Ausbildungsunterlage für die durchgängige Automatisierungslösung Totally Integrated Automation (T I A)

MODUL C2

Hochsprachenprogrammierung mit S7-SCL

Wir danken der Fa. Michael Dziallas Engineering und den Lehrkräften von beruflichen Schulen sowie weiteren Personen für die Unterstützung bei der Erstellung der Unterlage

SEITE:

Die folgenden Symbole führen durch dieses Modul:

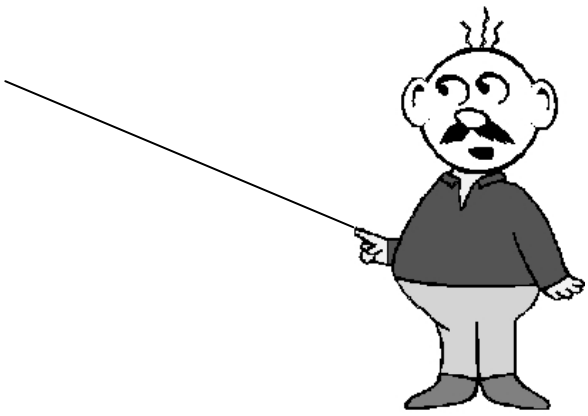
Information

Installation

Programmierung

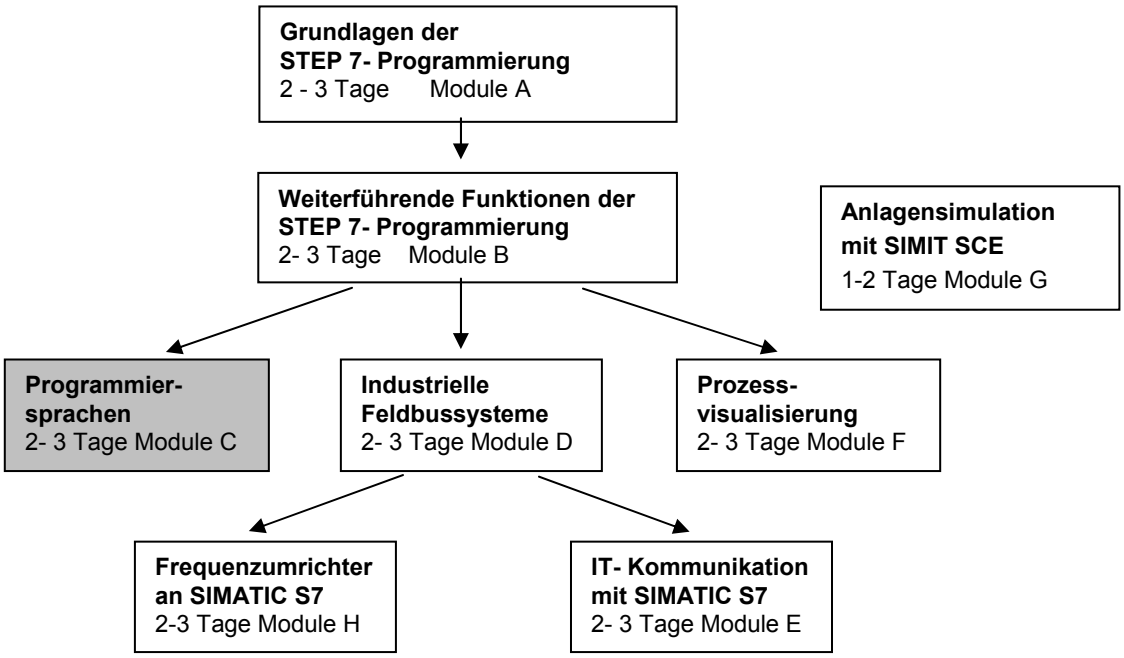
Beispielaufgabe

Hinweise



1. VORWORT

Das Modul C2 ist inhaltlich der Lehreinheit **, Programmiersprachen'** zugeordnet.



Lernziel:

Der Leser soll in diesem Modul die grundlegenden Funktionen der S7-SCL-Entwicklungsumgebung kenne lernen. Weiterhin sollen Testfunktionen zur Beseitigung logischer Programmierfehler gezeigt werden.

Voraussetzungen:

Für die erfolgreiche Bearbeitung dieses Moduls wird folgendes Wissen vorausgesetzt:

- Kenntnisse in der Handhabung von Windows
- Grundlagen der SPS- Programmierung mit STEP 7 (z.B. Modul A3 - ‚Startup' SPS- Programmierung mit STEP 7)
- Grundlagenkenntnisse über Hochsprachenprogrammierung wie z.B. Pascal.

Benötigte Hardware und Software

1 PC, Betriebssystem Windows XP Professional mit SP2 oder SP3 / Vista 32 Bit Ultimate und Business / Server 2003 SP2 mit 600MHz (nur XP) / 1 GHz und 512MB (nur XP) / 1 GB RAM, freier Plattenspeicher ca. 650 - 900 MB, MS-Internet-Explorer 6.0 und Netzwerkkarte

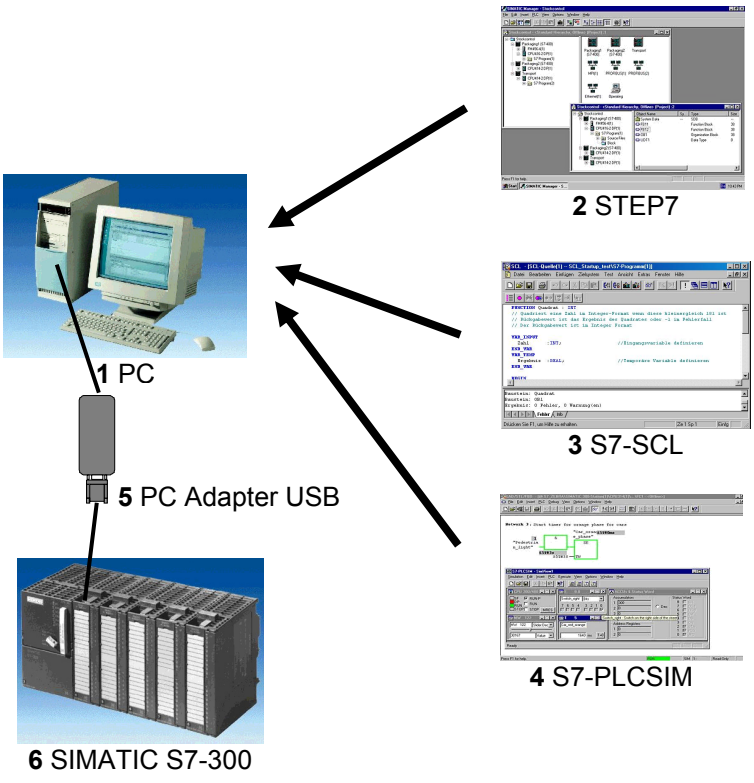
2 Software STEP7 V 5.4

3 Software: S7-SCL V5.x

4 Software: S7-PLCSIM V5.x

3 MPI- Schnittstelle für den PC (z.B. PC Adapter USB)

6 SPS SIMATIC S7-300 mit mindestens einer digitalen Ein- und Ausgabebaugruppe. Die Eingänge müssen auf ein Schaltfeld herausgeführt sein.
Beispielkonfiguration:
- Netzteil: PS 307 2A
- CPU: CPU 314
- Digitale Eingänge: DI 16x DC24V
- Digitale Ausgänge: DO 16x DC24V / 0,5 A



2. HINWEISE ZUR PROGRAMMIERSPRACHE S7-SCL

S7-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 Ablaufsprache SFC „Sequential Function Chart“. S7-SCL enthält neben Hochsprachenelementen auch typische Elemente der SPS wie Eingänge, Ausgänge, Zeiten, Merker, Bausteinaufrufe usw. als Sprachelemente. Sie unterstützt das Bausteinkonzept von STEP 7 und ermöglicht daher neben AWL, KOP und FUP die normkonforme Programmierung von Bausteinen. D.h. S7-SCL ergänzt und erweitert die Programmiersoftware STEP 7 mit ihren Programmiersprachen KOP, FUP und AWL.

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

Bausteine, die mit S7-SCL programmiert sind, können Sie mit AWL-, KOP- und FUP- Bausteinen mischen. Das bedeutet, dass ein mit S7- SCL programmierter Baustein einen anderen Baustein, der in AWL, KOP oder FUP programmiert ist, aufrufen kann. Entsprechend können S7-SCL Bausteine auch in AWL-, KOP- und FUP- Programmen aufgerufen werden

S7-SCL-Bausteine können im konkreten Anwendungsfall in die STEP 7-Programmiersprache AWL rückübersetzt werden. Die Rückübersetzung nach S7-SCL ist nicht möglich.
Beim Übersetzungsvorgang werden aus dem zuvor editierten Programm die Bausteine erzeugt, die ab der CPU 314 auf allen CPUs des Automatisierungssystems S7-300/400 ablauffähig sind.

Die Testfunktionen von S7-SCL ermöglichen die Suche nach logischen Programmierfehlern in einer fehlerfreien Übersetzung. Die Fehlersuche erfolgt dabei in der Quellsprache.

3. INSTALLATION DER SOFTWARE S7-SCL

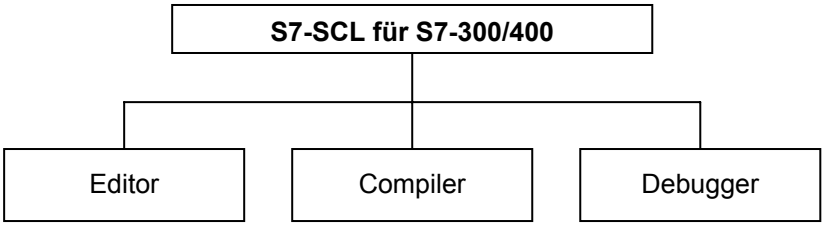
S7-SCL ist ein Optionspaket zu STEP 7, setzt also voraus, dass STEP 7 bereits auf Ihrem Rechner installiert ist. (Siehe Modul A2 – Installation von STEP 7 V5.x / Handhabung der Autorisierung).
S7-SCL wird auf CD-ROM mit einer beiliegenden Diskette ausgeliefert, wobei die Diskette den License Key (Autorisierung) enthält, die auf den PC übertragen werden muss und die Nutzung von S7-SCL erst ermöglicht.
Diese kann, um auf einem anderen PC genutzt zu werden, auch wieder auf die Diskette oder USB-Stick zurückgeholt werden. Ab der STEP 7 Professional V5.3 kann diese Lizenz auch über ein Netzwerk verwaltet werden. Zum Thema Installation und Übertragung der Autorisierungen sehen Sie bitte auch Modul A2 – Installation von STEP 7 V5.x / Handhabung der Autorisierung.

Um nun S7-SCL zu installieren, gehen Sie bitte folgendermaßen vor:

1. Legen Sie die CD von S7-SCL in das CD- ROM- Laufwerk ein.
2. Das Setup-Programm wird nun automatisch gestartet. Falls nicht, starten Sie es, indem Sie auf die Datei ‚→ **setup.exe**' doppelklicken.
   Das Setup-Programm führt Sie durch die gesamte Installation von S7-SCL.
3. Für die Nutzung von S7-SCL ist auf Ihrem Rechner ein License Key (Autorisierung), d.h. eine Nutzungsberechtigung, erforderlich. Diese müssen Sie von der Autorisierungsdiskette auf den Rechner übertragen.
   Dies geschieht am Ende der Installation. Dort werden Sie in einem Dialogfenster vom Setup-Programm gefragt, ob Sie die Autorisierung durchführen wollen. Wenn Sie ‚**Ja'** wählen, müssen Sie nur noch die Autorisierungsdiskette einlegen und die Berechtigung wird auf Ihren Rechner übertragen.

4 DIE ENTWICKLUNGSUMGEBUNG S7-SCL

Zur Verwendung und zum Einsatz von S7-SCL gibt es eine Entwicklungsumgebung, die sowohl auf spezifische Eigenschaften von S7-SCL als auch auf STEP 7 abgestimmt ist. Diese Entwicklungsumgebung besteht aus einem Editor, einem Compiler und einem Debugger.



Editor

Der S7-SCL-Editor ist ein Texteditor, mit dem beliebige Texte bearbeitet werden können. Die zentrale Aufgabe, die Sie mit ihm durchführen, ist das Erzeugen und Bearbeiten von Quelldateien für STEP 7-Programme. In einer Quelldatei können Sie einen oder mehrere Bausteine programmieren. Während der Eingabe erfolgt keine Syntaxprüfung.

Folgende Möglichkeiten bietet der Editor:
- Editieren einer kompletten Quelldatei mit einem oder mehreren Bausteinen.
- Editieren einer Übersetzungssteuerdatei, mit der Sie das Übersetzen mehrerer Quelldateien automatisieren können.
- Benutzen zusätzlicher Funktionen, die Ihnen eine Bearbeitung des Quelltextes ermöglichen, z.B. Suchen und Ersetzen.
- Einstellung des Editors nach Ihren Anforderungen, z.B. durch syntaxgerechtes Einfärben der verschiedenen Sprachelemente.

Compiler

Nachdem Sie ihre Quelldateien mit dem Editor erstellt haben, müssen Sie diese in S7-Bausteine übersetzt werden.

Folgende Möglichkeiten bietet der Compiler:
- Übersetzen einer S7-SCL-Quelldatei mit mehreren Bausteinen in einem Übersetzungslauf.
- Übersetzen mehrerer S7-SCL-Quelldateien mittels einer Übersetzungssteuerdatei, die die Namen der Quelldateien enthält.
- Selektives Übersetzen einzelner Bausteine aus einer Quelle.
- Einstellen des Compilers nach Ihren Anforderungen.
- Anzeigen aller Fehler und Warnungen, die beim Übersetzen auftreten.
- Lokalisieren der fehlerhaften Stelle im Quelltext, optional mit Fehlerbeschreibung und Angaben zur Fehlerbeseitigung.

Debugger

Der S7-SCL-Debugger bietet die Möglichkeit, ein Programm in seinem Ablauf im AS zu kontrollieren, und damit mögliche logische Fehler zu finden.

S7-SCL bietet dazu zwei verschiedene Testmodi an:
- Schrittweise Beobachten
- Kontinuierlich Beobachten

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

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

5. BEISPIELAUFGABE

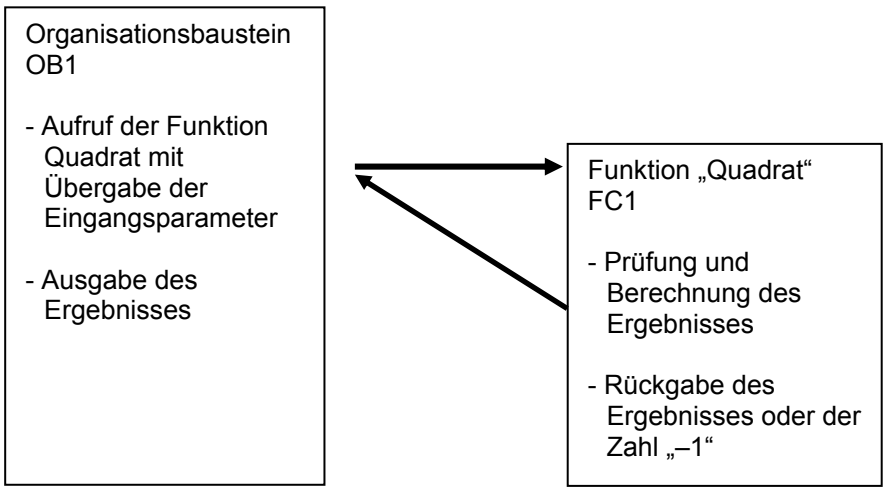
In diesem Beispiel soll eine natürliche Zahl quadriert werden. Bevor das Ergebnis berechnet wird, soll geprüft werden, ob der Betrag der Zahl größer oder gleich 181 ist. Ist dies der Fall kann die Rechenfunktion durchgeführt werden und das Ergebnis wird zurückgegeben. Wenn nicht soll der Rückgabewert „-1“ sein.

Die zu quadrierende Zahl soll im Integerformat über ein Eingangswort eingegeben werden. Die Ausgabe erfolgt über ein Ausgangswort. Die Überprüfung und Berechnung soll in einer Funktion programmiert sein.

Zuordnungsliste:

EW 0	Eingabe	Zu quadrierende Zahl
AW 0	Ausgabe	Ergebnis
FC 1	Quadrat	Quadratfunktion

Programmstruktur:

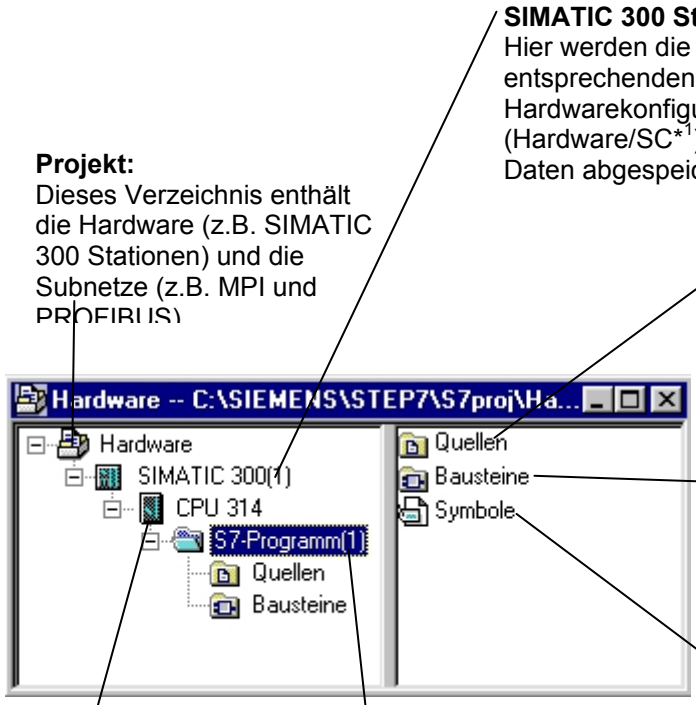


Hinweis: Der Wertebereich einer Zahl im Integerformat liegt zwischen –32768 und +32767. Die größtmögliche natürliche Zahl deren Quadrat in diesem Bereich liegt ist 181. Um eine Bereichsüberschreitung zu vermeiden wird vor der Quadrierung eine Prüfung vorgenommen.

6. STEP7- PROJEKT ANLEGEN

Die Dateiverwaltung erfolgt wie in STEP 7 mit dem **'SIMATIC Manager'.** Hier können z.B. Programmbausteine kopiert oder zur Weiterbearbeitung mit anderen Werkzeugen durch Anklicken mit der Maus aufgerufen werden. Die Bedienung entspricht den in WINDOWS üblichen Standards. So hat man z.B. die Möglichkeit mit einem Klick der rechten Maustaste das Auswahlmenü zu jeder Komponente zu erhalten.
In den Ordnern **‚SIMATIC 300 Station'** und **‚CPU'** wird der Hardwareaufbau der SPS abgebildet. Demzufolge ist ein solches Projekt auch immer hardwarespezifisch zu sehen.

Wie in STEP 7 wird jedes Projekt in einer fest vorgegebenen Struktur angelegt. Die Programme sind in den folgenden Verzeichnissen abgespeichert:



SIMATIC 300 Stationen:
Hier werden die entsprechenden Hardwarekonfigurations- (Hardware/SC*[1])und CPU-Daten abgespeichert.

Projekt:
Dieses Verzeichnis enthält die Hardware (z.B. SIMATIC 300 Stationen) und die Subnetze (z.B. MPI und PROFIBUS)

Quellen/SO*[1]:
Hier werden Quellen (z.B. SCL- Quellen) abgelegt, die durch Übersetzen in ablauffähige Programme umgewandelt werden können

Bausteine/AP-off*[1]:
Hier werden die Programmbausteine (OB, FB, FC, SFB, SFC, DB etc.) abgespeichert.

Symbole/SY*[1]:
Hier werden die Symbollisten zur Symbolischen Adressierung gespeichert.

CPU:
Hier wird das S7-Programm und die vernetzten Verbindungspartner (Verbindungen/CO*[1]) eingetragen.

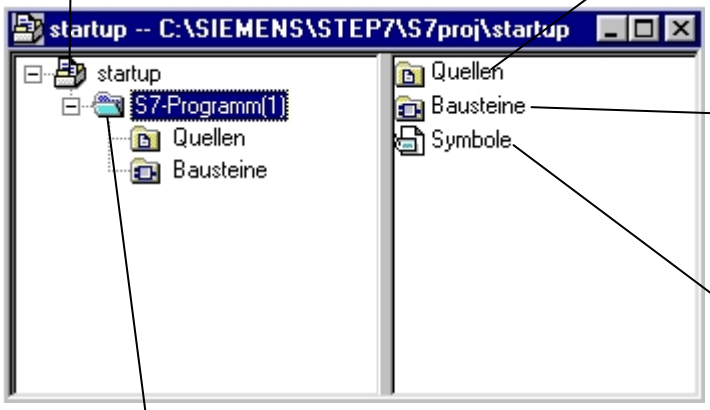
S7-Programm:
Hier werden die Anwenderprogramme (Bausteine/AP-off*[1]), Symboltabellen (Symbole/SY*[1]) und Quellen (Quellen/SO*[1]) verwaltet.

*[1] Bezeichnungen aus STEP 7 Version 2.x

Um ein Projekt unabhängig von der Hardwarekonfiguration zu erstellen gibt es jedoch die Möglichkeit ein Projekt anzulegen, dass diese Ordner nicht beinhaltet.

Es hat dann die folgende Struktur:

Projekt:
Dieses Verzeichnis enthält die Hardware (z.B. SIMATIC 300 Stationen) und die Subnetze (z.B. MPI und PROFIBUS)



Quellen/SO*[1]:
Hier werden Quellen (z.B. SCL- Quellen) abgelegt, die durch Übersetzen in ablauffähige Programme umgewandelt werden können.

Bausteine/AP-off*[1]:
Hier werden die Programmbausteine (OB, FB, FC, SFB, SFC, DB etc.) abgespeichert.

Symbole/SY*[1]:
Hier werden die Symbollisten zur Symbolischen Adressierung gespeichert.

S7-Programm:
Hier werden die Anwenderprogramme (Bausteine/AP-off*[1]), Symboltabellen (Symbole/SY*[1]) und Quellen (Quellen/SO*[1]) verwaltet.

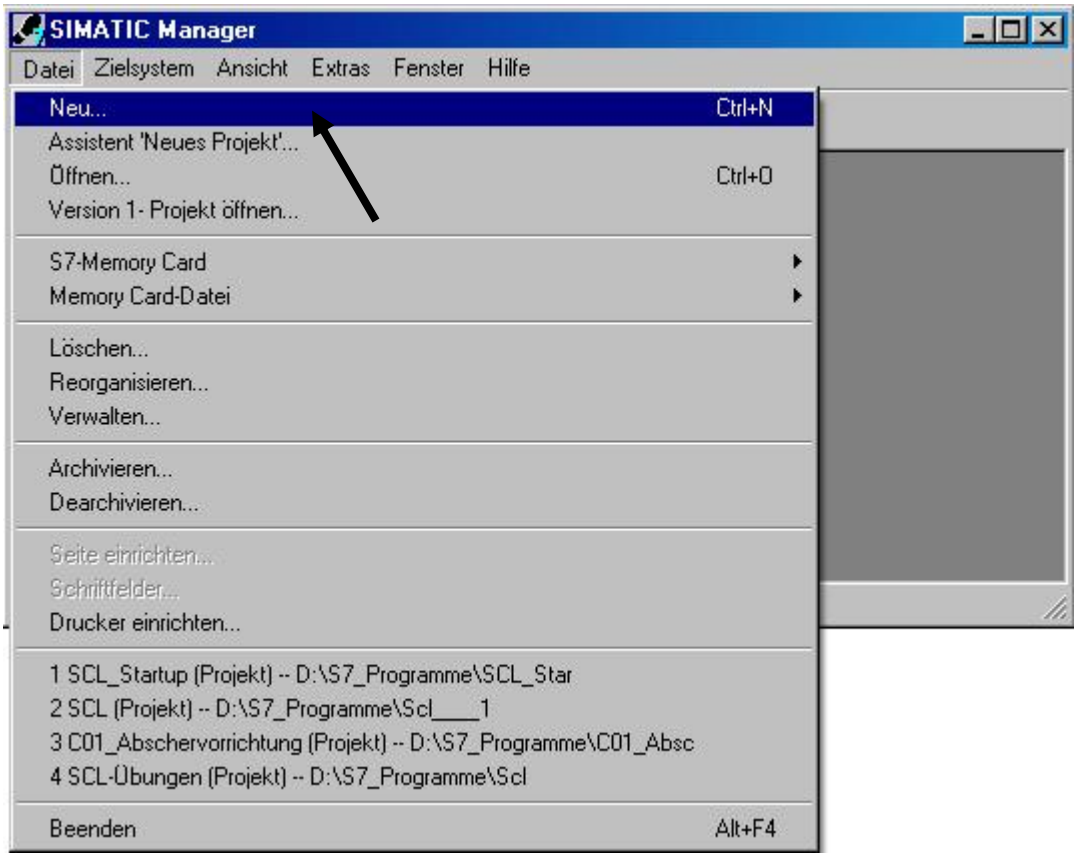
*[1] Bezeichnungen aus STEP 7 Version 2.x

Hinweis: Dieses Beispiel hier wird ohne Konfiguration der Hardware erstellt und somit können die Programme auf beliebige Konfigurationen der SIMATIC S7-300, S7-400 oder WinAC geladen werden. Lediglich die Adressen der Ein- und Ausgänge müssen von Fall zu Fall angepasst werden.

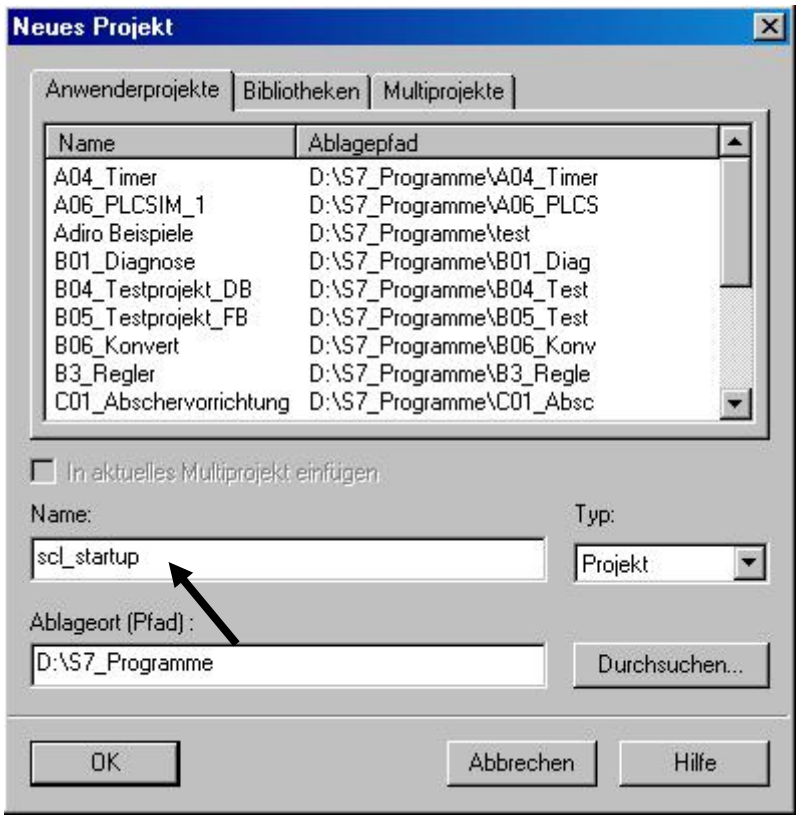
Folgende Schritte muss der Anwender ausführen, um ein Projekt zu erstellen, in dem dann das Lösungsprogramm geschrieben werden kann.

1. Das zentrale Werkzeug in STEP 7 ist der **‚SIMATIC Manager'**, der hier mit einem Doppelklick aufgerufen wird. (→ SIMATIC Manager)

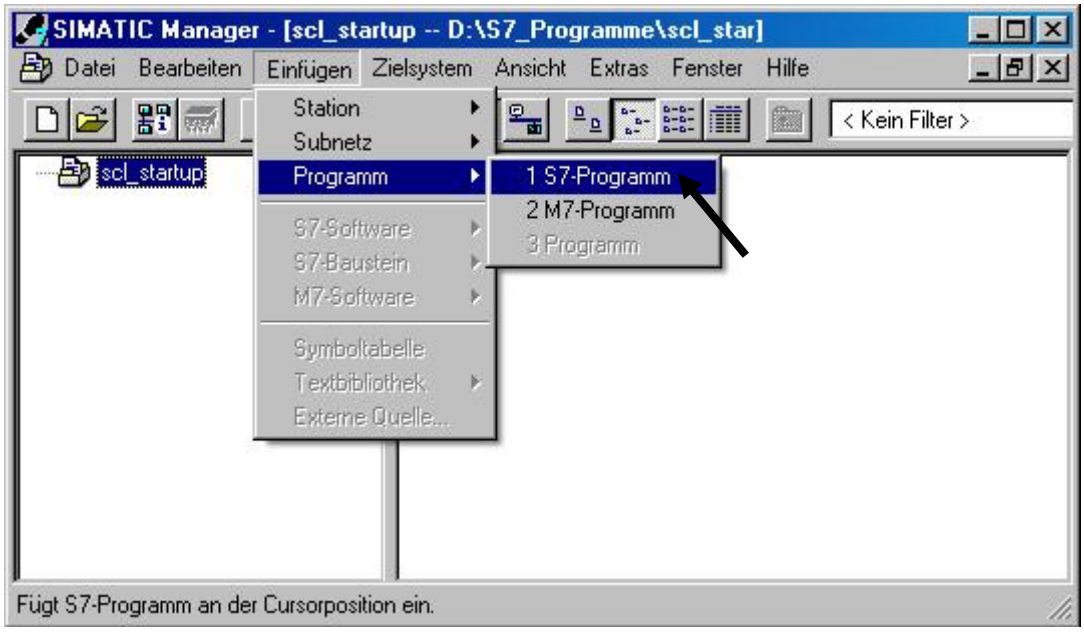
2. S7-SCL- Programme werden in STEP 7 Projekten verwaltet . Ein solches Projekt wird nun angelegt (→ Datei → Neu)



3. Dem Projekt wird nun der **‚Name' ‚scl_startup'** gegeben. (→ scl_startup → OK)

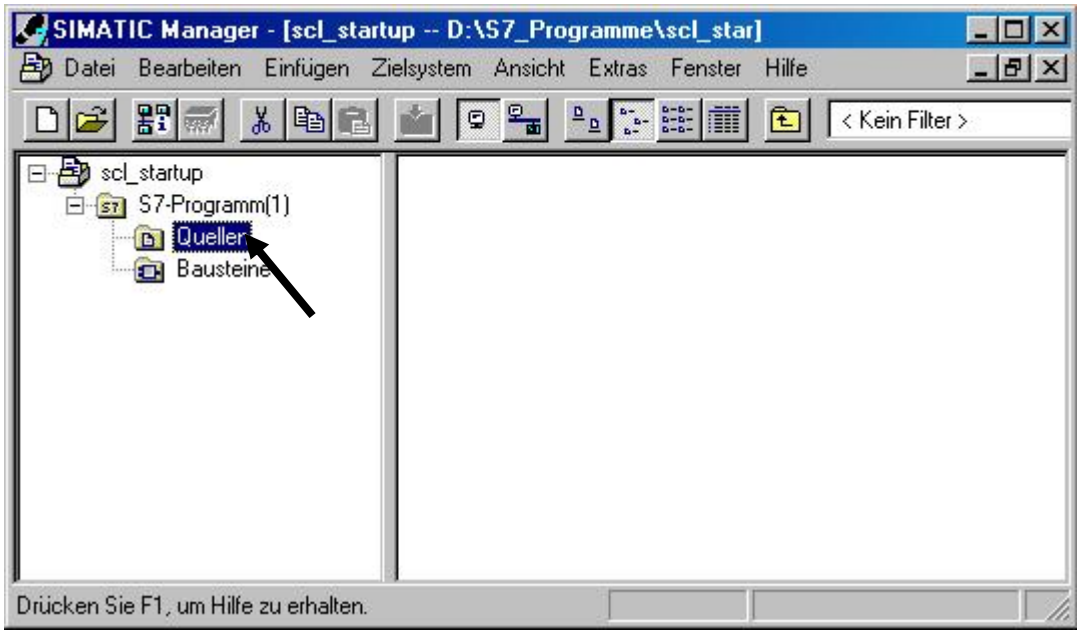


4. In dem Projekt **‚scl_startup'** wird dann ein neues **‚S7-Programm'** eingefügt. (→ scl_startup → Einfügen → Programm → S7-Programm)

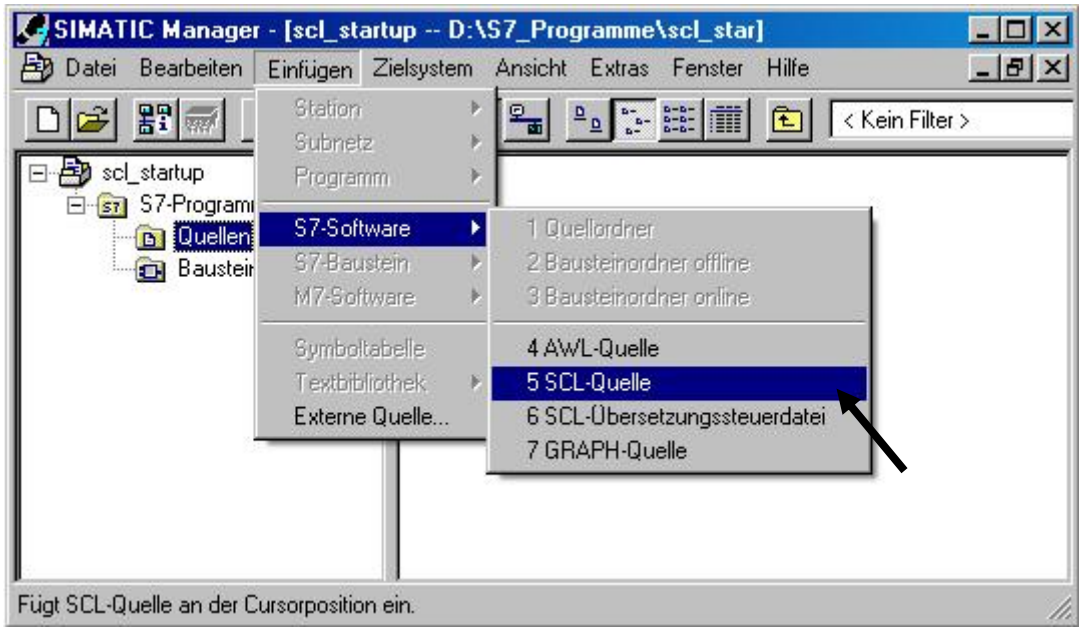


7. STEP 7 – PROGRAMM IN S7-SCL SCHREIBEN

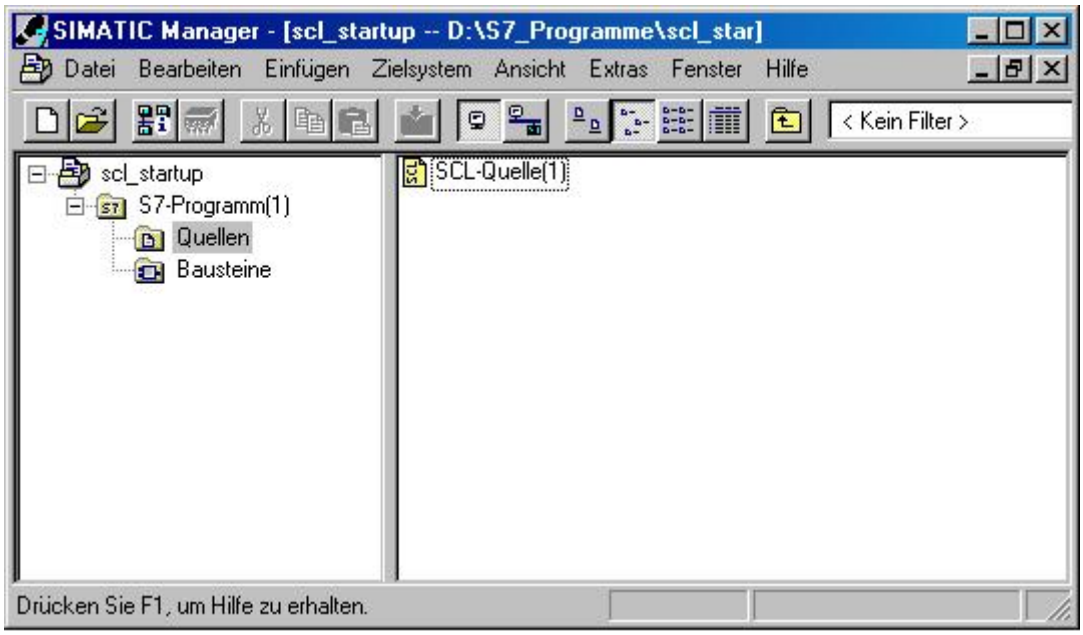
1. Um eine S7-SCL-Quelle einzufügen muss der Ordner **‚Quellen'** markiert werden. (→ Quellen)



2. Eine **‚SCL-Quelle'** wird nun eingefügt. (→ Einfügen → S7-Software → SCL-Quelle)

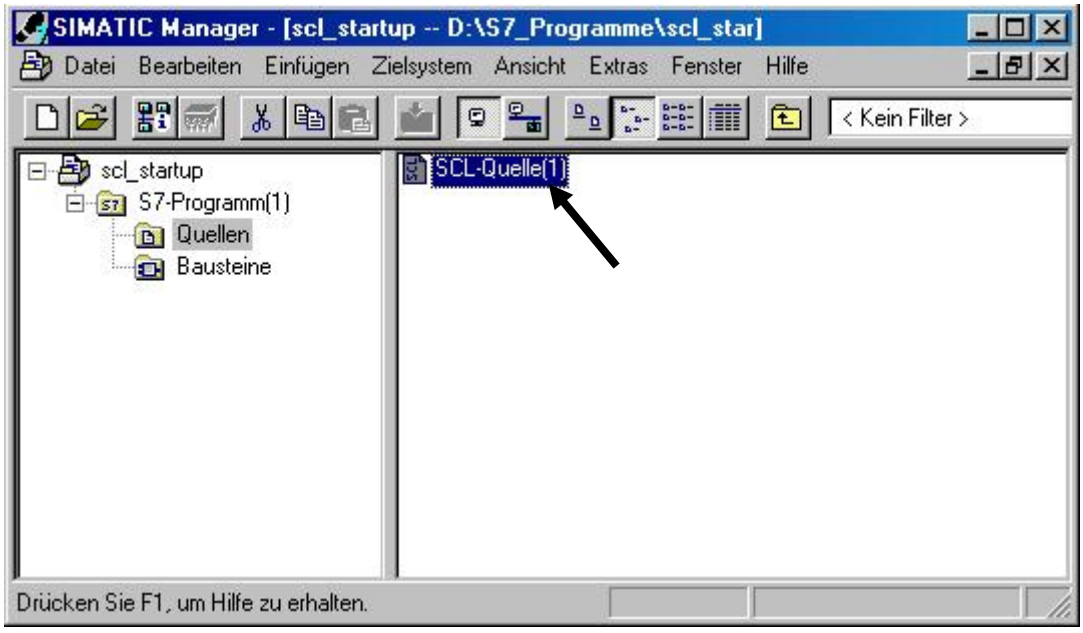


3. Im SIMATIC Manager steht nun eine SCL-Quelle zur Programmierung zur Verfügung.

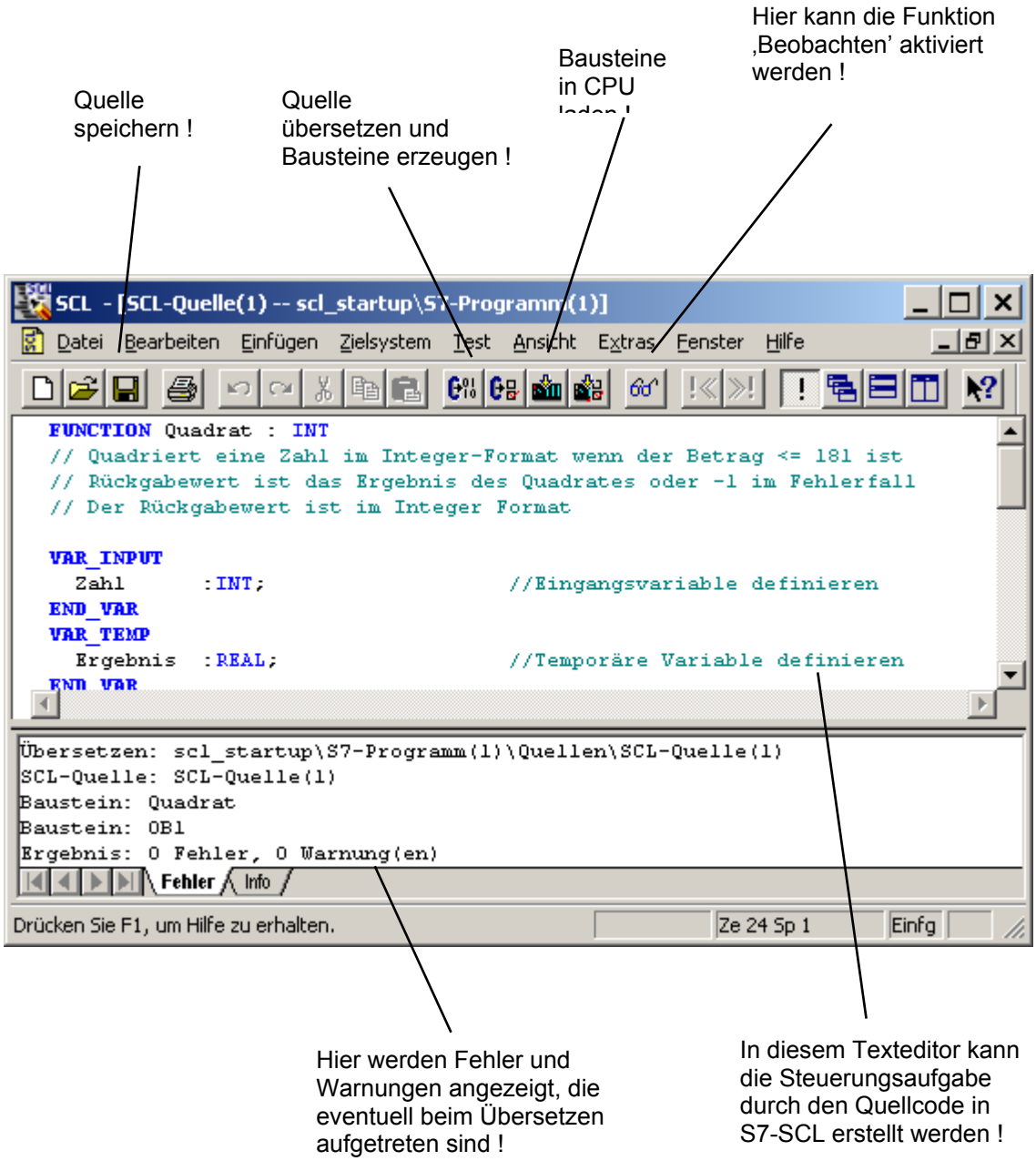


Hinweis: Sie haben die Möglichkeit den Namen der SCL- Quelle zu verändern. Dies ist bei einer komplexen Steuerungsaufgabe mit mehreren Quellen zur besseren Lesbarkeit des Programms sinnvoll. Klicken Sie hierzu die Quelle mit der Rechten Maustaste an, und wählen aus dem Menü **‚umbenennen'** aus.

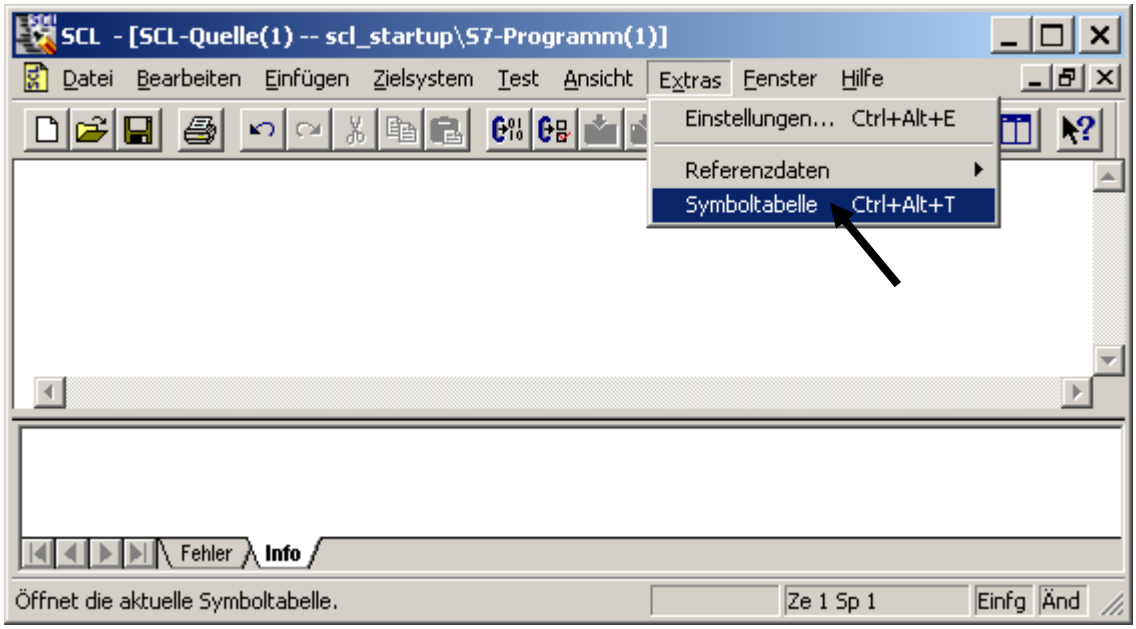
4. Der SCL-Editor soll nun gestartet werden. Dazu wird die Quelle im SIMATIC Manager mit einem Doppelklick geöffnet. (→ SCL-Quelle)



5. Die Programmieroberfläche für die Programmierung in SCL sieht dann folgendermaßen aus:

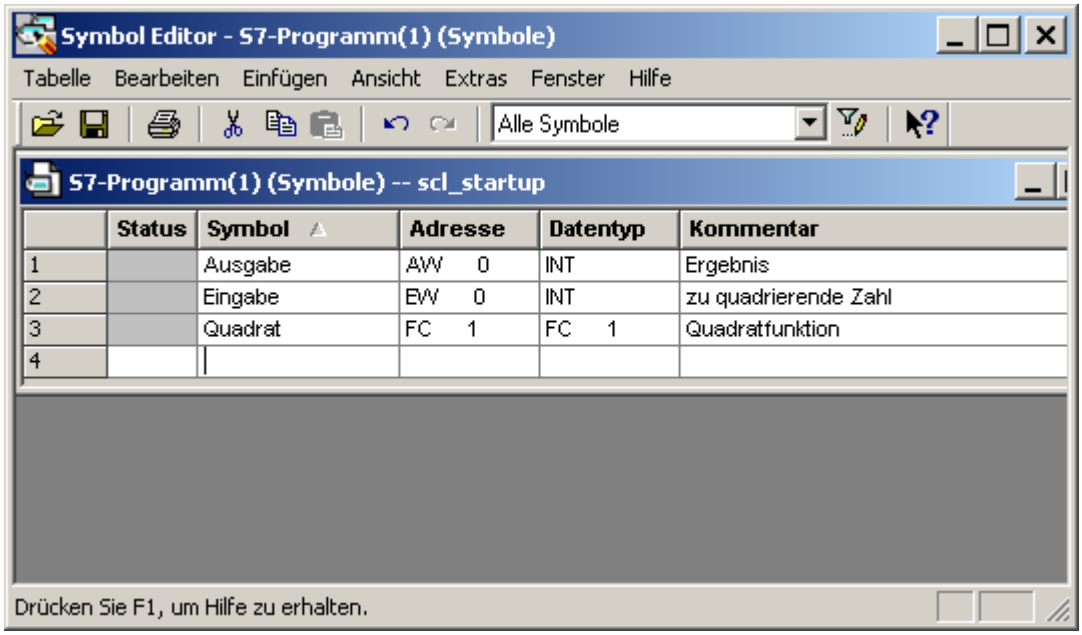


6. Zunächst soll die Symboltabelle erstellt werden. (→ Extras → Symboltabelle)



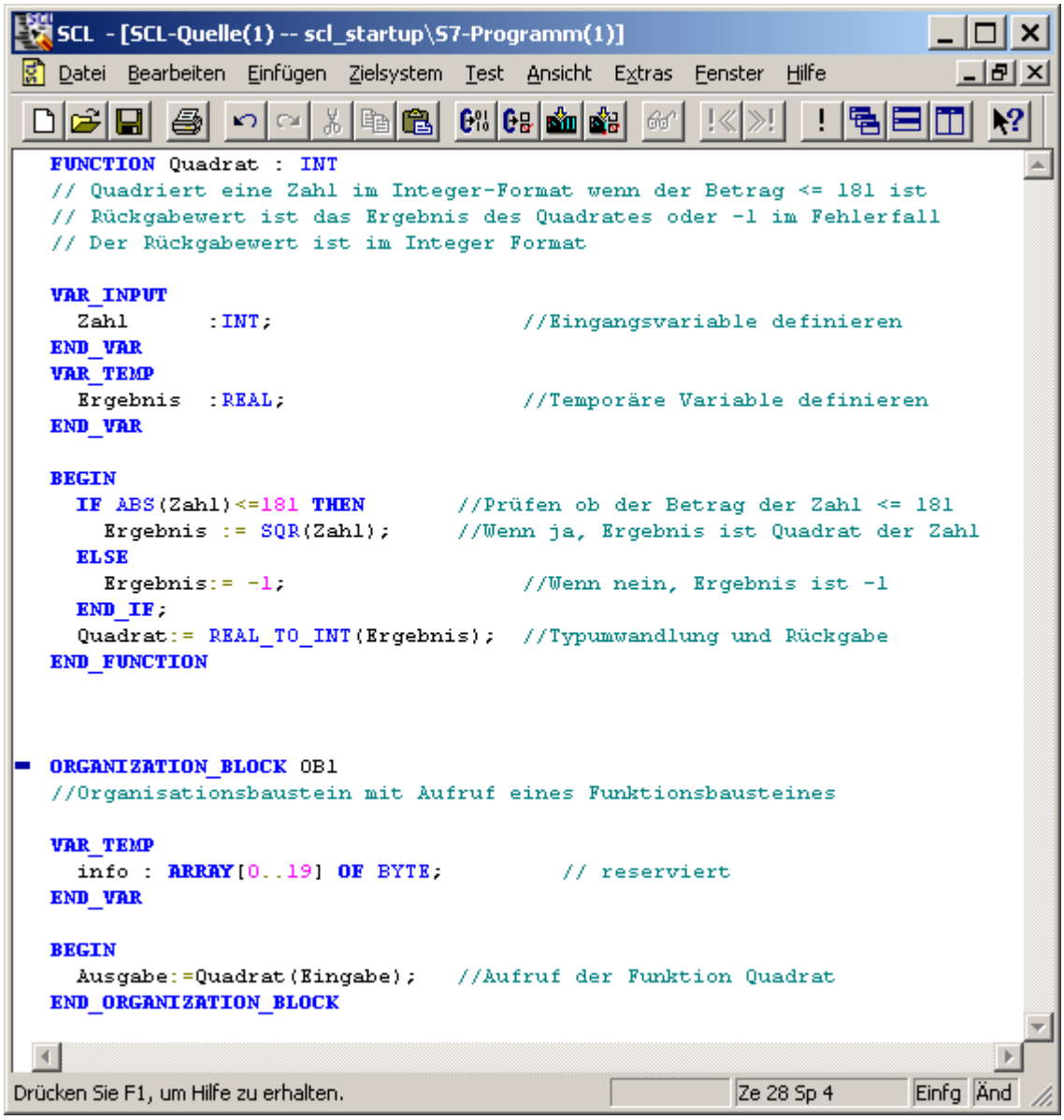
7. Operanden in Symboltabelle eintragen (→ Symbol → Adresse → Datentyp → Kommentar) und

Symbolliste speichern (→).



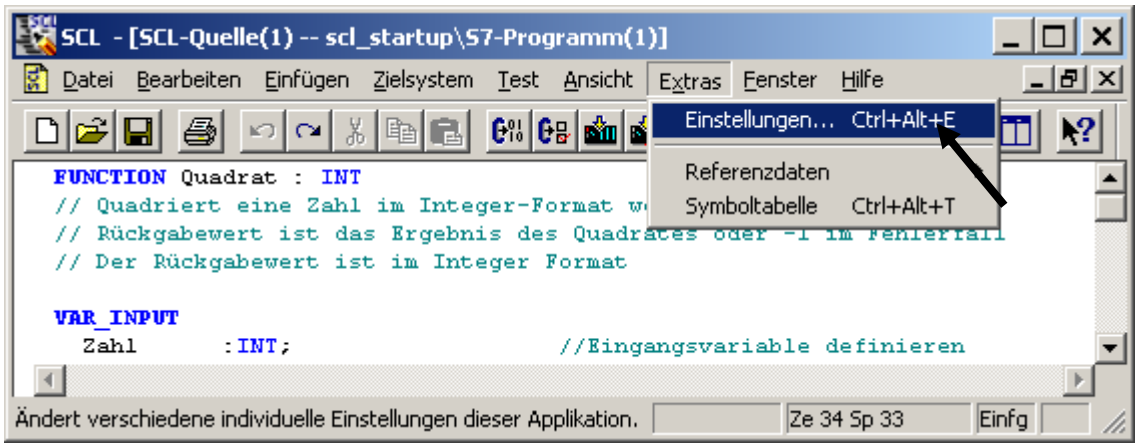
	Status	Symbol	Adresse	Datentyp	Kommentar
1		Ausgabe	AW 0	INT	Ergebnis
2		Eingabe	EW 0	INT	zu quadrierende Zahl
3		Quadrat	FC 1	FC 1	Quadratfunktion
4					

8. Nun kann das Programm im Textfeld des Editors unter Verwendung der symbolischen Namen eingegeben werden. Dies könnte für das Beispiel in S7-SCL folgendermaßen aussehen.

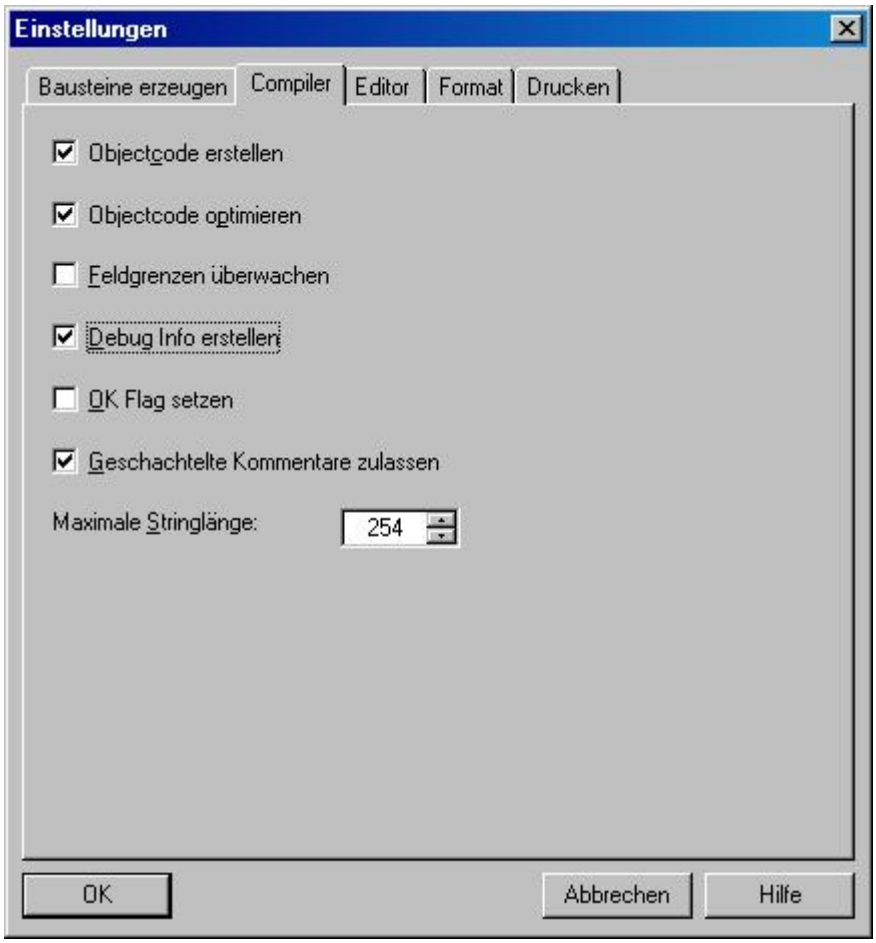


Hinweis: Sie haben die Möglichkeit alle Bausteine in einer Quelldatei zu programmieren. Dabei muss darauf geachtet werden, dass der aufgerufene Baustein immer vor dem aufrufenden Baustein programmiert ist.

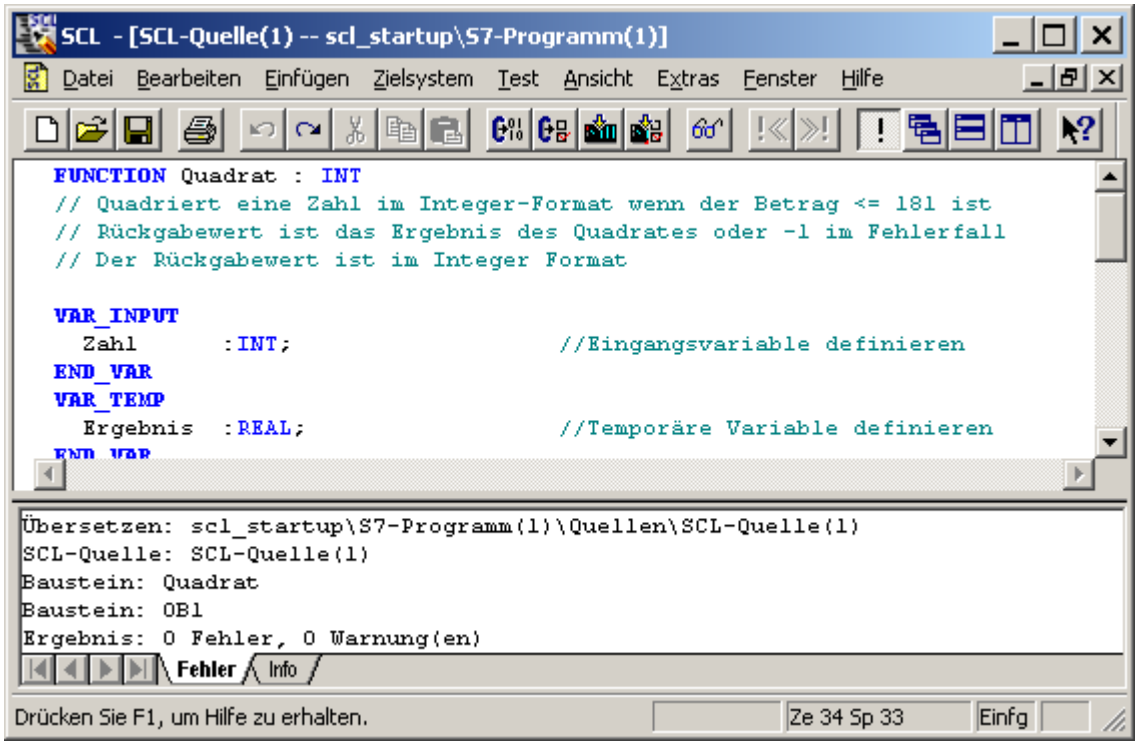
9. Bevor die Quelle gespeichert wird müssen die **‚Einstellungen'** des Compilers angepasst werden. (→ Extras → Einstellungen)



10. Im Register **‚Compiler'** müssen folgende Einstellungen gemacht werden. (→ Compiler → OK)



11. Das Programm kann nun gespeichert , übersetzt und auf Syntaxfehler untersucht werden. Wenn alle Fehler behoben sind, können die Bausteine in die CPU geladen werden. (→ → →)



Hinweis: Wenn mehrere Bausteine in einer Quelle programmiert sind, haben Sie die Möglichkeit die Bausteine einzeln zu übersetzen ( ) und einzeln in die CPU zu laden. ( )

8. STEP 7- PROGRAMM IN DER CPU TESTEN

Die Testfunktionen von S7-SCL bieten Ihnen die Möglichkeit, ein Programm in seinem Ablauf in der CPU zu kontrollieren und damit mögliche Fehler zu finden. Syntaxfehler werden bei der Übersetzung angezeigt. Laufzeitfehler in der Ausführung des Programms werden durch Systemalarme ebenfalls angezeigt; logische Programmierfehler können Sie mit den Testfunktionen ermitteln.

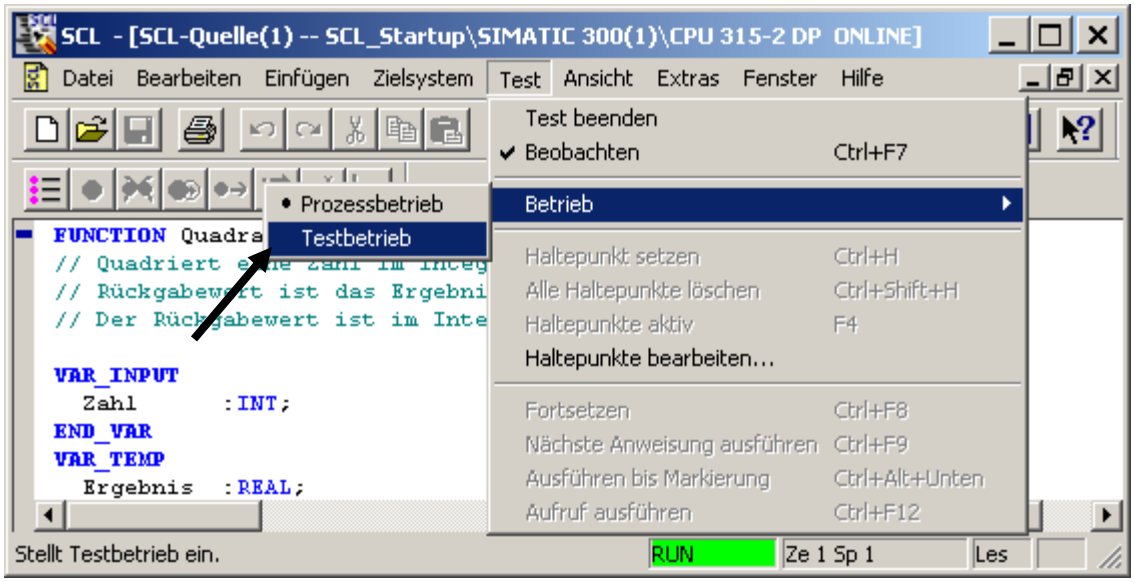
8.1 Betriebsmodi

Das Abfragen der Testinformationen bewirkt meist eine Verlängerung der Zykluszeiten. Um diese Verlängerung beeinflussen zu können, bietet SCL zwei verschiedene Betriebsmodi an.

Testbetrieb
Im Betriebsmodus **‚Testbetrieb'** wird der Beobachtungsbereich lediglich durch die Leistungsfähigkeit der angeschlossenen CPU begrenzt. Alle Testfunktionen sind ohne Einschränkung nutzbar. Größere Verlängerungen der CPU Zykluszeit können auftreten, da z.B. der Status von Anweisungen in programmierten Schleifen bei jedem Durchlauf ermittelt wird.

Prozessbetrieb
Im Betriebsmodus **‚Prozessbetrieb'** schränkt der SCL- Debugger den maximalen Beobachtungsbereich so ein, dass die Zykluszeiten beim Testvorgang die realen Ablaufzeiten des Prozesses nicht oder nur unwesentlich überschreiten.

1. Den gewünschten Betriebsmodus können sie im Menü **‚Test'** auswählen. Der aktuell eingestellte Modus ist durch einen Punkt markiert. (→ Test → Betrieb → Testbetrieb)



8.2 STEP7 Programm Beobachten

Zwei verschiedene Testfunktionen werden Ihnen von der Entwicklungsumgebung S7-SCL zur Verfügung gestellt.

Kontinuierlich Beobachten
Mit dieser Funktion können Sie Namen und aktuelle Werte von Variablen in der SCL- Quelle ausgeben. Während des Testlaufs werden die Werte der Variablen und Parameter dieses Bereichs in chronologischer Abfolge angezeigt und zyklisch aktualisiert.

Schrittweise Beobachten (nur S7-400-CPUs)
Mit dieser Funktion können Sie Haltepunkte setzen und anschließend einen Testlauf in Einzelschritten durchführen. Sie können den Programmalgorithmus z.B. Anweisung für Anweisung ausführen und beobachten, wie sich die Werte der bearbeiteten Variablen verändern.

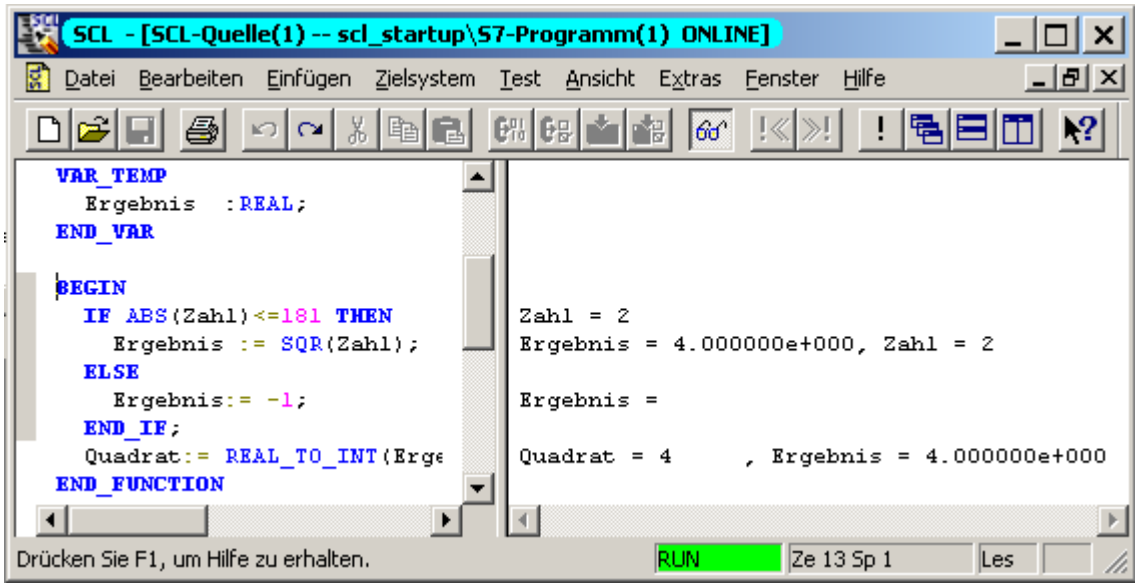
8.2.1 Kontinuierlich Beobachten

Beim kontinuierlichen Beobachten eines Programms können Sie eine Gruppe von Anweisungen testen. Diese Gruppe von Anweisungen nennt man auch Beobachtungsbereich. Liegt der Beobachtungsbereich in einem Programmteil, der in jedem Zyklus durchlaufen wird, können die Werte der Variablen in der Regel nicht aus aufeinander folgenden Zyklen erfasst werden.

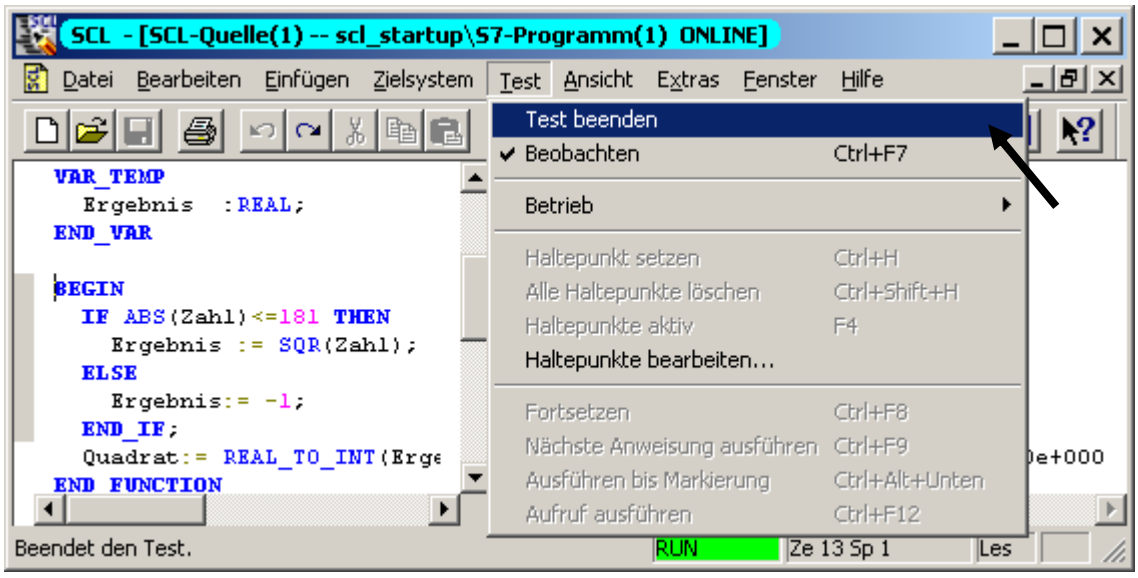
Werte, die sich im aktuellen Durchlauf geändert haben, und Werte, die sich nicht geändert haben, können farblich unterschieden werden.

Der Umfang der zu testenden Anweisungen ist durch die Leistungsfähigkeit der angeschlossenen CPU begrenzt. Da die SCL- Anweisungen des Quellcodes in unterschiedlich viele Anweisungen im Maschinencode abgebildet werden, ist die Länge des Beobachtungsbereichs variabel, sie wird vom SCL- Debugger ermittelt und markiert, wenn Sie die erste Anweisung des gewünschten Beobachtungsbereichs auswählen.

1. Durch einen Mausklick auf das Brillensymbol kann das Programm nun kontinuierlich beobachtet werden. Im linken Teilfenster wird der beobachtete Bereich mit einem grauen Balken dargestellt. Im rechten Teilfenster können die Werte abgelesen werden. (→)



3. Die Testfunktion muss im Menü **‚Test'** beendet werden. (→ Test → Test beenden)



Hinweis: Wenn Sie die Funktion **‚Beobachten'** über das Brillensymbol deaktivieren, bleibt die Fensterteilung erhalten. Das Programm ist weiterhin ONLINE geöffnet und somit nicht editierbar. Sie können jedoch Haltepunkte setzen, löschen oder bearbeiten und schrittweise den Programmablauf beobachten.

8.2.2 Schrittweise Beobachten

Beim Testen mit Haltepunkten erfolgt der Testlauf in Einzelschritten. Sie können den Programmalgorithmus Anweisung für Anweisung ausführen und beobachten, wie sich die Werte der bearbeiteten Variablen verändern. Die Anzahl der Haltepunkte ist dabei abhängig von der angeschlossenen CPU.
Nach dem Setzen von Haltepunkten können Sie das Programm zunächst bis zu einem Haltepunkt ausführen lassen. Beim Erreichen dieser Anweisung geht die CPU in den Betriebszustand HALT und von dort aus können Sie mit dem Schrittweisen Beobachten beginnen

Erlaubt Ihnen, gesetzte Haltepunkte wahlweise zu aktivieren/deaktivieren/löschen und festzulegen, dass bestimmte Haltepunkte nur in einer definierten Aufrufumgebung aktiv sind.

Setzt oder löscht Haltepunkte an der aktuellen Position des Mauszeigers. Dieser Menübefehl ist nur aktiv, wenn Sie nicht gerade die Funktion "Beobachten" ausführen, d.h. wenn der Menübefehl Beobachten nicht mit einem Häkchen markiert ist

Löscht alle Haltepunkte des aktuellen S7-Programms. Dieser Menübefehl ist nur aktiv, wenn Sie nicht gerade die Funktion "Beobachten" ausführen, d.h. wenn der Menübefehl Beobachten nicht mit einem Häkchen markiert ist.

Schaltet den Testmodus "Testen mit Haltepunkten" ein. Dieser Menübefehl ist nur aktiv, wenn Sie nicht gerade die Funktion "Beobachten" ausführen, d.h. wenn der Menübefehl Beobachten nicht mit einem Häkchen markiert ist.

Bewirkt die Ausführung des Programms bis zum nächsten Haltepunkt. Beim Erreichen dieses Haltepunkts geht die CPU in HALT.

Bewirkt die Ausführung der SCL-Anweisung an der aktuellen Position. Nach der Ausführung geht die CPU in HALT, die Einfügemarke wird zur nächsten Anweisung bewegt, und die Inhalte der Variablen in der zuletzt bearbeiteten Anweisung werden angezeigt.

Bewirkt die Ausführung des Programms bis zur aktuellen Position der Einfügemarke. Beim Erreichen der Einfügemarke geht die CPU in HALT.

Handelt es sich bei dem aufgerufenen Baustein um einen von SCL erzeugten Baustein, bewirkt der Menübefehl Aufruf ausführen, dass der aufgerufene Baustein geöffnet wird und das Programm bei der ersten Anweisung des Bausteins anhält. Jetzt können Sie mit den Einzelschrittfunktionen den Baustein schrittweise beobachten. Wird das Bausteinende erreicht, springt das Programm in den aufrufenden Baustein zurück und hält in der Zeile nach dem Bausteinaufruf an. Handelt es sich bei dem aufgerufenen Baustein um einen nicht von SCL erzeugten Baustein, wird der Aufruf übersprungen und die nachfolgende Programmzeile markiert.

Beispielaufgabe Erweitern und Testen

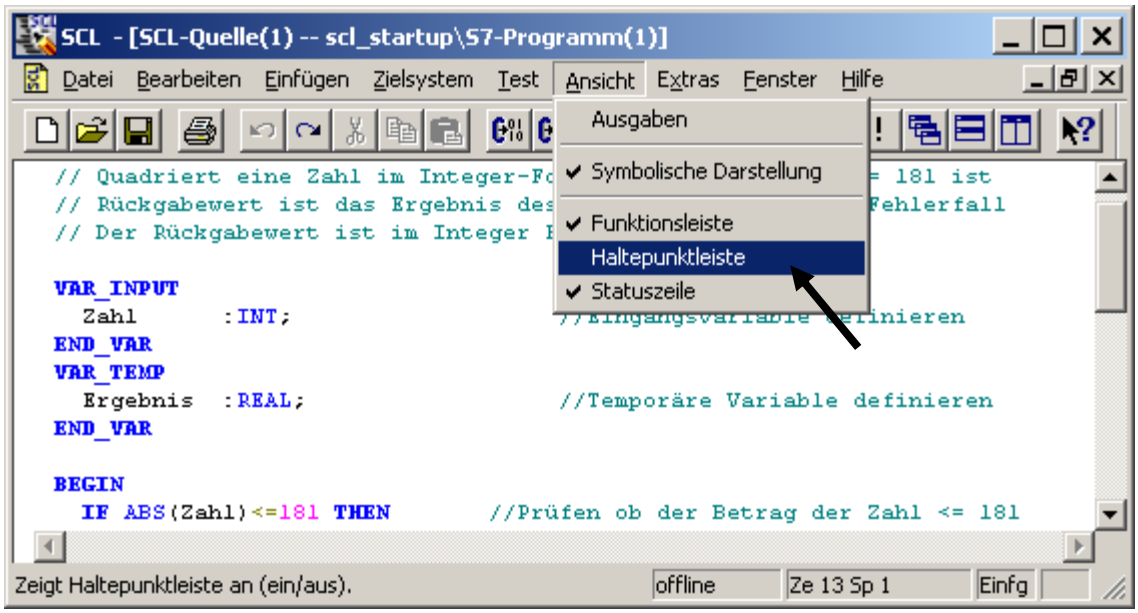
Um die verschiedenen Möglichkeiten im Testbetrieb besser zeigen zu können soll die Beispielaufgabe ergänzt werden.

Drei verschiedene Integerzahlen sollen quadriert und ausgegeben werden. Hierzu wird die Funktion Quadrat im OB1 drei mal hintereinander aufgerufen. Dabei sollen die Eingangsparameter bei jedem Aufruf auf das jeweilige Eingangswort angepasst werden. Die Ausgabe der verschiedenen Ergebnisse soll auf verschiedene Ausgangswörter gelegt werden.

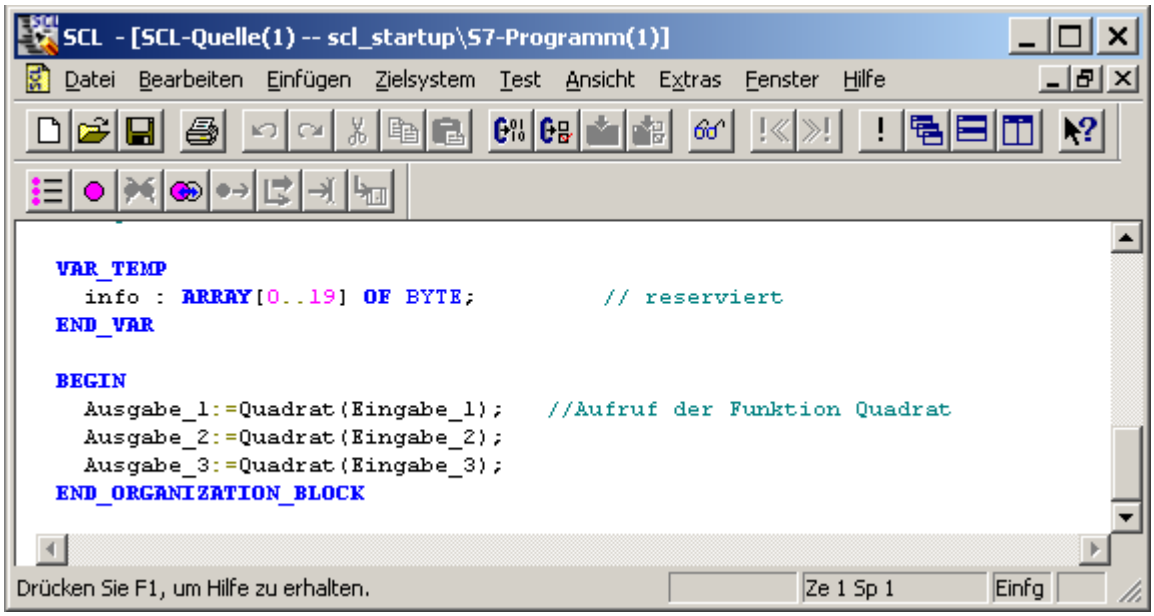
1. Ergänzen Sie hierzu die Symboltablle wie folgt.

EW 0	Eingabe_1	Erste zu quadrierende Zahl
EW 2	Eingabe_2	Zweite zu quadrierende Zahl
EW 4	Eingabe_3	Dritte zu quadrierende Zahl
AW 0	Ausgabe_1	Ergebnis 1
AW 2	Ausgabe_2	Ergebnis 2
AW 4	Ausgabe_3	Ergebnis 3
FC 1	Quadrat	Quadratfunktion

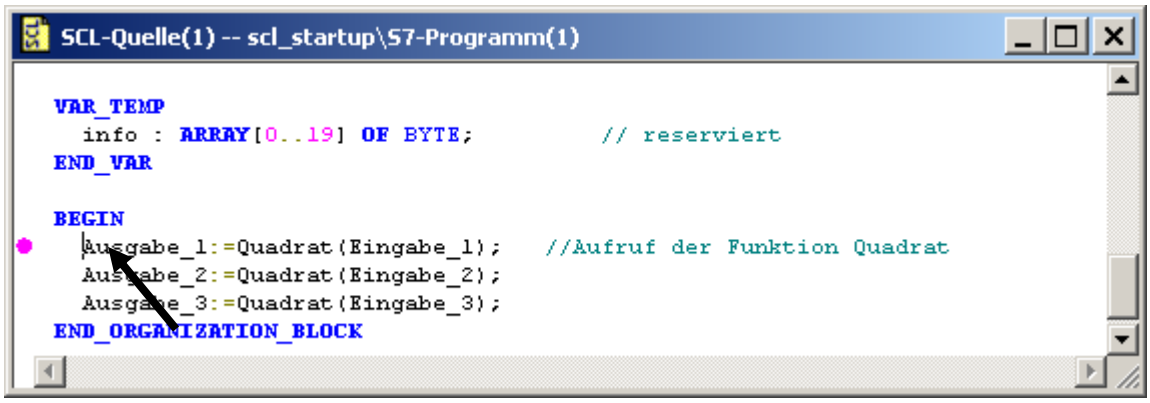
2. Die Symbolleiste für die Haltepunktbefehle können Sie im Menü Ansicht **‚Haltepunktleiste'** einblenden. Dies wird durch ein Häckchen am Menüpunkt bestätigt.
(→ Ansicht → Haltepunktleiste)



3. In der SCL- Quelle werden die zusätzlichen Aufrufe der Funktion Quadrat eingefügt. Das Programm muss dann gespeichert , übersetzt und auf Syntaxfehler untersucht werden. Wenn alle Fehler behoben sind, können die Bausteine in die CPU geladen werden.
(→ → →)

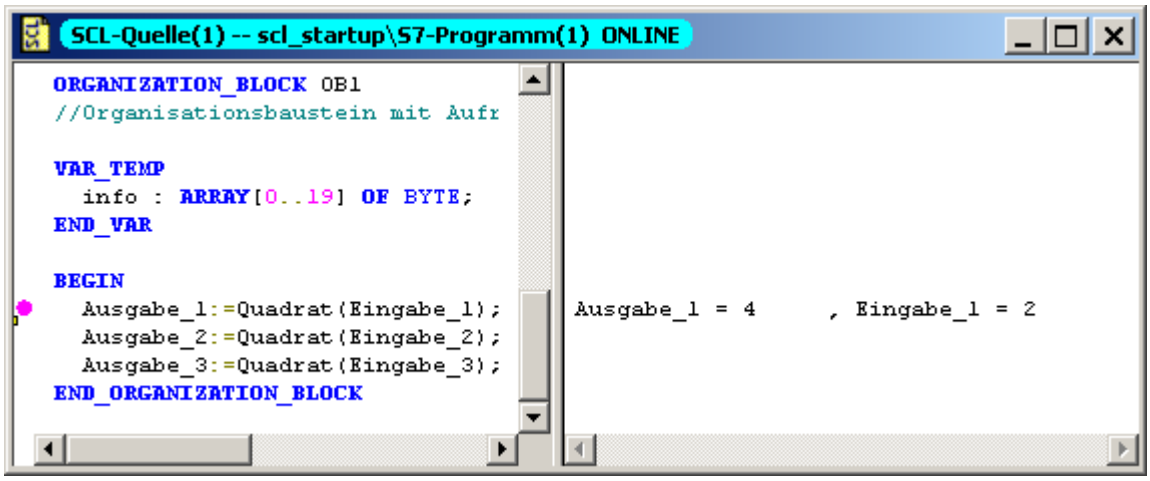


4. Setzen Sie die Markierung in die Zeile mit dem ersten Aufruf der Funktion FC1 und fügen Sie einen Haltepunkt ein, indem sie den Button anklicken. (→)

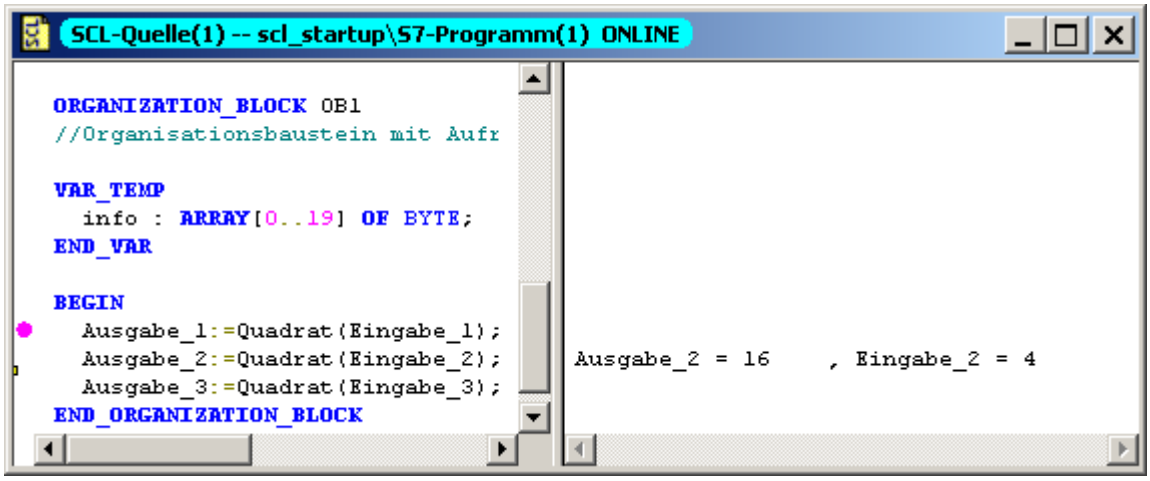


Hinweis: Sie haben die Möglichkeit mehrere Haltepunkte einzufügen. Markieren Sie hierzu die gewünschte Zeile und fügen Sie einen weiteren Haltepunkt ein . Um einen bereits gesetzten Haltepunkt zu löschen, markieren Sie diesen und verwenden Sie den gleichen Button .

5. Das Schrittweise Beobachten wird nun gestartet, indem die gesetzten Haltepunkte aktiviert werden . Klicken Sie auf um die erste Anweisung bearbeiten und im rechten Teilfenster anzeigen zu lassen. (→ →)

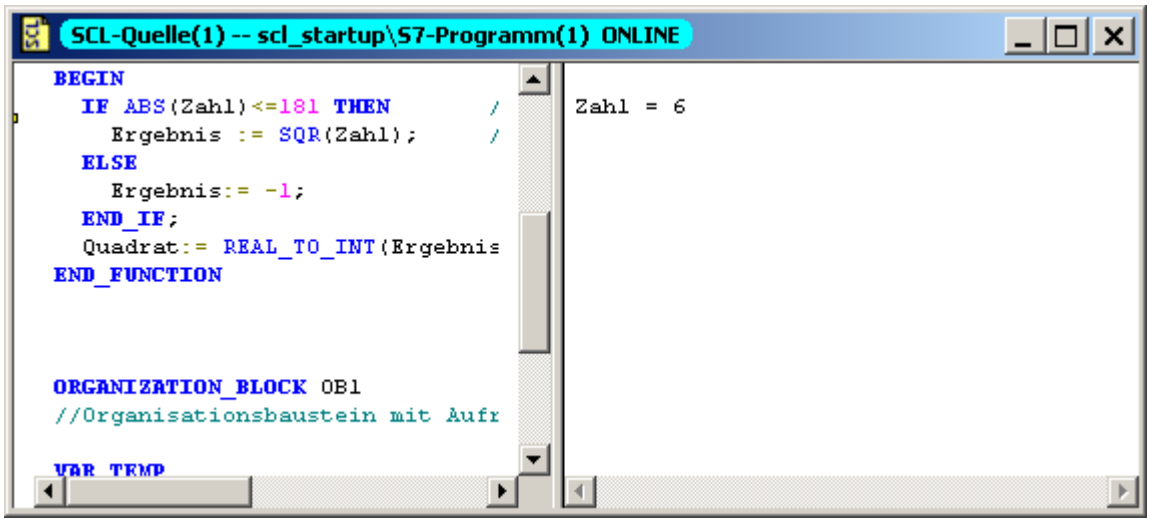


6. Die nächste Anweisung können Sie durch einen weiteren Klick auf bearbeiten lassen. (→)

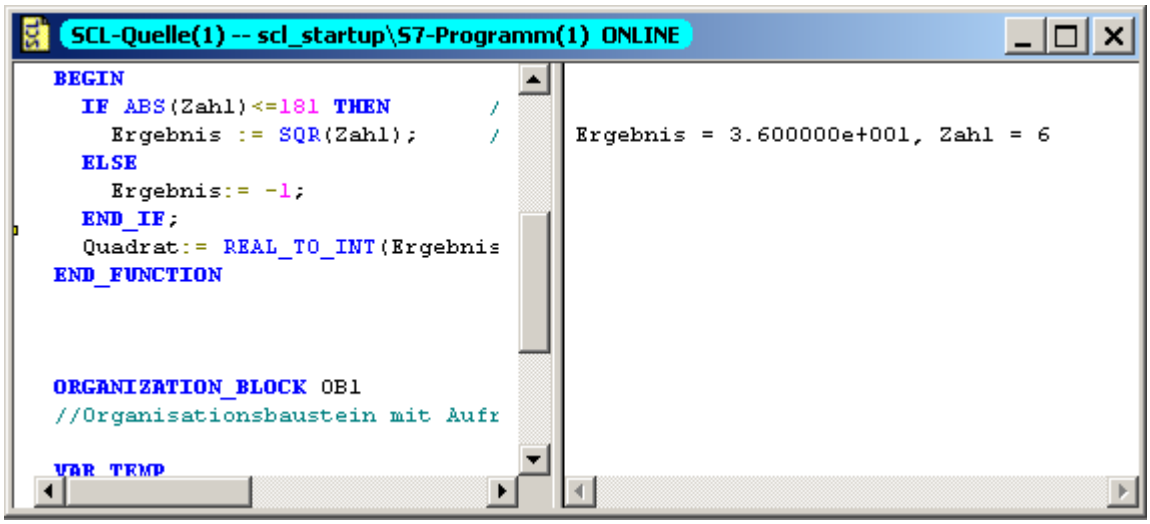


Hinweis: Beim Schrittweisen Beobachten werden nur die Werte der aktuell bearbeiteten Anweisung angezeigt. Bausteinaufrufe werden durch den Befehl **‚Zu nächster Anweisung'** zwar ausgeführt, werden aber nicht schrittweise angezeigt.

7. Ein aufgerufener Baustein kann durch den Befehl **‚Aufruf ausführen'** schrittweise beobachtet werden. Die erste Anweisung wird durch die Funktion **‚Zu nächster Anweisung'** angezeigt. (→ →)

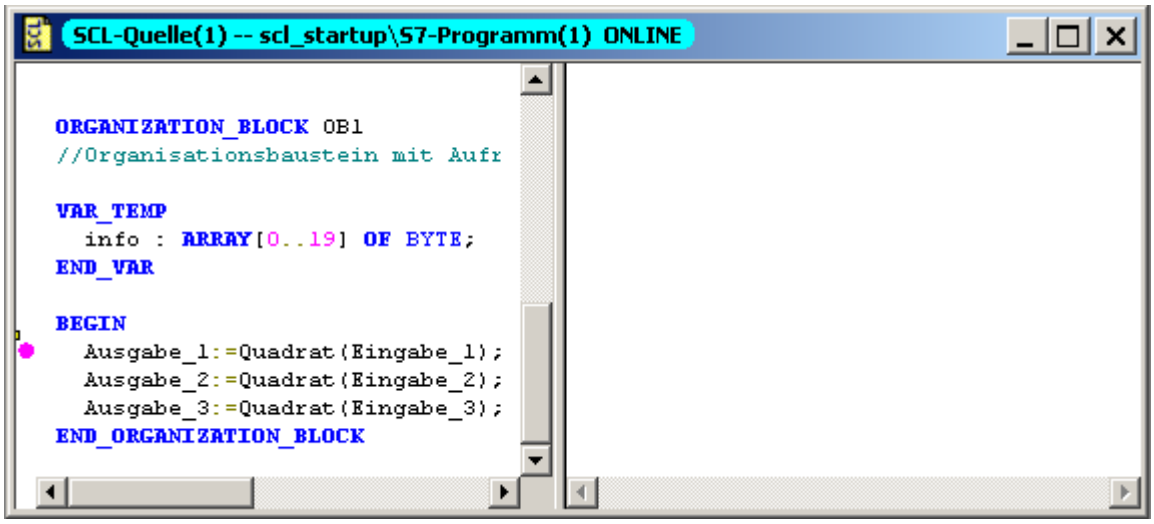


8. Jede weiter Anweisung wird durch die gleiche Funktion bearbeitet und dargestellt. (→)

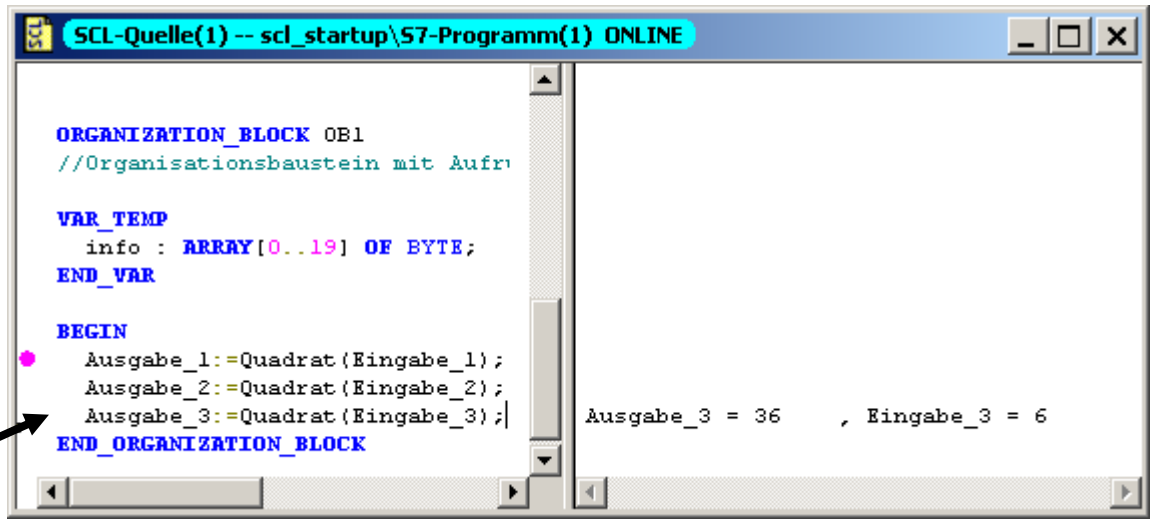


Hinweis: Ist das Bausteinende erreicht und wird der Befehl **‚Zu nächster Anweisung'** nochmals ausgeführt, wird die Anweisung nach dem Bausteinaufruf des aufrufenden Bausteins bearbeitet.

9. Mit dem Befehl **‚Fortsetzen'** wird das Programm bis zum nächsten Haltepunkt ausgeführt. Wenn bis zum Erreichen des Programmendes kein Haltepunkt gesetzt ist, wird der Programmzyklus erneut gestartet und bis zum nächsten Haltepunkt abgearbeitet. (→)



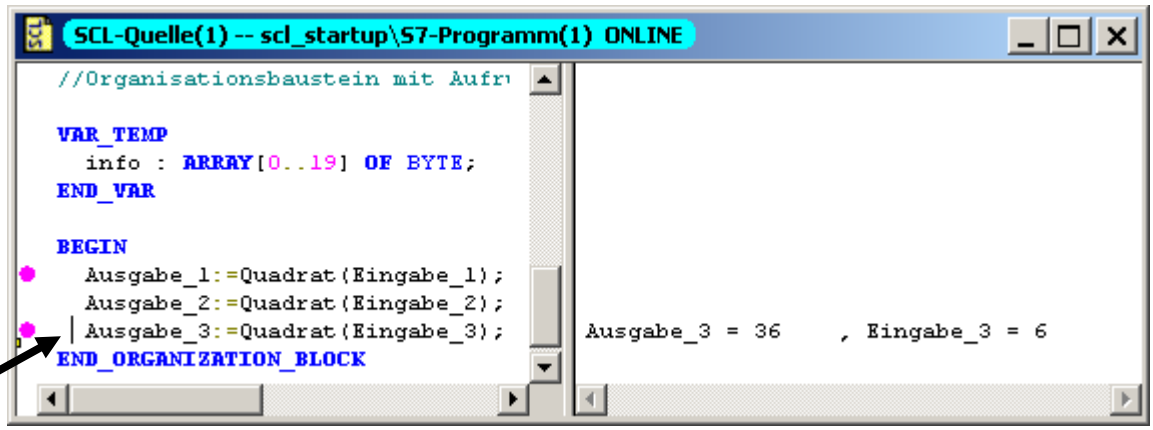
10. Markieren Sie nun den Anfang der Zeile mit dem dritten Aufruf der Funktion Quadrat. Mit dem Befehl **‚Zur Markierung'** wird das Programm bis zu dieser Anweisung bearbeitet. Mit **‚Zu nächster Anweisung'** kann diese bearbeitet und angezeigt werden. (→ →)



Hier markieren!

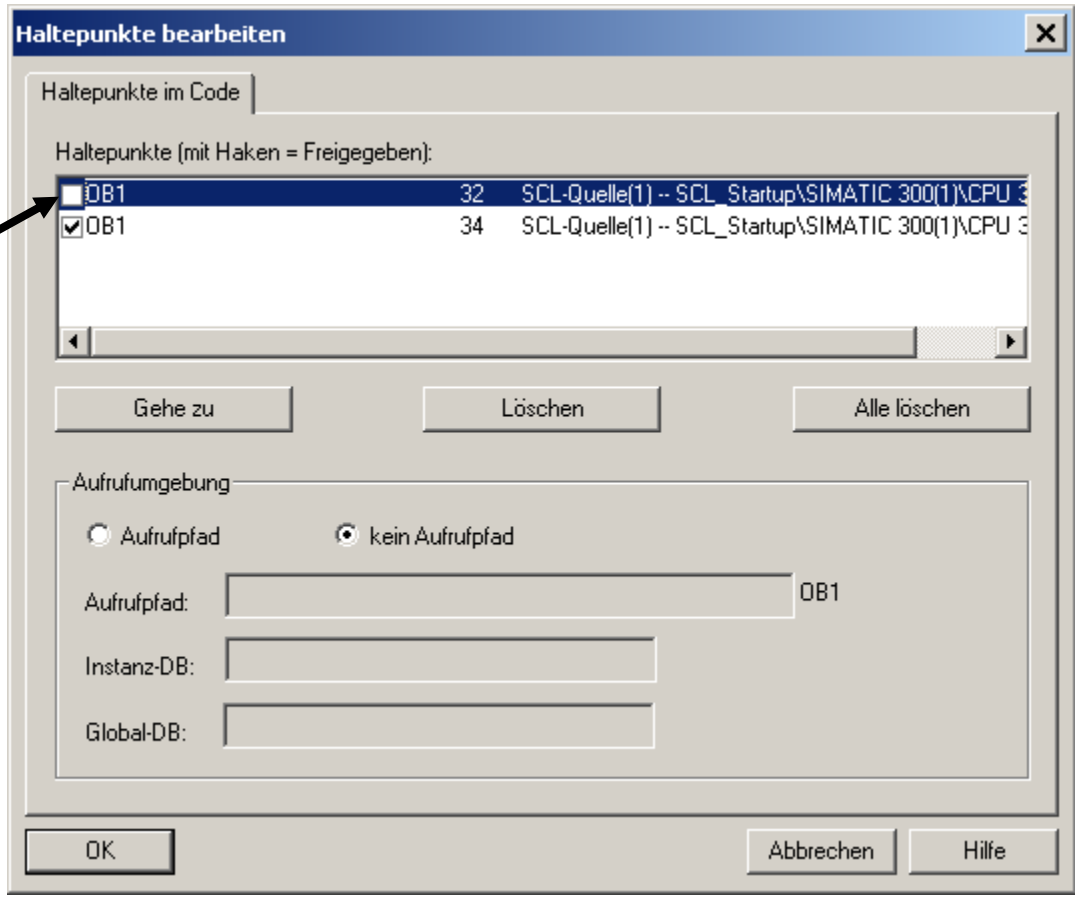
Hinweis: Der Befehl **‚Zur Markierung'** kann nur innerhalb eines Bausteines angewendet werden. Soll das Programm bis zu einer Stelle in einem anderen Baustein abgearbeitet werden, so ist dies mit einem Haltepunkt und dem Befehl **‚Fortsetzen'** zu realisieren.

11. Markieren Sie wieder den Anfang der Zeile mit dem dritten Aufruf der Funktion Quadrat. Fügen Sie einen weiteren Haltepunkt an dieser Stelle ein. (→)



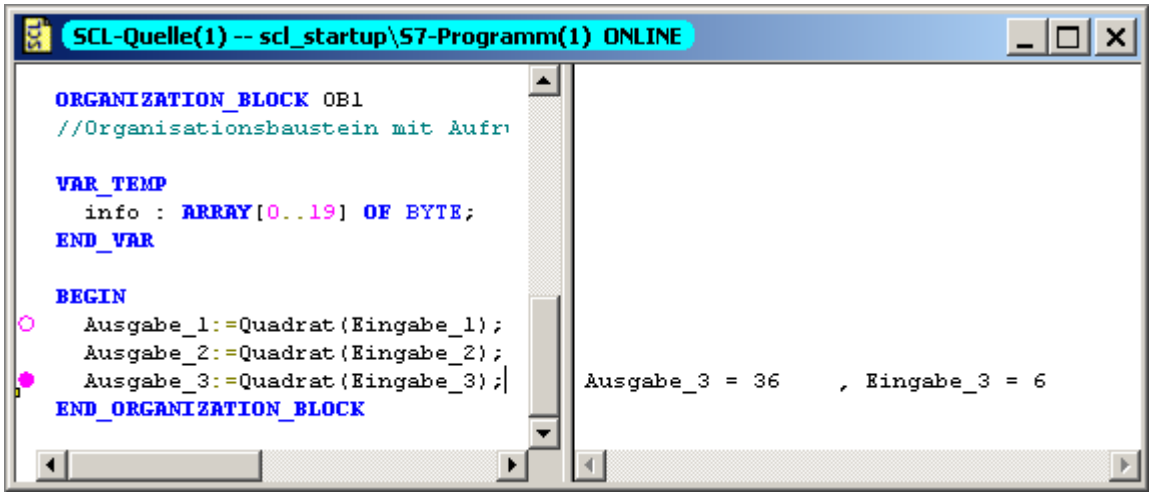
Hier markieren!

12. Mit **‚Haltepunkte Bearbeiten'** können Haltepunkte aktiviert/deaktiviert oder gelöscht werden. Deaktivieren Sie den ersten Haltepunkt im OB1. (→ → deaktivieren → OK)



Hier deaktivieren!

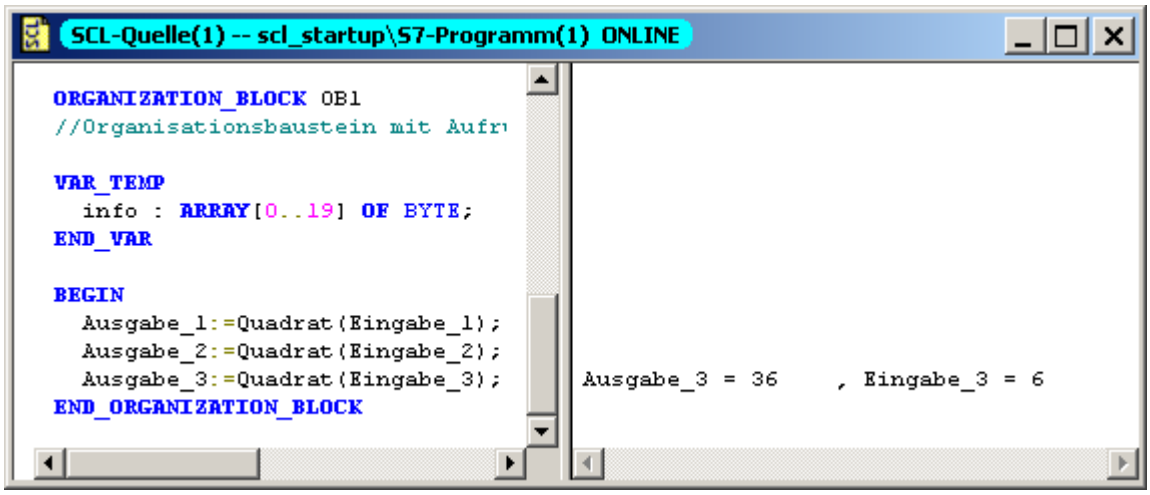
13. Wählen Sie nun den Befehl **‚Fortsetzen'** . Das Programm wird bis zum nächsten aktiven Haltepunkt abgearbeitet. Mit der Funktion **‚Zu nächster Anweisung'** werden die Werte der aktuellen Zeile angezeigt. (→ →)



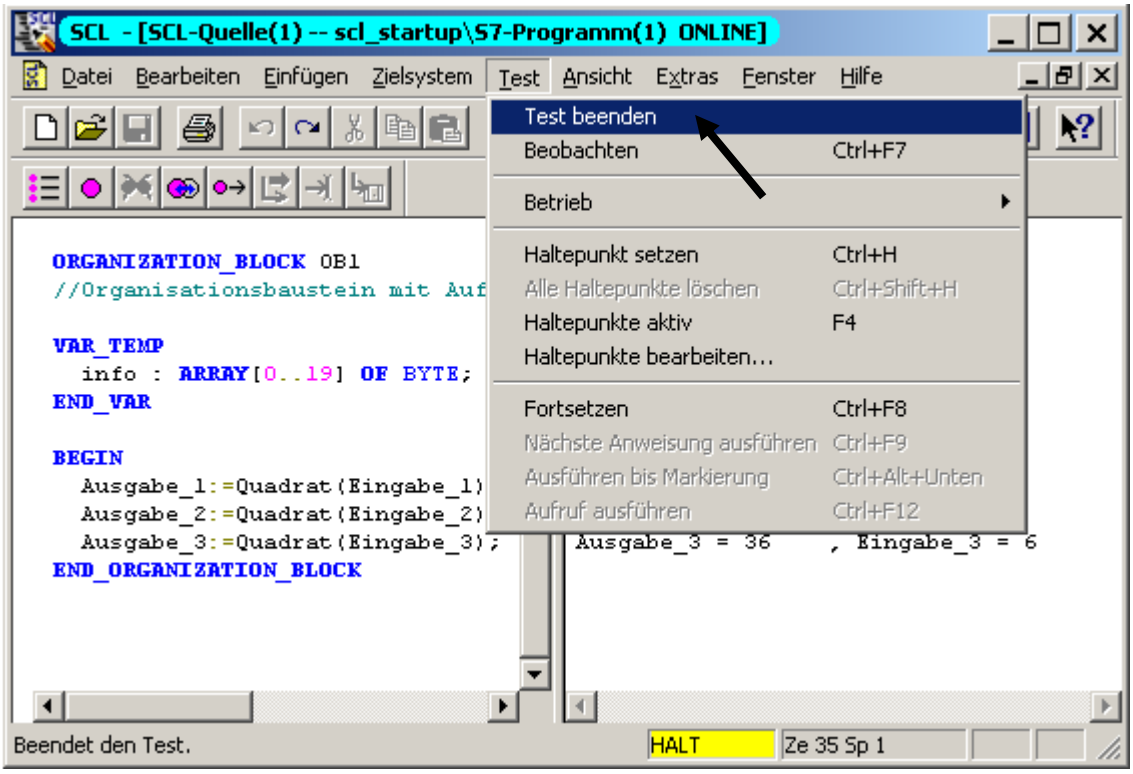
Hinweis: Durch die Funktion **‚Fortsetzen'** wird das Programm bis einschließlich der markierten Zeile bearbeitet. Die Anzeige wird jedoch nicht aktualisiert.

Diese wird durch die Funktion **‚Zu nächster Anweisung'** aktualisiert.

14. Wenn Sie die Haltepunkte ausschalten und löschen , befinden Sie sich immer noch im Testbetrieb. Das Teilfenster für die aktuellen Werte ist noch eingeblendet. (→ →)



15. Die Testfunktion muss im Menü **‚Test'** beendet werden. (→ Test → Test beenden)



Hinweis: Solange die Testfunktionen aktiv sind, kann im Texteditor keine Veränderung vorgenommen werden. Erst wenn der Test beendet wird, kann die SCL-Quelle wieder bearbeitet werden.