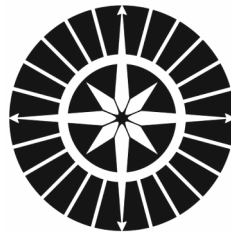


Herausgeber: Institut für Fernunterricht  
Rolf Fr. Weber Verlags-GmbH  
Verfasser: Prof. Dr. Heinrich Lepers  
Zulassungs-Nummer: 7220810



# Fernlehrgang

## SPS-Technik und IEC-Programmierung

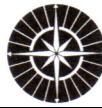
Programmieren von Automatisierungssystemen  
nach IEC 61131-3 mit CoDeSys und STEP 7

Staatlich geprüft



## Probelektion

9 Lehrbriefe, 3 DVDs, 1 USB-Stick



## Probelektion

des Fernlehrgangs

### „SPS-Technik und IEC-Programmierung“.

Diese Probelektion soll Ihnen zeigen, wie der Lehrgang, für den Sie sich interessieren,

- aufgebaut ist
- wie das Lehrmaterial gestaltet ist
- praktisch durchgeführt wird
- welches Ziel Sie erreichen können.

Die Probelektion soll aber auch verhindern, dass Sie sich dem falschen Studienziel zuwenden.

Die Probelektion enthält eine Auswahl von Seiten aus den Original-Lehrbriefen als Lehrstoffprobe. Sie können dadurch einen ersten Eindruck von der Aufbereitung des Lernstoffes gewinnen und die Lehrmethode kennenlernen.

Es ist möglich, dass diese Lehrstoffproben Ihnen noch nicht als ausreichend erscheinen. Es ist ebenso möglich, dass gerade die hier vorgestellten Themen nicht Ihr besonderes Interesse finden. Es ist auch nicht anzunehmen, dass Sie gleich alles verstehen oder gar, dass Sie aus diesen wenigen Seiten schon etwas Wesentliches lernen können. Das kann und soll nicht der Zweck der Probelektion sein.

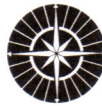
Wenn Sie sich auch nach Durchsicht der Lektion noch nicht entscheiden können, so empfehlen wir Ihnen ein

## Teststudium

Dieses ist völlig ohne Risiko für Sie, denn

- der erste Studienmonat ist ein Probemonat.
- Sie erhalten die ersten drei Lehrbriefe und können sich ein genaues Bild machen.
- Sie entscheiden erst nach Ablauf des Probemonats, ob Sie weiterstudieren wollen.

Wenn Ihnen das Teststudium nicht gefällt oder Sie das Studium aus irgendeinem Grunde nicht fortsetzen möchten, schicken Sie uns einfach das Lehrmaterial zurück, und die Sache ist erledigt.



## **Die Voraussetzungen für die Teilnahme an diesem Lehrgang**

Der Lehrgang vermittelt die Kenntnisse und Fähigkeiten über Speicherprogrammierbare Steuerungen und die nach IEC definierten Programmiersprachen, die Sie als echte SPS-Fachkraft im Berufsleben brauchen. Für die Teilnahme an diesem Lehrgang müssen Sie keine besonderen Voraussetzungen erfüllen. Allerdings sollten Sie neben einer abgeschlossenen Schulausbildung mindestens eine technische Berufsausbildung absolvieren oder absolviert haben.

Was sollten Sie sonst noch mitbringen? Sie sollten mathematisch und naturwissenschaftlich interessiert sowie mit dem Umgang eines Computers und dem Betriebssystem Windows vertraut sein. Da es sich um einen Praxis-Lehrgang handelt, sollten Sie einen herkömmlichen PC mit DVD-Laufwerk, Drucker und Internet-Zugang besitzen.

## **Das Lehrgangsziel**

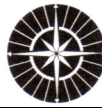
### **Ausbildung zum SPS-Spezialisten**

Der Kurs vermittelt Ihnen alle wichtigen Kenntnissen und Fähigkeiten über Speicherprogrammierbare Steuerungen und die nach IEC definierten Programmiersprachen für den erfolgreichen Einsatz im Beruf.

Sie erhalten einen umfassenden Einblick in die Grundlagen der Speicherprogrammierbaren Steuerungen sowie der Programmierung nach IEC-Norm 61131-3 und lernen die Basissprachen für Verknüpfungssteuerungen Anweisungsliste (AWL), Kontaktplan (KOP) und Funktionsplan (FUP) kennen.

Des Weiteren erfahren Sie alles über die modernen IEC-Programmiersprachen Funktionsbausteinsprache (FBS) und Strukturierter Text (ST) zur symbolischen Programmierung.

Abschließend lernen Sie die nach IEC-Norm definierte Ablaufsprache (AS) kennen, mit der Ablaufsteuerungen für bestimmte Prozesse – Chargen- und Batchprozesse – programmiert werden.



## **Die Praxis des Fernstudiums**

Sie können jederzeit mit Ihrem Fernstudium beginnen. Das Lehrmaterial besteht aus 9 Lehrbriefen und Datenträgern mit der benötigten Software CoDeSys und STEP 7 von Siemens für die zahlreichen Projektarbeiten. Diese Programmierwerkzeuge kommen auch in der Industrie hauptsächlich zum Einsatz.

Sie arbeiten die Lehrbriefe in Ruhe durch. Viele Beispiele erläutern den Text. Merksätze und Zusammenfassungen helfen Ihrem Gedächtnis. Die erworbenen Kenntnisse kontrollieren Sie anhand von eingestreuten Kontrollaufgaben. Die mitgelieferten Lösungen zeigen Ihnen, ob Sie mit Ihrem Studienerfolg zufrieden sein können.

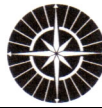
Am Schluss eines jeden Lehrbriefes wird eine Reihe von Aufgaben gestellt, die von Ihnen gelöst und eingesandt werden sollten. Ihre Lehrgangleiter korrigieren und bewerten diese Hausaufgaben und senden sie Ihnen innerhalb weniger Tage zurück. Sie geben Ihnen auch individuelle Hinweise, sofern dieses sich hier oder da als notwendig erweisen sollte.

Sie erhalten das gesamte Lehrmaterial in Quartalslieferungen jeweils für ein Studienvierteljahr im Voraus. Eine schnellere Lieferfolge kann vereinbart werden.

Die durchschnittliche Studiendauer beträgt 9 Monate. Als wöchentliche Studienzeit sind etwa 12 Stunden vorgesehen. Sie können jedoch auch schneller vorgehen oder sich mehr Zeit lassen. Sie sind bei Ihrer Zeiteinteilung an keine Fristen und Termine gebunden. Die Lösungen der Hausaufgaben können jederzeit – ohne zeitliche Begrenzung – eingesandt werden. Auch die Abschlussprüfung können Sie jederzeit ablegen. Die Betreuungszeit (fachliche Betreuung) endet jedoch mit der Abschlussprüfung.

## **Staatliche Anerkennung**

Der Lehrgang wurde von der Staatlichen Zentralstelle für Fernunterricht (ZFU) in Köln geprüft und unter der Nummer 7220810 zugelassen. Fernunterricht unterliegt in Deutschland einer strengen staatlichen Kontrolle. Die Zulassung gewährleistet, dass der Lehrstoff vollständig, fachlich einwandfrei und pädagogisch-didaktisch nach lernpsychologischen Erkenntnissen aufbereitet ist. Die Zentralstelle prüft auch die fachliche Qualifikation der Fernlehrer und die Inhalte des Unterrichtsvertrages zwischen Fernstudierenden und Fernschule. Fernunterricht verdient mit dieser Anerkennung Ihr Vertrauen.



## Lehrgangsinformationen

Alle Informationen zu diesem Fernlehrgang erhalten Sie mit unserem Studienführer. Diesen können Sie hier anfordern: <http://www.fernschule-weber.de/versand/infos.htm>. Sollten Sie weitere Fragen haben: Rufen Sie uns an! Telefon: 04487 / 263.

## Wie viel kostet der Lehrgang?

Die Lehrgangsgebühr finden Sie auf der Studienanmeldung zu unseren Fernlehrgängen, die Sie mit unserem Studienführer erhalten. Im Internet können Sie das Anmeldeformular hier herunterladen:

<http://www.fernschule-weber.de/public/Anmelden.pdf>.

In der Gebühr sind das gesamte Lehrmaterial, die fachliche und pädagogische Betreuung, die Korrektur und Bewertung der Aufgabenlösungen, die Gebühren für die Abschlussprüfung und das Abschlusszeugnis enthalten.

## Das Abschluss-Zeugnis

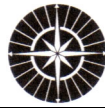
Sie erhalten das Fachlehrgangs-Abschlusszeugnis, wenn Sie die Hausaufgaben aus den Lehrbriefen gelöst und zur Korrektur eingesandt und die Lehrgangs-Abschlussprüfung bestanden haben. Die Abschlussprüfung ist eine Heimprüfung, die Sie zu Hause aufgrund von Prüfungsaufgaben anfertigen. Das Abschlusszeugnis zeigt Ihre Kenntnisse und beweist Ihre Fähigkeit zu selbständigem und zielstrebigem Arbeiten.

## Werden Fernschulzeugnisse anerkannt?

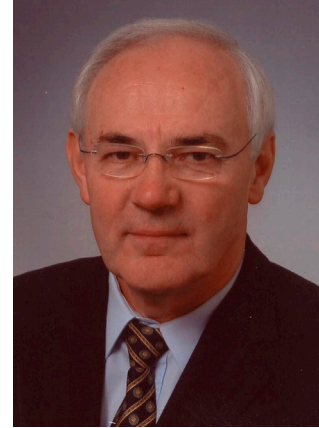
Fernunterricht ist die einzige Bildungsform in Deutschland mit „Verbraucherschutz“. Jeder Fernlehrgang muss von der Staatlichen Zentralstelle für Fernunterricht (ZFU) in Köln geprüft und zugelassen sein. Das Abschlusszeugnis der Fernschule WEBER ist der Nachweis der erfolgreichen Teilnahme an einem staatlich geprüften Fernlehrgang.

Abschlusszeugnisse der Fernschule WEBER werden in der Wirtschaft, Industrie und Öffentlichkeit als Abschluss eines bekannten privaten Fernlehrinstitutes anerkannt.

Personalchefs respektieren Fernstudierende im besonderen Maße. Wer sich aus eigener Initiative weiterbildet, wird auch immer ein zuverlässiger und strebsamer Mitarbeiter sein. Auch bei Beförderungen und Gehaltsverbesserungen haben Menschen, die sich in ihrer Freizeit weitergebildet haben, immer gute Aussichten.



## Ihr Lehrgangsteiter



Prof. Dr. Heinrich Lepers

Dr. Heinrich Lepers wurde 1941 geboren und war Professor an einer Fachhochschule im Fachbereich Elektrotechnik.

Seit über 30 Jahren ist Dr. Heinrich Lepers auf dem Gebiet der Steuerungs- und Regelungstechnik tätig. Er ist Verfasser zahlreicher Veröffentlichungen in einschlägigen Fachzeitschriften und als Buchautor aktiv. Des Weiteren arbeitete er mit anderen Experten ein Patent zum „Regelverfahren für Brennprozesse, insbesondere für die Zementherstellung im Drehrohren“ aus.

Dr. Lepers ist auf dem Gebiet der Steuerungs- und Regelungstechnik ein hochqualifizierter und anerkannter Fachmann. Dieser ausgewiesene Experte ist Ihr persönlicher Lehrgangsteiter beim Fernlehrgang „SPS-Technik und IEC-Programmierung“, der Ihnen sein Wissen verständlich darstellt und vermittelt. Dr. Lepers hat den Fernlehrgang selbst verfasst, betreut Sie als Teilnehmer, korrigiert Ihre Hausaufgaben und steht Ihnen bei allen Fragen zum Fachgebiet hilfreich zur Seite. Dies sind nicht hoch genug einzuschätzende Vorteile und für Sie die Gewähr für eine qualifizierte, hervorragende Ausbildung.

## Probelektion

**Inhalt:** Leseproben aus den Lehrbriefen 2, 4, 6 und 8

FERNSCHULE WEBER

---

„SPS-Technik und IEC-Programmierung“

Programmieren von Automatisierungssystemen nach IEC 61131-3 mit CoDeSys und STEP 7

© Institut für Fernunterricht – Postfach 21 61 – 26192 Großenkneten

Nachdruck, Vervielfältigung und Weitergabe nur mit Zustimmung des Verlages

### 3 Programmorganisationseinheiten nach IEC 61131-3

#### Inhalt dieses Kapitels

Es wird erklärt, dass ein SPS-Programm nach IEC 61131-3 Programmorganisationseinheiten (POEs) enthält und wie man diese definiert. POEs sind Funktionen (FC), Funktionsbausteine (FB) und Programme (P). In Bibliotheken liefert der Hersteller solche POEs; weitere POEs kann der Hersteller oder der Anwender erzeugen, indem er die in der IEC 61131-3 definierten Programmiersprachen verwendet. Die POEs dürfen nicht rekursiv verwendet werden, was bedeutet, dass eine POE nicht sich selbst verwenden darf.

#### Ziel dieses Kapitels

Sie kennen die Struktur eines SPS-Programms, das der IEC-Norm 61131-3 entspricht.

#### Wer sollte dieses Kapitel wie intensiv bearbeiten?

<b>Einstieg</b> <b>Ausbildungsziel</b>	Anfänger	Grundkenntnisse der SPS-Technik	S5- Programmierer
CoDeSys	2	2	2
STEP 7	2	2	2

#### Intensitäten:

0. Nicht lesen
1. Text durchsehen, kontrollieren, ob der Stoff bekannt ist
2. Text durcharbeiten und Aufgaben lösen
3. Text durcharbeiten, Aufgaben lösen, Beispiel-Projekte ansehen und Aufgaben-Projekte selbst auf dem PC realisieren
4. Text durcharbeiten, Aufgaben lösen, Beispiel- **und** Aufgaben-Projekte selbst auf dem PC realisieren

Die Erklärung der Tabelle finden Sie in der Studienanleitung.

## 3.1 Funktionen

### 3.1.1 Definition von Funktionen

Eine Funktion liefert ein Funktionsergebnis, das ausschließlich aus den Eingangswerten berechnet wird und unter dem Funktionsnamen zur Verfügung steht. Nach der Norm kann eine Funktion weitere Ausgangswerte berechnen, die als VAR\_OUTPUT oder VAR\_IN\_OUT definiert sind. Der Übersichtlichkeit wegen sollte man aber darauf verzichten und eine Funktion auf die Berechnung des Funktionswertes beschränken. Dieses Funktionsergebnis kann einem beliebigen elementaren oder abgeleiteten Datentyp zugeordnet werden. Damit kann das Ergebnis auch ein Feld oder eine Struktur sein.

Der Aufruf einer Funktion erfolgt durch Nennen des Funktionsnamens und Übergabe der Eingangswerte (Argumente). Eine Funktion kann auch als Argument oder als Operand verwendet werden oder andere Funktionen aufrufen.

### 3.1.2 Darstellung von Funktionen

Funktionen werden in grafischen Sprachen als Rechtecksymbole dargestellt. Der Name der Funktion wird oben in das Rechteck eingetragen. Der Signalfluss geht von links nach rechts. Damit müssen die Eingänge an die linke Begrenzungslinie gelegt werden. Das Funktionsergebnis wird rechts abgenommen. Beispiele für drei Funktionen in Funktionsbausteinsprache sind in der folgenden Abbildung gezeigt:

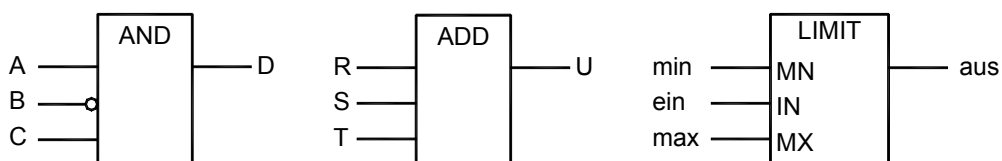


Abbildung 1: Beispiele für Funktionen in FBS

In der hier gezeigten Funktion AND wird eine logische UND-Verknüpfung zwischen den booleschen Variablen A, C und der Negation von B vorgenommen. Das Ergebnis wird der booleschen Variablen D zugewiesen.

Die Funktion ADD addiert die numerischen Variablen R, S und T und weist die Summe der numerischen Variablen U zu. In den Rechtecksymbolen sind hier die Eingangssignale nicht gekennzeichnet. Intern vergibt die Funktion diesen Eingängen von oben nach unten die Namen IN1, IN2, ..., wenn es sich - wie hier - um mehrere Eingänge handelt und IN, wenn nur ein Eingang existiert. Eingangssignale müssen innerhalb einer Funktion als VAR\_INPUT (oder als VAR\_IN\_OUT) deklariert sein.

Funktionen können aber auch bezeichnete Eingänge haben, wie das dritte Beispiel LIMIT zeigt. Diese Möglichkeit wird vor allem dann genutzt, wenn die Verarbeitung der verschiedenen Eingangssignale unterschiedlich ist. In dem Beispiel LIMIT wird das Eingangssignal „ein“ an der Eingangsvariablen IN nach unten auf den durch „min“ vorgegebenen Minimum-Wert MN und nach oben auf den durch „max“ vorgegebenen Maximum-Wert MX begrenzt. Der begrenzte Funktionswert wird in die Variable „aus“ übertragen. Bei abgeleiteten Funktionen werden die Eingänge mit den in VAR\_INPUT festgelegten Namen bezeichnet und in der dort angegebenen Reihenfolge angezeigt.

Den Eingängen einer Funktion können konstante Werte (Literele), Variable oder Funktionswerte übergeben werden. Wenn die Eingänge nicht belegt oder zugewiesen werden, nehmen die entsprechenden internen Variablen den in der Funktion festgelegten Anfangswert an. Als Eingangsgrößen in eine Funktion dürfen keine Ausdrücke verwendet werden, mit denen der zu übergebende Wert zunächst berechnet werden muss.

Das in der Funktion berechnete Funktionsergebnis wird mit dem Zuweisungszeichen „ := “ dem Funktionsnamen zugewiesen. Dieser Funktionswert muss an eine Variable weitergegeben werden. Da ich Ihnen empfehle, Ausgangsvariable vom Typ VAR\_OUTPUT oder VAR\_IN\_OUT nicht zu verwenden, mache ich zu deren Nutzung hier keine Angaben. Wenn man mehrere Ergebnisse berechnen möchte, sollte man dazu Funktionsbausteine verwenden oder den Funktionswert als einen Datentyp ARRAY oder STRUCT definieren.

Die im obigen Bild gezeigten Beispiele würden in der Sprache „strukturierter Text“ wie folgt formuliert. Da bei der Übergabe keine Ausdrücke erlaubt sind, muss die Negation von B vorher vorgenommen werden und z. B. der Variablen NB zugewiesen werden.

```
NB := NOT B ;  
D := AND(A, NB, C) ;  
U := ADD(R,S,T) ;  
aus := LIMIT(min, ein, max) ; (* nicht formal *)  
aus := LIMIT(IN:=ein, MN:=min, MX:=max) ; (* formal *)
```

#### Liste 13: Beispiele für Funktionen ST

Bei der nicht formalen Übergabe der aktuellen Eingangsparameter an die Funktion werden diese vollzählig in der festen Reihenfolge übergeben. Wenn hingegen die Übergabe formal durch Einzelzuweisungen erfolgt, ist die Reihenfolge der Eingangsparameter beliebig und sie muss nicht vollzählig sein. Dann würde für den nicht belegten Eingang dessen interne Variable mit dem in der Funktion festgelegten Anfangswert übernommen.



### Aufgabe 12

In der Sprache „Strukturierter Text“ wird die Funktion SEL wie folgt aufgerufen:

```
aus := SEL(G := TRUE, IN0 := ein0, IN1 := ein1); .
```

Zeichnen Sie diesen Funktionsaufruf in Funktionsbausteinsprache (FBS)!

Die Lösung finden Sie am Ende des Lehrbriefes.

### 3.1.3 Steuerung der Ausführung von Funktionen

Es ist möglich, die Berechnung in einer Funktion über ein boolesches (Freigabe-) Signal EN (Enable) ein- oder auszuschalten. Außerdem kann mit einem booleschen Ausgangssignal ENO (Enable Out) angezeigt werden, ob die Funktion ordnungsgemäß ausgeführt wurde. Innerhalb der Funktion muss demnach deklariert werden:

```
VAR_INPUT      EN  : BOOL := 1 ;      END_VAR  
VAR_OUTPUT     ENO : BOOL ;          END_VAR .
```

Diese Signale stehen im grafischen Symbol jeweils an oberster Stelle.

Durch die Zuweisung von 1 an den Eingang EN wird erreicht, dass dann, wenn dieser Eingang nicht belegt wird, die Funktion berechnet wird.

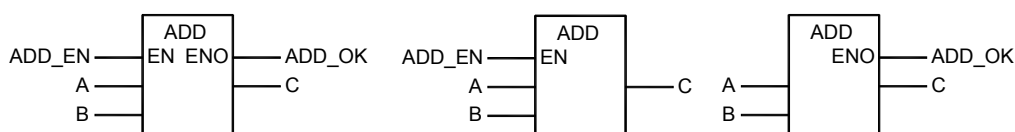


Abbildung 2: Verwendung von EN und ENO

Die Bedeutung von EN und ENO ist wie folgt:

Bei Funktionsaufruf ist EN = 0 :

Die Funktion wird nicht ausgeführt.  
ENO := 0

Bei Funktionsaufruf ist EN = 1 :

Die Funktion wird ausgeführt.  
ENO := 1, Berechnung ist OK.  
ENO := 0, Berechnung ist fehlerhaft.

Wenn man wissen muss, ob der Funktionswert vertrauenswürdig ist, muss man den Ausgang ENO abfragen. Dazu muss ENO einer Variablen zugewiesen werden, die grafisch an das Symbol wie im obigen Bild angebunden wird oder die wie im folgenden Beispiel in der Sprache „Strukturierter Text“ (ST) mit dem Symbol „=>“ zugewiesen wird.

```
C := ADD(EN:=ADD_EN, IN1:=A, IN2:=B, ENO =>ADD_OK) ;
```

Man kann je nach Bedarf auch nur eine dieser booleschen Größen EN oder ENO verwenden (vgl. Abbildung 2).

### 3.1.4 Deklaration von abgeleiteten Funktionen

Neben den elementaren Standardfunktionen, die in der Norm IEC 61131-3 definiert sind und vom Hersteller in einer Bibliothek mit der SPS geliefert werden, kann ein Hersteller oder Anwender weitere, so genannte „abgeleitete Funktionen“ definieren. Auch hier wird darauf verzichtet, die in der Norm vorgesehene Möglichkeit zu nutzen, neben dem Funktionswert weitere Ausgangswerte zu berechnen.

Eine abgeleitete Funktion hat die folgende Struktur:

```
FUNCTION Funktionsname : Datentyp für Funktionswert
    VAR_INPUT
        Namen der Eingangsgrößen : Datentypen ;
    END_VAR
    Falls erforderlich:
    VAR
        Namen der internen Variablen : Datentypen ;
    END_VAR

    (* Funktionsrumpf in einer der Sprachen KOP, FBS, AWL, ST *)

END_FUNCTION
```

Liste 14: Deklaration einer abgeleiteten Funktion

Es folgt ein einfaches Beispiel mit den bereits bekannten Funktionen ADD und LIMIT (vgl. Abbildung 1), mit denen die Summe aus R, S und T berechnet und anschließend das Ergebnis auf den Wertebereich zwischen 0.0 und 100.0 begrenzt wird.

1. Deklaration in der Textsprache ST

```
FUNCTION Begrenzte_Summe : REAL
```

```
(* Deklaration: *)
```

```
VAR_INPUT R, S, T : REAL ; END_VAR
```

```
(* Funktionsrumpf: *)
```

```
Begrenzte_Summe := LIMIT(0.0, ADD(R, S, T), 100.0) ;
```

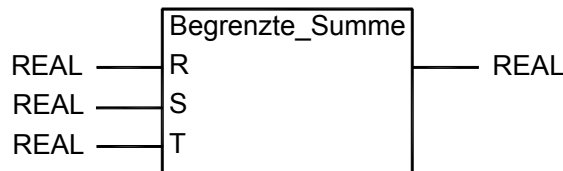
```
END_FUNCTION
```

Liste 15: Deklaration der Funktion „Begrenzte\_Summe“ in ST

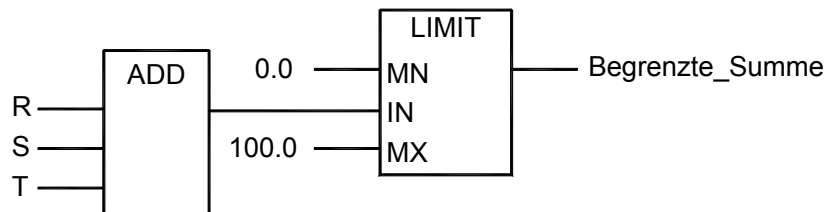
2. Deklaration in der grafischen Sprache FBS

```
FUNCTION
```

```
(* Festlegung der externen Schnittstelle: *)
```



```
(* Festlegung des Funktionsrumpfes: *)
```



```
END_FUNCTION
```

Abbildung 3: Deklaration der Funktion „Begrenzte\_Summe“ in FBS

Diese Funktion kann nun in anderen POEs verwendet werden. Bei einer Verwendung in ST müssen die angeschlossenen Variablen definiert und die Funktion aufgerufen werden.

```
Deklaration: VAR A, B, C, D : REAL ; END_VAR
```

```
Aufruf: D := Begrenzte_Summe(A, B, C) ;
```

In der grafischen Sprache FBS muss dieselbe Deklaration der Variablen vorgenommen werden. Der Aufruf der Funktion ist dann wie folgt:

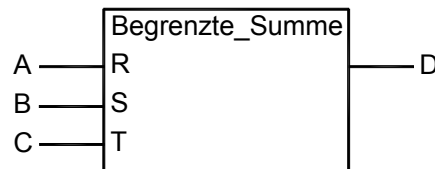


Abbildung 4: Aufruf der Funktion „Begrenzte\_Summe“ in FBS

### Aufgabe 13

In der abgeleiteten Funktion „Grenzsumme“ wird zu dem Eingangssignal X, das auf den Wertebereich zwischen „unten“ und „oben“ begrenzt wird, das Signal „Y“ addiert.

Schreiben Sie den Deklarationsteil und den Funktionsrumpf dieser abgeleiteten Funktion in strukturiertem Text!

Die Lösung finden Sie am Ende des Lehrbriefes.



### 3.1.5 Überladene Funktionen

Von überladenen Funktionen und anderen überladenen Berechnungen spricht man dann, wenn die Eingangssignale jedem beliebigen Datentyp einer Gruppe allgemeiner Datentypen angehören können. Eine Addition, die auf alle numerischen Daten vom Datentyp ANY\_NUM angewendet werden kann, kann also Eingangswerte verarbeiten, die z. B. alle INT oder alle REAL oder alle UINT oder alle LREAL usw. sind.

Bei überladenen Funktionen müssen die Eingänge alle gleichen elementaren oder abgeleiteten Datentyps sein. Falls notwendig, müssen die Datentypen durch Datentypumwandlung angepasst werden. Das Ergebnis einer überladenen Funktion entspricht dem Datentyp der Eingänge. Falls das Ergebnis einer Variablen mit einem anderen Datentyp zugewiesen werden soll, muss der Datentyp des Ergebnisses zuvor umgewandelt werden. Die Tabelle 19 zeigt Beispiele.

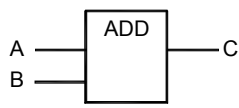
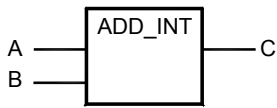
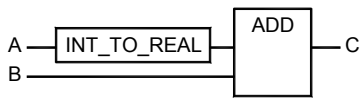
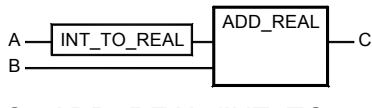
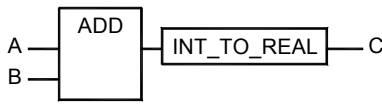
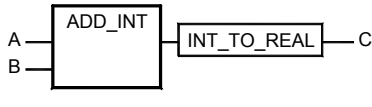
Typ-deklaration	Überladene Funktion	Funktion mit Typangabe
VAR A : INT; B : INT ; C : INT ; END_VAR	 $C := A + B;$	 $C := ADD\_INT(A, B);$
VAR A : INT; B : REAL; C : REAL; END_VAR	 $C := INT\_TO\_REAL(A) + B;$	 $C := ADD\_REAL (INT\_TO\_REAL(A), B);$
VAR A : INT; B : INT; C : REAL; END_VAR	 $C := INT\_TO\_REAL(A + B);$	 $C := INT\_TO\_REAL (ADD\_INT(A, B));$

Tabelle 19: Beispiele für explizite Typumwandlung

In dieser Tabelle sind Beispiele für überladene und typbezogene Funktionen gezeigt. Bei den typbezogenen Funktionen muss der Datentyp im Funktionsnamen hinter dem Unterstrich an den Funktionsnamen angehängt werden.

In der IEC 61131-3 sind Standardfunktionen definiert, die üblicherweise vom Hersteller in einer Bibliothek geliefert werden. Auf diese Funktionen wird in späteren Kapiteln und Lehrbriefen eingegangen.



Praxistipp:

Eine Funktion sollte nicht mehrere Ausgangswerte sondern ausschließlich den Funktionswert (ggf. als Feld oder Struktur) liefern. Es gibt also kein "VAR\_OUTPUT" und kein "VAR\_IN\_OUT"!

### Lösung Aufgabe 12

Die Funktion wird durch ein Rechtecksymbol dargestellt, wobei der Funktionsname oben im Rechteck angegeben wird. Die formalen Namen der Eingangsvariablen der Funktion stehen am linken Rand innerhalb des Rechtecks.

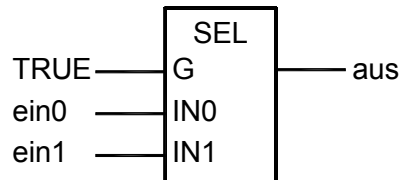


Abbildung 52: Funktionsaufruf in FBS

Der Ausgang trägt keinen speziellen Namen, da er mit dem Funktionsnamen übereinstimmt. Die aktuellen Parameter, die von außen an der linken Seite angeschlossen werden, werden beim Aufruf übergeben. Der Funktionswert wird nach der Bearbeitung der Funktion in die rechts angeschlossene Variable übertragen.

### Lösung Aufgabe 13

Die abgeleitete Funktion wird durch `FUNCTION ... END_FUNCTION` eingerahmt. Der Name der abgeleiteten Funktion erscheint hinter dem Schlüsselwort `FUNCTION` mit der Zuordnung zu einem Datentyp. Dann kommt der Deklarationsteil, in dem die Eingangsvariablen mit `VAR_INPUT ... END_VAR` und ggf. die internen Variablen mit `VAR ... END_VAR` deklariert werden.

```
FUNCTION Grenzsumme : REAL
    VAR_INPUT X, unten, oben, Y : REAL ; END_VAR
    Grenzsumme := Y + LIMIT(unten, X, oben) ;
END_FUNCTION
```

Liste 22: Funktion „Grenzsumme“ im ST

Im Funktions-Rumpf wird dem Funktionsnamen der Funktionswert zugewiesen. Hier werden beim Funktionsaufruf von `LIMIT` die aktuellen Eingangswerte „nicht formal“, d.h. in der richtigen Reihenfolge ohne explizite Zuweisung übergeben.

### 1.3 Modellierung des Prozesses mit einem Skript

Möchte man – wie hier beim Volumenmesser – ein Symbol rotieren lassen, so kann man z. B. bei einem Polygon unter „Absolute movement / Movement / Interior / Rotation“ eine Variable „Winkel“ angeben, die die Winkeldrehung in Grad angibt. Wenn diese Variable z. B. von 0 bis 90 schrittweise vergrößert wird, dreht sich das Symbol um 90 Grad um den bei Center definierten Punkt. In diesem Beispiel wird die Variable „Winkel“ in einem eigenständigen Programm „Modell“ berechnet, das unter der Registerkarte POU abgelegt wird.

Der Winkel wird in 10er-Schritten erhöht, bis der Winkel 90 Grad erreicht. Damit wird der Winkel wieder auf 0 Grad zurückgesetzt. Damit wird eine Viertelumdrehung erreicht.

Dieses Programm simuliert außerdem das Verhalten der Anlage, indem es dann, wenn der Winkel zwischen 80 und 90 Grad ist, für das Signal „GVL.Impuls“ den Wert TRUE und sonst den Wert FALSE ausgibt. Damit wird mit jeder Viertelumdrehung ein Impuls ausgelöst, was dem Durchfluss eines Liters entspricht.

Die Signale "Motor" und "Impuls" werden zwischen dem Prozess(modell) und der Steuerung ausgetauscht. Deshalb sind sie in der globalen Variablenliste "GVL" als globale Variable abgelegt worden.

Das Programm „Modell“ hat demnach das folgende Aussehen:

```
Modell
1  PROGRAM Modell
2  VAR_EXTERNAL
3      Motor:  BOOL;    (* Motor als globale Variable *)
4      Impuls:  BOOL;    (* Impuls als globale Variable *)
5  END_VAR
6  VAR
7      Winkel:  INT;    (* Winkelstellung des Volumenzählers *)
8  END_VAR
9
10 IF GVL.Motor THEN Winkel := Winkel + 10;
11     IF Winkel > 89 THEN Winkel := 0;           END_IF;
12     IF Winkel > 79 THEN GVL.Impuls := TRUE;
13         ELSE GVL.Impuls := FALSE;
14     END_IF;
15 ELSE Winkel := 0;
16     GVL.Impuls := FALSE;
17 END_IF;
```

Abbildung 6: Programm „Modell“ zur Anlagensimulation

Damit dieses Programm ausgeführt wird, muss es von einer Task gesteuert werden. Diese wird unter der Applikation mit „Application / Taskkonfiguration / Objekt einfügen / Task / Skript\_Task“ eingefügt. Es gibt nunmehr drei Tasks: MainTask für die SPS-Applikation, Skript\_Task für die Prozesssimulation (Modell) und VISU-Task für die Visualisierung. Alle Tasks sollten synchron ablaufen, das bedeutet, dass alle zyklisch mit der selben Zykluszeit von 200 ms ablaufen sollen:

**Skript\_Task [Device: SPS-Logik: Application: Taskkonfiguration]**

Konfiguration

Priorität ( 0..31 ): 1

Typ  
Zyklisch Intervall (z.B. t#200ms): t#200ms

Watchdog  
 Aktiv  
Zeit (z.B. t#200ms):  
Empfindlichkeit:

POUs

POU	Kommentar
Modell	

Aufruf hinzufügen  
Aufruf löschen  
POU öffnen  
Eingabehilfe...  
Nach oben

Abbildung 7: Konfiguration der Task „Skript\_Task“

Neben der Zykluszeit muss noch angegeben werden, dass diese Task das Programm „Modell“ steuern soll, indem „Modell“ bei „Aufruf hinzufügen“ eingetragen wird. Man sollte noch kontrollieren, ob die beiden anderen Tasks ebenfalls auf die Zykluszeit 200 ms eingestellt worden sind.

Das Projekt enthält nun als Programmorganisationseinheiten (POU) neben den allgemeinen Einträgen vor allem das Modell der Anlage als Programm (PRG) und die Soft-SPS (Device / CoDeSys SP Win V3).

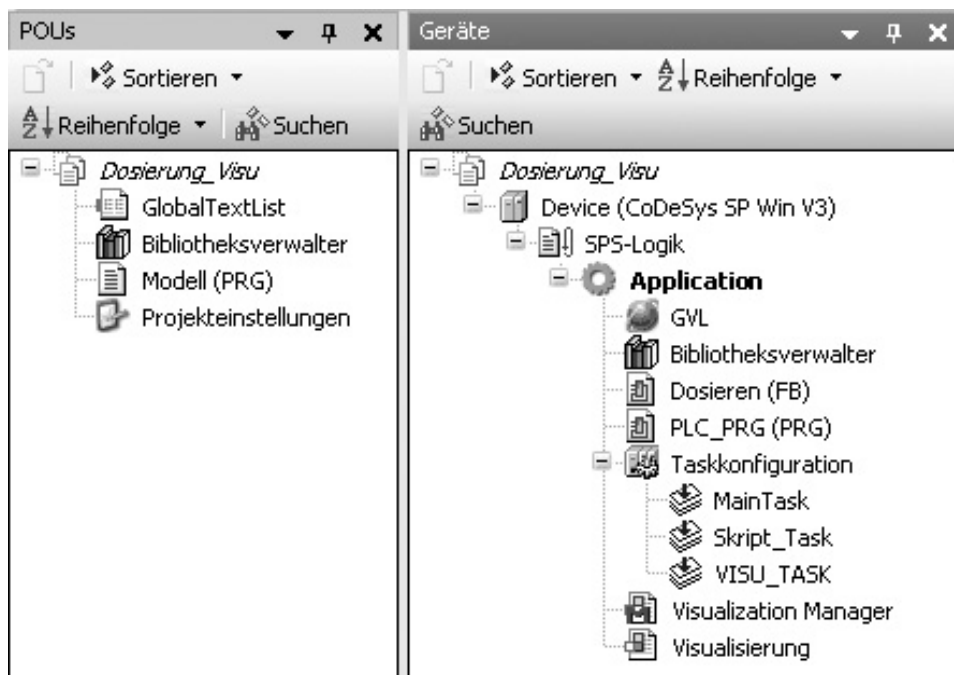


Abbildung 8: Struktur des fertigen Projektes „Dosierung“



#### Aufgabe 5:

Laden Sie das Projekt „Transport\_Visu4“ und speichern Sie es wieder unter „Transport\_Visu5“ ab.

Ausgehend von der Lösung in Aufgabe 4 soll nun zunächst der Transport durch Rotation der Speichen angezeigt werden.

Erstellen Sie analog zu dem Beispiel „Dosierung“ ein Modell des Prozesses, indem Sie zunächst nur den Drehwinkel „Winkel“ berechnen.

Modifizieren Sie die Eigenschaften der Polygone so, dass diese jeweils um den berechneten Winkel gedreht dargestellt werden.

Fügen Sie für die Simulation des Prozesses „Modell“ eine „Modell\_Task“ ein, die das Programm „Modell“ alle 200 ms aufruft. Kontrollieren Sie, ob bei den anderen Tasks ebenfalls eine Zykluszeit von 200 ms eingestellt ist.

Übersetzen Sie das Projekt, laden Sie es in die SPS und testen Sie, ob die Visualisierung erwartungsgemäß funktioniert.

Die Lösung finden Sie am Ende des Lehrbriefes.

Aufgabe 6:

Laden Sie das Projekt „Transport\_Visu5“ und speichern Sie es wieder unter „Transport\_Visu6“ ab.



Nun sollen Sie manuell herausfinden, welche Positionen das kleine Paket annehmen soll, wenn es auf das Band gesetzt wird und dort bis zum Ende des Bandes transportiert wird. Dazu fügen Sie zwei Rechtecke ein, in die Sie Werte für die horizontale Anfangs-Position „Hpos\_Anf“ und für die vertikale Anfangs-Position „Vpos\_Anf“ eingeben.

Ergänzen Sie nun das Skript „Modell“ so, dass Sie zunächst zusätzlich die boolesche Variable Button und die Integervariablen „Hpos“, „Vpos“, „Hpos\_Anf“ und „Vpos\_Anf“ deklarieren.

In der Visualisierung tragen Sie für den kleinen Button bei den Eigenschaften bei „Absolute movement / Movement /“ für X: „Modell.Hpos“ und für Y: „Modell.Vpos“ und bei „Eingabe / TAP / Variable“ „Modell.Button“ ein.

Im Programmcode soll nun dafür gesorgt werden, dass der kleine Button dann, wenn mit der Maustaste darauf gedrückt wird, die linke untere Ecke des Buttonsymbols auf die Koordinaten „Hpos\_Anf“ und „Vpos\_Anf“ gelegt wird. Wenn der Button auf „Vpos\_Anf“ steht und der Motor läuft (GVL.Motor = TRUE) soll die horizontale Position „Hpos“ mit jedem Zyklus um 10 Punkte erhöht werden. Sobald „Hpos“ einen Wert von mindestens 500 erreicht, soll der Button wieder in die Ursprungsposition (Vpos = Hpos = 0) gehen.

Die Lichtschranke „L1“ soll dann unterbrochen werden (FALSE), wenn der Button auf der vertikalen Position „Vpos\_Anf“ steht und sich zwischen den horizontalen Positionen „Hpos\_Anf“ und 100 bewegt. Entsprechendes gilt für „L2“ mit dem Unterschied, dass die Position der Lichtschranke „L2“ um 330 Punkte weiter rechts (höher) liegt.

Übersetzen Sie das Projekt, laden Sie es in die SPS und testen Sie, ob die Visualisierung erwartungsgemäß funktioniert.

Die Lösung finden Sie am Ende des Lehrbriefes.



#### Aufgabe 7:

Laden Sie das Projekt „Transport\_Visu6“ und speichern Sie es wieder unter „Transport\_Visu7“ ab.

Es gibt insgesamt drei verschieden lange Pakete. Die Länge unterscheidet sich jeweils um 50 Punkte, der vertikale Abstand zwischen den Paketen ist 30 Punkte.

Man kann die für den Transport erforderlichen Variablen „Button“, „Hpos“ und „Vpos“ in Felder ablegen, (Button[0..2], Hpos[0..2] und Vpos[0..2]), wobei mit dem Index 0 das kleine und mit Index 2 das große Paket adressiert wird. Hinweis: Wenn CoDeSys in einer virtuellen Maschine abläuft, ist es möglich, dass Sie keine eckigen Klammern „[„ und „]“ eingeben können. Sie sollten dann diese z. B. in Word schreiben und danach nach WinCC kopieren.

Die Pakete sind 50, 100 und 150 Punkte lang. Während bei einem Klick auf das kleine Paket dieses in die horizontale Anfangsposition „Hpos\_Anf“ gesetzt wird, müssen die anderen beiden Pakete um 50 bzw. 100 Punkte weiter links (kleinere Position) aufgesetzt werden. Die vertikalen Ausgangspositionen der Pakete unterscheiden sich jeweils um 30 Punkte. Daher muss die vertikale Anfangsposition „Vpos\_Anf“ bei den anderen beiden Paketen um 30 bzw. 60 Punkte größer sein.

Da die auf die Pakete bezogenen Daten in Feldern abgelegt sind, kann man das vorliegende Programm leicht durch eine Schleifenprogrammierung mit „FOR I := 0 TO 2 DO“ ... „END\_FOR;“ so modifizieren, dass nun alle Pakete auf das Band gelegt werden können. Dabei ist darauf zu achten, dass ein neues Paket erst dann auf das Band gelegt werden kann, wenn das davor aufgelegte Paket die Lichtschranke „L1“ bereits verlassen hat.

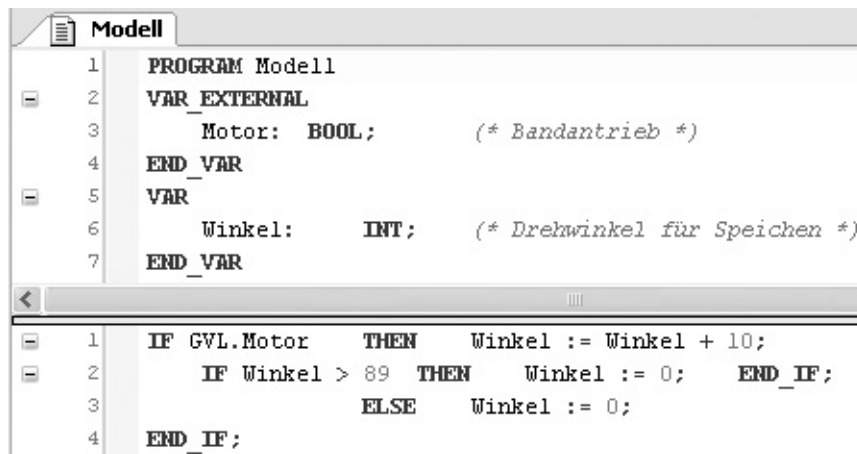
Modifizieren Sie das Programm „Modell“ nun so, dass nun alle Pakete aufgelegt werden können und die Lichtschranken durch alle Pakete betätigt werden. Tragen Sie in den Eigenschaften der Buttons bitte die notwendigen Variablen ein.

Übersetzen Sie das Projekt, laden Sie es in die SPS und testen Sie, ob die Visualisierung erwartungsgemäß funktioniert.

Die Lösung finden Sie am Ende des Lehrbriefes.

### Lösung Aufgabe 5:

Zunächst werden im Modell nur die Variablen und Anweisungen angegeben, die für die Drehung der Speichen erforderlich sind:



```
1 PROGRAM Modell
2 VAR_EXTERNAL
3     Motor: BOOL;      (* Bandantrieb *)
4 END_VAR
5 VAR
6     Winkel: INT;     (* Drehwinkel für Speichen *)
7 END_VAR
8
9 IF GVL.Motor THEN Winkel := Winkel + 10;
10     IF Winkel > 89 THEN Winkel := 0; END_IF;
11     ELSE Winkel := 0;
12 END_IF;
```

Abbildung 47: Modell für die Winkeldrehung

Die Speichen (Polygone) müssen nun noch mit der Variablen „Winkel“ in Rotation versetzt werden, indem die Eigenschaften entsprechend eingestellt werden. Dazu ist jeweils bei „Absolute movement / Movement / Interior rotation“ die Variable „Modell.Winkel“ einzutragen.

### Lösung Aufgabe 6:

In dem oberen Rechteck wird „Hpos\_Anf“ und im unteren „Vpos\_Anf“ eingegeben. Der kleine Button befindet sich gerade in der Lichtschranke „L2“.

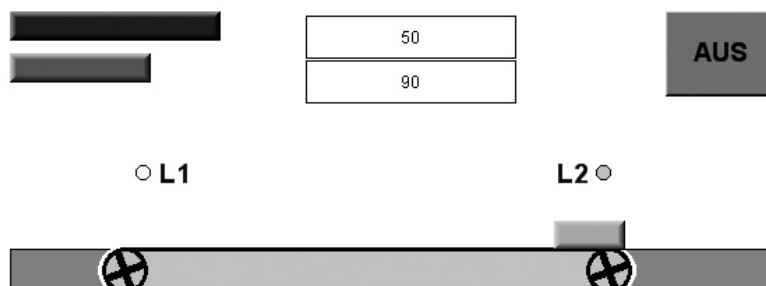


Abbildung 48: Bedienfeld im Test

Das Programm des Modells für die Testphase mit dem kleinen Button hat nun das folgende Aussehen:

```
Modell
1 PROGRAM Modell
2 VAR_EXTERNAL
3   Motor: BOOL; (* Bandantrieb *)
4 END_VAR
5 VAR
6   Winkel: INT; (* Drehwinkel für Speichen *)
7   Button: BOOL; (* Pakete wurden angeklickt *)
8   Hpos: INT; (* horizontale Positionen *)
9   Vpos: INT; (* vertikale Positionen *)
10  Hpos_Anf: INT; (* horizontale Anfangs-Positionen *)
11  Vpos_Anf: INT; (* vertikale Anfangs-Positionen *)
12 END_VAR
13
14 IF GVL.Motor THEN Winkel := Winkel + 10;
15                 IF Winkel > 89 THEN Winkel := 0; END_IF;
16                 ELSE Winkel := 0;
17 END_IF;
18
19 IF Button THEN Vpos := Vpos_Anf; Hpos := Hpos_Anf; END_IF;
20 IF Vpos = Vpos_Anf AND GVL.Motor THEN Hpos := Hpos + 10; END_IF;
21 IF Hpos > 500 THEN Vpos := 0; Hpos := 0; END_IF;
22
23 L1 := TRUE; (* Lichtschranke offen *)
24 L2 := TRUE;
25 IF Vpos = Vpos_Anf AND (Hpos >= Hpos_Anf AND Hpos <= 100) THEN L1 := FALSE; END_IF;
26 IF Vpos = Vpos_Anf AND (Hpos >= Hpos_Anf+330 AND Hpos <= 430) THEN L2 := FALSE; END_IF;
```

Abbildung 49: Programmcode für das Modell in der Testphase

### Lösung Aufgabe 7:

In der Visualisierung wurde in der letzten Aufgabe für den kleinen Button bei den Eigenschaften bei „Absolute movement / Movement“ für X: „Modell.Hpos“ und für Y: „Modell.Vpos“ und bei „Eingabe / TAP / Variable“ „Modell.Button“ eingetragen. Nun müssen Sie für alle drei Buttons die indizierten Variablen „Modell.Hpos[I]“, „Modell.Vpos[I]“ und „Modell.Button[I]“ eintragen, wobei für I = 0 für den kleinen, I = 1 für den mittleren und I = 2 für den langen Button genommen werden muss.

### 1.1.3 Die Betriebsarten des Reglers

Dem Signalflussplan des Regelkreises kann man entnehmen, dass der Regler den Soll-Istwertvergleich durchführt, indem er vom Sollwert  $W$  den Istwert  $X$  subtrahiert (oder umgekehrt). Bei einem P-Regler wird diese Regelabweichung mit dem Übertragungsfaktor  $K_R$  des Reglers multipliziert und so in die Stellgröße  $Y$  umgerechnet.

Die Wirkungsweise des Reglers legt fest, ob bei Vergrößerung der Regelgröße die Stellgröße auch größer oder kleiner wird. Im ersten Fall spricht man von einem positiven und im zweiten von einem negativen Übertragungsverhalten. Mit dem Vorzeichen "neg / pos" wird diese Wirkungsweise festgelegt.

Jeder Regler enthält einen "Hand/Automatik-Schalter", mit dem der Regelkreis geschlossen werden kann. Bei einer Inbetriebnahme des Reglers wird dieser zunächst in der Betriebsart "Hand" (H) betrieben, wobei der Bediener die Regelgröße mit der Handstellgröße  $Y_h$  beeinflusst. Erst wenn die Regelgröße sich in der Nähe des Sollwertes befindet, wird der Regler in die Betriebsart "Automatik" (A) gestellt und damit die vom Regler ermittelte Stellgröße auf das Stellorgan gegeben.

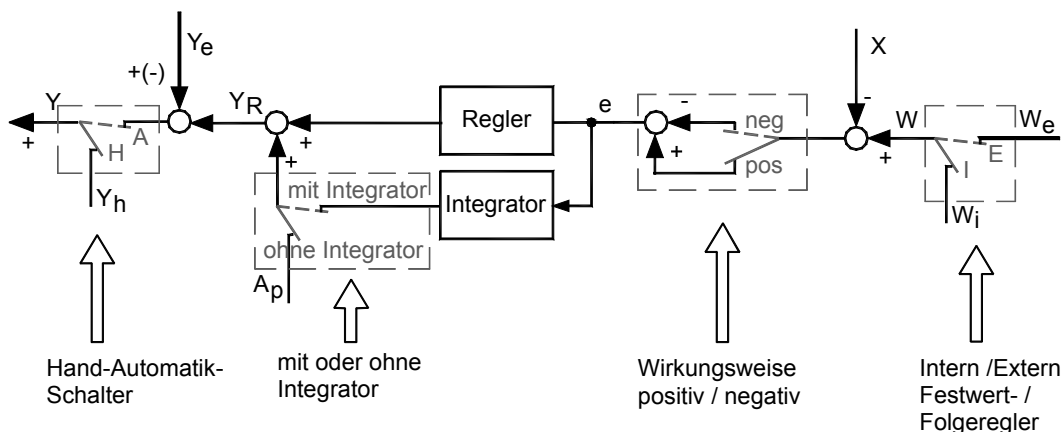


Abbildung 9: Betriebsarten des Reglers

Aufgabe einer Regelung kann es sein, trotz vorhandener Störungen die Regelgröße auf einem festen Sollwert zu halten (Festwertregelung) oder die Regelgröße an sich laufend verändernde Sollwerte anzupassen (Folgeregelung). Bei einer Festwertregelung wird der weitgehend konstante Sollwert intern ( $W_i$ ) vorgegeben. Bei einer Folgeregelung wird der Sollwert durch ein externes Signal ( $W_e$ ), das z. B. von einem Führungsregler stammt, vorgegeben.

Wenn im stationären Zustand keine bleibende Regelabweichung akzeptiert werden kann, muss diese dadurch beseitigt werden, dass mit einem Integrator der Arbeitspunkt des Reglers so verschoben wird, dass die Regelabweichung zu Null wird. Oft wird die Stellung des entsprechenden Schalters durch den Zahlenwert für die Nachstellzeit (vgl. später im Abschnitt 1.2.2) vorgegeben. Dabei wird mit  $T_n = 0$  s der Schalter in die Stellung "ohne Integrator" gelegt und damit der Integrator abgeschaltet. Dann muss allerdings der Arbeitspunkt eingestellt werden.

## 1.2 Dynamisches Verhalten von Reglern

### 1.2.1 Der Abtastvorgang

Im Gegensatz zu analogen Geräten, die die Signale kontinuierlich und parallel erfassen und verarbeiten, kann in einem digitalen Rechnersystem die Signalerfassung und Verarbeitung nur zyklisch und innerhalb eines Zyklus nacheinander erfolgen. Diese Arbeitsweise ist dadurch gekennzeichnet, dass zu Beginn eines Zyklus die Signalwerte erfasst werden. Anschließend werden z. B. die Stellgrößen berechnet und am Ende des Zyklus ausgegeben.

Das folgende Bild zeigt die Anbindung eines digitalen Reglers an einen Prozess:

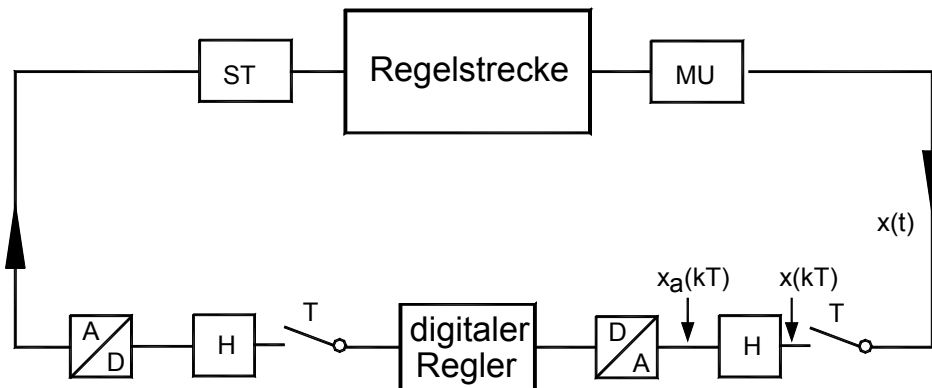


Abbildung 10: Abtastvorgang im Regelkreis

Durch das zyklische Abtasten der von einem Messgerät (Messumformer MU) gelieferten Messsignale entsteht aus dem kontinuierlichen Signal  $x(t)$  eine Folge von Abtastwerten  $x(kT)$ , die den Signalwert zu den äquidistanten Zeitpunkten  $kT$  angeben. Die Abtastzeitpunkte  $kT$  ergeben sich aus der konstanten Abtastzeit  $T$  und der ganzzahligen Zählvariablen  $k$ .

In einem Halteglied H werden die abgetasteten Signalwerte bis zum nächsten Abtastzeitpunkt konstant gehalten und anschließend vom Analogwert in den entsprechenden Digitalwert umgewandelt (D/A). Nach der Berechnung der Stellgröße und der Übergabe an das Halteglied (H) wird diese digitale Stellgröße in den entsprechenden Analogwert umgewandelt (A/D). Dieser Analogwert wird auf das Stellorgan (ST) gegeben und beeinflusst so die Regelstrecke.

Der Abtastvorgang soll hier etwas ausführlicher betrachtet werden:

Entsprechend der nächsten Abbildung ergibt sich aus der kontinuierlichen Regelgröße  $x(t)$  durch die Abtastung ein treppenförmiges Signal  $x_a(kT)$ .

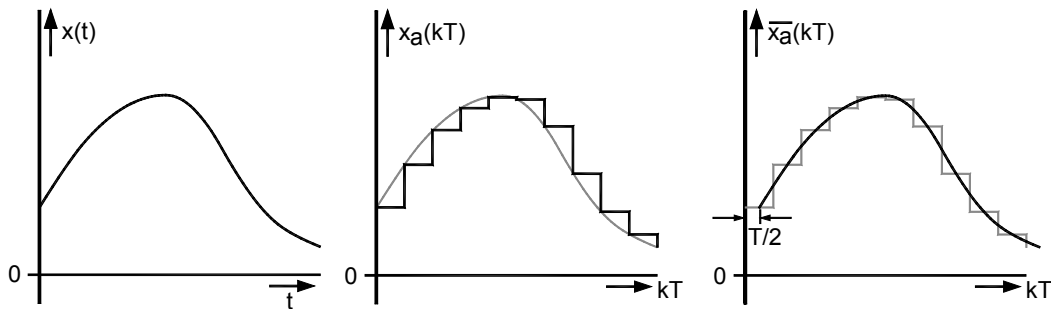


Abbildung 11: Abgetastete Regelgröße

Wenn man den treppenförmigen Verlauf des Signals glättet, entsteht daraus ein Signal, das in etwa dem ursprünglichen Signal entspricht, allerdings um eine halbe Abtastzeit verspätet ist. Durch den Abtastvorgang entsteht also eine Totzeit in der Größe einer halben Abtastzeit  $T_t = 0,5 T$ .

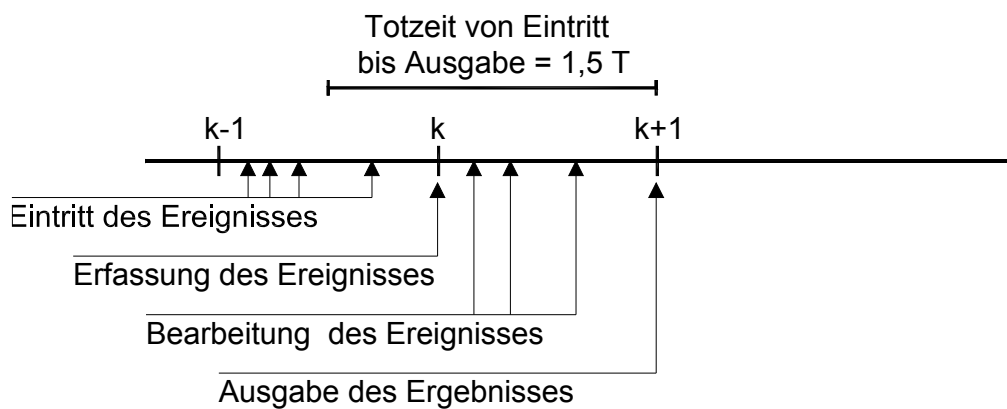


Abbildung 12: Reihenfolge der Verarbeitung des abgetasteten Signals

Diese Tatsache ergibt sich aus der obigen Abbildung, weil im Mittel eine halbe Abtastzeit vergangen ist, bis nach dem Eintritt eines Ereignisses dieses im nächsten Abtastzeitpunkt  $kT$  erfasst wird.

Danach beginnt die Verarbeitung des Signals im Regelalgorithmus. Das Ergebnis wird erst im nächsten bevorstehenden Zeitpunkt  $(k+1)T$  ausgegeben.

Somit ergibt sich also insgesamt von dem Zeitpunkt des Eintrittes des Ereignisses bis zur Ausgabe der Reaktion des digitalen Reglers eine Totzeit von dem 1,5-fachen der Abtastzeit. Da jede Totzeit für einen Regelkreis destabilisierend wirkt, muss aus Stabilitätsgründen darauf geachtet werden, dass die Abtast- oder Zykluszeit  $T$  nicht zu groß wird.



**Merke:**

**Abtastvorgang:**

**Durch das Abtasten eines kontinuierlichen Prozesssignals wird das Signal nur zu den Abtastzeitpunkten erfasst. Das aus dem Halteglied kommende und gemittelte Signal ist gegenüber dem ursprünglichen Signal um die halbe Abtastzeit zeitversetzt. Das führt zu einer entsprechenden Totzeit.**

**Totzeiten sind Zeiten, in denen ein System auf eine Änderung am Eingang nicht reagiert (sich tot stellt). Dadurch kommt es leicht zu Schwingungen im Regelkreis oder sogar zur Instabilität!**

## 1.2.2 Differenzgleichungen des P-, I- und PI-Reglers

**P-Regler:**

Nach der Erfassung der Regelabweichung ( $e_k = w_k - x_k$  bzw.  $e_k = x_k - w_k$ ) zum Zeitpunkt  $kT$  wird beim P-Regler die Veränderung der Stellgröße  $y_k$  aus der Multiplikation vom Übertragungsfaktor  $K_R$  mit dieser Regelabweichung berechnet:

$$y_k = K_R * e_k .$$

Den Verlauf der Regelabweichung  $e_k$  und der für  $K_R = 1,5$  ermittelten Stellgröße  $y_k$  zeigt die nächste Abbildung.

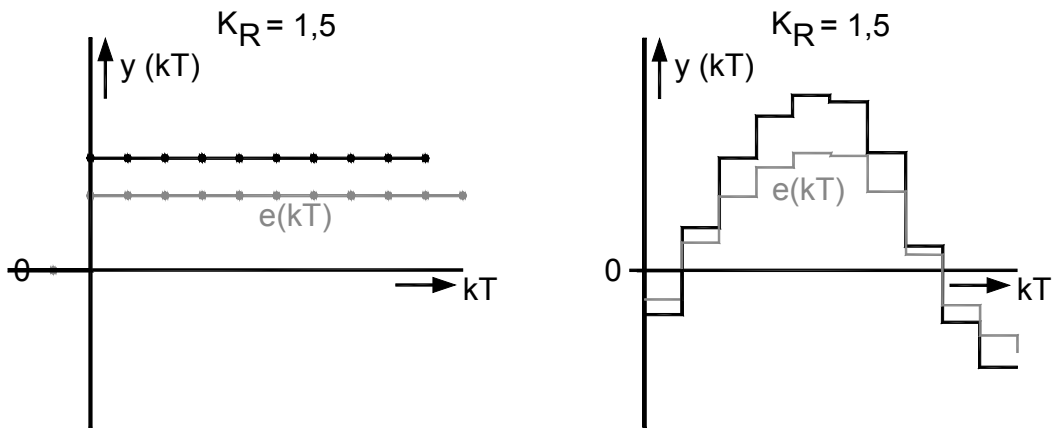


Abbildung 13: Regelabweichung und Stellgröße beim P-Regler

Kurz nach dem Erfassen der Signalwerte zum Zeitpunkt  $kT$  wird der Algorithmus berechnet und damit der Ausgangswert  $y_k$  ermittelt. Er wird allerdings erst beim nächsten Zyklus ausgegeben.

### I-Regler:

Beim I-Regler wird die Regelabweichung aufintegriert (die Fläche unter der Funktion wird berechnet) und mit dem Integrierbeiwert  $K_I$  multipliziert.

Die Fläche unter der Signalkurve  $e_k$  verändert sich in jedem Intervall um  $K_I * e_{k-1} * T$ , womit sich die neue Stellgröße  $y_k$  aus der alten Stellgröße  $y_{k-1}$  und im vorhergehenden Zyklus veränderten Fläche ergibt:

$$y_k = y_{k-1} + K_I * T * e_{k-1} .$$

Für  $K_I = 0,2 / T$  wird aus der vorgegebenen Regelabweichung  $e_k$  der folgende Verlauf der Stellgröße  $y_k$ , wenn zum Zeitpunkt  $t = 0$  das Ausgangssignal des I-Reglers  $y_0 = 0$  ist.

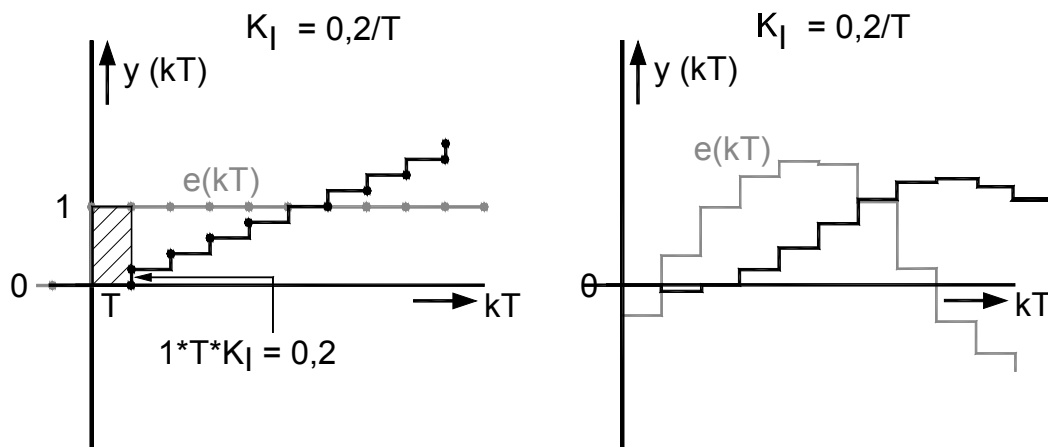


Abbildung 14: Regelabweichung und Stellgröße beim I-Regler

Im Intervall zwischen den Abtastzeitpunkten  $k = 0$  und  $k = 1$  wird der Signalwert zum Zeitpunkt  $kT = 0$  s mit der Zykluszeit  $T$  multipliziert und liefert so die Fläche unter der Kurve des Signals im Intervall  $k = 0$  bis  $k=1$ . Diese Fläche muss nun noch mit dem Integrierbeiwert  $K_I$  multipliziert werden.

Hier ergibt sich für einen Sprung der Regelabweichung von 0 auf den Wert 1 die folgende Rechnung:

$$e_{k-1} * T * K_I = 1 * T * 0,2 / T = 0,2 .$$

Dieses Ergebnis wird zum Zeitpunkt  $k=1$  ausgegeben.

Am Verlauf der Stellgröße im rechten Bild kann man sehen, dass der Ausgang so lange größer wird, wie die Regelabweichung positiv ist. Erst wenn die Regelabweichung negativ wird, nimmt die Stellgröße des I-Reglers wieder ab. Außerdem wird der Ausgang unverändert bleiben, wenn die Regelabweichung gleich Null ist.

**PI-Regler:**

Beim PI-Regler wird die Regelabweichung sowohl proportional als auch integrierend übertragen. Beide Teilsysteme arbeiten parallel, so dass sich die Stellgröße aus der Summe der beiden Einzelsysteme ergibt:

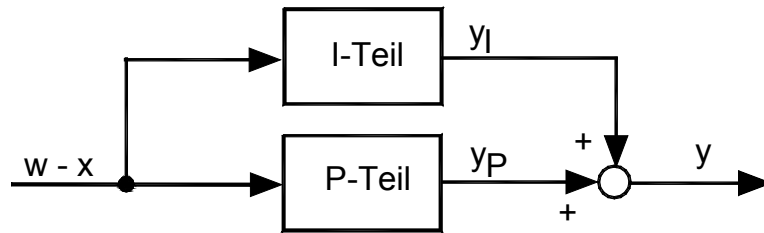


Abbildung 15: Struktur eines PI-Reglers

$$y_{P\ k} = K_R \cdot e_k$$

$$y_{I\ k} = y_{I\ k-1} + K_I \cdot T \cdot e_{k-1}$$

$$y_k = y_{P\ k} + y_{I\ k}$$

Für die Parameter  $K_R = 1,5$  und  $K_I = 0,2 / T$  ergibt sich bei den vorgegebenen Regelabweichungen die folgende Stellgröße, die sich aus der Addition der beiden zuvor gezeigten Stellgrößen berechnen.

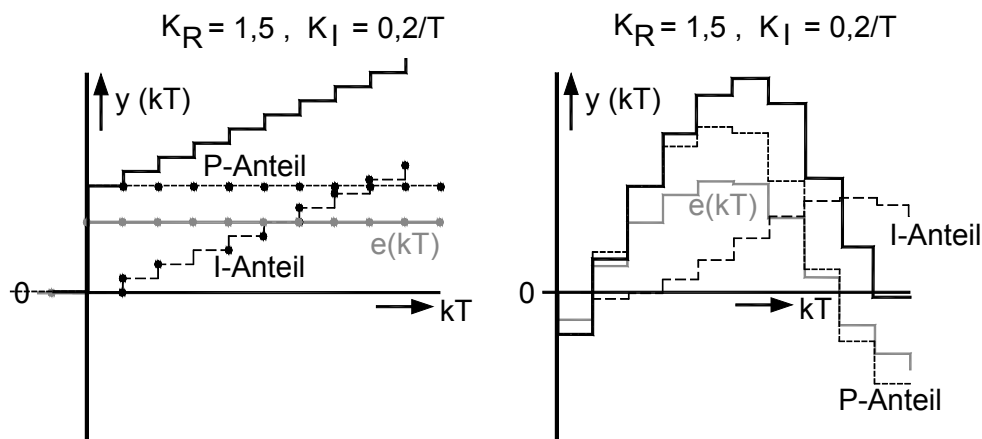


Abbildung 16: Regelabweichung und Stellgröße beim PI-Regler

Beim PI-Regler werden also die beiden Ergebnisse  $y_{P\ k}$  und  $y_{I\ k}$  addiert und ausgegeben.

Der Integrierbeiwert  $K_I$  weist eine Dimension auf, die sich ergibt aus dem Quotienten der Einheit der Stellgröße dividiert durch die Einheit der Regelabweichung und zusätzlich dividiert durch eine Zeit. Man kann den Übertragungsfaktor  $K_R$ , der die Einheit der Stellgröße dividiert durch die Einheit der Regelabweichung hat, aus dem Integrierbeiwert  $K_I$  ausklammern. Dann erhält man:

$$K_I = K_R / T_n .$$

Mit  $T_n$  wird die sogenannte Nachstellzeit angegeben, die als Parameter an einem PI-Regler eingestellt werden kann.



**Merke:**

**P-Regler:**

**Die Veränderung der Stellgröße  $y$  ist jederzeit proportional zur Regelabweichung  $e$ . Er reagiert auf eine Änderung der Regelabweichung sehr schnell.**

**I-Regler:**

**Die Stellgröße  $y$  ist nur dann konstant, wenn die Regelabweichung  $e = 0$  ist. Dadurch gibt es keine stationäre Regelabweichung.**

**PI-Regler:**

**Er vereinigt beide Eigenschaften vom P- und vom I-Regler. Er reagiert also schnell und lässt keine stationäre Regelabweichung zu.**

**Aufgabe 3:**

Für den PI-Regler wurden folgende Differenzgleichungen abgeleitet. Dabei sind  $e$  die Regelabweichung ( $x - w$ ) bzw. ( $w - x$ ) und  $y$  die Stellgröße:



$$y_{Pk} = K_R * e_k$$

$$y_{Ik} = y_{I\ k-1} + K_I * T * e_{k-1}$$

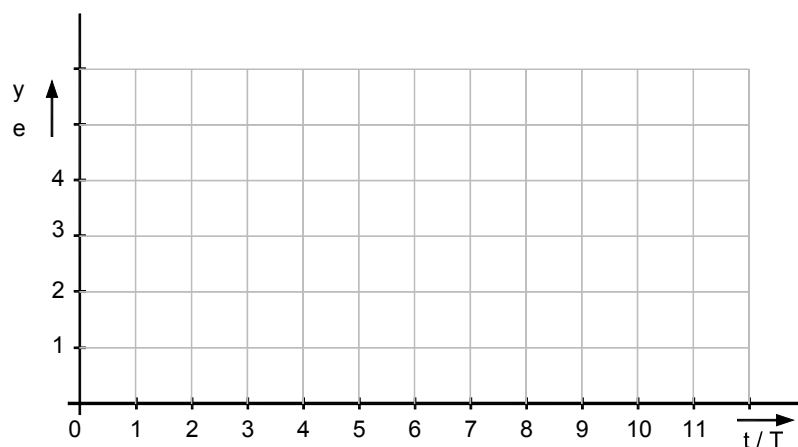
$$y_k = y_{Pk} + y_{Ik}$$

3.1: Bestimmen Sie für  $K_R = 2$ ,  $K_I = 5 / s$  und  $T = 0,1 s$  die in der Tabelle vorgegebenen Werte:

k	$e_k$	$y_{Pk}$	$y_{Ik}$	$y_k$
-1	0,0	0,0	0,0	0,0
0	1,0			
1	1,0			
2	1,0			
3	1,0			
4	1,0			

3.2: Wie groß ist bei den angegebenen Parametern die Nachstellzeit  $T_n$ ?

3.3: Zeichnen Sie die Punkte für  $e$  und  $y$  in das nachfolgende Diagramm und markieren Sie die Reglerparameter  $K_R$  und  $T_n$ !



Die Lösung finden Sie am Ende des Lehrbriefes.

**Lösung Aufgabe 3:**

3.1:

k	$e_k$	$y_{Pk}$	$y_{Ik}$	$y_k$
-1	0,0	0,0	0,0	0,0
0	1,0	2,0	0,0	2,0
1	1,0	2,0	0,5	2,5
2	1,0	2,0	1,0	3,0
3	1,0	2,0	1,5	3,5
4	1,0	2,0	2,0	4,0

3.2:  $K_I = K_R / T_n \rightarrow T_n = K_R / K_I = 2 / (5 / s) = 0,4 \text{ s}$ .

In der Tabelle kann man sehen, dass der Proportionalanteil sofort einen Sprung um 2 macht. Der Integrator benötigt 4 Abtastzeiten, um dieselbe Änderung der Stellgröße zu produzieren. Das entspricht genau  $0,1 \text{ s} * 4 = 0,4 \text{ s}$ , was mit der Nachstellzeit  $T_n$  übereinstimmt.

3.3:

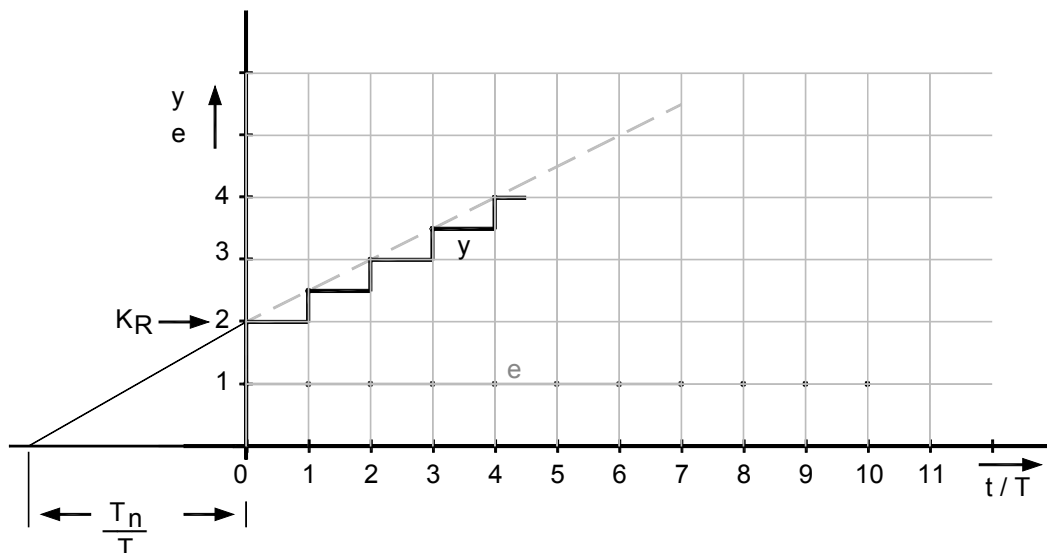


Abbildung 103: Übergangsfunktion des PI-Reglers

### 7.1.2 Visualisierung des Reglers „Mein\_2Pkt\_Regler“

Zunächst wird das Projekt „2PktRegl“ geladen und mit dem Projektnamen „2PktRgVs“ abgespeichert. Zusätzlich öffnen wir das Projekt „PIDT1Vis“ aus dem Ordner „C:/STEP7\_Projekte/Beispiele/LB6“ und kopieren das „Bediengerat\_1“ nach „2PktRgVs“.

Um dieses Bediengerät in das Projekt zu integrieren, müssen einige Einstellungen vorgenommen werden:

1. Die beiden Geräte „SIMATIC 300-Station“ und „Bediengerat\_1“ müssen mit „Netz konfigurieren“ an den MPI-Bus angeschlossen werden.
2. Alle zwischen PLCSIM und WinCC flexible auszutauschenden Variablen müssen in WinCC (DB51) als globale Variable deklariert und aus der Symboltabelle entfernt werden. Außerdem muss die Variablen-tabelle aus der Symboltabelle und dem Bausteinordner entfernt werden. Die Anschlüsse des CFC-Plans sind mit den Variablen aus dem globalen Datenbaustein WinCC (DB51) zu verbinden.
3. In WinCC flexible müssen alle Variablen aus dem globalen Datenbaustein WinCC übernommen werden.
4. Die Frontplatte muss nun an die Erfordernisse des Zweipunktreglers angepasst werden. Dazu löscht man die überflüssigen Komponenten und fügt die nun zusätzlich erforderlichen Komponenten ein, indem entsprechende Einträge kopiert werden. Dabei kann die Frontplatte das folgende Aussehen annehmen:

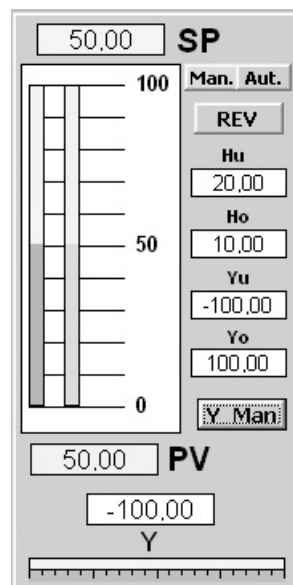


Abbildung 60: Frontplatte des Zweipunktreglers

Anschließend müssen die Verknüpfungen zu den Variablen des Projektes angepasst werden.

Damit die Arbeitsweise des Zweipunktreglers anschaulich dargestellt werden kann, erstellen wir eine weitere Visualisierung auf dieser Bedienoberfläche. Hierin wird mit einem Schieberegler (Scrollbar) die Regelgröße PV im Bereich zwischen 0 und 100 vorgegeben. Da der Schieberegler nur ganze Zahlen akzeptiert, wird die Variable „PV\_Int“ als interne WinCC flexible-Variable definiert. In einer Zeichnung, die mit irgendeinem Grafik-Programm erstellt und als bmp-Datei im Projektordner abgespeichert wird, wird die Kennlinie für bestimmte Parameter gezeichnet.

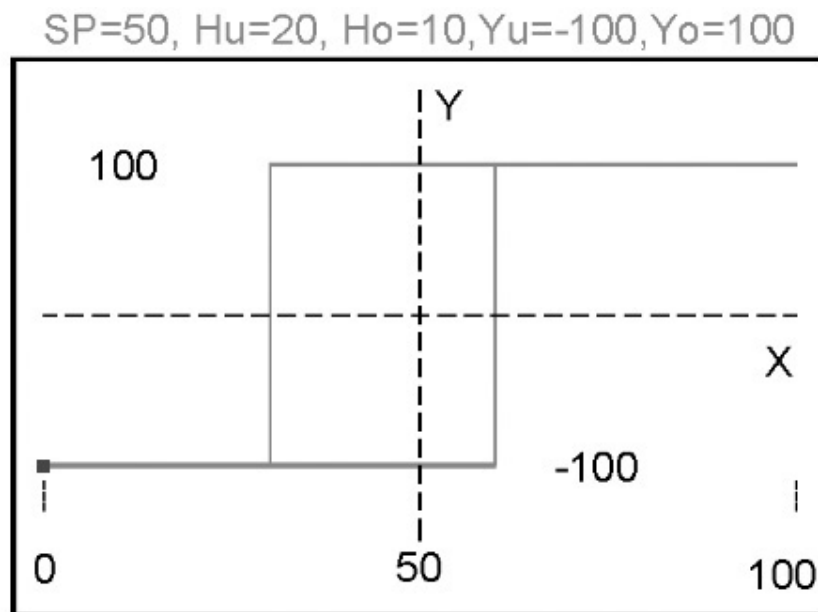
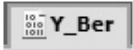



Abbildung 61: Kennlinie des 2-Punkt-Reglers

Dieses Bild kann man in die Visualisierung einfügen, indem man aus der Werkzeugleiste „Basisobjekte / Grafikanzeige“ eine Grafikanzeige in die Arbeitsfläche zieht, unter „Allgemein“ mit dem linken Icon (Neue Grafik aus Datei erstellen) aus der Ordnerübersicht die Datei mit der Grafik auswählt, sowie die Position und die Abmessungen einstellt. Die Datei sollte zuvor in den Projektordner gelegt worden sein.

Zur Anzeige des aktuellen Betriebspunktes auf der Kennlinie wird ein rotes Quadrat mit 10 x 10 Pixel angelegt, dessen Position mit „Animationen / Direkte Bewegung“ mit Hilfe der internen WinCC flexible Variablen „PV\_Skala“ und „Y\_Skala“ verändert wird. Diese internen Variablen werden in den Skripten „PV\_ber“ und „Y\_Ber“ berechnet. Damit die Positionen genau zur dargestellten Kennlinie passen, müssen die Anfangs- und Endpositionen ermittelt und daraus die Berechnungsvorschrift bestimmt werden. Die Skripte sind:



```
12|SmartTags("Y_Skala")=-0.8*SmartTags("WinCC.Y")
```



```
12|SmartTags("WinCC.PV")=1.0*SmartTags("PV_Int")  
13|SmartTags("PV_Skala")=3*SmartTags("PV_Int")
```

Abbildung 62: Skripte "Y\_Ber" und "PV\_Ber"

In dieser Grafik muss die vertikale Position bei  $Y = -100$  um 80 Pixel unter und bei  $Y = 100$  um 80 Pixel über der Startposition liegen. Y muss also mit  $-0,8$  multipliziert werden.

Zunächst wird bei Veränderung des Schiebers der Integerwert PV\_Int in den entsprechenden Real-Wert „WinCC.PV“ umgerechnet. Danach wird die horizontale Verschiebung des roten Quadrates berechnet, wobei zu einem Wert PV\_Int = 100 eine Verschiebung um 300 gehört. Das führt zu einer Multiplikation mit 3.

Die Skripte werden bei Wertänderung der Variablen „PV\_Int“ bzw. „WinCC.Y“ gestartet.

Während des Testes des Reglers kann man sehen, wie der Betriebspunkt auf der Kennlinie hin- und hergeht und an den Schaltpunkten zwischen den beiden möglichen Werten springt. Eine Momentaufnahme sehen Sie hier:

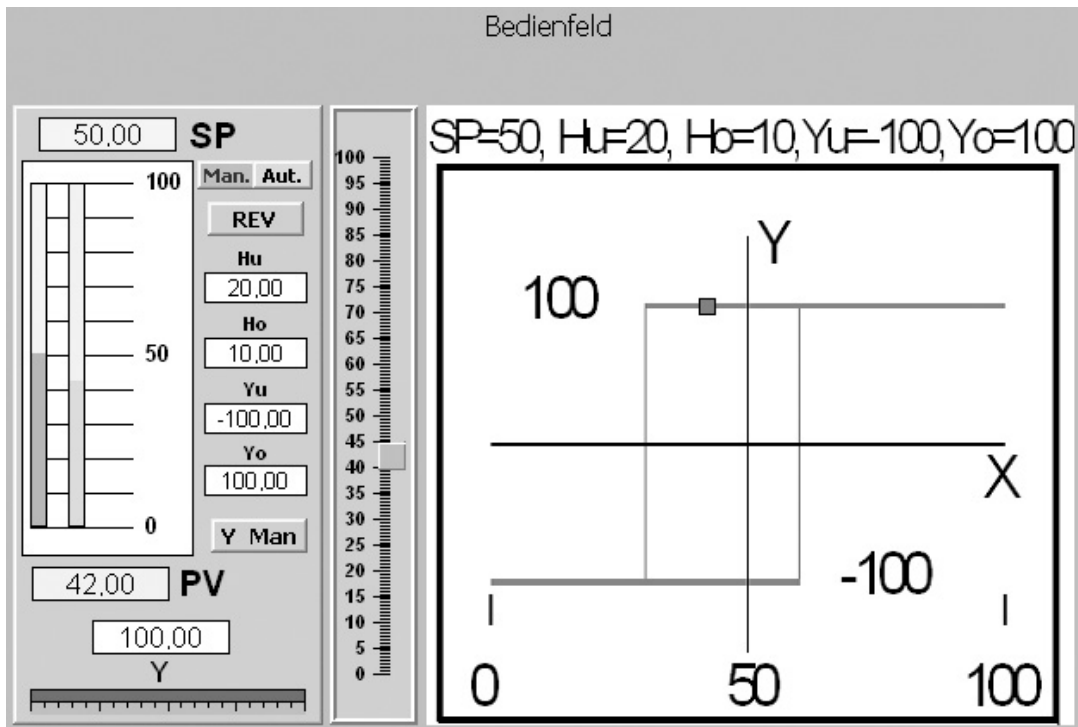


Abbildung 63: Visualisierung des 2-Punkt-Reglers

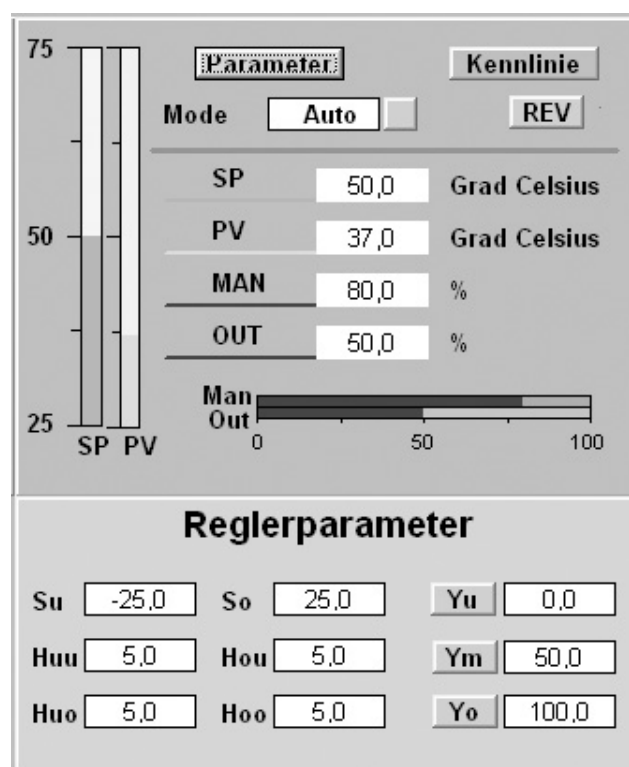
Nach dem erfolgreichen Test wird das Projekt abgespeichert. Es steht nun für die 2-Punkt-Regelung der Neutralisationsanlage zur Verfügung.



### Aufgabe 13:

Entsprechend der Visualisierung zum PI-Regler im Lehrbrief 6 soll für den in Aufgabe 12 erstellten 3-Punkt-Regler im Projekt „3PktRgVs“ eine Visualisierung angelegt werden. Dazu wird zunächst das Projekt „3PktRegl“ geöffnet und mit dem neuen Namen „3PktRgVs“ abgespeichert.

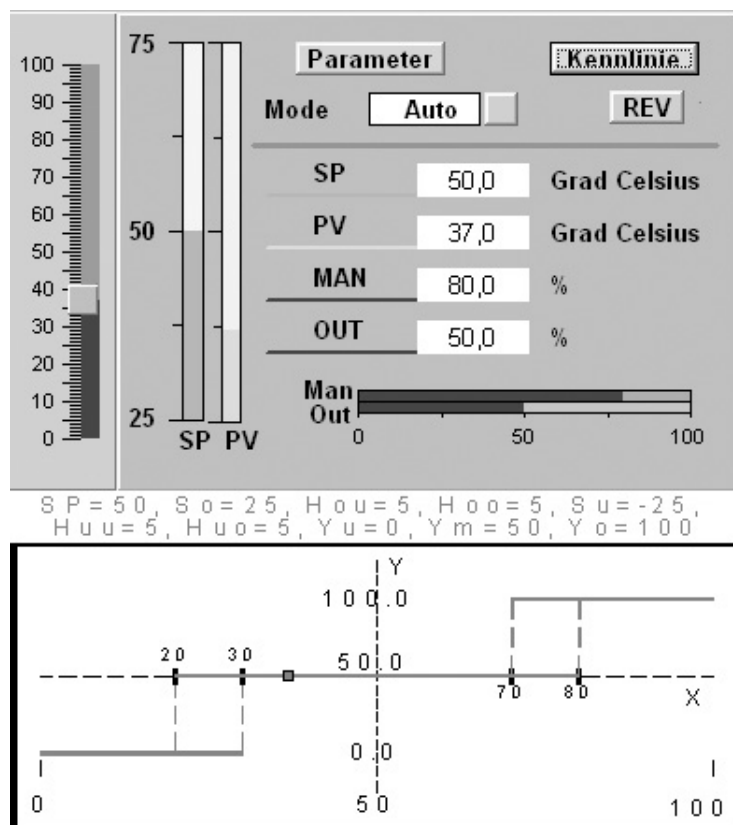
13.1 Erstellen Sie für den 3-Punkt-Regler eine Anzeige und ein Parametrierfeld nach dem folgenden Muster:



In der Betriebsart „Manuell“ soll mit dem Eingangssignal Y\_MAN entweder Yu, Ym oder Yo ausgegeben werden. Die Werte für die manuelle Stellgröße werden in den Eingabefeldern neben den entsprechenden Buttons Yu, Ym und Yo vorgegeben. Die konkrete Auswahl wird mit den Buttons vorgenommen, wobei mit einem Klick auf eine der drei Tasten die booleschen Werte („B\_Yu“, „B\_Ym“ oder „B\_Yo“) der anderen Tasten auf FALSE, der boolesche Wert der angeklickten Taste auf TRUE und die Handstellgröße auf den dazu gehörenden Stellwert gesetzt werden. Das Zahlenfeld der ausgewählten Stellgröße wird gelb hinterlegt.

Sie können die Anzeige und das Parametrierfeld des PI-Reglers aus dessen Projekt im Lehrbrief 6 übernehmen und entsprechend der Abbildung anpassen. Anschließend müssen Sie auch die Anbindung an die Variablen anpassen. Gehen Sie dabei so vor, wie das im Beispiel gezeigt wurde.

- 13.2 Damit man die Funktion des 3-Punkt-Reglers gut beobachten kann, soll auch hier eine Visualisierung mit Kennlinie und Schieberegler entsprechend der folgenden Abbildung erstellt werden. Mit dem Button „Kennlinie“ werden die Kennlinie und das Quadrat, das den Betriebspunkt des Reglers angeben, sichtbar und unsichtbar geschaltet. Der Schieberegler liefert zum Test der Reglerkennlinie Regelgrößen zwischen 0 und 100, obwohl für den späteren Einsatz nur Regelgrößen zwischen 25 und 75 Grad Celsius erwartet werden.



- 13.3 Testen Sie die Funktion des 3-Punkt-Reglers!  
Wenn er fehlerfrei funktioniert, legen Sie auch diesen in der Bibliothek „Meine\_IEC\_Bib“ im Ordner „Regler / Bausteine“ ab!

Die Lösung finden Sie am Ende des Lehrbriefes.

## 7.2 pH-Regelung mit Zweipunkt-Regler

Nun soll der Zweipunktregler mit Hysterese für die pH-Regelung verwendet werden. Dazu gehen wir von dem Projekt „2PktRgVs“ aus. Wir speichern es mit dem Namen „2Pkt\_pH“ wieder ab. Aus dem Projekt „pHDyn-PID“ aus dem Lehrbrief 7 werden der FB „pHRegelstrecke“, die Funktion „Skalierung“ und der CFC-Plan „PID\_pH“ durch Kopieren übernommen.

Nach dem Übersetzen der SCL-Quellen und dem Bereinigen der Symboltabelle und des Bausteinordners, indem die Elemente des PID-Reglers (Mein\_PIDT1-Regler, Mein\_I, Mein\_D, FB103, FB106 u. ä.) entfernt bzw. eingetragen (FB107) werden, kann der CFC-Plan „Zweipunkt-Regler“ entfernt und „PID\_pH“ in „2PktPHRK“ umbenannt werden. Hierin wird der PID-Regler gegen den Zweipunktregler ausgetauscht.

Hier sehen Sie die verschiedenen Einstellungen:

Status	Symbol	Adresse	Datentyp	Kommentar
	Cycle Execution	OB 1	OB 1	
	Mein_2Pkt_Regler	FB 116	FB 116	
	pHRegelstrecke	FB 1	FB 1	
	Skalierung	FC 101	FC 101	
	WinCC	DB 51	DB 51	

Abbildung 64: Symboltabelle

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	Auto	BOOL	FALSE	
+2.0	Ho	REAL	0.000000e+000	
+6.0	Hu	REAL	0.000000e+000	
+10.0	PV	REAL	0.000000e+000	
+14.0	REV	BOOL	FALSE	
+16.0	SP	REAL	0.000000e+000	
+20.0	Y	REAL	0.000000e+000	
+24.0	Y_MAN	BOOL	FALSE	
+26.0	Yo	REAL	0.000000e+000	
+30.0	Yu	REAL	0.000000e+000	
+34.0	Z	INT	3	
+36.0	PV_phys	REAL	0.000000e+000	
+40.0	Y_phys	REAL	0.000000e+000	
=44.0		END_STRUCT		

Abbildung 65: Globale Variablen in WinCC

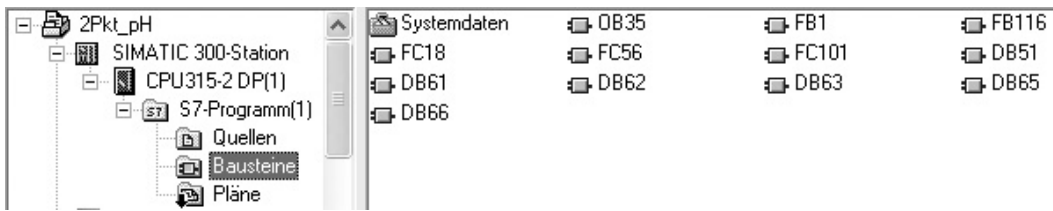


Abbildung 66: Baueinordner

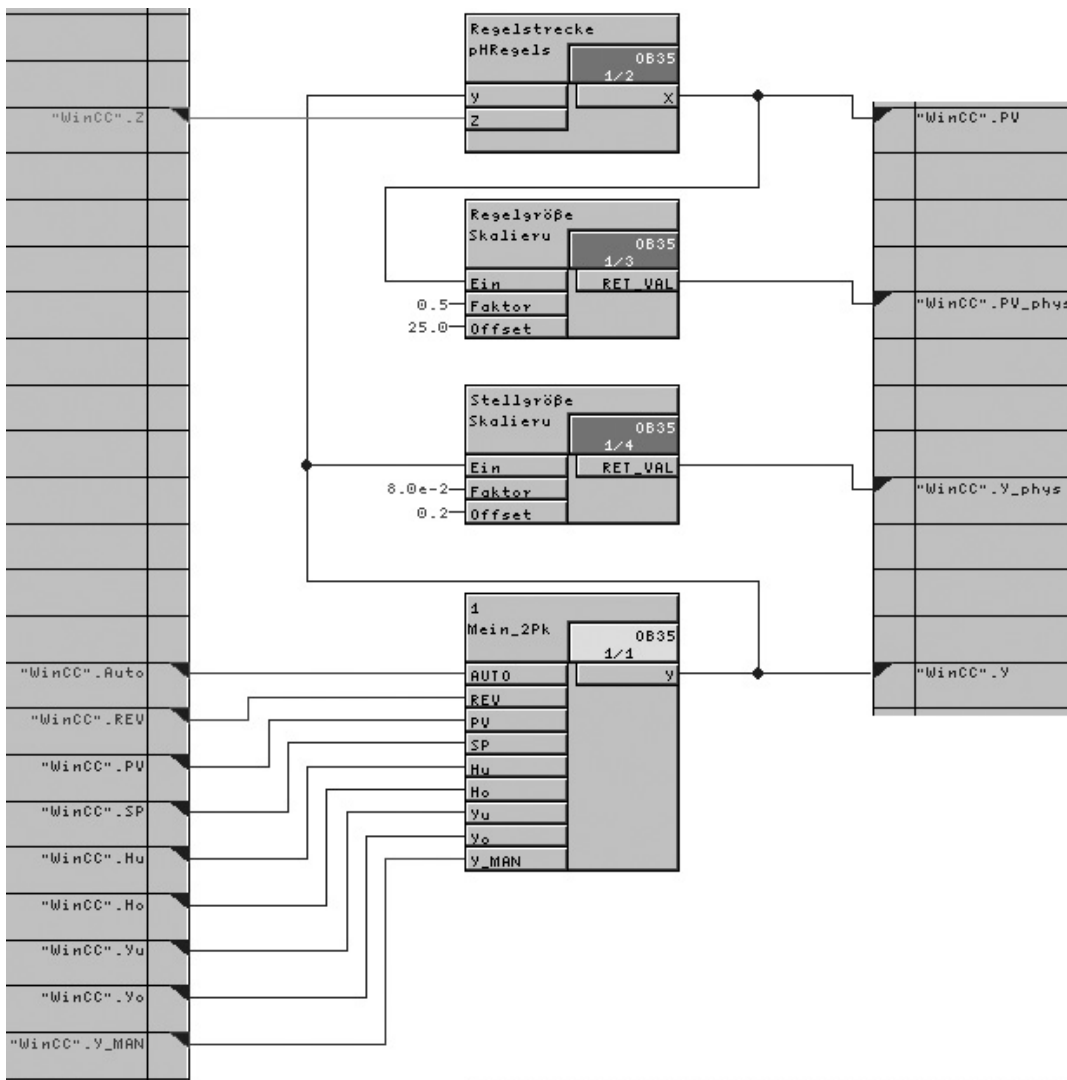


Abbildung 67: CFC-Plan

Die Regelstrecke übernimmt die Stellgröße Y und liefert die Regelgröße X in %. Die Störgröße wird als ganze Zahl zwischen 1 und 6 vorgegeben. Damit diese Signale später nach Bedarf mit ihren physikalischen Einheiten

(bar bzw. dimensionslos für pH) oder in % angezeigt bzw. im Trend aufgezeichnet werden können, benötigen wir zur Skalierung die Funktion „Skalierung“.

Nach dem fehlerfreien Übersetzen kann das Programm nach PLCSIM geladen und gestartet werden.

Als nächstes müssen wir die Visualisierung des Prozesses „pH-Regelung“, das ja ursprünglich für den PIDT1-Regler konzipiert war, an den Zweipunktregler anpassen. Diese Visualisierung soll folgendes Aussehen erhalten. Am linken Rand sieht man das Trenddiagramm, das soweit verkleinert wurde, dass man die anderen Elemente sehen kann. Vor dem Übersetzen (Generieren) wird dieses Diagramm über die ganze Bedienoberfläche gezogen, damit es mit dem Trend-Button im Runtime-Modus über alle Bedienelemente gelegt wird.

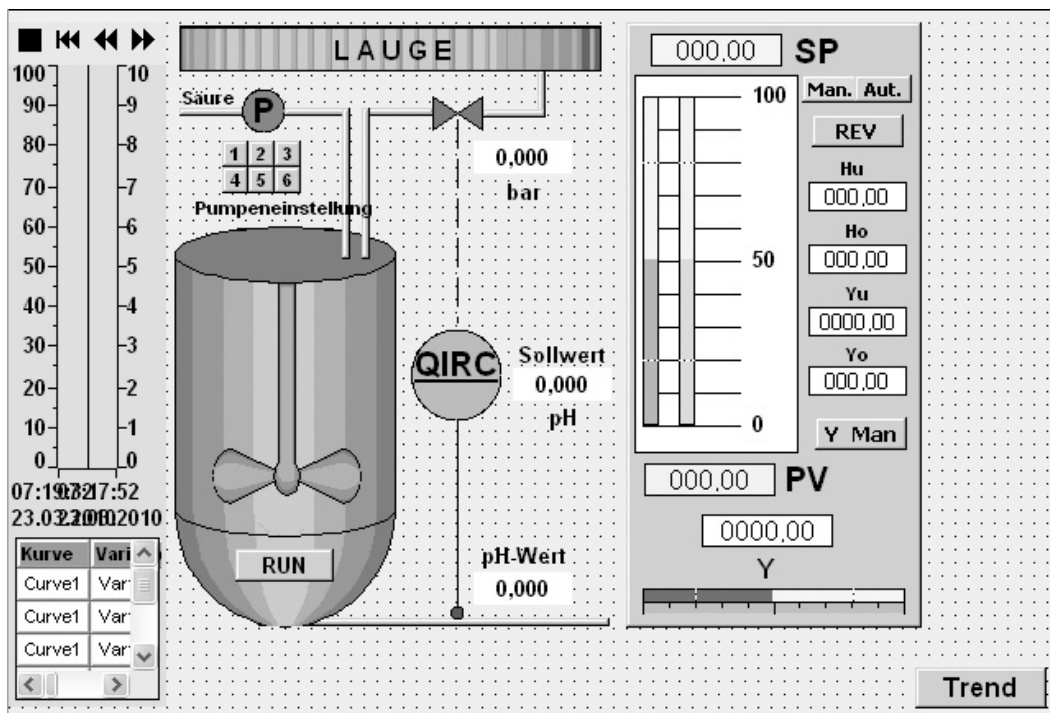


Abbildung 68: Fließbild und Fronttafel der pH-Regelung

Damit wir die Visualisierung nicht komplett neu formulieren müssen, kopieren wir aus dem Projekt „pHDynPID“ das Bild „pH-Regelung“ zusätzlich in unser Projekt. Dazu müssen allerdings in den WinCC flexible- Projekten die Bilder „pH-Regelung“ und „Bedienfeld“ geöffnet werden. Dann kann man das Bild „pH-Regelung“ kopieren und in das Projekt „2Pkt\_pH“ unter „Bediengeräet\_1 / Bilder“ einfügen. Nun sollte das Projekt „pHDynPID“ mit WinCC flexible geschlossen werden.

Das Bild „pH-Regelung“ kann man nun in „pH\_2Pkt\_Regelung“ umbenennen und darin die Frontplatte mit der Parametereingabe löschen. Aus dem Bild „Bedienfeld“ kopiert man die Frontplatte des Zweipunktreglers an die Stelle, wo zuvor die Frontplatte des PID-Reglers stand. Danach kann man das Bedienfeld löschen. Mit „Geräteeinstellungen / Geräteeinstellungen / Startbild muss „pH\_2Pkt\_Regelung“ als Startbild definiert werden.

Nun müssen wir alle Anzeigen und Eingaben darauf kontrollieren, ob die Anbindungen des CFC-Plans an die Variablen z. B. in WinCC (DB51) richtig eingestellt sind. Ggf. müssen diese Angaben korrigiert werden. Einige Skripte zur Berechnung von internen Größen können gelöscht bzw. müssen geändert oder eingefügt werden.

**Praxistipp:**

Bei der Fehlerkorrektur geht man am einfachsten so vor, dass man die Änderungen, die man schnell findet, vornimmt und danach das WinCC flexible-Projekt generiert. Danach korrigiert man weitere Fehler, die man leicht korrigieren kann. Man sollte nicht versuchen, zu Beginn bereits alle Fehler zu beheben, weil Fehler oft Folgefehler sind und bei der Behebung eines Fehlers automatisch verschwinden werden. Man übersetzt also nach der Korrektur einiger Fehler das Projekt erneut und wendet sich dann den verbliebenen Fehlern zu.



Da wir das Zeitverhalten der Zweipunktregelung beobachten wollen, benötigen wir das Trenddiagramm. Hier definieren wir die Signale PV, SP, Y in Prozent auf der linken Achse und Z als dimensionslose Zahl auf der rechten Achse.

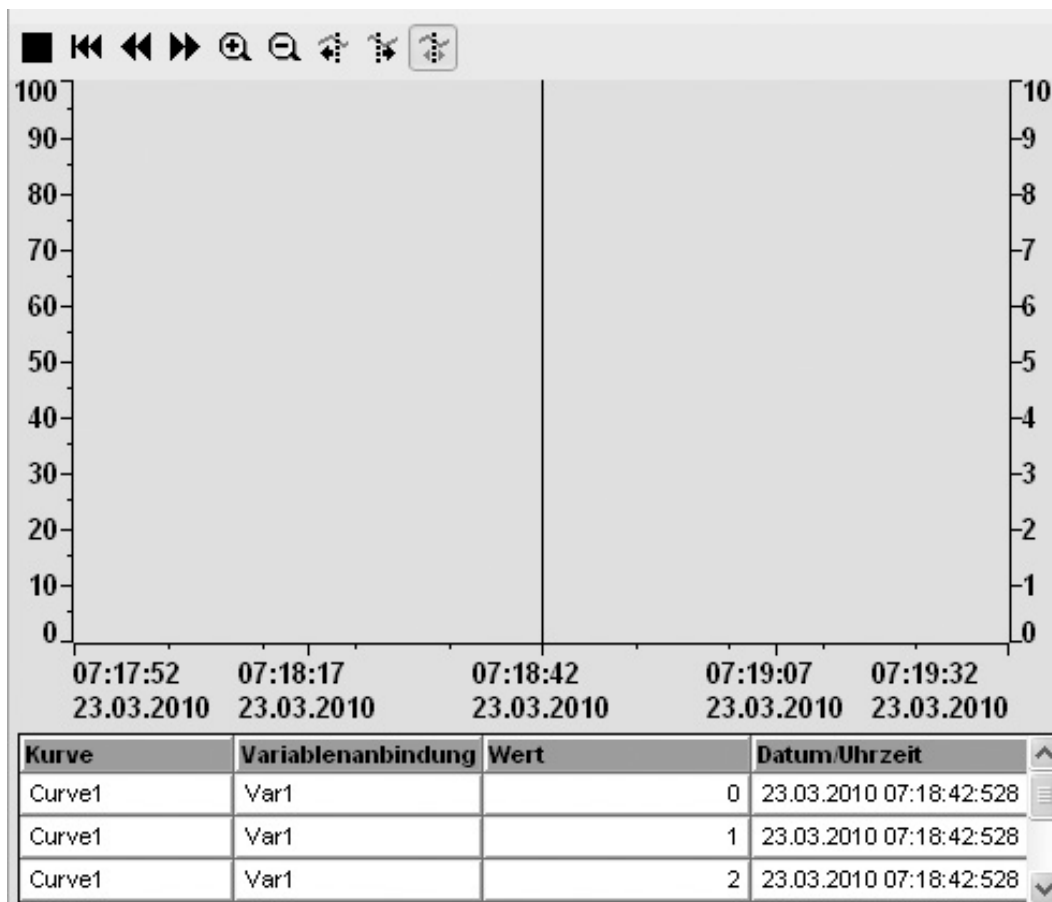


Abbildung 69: Trenddiagramm für die pH-Reglung

Wenn alle Änderungen durchgeführt und das Projekt fehlerfrei übersetzt wurde, kann man die Regelung testen.

Zunächst müssen wir entscheiden, auf welchen Wert wir die kleine und auf welchen die große Stellgröße einstellen. Ein Blick auf das früher gezeigte Kennlinienfeld zeigt, dass man alle Störgrößen zwischen 1 und 5 mit Stellgrößen zwischen 25 und 75 % ausregeln kann. Wenn wir davon ausgehen dürfen, dass die Pumpe so gut wie nie in Stufe 6 steht, können wir auf das Ausregeln dieser Störung verzichten. Je näher die beiden Stellgrößen zusammen liegen, umso kleiner wird die Amplitude der Arbeitsbewegung.

Als Hysteresebreite wählen wir jeweils 1 %. Dadurch wird die Amplitude durch die Hysterese um 2 % größer ausfallen.

Die nächste Abbildung zeigt das Störverhalten:

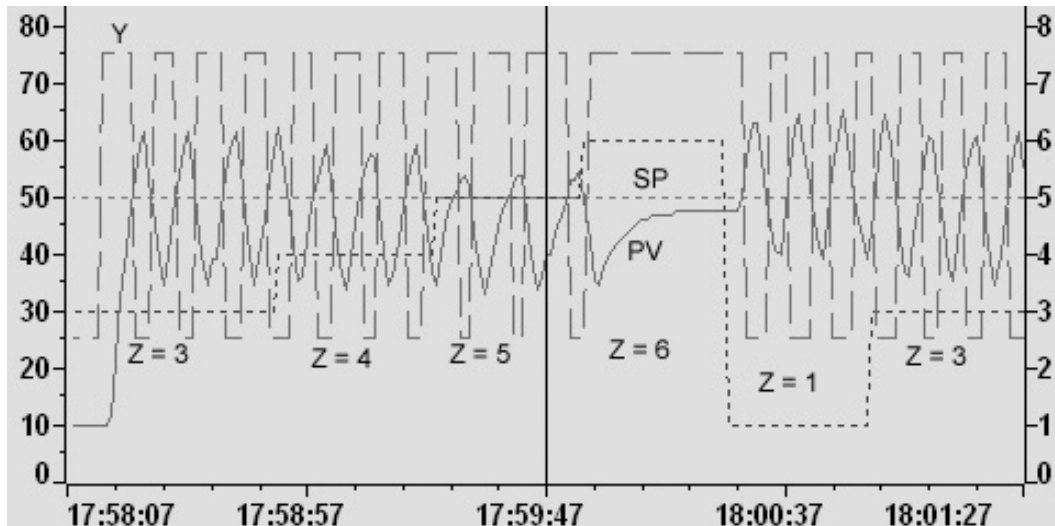


Abbildung 70: Störverhalten der Zweipunkt-pH-Regelung

Man sieht, dass die Regelgröße nach dem Umschalten der Betriebsart des Reglers von „Manuell“ auf „Automatik“ mit dem Verhalten einer Totzeit und Verzögerung 1. Ordnung (Dynamik der Regelstrecke) zum Sollwert ansteigt und anschließend um den Sollwert eine Arbeitsbewegung durchführt. Man erkennt, dass der Mittelwert der Arbeitsbewegung immer entweder unter oder über dem Sollwert liegt, also eine bleibende Regelabweichung auftritt. Für die Pumpenstellung  $Z = 6$  ist die Störung so groß, dass der Regler mit der maximal möglichen Stellgröße von 75 % den Störeinfluss nicht kompensieren kann.

Für das nächste Trenddiagramm wurde die Zeitachse verändert. Um die Schaltpunkte deutlicher werden zu lassen, wurde die Hysteresebreite jeweils auf 5 % eingestellt.

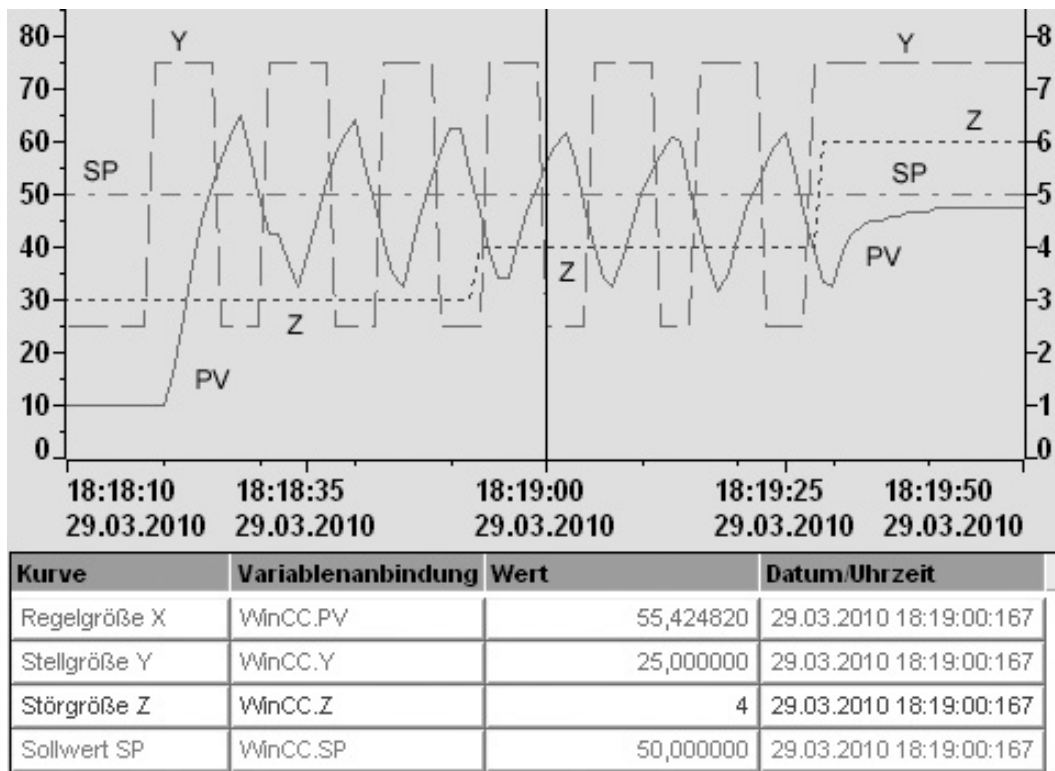


Abbildung 71: Ausschnitt aus dem Trenddiagramm

Wenn man die Stellgröße und die Regelgröße in der Nähe des Sollwertes genau betrachtet, kann man erkennen, dass der Schaltvorgang jeweils um 5 % über- bzw. unterhalb des Sollwertes erfolgt.

Nach einer Veränderung des Sollwertes von pH=7 auf pH=8 liegen die Schaltpunkte 5 % über bzw. unter dem Sollwert pH=8. Da der Weg bis zum stationären Endwert für Y = 75 % wesentlich kürzer ist als der zum unteren stationären Endwert für Y = 25 %, läuft die Regelgröße langsam hoch und schnell runter. Dadurch liegt der Mittelwert der Arbeitsbewegung deutlich unterhalb des Sollwertes. Entsprechend umgekehrt sieht die Situation für einen Sollwert von pH=6 aus.

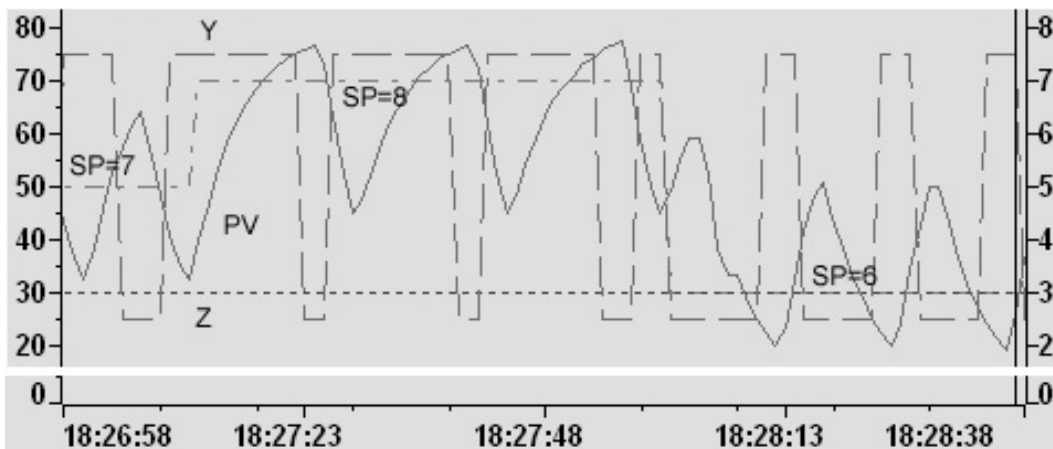


Abbildung 72: Führungsverhalten der Zweipunkt-pH-Regelung

In diesem Beispiel sieht man, dass die Regelgröße meistens um den Sollwert pendelt. Diese Arbeitsbewegung ist der Preis für den Einsatz eines sehr einfachen und damit billigen Reglers und Stellorgans. In vielen praktischen Prozessen kann man eine solche Arbeitsbewegung zugunsten einer preiswerten Lösung in Kauf nehmen. Dabei treten allerdings häufig wesentlich kleinere Amplituden der Arbeitsbewegung auf als hier.

#### Aufgabe 14:

Nun soll der Dreipunktregler aus dem Projekt „3PktRgVs“ zur Temperaturregelung am Wärmetauscher eingesetzt werden. Öffnen Sie dazu das Projekt „WaeDynPI“ aus dem Ordner „C:/STEP7\_Projekte/Aufgaben/LB7“ und speichern Sie es mit dem Namen „Waet3Pkt“ im Ordner „C:/STEP7\_Projekte/Aufgaben/LB8“ wieder ab!

Falls Sie den FB „Mein\_3Pkt\_Regler“ noch nicht in der Bibliothek „Meine\_IEC\_Bib“ abgelegt haben, kopieren Sie den FB „Mein\_3Pkt\_Regler“ in der Bibliothek „Meine\_IEC\_Bib“ in den Ordner „Regler / Bausteine“!

Öffnen Sie zusätzlich das Projekt „3PktRgVs“ aus dem Ordner „C:/STEP7\_Projekte/Aufgaben/LB8“ als Vorlageprojekt und kopieren Sie aus dem Ordner „Quellen“ dieses Projektes den Regler „Mein\_3Pkt\_Regler“ in das neue Projekt und löschen Sie die SCL-Quelle „Mein\_PI\_Regler“! Kopieren Sie entsprechend aus dem Bausteinordner von „3PktRgVs“ den FB117 (Mein\_3Pkt\_Regler) und den Datenbaustein DB51 (WinCC) in das neue Projekt und löschen Sie FB112 (Mein\_PI\_Regler)! DB51 wird überschrieben und FB103 (Mein\_I) muss noch gelöscht werden. Tragen Sie in DB51 (WinCC) zusätzlich die booleschen Variablen „RUN“ und „FP“ (real) ein!



Öffnen Sie den CFC-Plan „Regelkreis“ und tauschen Sie darin den PI-Regler gegen den Dreipunktregler aus und verbinden Sie die Anschlüsse mit den WinCC-Variablen! Übersetzen Sie den CFC-Plan als Programm, korrigieren Sie ggf. Fehler, öffnen, laden und starten Sie PLCSIM!

Nun soll die Visualisierung zusammengestellt werden. Dazu muss in beiden Projekten (neu: „Waet3Pkt“ / Vorlage: 3PktRgVs“) WinCC flexible mit „RM auf WinCC flexible RT / Objekt öffnen“ gestartet werden. Öffnen Sie mit einem DK auf das jeweilige Bild die Startbilder „Temperaturregelung“ und „Bedienfeld“! Kopieren Sie nun aus der Vorlage das Bild „Bedienfeld“ in das neue Projekt!

Löschen Sie im Bild „Temperaturregelung“ die Bedienelemente für den PI-Regler und übernehmen Sie aus dem Bild „Bedienfeld“ die Bedienelemente des Dreipunktreglers (ohne Schieberegler und Kennlinie)! Dabei werden gleichzeitig die Variablen, die mit der Grafik verknüpft sind, in das neue Projekt übernommen! Lassen alle doppelt vorkommenden Variablen (...\_0, ...\_1 etc.) in der Variablenliste stehen!

Generieren Sie danach die Visualisierung und korrigieren Sie noch vorhandene Fehler!

Wenn Sie das Projekt fehlerfrei übersetzt haben, können Sie die Funktion testen.

Stellen Sie den Dreipunkt-Regler bitte als Zweipunktregler mit  $Y_u = 25\%$ ,  $Y_o = 75\%$  sowie einer Hysteresebreite von 1 Grad nach oben und nach unten ein! Der Sollwert soll 50 Grad betragen.

Nehmen Sie mit einem Trenddiagramm den Verlauf von Stell- Regel- und Störgröße sowie vom Sollwert auf, nachdem der Dreipunktregler bei einem Produktstrom von 10 t/h und einer Handstellgröße von 25 % von „Manuell“ nach „Auto“ umgestellt wurde!

Zeichnen Sie im Trenddiagramm auf, wie die Regel- und die Stellgröße ausgehend vom einem Sollwert von 50 Grad und einem Produktstrom von 10 t/h auf Veränderungen des Produktstroms auf 12 t/h, anschließend auf 8 t/h und zurück auf 10 t/h reagiert! Kommentieren Sie den Verlauf von Regel- und Stellgröße!

Wiederholen Sie diesen Versuch, wobei allerdings nun der Produktstrom konstant auf 10 t/h stehen bleiben soll und der Sollwert von 50 Grad auf 40 Grad, danach auf 60 Grad und zurück auf 50 Grad geht. Kommentieren Sie den Verlauf von Regel- und Stellgröße!

Die Lösung finden Sie am Ende des Lehrbriefes.

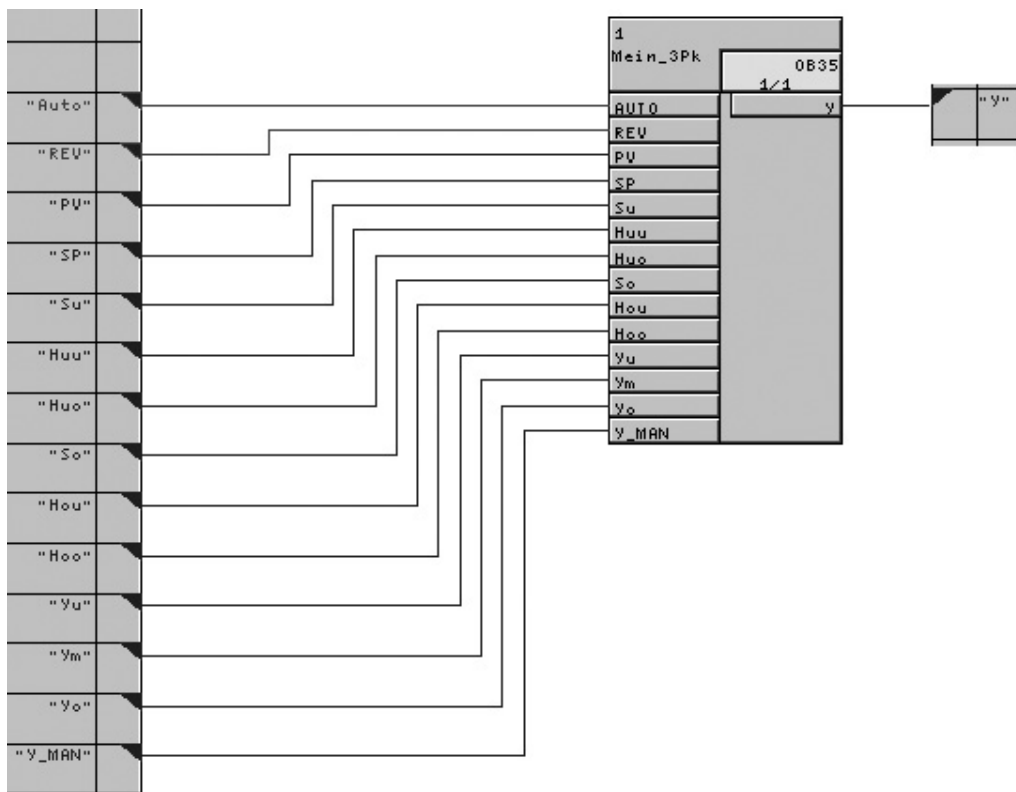


Abbildung 113: CFC\_Plan „Dreipunktregler“

Auf einen Test an Hand einer Wertetabelle wird hier verzichtet, da wir im nächsten Projekt eine Visualisierung der Kennlinie und der Schaltvorgänge anlegen werden.

### Lösung Aufgabe 13:

- 13.1 Projekt „3PktRegl“ öffnen und wieder unter „3PktRgVs“ abspeichern. Dieses Projekt erneut öffnen und zusätzlich das Projekt „PI\_R\_Vis“ aus den Aufgaben von Lehrbrief 6 öffnen. Daraus das „Bediengerat\_1“ kopieren und in „3PktRgVs“ einfügen.

Globalen Datenbaustein „WinCC“ (DB51) anlegen und alle Variablen aus der Symboltabelle hier eintragen. Diese Variablen aus der Symboltabelle entfernen.

Im CFC-Plan „Dreipunktregler“ die Anschlüsse mit den WinCC-Variablen verbinden.

Den CFC-Plan als Programm übersetzen und PLCSIM laden und starten.

Die alten WinCC-Variablen in WinCC flexible löschen und die neuen aus diesem Projekt einfügen.

Nun das Bedienfeld anpassen und alle angesprochenen Variablen neu zuordnen

Als interne WinCC flexible\_Variablen für den Aufruf des Parametrierfeldes die boolesche Variable „Parameter“ und für den Zustand der Buttons für die Handstellgröße die internen „B\_Yu“, „B\_Ym“ und „B\_Yo“ definieren und zuordnen.

Testen, ob das Projekt fehlerfrei übersetzt werden kann und ob die Funktionsweise korrekt ist.

- 13.2 Einen Schieberegler einfügen für die Regelgröße PV, die hier unabhängig von der späteren Verwendung Werte zwischen 0 und 100 aufweist. Da hier ganze Zahlen geliefert werden, müssen wir eine interne Variable „PV\_Int“ für „WinCC flexible“ definieren

Mit einem Grafikprogramm wird die Reglerkennlinie für eine bestimmte Reglereinstellung gezeichnet und als Grafikanzeige in die Visualisierung eingefügt. Damit diese Anzeige nach Belieben ein- und ausgeschaltet werden kann, wird die interne WinCC flexible-Variable „Kennlinie“ definiert, mit der die Grafikanzeige sichtbar und unsichtbar gemacht wird. Der aktuelle Betriebspunkt, der durch die Regelgröße PV und die Stellgröße Y vorgegeben wird, wird durch die Position eines Quadrates angezeigt, das mit direkter Bewegung über die internen WinCC flexible-Variablen „PV\_Skala“ und „Y\_Skala“ vom Typ Integer verschoben wird. Diese beiden Variablen werden im Skript „PV\_Ber\_Skala“ berechnet. Dieses dient zur Berechnung der Regelgröße „WinCC.PV“ und der Skalenwerte „PV\_Skala“ und „Y\_Skala“ und wird aufgerufen, wenn „PV\_Int“ oder wenn „WinCC.Y“ sich ändert. Auch das Quadrat wird mit der booleschen Variablen „Kennlinie“ sichtbar und unsichtbar geschaltet.

Für die Berechnungen im Skript „PV\_Ber\_Skala“ müssen in der Grafik die Startposition und die Veränderungen in PV- und Y-Richtung ermittelt werden.

```
SmartTags("WinCC.PV")=1.0*SmartTags("PV_Int")  
SmartTags("PV_Skala")=3.6*SmartTags("WinCC.PV")  
SmartTags("Y_Skala")=- 0.82*SmartTags("WinCC.Y")
```

Abbildung 114: Skript "PV\_Ber\_Skala"

Eine Momentaufnahme der Visualisierung mit Reglerkennlinie wurde bereits in der Aufgabenstellung gezeigt.

### Lösung Aufgabe 14:

Nach dem Kopieren und Löschen der neuen bzw. nicht mehr benötigten Funktionsbausteine enthält das Projekt folgende Bausteine:

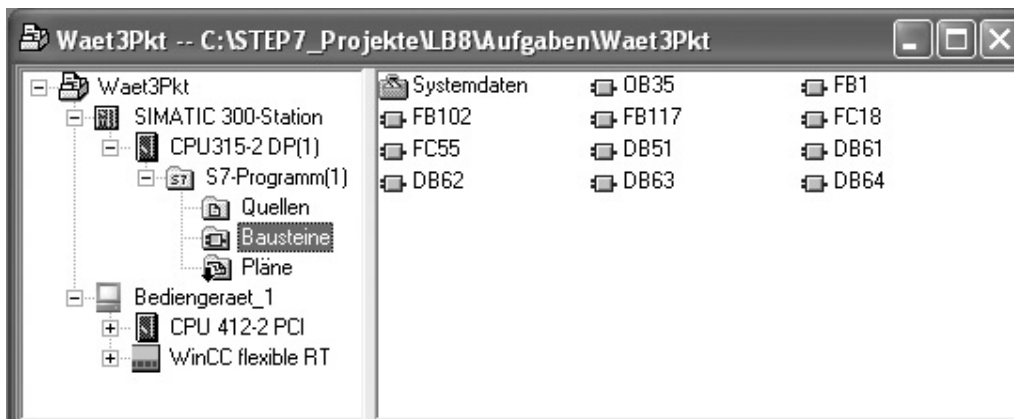


Abbildung 115: Bausteine des Projektes

Der globale Datenbaustein „WinCC“ (DB51) sieht nach der Bearbeitung folgendermaßen aus:

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	Auto	BOOL	FALSE	
+2.0	Hoo	REAL	0.000000e+000	
+6.0	Hou	REAL	0.000000e+000	
+10.0	Huo	REAL	0.000000e+000	
+14.0	Huu	REAL	0.000000e+000	
+18.0	PV	REAL	0.000000e+000	
+22.0	REV	BOOL	FALSE	
+24.0	So	REAL	0.000000e+000	
+28.0	SP	REAL	0.000000e+000	
+32.0	Su	REAL	0.000000e+000	
+36.0	Y	REAL	0.000000e+000	
+40.0	Y_MAN	REAL	0.000000e+000	
+44.0	Ym	REAL	0.000000e+000	
+48.0	Yo	REAL	0.000000e+000	
+52.0	Yu	REAL	0.000000e+000	
+56.0	RUN	BOOL	FALSE	
+58.0	FP	REAL	0.000000e+000	
=62.0		END_STRUCT		

Abbildung 116: Globaler Datenbaustein "WinCC"

An diese globalen Daten wird nun der CFC-Plan angeschlossen, nachdem der PI-Regler gegen den Dreipunktregler ausgetauscht worden ist.

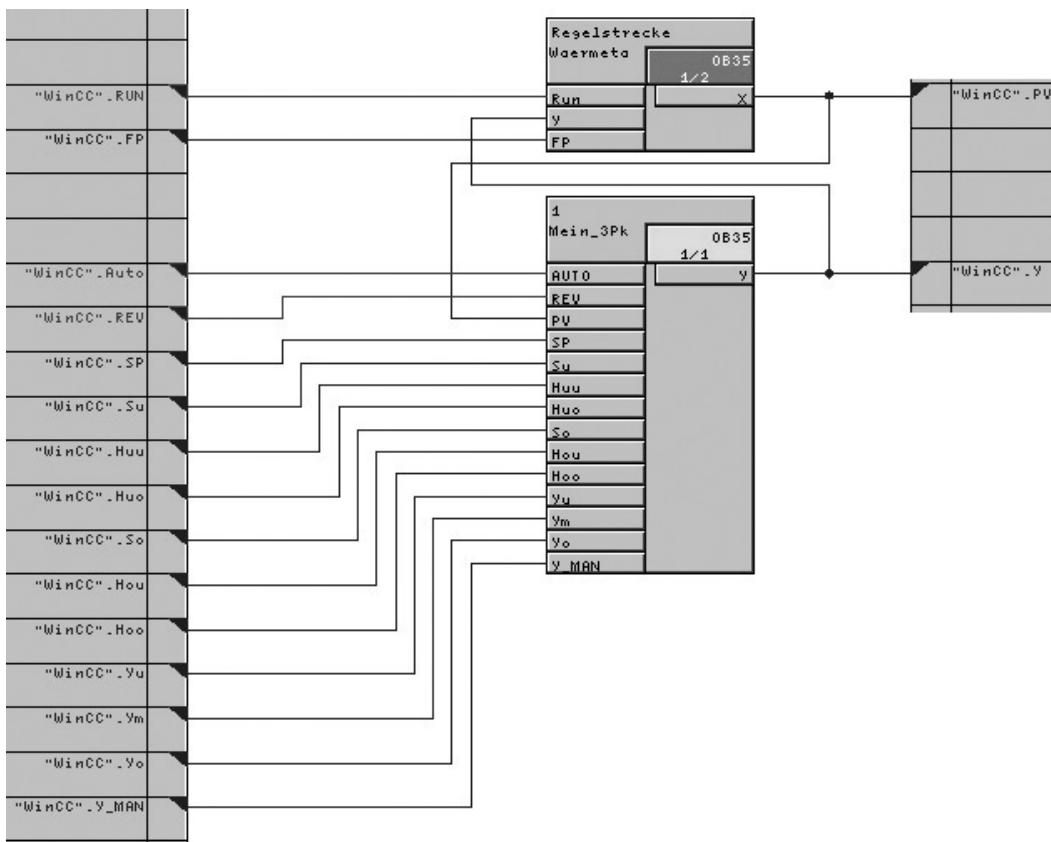


Abbildung 117: Veränderter CFC\_Plan "Regelkreis"

Man sollte nun genau den Anweisungen in der Aufgabenstellung folgen. Beim Kopieren von Grafikelementen werden die daran angebotenen Variablen mit übertragen und unter „Variablen“ eingetragen. Wenn dieser Name bereits vorhanden ist, wird mit „\_n“ eine Nummer angehängt. Man sollte diese Einträge stehen lassen, damit die Anbindungen der Grafikelemente erhalten bleiben.

Die Trendanzeige liegt über allen anderen Elementen der Grafik, damit man mit dem Button „Trend“ diese ein- und ausschalten kann. Damit man bei der Bearbeitung an die anderen Elemente gelangen kann, wird die Grafikanzeige des Trends verkleinert. Vor der endgültigen Verwendung muss das Fenster für den Trend wieder „aufgezogen“ werden.

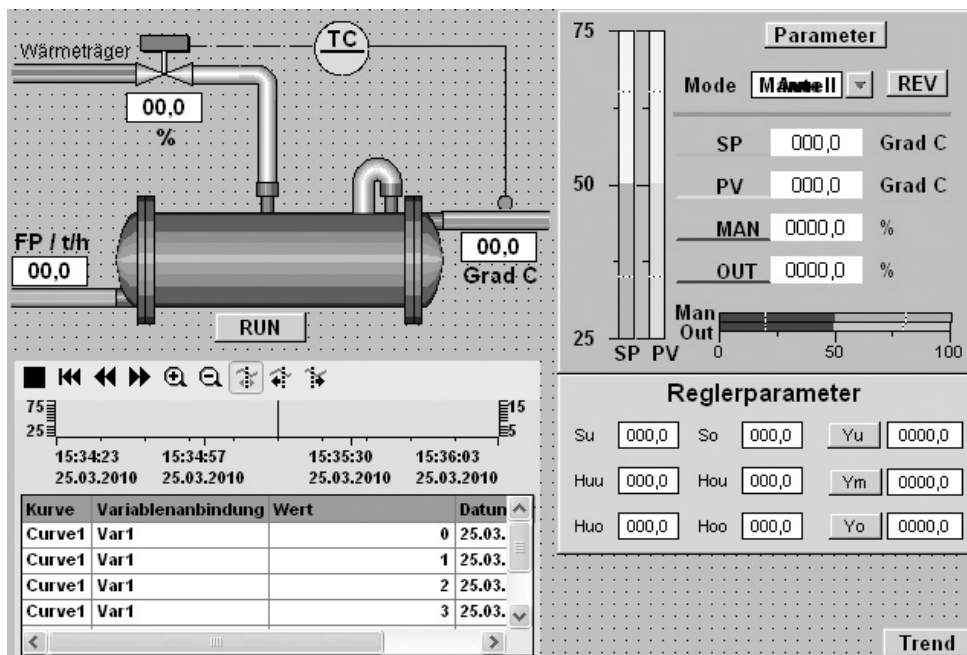


Abbildung 118: Grafikanzeige für das Projekt "Waet3Pkt"

Für die Einstellung des Dreipunktreglers als Zweipunktregler entsprechend der Aufgabenstellung ergibt sich folgendes Bild für die Anzeige und die Parameter:

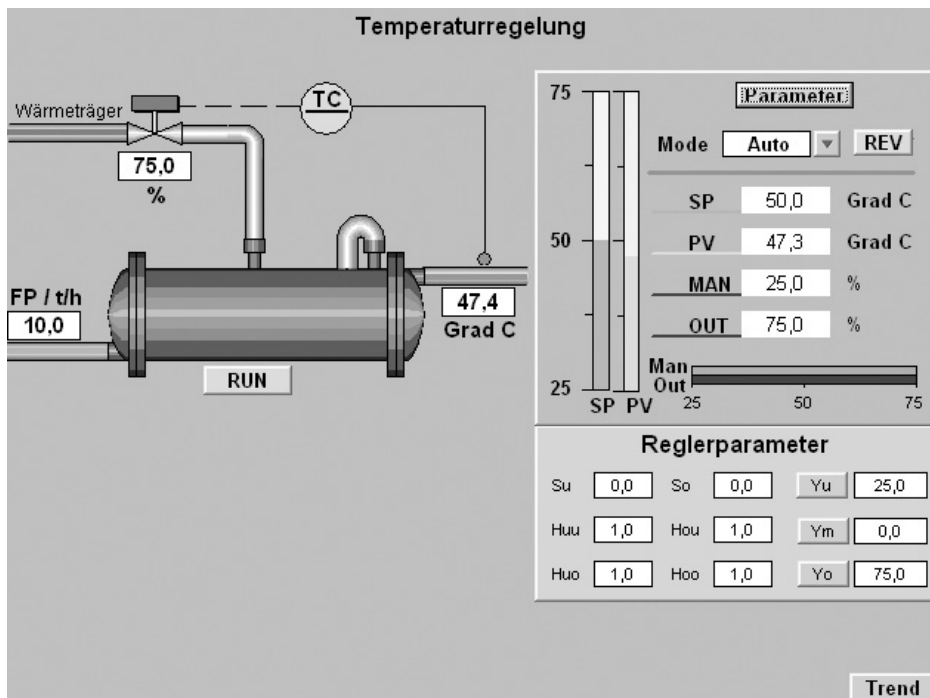


Abbildung 119: Visualisierung mit den eingestellten Parametern

Der Schaltungspunkt des Zweipunktreglers soll beim Sollwert  $SP = 50$  Grad liegen. Damit müssen  $S_u$  und  $S_o$  auf  $S_u = S_o = 0$  eingestellt werden. Die Hysterese ist jeweils 1 Grad. Als untere Stellgröße wird  $Y_u = 25\%$  und für die obere Stellgröße wird  $Y_o = 75\%$  eingestellt. Der Sollwert ist  $SP = 50$  Grad. Wegen des positiven Übertragungsverhaltens der Regelstrecke (bei größer werdender Stellgröße wird die Regelgröße ebenfalls größer) muss der Regler auf  $REV = TRUE$  gestellt werden.

Für  $Y_m$  muss kein spezieller Wert vorgegeben werden, weil für die Regelabweichung kein Intervall zwischen  $S_u + H_{uo} < SP - PV < S_o - H_{ou}$  existiert.

Wenn man die Temperaturregelung ausgehend von  $Y = 25\%$  in die Betriebsart „Auto“ nimmt, steigt die Regelgröße entsprechend der folgenden Abbildung an:

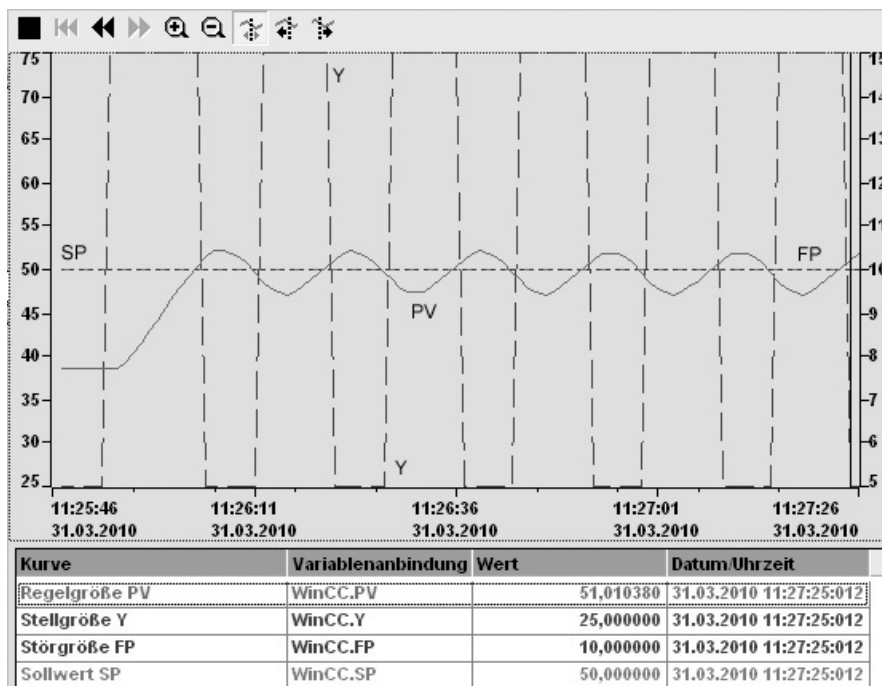


Abbildung 120: Übergang von Handbetrieb in Automatikbetrieb

Man kann erkennen, dass der Regler bei dem Wechsel der Betriebsart „Manuell“ nach „Automatik“ die Stellgröße von 25 % auf 75 % schaltet. Mit der Dynamik einer Verzögerung höherer Ordnung steigt die Temperatur an. Wenn diese den Sollwert um 1 Grad (2 %) überschreitet, schaltet der Regler die Stellgröße wieder zurück auf 25 %. Die Temperatur steigt aber wegen der Verzugszeit weiter an, um dann wieder abzufallen. Wenn die Temperatur den Sollwert um 1 Grad unterschreitet, schaltet der Regler die Stellgröße wieder auf 75 %. Nach einem weiteren Abfall der Temperatur wegen der Verzugszeit steigt die Temperatur dann wieder an.

Der Verlauf der Temperatur ist hier abgerundet, weil die Regelstrecke mit Ausgleich ist und eine Verzögerung höherer Ordnung aufweist. Bei der theoretischen Betrachtung in den Kapitel 1.3 und 1.4 wurde von einer Totzeit und Verzögerung 1. Ordnung ausgegangen, was im Verlauf der Regelgröße zu einem plötzlichen Richtungswechsel der Regelgröße und damit zu einem Knick im Verlauf der Regelgröße führte.

Das Störverhalten der Zweipunktregelung zeigt die nächste Abbildung:

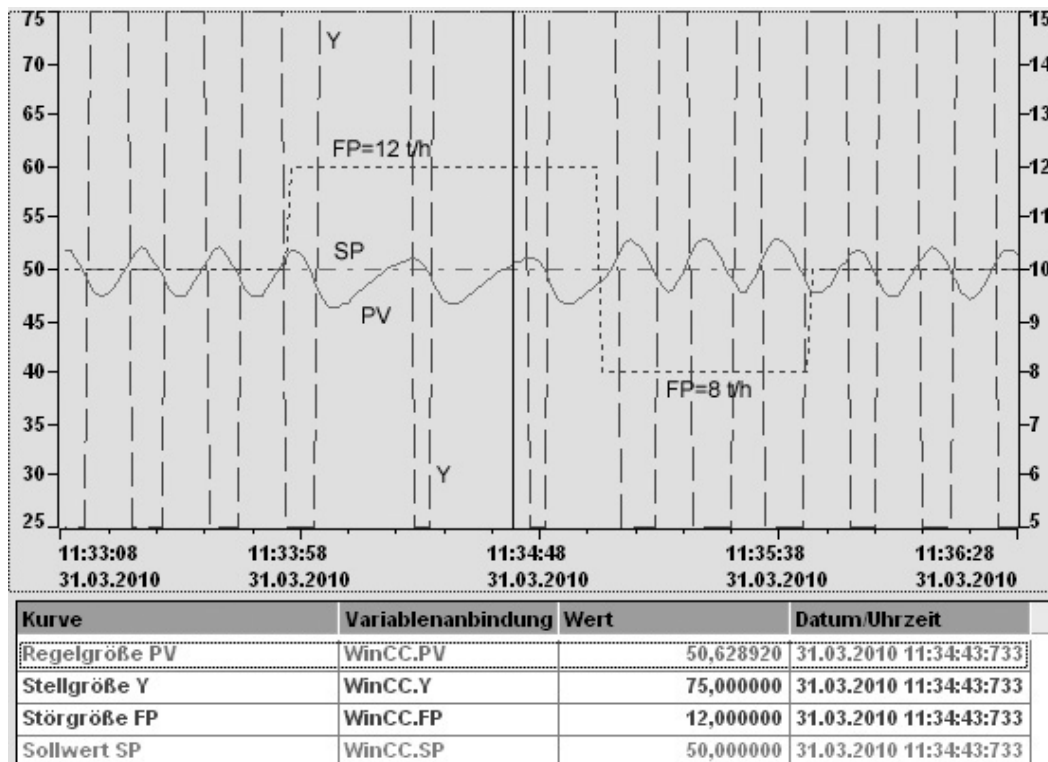


Abbildung 121: Störverhalten der Zweipunktregelung

Man sieht, dass die Regelgröße immer um den Sollwert bei 50 Grad schwingt und die Schaltpunkte jeweils um 1 Grad (2 %) über bzw. unter dem Sollwert liegen. Für  $FP = 12 \text{ t/h}$  liegt der Mittelwert der Arbeitsbewegung deutlich unter dem Sollwert, weil bei dem mit 75 % größten Stellwert und dem Produktstrom  $FP = 12 \text{ t/h}$  die Regelgröße nur auf einen relativ kleinen Endwert von etwas mehr als 52 Grad (siehe Kennliniefeld) geht. Für  $FP = 10 \text{ t/h}$  und  $12 \text{ t/h}$  würden wesentlich höhere Endwerte erreicht werden können, weshalb die Regelgröße bei 50 Grad dann deutlich schneller ansteigt.

Bei einer Veränderung des Sollwertes wird wieder jeweils bei Unter- bzw. Überschreiten dieses Sollwertes geschaltet.

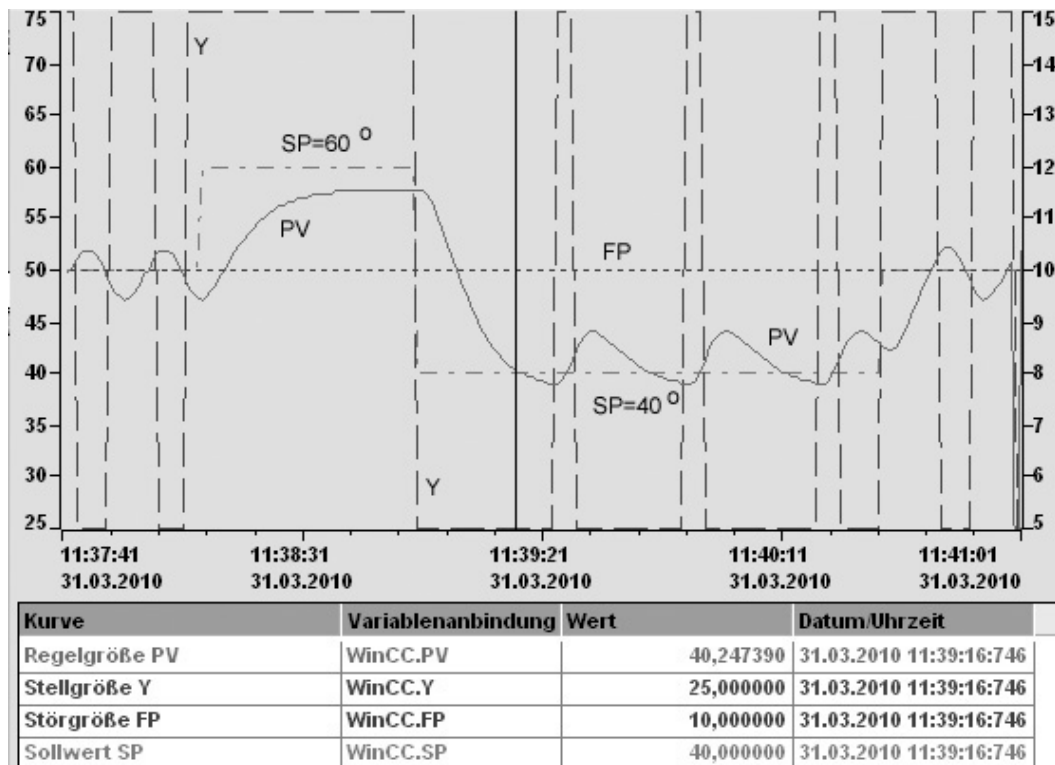


Abbildung 122: Führungsverhalten der Zweipunktregelung

Bei einer Stellgröße von 75 % wird bei  $FP = 10 \text{ t/h}$  eine Temperatur von ca. 58 Grad erreicht. Das ist weniger als die Temperatur, die mit  $SP = 60$  Grad angestrebt wird. Der Zweipunktregler kann also die Sollvorgabe nicht umsetzen, wenn der große Stellwert nicht größer als 75 % wird.

Beim Produktstrom von  $10 \text{ t/h}$  wird mit 25 % als Stellgröße eine Temperatur von ca. 38 Grad erreicht (siehe Kennlinienfeld). Daher fällt die Temperatur in der Nähe von 40 Grad (30 %) sehr langsam. Das führt dazu, dass hier der Mittelwert der Arbeitsbewegung höher ist als der Sollwert.