# Inhaltsverzeichnis

# 1 Anweisung, Berechnungen

**Lernziele:** – Einfaches Programm in der Sprache "ST" schreiben und testen
– Programmorganisationseinheiten (POEs), Variable, Anweisungen, Operatoren, Visualisierung

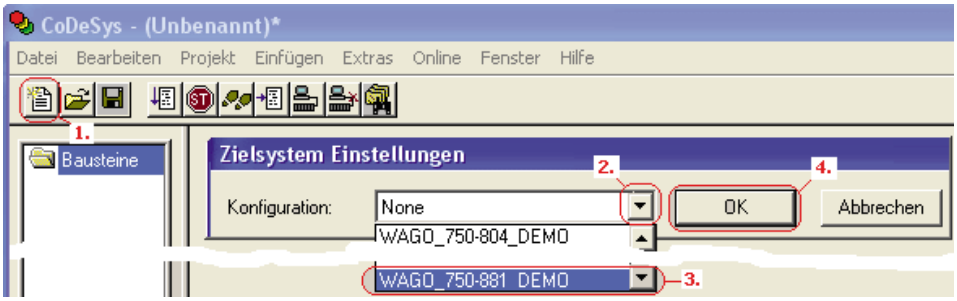
**Aufgabe 1.1** **Leistungsberechnung 1:** Die Scheinleistung, das Produkt aus Strom und Spannung, soll mit Hilfe eines Software-Bausteins berechnet werden. Die Messwerte I und U werden über den PC mit Hilfe einer Visualisierung eingegeben.

**Vollziehen Sie** das gezeigte Beispiel nach.
Entscheiden Sie sich zunächst, in welcher der drei Anwendungen Sie die Aufgabe lösen wollen.

**Vorgehensweise:**

1. **Neues Projekt:**
   In **CoDeSys V2.3**
   **Starten Sie** die Anwendung und **wählen Sie** im Menü Datei–Neu... **Konfigurieren Sie** als Zielsystem z.B. den Controller (SPS) WAGO_750-881_Demo.

Neues Projekt




**In e!COCKPIT (CODESYS V3)**, neues Projekt
**Starten Sie** die Anwendung **e!COCKPIT** und **wählen Sie** in der Produktserie 750… als Gerät z.B. einen PFC100 oder PFC200 und für den Programmbaustein PLC_PRG die **Sprache ST** aus.

**In CODESYS V3 Demo** neues Projekt, Aufgabe 1.1

**Starten Sie** die Anwendung, fügen Sie ein neues Standardprojekt ein, wählen Sie als programmierbares Gerät (Zielsystem) CODESYS Control Win V3, die Soft PLC(SPS).


PLC_PRG

MAIN

zyklisch

**2. Software-Baustein einfügen:**

Der Baustein PLC_PRG wird vom Automatisierungssystem, **freilaufend zyklisch** aufgerufen. Die Anweisungen im PLC_PRG werden sequenziell (nacheinander) abgearbeitet. Andere Zielsystemhersteller bezeichnen diesen Baustein auch als **MAIN** (Hauptprogramm). Nach IEC gibt es drei Programmorganisationseinheiten (POEs bzw. POUs): das Programm, den Funktionsblock (FB) und die Funktion (FC). FBs und FCs lernen Sie ab Kapitel 8 kennen.

**In CoDeSys V2.3** Baustein einfügen, Aufgabe 1.1

Wählen Sie den Baustein **PLC_PRG**, den Typ Programm und stellen Sie die Sprache auf ST, wie das nachfolgende Bild zeigt.

**In e!COCKPIT und in CODESYS V3 Demo** Baustein einfügen, Aufgabe 1.1

In diesen Anwendungen wird der Programmbaustein PLC_PRG automatisch eingefügt.

**3. Variable deklarieren und Anweisungen schreiben:**

Variable deklarieren

REAL

Präfix r

Anweisung

Ausdruck

Zuweisung

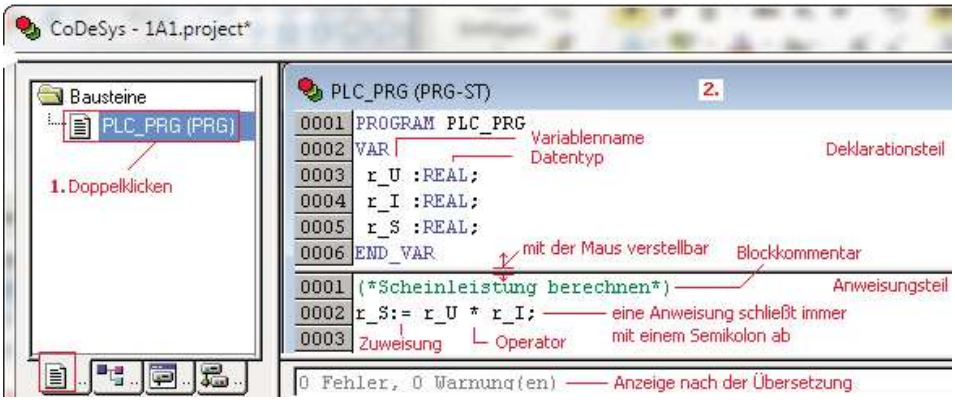
Operator
+ – * /

Kommentar

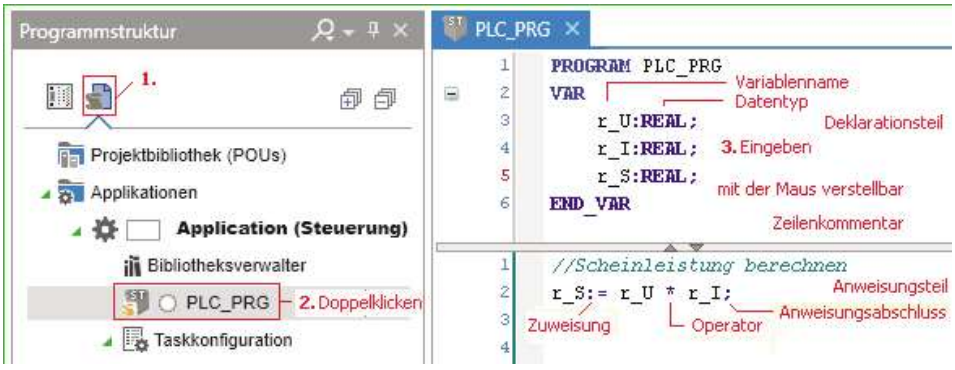
Wie Sie den folgenden Bildern entnehmen können, werden im oberen Teil des PLC_PRG-Fensters, dem Deklarationsteil, die Variablen deklariert, also "Platzhalter" für die Messwerte U, I und das Ergebnis S erstellt. Die Bezeichner, die Variablennamen, wie r_U ... und der Datentyp, hier REAL, werden zwischen den Schlüsselwörtern VAR und END_VAR eingetragen. Das Präfix r zeigt, dass die Variable vom Datentyp REAL ist. REAL heißt, die Variable kann den Wert von $+/-1.18*10^{-38}$... $+/-3.4*10^{38}$ annehmen.

Im unteren Teil des Fensters, dem Anweisungsteil, werden die Anweisungen geschrieben, also die Anweisungen zur Berechnung der Scheinleistung. Das Ergebnis des Ausdrucks r_U * r_I wird der Variablen mit dem Namen r_S zugewiesen. Das Zuweisungszeichen ist :=.
Beachten Sie: Die Wertzuweisung erfolgt immer von rechts nach links. Eine Anweisung wird mit einem Semikolon abgeschlossen. Leerzeichen können zwischen Bezeichner und Operatoren eingefügt werden.
Über die Tastatur kann ein Operator (+, -, *, /) eingefügt werden. Beachten Sie die Prioritäten von Operatoren, Sie finden diese im Anhang Operatoren.
Ein Kommentar unterstützt die Lesbarkeit eines Programms, der Blockkommentar steht zwischen den Zeichen (* und *), der Zeilenkommentar nach den //, dieser ist nur in CODESYS V3 einfügbar.

**In CoDeSys V2.3** Variable deklarieren und Anweisungen schreiben, Aufgabe 1.1




**In e!COCKPIT oder CODESYS V3** Variable deklarieren und Anweisungen schreiben, Aufgabe 1.1



Übersetzen

**4. Programmcode übersetzen:**

Um ein lauffähiges Programm zu erzeugen, muss der Programmcode übersetzt werden. Achten Sie darauf, dass keine Fehlermeldungen ausgegeben werden. Korrigieren Sie eventuell angezeigte Fehler, z. B. ein fehlender Strichpunkt.

**In CoDeSys V2.3:** Wählen Sie im Menü Projekt–Alles übersetzen.

**In e!COCKPIT:** Wählen Sie im Menü Programm–Übersetzen.

**In CODESYS V3 Demo:** Wählen Sie im Menü Erstellen–Übersetzen.

**5. Ein-/Ausgabe, Visualisierung:**

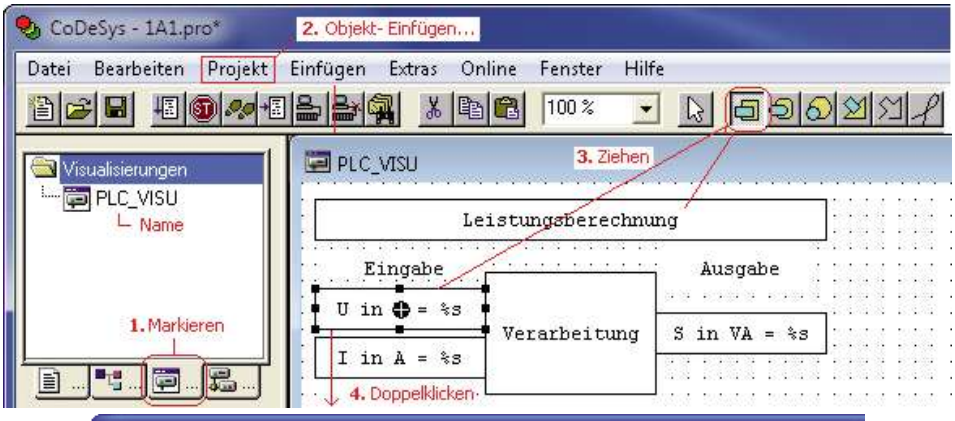
Mit Hilfe einer Visualisierung wird eine Ein- und Ausgabemöglichkeit der Variablen geschaffen. Sie können so auch das Programm leichter testen.

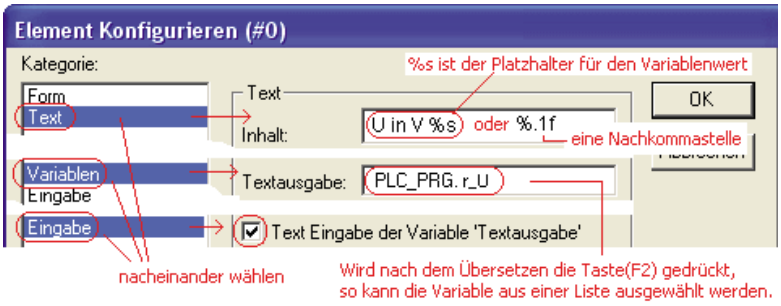
Visualisierung

**In CoDeSys V2.3** Vorgehensweise, um die Visualisierung zu erstellen, Aufgabe 1.1:

1. Im Projektbaum Visualisierung auswählen, siehe folgendes Bild.
2. Visualisierungsfenster einfügen–Menü Projekt–Objekt–Einfügen...Name, z. B. PLC_VISU vergeben. Beachten Sie auch dazu das Kapitel 17.
3. Grafik-Elemente einfügen, z. B. das Eingaberechteck für den Variablenwert r_U.
4. Doppelklicken Sie das eingefügte Eingaberechteck und konfigurieren Sie dieses. Kopien können Sie für die weiteren Ein-/Ausgaberechtecke einfügen und umkonfigurieren, z. B. für den Wert der Variablen `r_I` usw.

Visualisierungseditor V2.3



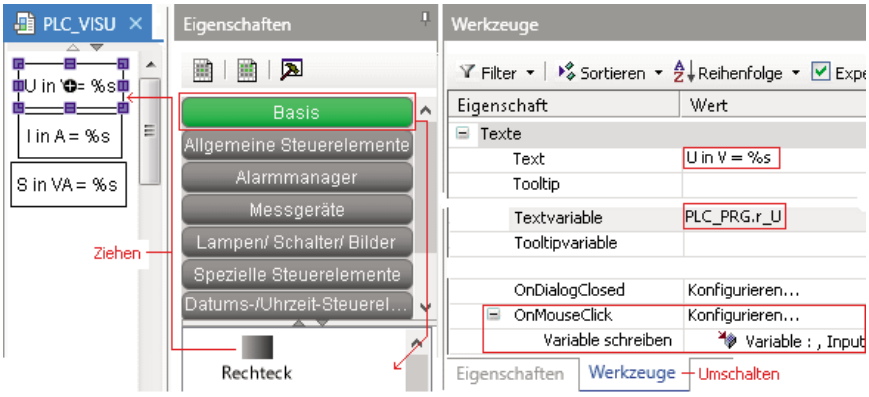




Siehe auch CODESYS-Menü Hilfe–Inhalt–Index–Visualisierung.

**In e!COCKPIT oder CODESYS V3** Visualisierung erstellen, Aufgabe 1.1
Vorgehensweise:

1. Fügen Sie bei markierter Application in der Programmstruktur über das Kontextmenü das Visualisierungselement ein.
2. Wählen Sie für die Eingabe des Variablenwertes von r_U im Werkzeugfenster unter Basis das Rechteck aus und ziehen es in das Arbeitsfenster.
3. Konfigurieren Sie das Eingaberechteck. Sie können das konfigurierte Rechteck kopieren und einfügen und umkonfigurieren für die Variablenwerte r_I und r_S.

Visualisierungs-editor V3



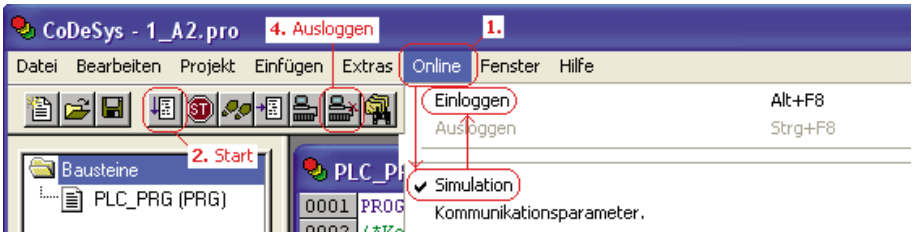

**6. a) Testen mit der PC-Simulation und der Visualisierung:**

Da die Programmorganisationseinheit (POE bzw. POU) PLC_PRG freilaufend zyklisch vom Betriebssystem aufgerufen wird, wird auch der Wert der Variablen r_S immer wieder neu berechnet.

**In CoDeSys-Fenster V2.3** testen mit der Simulation und der Visualisierung, Aufgabe 1.1
Vorgehensweise:

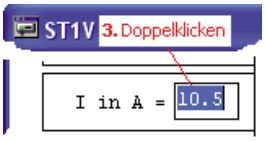
1. Menü **Online**–Simulation mit Häkchen und im gleichen Menü **Einloggen**.
2. Menü Online–**Start**. Die Anweisungen im Programm werden jetzt abgearbeitet.

Online Simulation Ein-/Aus-loggen




3. Im Visualisierungsfenster können die Werte der Variablen U, I durch Anklicken verändert werden.
   Beachten Sie:
   An Stelle eines Kommas muss ein **Punkt** eingegeben werden.

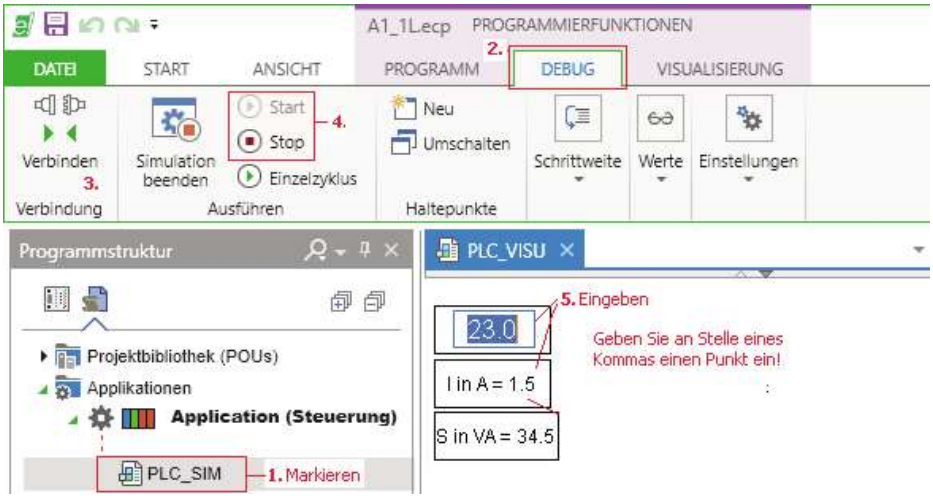
Punkt




4. Um den Code weiter bearbeiten zu können, müssen Sie über Menü Online– wieder ausloggen.

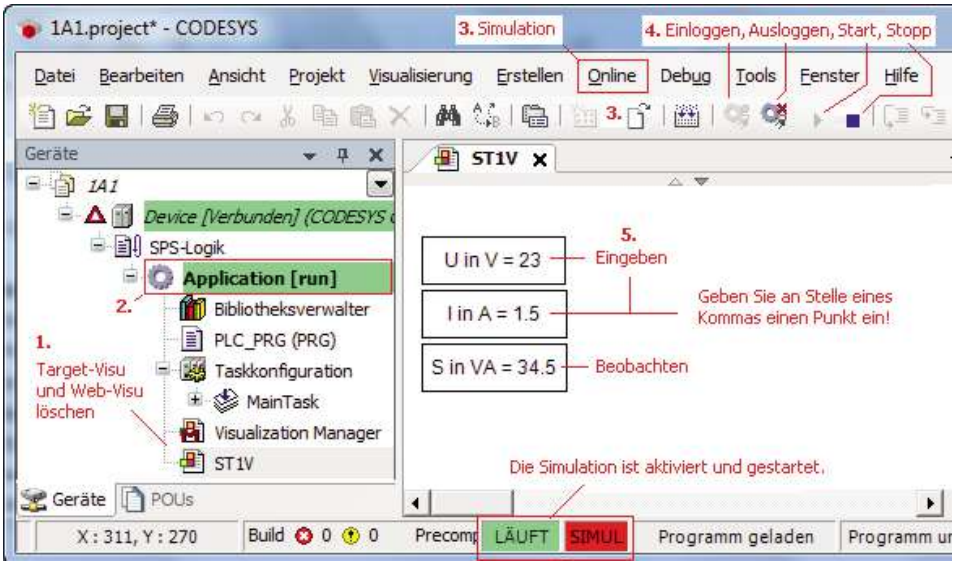
**In e!COCKPIT (CODESYS V3)** testen mit der Simulation und der Visualisierung, Aufgabe 1.1

Markieren Sie die Applikation, wählen Sie im Menü DEBUG- Simulation starten und Start. Geben Sie in der Visualisierung die Werte für U und I ein, wie dies das nachfolgende Bild zeigt. Beobachten Sie die Werte von S.




**In CODESYS V3 Demo** testen mit der Simulation und der Visualisierung, Aufgabe 1.1

Löschen Sie in der Projektstruktur die Objekte Target-Visualisierung und Web-Visualisierung. Markieren Sie die Applikation, aktivieren Sie im Menü Online die Simulation, loggen Sie ein und dann im Menü Debug- Start. Geben Sie in der Visualisierung die Werte für U und I ein, wie dies das nachfolgende Bild zeigt. Beobachten Sie die Werte von S.

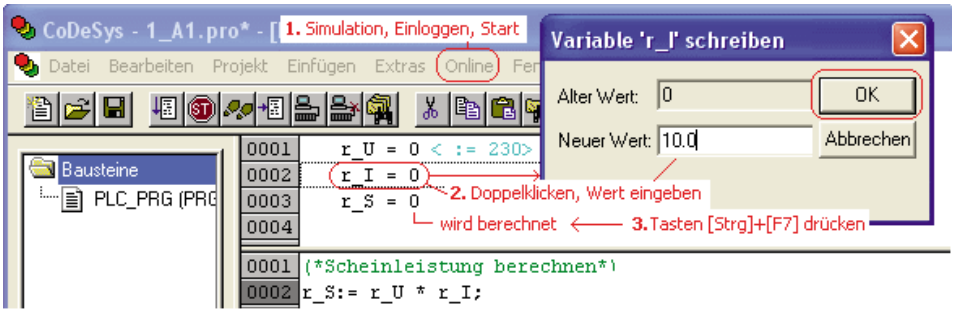


**b) Testen nur mit der PC-Simulation:**

**Testen nur mit der Simulation – In CoDeSys V2.3** testen, Aufgabe 1.1

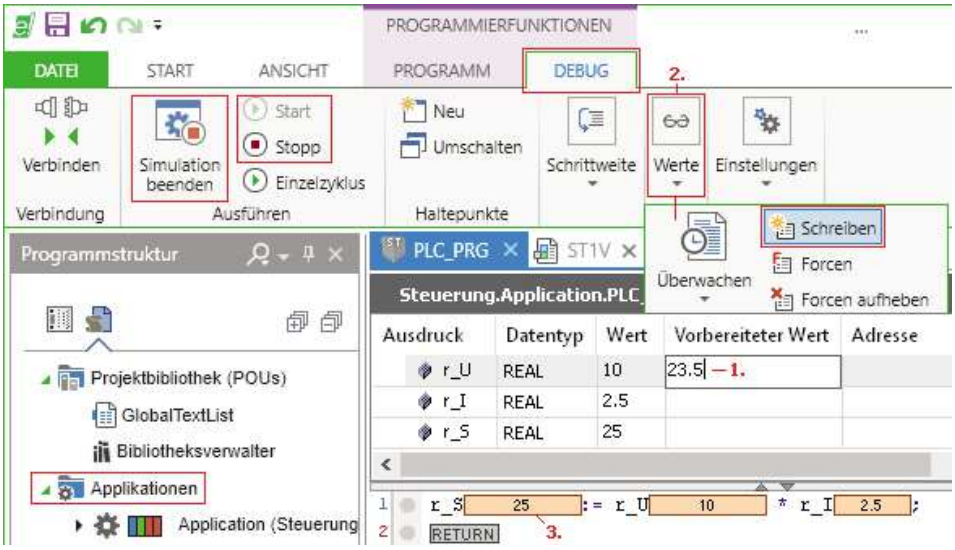
Doppelklicken Sie nach dem Aktivieren der Simulation, dem Einloggen und Start im Deklarationsteil die Variable, geben Sie den Wert ein und wählen Sie im Menü Online–Werte schreiben. Überprüfen Sie das Ergebnis von `r_S`.

Werte schreiben




**In e!COCKPIT** testen – nur mit der Simulation, Aufgabe 1.1

1. Geben Sie nach Simulation starten den Wert in die Spalte – Vorbereiteter Wert – ein.
2. Wählen Sie im Menü DEBUG–Werte schreiben.
3. Überprüfen Sie das Ergebnis.




**In CODESYS V3 Demo** testen – nur mit der Simulation, Aufgabe 1.1

Aktivieren Sie im Menü Online die Simulation und wählen Sie dann im Menü Online–Einloggen. Wählen Sie im Menü DEBUG–Start, geben Sie in der Spalte „Vorbereiteter Wert" Werte ein (siehe Bild oben) und im Menü DEBUG–Werte schreiben. Überprüfen Sie das Ergebnis in `r_S`.

## Aufgabe 1.2 Leistungsberechnung 2

Die Scheinleistung S, die Blindleistung Q sowie der Leistungsfaktor λ sollen mit Hilfe eines Software-Bausteins berechnet werden.
Die Messwerte U, I und die Wirkleistung P werden über den PC mit Hilfe einer Visualisierung eingegeben, S, Q und λ werden berechnet und angezeigt.

$$S = U * I; \quad Q = \sqrt{S^2 - P^2}; \quad \lambda = \frac{P}{S}$$

Da nicht für alle Rechenoperationen Operatoren wie +, -, *, /... zur Verfügung stehen, werden **IEC-Operatoren** bzw. **Standardfunktionen** verwendet. Sie finden eine Auflistung im Anhang.
Benutzen Sie zur Berechnung der Wurzel die Funktion SQRT(...) und für das Quadrat die Funktion EXPT(...,2).
Siehe auch CODESYS–Hilfe–Index ... SQRT bzw. EXPT.

**Vorgehensweise:**

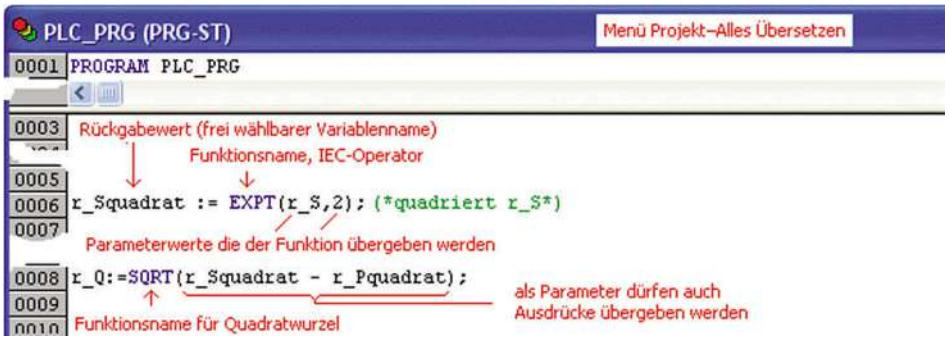
1. Benutzen Sie das vorhergehende Projekt als Vorlage.
2. **Deklarieren Sie** die neuen Variablen und **ergänzen Sie** den Code. Bei mehreren Anweisungen ist es sinnvoll, den Code öfter zwischendurch zu **übersetzen**, um **Syntax-Fehler** (Abweichungen von den Sprachvereinbarungen) leichter zu finden. Beim Übersetzen wird der Maschinencode für das Zielsystem, z. B. den Controller 750-881 oder 750-8100, erzeugt.

Übersetzen

Syntax-Fehler

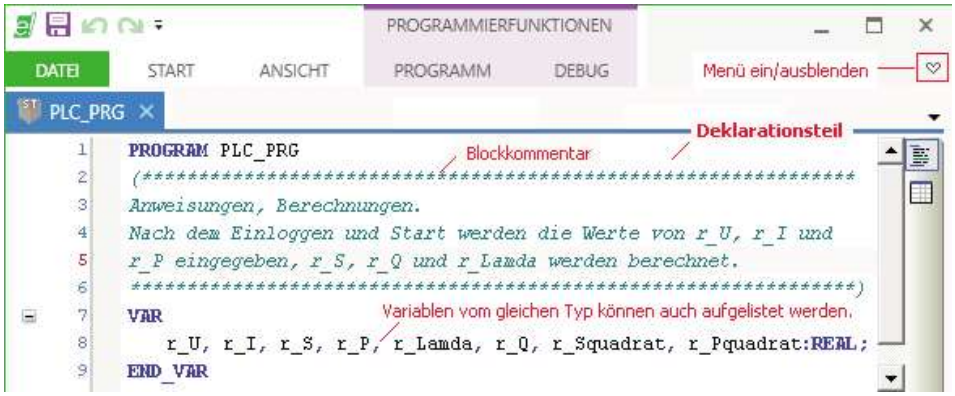
3. **Ergänzen Sie** die Visualisierung mit den weiteren Ein- und Ausgabemöglichkeiten.
4. **Testen Sie** die Funktion des Programms, siehe Aufgabe 1.1.
   Beispiel: r_U = 230; r_I = 10,5; r_P = 1500 ergibt r_S = 2415; r_lamda = 0,621...;
   r_Q = 1892,677

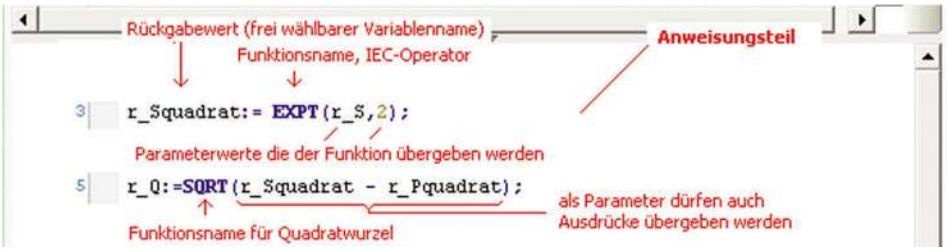
**In CoDeSys V2.3,** Aufgabe 1.2




SQRT()
EXPT(,)

**In e!COCKPIT,** Aufgabe 1.2

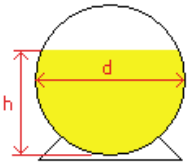





**In CODESYS V3,** Aufgabe 1.2

Geben Sie den Code wie im vorhergehenden Bild ein und wählen Sie im Menü Erstellen–Übersetzen.

## Übung 1.1 Volumenberechnung




Das Füllvolumen eines kugelförmigen Tanks, der mit einer Flüssigkeit gefüllt ist, soll berechnet werden. Das Füllvolumen V ist abhängig von der Füllhöhe h und dem Durchmesser d.

$$V = \pi * h^2 * (d/2 - h/3);$$

Bei d = 2,5 m und h = 1,5 m -> V = 5,29875 $m^3$

**Stichpunkte** Nach der Bearbeitung des Kapitels sollten die folgenden Stichpunkte für Sie verständlich sein:

Neues Projekt, PLC_PRG, MAIN, zyklisch, Variable deklarieren, REAL, Präfix, Anweisung, Ausdruck, Operator, Kommentar, Visualisierung, Online-Simulation, SQRT(), EXPT(,), übersetzen, Syntaxfehler

# 1. Selbsttest – Anweisung, Berechnungen

Wählen Sie die richtigen Ergänzungen bzw. Aussagen aus.

---

1. Ändert sich ein Eingabewert, so wird der Ausgabewert sofort neu berechnet, da der Programmbaustein PLC_PRG

   - ☐ bei einer Änderung eines Eingabewertes neu aufgerufen wird.
   - ☐ nur einmal aufgerufen wird.
   - ☐ zyklisch aufgerufen wird.
   - ☐ sequenziell aufgerufen wird.

---

2. Das Zuweisungszeichen im Programm ist das

   ☐ = ☐ ; ☐ (**) ☐ :=

---

3. Eine Variable ist ein Platzhalter für Werte; ihr Wertebereich und somit die Speichergröße wird durch

   - ☐ den Variablennamen festgelegt.
   - ☐ den Datentyp festgelegt.
   - ☐ den Operator festgelegt.
   - ☐ die Anweisung festgelegt.

---

4. `r_Q:= SQRT(r_Squadrat - r_Pquadreat);`

   - ☐ Der Wert von r_Q wird der Variablen SQRT zugewiesen.
   - ☐ Der Wert der Quadratwurzel des Ausdrucks wird der Variablen r_Q zugewiesen.
   - ☐ Die Differenz von `r_Squadrat - r_Pquadrat` wird quadriert und in die Variable r_Q geschrieben.
   - ☐ Die Anweisung ist nicht übersetzbar, da ein Syntaxfehler vorliegt.

---

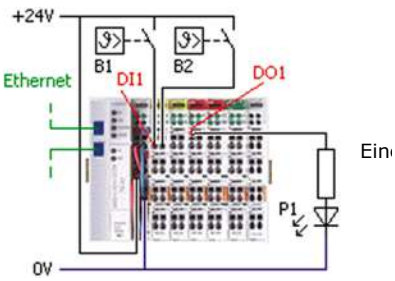
Einen Selbsttest zu diesem Kapitel finden Sie auf InfoClick. Doppelklicken Sie den Link -> Aufgaben, Übungen, Lösungen und dann den Link Selbsttest.

# 2 Boolesche Operationen, Steuerungskonfiguration

**Lernziele:**
- Anweisungen mit logischen Operationen schreiben, Datentyp BOOL
- Steuerungskonfiguration, Zugriff auf Ein- und Ausgänge, Schlüsselwort AT
- Von der Funktionstabelle und vom KV-Diagramm zur Anweisung

## Aufgabe 2.1 Antivalenz-Funktion

Es sind zwei Temperatursensoren B1 und B2 in einem Behälter eingebaut und an der SPS angeschlossen. Sind beide Sensoren funktionsfähig, so haben bei Übertemperatur beide Sensoren angesprochen, d.h., deren Kontakte sind geschlossen. Liegt die Behältertemperatur unter dem Grenzwert, so sind beide Kontakte offen. rote Leuchtdiode (LED), am Ausgang der SPS, soll anzeigen, wenn einer der beiden Sensoren defekt ist.
**Vollziehen Sie** das gezeigte Beispiel nach.
**Erstellen Sie** dazu ein neues Projekt (Kapitel 1).




Ein

### Prozessabbild

Die Zustände der Eingänge (logisch 1 bzw. 0) werden vom Betriebssystem des Controllers zyklisch in das Prozessabbild der Eingänge (PAE) geschrieben. Die Zustände der Ausgänge werden aus dem Prozessabbild der Ausgänge (PAA) gelesen. Die Prozessabbilder sind Speicherbereiche im Controller. Die Werte im PAE sind über die Adressen %I..., z. B. %IX2.0 vom Programm lesbar, die Werte im PAA werden über die Adressen %Q..., z. B. %QX0.0 vom Programm beschrieben.

PAE

PAA

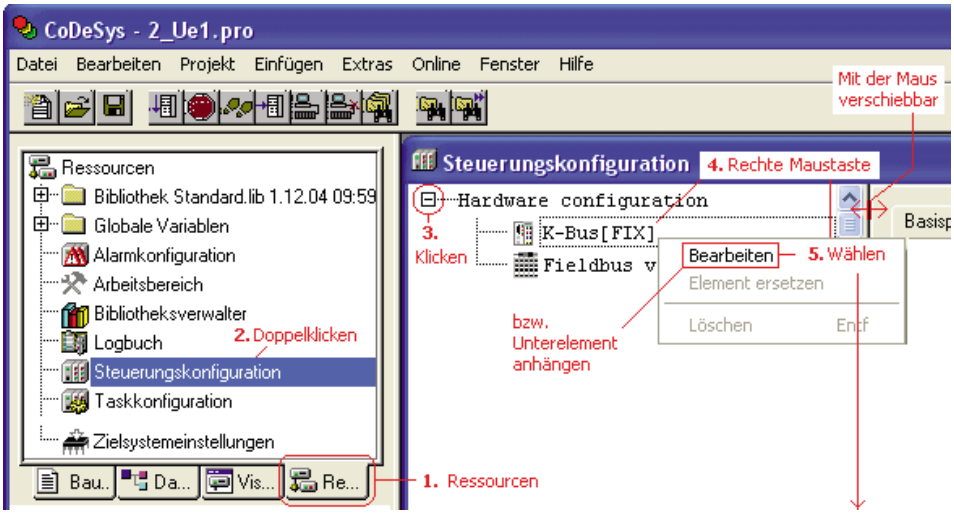
### Vorgehensweise:

1. **Welche Ein- und Ausgänge (I/Os) stehen bei der SPS bzw. dem Controller zur Verfügung?**

   Konfigurieren Sie die Steuerung mit Digital-Inputs (Eingänge) und -Outputs (Ausgänge) wie im Bild unten beschrieben. Handbuch-Auszüge der Baugruppen finden Sie im Anhang.

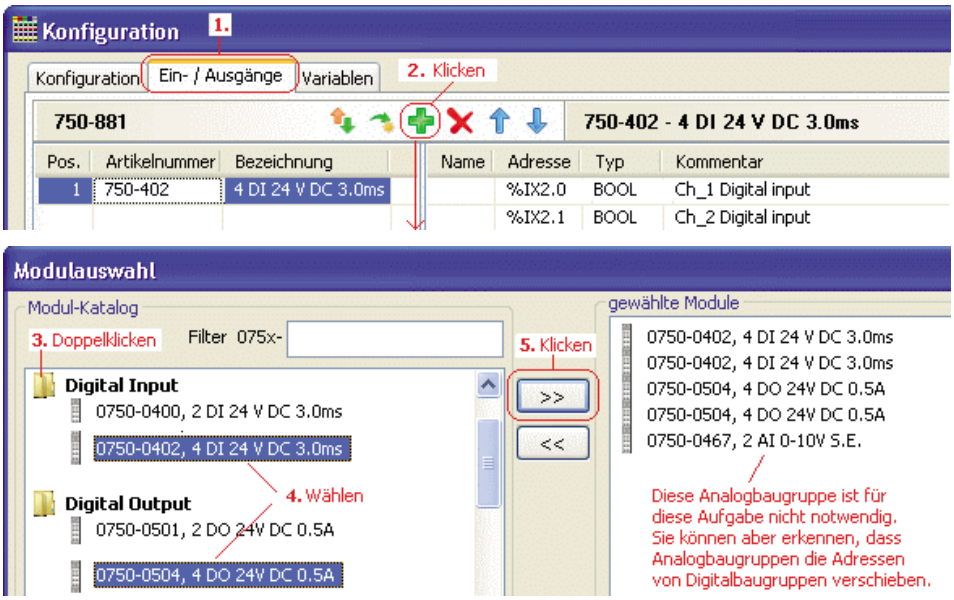
I/Os

**In CoDeSys V2.3** Steuerung konfigurieren, Aufgabe 2.1




Ressourcen

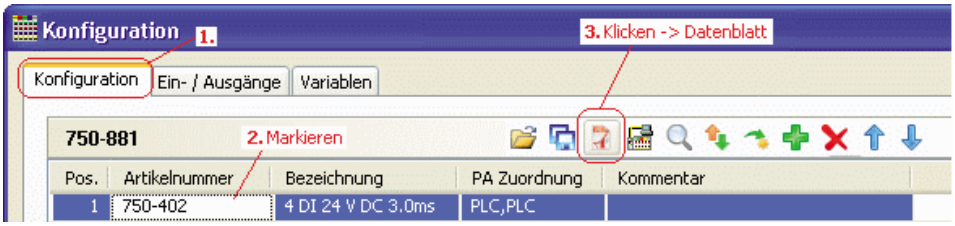
Steuerungs-konfiguration




Digital Input

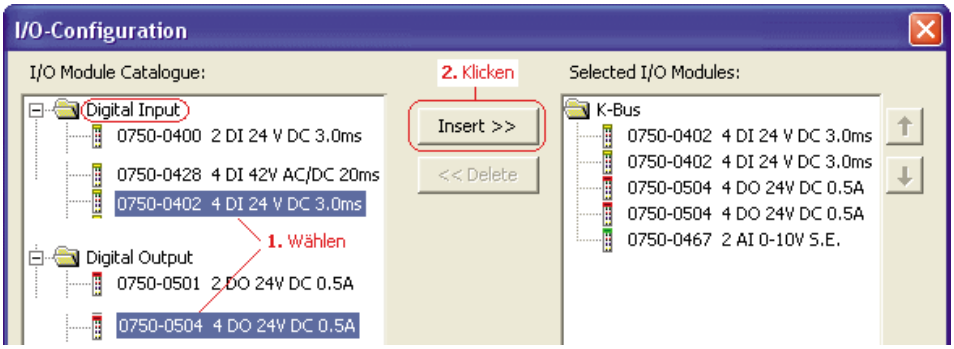
Digital Output

Sie können auch aus dem I/O-Konfigurator auf die **Datenblätter** der Baugruppen zugreifen.




Datenblatt

Falls Sie eine ältere WAGO-I/O-Pro-Version verwenden, wird die I/O-Konfiguration etwas anders dargestellt (Bild unten).



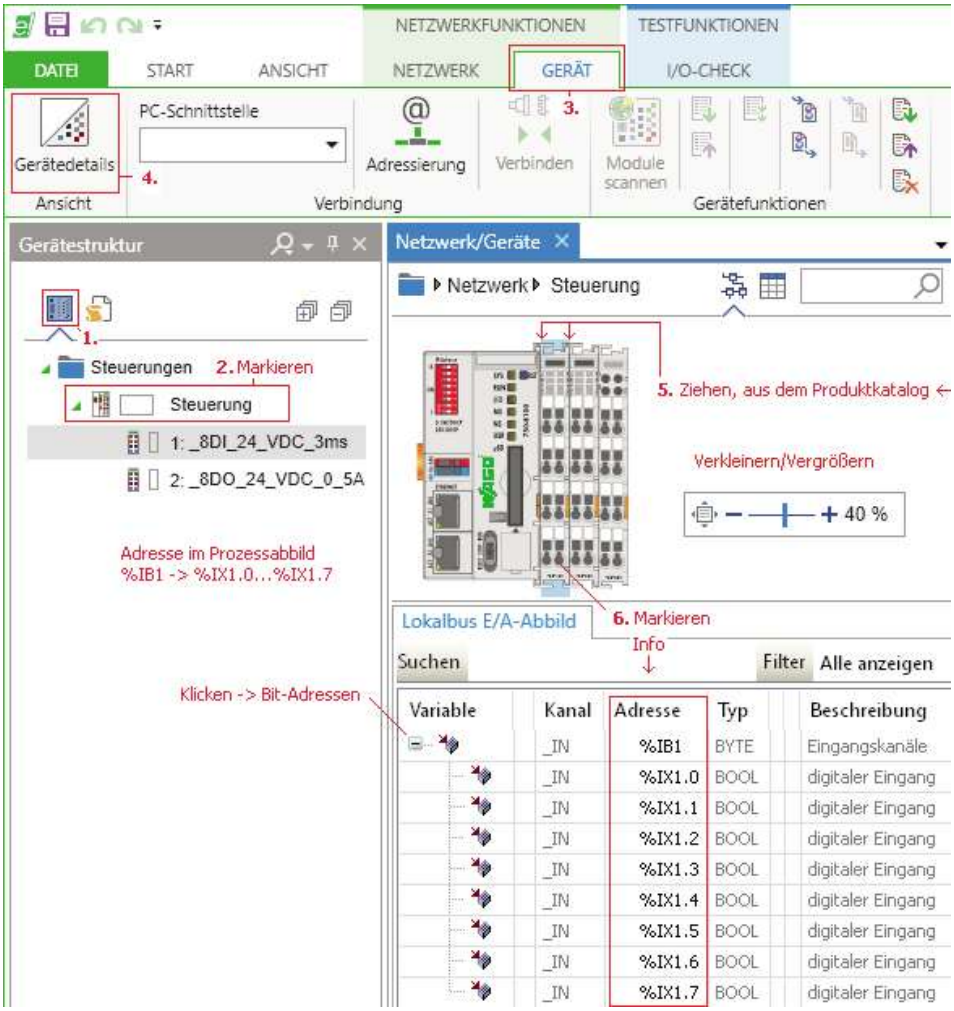

I/O-Konfiguration

Die IEC-Adresse eines Ein- oder Ausgangs ist in der Steuerungskonfiguration ablesbar. Abhängig von den Baugruppen belegen diese bestimmte Adressen.

Anmerkung:
Mit Hilfe der Software WAGO-I/O-CHECK und einer Verbindung zur Steuerung können Sie online die Module ermitteln und in Ihr Projekt übernehmen.

**In e!COCKPIT** Steuerung konfigurieren, Aufgabe 2.1

Markieren Sie die Steuerung und wählen Sie im Menü Geräte-Gerätedetails. Ziehen Sie mit der Maus aus dem Produktkatalog I/O-Systeme- 750- Ein- und Ausgangsklemmen zum Controller. Öffnen Sie dazu im Produktkatalog den jeweiligen Ordner.




In der Geräteansicht und bei markierter Steuerung können Sie auch bei verbundenem Gerät die Konfiguration online ermitteln. Wählen Sie im Menü Gerät-Verbinden und Module ermitteln.

**In CODESYS V3 Demo** Steuerung konfigurieren, Aufgabe 2.1

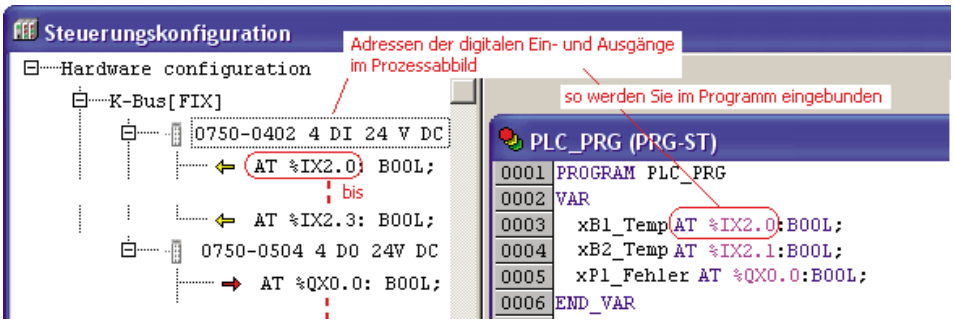
In dieser Demo von der Firma 3S können Sie keine I/O-Baugruppen konfigurieren. Geben Sie das Programm ein und ignorieren Sie beim Übersetzen die Warnung – fehlende I/Os.

2. **Wie werden die Ein-/Ausgänge im Programm eingebunden?**
   **Wie werden die Ein-/Ausgänge den Variablen zugeordnet?**

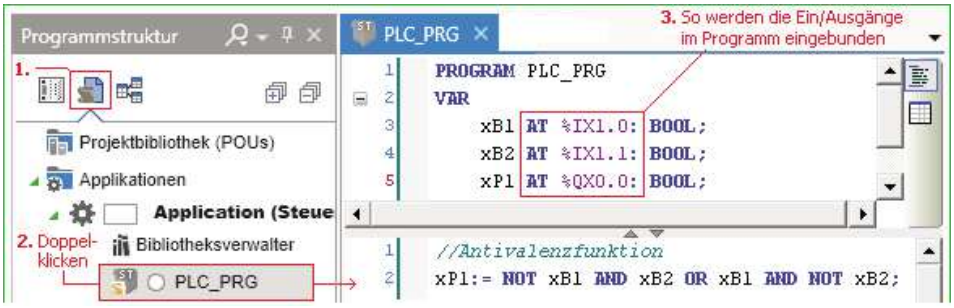
   Mit dem Schlüsselwort `AT` wird der Variablen bei der Deklaration ein absoluter Speicherplatz zugeordnet. Die Adresse, z.B. %IX2.0 oder %IX1.0, entnehmen Sie der Steuerungs- bzw. Gerätekonfiguration.

**In CoDeSys V2.3** Variablendeklaration, Aufgabe 2.1

AT

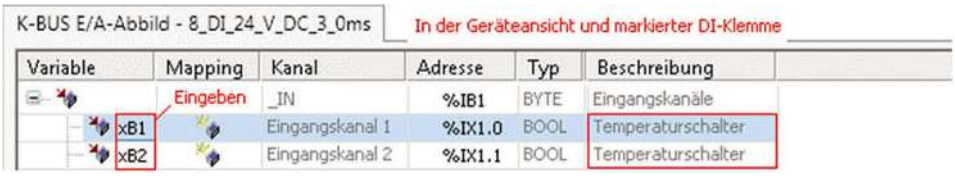



**In e!COCKPIT oder CODESYS V3** Variablendeklaration und Programmcode, Aufgabe 2.1




Mappen

**Mappen:** Sie können auch die Variable mappen, wenn diese auf I/O-Klemmen zugreifen. Die Deklaration dieser Variablen im Deklarationsteil von PLC_PRG kann dann entfallen.




AND, OR, NOT

**3. Wie sieht der Code in ST aus? Wie lautet die Anweisung?**

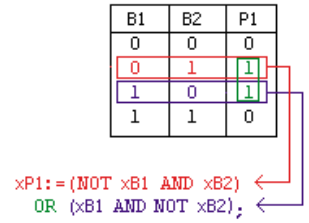
Mit Hilfe einer Funktionstabelle wird die Anweisung ermittelt (siehe Bild rechts). Es können die Operatoren **NOT, AND, OR** verwendet werden.

BOOL

Beachten Sie:

Der Datentyp der Variablen ist **BOOL**, da ihr Wert nur TRUE (wahr, logisch 1) oder FALSE (falsch, logisch 0) sein kann. Die Bezeichner, die Namen der Variablen, sollten den Präfix x bekommen, z. B. `xB1`... `xP1`.

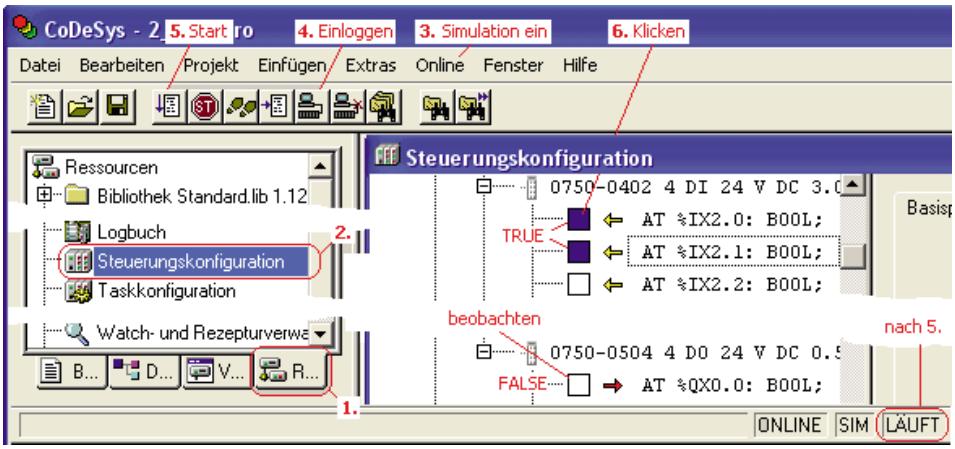
Von der Funktionstabelle zur Anweisung



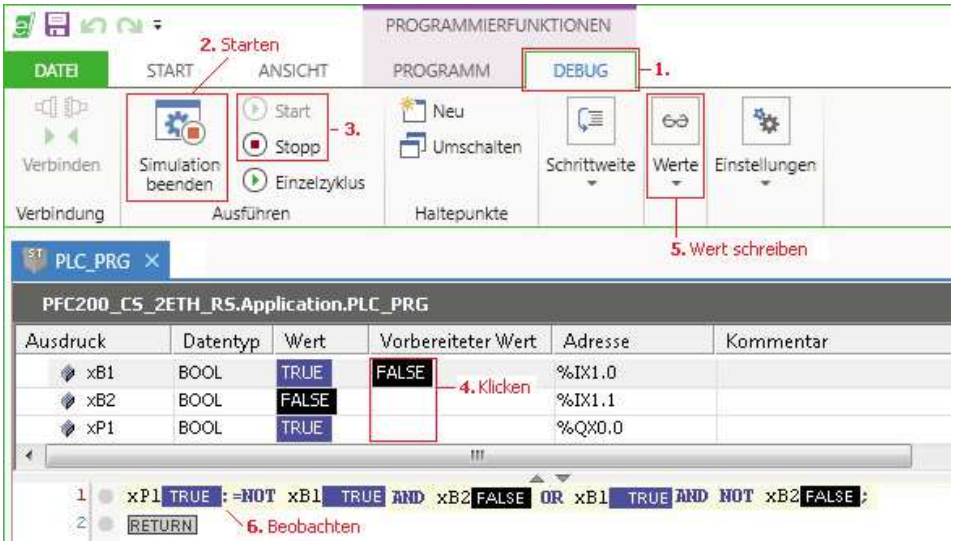

Nachdem Sie die Variablen deklariert haben, geben Sie im Anweisungsteil den Code ein. Returns, Leerzeichen sowie Tabulatoren können Sie zur Strukturierung Ihres Codes einsetzen. Diese Zeichen werden beim Übersetzen nicht gelesen. Den Programmcode zeigt das Bild oben auf dieser Seite. Da die Sprache genormt ist, ist er in allen Entwicklungsumgebungen gleich.

Test

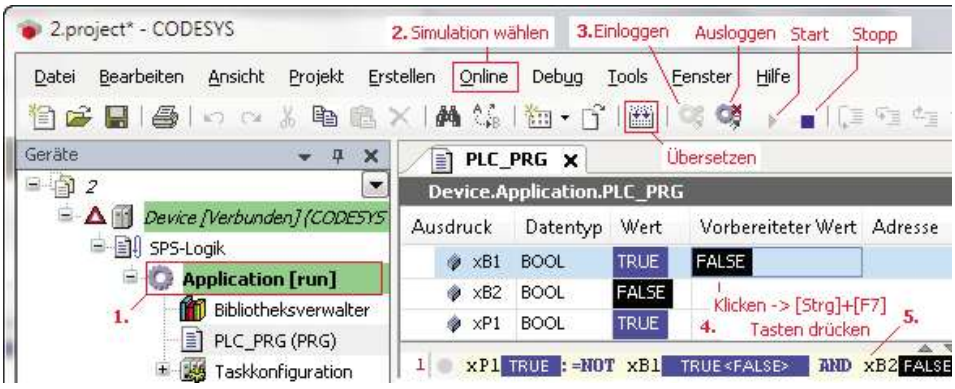
**4.** Testen Sie das Programm **CoDeSys V2.3** – Testen Sie mit der Simulation, Aufgabe 2.1, beachten Sie dabei die Reihenfolge.




**In e!COCKPIT** testen mit der Simulation, Aufgabe 2.1




**In CODESYS V3 DEMO** testen mit der Simulation, Aufgabe 2.1



Falls Sie den Variablen Ein- bzw. Ausgangsadressen zugeordnet haben, so ignorieren Sie nach dem Übersetzen die Meldungen.

5. **Übertragen Sie** Ihr getestetes Programm in den Controller, falls Ihnen einer zur Verfügung steht. Wie Sie die Kommunikation über das Ethernet konfigurieren können, können Sie im Anhang nachlesen.

**Übung 2.1** **Äquivalenz-Funktion** (EQ, equal)

EQ

Eine grüne LED (P2) soll am zweiten Ausgang (%QX0.1) anzeigen, dass kein Sensor defekt ist.

1. **Ergänzen Sie** das Programm, füllen Sie zunächst die Tabelle aus und entwickeln Sie daraus die Anweisung.

| B1 | B2 | P2 |
|---|---|---|
| 0 | 0 | ... |
| 0 | ... | ... |
| 1 | ... | ... |
| 1 | 1 | ... |

```
xP2 := .....
```

2. **Überprüfen Sie** die Funktion. Siehe Beispiel oben.

**Übung 2.2** **XOR**

XOR

**Vereinfachen Sie** die Aufgabe 2.1 mit dem XOR-Operator. Schreiben Sie die Anweisung auf. Siehe CODESYS-Menü Hilfe–Suchen–XOR eingeben...

```
xP1_Uebertemp:= .....
```

**Übung 2.3** **Funktion 2 aus 3**

| B1 | B2 | B3 | P1 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | ... |
| 0 | 1 | 0 | ... |
| 0 | 1 | 1 | **1** |
| 1 | 0 | 0 | ... |
| 1 | 0 | 1 | ... |
| 1 | 1 | 0 | ... |
| 1 | 1 | 1 | **1** |

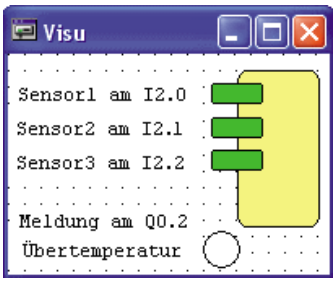
In einem Behälter soll die Temperatur überwacht werden. Es sind drei Temperaturschalter (B1 ... B3) eingebaut. Die Signale der Temperaturschalter sollen so verknüpft werden, dass eine Meldeleuchte (P1) "Übertemperatur" anzeigt, wenn zwei oder alle drei Temperaturschalter angesprochen haben.

**Schreiben Sie** die Gleichung mit Hilfe der Funktionstabelle auf.
**Ergänzen Sie** zunächst die Einträge in der Tabelle.

```
P1 := .....
   OR .....
   OR .....
   OR .....
```

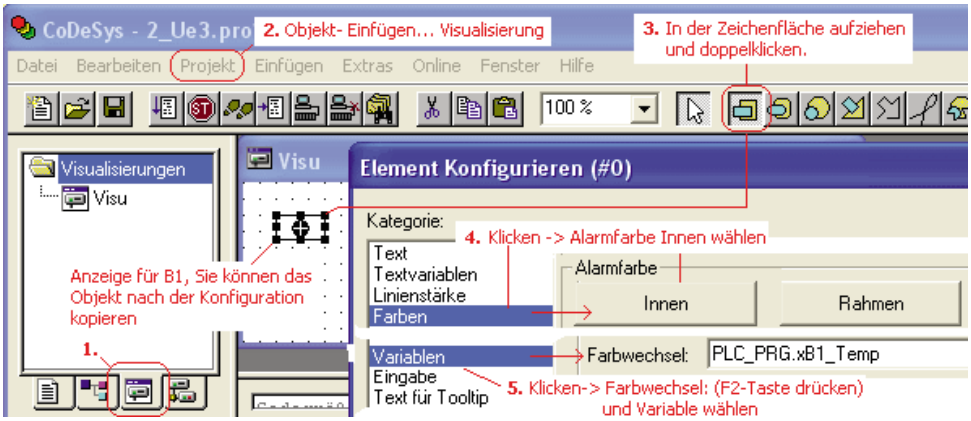
**Geben Sie** den Code ein.

**Erstellen Sie** eine Visualisierung – siehe folgendes Bild – und testen Sie die Funktion.



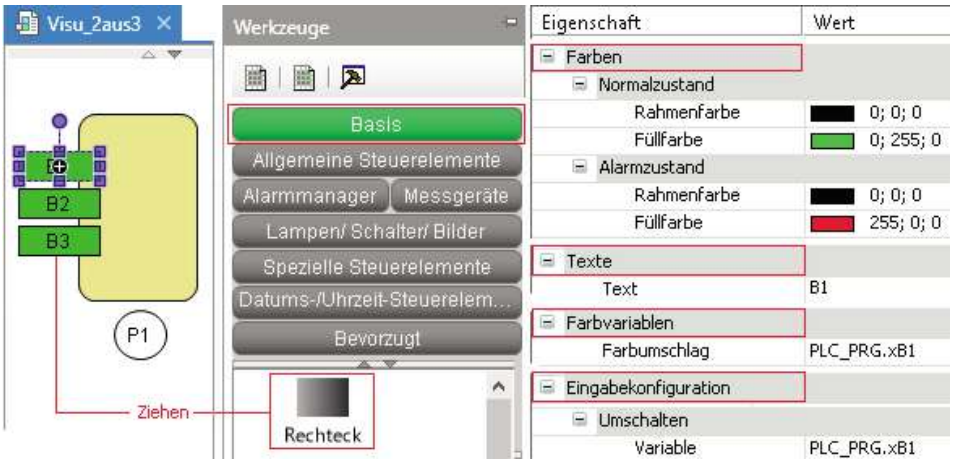
**In CoDeSys V2.3** Visualisierung, Übung 2.3

Visualisierung



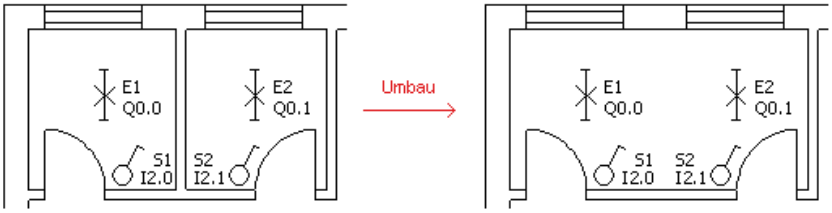

**In e!COCKPIT oder CODESYS V3** Visualisierung, Übung 2.3

Fügen Sie bei markierter Application eine Visualisierung ein.




## Übung 2.4 Wechselschaltung

Die Gebäudeinstallation beinhaltet einen I/O-Controller zur Gebäudeautomatisierung.
Sie erhalten den Auftrag, eine bestehende Beleuchtungsanlage für zwei unabhängige Räume umzuprogrammieren, da die Zwischenmauer entfernt wird. Beide Leuchten sollen dann von S1 und S2 schaltbar sein (Wechselschaltung).



| S2 | S1 | E2 = E1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | ... |
| 1 | 0 | ... |
| 1 | 1 | ... |

**Entwerfen Sie** mit Hilfe der Funktionstabelle die Anweisung für das Programm.

E1 := .....

E2 := E1; Testen Sie die Funktion.

## Aufgabe 2.2 KV-Diagramm, Funktion 2 aus 3

Funktions-tabelle -> Anweisung

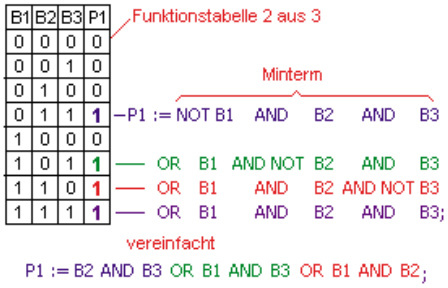
Bei Verknüpfungssteuerungen ohne Speicherverhalten kann mit Hilfe einer **Funktionstabelle** die notwendige Anweisung ermittelt werden.

Bei der disjunktiven Normalform (DNF) werden:

- die Eingänge UND-verknüpft, bei denen der Ausgang "1" ist -> es entstehen die Minterme,

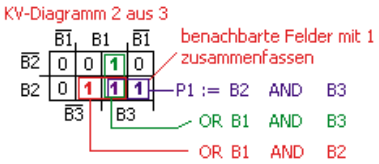
Minterme

- die entstandenen **Minterme** werden ODER-verknüpft,
- die ermittelte Funktionsgleichung wird vereinfacht.




KV-Diagramm

Wird das **KV-Diagramm** benutzt, entfällt die Vereinfachung, da durch die Zusammenfassung von Feldern mit "1" die Funktionsgleichung in minimierter Form ermittelt wird.
Vollziehen Sie den Eintrag im KV-Diagramm nach.



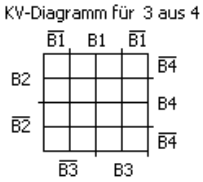

Auf der **InfoClick**-Webseite finden Sie eine kleine Animation.

**Testen Sie** die Richtigkeit in CODESYS.

1. Wie viele Ein- und Ausgänge sind nötig? -> Steuerungskonfiguration
2. Geben Sie die aus dem KV-Diagramm entwickelte Gleichung als Anweisung ein.
3. Testen Sie das Programm, simulieren Sie dabei B1 ... B3.
4. Falls Ihnen ein Controller zur Verfügung steht, übertragen Sie Ihr getestetes Programm in den Controller. Siehe Anhang „Programm in den Controller laden".

## Übung 2.5 Funktion 3 aus 4

**Schreiben Sie** mit Hilfe des KV-Diagramms die Anweisung auf.
**Testen Sie** die Richtigkeit Ihrer Ergebnisse in CODESYS.




P1:= .....

OR .....

OR .....

OR ..... ;

**Übung 2.6** **Funktion 2 aus 3 mit Öffnerkontakten**

Drahtbruch

**Ändern Sie** das Programm, da die **Öffnerkontakte** der Sensoren benutzt werden, um einen **Drahtbruch** zwischen Sensor und der Steuerung erkennen zu können. Ein Drahtbruch wirkt so, als ob der Öffner, z. B. ein Meldekontakt, angesprochen hätte.

**Übung 2.7** **Programmieren nach einer Funktionstabelle**

Schreiben Sie das Programm mit Hilfe der Funktionstabelle.

| S2 | S1 | P3 | P2 | P1 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

**In e!COCKPIT**

Falls Sie eine Steuerung zur Verfügung haben, so können Sie die Konfiguration der Module wie I/O-Klemmen usw. aus der Steuerung lesen.
Projektieren Sie nur die Steuerung, verbinden Sie die Steuerung mit dem PC und wählen Sie in der Geräteansicht im Menü Gerät-Verbinden und -Module ermitteln. Beachten Sie die Hinweise unter Anhang "Programm in den Controller laden, Kommunikation konfigurieren".

**Stichpunkte** Steuerungskonfiguration, Digital-Input, -Output, Datenblatt, PAE, PAA, AT, AND, OR, NOT, BOOL, XOR, Visualisierung, Funktionstabelle -> Anweisung, Minterme, KV-Diagramm, Drahtbruch

# 2. Selbsttest – Boolesche Operationen

Wählen Sie die richtigen Ergänzungen bzw. Aussagen aus.

---

1. Am Anfang jedes Programmzyklus wird der Speicherbereich des PAA und des PAE aktualisiert. Wurde eine Eingabebaugruppe auf Byte 1 konfiguriert, so kann im Anwenderprogramm eine Variable mit Hilfe des Schlüsselworts "AT" auf den Speicherbereich

   ☐ %QX0.0...%QX0.7 zeigen. ☐ %IX0.0...%IX0.7 zeigen.

   ☐ %IX1.0...%IX1.7 zeigen. ☐ %QX1.0...%QX1.7 zeigen.

---

2. Eine Variable vom Datentyp

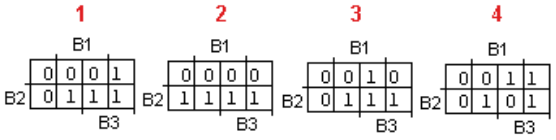
   ☐ INT ☐ REAL ☐ DINT ☐ BOOL

   kann nur die Werte "TRUE" oder "FALSE" annehmen.

---

3. xP1:= xB1 AND xB2 OR xB1 AND xB3 OR xB2 AND xB3;

   Das KV-Diagramm ☐ 1 ☐ 2 ☐ 3 ☐ 4 entspricht der Anweisung.




---

4.

| B2 | B1 | led_ok |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Die Anweisung zur Tabelle lautet:

☐ xLed_ok:=(NOT xB2 AND xB1)OR(xB2 AND NOT xB1);

☐ xLed_ok:=(NOT xB2 OR xB1)AND(xB2 OR NOT xB1);

☐ xLed_ok:=(NOT xB2 AND NOT xB1)OR(xB2 AND xB1);

☐ xLed_ok:=(NOT xB1 AND xB2)OR(xB2 AND NOT xB1);

---

Einen Selbsttest zu diesem Kapitel finden Sie auf InfoClick sowie aktuelle Hinweise zum Umgang mit CoDeSys V2.3, e!COCKPIT und CODESYS V3.

# 3 Datentypen, Codierungen

**Lernziele:** Datentypen, Codes, Zahlendarstellung und Datentypenumwandlung kennen lernen

**Einführung:** Ein Computersystem kann Daten nur binär (0 und 1) verarbeiten. Um ein Bitmuster richtig zu interpretieren, muss bei der Deklaration einer Variablen ein Datentyp angegeben werden. Damit sind die Codierung, der Wertebereich, die Größe des notwendigen Speichers sowie die gültigen Operationen und Funktionen festgelegt.
Beispiel: Das Bitmuster 1111_1111_1111_1111 wird beim Datentyp "WORD" als 65535, jedoch beim Datentyp INT als -1 interpretiert.

## 3.1 Datentypen für logische Werte

BOOL
BYTE
WORD
DWORD

Für logische Operationen wie `AND, OR, XOR, NOT` ... werden folgende Datentypen gewählt:

| | | | |
|---|---|---|---|
| 1 Bit | -> **BOOL** | z. B. | `xQ1` |
| 8 Bits = 1 Byte | -> **BYTE** | z. B. | `bySensorgruppe` |
| 16 Bits = 1 Word | -> **WORD** | z. B. | `wMaske` |
| 32 Bits = 1 Doppelwort | -> **DWORD** | z. B. | `dwStatus` |
| 64 Bits = 1 Langwort | -> **LWORD** | z. B. | `lwFehler` in CODESYS V3 |

**Aufgabe 3.1 Sensorgruppen vergleichen**

Das Bitmuster der Variablen `bySensorgruppe1` soll mit Hilfe des `XOR`-Operanten mit dem Bitmuster der Variablen `bySensorgruppe2` verglichen werden. Bei Abweichungen zeigt die Variable `byErgebnis` dies an.
Wenn Sie mit e!COCKPIT oder I/O-Pro arbeiten, **konfigurieren Sie** die Steuerung, z. B. mit 2x8DI 750-430 und 1x8DO 750-530, geben Sie den Code ein und **analysieren Sie** das Programm mit der PC-Simulation. %IB1 ist die IEC-Adresse des Eingangsbytes 1, %QB0 die des Ausgangsbytes 0. Schalten Sie die Darstellung der Werte um, wie dies im Folgenden beschrieben ist, um die verschiedenen Darstellungen von Variablenwerten kennen zu lernen. Die Adressen können Sie der Steuerungskonfiguration entnehmen. Lassen Sie die Steuerungskonfiguration weg, wenn Sie mit CODESYS V3 arbeiten.

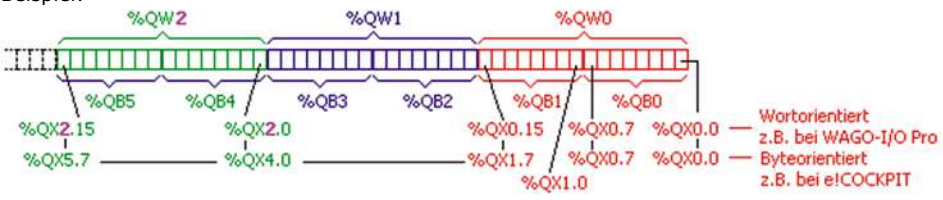
```
PROGRAM PLC_PRG
VAR
 bySensorgruppe1 AT %IB?:BYTE;
 bySensorgruppe2 AT %IB?:BYTE;
 byErgebnis AT %QB0:BYTE;
END_VAR
byErgebnis:= bySensorgruppe1 XOR bySensorgruppe2;
```

Alle Bits der Variablen byErgebnis sind FALSE, wenn beide Sensorgruppen den gleichen Wert haben.

Beachten Sie die **Speicherorganisation**, sie ist vom Zielsystem abhängig.

Speicher-
organisation

Beispiel:



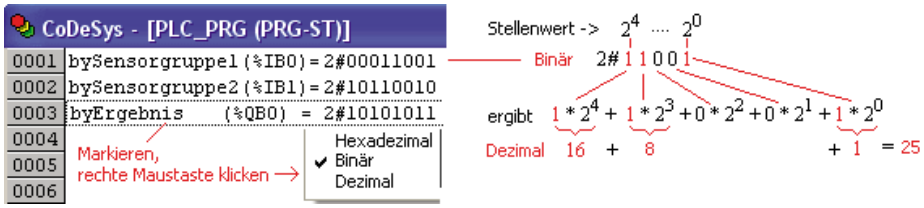
### Darstellung der Werte und Codierung

**Binärcode:** Es verdoppelt sich die Wertigkeit der Bits von rechts nach links.

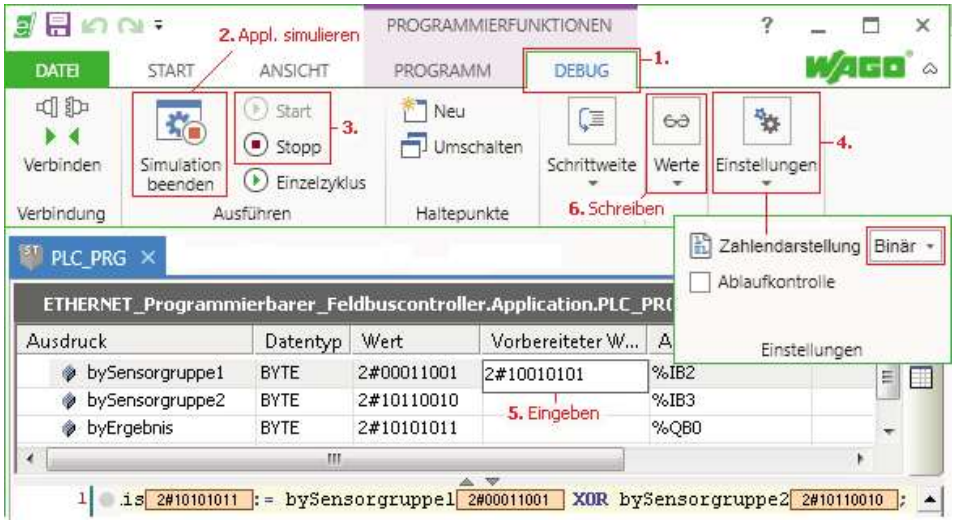
Aktivieren Sie die Simulation, loggen Sie ein und starten Sie die Abarbeitung des Programms. Sie können die Darstellung im Editor umschalten, wie im Bild unten gezeigt wird.

**In CoDeSys V2.3** Test und Darstellung, Aufgabe 3.1

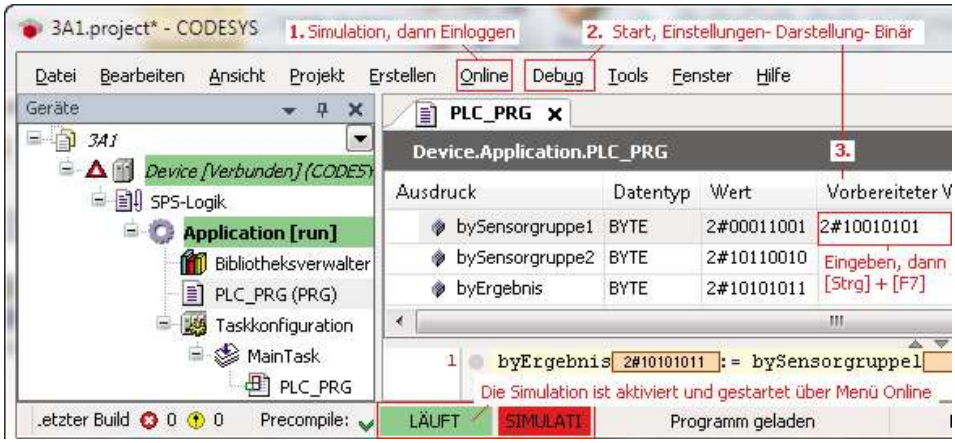
Binärcode 2#...




**In e!COCKPIT** Test und Darstellung, Aufgabe 3.1




**In CODESYS V3 Demo** Test und Darstellung, Aufgabe 3.1

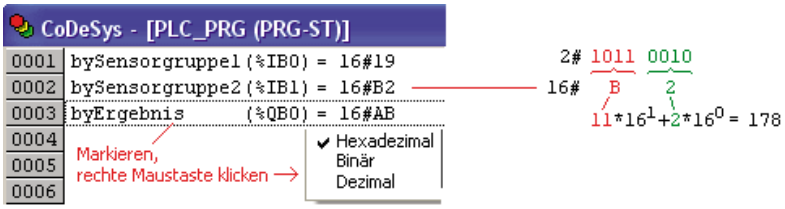


### Sedezimalen(Hex)-Code

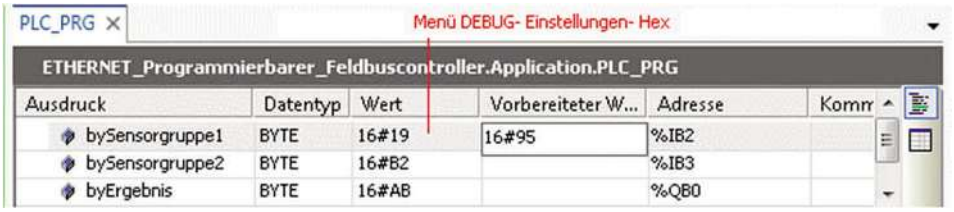
Um vielstellige binärcodierte Zahlen leichter lesen zu können, werden sie oft im Sedezimalen-(Hex)-Code dargestellt. Der Sedezimalcode kennt 16 Zeichen, 0, 1, 2 ... 9, A, B, ... F (A steht für 10 ... F für 15).
Es können dadurch vier Stellen zu einem Zeichen zusammengefasst werden.

**In CoDeSys V2.3** Test und Darstellung, Aufgabe 3.1

Hexcode 16#...




**In e!COCKPIT oder CODESYS V3 Demo** Test und Darstellung, Aufgabe 3.1




### Übung 3.1 Zahlendarstellung

BYTE

Die Werte der Variablen `bySensorgruppe1` (siehe oben) sollen unterschiedlich dargestellt werden. **Füllen Sie** die Tabelle aus.

Beispiele:

| binär | hexadezimal | dezimal |
|---|---|---|
| 2#1001_0101 | | |
| | 16#A2 | |
| | | 19 |

Steht das Bitmuster im Vordergrund, so sollte die binäre oder die hexadezimal Darstellung gewählt werden.

### Aufgabe 3.2 Maske

Maske

WORD

Initialwert

Nur die Bits 0 und 8 ... 11 vom %IW0 (Eingangswort 0, dies ist ein Speicherbereich des PAE) sollen die gleichen Bits vom %QW0 steuern. Die restlichen Bits werden mit Hilfe einer "**Maske**" ausmaskiert.
Der Variablen `wMaske` wird ein **Initialwert** zugewiesen. Nach einem Neustart nimmt die Variable dadurch diesen Wert an.
**Vollziehen Sie** das Beispiel nach. Sie können auch 16#0F01 als Initialisierungswert für wMaske eingeben.

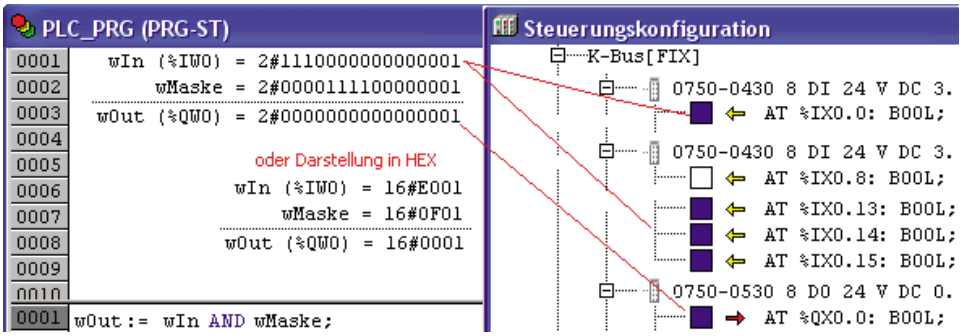
```
VAR
   wIn AT %IW0:WORD;
   wMaske:WORD:= 2#0000_1111_0000_0001;
   wOut AT %QW0:WORD;
END_VAR
```

Bit 11...Bit 8 ... Bit 0

Initialisierung der Variable

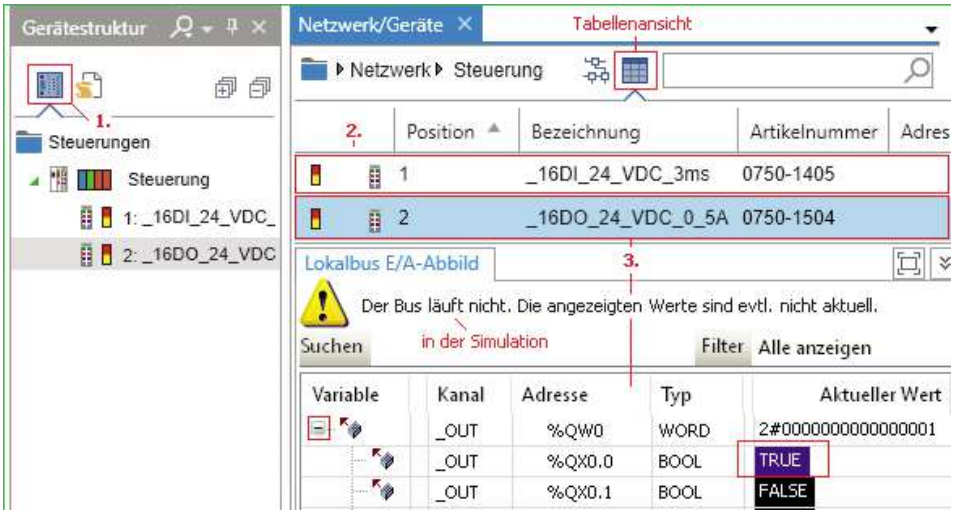
```
wOut:= wIn AND wMaske;
```

**In CoDeSys V2.3** Test, Aufgabe 3.2




**In e!COCKPIT** Test, Aufgabe 3.2

1. Wählen Sie nach dem Start der Simulation im Menü DEBUG in der Programmstruktur die Geräteansicht.
2. Markieren Sie die Eingangsklemme, schreiben Sie einen Wert, achten Sie auf die Adresse.
3. Beurteilen Sie die Werte der Ausgangsklemme.




**In CODESYS V3 Demo** Test, Aufgabe 3.2

Testen Sie Ihr Programm in der Programmansicht (siehe Aufgabe 3.1), da keine Gerätekonfiguration möglich ist.

Beachten Sie:
Die Datentypen `BOOL, BYTE, WORD, DWORD` werden dann benutzt, wenn das Bitmuster im Vordergrund steht. Der Wert der Variablen sollte daher binär, also 2#..., oder sedezimal, also 16#..., eingegeben werden.

## 3.2 Datentypen für ganze Zahlen

Bei Rechenoperationen steht das Bitmuster nicht im Vordergrund. Für Rechenoperationen mit ganzen Zahlen werden folgende Datentypen gewählt:

INT – **INT** (16Bit-Integer) $2^{16}$ = 65536 Zahlen mit der Festlegung –32768...32767 oder wenn nötig
DINT – **DINT** (32Bit-Double Integer) Wertebereich = –2.147.483.648...2.147.483.647.
LINT – **LINT** (64Bit-Integer) in CODESYS V3..

MSB Das hochwertigste Bit (**MSB**-most significant bit) ist das **Vorzeichenbit.**

Beispiel: **0**000_0000_0000_0101 = 5; **1**111_1111_1111_1011 = **–**5

Soll die Zahl – 5 dargestellt werden, so bildet der Controller das **2er-Komplement** von 5, das heißt, er bildet das Komplement (Negation, NOT)

5 = 0000_0000_0000_0101 => 1111_1111_1111_1010
und dann wird 1 addiert => 1111_1111_1111_1011 = –5

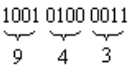
Weitere Datentypen für ganze Zahlen:

SINT – **SINT** (8 Bit-Integer) klein, engl. *short* $2^8$ = 255 Zahlen mit der Festlegung –128...127.
USINT – **USINT** (8 Bit-Integer) vorzeichenlos, engl. *unsigned* Wertebereich = 0...255.
UINT – **UINT** (16 Bit-Integer) Wertebereich = 0...65535.
UDINT – **UDINT** (32 Bit-Integer) $2^{32}$ = 4294967296 Zahlen Wertebereich = 0...4294967295.
ULINT – **ULINT** (64 Bit-Integer) in CODESYS V3..

Das jeweilige Präfix, das Sie für die Variablennamen verwenden sollten, können Sie im Anhang: Begriffe unter Datentypen finden.

BCD Zahlen können auch **BCD**-codiert dargestellt werden (bei S5-SPSn).

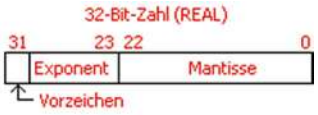
(Packed -> 4 Bits pro Dezimalziffer): Beispiel: 943



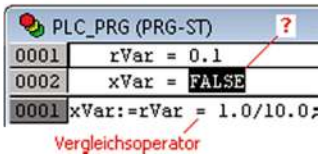

## 3.3 Datentyp für rationale Zahlen

REAL Für Rechenoperationen mit Gleitpunktzahlen (REAL-Zahlen) wird der Datentyp **REAL** (32-Bit Gleitkomma-Zahl) oder LREAL (64-Bit Gleitkomma-Zahl) gewählt. Die Wertebereiche sind vom verwendeten Zielsystem abhängig, benutzen Sie die jeweilige Hilfe in Ihrem System.

Beispiele:
0.2; -314.06; 3.5e+10 (der Exponent muss ganzzahlig sein)
Beachten Sie:
Geben Sie kein Komma, sondern einen Punkt ein.




Vergleichs-operator Vorsicht bei **Vergleichsoperationen** auf gleich (Bild rechts).
Das Ergebnis des Vergleichsausdrucks ist vom Datentyp BOOL und kann der Variablen xVar zugewiesen werden.
Zum Beispiel kann 1/10 binär nicht genau dargestellt werden.
Sie finden auf InfoClick einen Link -Format des Datentyps REAL-.




## 3.4 Datentypen für Zeiten

Für Zeitfunktionen gibt es die Datentypen:

1. **DATE** (16 Bit), der Wert in den 16 Bits entspricht der Anzahl der Tage seit 01.01.1990. Beispiele: D#1990-01-01 (=16#0000), D#2168-12-31 (=16#FF62).

TIME, 2. **TIME** = Zeitdauer (32 Bit), der Wert der 32 Bits wird mit Vorzeichen abgelegt. Beispiele: T#24d20h31m23s647ms (=16#7FFF_FFFF), T#-24d20h31m23s648ms (=16#8000_0000). Siehe Kapitel 9.

DATE, TOD

3. **TIME_OF_DAY** = Tageszeit (32 Bit), der Wert der 32 Bits enthält die Anzahl der Millisekunden seit Tagesbeginn.
   Beispiele: TOD#00:00:00:00 (=16#0000_0000) bis TOD#23:59:59.999 (=16#0526_5BFF)
4. **DATE_AND_TIME** = Datum und Uhrzeit, es ist ein zusammengesetzter Datentyp.
   Beispiel: DT#2089-12-31-23:59:59.999. Eine Übung zu den Datentypen für Zeiten finden Sie im Kapitel 7.

## 3.5 Datentypenumwandlung

Mit Hilfe von IEC-Operatoren zur Datentypenumwandlung wie `WORD_TO_INT();` `INT_TO_WORD();` `BYTE_TO_BOOL()` ... können Datentypen umgewandelt werden. In CODESYS V3 müssen Sie den Eingangstyp nicht angeben, z.B. `TO_INT();` `TO_BOOL()` ... Bei einigen Compilern müssen Datentypenumwandlungen von z.B. `INT_TO_REAL()` oder `BYTE_TO_WORD()` usw. nicht explizit angegeben werden. Es wird eine implizierte Datentypenumwandlung durchgeführt.
Siehe auch CODESYS-Menü Hilfe-> Suchen... Typkonversionsoperatoren.

### Aufgabe 3.3 Typenumwandlung und Vergleichsoperatoren

a) **Untersuchen Sie** die Funktion des Programms.

xx_TO_xx

```
PROGRAM PLC_PRG
(*2 aus 3, mit Temperaturschalter Schließer und Anzeige*)
VAR
 xB1 AT %IX2.0:BOOL; xB2 AT %IX2.1:BOOL; xB3 AT %IX2.2:BOOL;
 xP1 AT %QX0.1:BOOL;(*Anzeige*)
END_VAR
```

Vergleichs-operator

```
xP1:= BOOL_TO_USINT(xB1)+ BOOL_TO_USINT(xB2)+ BOOL_TO_USINT(xB3)>= 2;
```

Das Ergebnis des Vergleichsausdrucks ist vom Datentyp BOOL und kann der Variablen xP1 zugewiesen werden.
**Vergleichen Sie** mit der Übung 2.3. Das Programm hat die Funktion .....

b) Entfernen Sie das Zeichen > vom Vergleichsoperator >= und **untersuchen Sie** die Funktion.
   Ergebnis:
   Nur wenn .....

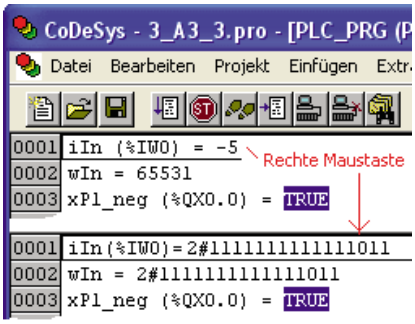
### Aufgabe 3.4 Bitmuster und Wert

**Untersuchen Sie** die Funktion des Programms, lassen Sie die Variablenwerte dezimal als auch binär darstellen. Beachten Sie die Adressierung in der Steuerungskonfiguration.

```
PROGRAM PLC_PRG
(*Autor: ???, Version: 1.0*)
VAR
 iIn AT %IW0:INT;(*eingeben*)
 wIn:WORD;(*beobachten*)
 xP1_neg AT %QX0.0:BOOL;(ist negativ*)
END_VAR

wIn:=INT_TO_WORD(iIn);
(*iIn positiv oder negativ*)
(*ausmaskieren*)
xP1_neg:= WORD_TO_BOOL(wIn AND 16#8000);
```

**In CoDeSys V2.3**, Test, Aufgabe 3.4



**In e!COCKPIT oder CODESYS V3 Demo** Test, Aufgabe 3.4

Wählen Sie nach dem Start der Simulation im Menü DEBUG–Start und Einstellungen Dezimal oder Binär. Überprüfen Sie die Variablenwerte.

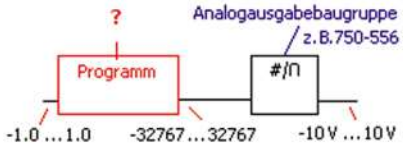
**Anmerkung:**
Da `iIn` nicht mit `AND` verknüpft werden kann, um das Vorzeichen auszumaskieren, wird die Datentypenumwandlungsfunktion `INT_TO_WORD` verwendet.
Um festzustellen, dass `iIn` negativ ist, könnte auch ein Vergleichsoperator verwendet werden. Wie würde die Anweisung aussehen?

`xP1_neg:=` .....

## Aufgabe 3.5 Analogwertverarbeitung 1

An der Analogausgabebaugruppe ist der Analogeingang –10 V...10 V eines Frequenzumrichters zur Drehfrequenzsteuerung angeschlossen.

Die Baugruppe wandelt die Zahl –32767...32767 im Speicher %QW0 in eine Ausgangsspannung von –10 V...10 V um.
Die Adresse, hier %QW0, ist von der Steuerungskonfiguration abhängig.




**Analysieren Sie** das Programm, **füllen Sie** die Tabelle aus und beantworten Sie die Fragen. Übernehmen Sie die Adressen aus der Steuerungskonfiguration.

```
PROGRAM PLC_PRG
(*Zum Test Wert von rIn verändern*)
(*Wandelt -1…1 in -32767…32767*)
VAR
 rIn:REAL;(*-1.0 ... 1.0*)
 iOut AT %QW0:INT;
 wOut:WORD;
 xP1_neg AT %QX2.0:BOOL;
END_VAR

iOut:= REAL_TO_INT(rIn * 32767.0);
wOut:= INT_TO_WORD(iOut);
(*Anzeige:Wert ist negativ*)
xP1_neg:=WORD_TO_BOOL(wOut AND 8000);
```

| rIn | iOut | wOut | xP1_neg |
|---|---|---|---|
| 0.25 | 8192 | 16# ..... | FALSE |
| 0.5 | ..... | 16# ..... | ..... |
| 0.75 | ..... | 16# ..... | ..... |
| 1.0 | ..... | 16# ..... | ..... |
| -0.25 | ..... | 16# ..... | ..... |

Welche Aufgabe übernimmt der Programmcode in der

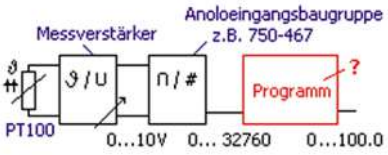
– Zeile 1 im Anweisungsteil? .....

– Zeile 2 im Anweisungsteil? .....

– Zeile 3 im Anweisungsteil? .....

## Übung 3.2 Analogwertverarbeitung, Skalieren

An der Analogeingabebaugruppe ist zur Temperaturmessung über einen Messverstärker ein temperaturabhängiger Widerstand angeschlossen. Der Messverstärker wandelt die Widerstandsänderung des PT100 bei 0...100 °C in eine Spannung 0...10 V um.
Die Analogeingabebaugruppe wandelt die 0...10 V in eine Zahl 0...32760 um und legt diese alle 2 ms in den Speicher ab.

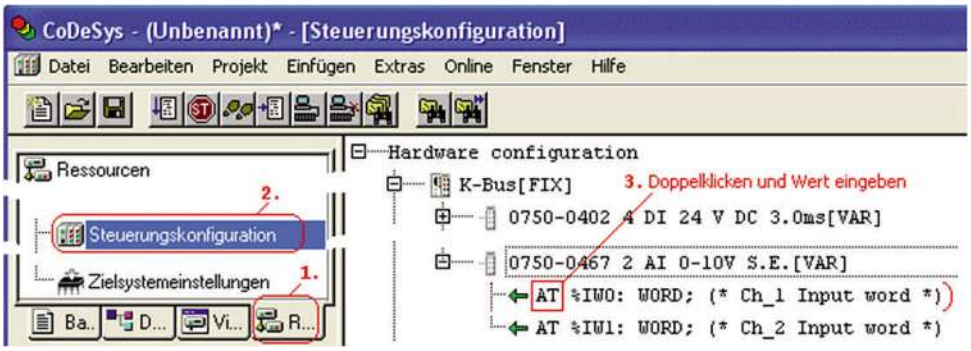



Das zu erstellende Programm soll die Zahl im %IWx in den Wert 0...100.0 umsetzen.

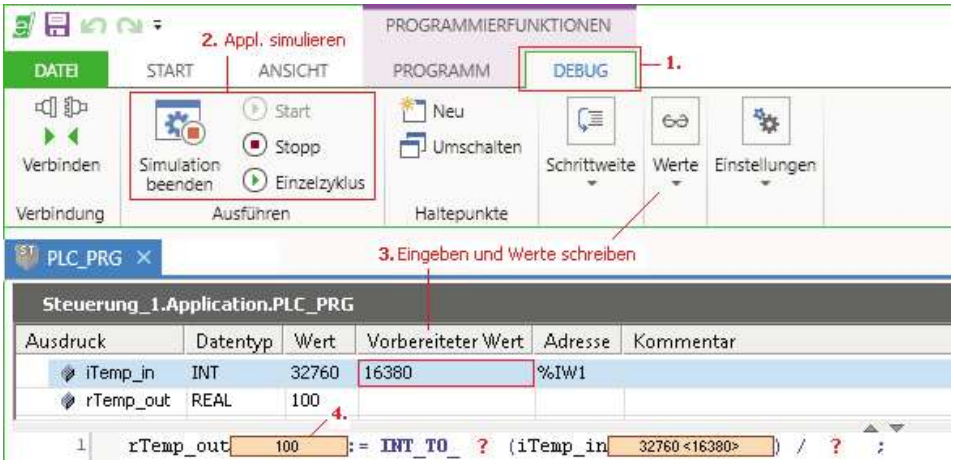
**Geben Sie** beim Testen des Programms die Zahl 0...32760 vor, z. B. 16380 ergibt 50 (50,0 °C) als Ergebnis.

**Testen Sie** das Programm.
Die Adresse des Speicherplatzes können Sie der Steuerungskonfiguration entnehmen.
Anmerkung:
Sie könnten auch die AI-Klemme 750-461 verwenden, der Messverstärker könnte dann entfallen.

**In CoDeSys V2.3** Test, Übung 3.2




**In e!COCKPIT** Test, Übung 3.2



Falls Sie einen Controller mit den entsprechenden Klemmen haben, **übertragen Sie** Ihr getestetes Programm in den Controller.

**In CODESYS V3 Demo** Test, Übung 3.2

Hier ist keine I/O-Konfiguration möglich, Sie können Ihr Programm auch ohne I/O-Baugruppen testen.

## 3.6 Zusammengesetzte Datentypen

Zusammengesetzte Datentypen basieren auf elementaren Datentypen, wie BOOL...DWORD, ganzzahlige Datentypen USINT...DINT, Fließpunktzahlen REAL, LREAL und TIME.

STRING

1. **DATE_AND_TIME**, siehe oben  Datentyp für Zeiten.
2. **STRING**, z. B. `sOut:STRING:= 'Palette ist angekommen';.`
3. **STRUCT**, es ist eine Zusammenstellung verschiedener Variablen, siehe dazu Kapitel 7.
4. **Aufzählungstyp**, er besteht aus einer Menge von String-Konstanten, siehe dazu Kapitel 7.
5. **ARRAY,** es sind **Feldvariablen**, diese können über einen Index angesprochen werden. Die Variablen müssen vom gleichen Datentyp sein.

### 3.6.1 Feldvariable – ARRAY

ARRAY

Mit Feldvariablen (Feld engl. *array*) können Sie mehrere Variablen vom gleichen Datentyp und Namen über einen Index ansprechen. Bei der Deklaration können die Variablen **initialisiert** werden, das heißt, die Variablen werden beim Neustart mit dem deklarierten Anfangswert beschrieben. Weitere Aufgaben finden Sie ab Kapitel 5.

**Aufgabe 3.6 Ein- und zweidimensionale Feldvariable**

**Vollziehen Sie** die Beispiele nach.

**Eindimensionale Feldvariable**
Ein Antrieb soll 5 Positionen anfahren; die Positionen werden in einer eindimensionalen Feldvariablen gespeichert.

**In CoDeSys V2.3** Variable initialisieren und lesen, Aufgabe 3.6

Initialisierung

PLC_PRG (PRG-ST)

```
0002 VAR
0003     rPosition: ARRAY[0..4] OF REAL:= -10.5, 20.2, 35.3, 40.0, 55.5;
0004     rAnfangsposition, rEndposition: REAL;
0005 END_VAR
0006
```

```
0001 (*Werte aus der Feldvariable auslesen*)
0002 rAnfangsposition:= rPosition[0];(*Wert = .....*)
0003 rEndposition:= rPosition[4];(*Wert = .....*)
```