**Passende SCE Trainer Pakete zu dieser Lern-/Lehrunterlage**



Lern-/Lehrunterlagen  
  
Siemens Automation Cooperates with Education (SCE) | Ab Version V14 SP1

**siemens.de/sce**

TIA Portal Modul 051-201

Hochsprachenprogrammierung

mit SCL und SIMATIC S7-1200

* **SIMATIC S7-1200 AC/DC/RELAIS 6er "TIA Portal"**  
  Bestellnr.: 6ES7214-1BE30-4AB3
* **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**  
  Bestellnr.: 6ES7214-1AE30-4AB3
* **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (für S7-1200) 6er "TIA Portal"**  
  Bestellnr.: 6ES7822-0AA04-4YE5

Bitte beachten Sie, dass diese Trainer Pakete ggf. durch Nachfolge-Pakete ersetzt werden. Eine Übersicht über die aktuell verfügbaren SCE Pakete finden Sie unter:[siemens.de/sce/tp](http://www.siemens.de/tp)

**Fortbildungen**

Für regionale Siemens SCE Fortbildungen kontaktieren Sie Ihren regionalen SCE Kontaktpartner:

[siemens.de/sce/contact](http://www.siemens.de/contact)

**Weitere Informationen rund um SCE**

[siemens.de/sce](http://www.siemens.de/sce)

**Verwendungshinweis**  
Die SCE Lern-/Lehrunterlage für die durchgängige Automatisierungslösung Totally Integrated Automation (TIA) wurde für das Programm „Siemens Automation Cooperates with Education (SCE)“ speziell zu Ausbildungszwecken für öffentliche Bildungs- und F&E-Einrichtungen erstellt. Die Siemens AG übernimmt bezüglich des Inhalts keine Gewähr.

Diese Unterlage darf nur für die Erstausbildung an Siemens Produkten/Systemen verwendet werden. D.h. sie kann ganz oder teilweise kopiert und an die Auszubildenden zur Nutzung im Rahmen deren Ausbildung ausgehändigt werden. Die Weitergabe sowie Vervielfältigung dieser Unterlage und Mitteilung ihres Inhalts ist innerhalb öffentlicher Aus- und Weiterbildungsstätten für Zwecke der Ausbildung gestattet.

Ausnahmen bedürfen der schriftlichen Genehmigung durch die Siemens AG Ansprechpartner:   
Herr Roland Scheuerer roland.scheuerer@siemens.com.

Zuwiderhandlungen verpflichten zu Schadensersatz. Alle Rechte auch der Übersetzung sind vorbehalten, insbesondere für den Fall der Patentierung oder GM-Eintragung.

Der Einsatz für Industriekunden-Kurse ist explizit nicht erlaubt. Einer kommerziellen Nutzung der Unterlagen stimmen wir nicht zu.

Wir danken der TU Dresden, besonders Prof. Dr.-Ing. Leon Urbas und der Fa. Michael Dziallas Engineering und allen weiteren Beteiligten für die Unterstützung bei der Erstellung dieser SCE Lern-/Lehrunterlage.

Inhaltsverzeichnis

[1. Zielstellung 4](#_Toc485765101)

[2. Voraussetzung 4](#_Toc485765102)

[3. Benötigte Hardware und Software 5](#_Toc485765103)

[4. Theorie 6](#_Toc485765104)

[4.1 Zur Programmiersprache SCL 6](#_Toc485765105)

[4.2 Zur Entwicklungsumgebung SCL 6](#_Toc485765106)

[5. Aufgabenstellung 9](#_Toc485765107)

[5.1 Beispielaufgabe Tankinhalt 9](#_Toc485765108)

[5.2 Erweiterung der Beispielaufgabe 9](#_Toc485765109)

[6. Planung 9](#_Toc485765110)

[6.1 Globaler Datenbaustein „Daten\_Tank“ 9](#_Toc485765111)

[6.2 Funktion „Berechnung\_Tankinhalt“ 10](#_Toc485765112)

[6.3 Erweiterung der Funktion „Berechnung\_Tankinhalt“ 10](#_Toc485765113)

[7. Strukturierte Schritt-für-Schritt-Anleitung 11](#_Toc485765114)

[7.1 Dearchivieren eines vorhandenen Projekts 11](#_Toc485765115)

[7.2 Speichern des Projektes unter einem neuen Namen 12](#_Toc485765116)

[7.3 Anlegen des Datenbausteins „Daten\_Tank“ 12](#_Toc485765117)

[7.4 Erstellen der Funktion „Berechne\_Inhalt“ 14](#_Toc485765118)

[7.5 Schnittstelle der Funktion „Berechne\_Inhalt“ festlegen 15](#_Toc485765119)

[7.6 Programmierung der Funktion „Berechne\_Inhalt“ 16](#_Toc485765120)

[7.7 Programmierung des Organisationsbausteins „Main [OB1]“ 17](#_Toc485765121)

[7.8 Programm übersetzen und laden 19](#_Toc485765122)

[7.9 Organisationsbaustein beobachten und testen 20](#_Toc485765123)

[7.10 Erweiterung der Funktion „Berechne\_Inhalt“ 22](#_Toc485765124)

[7.11 Organisationsbaustein anpassen 27](#_Toc485765125)

[7.12 Programm übersetzen, speichern und laden 28](#_Toc485765126)

[7.13 Organisationsbaustein beobachten und testen 29](#_Toc485765127)

[7.14 Funktion „Berechne\_Inhalt“ beobachten und testen 31](#_Toc485765128)

[7.15 Archivieren des Projektes 34](#_Toc485765129)

[8. Checkliste 35](#_Toc485765130)

[9. Übung 36](#_Toc485765131)

[9.1 Aufgabenstellung – Übung 36](#_Toc485765132)

[9.2 Planung 37](#_Toc485765133)

[9.3 Checkliste – Übung 37](#_Toc485765134)

[10. Weiterführende Information 38](#_Toc485765135)

Hochsprachenprogrammierung   
mit SCL und S7-1200

# Zielstellung

In diesem Kapitel lernen Sie die grundlegenden Funktionen der Hochsprache SCL kennen. Des Weiteren werden Testfunktionen zur Beseitigung logischer Programmierfehler aufgezeigt.

Es können die unter Kapitel 3 aufgeführten SIMATIC S7-Steuerungen eingesetzt werden.

# Voraussetzung

Dieses Kapitel baut auf der Hardwarekonfiguration einer SIMATIC S7-1200 auf. Es kann mit beliebigen Hardwarekonfigurationen, die digitale Eingangs- und Ausgangskarten besitzen, realisiert werden. Zur Durchführung dieses Kapitels können Sie z.B. auf das folgende Projekt zurückgreifen:

„SCE\_DE\_011\_101\_Hardwarekonfiguration\_CPU1214C…..zap14“

Weiterhin sollten Grundlagenkenntnisse über Hochsprachenprogrammierung wie z.B. Pascal bekannt sein.

# Benötigte Hardware und Software

**1** Engineering Station: Voraussetzungen sind Hardware und Betriebssystem (weitere Informationen siehe Readme/Liesmich auf den TIA Portal Installations-DVDs)

**2** Software SIMATIC STEP 7 Basic im TIA Portal – ab V14 SP1

**3** Steuerung SIMATIC S7-1200, z.B. CPU 1214C DC/DC/DC – ab Firmware V4.2.1

**4** Ethernet-Verbindung zwischen Engineering Station und Steuerung



**2** SIMATIC STEP 7 Basic (TIA Portal) ab V14 SP1



**1** Engineering Station

**4** Ethernet-Verbindung



**3** Steuerung SIMATIC S7-1200

# Theorie

## Zur Programmiersprache SCL

SCL (Structured Control Language) ist eine höhere Programmiersprache, die sich an Pascal orientiert und eine strukturierte Programmierung ermöglicht. Die Sprache entspricht der in der Norm DIN EN-61131-3 (IEC 61131-3) festgelegten Programmiersprache ST „Structured Text“.   
SCL enthält neben Hochsprachenelementen auch typische Elemente der SPS als Sprachelemente wie Eingänge, Ausgänge, Zeiten, Merker, Bausteinaufrufe usw. Sie unterstützt das Bausteinkonzept von STEP 7 und ermöglicht daher neben Kontaktplan (KOP) und Funktionsplan (FUP) die normkonforme Programmierung von Bausteinen. D. h. SCL ergänzt und erweitert die Programmiersoftware STEP 7 mit ihren Programmiersprachen KOP und FUP.

Sie müssen nicht jede Funktion selbst erstellen, sondern können auf vorgefertigte Bausteine wie Systemfunktionen oder Systemfunktionsbausteine zurückgreifen, die im Betriebssystem der Zentralbaugruppe vorhanden sind.

Sie können Bausteine, die mit SCL programmiert sind, mit KOP- und FUP-Bausteinen mischen. Das bedeutet, dass ein mit SCL programmierter Baustein einen anderen Baustein, der in KOP oder FUP programmiert ist, aufrufen kann. Entsprechend können SCL Bausteine auch in KOP- und FUP-Programmen aufgerufen werden.

Es können ebenfalls SCL-Netzwerke in KOP- und FUP-Bausteinen eingefügt werden.

Die Testfunktionen von SCL ermöglichen die Suche nach logischen Programmierfehlern in einer fehlerfreien Übersetzung.

## Zur Entwicklungsumgebung SCL

Zur Verwendung und zum Einsatz von SCL gibt es eine Entwicklungsumgebung, die sowohl auf spezifische Eigenschaften von SCL, als auch auf STEP 7 abgestimmt ist. Diese Entwicklungs-umgebung besteht aus einem Editor/Compiler und einem Debugger.

**SCL für S7-1200**

**Debugger**

**Editor/Compiler**

**Editor/Compiler**

Der SCL-Editor ist ein Texteditor, mit dem beliebige Texte bearbeitet werden können. Seine zentrale Aufgabe ist das Erzeugen und Bearbeiten von Bausteinen für STEP 7-Programme. Während der Eingabe erfolgt eine grundlegende Syntaxprüfung, welche das fehlerfreie Programmieren vereinfacht. Syntaxfehler werden in unterschiedlichen Farben dargestellt.

**Der Editor bietet folgende Möglichkeiten:**

* Programmierung eines S7-Bausteines in der Sprache SCL
* Komfortables Einfügen von Sprachelementen und Bausteinaufrufen mittels Drag&Drop
* Direkte Syntaxprüfung während der Programmierung
* Einstellung des Editors nach Ihren Anforderungen, z.B. durch syntaxgerechtes Einfärben der verschiedenen Sprachelemente
* Überprüfung des fertiggestellten Bausteines mittels übersetzen
* Anzeigen aller Fehler und Warnungen, die beim Übersetzen auftreten
* Lokalisieren der fehlerhaften Stellen im Baustein, optional mit Fehlerbeschreibung und Angaben zur Fehlerbeseitigung

**Debugger**

Der SCL-Debugger kann ein Programm in seinem Ablauf im Automatisierungssystem (AS) kontrollieren und somit mögliche logische Fehler finden.

SCL bietet dazu zwei verschiedene Testmodi an:

* Kontinuierliches Beobachten
* Schrittweises Beobachten

Mit dem „Kontinuierlichen Beobachten“ können Sie eine Gruppe von Anweisungen innerhalb eines Bausteins testen. Während des Testlaufs werden die Werte der Variablen und Parameter in chronologischer Abfolge angezeigt und – sofern möglich – zyklisch aktualisiert.

Beim „Schrittweisen Beobachten“ wird der logische Programmablauf nachvollzogen. Sie können den Programm-Algorithmus Anweisung für Anweisung ausführen und in einem Ergebnisfenster beobachten, wie sich die dabei bearbeiteten Variableninhalte verändern.

Ob das „Schrittweise Beobachten“ möglich ist, hängt von der eingesetzten CPU ab. Diese muss den Einsatz von Haltepunkten unterstützen. Die in diesem Dokument eingesetzte CPU unterstützt keine Haltepunkte.

# Aufgabenstellung

## Beispielaufgabe Tankinhalt

Im ersten Teil soll die Berechnung eines Tankinhaltes programmiert werden.

## Erweiterung der Beispielaufgabe

Im zweiten Teil soll die Aufgabe erweitert und eine Fehlerauswertung programmiert werden.

# Planung

Der Tank hat die Form eines stehenden Zylinders. Der Füllstand des Inhaltes wird mit einem Analogsensor gemessen. Für den ersten Test soll der Wert des Füllstandes schon normiert, in der Einheit „Meter“, vorliegen.

Globale Parameter wie z.B. der Durchmesser und die Höhe des Tanks sollen in einem globalen Datenbaustein „Daten\_Tank“ strukturiert abgelegt werden.

Das Programm zur Berechnung des Inhaltes soll in einer Funktion „Berechnung\_Tankinhalt“ geschrieben werden und die Parameter die Einheit „Meter“ bzw. „Liter“ verwenden.

## Globaler Datenbaustein „Daten\_Tank“

Die globalen Parameter werden in einem globalen Datenbaustein in mehreren Strukturen abgelegt.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Datentyp** | **Startwert** | **Kommentar** |
| Abmessungen | STRUCT |  |  |
| Hoehe | REAL | 12.0 | in Meter |
| Durchmesser | REAL | 3.5 | in Meter |
| Messwerte | STRUCT |  |  |
| Fuellstand\_per | INT | 0 | Wert zwischen 0...27648 |
| Fuellstand\_skal | REAL | 0.0 | Wert zwischen 0…12.0 |
| Inhalt | REAL | 0.0 | Inhalt des Tanks in Liter |
| Fehlerflags | STRUCT |  |  |
| berechne\_inhalt | BOOL |  | im Fehlerfall = TRUE |
| skalieren | BOOL |  | im Fehlerfall = TRUE |

Tabelle 1: Parameter im Datenbaustein "Daten\_Tank"

## Funktion „Berechnung\_Tankinhalt“

Dieser Baustein berechnet den Inhalt des Tanks in Litern.

Im ersten Schritt soll keine Überprüfung auf Sinnhaftigkeit der übergebenen Parameter erfolgen.

Für diesen Schritt sind folgende Parameter erforderlich:

|  |  |  |
| --- | --- | --- |
| **Input** | **Datentyp** | **Kommentar** |
| Durchmesser | REAL | Durchmesser des Zylindertanks in Meter |
| Fuellstand | REAL | Füllstand des Tankinhaltes in Meter |
| **Output** |  |  |
| Inhalt | REAL | Inhalt des Zyklindertanks in Liter |

Tabelle 2: Parameter für FC "Berechnung\_Tankinhalt" im ersten Schritt

Zur Lösung der Aufgabe wird die Formel zur Volumenberechnung eines stehenden Zylinders angewendet. Der Umrechnungsfaktor 1000 wird verwendet, um das Ergebnis in Litern zu berechnen.

2

 => 

## Erweiterung der Funktion „Berechnung\_Tankinhalt“

Der zweite Schritt prüft, ob der Durchmesser größer als Null ist. Des Weiteren soll getestet werden, ob der Füllstand größer oder gleich Null und kleiner oder gleich der Höhe des Tanks ist.

Im Fehlerfall wird der neue Parameter „er“ auf TRUE gesetzt und der Parameter „Inhalt“ erhält den Wert -1.

Erweitern Sie dazu die Schnittstelle um den Parameter „er“ und „Hoehe“.

|  |  |  |
| --- | --- | --- |
| **Input** | **Datentyp** | **Kommentar** |
| Hoehe | REAL | Hoehe des Zylindertanks in Meter |
| Durchmesser | REAL | Durchmesser des Zylindertanks in Meter |
| Fuellstand | REAL | Füllstand des Tankinhaltes in Meter |
| **Output** |  |  |
| er | BOOL | Fehlerflag; bei Fehler = TRUE |
| Inhalt | REAL | Inhalt des Zyklindertanks in Liter |

Tabelle 3: Parameter für FC "Berechnung\_Tankinhalt" im zweiten Schritt

# Strukturierte Schritt-für-Schritt-Anleitung

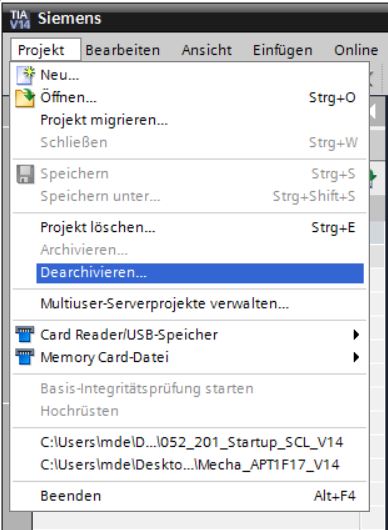
Im Folgenden finden Sie eine Anleitung zur Umsetzung der Planung. Sollten Sie gut zurechtkommen, so reichen Ihnen die nummerierten Schritte zur Bearbeitung aus. Ansonsten folgen Sie einfach den folgenden detaillierten Schritten der Anleitung.

## Dearchivieren eines vorhandenen Projekts

* Bevor Sie mit der Programmierung beginnen können, benötigen Sie ein Projekt mit einer Hardwarekonfiguration.   
  (z.B. SCE\_DE\_011-101\_Hardwarekonfiguration\_CPU1214C\_....zap14)

Zum Dearchivieren eines vorhandenen Projekts müssen Sie aus der Projektansicht heraus unter → Projekt → Dearchivieren das jeweilige Archiv aussuchen. Bestätigen Sie Ihre Auswahl anschließend mit „öffnen“.

(→ Projekt → Dearchivieren → Auswahl eines .zap-Archivs → öffnen)

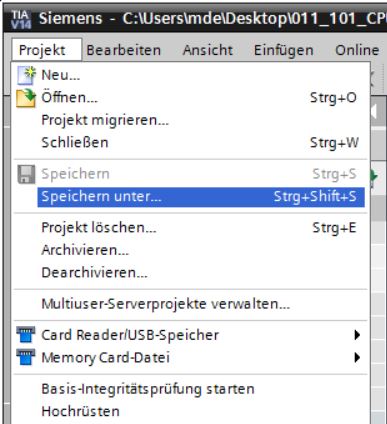


* Als nächstes kann das Zielverzeichnis ausgewählt werden, in welchem das dearchivierte Projekt gespeichert werden soll. Bestätigen Sie Ihre Auswahl mit „OK“.

(→ Projekt → Speichern unter → OK)

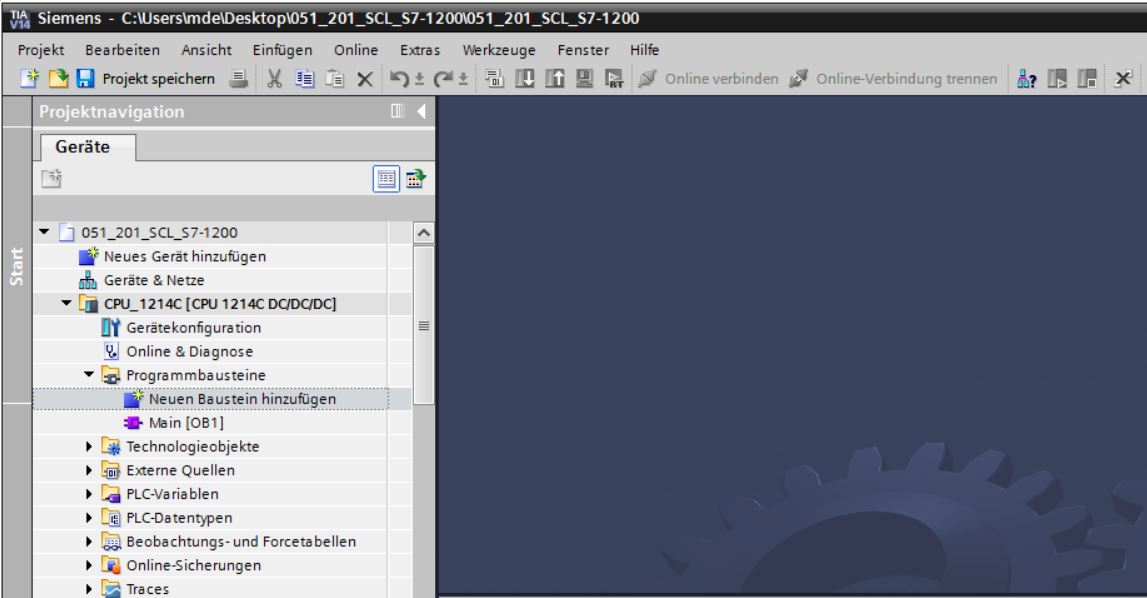
## Speichern des Projektes unter einem neuen Namen

* Das geöffnete Projekt speichern Sie unter dem Namen 051-201\_SCL\_S7-1200.   
  (→ Projekt → Speichern unter … → 051-201\_SCL\_S7-1200 → Speichern)



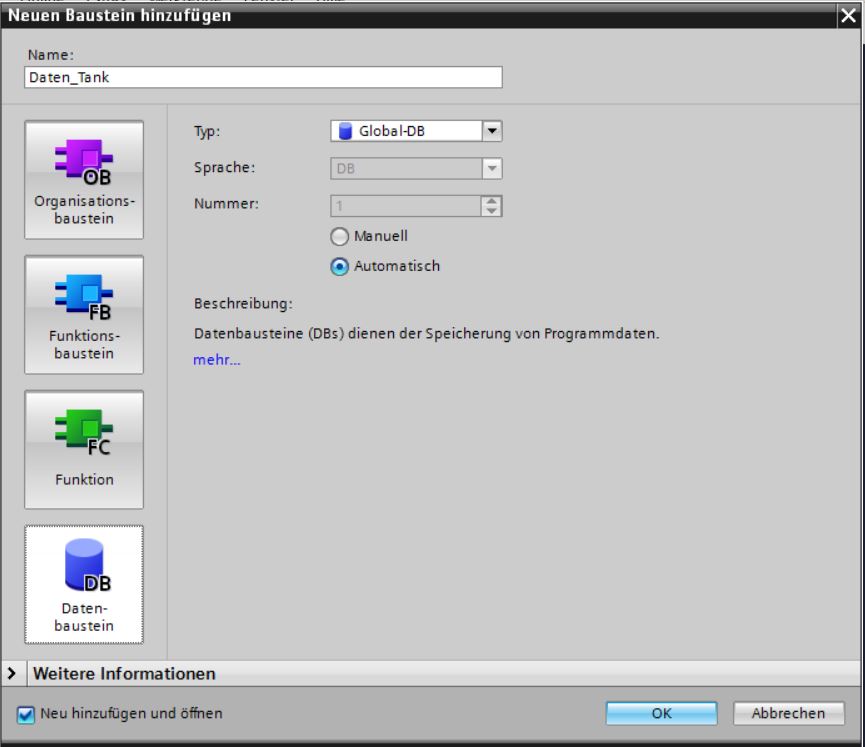
## Anlegen des Datenbausteins „Daten\_Tank“

* Navigieren Sie in der Projektansicht zu den → Programmbausteinen und erstellen Sie einen neuen Baustein, indem Sie doppelklicken auf → Neuen Baustein hinzufügen.

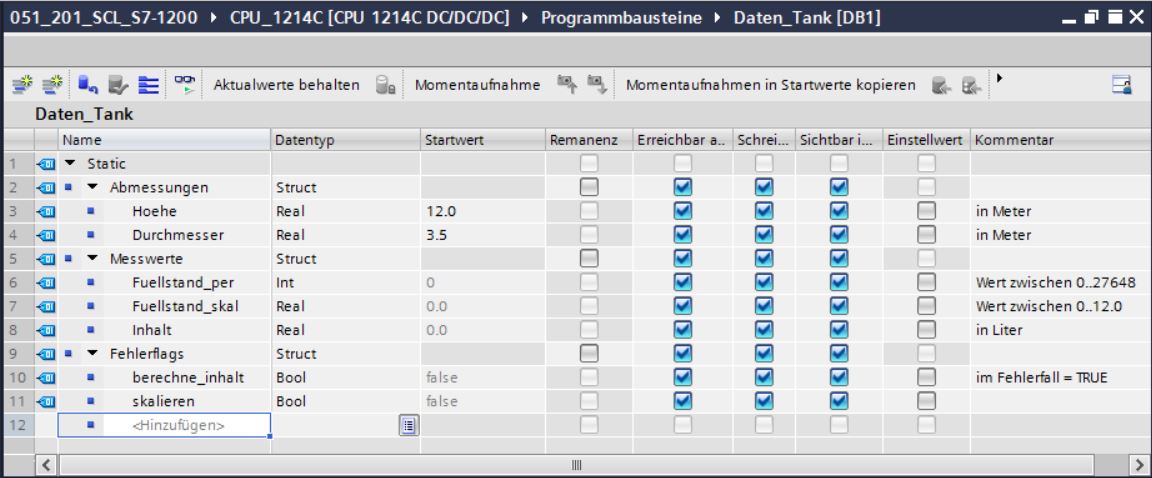


* Wählen Sie nun einen Datenbaustein an und geben den Namen ein.

(→→ „Daten\_Tank“ → OK)

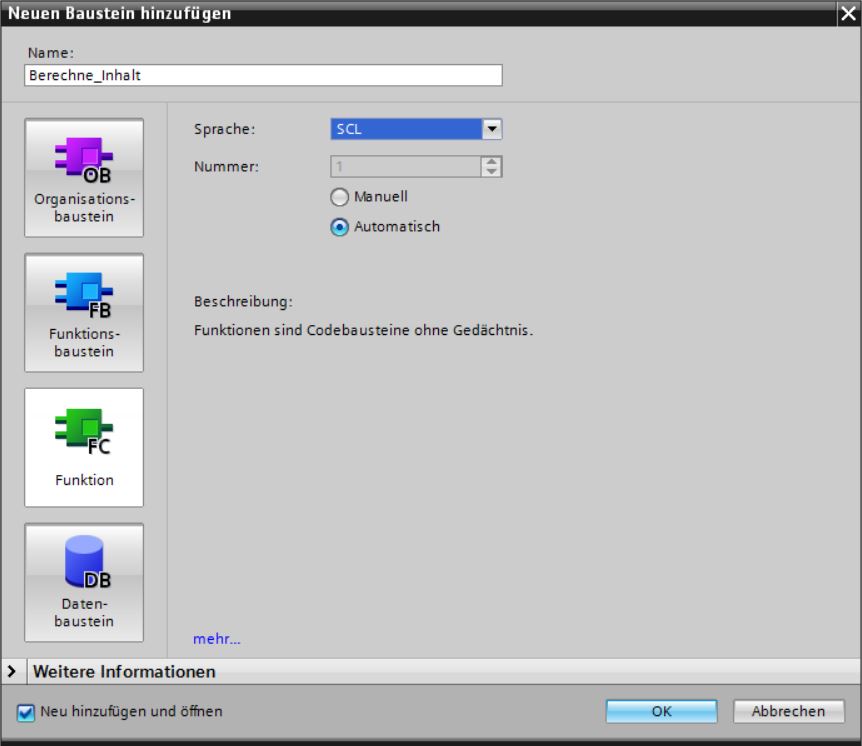


* Geben Sie anschließend die unten angegebenen Namen der Variablen mit Datentyp, Startwert und Kommentar ein.



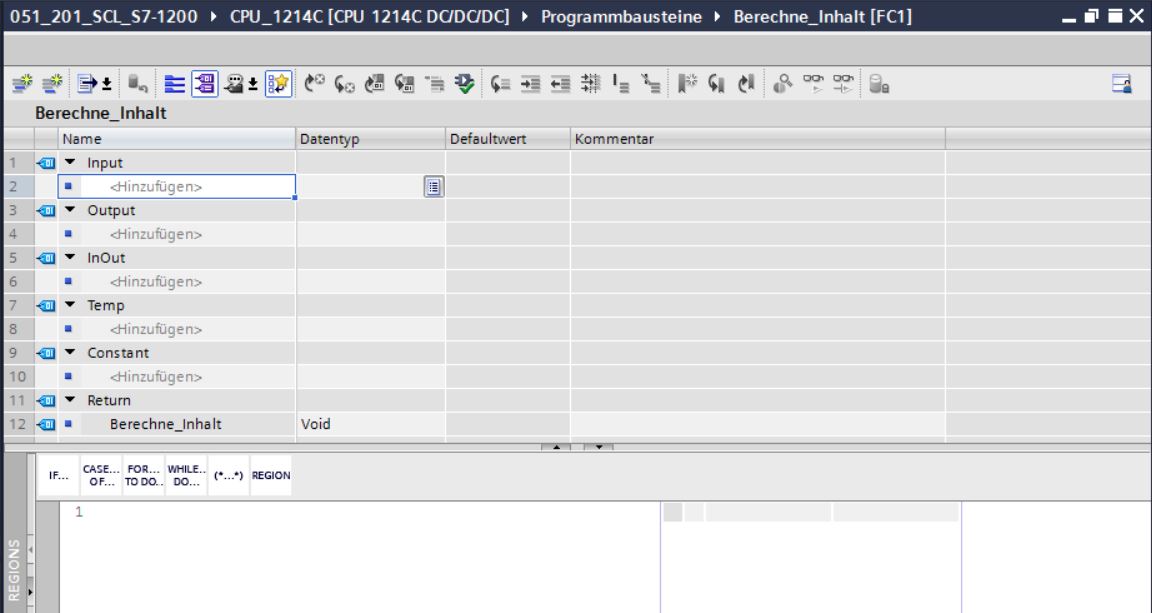
## Erstellen der Funktion „Berechne\_Inhalt“

* Jetzt fügen Sie eine Funktion hinzu, geben den Namen ein und wählen die Sprache aus.  
  (→ Neuen Baustein hinzufügen → → „Berechne\_Inhalt“ → SCL → OK)



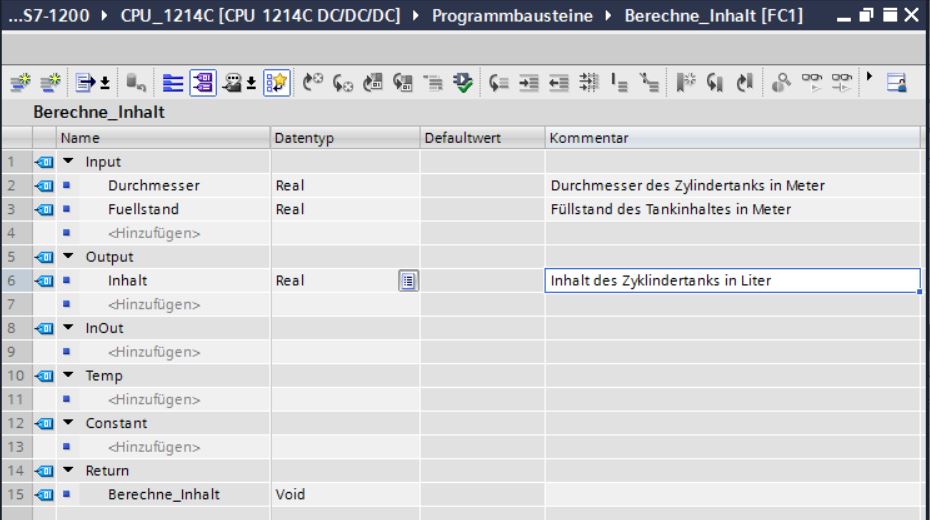
## Schnittstelle der Funktion „Berechne\_Inhalt“ festlegen

* Im oberen Abschnitt Ihrer Programmieransicht finden Sie die Schnittstellenbeschreibung Ihrer Funktion.



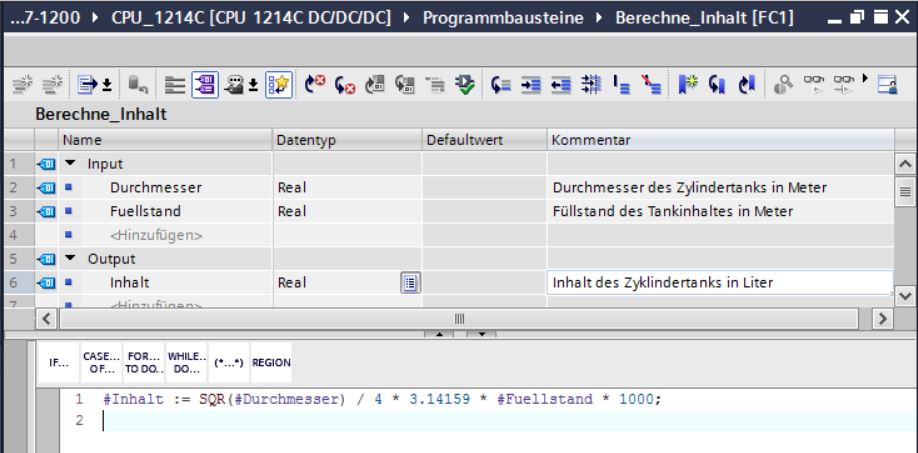
* Legen Sie die folgenden Input- und Outputparameter an.

(→ Name → Datentyp → Kommentar)



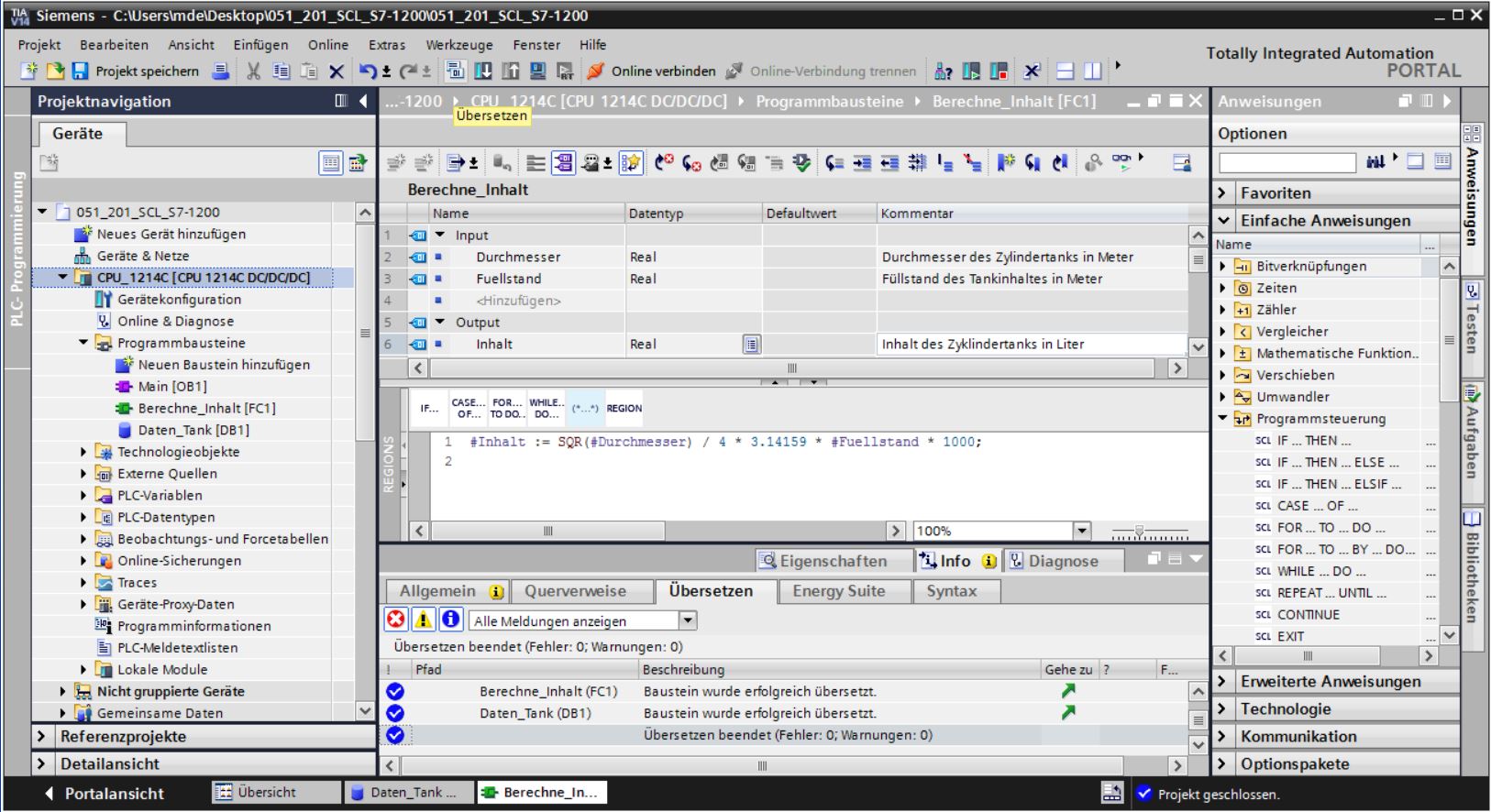
## Programmierung der Funktion „Berechne\_Inhalt“

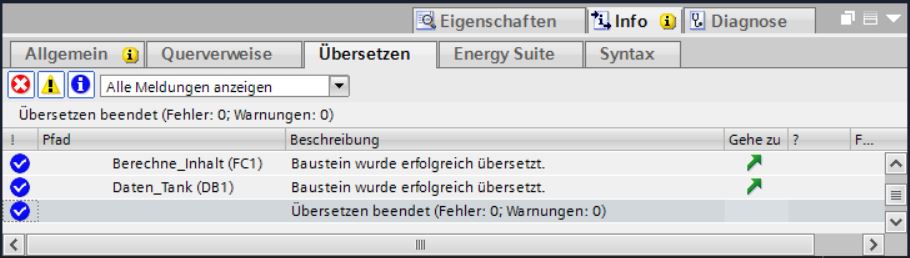
* Geben Sie unten stehendes Programm ein. (→ Programm eingeben)



* Übersetzen Sie nun Ihr Programm und überprüfen es auf syntaktische Fehler.   
  Diese werden im Inspektorfenster unterhalb der Programmierung angezeigt.  
  Beheben Sie gegebenenfalls die Fehler und übersetzen Sie anschließend erneut.

Speichern Sie danach Ihr Programm. (→  → Fehler beheben → )

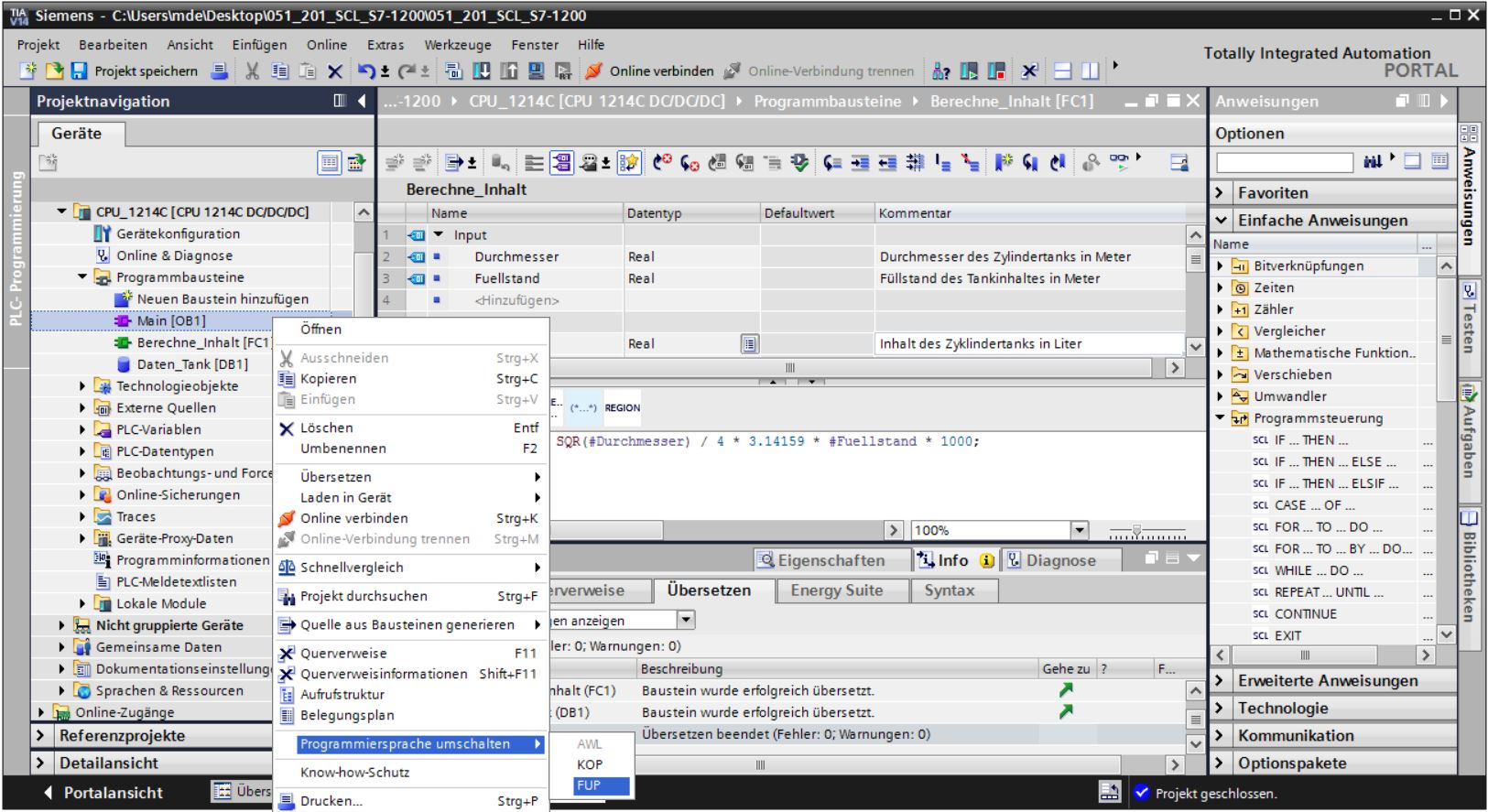




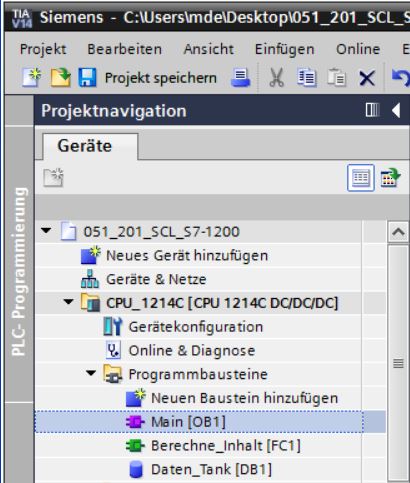
## Programmierung des Organisationsbausteins „Main [OB1]“

* Vor der Programmierung des Organisationsbausteins „Main [OB1]“ stellen Sie die Programmiersprache auf FUP um. Klicken Sie hierzu vorher mit der linken Maustaste im Ordner „Programmbausteine“ auf „Main [OB1]“.

(→ CPU\_1214C[CPU 1214C DC/DC/DC] → Programmbausteine → Main [OB1] → Programmiersprache umschalten → FUP)

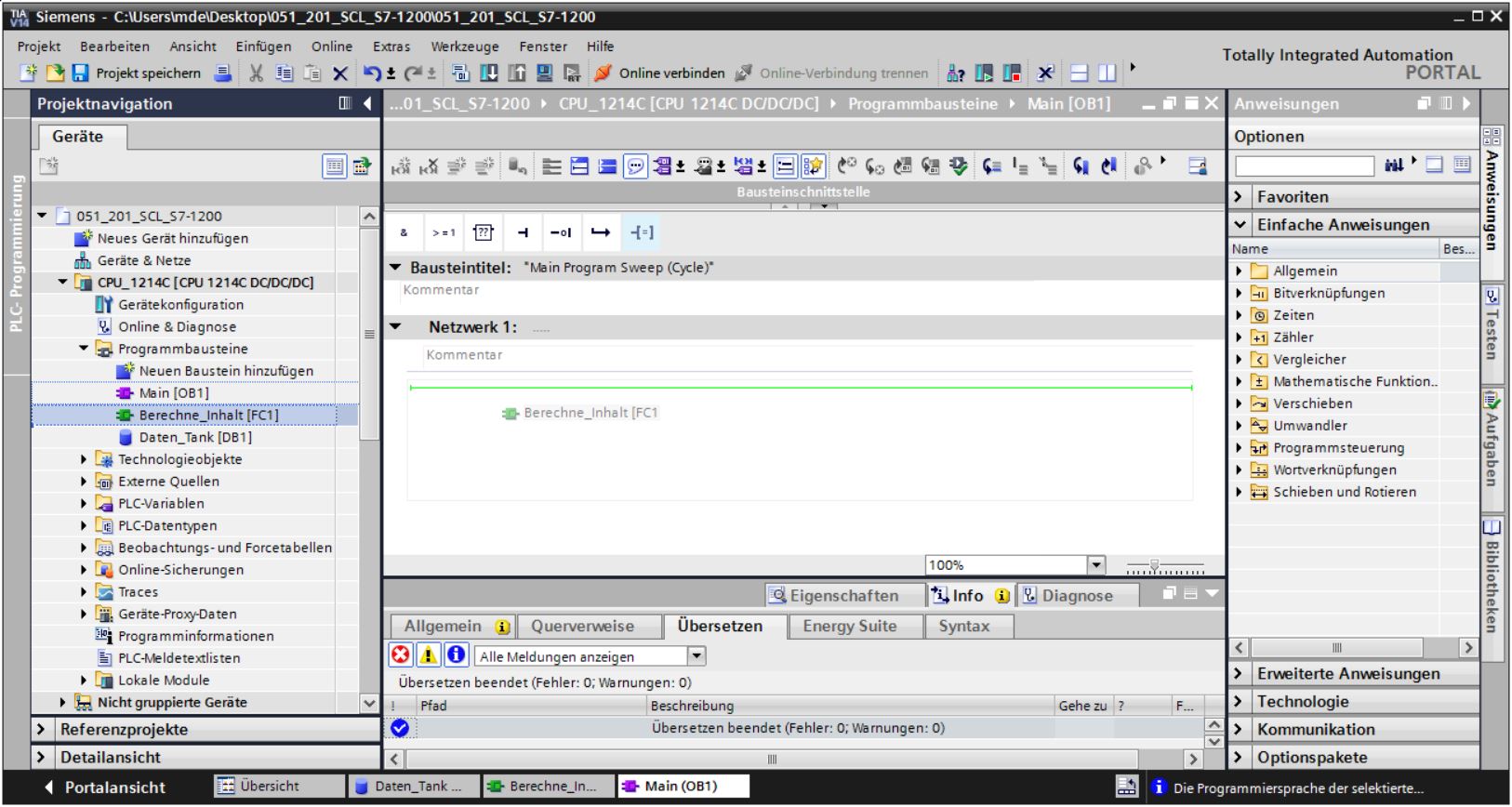


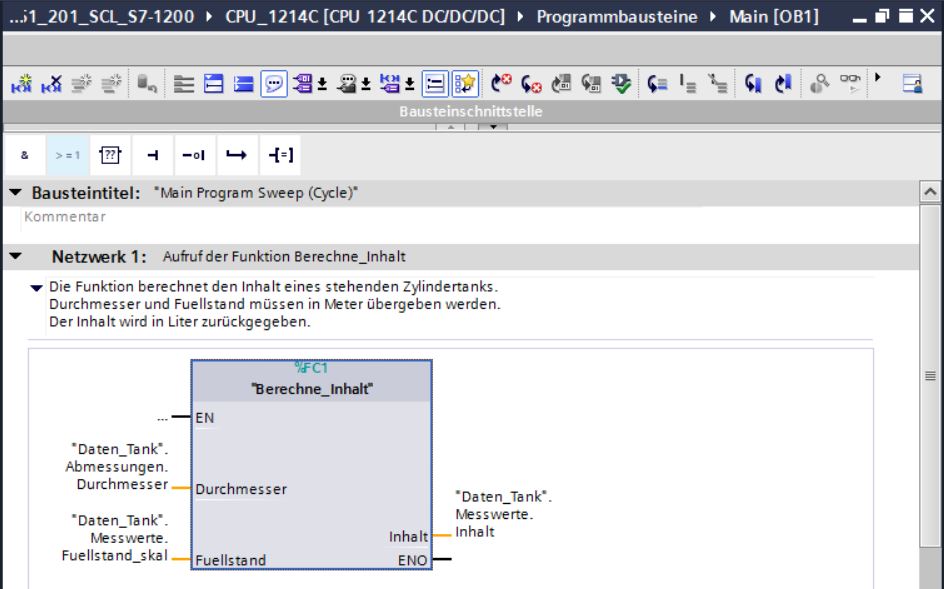
* Öffnen Sie nun den Organisationsbaustein „Main [OB1]“ mit einem Doppelklick.



* Rufen Sie die Funktion „Berechne\_Inhalt“ im ersten Netzwerk auf. Vergeben Sie Netzwerktitel, Kommentar und beschalten Sie die Parameter.

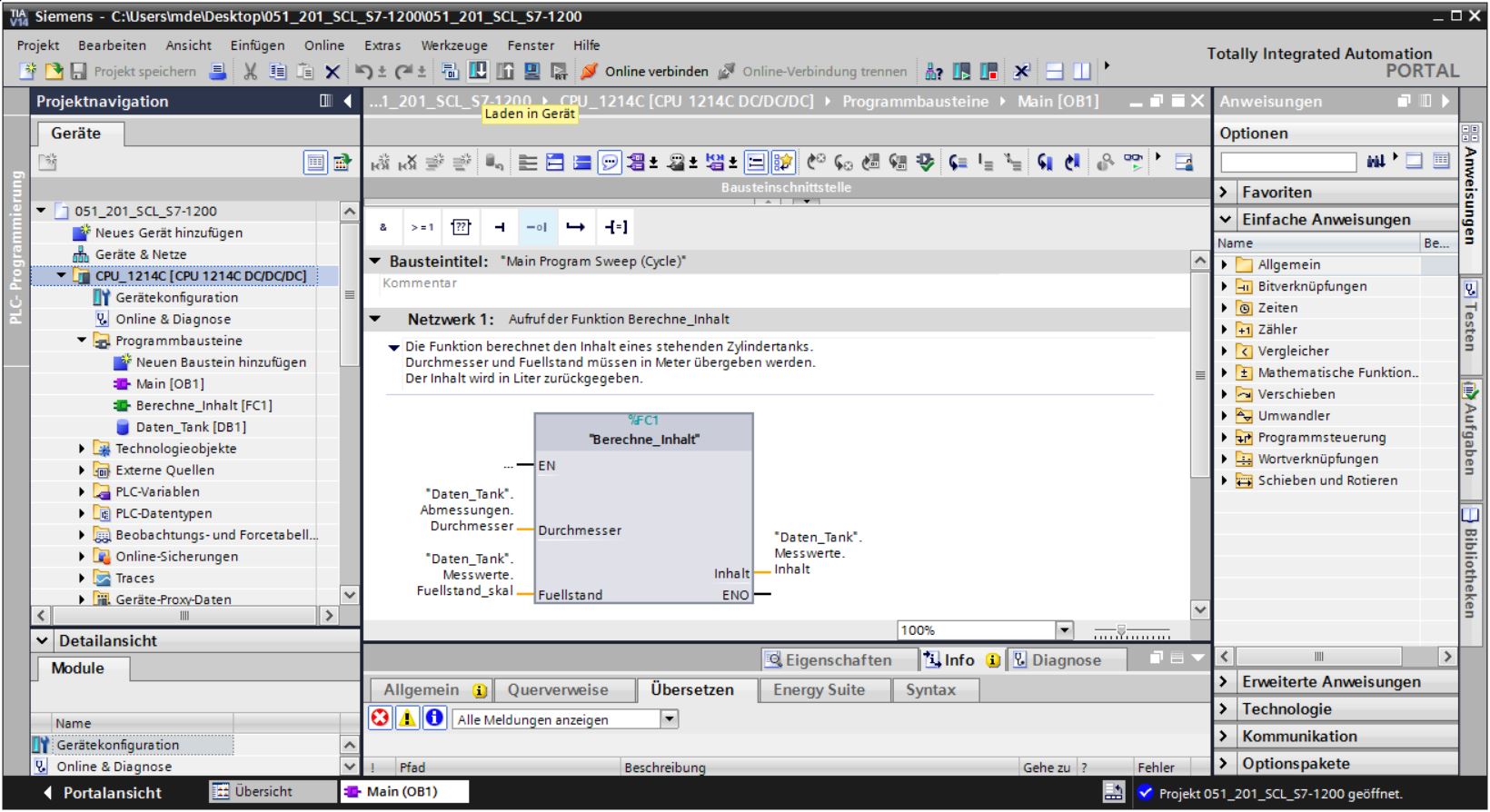
(→ Aufruf „Berechne\_Inhalt“ → Netzwerktitel vergeben → Netzwerkkommentar schreiben → Parameter beschalten)



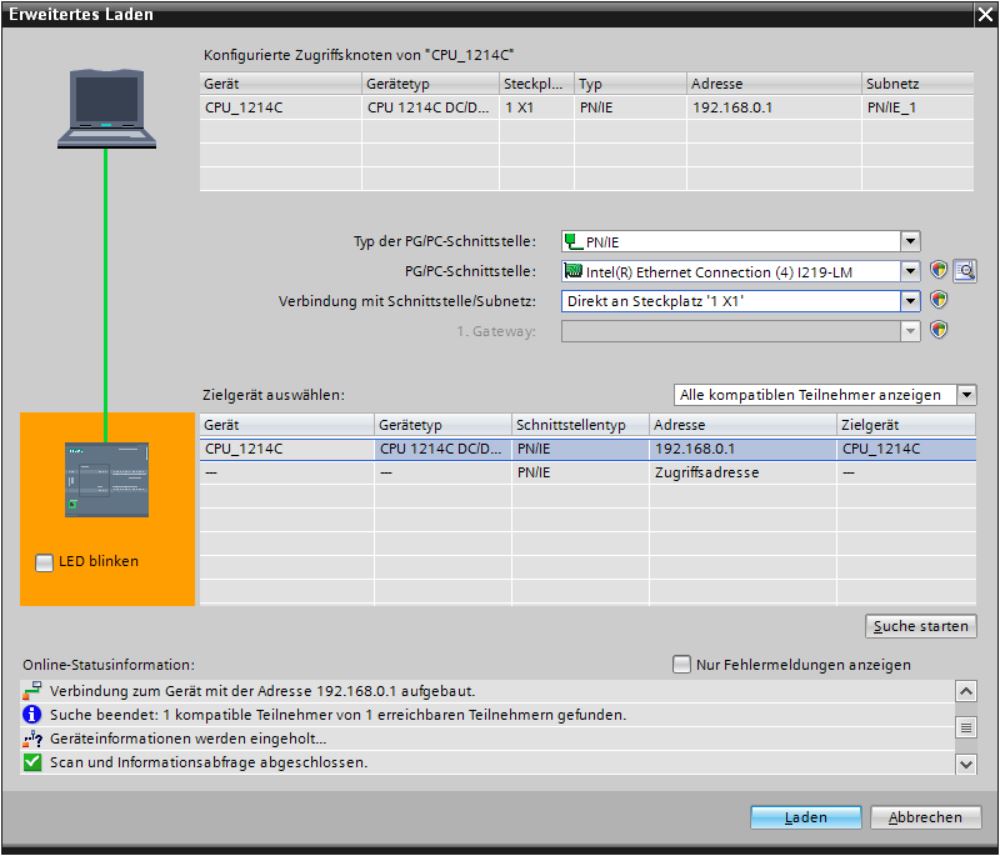


## Programm übersetzen und laden

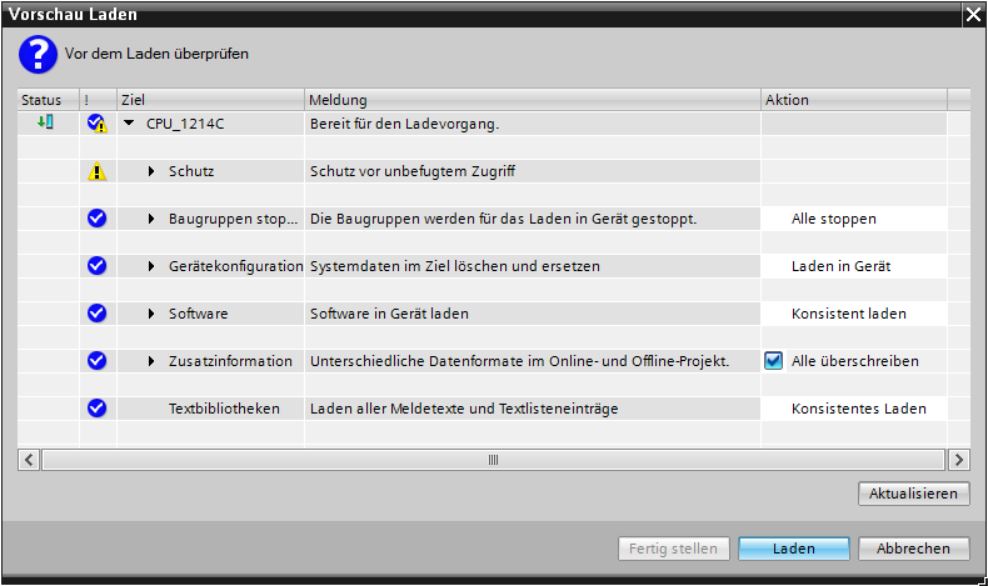
* Klicken Sie auf den Ordner „Programmbausteine“ und übersetzen Sie das gesamte Programm. Nach erfolgreichem Übersetzen speichern Sie Ihr Projekt und laden es in die Steuerung. (→  →  → )



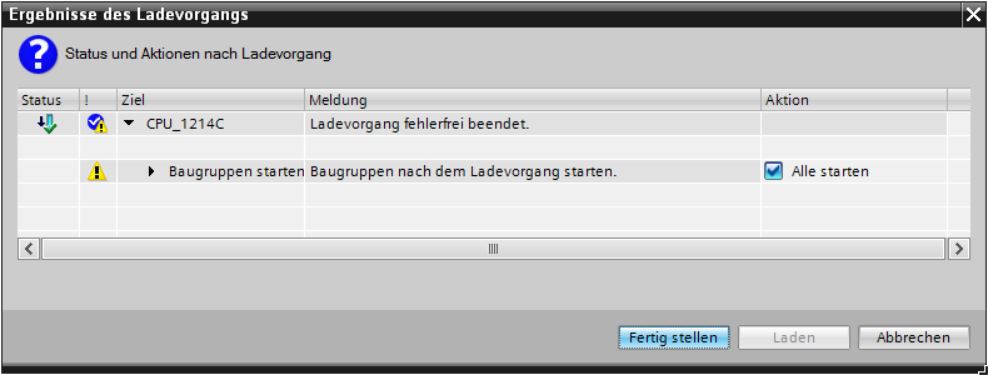
* PG/PC-Schnittstelle auswählen → Subnetz auswählen → Suche starten → Laden



* Eventuell Auswahl treffen → Laden

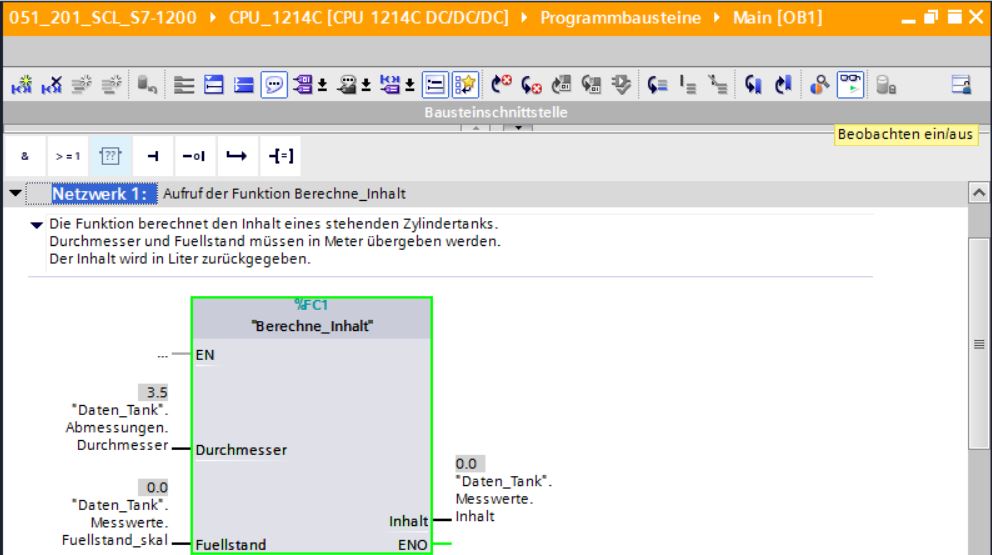


* Fertig stellen



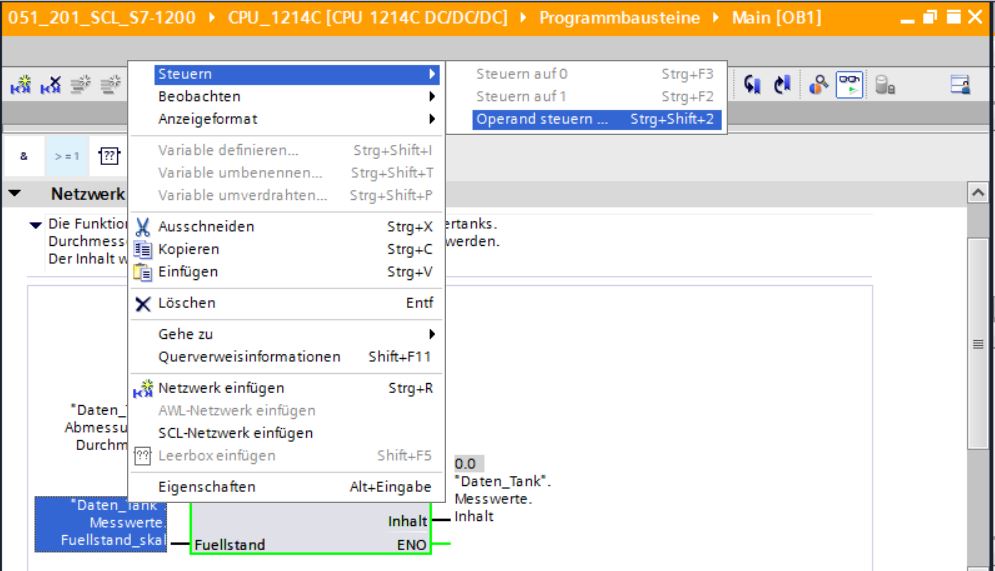
## Organisationsbaustein beobachten und testen

* Klicken Sie im geöffneten OB1 auf das SymbolD:\00_DATA\SIEMENS\Unterlagen\08_Ausbildungsunterlage_TIA-Portal_R1502_dt\032-100 FC-Programmierung\pics\055b.jpg, um den Baustein zu beobachten.

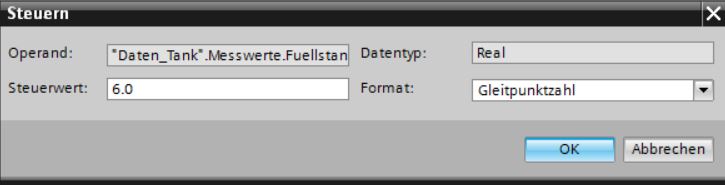


* Testen Sie Ihr Programm, indem Sie einen Wert in die Variable „Fuellstand\_skal“ im Datenbaustein schreiben.

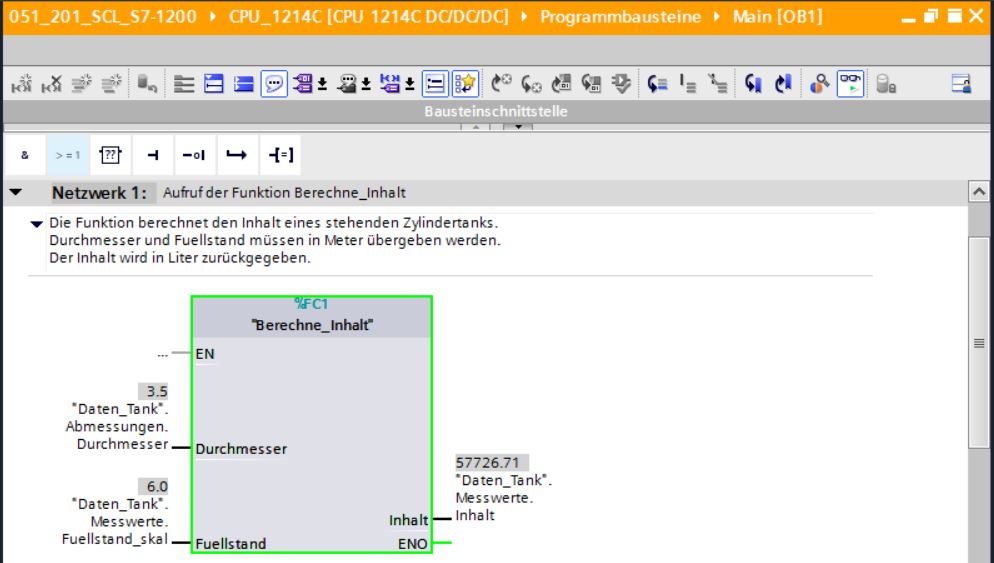
(→ Rechtsklick auf „Fuellstand\_skal“ → Menü „Steuern“ → Operand steuern)



* Wert 6.0 eintragen → OK



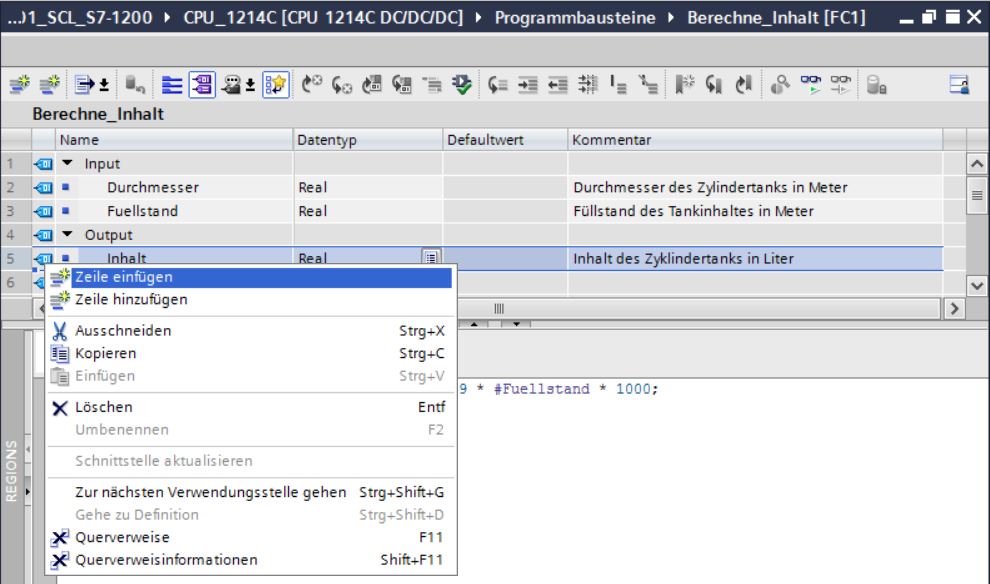
* Überprüfen Sie das Ergebnis auf Richtigkeit.



## Erweiterung der Funktion „Berechne\_Inhalt“

* Öffnen Sie die Funktion „Berechne\_Inhalt“ und fügen Sie durch Rechtsklick, auf die Zeile in der Schnittstelle, eine weitere Zeile bei den Outputparametern ein.

(→ „Berechne\_Inhalt“ öffnen → Rechtsklick auf Zeile 5 → Zeile einfügen)



* Tragen Sie den Parameter „er“ mit Datentyp BOOL und Kommentar ein.

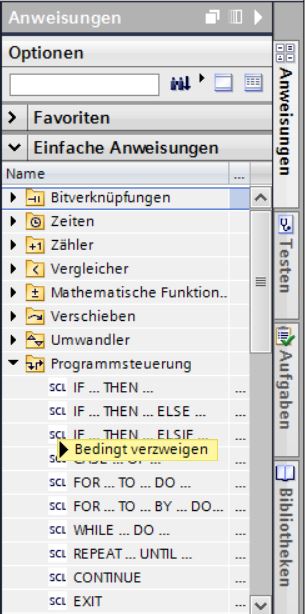


* Nachfolgend fügen Sie auf die gleiche Weise die Input- Variable „Hoehe“ mit Datentyp Real und Kommentar ein.



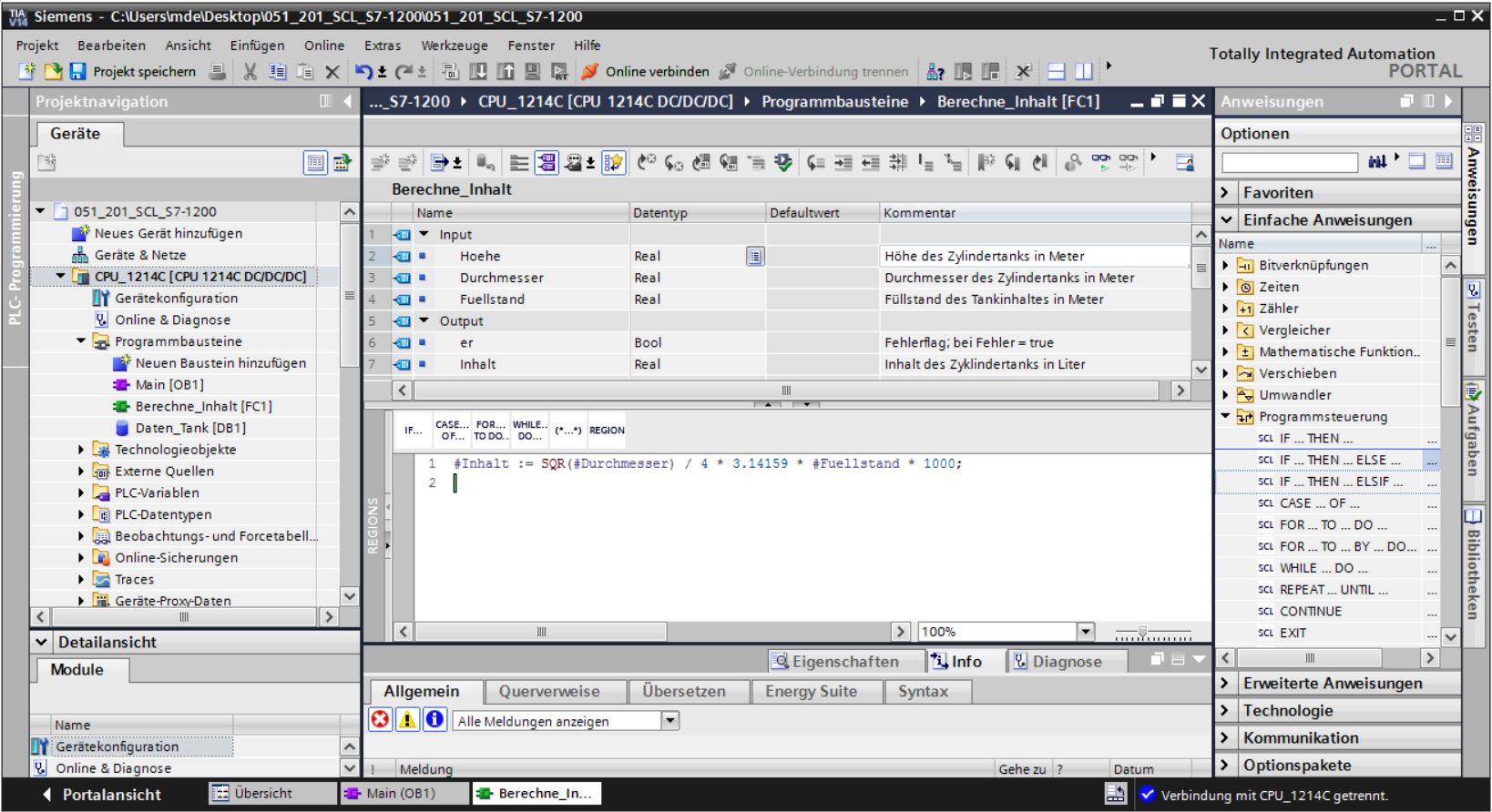
* Navigieren Sie danach zur Kontrollstruktur „IF…THEN…ELSE“ aus dem Ordner „Programmsteuerung“ der einfachen Anweisungen.

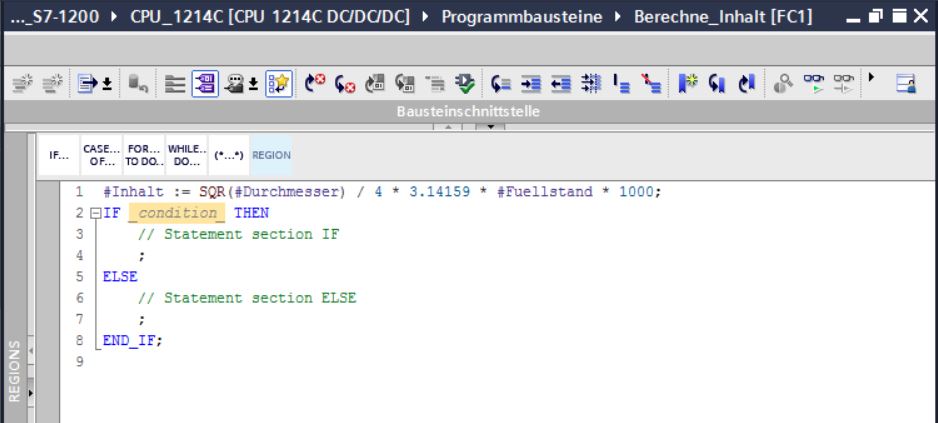
(→ Anweisungen → Einfache Anweisungen → Programmsteuerung → „IF...THEN…ELSE“)



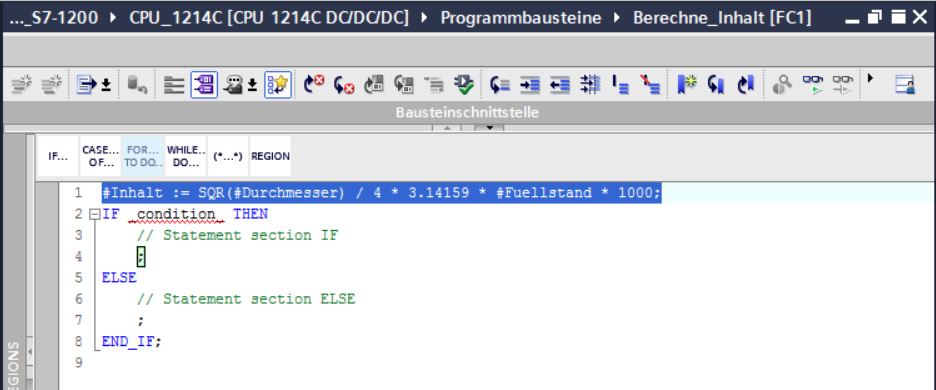
* Ziehen Sie anschließend die Kontrollstruktur „IF...THEN...ELSE“ per Drag&Drop in die zweite Zeile des Programms.

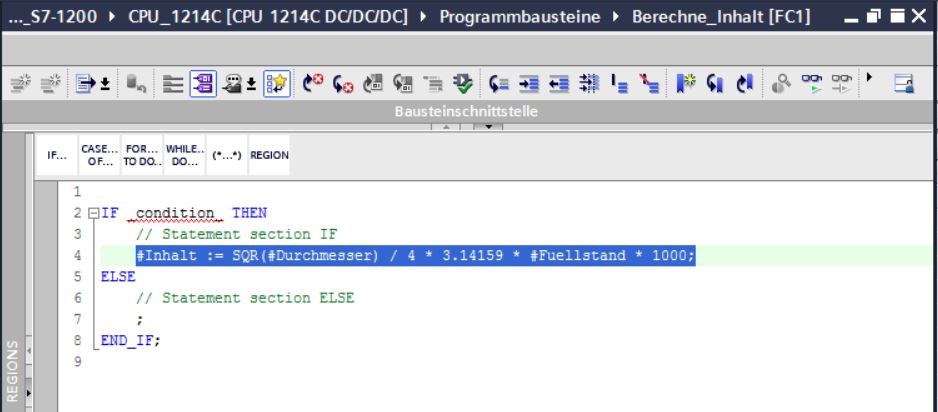
(→ „IF…THEN…ELSE“ → Drag&Drop)





* Markieren Sie die mathematische Formel und ziehen Sie diese per Drag & Drop auf das Semikolon vor dem ELSE. (→ Markieren → Drag & Drop)

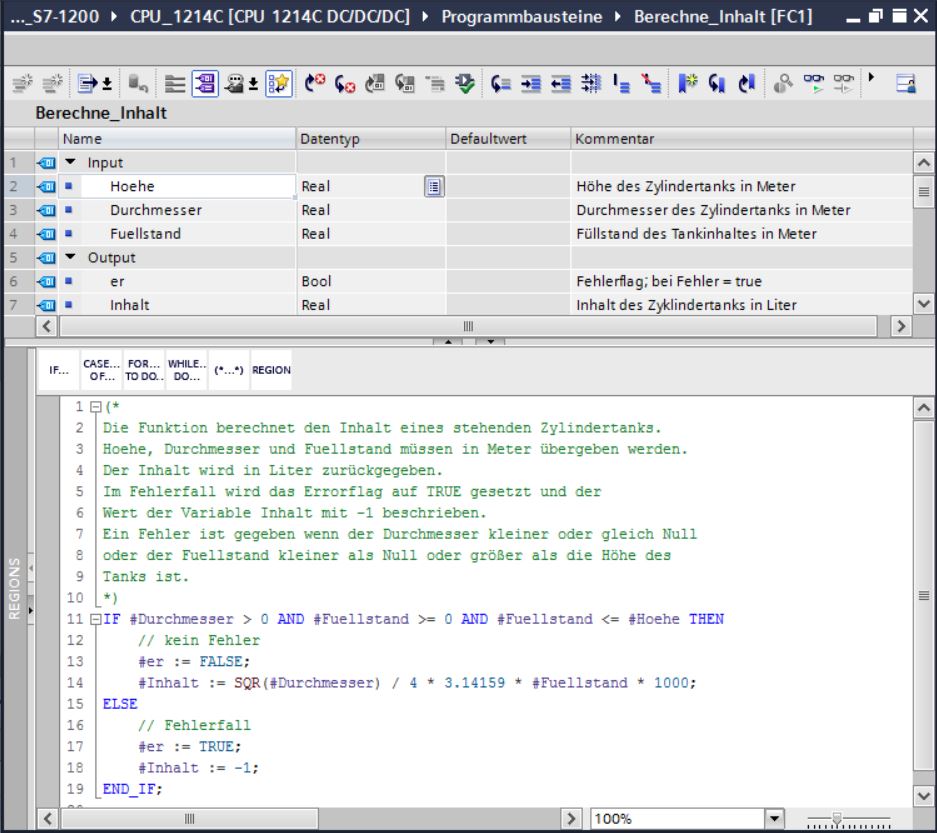




* Vervollständigen Sie die Funktion und überprüfen Ihr Programm durch Übersetzen.  
  (→ Programm ergänzen → )

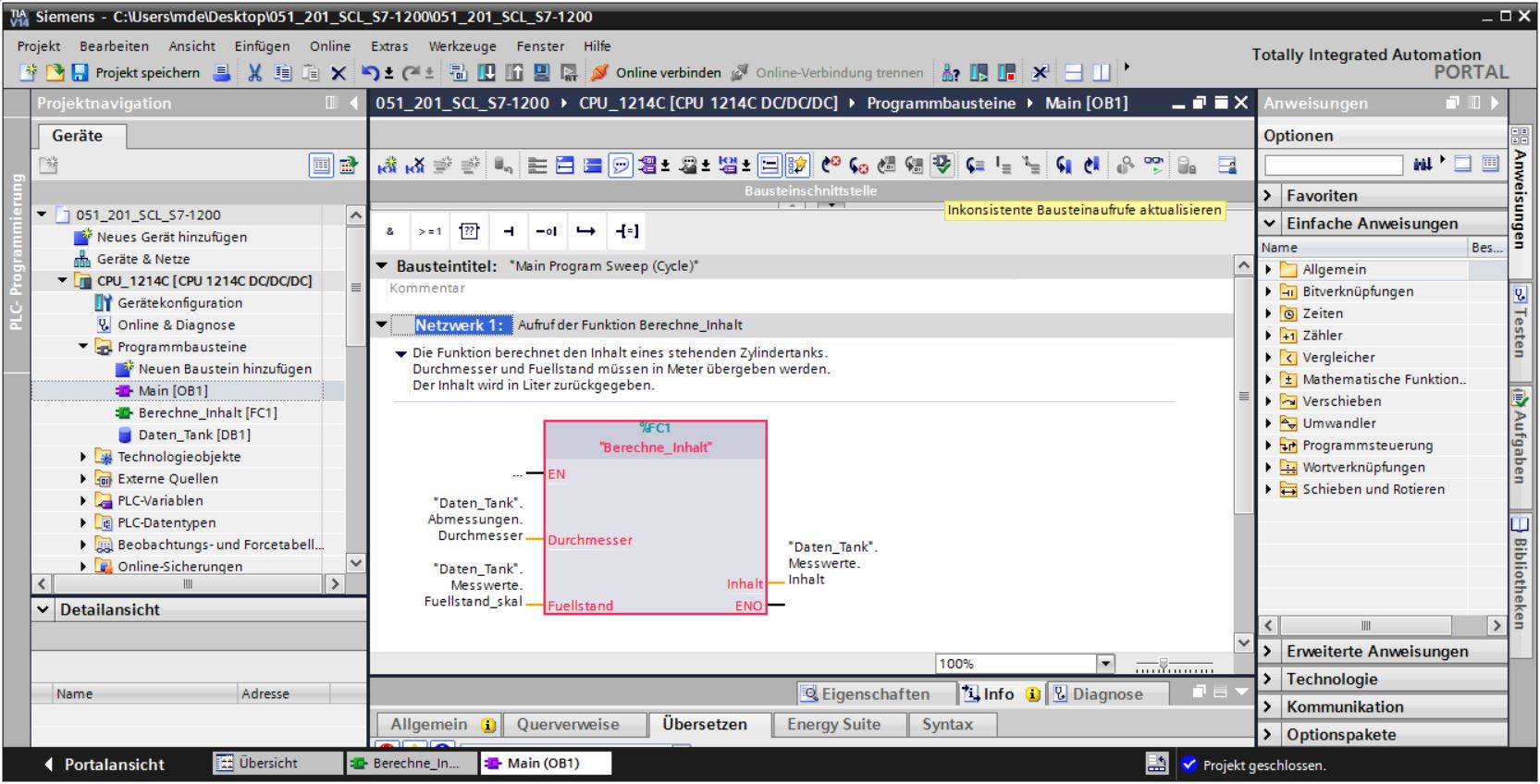


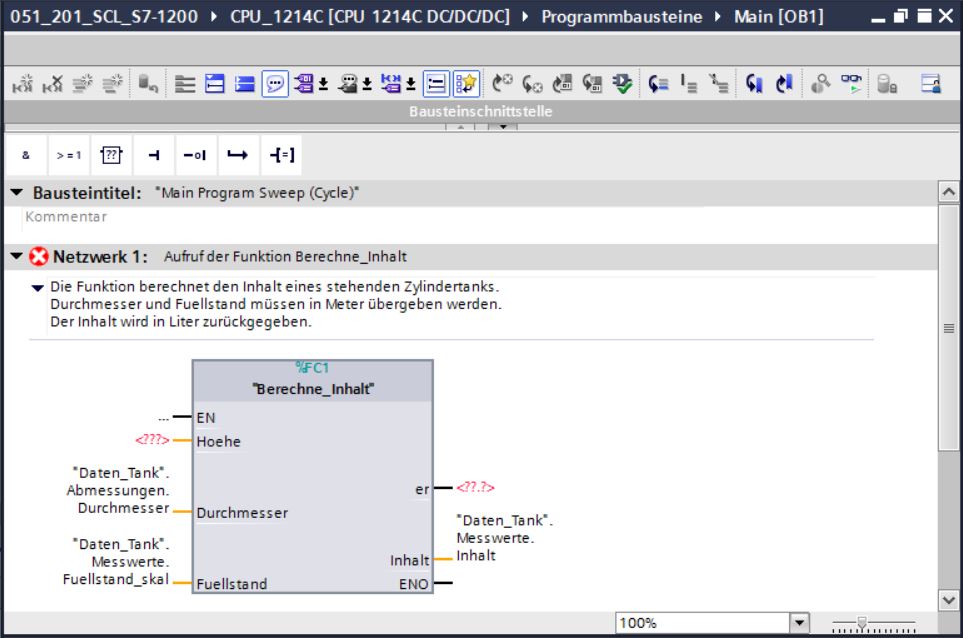
* Kommentare können mit „(\*\*)“ als Blockkommentar und mit „//“ als Zeilenkommentar eingefügt werden. Jetzt können Sie Ihr Programm durch Kommentare ergänzen.

(→ Blockkommentar ab Zeile 1 einfügen → Zeilenkommentar in Zeile 12/16 einfügen) 

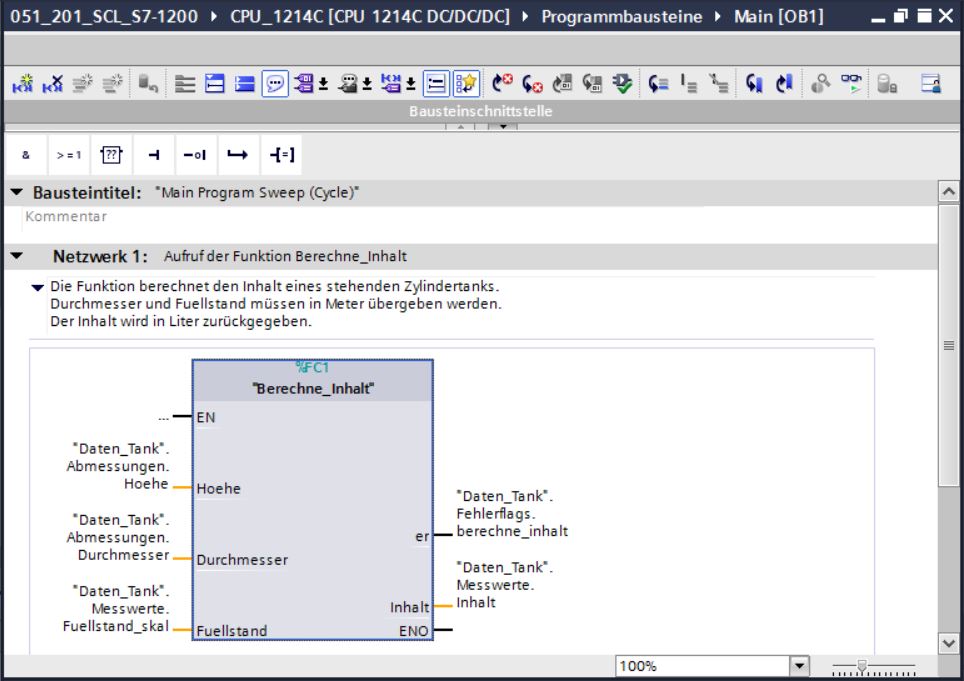
## Organisationsbaustein anpassen

* Öffnen Sie den OB1 und aktualisieren Sie die inkonsistenten Bausteinaufrufe durch Klicken auf . (→ OB1 öffnen → )





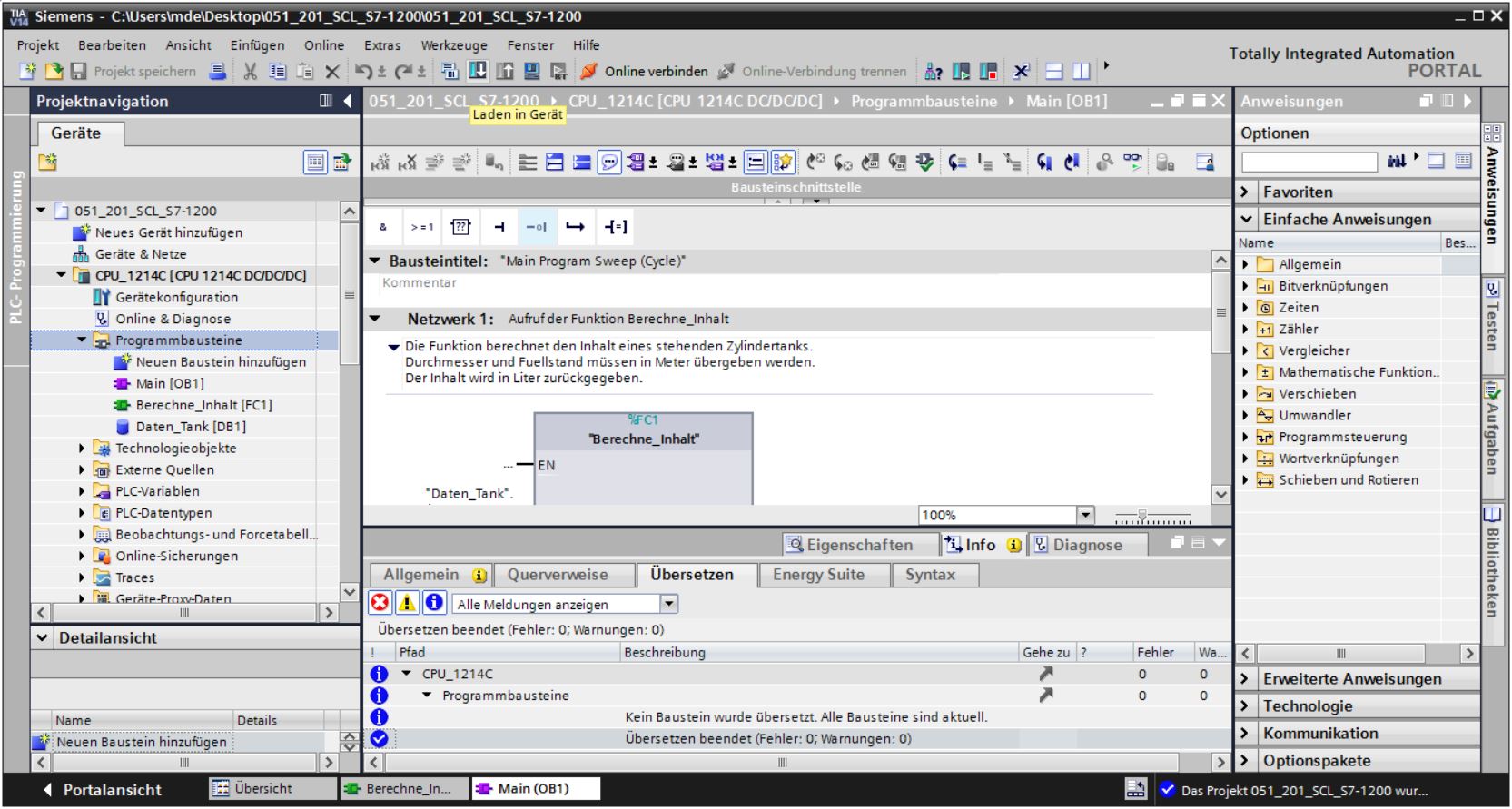
* Ergänzen Sie die Beschaltung der Parameter „er“ und „Hoehe“.



## Programm übersetzen, speichern und laden

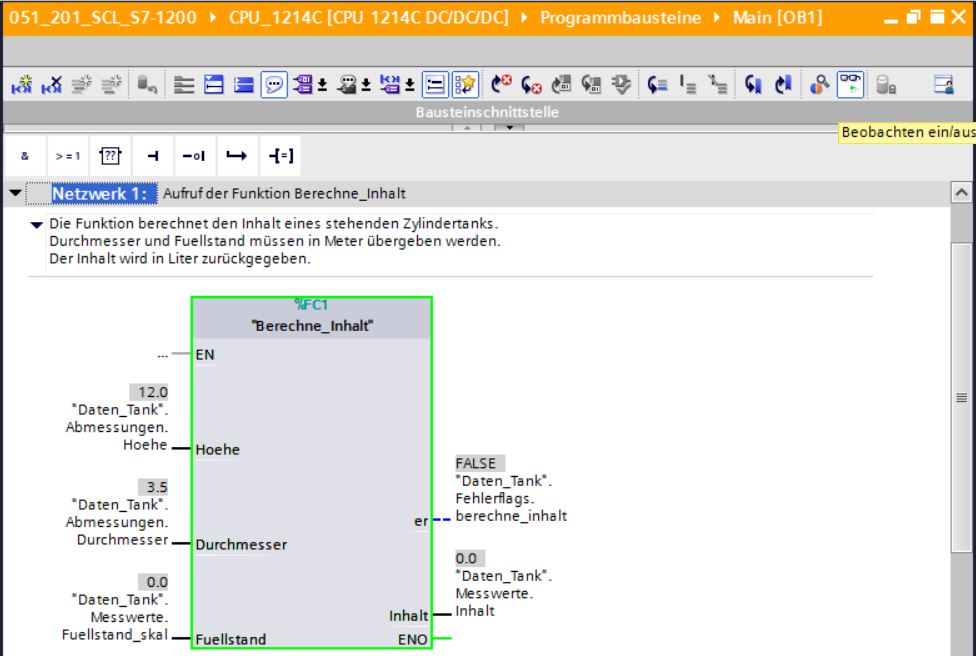
* Klicken Sie auf den Ordner „Programmbausteine“, übersetzen Sie das gesamte Programm und speichern Sie es abschließend. Nach erfolgreichem Übersetzen und Speichern laden Sie das Projekt in die Steuerung.

(→ Programmbausteine →  →  → )



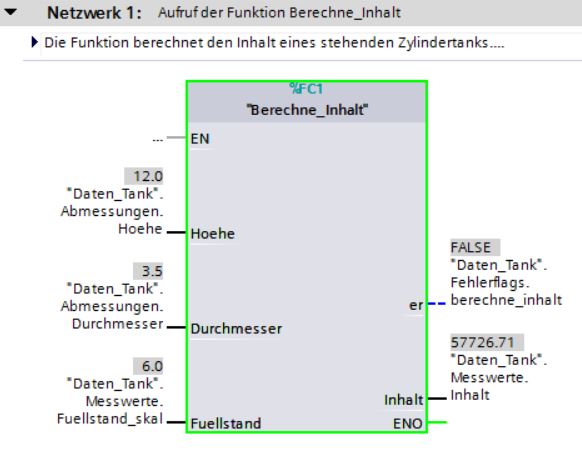
## Organisationsbaustein beobachten und testen

* Klicken Sie im geöffneten OB1 auf das Symbol D:\00_DATA\SIEMENS\Unterlagen\08_Ausbildungsunterlage_TIA-Portal_R1502_dt\032-100 FC-Programmierung\pics\055b.jpg, um den Baustein zu beobachten.



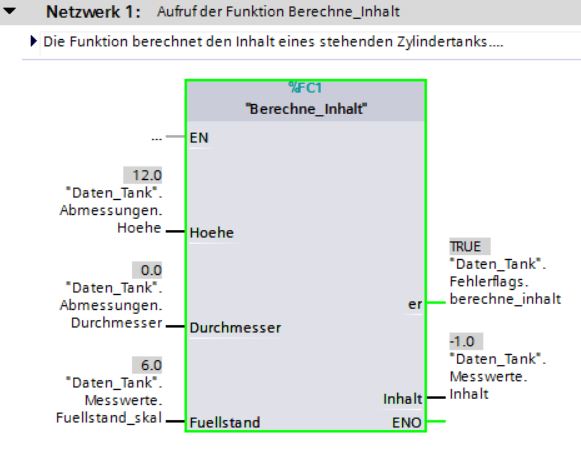
* Testen Sie Ihr Programm, indem Sie einen Wert in die Variable „Fuellstand\_skal“ im Datenbaustein schreiben.

(→ Rechtsklick auf „Fuellstand\_skal“ → Menü „Steuern“ → Operand steuern → Wert 6.0 eingeben → OK → Überprüfen)



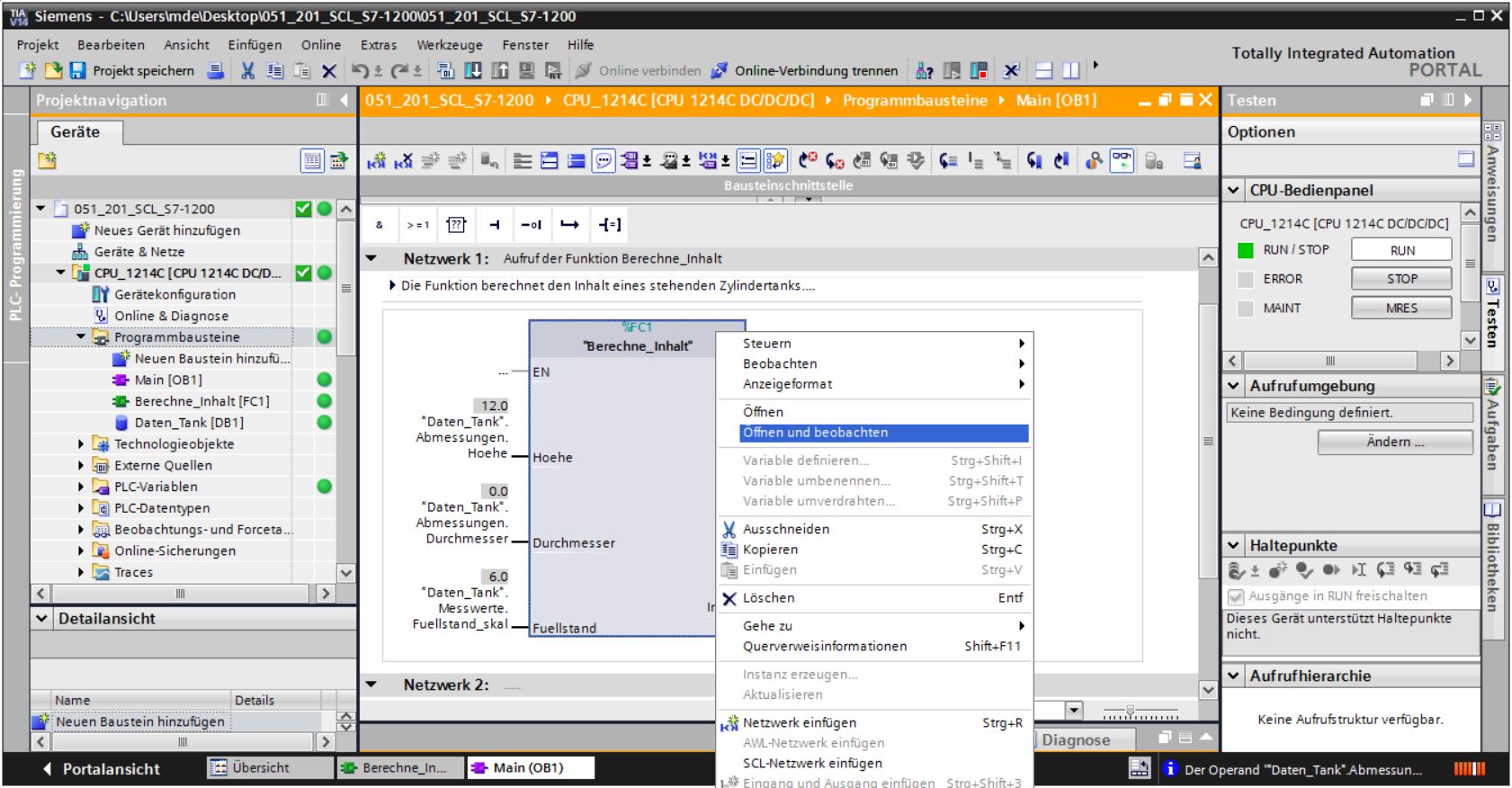
* Testen Sie nachfolgend, ob ein Fehler ausgegeben wird, indem Sie den Durchmesser auf null setzen.

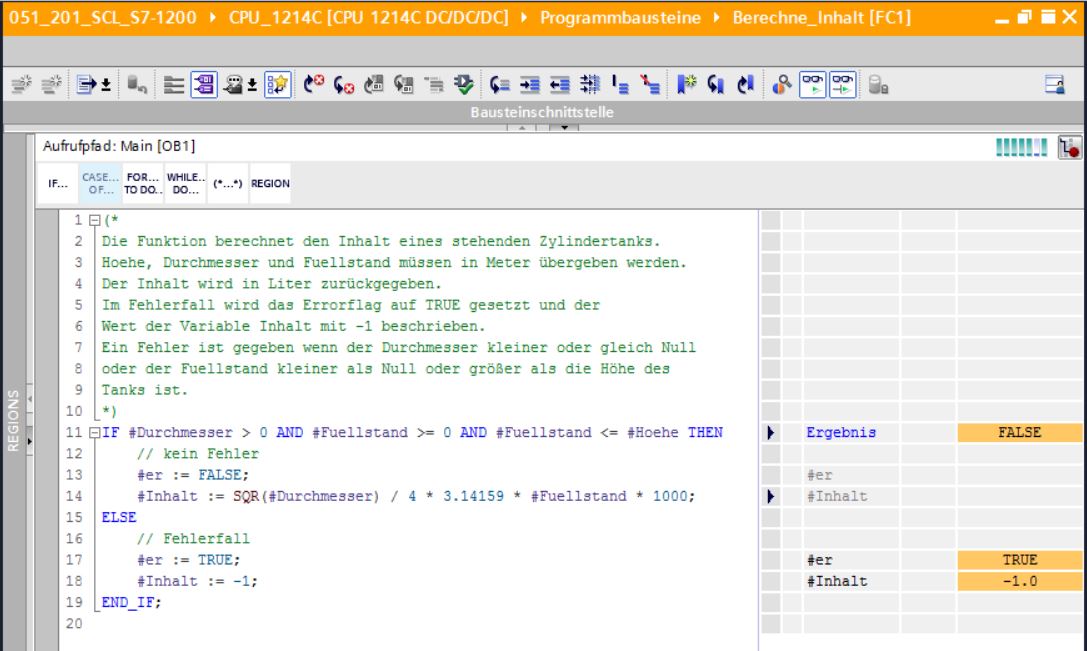
(→ Rechtsklick auf „Durchmesser“ → Menü „Steuern“ → Operand steuern → Wert 0.0 eingeben → OK → Überprüfen)



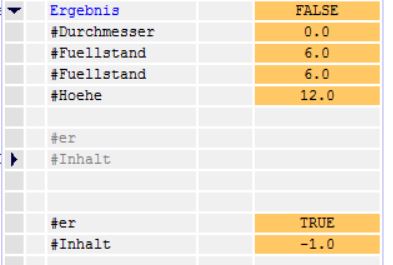
## Funktion „Berechne\_Inhalt“ beobachten und testen

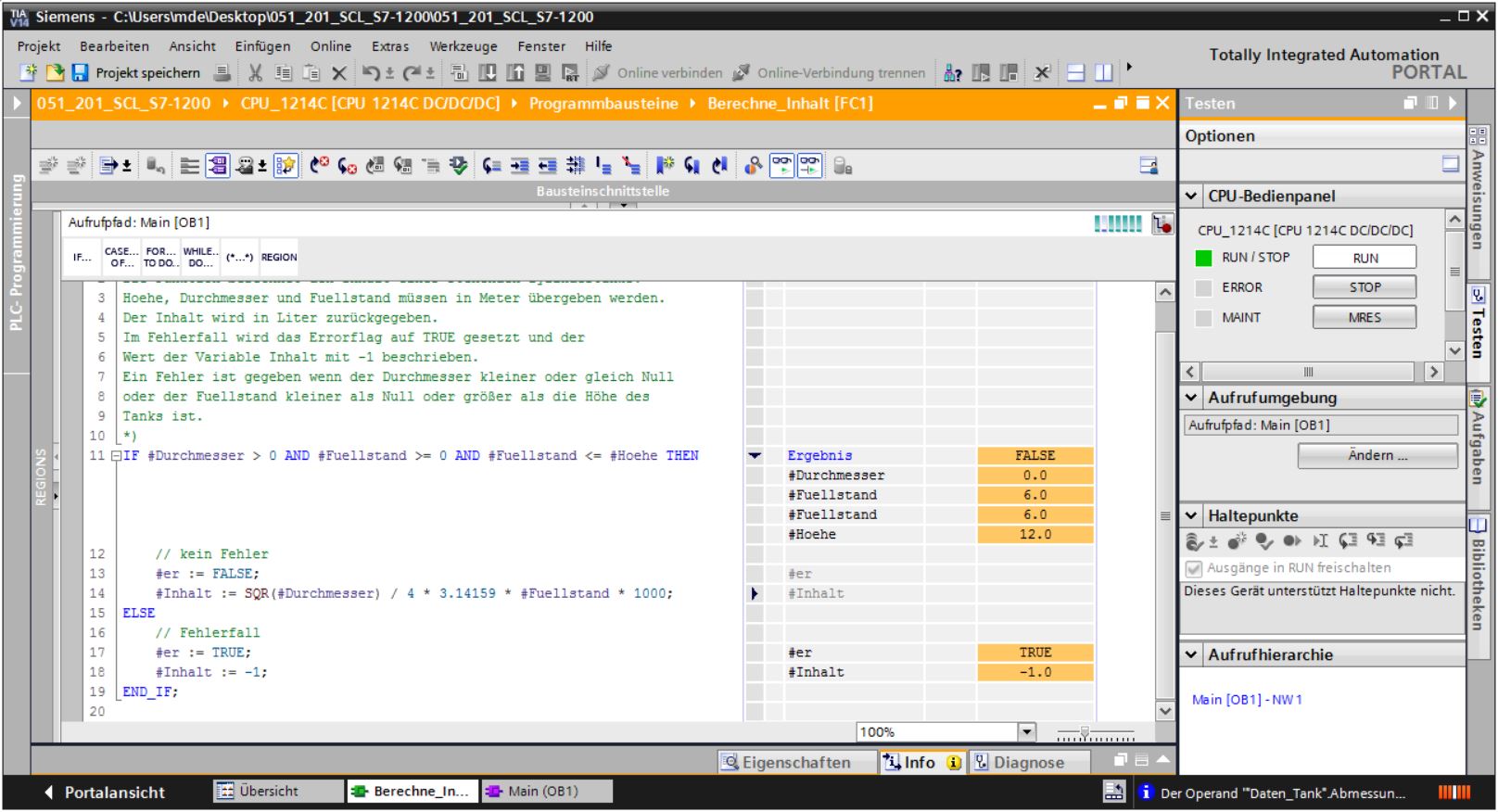
* Öffnen und beobachten Sie schließlich die Funktion „Berechne\_Inhalt“, indem Sie mit Rechtsklick auf die Funktion gehen und den Menüpunkt „Öffnen und beobachten“ auswählen. (→ Rechtsklick auf Funktion → Öffnen und beobachten)



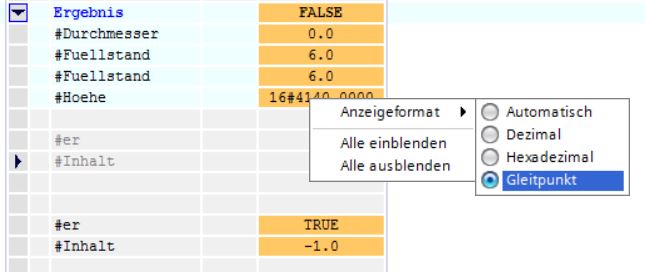


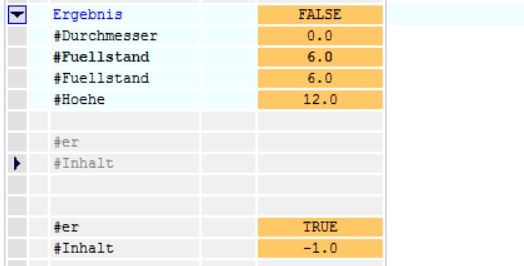
* Sie können die Werte der einzelnen Variablen der IF-Abfrage per Klick auf den schwarzen Pfeil  einblenden. (→)





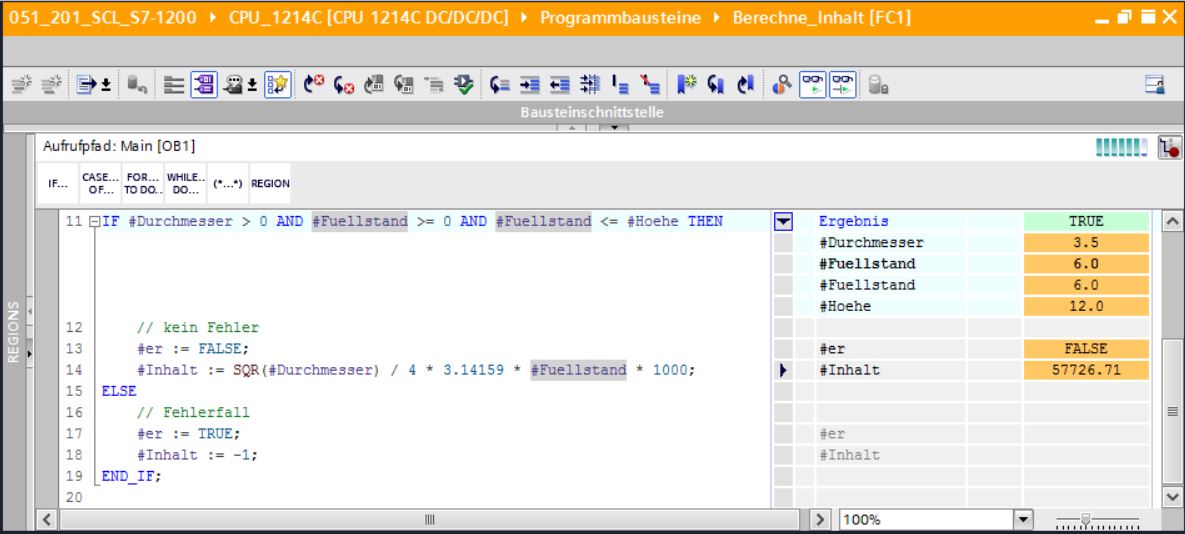
* Das Anzeigeformat kann durch Rechtsklick auf die Variable angepasst werden.   
  (→ Rechtsklick auf Variable → Anzeigeformat → Gleitpunkt)





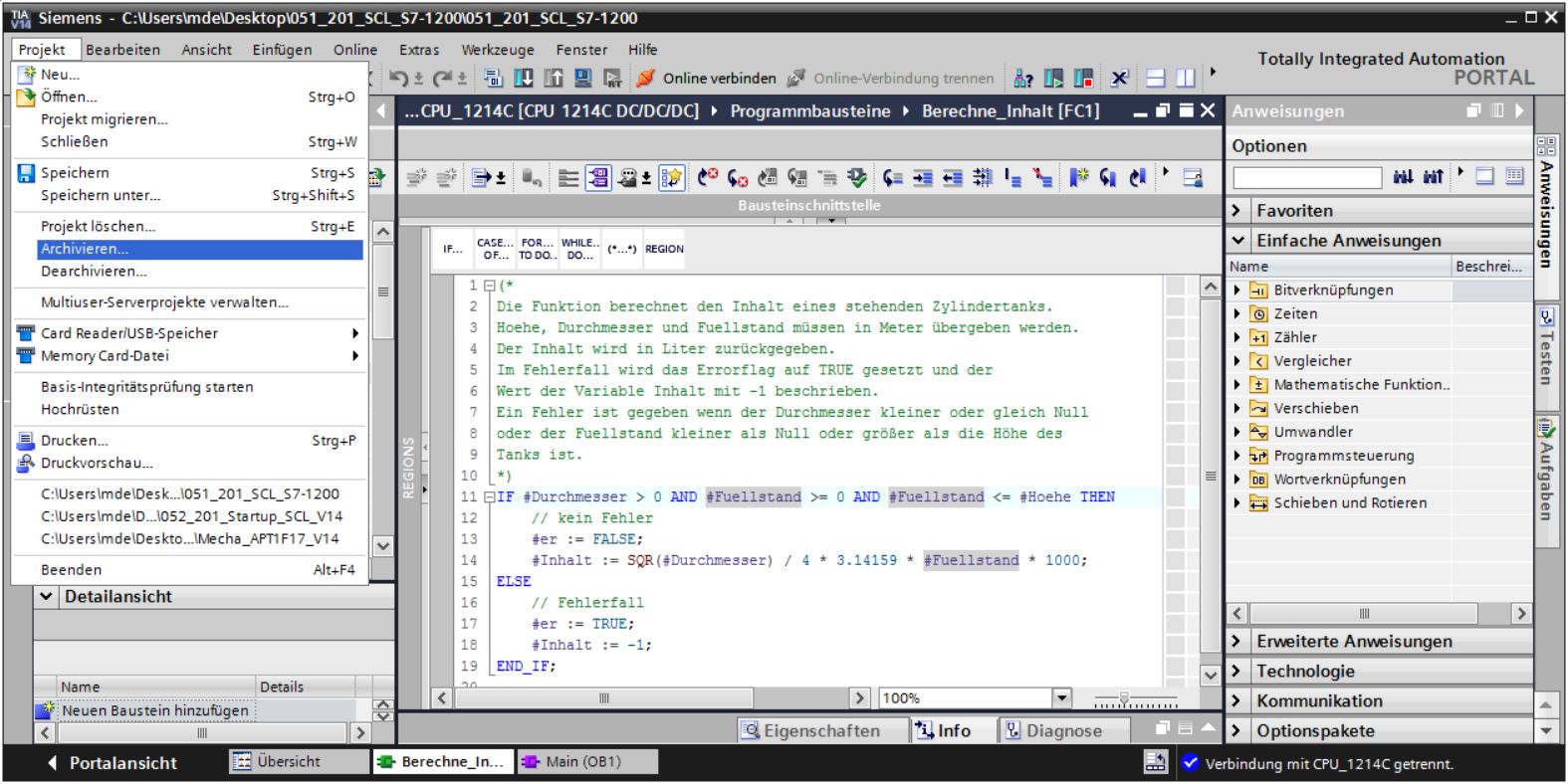
* Testen Sie nun den anderen Zweig der IF-Verzweigung, indem Sie den Durchmesser im OB1 wieder auf 3.5 Meter steuern.

(→ OB1 öffnen → Durchmesser auf 3.5 steuern → Funktion öffnen und beobachten)



## Archivieren des Projektes

* Zum Abschluss soll das komplette Projekt noch archiviert werden. Wählen Sie bitte im Menüpunkt → „Projekt“ → „Archivieren …“ aus. Öffnen Sie den Ordner, in welchem Sie Ihr Projekt archivieren wollen und speichern Sie es als Dateityp „TIA Portal-Projektarchiv“ ab.   
  (→ Projekt → Archivieren → TIA Portal-Projektarchiv → Dateiname: SCE\_DE\_051-201 SCL\_S7-1200… → Archivieren)



# Checkliste

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Beschreibung** | **Geprüft** |
| 1 | Übersetzen erfolgreich und ohne Fehlermeldung |  |
| 2 | Laden erfolgreich und ohne Fehlermeldung |  |
| 3 | Operand steuern (Durchmesser = 0.0)  Ergebnis Variable Inhalt = -1  Ergebnis Variable „er“ = TRUE |  |
| 4 | Operand steuern (Durchmesser = 3.5 und  Fuellstand\_skal = 0)  Ergebnis Inhalt = 0  Ergebnis Variable „er“ = FALSE |  |
| 5 | Operand steuern (Fuellstand\_ska l= 6.0)  Ergebnis Inhalt = 57726.72  Ergebnis Variable „er“ = FALSE |  |
| 6 | Operand steuern (Fuellstand\_skal = 12.0)  Ergebnis Inhalt = 115453.4  Ergebnis Variable „er“ = FALSE |  |
| 7 | Operand steuern (Fuellstand\_skal = 14.0)  Ergebnis Inhalt = -1  Ergebnis Variable „er“ = TRUE |  |
| 8 | Projekt erfolgreich archiviert |  |

# Übung

## Aufgabenstellung – Übung

In dieser Übung wird eine Funktion „Skalieren“ programmiert. Das Programm soll allgemeingültig für jegliche positiven Analogwerte anwendbar sein. In unserer Beispielaufgabe „Tank“ wird der Füllstand über einen Analogsensor eingelesen und mittels dieser Funktion skaliert im Datenbaustein abgelegt.

Im Fehlerfall soll der Baustein das Errorflag „er“ auf TRUE und als Ergebnis den Parameter „Analog\_skal“ auf null setzen. Ein Fehlerfall besteht, wenn der Parameter „mx“ kleiner oder gleich „mn“ ist.

Die Funktion muss folgende Parameter beinhalten.

|  |  |  |
| --- | --- | --- |
| **Input** | **Datentyp** | **Kommentar** |
| Analog\_per | INT | Analogwert von der Peripherie zwischen 0..27648 |
| mx | REAL | Maximum des neuen Maßstabs |
| mn | REAL | Minimum des neuen Maßstabs |
| **Output** |  |  |
| er | BOOL | Fehlerflag, kein Fehler = 0, Fehler = 1 |
| Analog\_skal | REAL | Analogwert skaliert zwischen mn..mx  Im Fehlerfall = 0 |

Zur Lösung der Aufgabe wird folgende Formel verwendet:



Für diese Übungsaufgabe ist ein Analogsignal notwendig. Der hierzu verwendete Operand muss in die PLC-Variablentabelle eingetragen werden.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Datentyp** | **Adresse** | **Kommentar** |
| B1 | INT | %EW64 | Füllstand zwischen 0..27648 |

## Planung

Planen Sie nun selbstständig die Umsetzung der Aufgabenstellung!

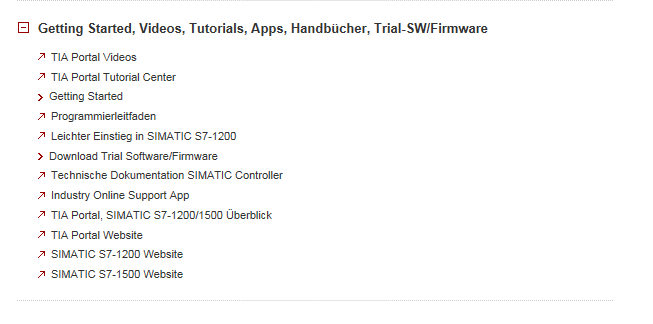
## Checkliste – Übung

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Beschreibung** | **Geprüft** |
| 1 | Operand in PLC-Variablentabelle eingefügt |  |
| 2 | Funktion FC: „Skalieren“ erstellt |  |
| 3 | Schnittstelle definiert |  |
| 4 | Funktion programmiert |  |
| 5 | Funktion „Skalieren“ ins Netzwerk 1 des OB1 eingefügt |  |
| 6 | Eingangsvariablen verschaltet |  |
| 7 | Ausgangsvariablen verschaltet |  |
| 8 | Übersetzen erfolgreich und ohne Fehlermeldung |  |
| 9 | Laden erfolgreich und ohne Fehlermeldung |  |
| 10 | Analogwert für Füllstand auf null gesetzt  Ergebnis Fuellstand\_skal = 0  Ergebnis er = FALSE |  |
| 11 | Analogwert für Füllstand auf 27648 gesetzt  Ergebnis Fuellstand\_skal = 12.0  Ergebnis er = FALSE |  |
| 12 | Analogwert für Füllstand auf 13824  Ergebnis Fuellstand\_skal = 6.0  Ergebnis er = FALSE |  |
| 13 | Operand steuern (mx = 0.0)  Ergebnis Fuellstand\_skal = 0  Ergebnis Variable er = TRUE |  |
| 14 | Projekt erfolgreich archiviert |  |

# Weiterführende Information

Zur Einarbeitung bzw. Vertiefung finden Sie als Orientierungshilfe weiterführende Informationen, wie z.B. Getting Started, Videos, Tutorials, Apps, Handbücher, Programmierleitfaden und Trial Software/Firmware, unter nachfolgendem Link:   
  
[siemens.com/sce/s7-1200](http://www.siemens.com/sce/s7-1200)

**Voransicht „Weiterführende Informationen“**



Weitere Informationen

Siemens Automation Cooperates with Education  
**siemens.de/sce**

SCE Lehrunterlagen  
**siemens.de/sce/module**

SCE Trainer Pakete  
**siemens.de/sce/tp**

SCE Kontakt Partner   
**siemens.de/sce/contact**

Digital Enterprise  
**siemens.de/digital-enterprise**

Industrie 4.0   
**siemens.de/zukunft-der-industrie**

Totally Integrated Automation (TIA)  
**siemens.de/tia**

TIA Portal  
**siemens.de/tia-portal**

SIMATIC Controller  
**siemens.de/controller**

SIMATIC Technische Dokumentation   
**siemens.de/simatic-doku**

Industry Online Support  
**support.industry.siemens.com**

Katalog- und Bestellsystem Industry Mall   
**mall.industry.siemens.com**

Siemens AG  
Digital Factory   
Postfach 4848  
90026 Nürnberg  
Deutschland

Änderungen und Irrtümer vorbehalten  
© Siemens AG 2018

**siemens.de/sce**

