann keine Datenübertragung mit der Steuerung erfolgen.

8.3.2. Verbindung mit IP-Symcon

Danach muss die Verbindung mit der Software IP-Symcon hergestellt werden. Hierbei wird der Skriptordner der Software IP-Symcon eingetragen, weil dort das PHP Skript nachhinein erstellt wird, welches für die Datenübertragung sorgt. Hierbei wird intern abgefragt ob der angegebene Ordner existiert.

Durch den Klick des Buttons „Verbinden“ erfolgt eine Abfrage, ob das Verzeichnis „C:\IP-Symcon\scripts“ vorhanden ist. Es wird nicht überprüft ob dieses Verzeichnis das richtige Verzeichnis ist. Bei erfolgreichem oder nicht erfolgreichem Verbinden wird dieser Zustand ebenfalls neben dem Button und in dem Fehlerlogger vermerkt.



Ohne die Verbindung mit dem richtigen Ordner erfolgt keine richtige Datenübertragung zum IP-Symcon.

8.4. Datenübertragung

8.4.1. Daten lesen von SPS / Daten senden an IP-Symcon

Die Schnittstelle verbindet sich mit dem SPS. Es kopiert die Datei mit dem Wertinhalt aus dem FTP-Server der SPS in das Anwendungsverzeichnis. Der Wert wird dann im Datenlogger angezeigt und zugleich wird diese Datei mit dem Wertinhalt in den Skript-Ordner von IP-Symcon kopiert.

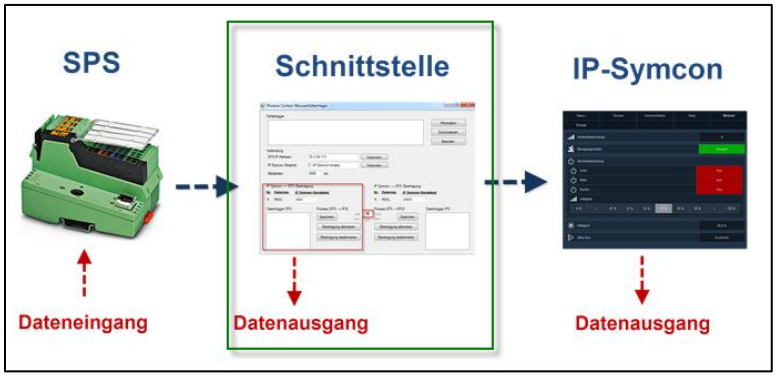
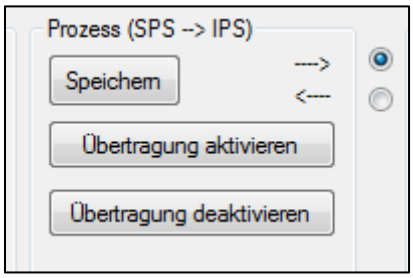


Abbildung 21: Richtung der Datenübertragung (SPS->IPS)

Die Richtung der Datenübertragung muss vom Benutzer festgelegt werden. Die Datenübertragung erfolgt im Hintergrund im vorgegebenen Intervall.



Die beiden Optionsfelder legen die Richtung des Datenflusses fest. Die obere Option ist die Datenübertragung von der SPS zu IP-Symcon. Durch Klick des Buttons „Übertragung aktivieren“ erfolgt die Datenübertragung im Hintergrund. Die Deaktivierung des Übertragungsprozesses erfolgt durch den Button „Übertragung deaktivieren“. Eine Sicherheitsfunktion sorgt für die Deaktivierung der entsprechenden Buttons nach Betätigung, sodass ein Laufzeitfehler vermieden wird.

8.4.2. Daten senden an SPS / Daten lesen von IP-Symcon

Die Schnittstelle kopiert beim „Lesevorgang“ die Textdatei mit dem Wertinhalt aus dem Skriptordner von IP-Symcon in das Anwendungsverzeichnis. Hierbei werden ebenfalls die aktuellen Werte in dem Datenlogger angezeigt. Diese Textdatei wird danach auf den FTP-Server der SPS kopiert.

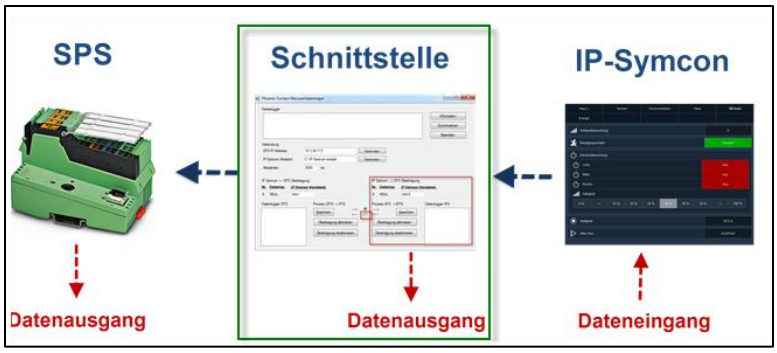
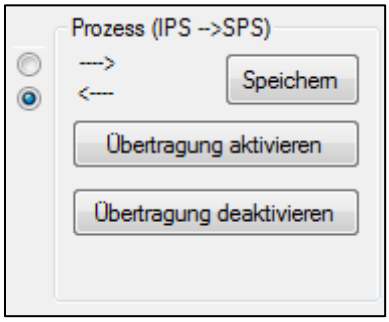


Abbildung 22: Richtung der Datenübertragung (IPS->SPS)

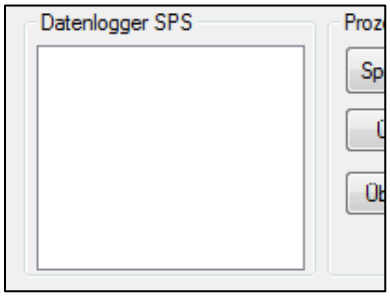


Bei Datenflussrichtung von IP-Symcon zur SPS muss die untere Option ausgewählt werden. Danach kann durch die entsprechenden Buttons die Datenübertragung gestartet oder gestoppt werden.

8.5. Datenverarbeitung

8.5.1. Datenlogger (SPS)

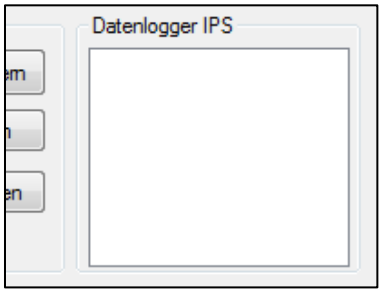
Damit der Nutzer die Datenübertragung verfolgen kann, nimmt der Datenlogger die aktuellen Messwerte auf und zeigt diese anschließend.



Das Schreiben in den Datenlogger erfolgt von der Anwendung automatisch nach Starten der Übertragung von der Steuerung zu IP-Symcon. Hier werden die aktuellen Abtastwerte der Steuerung angezeigt.

8.5.2. Datenlogger (IP-Symcon)

Es handelt sich hierbei um die identische Funktion wie beim Datenlogger (SPS).



Hierbei startet das Loggen nach Starten der Datenübertragung von IP-Symcon zur Steuerung. Es werden die Abtastwerte der Software IP-Symcon angezeigt.

8.5.3. Datenspeicherung

Zur Auswertung der Daten der Übertragung dient das Speichern dieser Daten als Textformat. Dabei wird neben dem Messwert auch der Zeitstempel aufgenommen.

Durch Klicken des Buttons „Speichern“ wird die Textdatei mit den Messwerten geöffnet. Dieser sollte in ein separates Verzeichnis kopiert werden.

8.6. Sonstige Eigenschaften und Einstellungen

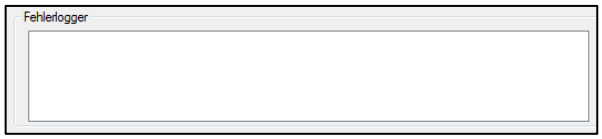
8.6.1. Abtastzeit

Die Abtastrate ist das Zeitintervall für das Schreiben bzw. Lesen von Daten. Je nach Anwendung sollte dieser entsprechend angepasst werden. Es gilt sowohl für das Lesen und Schreiben von der Steuerung und IP-Symcon. Je kleiner die Abtastrate ist, desto größer ist die Auflösung der Messwertreihe. Jedoch ist bei einer sehr niedrigen Abtastrate die Wahrscheinlichkeit eines Zugriffsfehlers auf den FTP-Server möglich. In dem Fehlerfall müsste die Steuerung neugestartet werden. Diese Zeit sollte mit den Zeiten der Steuerung und IP-Symcon übereinstimmen.

Wenn keine oder eine falsche Zeit angegeben wird, so wird der Defaultwert von 3000ms übernommen. Die Abtastrate wird beim Starten der Datenübertragung übernommen.

8.6.2. Fehlerlogger

In dem Fehlerlogger werden alle Fehler- und Zustandsmeldungen mit Zeitstempel aufgefasst. Dadurch wird die Suche nach Fehler im Fehlerfall erleichtert. Hierbei werden die Zustände der FTP-Verbindung zur SPS, Datenübertragungszustand und die Verbindung zur IP-Symcon aufgenommen. Die Fehler die in der Steuerung intern oder in der Software IP-Symcon intern auftreten, werden hier nicht angezeigt.

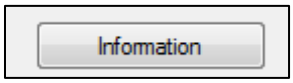


8.6.3. Rücksetzung der Daten

Die Rücksetzung setzt in der Oberfläche alle Daten zurück. Es dient nicht zur Rücksetzung der internen Fehler in der Steuerung oder IP-Symcon.

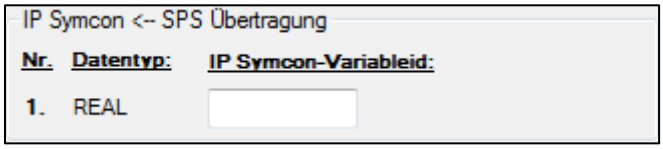
8.6.4. Information / Hilfe

Diese kleine Hilfe dient zur Erleichterung der Bedienung der grafischen Oberfläche.



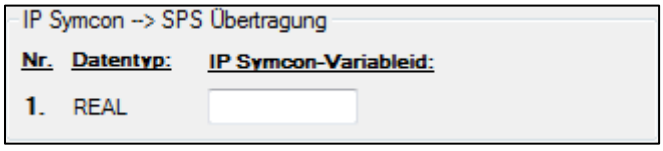
8.6.5. Variable-ID bei Übertragung von SPS zur IP-Symcon

Bei diesem Textfeld muss die Variablennummer von IP-Symcon eingegeben werden, bei denen die Messwerte aus dem SPS geschrieben werden. Die Variablennummer ist eine Pflicht und diese Variable muss in IP-Symcon existieren.



8.6.6. Variable-ID bei Übertragung von IP-Symcon zur SPS

Bei der Übertragung der Messwerte von IP-Symcon zur SPS muss die Variablennummer angegeben werden, von denen die Werte gelesen werden und zur SPS gesendet werden.



9. Ergebnis

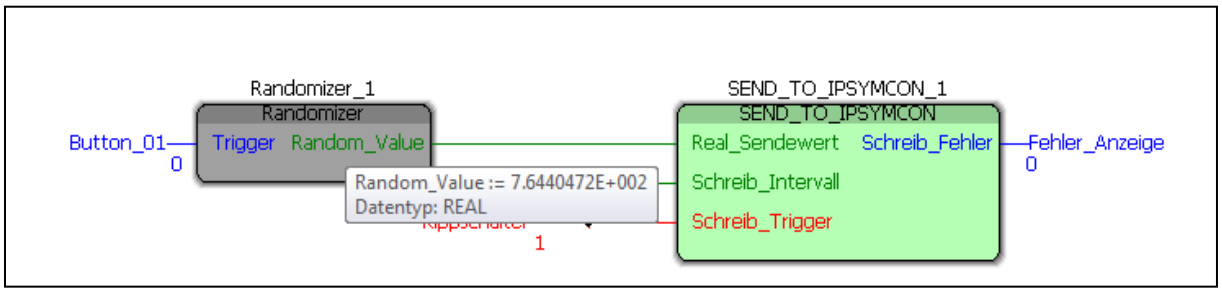
9.1. Analogwertverarbeitung

Die Auswertung der Analogwertverarbeitung ist unter dem Kapitel 5.5. .

9.2. Datenübertragung zwischen Steuerung und IP-Symcon

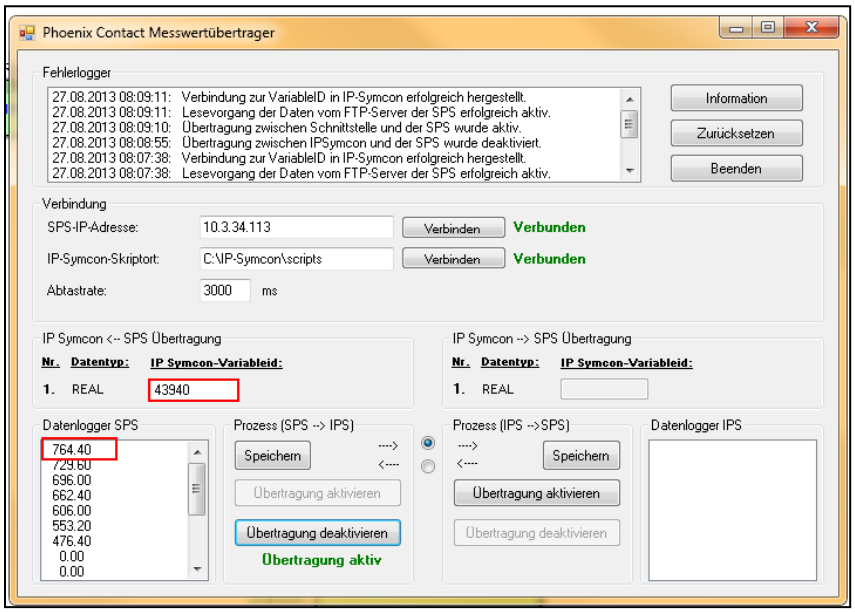
9.1. Steuerung

Hierbei wird durch einen Zahlenersteller das Modul zur Datenübertragung gefüttert.



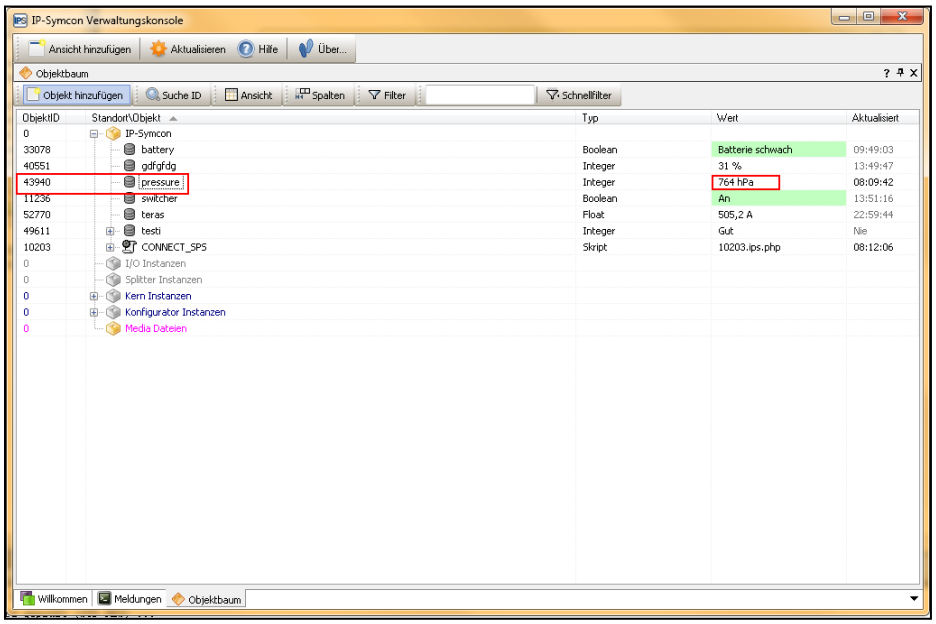
9.2. Schnittstellen-Anwendung

Die Zahlen aus der Steuerung tauchen in der Schnittstelle auf und werden für IP-Symcon vorbereitet.



9.3. IP-Symcon

In IPSymcon werden die Werte letztendlich in die Variablen geschrieben. Das Format wird dementsprechend angepasst.



10. Fazit

Durch geschicktes Vorgehen und Programmieren ist es möglich, zwei oder mehrere verschiedene bzw. inkompatible Komponenten miteinander zu koppeln. In der Programmiersprache der Steuerungstechnik gibt es zu große Einschränkungen im Vergleich zu den Hochsprachen. Folglich kann der Programmierer nur durch Tricks und Erfahrung hierbei ein gewünschtes Resultat erzielen.

11. Abbildungsverzeichnis

12. Quellenverzeichnis

12.1. Textquellen

Kapitel 2:

http://de.wikipedia.org/wiki/Phoenix_Contact (Wikipedia, 07/2013)

https://www.phoenixcontact.com/online/portal/de?1dmy&urile=wcm%3apath%3a/dede/web/main/products/subcategory_pages/Compact_controllers_P-21-01/f708b4f8-45fa-4ae7-ab62-0828235ebe53 (Phoenix Contact, 07/2013)

https://www.phoenixcontact.com/online/portal/de?uri=pxc-oc-itemdetail:pid=2985330&library=dede&pcck=P-13-09-01-01&tab=1 (Phoenix Contact, 07/2013)

Kapitel 3:

http://www.ip-symcon.de/produkt/hardware/#xcomfort (IPSymcon, 07/2013)

Kapitel 4:

http://openbook.galileocomputing.de/einstieg_vb_2010/ (Thomas Theis, 2010)

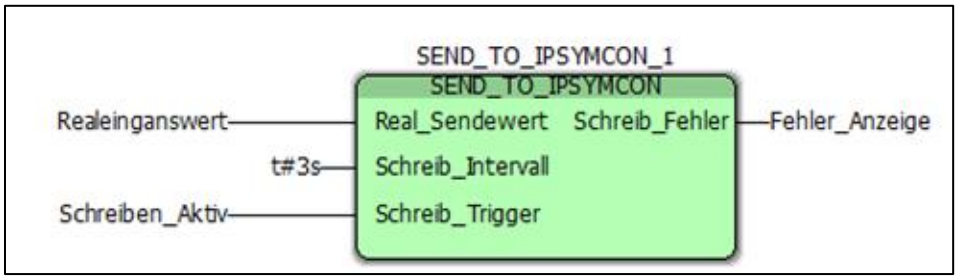
12.2. Abbildungsquellen

[1] http://www.dailygreen.de/wp-content/uploads/2009/05/phoenix-contact.jpg (Dailygreen, 07/2013)
[2] http://www.phoenixcontact.at/local_content_images/aut_ILC_150_eth_large.jpg (Phoenix Contact, 07/2013)
[3] http://www.et-inf.fho-emden.de/~elmalab/projekte/taenzer2/image010.jpg (Frank Erdmann, 07/2013)
[4] http://www.smarthome-deutschland.de/images/upload/members/logo/ipSymconLogo.png (Smarthome, 07/2013)
[5] http://www.heise.de/software/screenshots/t15398.jpg (Heise , 07/2013)
[6] http://www.hans-hats.de/images/ipsymconwebretro-600.jpg (Hans-Hats, 07/2013)
[7] http://www.interxion.com/Global/Homepage/Customer%20Logos/Microsoft.png (Interxion, 07/2013)
[8] http://blogs.msdn.com/blogfiles/kathleen/WindowsLiveWriter/VideoAutoImplementedPropertiesinVisualBa_ADED/image_3.png (Microsoft, 07/2013)

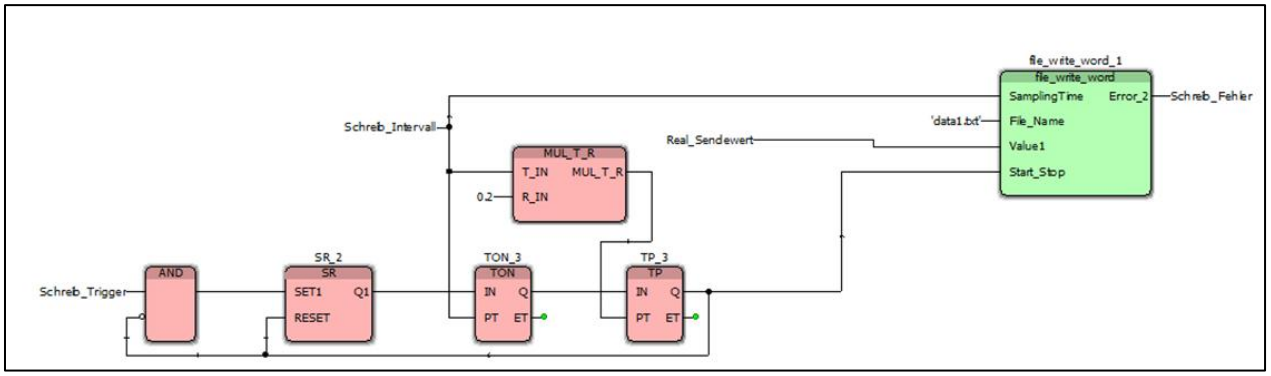
13. Quelltext (Phoenix Contact SPS)

13.1. Schreibprozess (von SPS zur IP-Symcon)

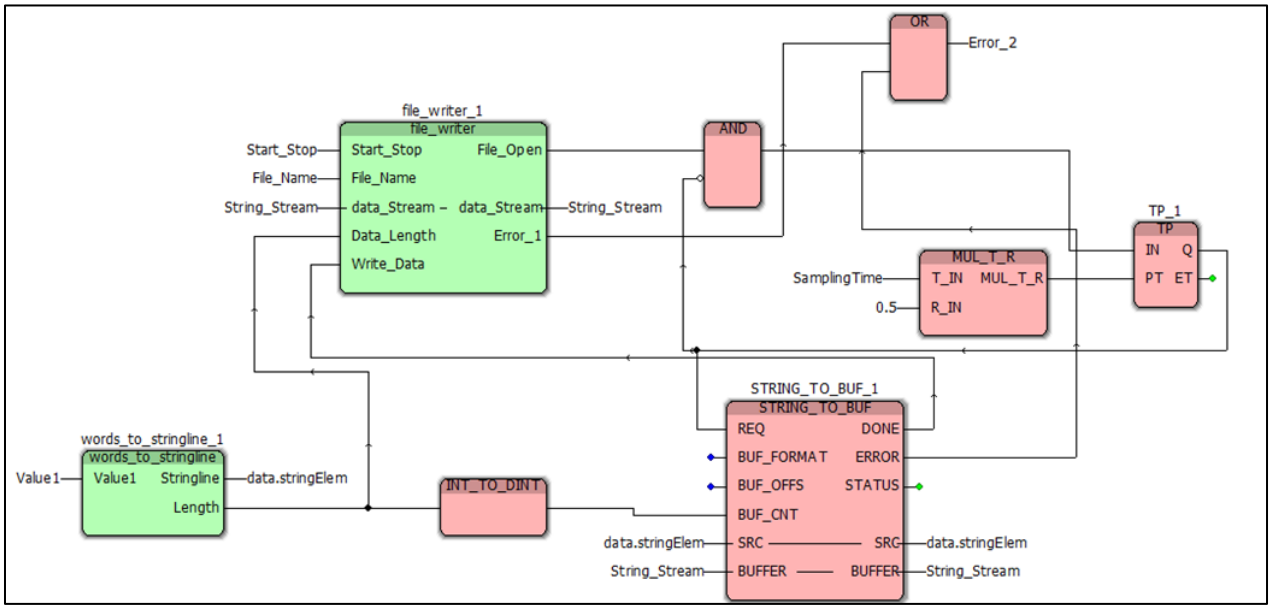
13.1.1. Modulaufruf zur Datenübertragung



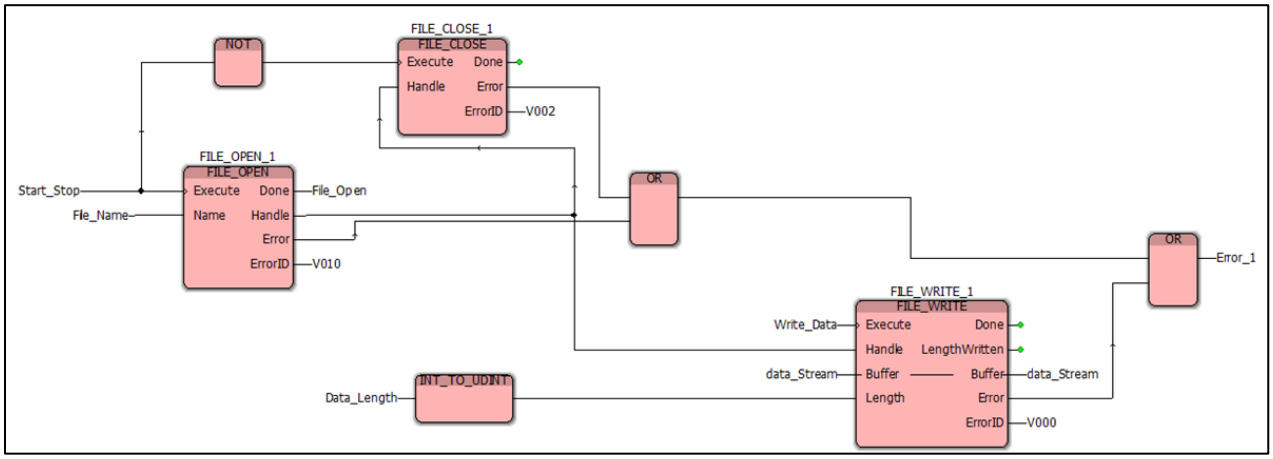
13.1.2. Modulinhalt „SEND_TO_IPSYMCON“



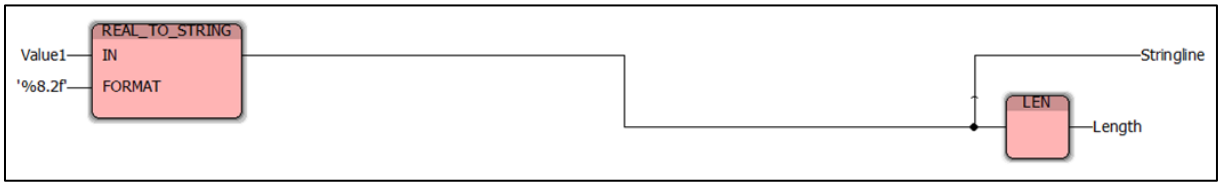
13.1.3. Unterfunktion „file_write_word“



13.1.4. Unterfunktion „file_writer“

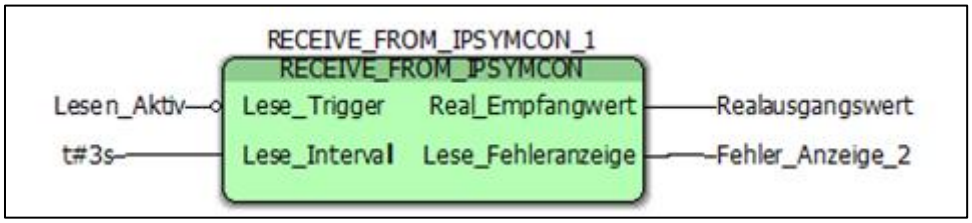


13.1.5. Unterfunktion „words_to_stringline“

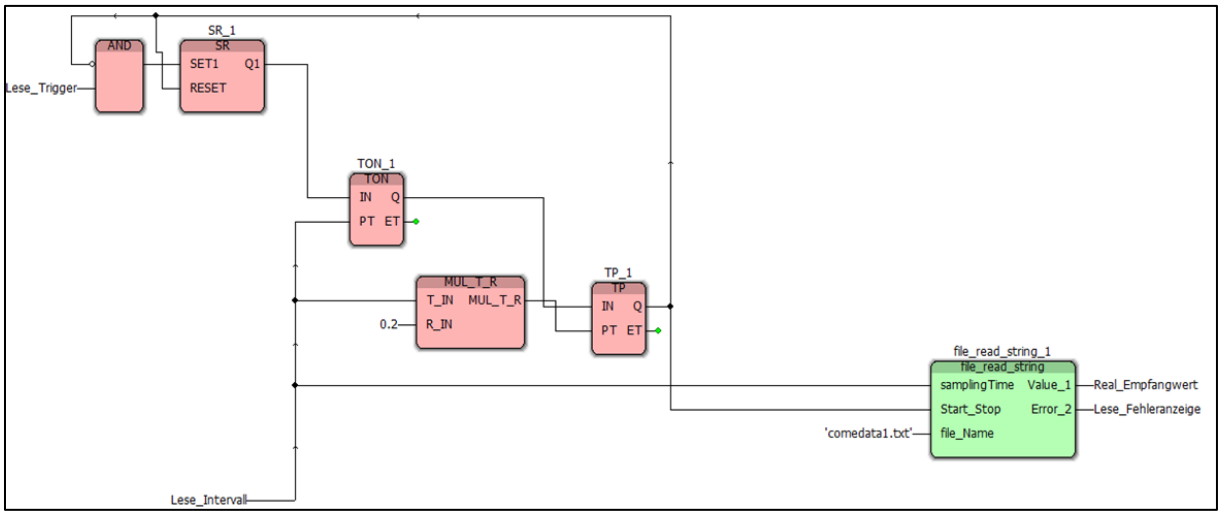


13.2. Leseprozess (von IP-Symcon zur SPS)

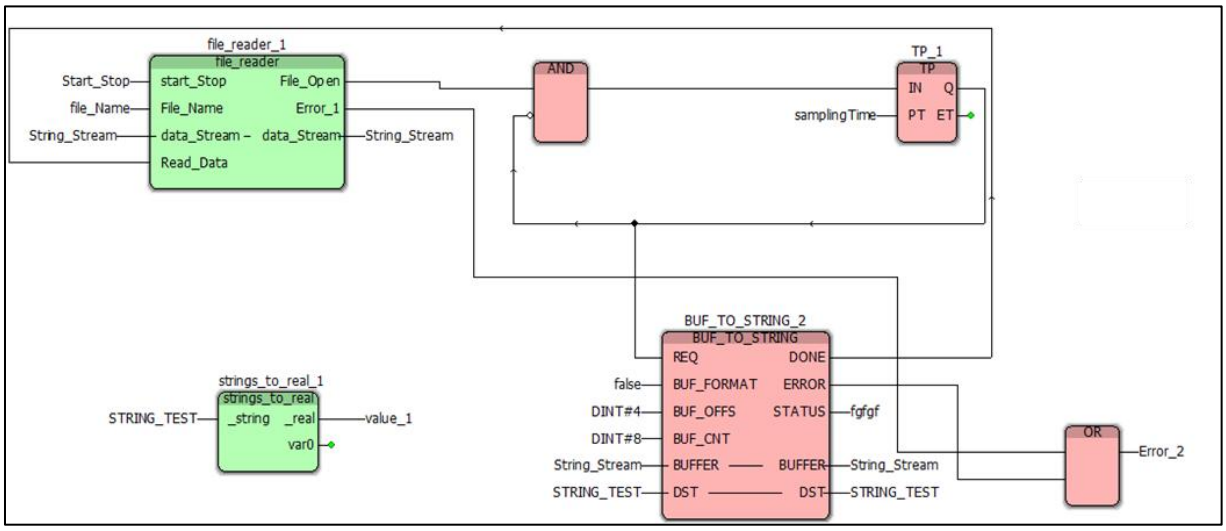
13.2.1. Modulaufruf zur Datenübertragung



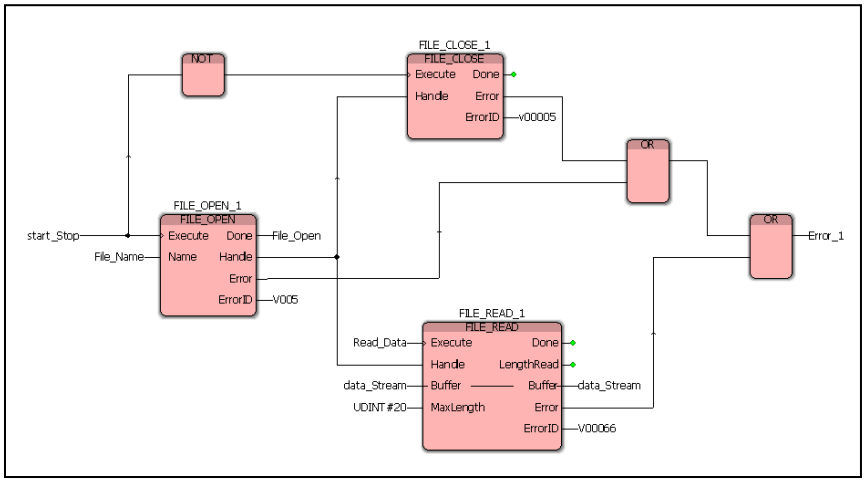
13.2.2. Modulinhalt „RECEIVE_FROM_IPSYMCON“



13.2.3. Unterfunktion „file_read_string“



13.2.4. Unterfunktion „file_reader“



13.2.5. Unterfunktion „strings_to_real“

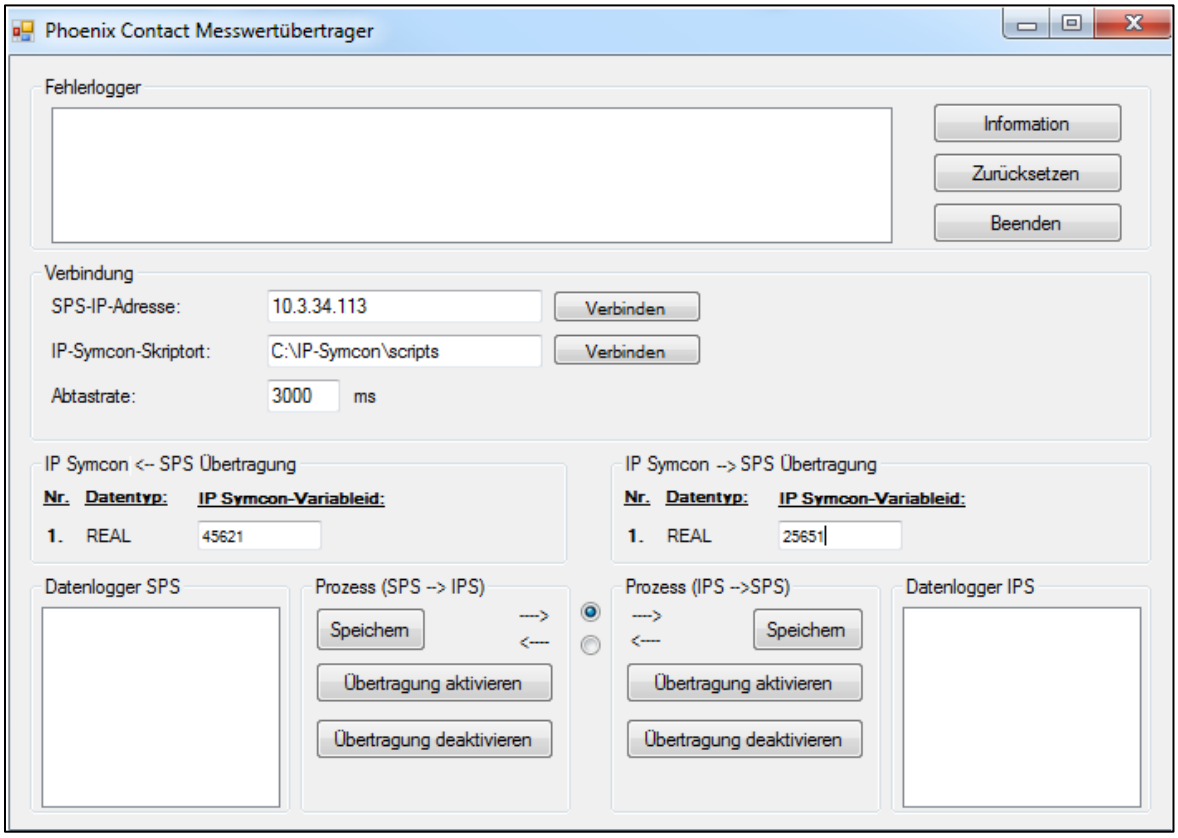
```
VAR_INPUT
        _string :       STRING;
END_VAR
VAR_OUTPUT
        var0 :  BOOL;
        real :  REAL;
END_VAR
VAR
        vgl_string :    STRING;
        Pos         :   INT;
        _stemp      :   STRING;
        string_length : INT;
        zaehler :       INT;
        _stemp2 :       STRING;
END_VAR


if EQ_STRING('0',_string)
THEN ;
ELSE
var0:=FALSE;
_stemp:=_string;
_stemp2:=_string;
string_length:=LEN(_string);  (* 6 *)
for zaehler:=1 TO string_length BY 1 DO
                pos:=string_length-zaehler;
                vgl_string := LEFT(_stemp , 1);
                _stemp := RIGHT(_string, pos);
                if EQ_STRING(vgl_string,'0') OR EQ_STRING(vgl_string,'1') OR
                EQ_STRING(vgl_string,'2') OR  EQ_STRING(vgl_string,'3') OR EQ_STRING(vgl_string,'4')
                OR EQ_STRING(vgl_string,'5') OR
                 EQ_STRING(vgl_string,'6') OR EQ_STRING(vgl_string,'7') OR EQ_STRING(vgl_string,'8')
                OR EQ_STRING(vgl_string,'9') OR EQ_STRING(vgl_string,'.') OR EQ_STRING(vgl_string,'
                ')
                THEN ;
                ELSE var0:=TRUE;
                END_IF;
END_FOR;
if var0=TRUE
THEN _real:=0.0;
ELSE
        _real:=STRING_TO_REAL(_string);
END_IF;
END_IF;
```

14. Quelltext (IP-Symcon)

```
<?php


$dateihandle = fopen("hka_phoenix\bool3.txt","r");
$zeile = fgets($dateihandle,4096);


 if( $zeile=="1")
 {
                // WRITE IN IPSymcon

                $error_1="0";
                $dateihandle = fopen("hka_phoenix\config.txt","r");
                $vid1 = fgets($dateihandle,4096);

                if(IPS_ObjectExists((int)$vid1)){
                $dateihandle = fopen("hka_phoenix\ipdata_1.txt","r");
                $value1 = fgets($dateihandle,4096);
                SetValue((int)$vid1, (float)$value1);
                }
                else {  $error_1 = "1";  }

                $dateihandle = fopen("hka_phoenix\ips_error.txt", "w");
                fwrite($dateihandle, $error_1 . "\r\n");
                fclose($dateihandle);

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////


 }

$dateihandle = fopen("hka_phoenix\bool2.txt","r");
$zeile = fgets($dateihandle,4096);

if( $zeile=="1")
 {

        // READ FROM IPSymcon

                $error_1="0";$error_2="0";$error_3="0";$error_3="0";$error_4="0";$error_5="0";
                $dateihandle = fopen("hka_phoenix\config2.txt","r");

                $vid1 = fgets($dateihandle,4096);


                if(IPS_ObjectExists((int)$vid1)){
                        $dateihandle = fopen("hka_phoenix\ospsdata_1.txt","w");

                        if(GetValue((int)$vid1)==""){
                                $value1 =0;
                        }
                        else{
                                $value1 = GetValue((int)$vid1);}
                                $value1= number_format( (float) $value1 , 1 );

                                $value1 = str_replace(",", "", $value1);

                                fwrite($dateihandle, $value1 );
                                fclose($dateihandle);
                        }
                        else {  $error_1 = "1";  }

                        $dateihandle = fopen("hka_phoenix\sps_error.txt", "w");
                        fwrite($dateihandle, $error_1 . "\r\n");
                        fclose($dateihandle);

 }

?>
```

15. Quelltext (Microsoft Visual Studio)

15.1. Benutzeroberfläche



15.2. Quelltext

```
Imports System.IO, System.Net, System.Text
Public Class Form1
    Dim directorysps As String
    Dim directoryips As String
    Friend scombo(5) As Label
    Friend icombo(5) As Label
    Friend stext(5) As TextBox
    Friend itext(5) As TextBox
    Friend slabel(5) As Label
    Friend ilabel(5) As Label
    Friend snr(5) As Label
    Friend inr(5) As Label
    Dim x1, y1 As Integer
    Dim elements As Integer = 5
    Dim connection_ok As Boolean = False
    Dim V_ID_BUG As Boolean = False
    Dim V_ID_BUG2 As Boolean = False
    Dim transfer_activ As Boolean = False
    Dim transfer_activ2 As Boolean = False
    Dim req As Boolean = False
    Dim req2 As Boolean = False
    Dim dirlocation As String
    Private Delegate Sub dlgUpdateStatus(ByVal text As String)
    Dim timer As Integer
    Dim flag, flag2, flag3 As Boolean


    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        File.Delete("output_sps.txt")
        File.Delete("output_ips.txt")
        Me.Close()
    End Sub



    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        buttonenabled()
        createelements()
        Init()

    End Sub

    Public Sub createelements()
        y1 = 40
        For i = 1 To 1 Step 1
            inr(i) = New Label
```

```
        With inr(i)
            .AutoSize = True
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 7.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .Location = New System.Drawing.Point(6, y1 + 3)
            .Name = "inr(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(22, 13)
            .TabIndex = 9
            .Text = i & ". "
            .Visible = True
        End With


        icombo(i) = New Label
        With icombo(i)
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 7.0!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .Location = New System.Drawing.Point(31, y1 + 3)
            .Name = "icombo(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(61, 20)
            .TabIndex = 9
            .Text = "REAL"
            .Visible = True
        End With


        itext(i) = New TextBox
        With itext(i)
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 8.0!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .Location = New System.Drawing.Point(101, y1 + 1)
            .Name = "itext(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(75, 18)
            .TabIndex = 9
            .Visible = True
        End With


        ilabel(i) = New Label
        With ilabel(i)
            .AutoSize = True
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .ForeColor = System.Drawing.Color.Red
            .Location = New System.Drawing.Point(180, y1 + 3)
            .Name = "ilabel(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(43, 13)
            .TabIndex = 10
            .Text = "     "
            .Visible = True
        End With



        ips.Controls.Add(inr(i))
        ips.Controls.Add(icombo(i))
        ips.Controls.Add(itext(i))
        ips.Controls.Add(ilabel(i))

        y1 = y1 + 25
    Next



    y1 = 40
    For i = 1 To 1 Step 1
        snr(i) = New Label
        With snr(i)
            .AutoSize = True
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 8.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .Location = New System.Drawing.Point(6, y1 + 3)
            .Name = "snr(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(22, 13)
            .TabIndex = 9
            .Text = i & ". "
            .Visible = True
        End With


        scombo(i) = New Label
        With scombo(i)
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 7.0!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .Location = New System.Drawing.Point(31, y1 + 3)
            .Name = "scombo(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(61, 20)
            .TabIndex = 9
            .Text = "REAL"
            .Visible = True
        End With


        stext(i) = New TextBox
        With stext(i)
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 7.0!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .Location = New System.Drawing.Point(101, y1 + 1)
            .Name = "stext(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(75, 18)
            .TabIndex = 9
            .Visible = True
        End With


        slabel(i) = New Label
        With slabel(i)
            .AutoSize = True
            .Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            .ForeColor = System.Drawing.Color.Red
            .Location = New System.Drawing.Point(180, y1 + 3)
            .Name = "slabel(" & CStr(i) & ")"
            .Size = New System.Drawing.Size(43, 13)
            .TabIndex = 10
            .Text = "     "
            .Visible = True
        End With


        sps.Controls.Add(snr(i))
        sps.Controls.Add(scombo(i))
        sps.Controls.Add(stext(i))
        sps.Controls.Add(slabel(i))

        y1 = y1 + 25
    Next

End Sub

Public Sub init()
    File.Delete(dirlocation & "bool2.txt")
    Dim myWriter As New StreamWriter(dirlocation & "bool2.txt", True)
    myWriter.Write("0")
    myWriter.Close()

    File.Delete(dirlocation & "bool3.txt")
    Dim myWriter2 As New StreamWriter(dirlocation & "bool3.txt", True)
    myWriter2.Write("0")
```

```
        myWriter2.Close()

    End Sub


    Private Sub saveip_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles saveip.Click


        If (spsip.Text = "" Or spsip.TextLength > 15) Then
            MsgBox("Bitte die IP-Adresse der SPS in Form XXX.XXX.XXX.XXX eingeben!")
        Else
            buglog.Items.Insert(0, DateTime.Now & ":   Verbindungsversuch mit dem FTP-Server der SPS!")

            Dim request = _
            DirectCast(WebRequest.Create _
            ("ftp://" & spsip.Text), FtpWebRequest)

            request.Credentials = _
            New NetworkCredential("", "")

            request.Method = WebRequestMethods.Ftp.ListDirectory

            Try
                Using response As FtpWebResponse = _
                DirectCast(request.GetResponse(), FtpWebResponse)
                    ' Folder exists here
                    req = True
                End Using
            Catch

            End Try

            If (req = True) Then
                directorysps = "ftp://" & spsip.Text & "/"

                ipstatus.Text = "Verbunden"
                buglog.Items.Insert(0, DateTime.Now & ":   Verbindung mit dem FTP-Server der SPS hergestellt..")
                ipstatus.ForeColor = Color.Green

                If (req2 = True) Then
                    ips.Visible = True
                    connection_ok = True
                    Button9.Enabled = True
                    Button6.Enabled = True
                    Button10.Enabled = True
                    Button4.Enabled = True
                End If
            Else
                buglog.Items.Insert(0, DateTime.Now & ":   Keine Verbindung mit der FTP der SPS hergestellt.")
                MsgBox("FTP der SPS konnte nicht erreicht werden!")
                ipstatus.Text = "Nicht verbunden"
                ipstatus.ForeColor = Color.Red
                connection_ok = False
            End If

        End If


    End Sub


    Private Sub savedirectory_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles savedirectory.Click

        If My.Computer.FileSystem.DirectoryExists(ipdirect.Text) Then
            directoryips = ipdirect.Text
            direstatus.Text = "Verbunden"
            buglog.Items.Insert(0, DateTime.Now & ":   Verbindung mit dem Ordner der Skripte von IPSymcon hergestellt.")
            direstatus.ForeColor = Color.Green
            req2 = True

            dirlocation = ipdirect.Text & "\hka_phoenix\"

            copytoips()



            If (req = True) Then
                connection_ok = True
                Button9.Enabled = True
                Button4.Enabled = True
                Button6.Enabled = True
                Button10.Enabled = True
            End If

        Else
            buglog.Items.Insert(0, DateTime.Now & ":   Keine Verbindung mit dem Ordner der Skripte von IPSymcon hergestellt.")
            MsgBox("Dieser Ordner konnte nicht erreicht werden!")
            direstatus.Text = "Nicht verbunden"
            direstatus.ForeColor = Color.Red
            connection_ok = False
        End If


    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
        If (abtast.Text >= 0) Then
            timer = abtast.Text
        Else
            timer = 2000
        End If



        If (spsips.Checked) Then
            w_Writevariableid()

            If connection_ok And Not V_ID_BUG Then
                Label3.Text = "Übertragung aktiv"
                Label3.ForeColor = Color.Green
                transfer_activ = True

                File.Delete("data1.txt")


                File.Delete(dirlocation & "bool3.txt")
                Dim myWriter2 As New StreamWriter(dirlocation & "bool3.txt", True)
                myWriter2.Write("1")
                myWriter2.Close()



                buglog.Items.Insert(0, DateTime.Now & ":   Übertragung zwischen Schnittstelle und der SPS wurde aktiv.")
                Button5.Enabled = True
                Button4.Enabled = False
                bgwping.RunWorkerAsync()
            Else
                Label3.Text = "Übertragung nicht aktiv"
                buglog.Items.Insert(0, DateTime.Now & ":   Übertragung zwischen Schnittstelle und der SPS wurde nicht aktiv.")
                Label3.ForeColor = Color.Red
```

```
            MsgBox("Übertragung nicht aktiviert! Verbindungen erst herstellen")
            transfer_activ = False
        End If
    End If
End Sub


Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    transfer_activ = False

    Button4.Enabled = True
    Button5.Enabled = False

    File.Delete(dirlocation & "bool3.txt")
    Dim myWriter2 As New StreamWriter(dirlocation & "bool3.txt", True)
    myWriter2.Write("0")
    myWriter2.Close()

    Label3.Text = "Übertragung nicht aktiv"
    buglog.Items.Insert(0, DateTime.Now & ":   Übertragung zwischen IPSymcon und der SPS wurde deaktiviert.")
    Label3.ForeColor = Color.Red
    bgwping.CancelAsync()

End Sub




Private Sub bgwPing_DoWork(ByVal sender As System.Object, ByVal e As System.ComponentModel.DoWorkEventArgs) Handles bgwping.DoWork
    Dim value_1, value_2, value_3, value_4, value_5 As String
    flag = False
    flag3 = False
    Do

        If bgwping.CancellationPending Then
            e.Cancel = True
            Exit Do
        End If

        Dim request As Net.FtpWebRequest = Net.FtpWebRequest.Create(directorysps & "data1.txt")

        request.Method = Net.WebRequestMethods.Ftp.DeleteFile
        request.Credentials = New Net.NetworkCredential("", "")

        '     Dim wc As New Net.WebClient
        '    wc.Credentials = New Net.NetworkCredential("", "")
        ' value_1 = wc.DownloadString(directorysps & "data1.txt")
        'value_1 = value_1.Replace(" ", "")
        'UpdateStatus(value_1)

        Try
            My.Computer.Network.DownloadFile(directorysps & "data1.txt", "data1.txt", "", "")
            If Not flag3 Then
                FTP_FAIL("Lesevorgang der Daten vom FTP-Server der SPS erfolgreich aktiv.")
                flag3 = True
            End If

        Catch
            FTP_FAIL("Fehler: Daten vom FTP-Server der SPS können nicht gelesen werden.")
            IO.File.Copy(System.AppDomain.CurrentDomain.BaseDirectory & "/bug/data1.txt", System.AppDomain.CurrentDomain.BaseDirectory & "data1.txt")
            flag3 = False
        End Try

        File.Delete("ips_error.txt")
        IO.File.Copy(dirlocation & "ips_error.txt", System.AppDomain.CurrentDomain.BaseDirectory & "ips_error.txt")
        Dim Datei2 As New StreamReader("ips_error.txt")
        value_2 = Datei2.ReadLine()
        Datei2.Close()


        If (value_2 = "1") Then
            FTP_FAIL("Fehler beim Schreibvorgang: Keine Verbindung zur VariableID in IP-Symcon hergestellt.")
        ElseIf (value_2 = "0" And Not flag) Then
            FTP_FAIL("Verbindung zur VariableID in IP-Symcon erfolgreich hergestellt.")
            flag = True
        End If


        Dim Datei As New StreamReader("data1.txt")
        value_1 = Datei.ReadLine()
        UpdateStatus(value_1)
        Datei.Close()

        File.Delete("data1.txt")
        File.Delete(dirlocation & "ipdata_1.txt")
        Dim myWriter3 As New StreamWriter(dirlocation & "ipdata_1.txt", True)
        myWriter3.Write(value_1)
        myWriter3.Close()



        Dim myWriter As New StreamWriter("output_sps.txt", True)
        myWriter.WriteLine(DateTime.Now & " ;   " & value_1)
        myWriter.Close()




        System.Threading.Thread.Sleep(timer)


    Loop


End Sub

Private Sub UpdateStatus(ByVal text As String)
    If datalog.InvokeRequired Then
        Dim dlg As New dlgUpdateStatus(AddressOf UpdateStatus)
        Me.Invoke(dlg, text)
    Else
        datalog.Items.Insert(0, text)



    End If

End Sub

Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button9.Click

    If (abtast.Text >= 0) Then
        timer = abtast.Text
    Else
        timer = 2000
    End If
```

```
    If (ipssps.Checked) Then
        s_Writevariableid()
        If req2 And Not V_ID_BUG2 Then
            Label11.Text = "Übertragung aktiv"
            Label11.ForeColor = Color.Green
            transfer_activ2 = True

            File.Delete(dirlocation & "bool2.txt")
            Dim myWriter As New StreamWriter(dirlocation & "bool2.txt", True)
            myWriter.Write("1")
            myWriter.Close()

            buglog.Items.Insert(0, DateTime.Now & ":   Übertragung zwischen IPSymcon und der Schnittstelle wurde aktiv.")
            Button9.Enabled = False
            Button8.Enabled = True
            bgwping2.RunWorkerAsync()

        Else
            Label11.Text = "Übertragung nicht aktiv"
            buglog.Items.Insert(0, DateTime.Now & ":   Übertragung zwischen IPSymcon und der Schnittstelle wurde nicht aktiv. Verbindungen kontrollieren.")
            Label11.ForeColor = Color.Red
            MsgBox("Übertragung nicht aktiviert! Verbindungen erst herstellen")
            transfer_activ2 = False
        End If
    End If




End Sub

Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button8.Click
    Button8.Enabled = False
    Button9.Enabled = True

    File.Delete(dirlocation & "bool2.txt")
    Dim myWriter As New StreamWriter(dirlocation & "bool2.txt", True)
    myWriter.Write("0")
    myWriter.Close()

    transfer_activ2 = False
    Label11.Text = "Übertragung nicht aktiv"
    buglog.Items.Insert(0, DateTime.Now & ":   Übertragung zwischen IPSymcon und der Schnittstelle wurde deaktiviert.")
    Label11.ForeColor = Color.Red
    bgwping2.CancelAsync()
End Sub

Private Sub bgwPing2_DoWork(ByVal sender As System.Object, ByVal e As System.ComponentModel.DoWorkEventArgs) Handles bgwping2.DoWork
    Dim value_1, value_2, value_3, value_4, value_5, temp_null As String

    flag2 = False
    Do
        temp_null = ""
        If bgwping2.CancellationPending Then
            e.Cancel = True
            Exit Do
        End If

        Dim FsyObjekt As Object
        FsyObjekt = CreateObject("Scripting.FileSystemObject")

        FsyObjekt.CopyFile(dirlocation & "ospsdata_1.txt", System.AppDomain.CurrentDomain.BaseDirectory)

        Dim Datei As New StreamReader("ospsdata_1.txt")
        value_1 = Datei.ReadLine()
        UpdateStatus2(value_1)
        Datei.Close()

        Dim myWriter As New StreamWriter("output_ips.txt", True)
        myWriter.WriteLine(DateTime.Now & " ;   " & value_1)
        myWriter.Close()


        For i = 1 To (12 - value_1.Length) Step 1

            temp_null = temp_null & "0"

        Next




        File.Delete("ospsdata_1.txt")
        Dim myWriter2 As New StreamWriter("ospsdata_1.txt", True)
        myWriter2.Write(temp_null & value_1)
        myWriter2.Close()

        Try
            My.Computer.Network.UploadFile("ospsdata_1.txt", directorysps & "comedata1.txt", "", "")
        Catch
            FTP_FAIL("Daten können auf den FTP-Server der SPS können nicht geschrieben werden.")
        End Try


        File.Delete("sps_error.txt")
        IO.File.Copy(dirlocation & "sps_error.txt", System.AppDomain.CurrentDomain.BaseDirectory & "sps_error.txt")
        Dim Datei2 As New StreamReader("sps_error.txt")
        value_2 = Datei2.ReadLine()
        Datei2.Close()


        If (value_2 = "1") Then
            FTP_FAIL("Fehler beim Lesevorgang: Keine Verbindung zur VariableID in IP-Symcon hergestellt.")
        ElseIf (value_2 = "0" And Not flag2) Then
            FTP_FAIL("Verbindung zur VariableID in IP-Symcon erfolgreich hergestellt.")
            flag2 = True
        End If



        System.Threading.Thread.Sleep(timer)



    Loop

End Sub

Private Sub UpdateStatus2(ByVal text As String)


    If datalog2.InvokeRequired Then
        Dim dlg As New dlgUpdateStatus(AddressOf UpdateStatus2)
        Me.Invoke(dlg, text)
    Else
        datalog2.Items.Insert(0, text)
```

```
        End If

    End Sub

    Private Sub FTP_FAIL(ByVal text As String)


        If buglog.InvokeRequired Then
            Dim dlg As New dlgUpdateStatus(AddressOf FTP_FAIL)
            Me.Invoke(dlg, text)
        Else
            buglog.Items.Insert(0, DateTime.Now & ":   " & text)

        End If

    End Sub

    Public Sub w_Writevariableid()
        If connection_ok Then
            If IsNumeric(itext(1).Text) And Not (itext(1).Text = String.Empty) Then

                File.Delete(dirlocation & "config.txt")
                Dim myWriter2 As New StreamWriter(dirlocation & "config.txt", True)
                myWriter2.Write(itext(1).Text)
                myWriter2.Close()
                V_ID_BUG = False
            Else
                buglog.Items.Insert(0, DateTime.Now & ":   Fehler: Sie müssen eine gültige VariableID im linken Textfeld angeben.")
                V_ID_BUG = True
            End If

        End If

    End Sub

    Public Sub s_Writevariableid()
        If connection_ok Then
            If IsNumeric(stext(1).Text) And Not (stext(1).Text = String.Empty) Then

                File.Delete(dirlocation & "config2.txt")
                Dim myWriter2 As New StreamWriter(dirlocation & "config2.txt", True)
                myWriter2.Write(stext(1).Text)
                myWriter2.Close()
                V_ID_BUG2 = False
            Else
                buglog.Items.Insert(0, DateTime.Now & ":   Fehler: Sie müssen eine gültige VariableID im rechten Textfeld angeben.")
                V_ID_BUG2 = True
            End If

        End If

    End Sub

    Public Sub buttonenabled()
        Button4.Enabled = False
        Button5.Enabled = False
        Button6.Enabled = False
        Button8.Enabled = False
        Button9.Enabled = False
        Button10.Enabled = False
    End Sub


    Private Sub spsips_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles spsips.Click
        itext(1).Enabled = True
        stext(1).Enabled = False
    End Sub


    Private Sub ipssps_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles ipssps.Click
        itext(1).Enabled = False
        stext(1).Enabled = True
    End Sub

    Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button7.Click
        File.Delete("output_sps.txt")
        File.Delete("output_ips.txt")
        connection_ok = False
        V_ID_BUG = False
        V_ID_BUG2 = False
        transfer_activ = False
        transfer_activ2 = False
        req = False
        req2 = False
        flag = False
        flag2 = False

        Button5_Click(Button5, EventArgs.Empty)
        Button8_Click(Button5, EventArgs.Empty)
        buttonenabled()
        spsip.Text = ""
        itext(1).Text = ""
        stext(1).Text = ""
        Label3.Text = "  "
        Label11.Text = "  "
        direstatus.Text = "  "
        ipstatus.Text = "  "
        buglog.Items.Clear()
        datalog.Items.Clear()
        datalog2.Items.Clear()

    End Sub

    Private Sub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button10.Click
        Process.Start("output_ips.txt")
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
        Process.Start("output_sps.txt")
    End Sub

    Public Sub copytoips()
        Try
            System.IO.Directory.Delete(ipdirect.Text & "\hka_phoenix", True)

        Catch
        End Try
        Try
            My.Computer.FileSystem.CopyDirectory("hka_phoenix", ipdirect.Text & "\hka_phoenix", False)

        Catch
        End Try


    End Sub

End Class
```