

SIMATIC

S7-SCL V5.3 für S7-300/400

Handbuch

Vorwort, Inhaltsverzeichnis	
Produktübersicht	1
Installation	2
Entwerfen eines S7-SCL-Programms	3
Bedienen von S7-SCL	4
S7-SCL-Grundbegriffe	5
S7-SCL-Programmstruktur	6
Datentypen	7
Deklaration lokaler Variablen und Parameter	8
Vereinbarung von Konstanten und Sprungmarken	9
Globale Daten	10
Ausdrücke, Operationen und Operanden	11
Anweisungen	12
Zähler und Zeiten	13
Standardfunktionen von S7-SCL	14
Sprachbeschreibung	15
Tipps und Tricks	16
Glossar, Index	

Sicherheitstechnische Hinweise

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.



Gefahr

bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.



Warnung

bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.



Vorsicht

mit Warndreieck bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht

ohne Warndreieck bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Achtung

bedeutet, dass ein unerwünschtes Ergebnis oder Zustand eintreten kann, wenn der entsprechende Hinweis nicht beachtet wird.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zugehörige Gerät/System darf nur in Verbindung mit dieser Dokumentation eingerichtet und betrieben werden. Inbetriebsetzung und Betrieb eines Gerätes/Systems dürfen nur von **qualifiziertem Personal** vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieser Dokumentation sind Personen, die die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Bestimmungsgemäßer Gebrauch

Beachten Sie Folgendes:



Warnung

Das Gerät darf nur für die im Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Fremdgeräten und -komponenten verwendet werden. Der einwandfreie und sichere Betrieb des Produktes setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

Copyright Siemens AG 2005 All rights reserved

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten, insbesondere für den Fall der Patenterteilung oder GM-Eintragung.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Vorwort

Zweck des Handbuchs

Dieses Handbuch gibt Ihnen einen vollständigen Überblick über das Programmieren mit S7-SCL. Es unterstützt Sie bei der Installation und Inbetriebnahme der Software. Die Vorgehensweise bei der Programmerstellung, den Aufbau von Anwenderprogrammen und die einzelnen Sprachelemente werden erläutert.

Es richtet sich an Programmierer von S7-SCL-Programmen und an Personen, die in den Bereichen Projektierung, Inbetriebsetzung und Service von Automatisierungssystemen tätig sind.

Wir empfehlen Ihnen, sich mit dem Beispiel aus Kapitel 2 "Entwerfen eines S7-SCL-Programms" vertraut zu machen. Es bietet einen leichten Einstieg in die Programmierung mit S7-SCL.

Erforderliche Grundkenntnisse

Zum Verständnis des Handbuchs sind allgemeine Kenntnisse auf dem Gebiet der Automatisierungstechnik erforderlich.

Außerdem werden Kenntnisse über die Verwendung von Computern oder PC-ähnlichen Arbeitsmitteln (z. B. Programmiergeräten) unter dem Betriebssystem MS Windows 2000 Professional bzw. MS Windows XP Professional vorausgesetzt. Da S7-SCL auf der Basissoftware STEP 7 aufsetzt, sollten Sie auch Kenntnisse im Umgang mit der Basissoftware haben, die im Handbuch "Programmieren mit STEP 7 V5.3" vermittelt werden.

Gültigkeitsbereich des Handbuchs

Das Handbuch ist gültig für das Softwarepaket S7-SCL V5.3 ab Service Pack1.

Dokumentationspakete zu S7-SCL und zur Basissoftware STEP 7

Die folgende Tabelle zeigt die Dokumentation zu STEP 7 und S7-SCL im Überblick:

Handbücher	Zweck	Bestell-Nummer
SCL Grund- und Referenzwissen mit <ul style="list-style-type: none"> S7-SCL für S7-300/400: Bausteine programmieren 	Grund- und Referenzwissen, das die Vorgehensweise bei der Programmerstellung, den Aufbau von Anwenderprogrammen und die einzelnen Sprachelemente erläutert.	Das Handbuch ist nicht einzeln bestellbar. Es ist auf der Produkt-CD, in der Manual Collection und im Internet verfügbar.
STEP 7-Grundwissen mit <ul style="list-style-type: none"> Erste Schritte und Übungen mit STEP 7 V5.3 Programmieren mit STEP 7 V5.3 Hardware konfigurieren und Verbindungen projektieren mit STEP 7 V5.3 Von S5 nach S7, Umsteigerhandbuch 	Das Grundwissen für technisches Personal, das das Vorgehen zur Realisierung von Steuerungsaufgaben mit STEP 7 und S7-300/400 beschreibt.	6ES7810-4CA07-8AW0
STEP 7-Referenzwissen mit <ul style="list-style-type: none"> Handbücher KOP/FUP/AWL für S7-300/400 Standard- und Systemfunktionen für S7-300/400 	Das Referenzwissen zum Nachschlagen, das die Programmiersprachen KOP, FUP und AWL sowie Standard- und Systemfunktionen ergänzend zum STEP 7-Grundwissen beschreibt.	6ES7810-4CA07-8AW1

Online-Hilfen	Zweck	Bestell-Nummer
Hilfe zu S7-SCL	Grund- und Referenzwissen zu S7-SCL als Online-Hilfe	Bestandteil des Softwarepaketes S7-SCL
Hilfe zu STEP 7	Das Grundwissen zum Programmieren und Hardware konfigurieren mit STEP 7 als Online-Hilfe	Bestandteil des Softwarepaketes STEP 7
Referenzhilfen zu AWL/KOP/FUP Referenzhilfe zu SFBs/SFCs Referenzhilfe zu Organisationsbausteinen Referenzhilfe zu IEC-Funktionen Referenzhilfe zu Systemattributen	Kontextsensitives Referenzwissen	Bestandteile des Softwarepaketes STEP 7

Online-Hilfe

Die Online-Hilfe bietet Ihnen Informationen an der Stelle an, an der Sie sie benötigen. So können Sie schnell und zielsicher Information nachschlagen, ohne in Handbüchern suchen zu müssen. In der Online-Hilfe finden Sie:

- **Hilfethemen:** bietet verschiedene Zugänge zum Anzeigen von Hilfeinformation.
- **Hilfe zum Kontext** (Taste F1): zeigt Informationen zum markierten Objekt oder zum aktiven Dialogfeld bzw. Fenster an.
- **Einführung:** gibt einen knappen Überblick über Anwendung, wesentliche Merkmale und Funktionsumfang der Applikation.
- **Erste Schritte:** fasst erste Handlungen zusammen, die Sie durchführen müssen, um zu einem ersten Erfolg zu kommen.
- **Hilfe benutzen:** bietet eine Beschreibung der Möglichkeiten, die Ihnen zur Verfügung stehen, um bestimmte Informationen in der Hilfe zu finden.
- **Info:** liefert Informationen zur aktuellen Version der Applikation.

Weitere Unterstützung

Bei Fragen zur Nutzung der im Handbuch beschriebenen Produkte, die Sie hier nicht beantwortet finden, wenden Sie sich bitte an Ihren Siemens-Ansprechpartner in den für Sie zuständigen Vertretungen und Geschäftsstellen.

Ihren Ansprechpartner finden Sie unter:

<http://www.siemens.com/automation/partner>

Den Wegweiser zum Angebot an technischen Dokumentationen für die einzelnen SIMATIC Produkte und Systeme finden Sie unter:

<http://www.siemens.de/simatic-tech-doku-portal>

Den Online-Katalog und das Online-Bestellsystem finden Sie unter:

<http://mall.automation.siemens.com/>

Trainingscenter

Um Ihnen den Einstieg in das Automatisierungssystem S7 zu erleichtern, bieten wir entsprechende Kurse an. Wenden Sie sich bitte an Ihr regionales Trainingscenter oder an das zentrale Trainingscenter in D 90327 Nürnberg.

Telefon: +49 (911) 895-3200.

Internet: <http://www.sitrain.com>

Technical Support

Sie erreichen den Technical Support für alle A&D-Produkte

- Über das Web-Formular für den Support Request
<http://www.siemens.de/automation/support-request>
- Telefon: + 49 180 5050 222
- Fax: + 49 180 5050 223

Weitere Informationen zu unserem Technical Support finden Sie im Internet unter
<http://www.siemens.com/automation/service>

Service & Support im Internet

Zusätzlich zu unserem Dokumentations-Angebot bieten wir Ihnen im Internet unser komplettes Wissen online an.

<http://www.siemens.com/automation/service&support>

Dort finden Sie:

- den Newsletter, der Sie ständig mit den aktuellsten Informationen zu Ihren Produkten versorgt.
- die für Sie richtigen Dokumente über unsere Suche in Service & Support.
- ein Forum, in welchem Anwender und Spezialisten weltweit Erfahrungen austauschen.
- Ihren Ansprechpartner für Automation & Drives vor Ort.
- Informationen über Vor-Ort Service, Reparaturen, Ersatzteile. Vieles mehr steht für Sie unter dem Begriff "Leistungen" bereit.

Inhaltsverzeichnis

1	Produktübersicht	1-1
1.1	Anwendungsbereich von S7-SCL	1-1
1.2	Arbeitsweise von S7-SCL	1-2
1.3	Welche Funktionen bietet S7-SCL?	1-4
1.4	Was ist neu in Version V5.3 SP1?	1-6
2	Installation	2-1
2.1	Automation License Manager	2-1
2.1.1	Nutzungsberechtigung durch den Automation License Manager	2-1
2.1.2	Installieren des Automation License Managers	2-4
2.1.3	Regeln für den Umgang mit License Keys.....	2-5
2.2	Installation	2-6
2.2.1	Installationsvoraussetzungen.....	2-6
2.2.2	Installieren von S7-SCL	2-6
3	Entwerfen eines S7-SCL-Programms	3-1
3.1	Willkommen beim Einsteigerbeispiel "Messwerterfassung"	3-1
3.2	Aufgabenstellung	3-2
3.3	Aufbau eines strukturierten Programms mit S7-SCL.....	3-4
3.4	Festlegen der Teilaufgaben	3-6
3.5	Festlegen der Schnittstellen zwischen den Bausteinen.....	3-8
3.6	Festlegen der Ein-/Ausgabe Schnittstelle	3-10
3.7	Festlegen der Reihenfolge der Bausteine in der Quelle	3-11
3.8	Festlegen der Symbole	3-11
3.9	Erstellen der Funktion QUADRAT	3-12
3.9.1	Anweisungsteil der Funktion QUADRAT	3-12
3.10	Erstellen des Funktionsbausteins AUSWERTEN	3-13
3.10.1	Flussdiagramm von AUSWERTEN.....	3-13
3.10.2	Vereinbarungsteil des FB AUSWERTEN.....	3-14
3.10.3	Anweisungsteil des FB AUSWERTEN.....	3-15
3.11	Erstellen des Funktionsbausteins ERFASSEN.....	3-17
3.11.1	Flussdiagramm von Erfassen.....	3-17
3.11.2	Vereinbarungsteil des FB ERFASSEN	3-18
3.11.3	Anweisungsteil des FB ERFASSEN	3-20
3.12	Erstellen des Organisationsbausteins ZYKLUS.....	3-23
3.13	Testdaten	3-25

4	Bedienen von S7-SCL	4-1
4.1	Starten der S7-SCL-Software	4-1
4.2	Bedienoberfläche	4-2
4.3	Anpassen der Bedienoberfläche	4-3
4.4	Anlegen und Hantieren einer S7-SCL-Quelle	4-4
4.4.1	Anlegen einer neuen S7-SCL-Quelle.....	4-4
4.4.2	Öffnen einer S7-SCL-Quelle	4-5
4.4.3	Schließen einer S7-SCL-Quelle	4-5
4.4.4	Öffnen von Bausteinen.....	4-6
4.4.5	Festlegen der Objekteigenschaften	4-6
4.4.6	Erzeugen von S7-SCL-Quellen mit einem Standard-Editor.....	4-6
4.4.7	Bausteinschutz einrichten	4-7
4.5	Richtlinien für S7-SCL-Quellen	4-7
4.5.1	Allgemeine Regeln für S7-SCL-Quellen	4-7
4.5.2	Reihenfolge der Bausteine.....	4-8
4.5.3	Verwendung symbolischer Adressen.....	4-8
4.6	Editieren in S7-SCL-Quellen	4-9
4.6.1	Rückgängigmachen der letzten Editieraktion.....	4-9
4.6.2	Wiederherstellen einer Editieraktion	4-9
4.6.3	Suchen und Ersetzen von Textobjekten	4-9
4.6.4	Markieren von Textobjekten.....	4-10
4.6.5	Kopieren von Textobjekten	4-10
4.6.6	Ausschneiden von Textobjekten	4-11
4.6.7	Löschen von Textobjekten	4-11
4.6.8	Positionieren der Einfügemarke in einer bestimmten Zeile	4-12
4.6.9	Syntaxgerechtes Einrücken von Zeilen.....	4-13
4.6.10	Einstellen von Schriftstil und -farbe.....	4-14
4.6.11	Platzieren von Lesezeichen im Quelltext	4-15
4.6.12	Einfügen von Vorlagen.....	4-16
4.6.12.1	Einfügen von Bausteinvorlagen	4-16
4.6.12.2	Einfügen von Bausteinaufrufen.....	4-16
4.6.12.3	Einfügen von Vorlagen für Kommentar.....	4-16
4.6.12.4	Einfügen von Parametervorlagen	4-17
4.6.12.5	Einfügen von Kontrollstrukturen.....	4-17
4.7	Übersetzen eines S7-SCL-Programms.....	4-18
4.7.1	Wissenswertes zum Übersetzen.....	4-18
4.7.2	Einstellen des Compilers.....	4-19
4.7.3	Übersetzen des Programms	4-20
4.7.4	Erstellen einer Übersetzungssteuerdatei	4-21
4.7.5	Beheben von Fehlern nach dem Übersetzen	4-21
4.8	Speichern und Drucken einer S7-SCL-Quelle	4-22
4.8.1	Speichern einer S7-SCL-Quelle.....	4-22
4.8.2	Einstellen des Seitenformats	4-22
4.8.3	Drucken einer S7-SCL-Quelle.....	4-23
4.8.4	Einstellen der Druckoptionen	4-24
4.9	Laden der erstellten Programme	4-25
4.9.1	Urlöschen des CPU-Speichers	4-25
4.9.2	Laden von Anwenderprogrammen in die CPU	4-25
4.10	Testen der erstellten Programme	4-27
4.10.1	Die Testfunktionen von S7-SCL.....	4-27
4.10.2	Wissenswertes zur Testfunktion "Beobachten"	4-28
4.10.3	Wissenswertes zum "Testen mit Haltepunkten/Einzelschrittmodus"	4-30

4.10.4	Schritte zum Beobachten	4-31
4.10.4.1	Definieren einer Aufrufumgebung für Bausteine	4-32
4.10.5	Schritte zum Testen mit Haltepunkten	4-33
4.10.5.1	Definieren von Haltepunkten	4-33
4.10.5.2	Starten des Tests mit Haltepunkten	4-33
4.10.5.3	Beenden des Tests mit Haltepunkten	4-34
4.10.5.4	Aktivieren, Deaktivieren und Löschen von Haltepunkten	4-34
4.10.5.5	Definieren einer Aufrufumgebung für Haltepunkte	4-35
4.10.5.6	Testen im Einzelschrittmodus	4-36
4.10.6	Verwenden der STEP 7-Testfunktionen	4-37
4.10.6.1	Erzeugen bzw. Anzeigen der Referenzdaten	4-37
4.10.6.2	Beobachten und Steuern von Variablen	4-38
4.10.6.3	Prüfen der Bausteinkonsistenz	4-38
4.11	Anzeigen und Ändern von CPU-Eigenschaften	4-40
4.11.1	Anzeigen und Ändern des Betriebszustands der CPU	4-40
4.11.2	Anzeigen und Einstellen von Datum und Uhrzeit der CPU	4-40
4.11.3	Anzeigen der CPU-Daten	4-41
4.11.4	Auslesen des Diagnosepuffers der CPU	4-41
4.11.5	Anzeigen/Komprimieren des Anwenderspeichers der CPU	4-41
4.11.6	Anzeigen der Zykluszeit der CPU	4-42
4.11.7	Anzeigen des Zeitsystems der CPU	4-42
4.11.8	Anzeigen der Bausteine in der CPU	4-42
4.11.9	Anzeigen der Informationen zur Kommunikation der CPU	4-43
4.11.10	Anzeigen der Stacks der CPU	4-43
5	S7-SCL-Grundbegriffe	5-1
5.1	Interpretation der Syntaxdiagramme	5-1
5.2	Zeichensatz	5-4
5.3	Reservierte Wörter	5-5
5.4	Bezeichner	5-6
5.5	Standardbezeichner	5-7
5.6	Baustein-Bezeichnung	5-7
5.7	Operandenkennzeichen	5-9
5.8	Timer-Bezeichnung	5-10
5.9	Zähler-Bezeichnung	5-10
5.10	Zahlen	5-11
5.11	Zeichenketten	5-13
5.12	Symbol	5-14
5.13	Kommentarblock	5-15
5.14	Zeilenkommentar	5-16
5.15	Variablen	5-17

6	S7-SCL-Programmstruktur	6-1
6.1	Bausteine in S7-SCL-Quellen	6-1
6.2	Reihenfolge der Bausteine	6-2
6.3	Allgemeiner Aufbau eines Bausteins	6-3
6.4	Bausteinanfang und -ende	6-3
6.5	Bausteinattribute	6-5
6.6	Bausteinkommentar	6-8
6.7	Systemattribute für Bausteine	6-9
6.8	Vereinbarungsteil	6-10
6.9	Systemattribute für Parameter	6-11
6.10	Anweisungsteil	6-12
6.11	Anweisungen	6-13
6.12	Aufbau eines Funktionsbausteins (FB)	6-14
6.13	Aufbau einer Funktion (FC)	6-16
6.14	Aufbau eines Organisationsbausteins (OB)	6-18
6.15	Aufbau eines Datenbausteins (DB)	6-19
6.16	Aufbau eines anwenderdefinierten Datentyps	6-22
6.17	Compileroptionen in S7-SCL-Quellen	6-24
7	Datentypen	7-1
7.1	Übersicht über die Datentypen in S7-SCL	7-1
7.2	Elementare Datentypen	7-3
7.2.1	Bitdatentypen	7-3
7.2.2	Zeichentypen	7-3
7.2.3	Numerische Datentypen	7-3
7.2.4	Zeittypen	7-4
7.3	Zusammengesetzte Datentypen	7-5
7.3.1	Datentyp DATE_AND_TIME	7-5
7.3.2	Datentyp STRING	7-7
7.3.3	Datentyp ARRAY	7-10
7.3.4	Datentyp STRUCT	7-12
7.4	Anwenderdefinierte Datentypen	7-14
7.4.1	Anwenderdefinierte Datentypen (UDT)	7-14
7.5	Datentypen für Parameter	7-16
7.5.1	Datentypen TIMER und COUNTER	7-16
7.5.2	BLOCK-Datentypen	7-17
7.5.3	Datentyp POINTER	7-17
7.6	Datentyp ANY	7-19
7.6.1	Beispiel zum Datentyp ANY	7-20
8	Deklaration lokaler Variablen und Parameter	8-1
8.1	Lokale Variablen und Bausteinparameter	8-1
8.2	Allgemeine Syntax einer Variablen- oder Parameterdeklaration	8-3
8.3	Initialisierung	8-4
8.4	Deklarieren von Sichten auf Variablenbereiche	8-6
8.5	Verwendung von Multiinstanzen	8-8
8.6	Instanzdeklaration	8-8
8.7	Flags (OK-Flag)	8-9
8.8	Deklarationsabschnitte	8-10
8.8.1	Übersicht der Deklarationsabschnitte	8-10
8.8.2	Statische Variablen	8-11
8.8.3	Temporäre Variablen	8-12
8.8.4	Bausteinparameter	8-13

9	Vereinbarung von Konstanten und Sprungmarken	9-1
9.1	Konstanten	9-1
9.1.1	Vereinbarung symbolischer Namen für Konstanten	9-2
9.1.2	Datentypen von Konstanten	9-3
9.1.3	Schreibweisen von Konstanten	9-4
9.1.3.1	Bit-Konstanten	9-6
9.1.3.2	Ganzzahl-Konstante	9-7
9.1.3.3	Realzahl-Konstante	9-8
9.1.3.4	Char-Konstante (Einzelzeichen)	9-9
9.1.3.5	String-Konstante	9-11
9.1.3.6	Datumskonstante	9-13
9.1.3.7	Zeitdauer-Konstante	9-13
9.1.3.8	Tageszeit-Konstanten	9-16
9.1.3.9	Datum- und Zeitkonstante	9-17
9.2	Vereinbarung von Sprungmarken	9-18
10	Globale Daten	10-1
10.1	Übersicht über globale Daten	10-1
10.2	Speicherbereiche der CPU	10-2
10.2.1	Übersicht der Speicherbereiche der CPU	10-2
10.2.2	Absoluter Zugriff auf Speicherbereiche der CPU	10-3
10.2.3	Symbolischer Zugriff auf Speicherbereiche der CPU	10-5
10.2.4	Indizierter Zugriff auf Speicherbereiche der CPU	10-6
10.3	Datenbausteine	10-7
10.3.1	Übersicht der Datenbausteine	10-7
10.3.2	Absoluter Zugriff auf Datenbausteine	10-8
10.3.3	Indizierter Zugriff auf Datenbausteine	10-10
10.3.4	Strukturierter Zugriff auf Datenbausteine	10-11
11	Ausdrücke, Operationen und Operanden	11-1
11.1	Übersicht der Ausdrücke, Operationen und Operanden	11-1
11.2	Operationen	11-2
11.3	Operanden	11-3
11.4	Syntax eines Ausdrucks	11-5
11.5	Einfacher Ausdruck	11-7
11.6	Arithmetische Ausdrücke	11-8
11.7	Logische Ausdrücke	11-10
11.8	Vergleichsausdrücke	11-12
12	Anweisungen	12-1
12.1	Wertzuweisungen	12-1
12.1.1	Wertzuweisungen mit Variablen eines elementaren Datentyps	12-2
12.1.2	Wertzuweisungen mit Variablen vom Typ STRUCT und UDT	12-3
12.1.3	Wertzuweisungen mit Variablen vom Typ ARRAY	12-5
12.1.4	Wertzuweisungen mit Variablen vom Typ STRING	12-7
12.1.5	Wertzuweisungen mit Variablen vom Typ DATE_AND_TIME	12-8
12.1.6	Wertzuweisungen mit Absolutvariablen für Speicherbereiche	12-9
12.1.7	Wertzuweisungen mit globalen Variablen	12-10
12.2	Kontrollanweisungen	12-12
12.2.1	Übersicht der Kontrollanweisungen	12-12
12.2.2	Bedingungen	12-13
12.2.3	IF-Anweisung	12-14
12.2.4	CASE-Anweisung	12-16
12.2.5	FOR-Anweisung	12-18

12.2.6	WHILE-Anweisung	12-21
12.2.7	REPEAT-Anweisung	12-22
12.2.8	CONTINUE-Anweisung.....	12-23
12.2.9	EXIT-Anweisung	12-24
12.2.10	GOTO-Anweisung.....	12-25
12.2.11	RETURN-Anweisung	12-26
12.3	Aufruf von Funktionen und Funktionsbausteinen	12-27
12.3.1	Aufruf und Parameterübergabe	12-27
12.3.2	Aufruf von Funktionsbausteinen	12-29
12.3.2.1	Aufruf von Funktionsbausteinen (FB oder SFB)	12-29
12.3.2.2	Versorgung der FB-Parameter.....	12-31
12.3.2.3	Eingangszuweisung (FB)	12-33
12.3.2.4	Durchgangszuweisung (FB).....	12-34
12.3.2.5	Ausgangswerte lesen (FB-Aufruf).....	12-35
12.3.2.6	Beispiel zum Aufruf als globale Instanz	12-35
12.3.2.7	Beispiel zum Aufruf als lokale Instanz	12-37
12.3.3	Aufruf von Funktionen	12-38
12.3.3.1	Aufruf von Funktionen (FC).....	12-38
12.3.3.2	Rückgabewert (FC).....	12-39
12.3.3.3	FC-Parameter	12-40
12.3.3.4	Eingangszuweisung (FC).....	12-41
12.3.3.5	Ausgangs-/Durchgangszuweisung (FC)	12-42
12.3.3.6	Beispiel für einen Funktionsaufruf.....	12-44
12.3.4	Implizit definierte Parameter	12-45
12.3.4.1	Eingangsparameter EN.....	12-45
12.3.4.2	Ausgangsparameter ENO.....	12-46
13	Zähler und Zeiten	13-1
13.1	Zähler	13-1
13.1.1	Zählfunktionen.....	13-1
13.1.2	Aufruf von Zählfunktionen	13-1
13.1.3	Parameterversorgung bei Zählfunktionen.....	13-3
13.1.4	Eingabe und Auswertung des Zählerwerts	13-5
13.1.5	Aufwärtszähler (S_CU)	13-6
13.1.6	Abwärtszähler (S_CD)	13-6
13.1.7	Auf- und Abwärtszähler (S_CUD)	13-7
13.1.8	Beispiel zu Zählfunktionen	13-7
13.2	Zeiten	13-9
13.2.1	Zeitfunktionen.....	13-9
13.2.2	Aufruf von Zeitfunktionen	13-9
13.2.3	Parameterversorgung bei Zeitfunktionen.....	13-11
13.2.4	Eingabe und Auswertung des Zeitwerts	13-13
13.2.5	Zeit als Impuls starten (S_PULSE)	13-15
13.2.6	Zeit als verlängerter Impuls starten (S_PEXT)	13-16
13.2.7	Zeit als Einschaltverzögerung starten (S_ODT)	13-17
13.2.8	Zeit als speichernde Einschaltverzögerung starten (S_ODTS)	13-18
13.2.9	Zeit als Ausschaltverzögerung starten (S_OFFDT).....	13-19
13.2.10	Beispiel zu Zeitfunktionen	13-20
13.2.11	Auswahl des richtigen Zeitglieds.....	13-21

14	Standardfunktionen von S7-SCL	14-1
14.1	Datentyp-Konvertierungsfunktionen.....	14-1
14.1.1	Konvertierung von Datentypen	14-1
14.1.2	Implizite Datentypkonvertierung.....	14-2
14.1.2.1	Konvertierungsfunktionen Klasse A	14-2
14.1.3	Standardfunktionen zur expliziten Datentyp-Konvertierung.....	14-3
14.1.3.1	Konvertierungsfunktionen Klasse B	14-3
14.1.3.2	Funktionen zum Runden und Abschneiden	14-6
14.1.3.3	Beispiele zur Konvertierung mit Standardfunktionen	14-7
14.2	Numerische Standardfunktionen.....	14-9
14.2.1	Allgemeine arithmetische Standardfunktionen.....	14-9
14.2.2	Logarithmische Funktionen	14-9
14.2.3	Trigonometrische Funktionen	14-10
14.2.4	Beispiele zu numerischen Standardfunktionen.....	14-10
14.3	Bitstring Standardfunktionen	14-11
14.3.1	Beispiele zu Bitstring Standardfunktionen	14-12
14.4	Funktionen zur Verarbeitung von Zeichenketten	14-13
14.4.1	Funktionen zur Stringmanipulation	14-13
14.4.2	Funktionen zum Stringvergleich.....	14-17
14.4.3	Funktionen zur Wandlung des Datenformats.....	14-19
14.4.4	Beispiel zur Verarbeitung von Zeichenketten	14-21
14.5	Funktionen zur Selektion von Werten	14-23
14.5.1	Funktionen zur Selektion von Werten	14-23
14.6	SFCs, SFBs und Standardbibliothek	14-27
14.6.1	Systemfunktionen/-funktionsbausteine und Standardbibliothek	14-27
14.6.2	Übergabeschnittstelle zu OB	14-29
15	Sprachbeschreibung	15-1
15.1	Formale Sprachbeschreibung.....	15-1
15.1.1	Übersicht Syntaxdiagramme	15-1
15.1.2	Regeln	15-2
15.1.3	Terminale der lexikalischen Regeln	15-5
15.1.4	Formatierungs-, Trennzeichen und Operationen	15-7
15.1.5	Schlüsselwörter und vordefinierte Bezeichner.....	15-10
15.1.6	Operandenkennzeichen und Bausteinschlüsselwörter	15-13
15.1.7	Übersicht Non-Terminale	15-14
15.1.8	Übersicht Token	15-15
15.1.9	Bezeichner	15-15
15.1.10	Namensvergabe bei S7-SCL	15-17
15.1.11	Vordefinierte Konstanten und Flags.....	15-20
15.2	Lexikalische Regeln	15-21
15.2.1	Übersicht: Lexikalische Regeln	15-21
15.2.2	Bezeichnungen	15-21
15.2.3	Konstanten	15-23
15.2.4	Absolutadressierung	15-29
15.2.5	Kommentare.....	15-31
15.2.6	Bausteinattribute	15-32
15.2.7	Compileroptionen	15-33

15.3	Syntaktische Regeln	15-34
15.3.1	Übersicht: Syntaktische Regeln	15-34
15.3.2	Gliederungen von S7-SCL-Quellen	15-35
15.3.3	Aufbau der Vereinbarungsteile	15-37
15.3.4	Datentypen in S7-SCL	15-42
15.3.5	Anweisungsteil	15-45
15.3.6	Wertzuweisungen.....	15-47
15.3.7	Aufruf von Funktionen und Funktionsbausteinen	15-50
15.3.8	Kontrollanweisungen.....	15-52
16	Tipps und Tricks	16-1

Glossar

Index

1 Produktübersicht

1.1 Anwendungsbereich von S7-SCL

S7-SCL (Structured Control Language) ist eine PASCAL-orientierte Hochsprache zur Programmierung von Speicherprogrammierbaren Steuerungen mit SIMATIC S7.

PLCopen-Zertifikat

S7-SCL entspricht der in der Norm IEC 61131-3 definierten textuellen Hochsprache ST (Structured Text) und wurde für die Zertifizierung für den Reusability Level vorbereitet.

Anwendungsbereich

S7-SCL ist optimiert für die Programmierung von Speicherprogrammierbaren Steuerungen und enthält sowohl Sprachelemente aus der Programmiersprache PASCAL als auch typische SPS-Elemente, wie z.B. Ein- / Ausgänge, Zeiten und Zähler.

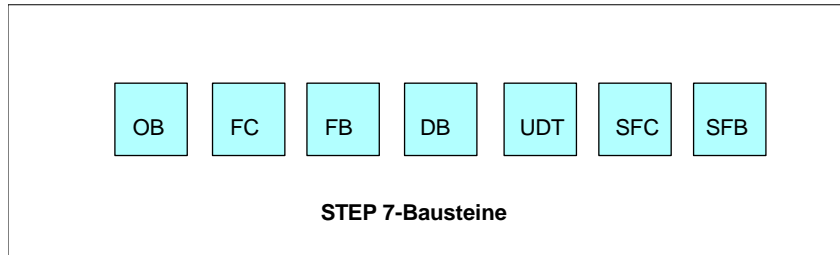
S7-SCL ist besonders für folgende Aufgaben geeignet:

- Programmierung komplexer Algorithmen
- Programmierung mathematischer Funktionen
- Daten- bzw. Rezepturverwaltung
- Prozessoptimierung

1.2 Arbeitsweise von S7-SCL

Integration in STEP 7

S7-SCL unterstützt das STEP 7-Bausteinkonzept.



Sie können folgende STEP 7-Bausteine mit S7-SCL erstellen:

- OB
- FC
- FB
- DB
- UDT

In einem S7-Programm können S7-SCL-Bausteine auch mit Bausteinen aus anderen STEP 7-Programmiersprachen kombiniert werden. Diese Bausteine können sich gegenseitig aufrufen. S7-SCL-Bausteine lassen sich auch in Bibliotheken speichern und von dort aus in anderen Sprachen verwenden.

Da S7-SCL-Programme als ASCII-Quellen programmiert werden, sind sie leicht importierbar und exportierbar.

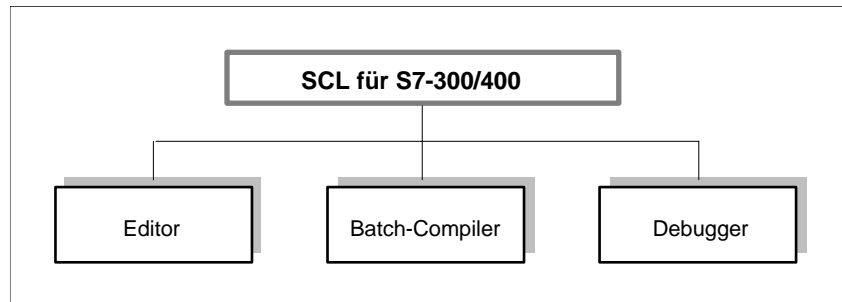
S7-SCL-Bausteine können in die STEP 7-Programmiersprache AWL (Anweisungsliste) rückübersetzt werden. Beachten Sie jedoch, dass Sie nach einmaligem Speichern in AWL nicht mehr in S7-SCL bearbeitbar sind.

Entwicklungsumgebung

Zum praktischen Einsatz bietet S7-SCL eine leistungsfähige Entwicklungsumgebung, die sowohl auf spezifische Eigenschaften von S7-SCL als auch auf STEP 7 abgestimmt ist. Die Entwicklungsumgebung besteht aus folgenden Komponenten:

- einem **Editor**, um Programme bestehend aus Funktionen (FC), Funktionsbausteinen (FB), Organisationsbausteinen (OB), Datenbausteinen (DB) und anwenderdefinierten Datentypen (UDT) zu programmieren. Der Programmierer wird dabei durch leistungsfähige Funktionen unterstützt.
- einem **Batch-Compiler** um das zuvor editierte Programm in MC7-Maschinencode zu übersetzen. Der erzeugte MC7-Code ist ab der CPU 314 auf allen CPUs des Automatisierungssystem S7-300/400 ablauffähig.
- einem **Debugger**, um eine Suche nach logischen Programmfehlern in einer fehlerfreien Übersetzung zu ermöglichen. Die Fehlersuche erfolgt dabei in der Quellsprache.

Folgendes Bild gibt einen Überblick über die Komponenten der Entwicklungsumgebung:



1.3 Welche Funktionen bietet S7-SCL?

S7-SCL verfügt über alle Vorteile einer höheren Programmiersprache. Darüber hinaus bietet S7-SCL Leistungsmerkmale, die speziell zur Unterstützung von strukturierter Programmierung entworfen wurden:

Bausteinbibliotheken

Vorgefertigte Bausteine werden in Bibliotheken mitgeliefert, z.B.:

- Systemfunktionen
- IEC-Funktionen
- Konvertierungsfunktionen

Ein Dialogfeld unterstützt die Navigation in der Bibliothek. Durch Selektion eines Bausteins wird die Parameterschablone der Funktion automatisch in die bearbeitete Datei kopiert. Sie müssen nur noch die gewünschten Parameter eingeben.

Programmvorlagen

Der S7-SCL-Editor bietet diverse Vorlagen zum Einfügen, die nur noch ausgefüllt werden müssen:

- Vorlagen für Bausteine (z.B. Funktionsbausteine, Datenbausteine) und deren Aufrufe
- Vorlagen für Baustein-Kommentare, Baustein-Parameter und Konstanten
- Vorlagen für Kontrollstrukturen (IF, CASE, FOR, WHILE, REPEAT)

Sprachelemente aus der Hochsprachenprogrammierung

Einfache, schnelle und wenig fehleranfällige Programmerstellung durch Verwendung leistungsfähiger Sprachkonstrukte, z.B.:

- Laufschleifen
- Alternativverzweigungen (IF ... THEN ... ELSE)
- Sprünge

Leichte Verständlichkeit des Programms

Folgende Leistungsmerkmale erhöhen die Lesbarkeit des Programms:

- Vollsymbolische Programmierung
- Kommentare
- Elementare und selbstdefinierte Datentypen
- Anzeige von Querverweisen
- Automatische Formatierung der Eingabe durch Einrückung
- Syntaxgerechte Einfärbung der Sprachelemente

Debugger auf Hochsprachenniveau

Der Debugger ermöglicht einen einfachen Programmtest auf Hochsprachenniveau. Er bietet folgende Funktionalität:

- Kontinuierliches Beobachten des Programmablaufs
- Schrittweises Beobachten mit Hilfe von individuell setzbaren Haltepunkten.
- Step-in-Funktionalität (Möglichkeit, während des Tests in aufgerufene Bausteine hineinzuspringen)

1.4 Was ist neu in Version V5.3 SP1?

Spracherweiterungen

S7-SCL V5.3 SP1 wurde um Sprachmittel erweitert, die die IEC 61131-3 definiert:

- Funktionen zur Verarbeitung von Zahlenwerten als interne S7-SCL-Funktionen (SEL, MAX, MIN, LIMIT, MUX)
- Unterstützung der BCD-Darstellung von Ganzzahlen durch Konvertierungsfunktionen (BCD_TO_INT, INT_TO_BCD, etc.)
- Zuweisungsoperator => für Ausgangsparameter von Funktionen
- Feld-Initialisierung mit Klammerung
- Neue Konvertierungsfunktionen (BYTE_TO_INT, INT_TO_BYTE, etc.)

Compiler-Einstellungen in der Quelle

Compiler-Einstellungen können in S7-SCL-Quellen oder Übersetzungssteuerdateien abgelegt werden. Sie haben so die Möglichkeit, die Eigenschaften einer Übersetzung quellspezifisch zu hinterlegen.

Erweiterte Testfunktionen

- Bausteininkonsistenzen und Zeitstempelkonflikte in S7-SCL-Bausteinen können mit der STEP 7-Testfunktion "Bausteininkonsistenz prüfen" ermittelt und korrigiert werden. Diese Testfunktion ist ab STEP 7 Version 5.3 SP2 verfügbar.
- Die Testfunktion "Beobachten" kann durch die Definition von Aufrufumgebungen gezielter eingesetzt werden.
- Der Beobachtungsbereich für die Testfunktion "Beobachten" kann gezielt eingeschränkt werden, indem Sie einen Abschnitt in der Quelle markieren.

Ausdruck in Farbe

S7-SCL-Quellen können farbig ausgedruckt werden.

Erweiterte Suchfunktion

S7-SCL ermöglicht nun auch das Suchen von der Cursorposition aufwärts und das Suchen innerhalb einer Markierung.

Platzieren von Lesezeichen im Quelltext

Mit Hilfe von Lesezeichen können Sie schnell innerhalb einer Quelle navigieren.

Erstellung von S7-SCL-Bausteinen mit fremdsprachigen Zeichensätzen

S7-SCL-Quellen können Texte aus fremdsprachigen Zeichensätzen enthalten. Sie können so Bausteine für den weltweiten Markt erstellen, bei denen die wesentlichen für den Benutzer sichtbaren Teile in fremdsprachigen Zeichensätzen erscheinen (z.B. symbolische Bausteinnamen, Attribute und Kommentare).

Weitere Hinweise zu fremdsprachigen Zeichensätzen entnehmen Sie bitte der Liesmich-Datei.

2 Installation

2.1 Automation License Manager

2.1.1 Nutzungsberechtigung durch den Automation License Manager

Automation Licence Manager

Für die Nutzung der Programmiersoftware wird ein produktspezifischer License Key (Nutzungsberechtigung) benötigt, dessen Installation ab der V5.3 von S7-SCL mit dem Automation License Manager durchgeführt wird.

Der Automation License Manager ist ein Software-Produkt der Siemens AG. Er wird systemübergreifend zur Handhabung von License Keys (technischer Repräsentanten von Lizenzen) eingesetzt.

Den Automation License Manager finden Sie:

- auf der Produkt-CD von STEP 7
- auf den Internetseiten des A&D Customer Support der Siemens AG als WebDownload.

Im Automation License Manager ist eine Online-Hilfe integriert, die Sie nach der Installation kontextsensitiv über die F1-Taste oder über den Menübefehl **Hilfe > Hilfe zum License Manager** aufrufen können. In dieser Hilfe erhalten Sie detaillierte Informationen zur Funktionalität und Handhabung des Automation License Managers.

Lizenzen

Für die Nutzung von lizenziert geschützten Programmpaketen von STEP 7 werden Lizenzen benötigt. Eine Lizenz wird als Recht zur Nutzung von Produkten vergeben. Die Repräsentanten dieses Rechtes sind:

- das CoL (**Certificate of License**) und
- der License Key.

Certificate of License (CoL)

Das im Lieferumfang der jeweiligen Produkte enthaltene "Certificate of License" ist der juristische Nachweis des Nutzungsrechtes. Das jeweilige Produkt darf nur durch den Besitzer des CoL oder beauftragte Personen genutzt werden.

License Keys

Der License Key ist der technische Repräsentant einer Lizenz (elektronischer Lizenzstempel).

Für jede Software, die lizenzrechtlich geschützt ist, wird von der SIEMENS AG ein License Key vergeben. Erst wenn nach dem Starten der Software auf einem Rechner das Vorhandensein eines gültigen License Keys festgestellt wurde, kann die jeweilige Software entsprechend der mit diesem License Key verbundenen Lizenz- und Nutzungsbedingungen genutzt werden.

Hinweise

- Sie können die Software von S7-SCL zum kurzen Kennen lernen von Bedienoberfläche und Funktionsumfang auch ohne License Key verwenden.
- Die uneingeschränkte Nutzung unter Berücksichtigung der lizenzrechtlichen Vereinbarungen ist jedoch nur mit installiertem License Key zulässig und möglich.
- Wenn Sie den License Key **nicht** installiert haben, werden Sie in regelmäßigen Abständen aufgefordert, die Installation vorzunehmen.

License Keys können wie folgt abgelegt sein und zwischen den einzelnen Speichermedien transferiert werden:

- auf License Key-Disketten,
- auf lokalen Festplattenspeichern und
- auf Festplattenspeichern von Rechnern im Netzwerk.

Weiterführende Informationen über die Handhabung von License Keys entnehmen Sie bitte der Online-Hilfe zum Automation License Manager.

Lizenz-Typen

Für Software-Produkte der Siemens AG wird zwischen folgenden anwendungsorientierten Lizenz-Typen unterschieden. Das Verhalten der Software wird durch die für diese Lizenz-Typen unterschiedlichen License Keys gesteuert. Die Art der Nutzung ergibt sich aus dem jeweiligen Certificate of License.

Lizenz-Typ	Beschreibung
Single License	Die Nutzung der Software ist zeitlich unbegrenzt auf einem beliebigen Rechner zulässig.
Floating License	Zeitlich unbegrenzte, auf Nutzung über ein Netzwerk bezogenes Nutzungsrecht ("remote"Nutzung) einer Software.
Trial License	Die Nutzung der Software ist beschränkt auf: <ul style="list-style-type: none"> • eine Gültigkeit von maximal 14 Tagen, • eine bestimmte Anzahl von Tagen ab der Erstnutzung, • die Nutzung für Tests und zur Validierung (Haftungsausschluss).
Rental License	Die Nutzung der Software ist beschränkt auf: <ul style="list-style-type: none"> • eine Gültigkeit von maximal 50 Tagen • eine bestimmte Anzahl von Stunden bei Nutzung
Upgrade License	Für ein Upgrade können spezifische Anforderungen an den Systemzustand gefordert sein: <ul style="list-style-type: none"> • Mit einer Upgrade License kann eine Lizenz einer "Alt-"Version x auf eine Version >x+... umgestellt werden. • Ein Upgrade kann z. B. durch eine Erweiterung des Mengengerüsts notwendig sein.

2.1.2 Installieren des Automation License Managers

Der Automation License Manager wird über ein Setup installiert. Die Installations-Software für den Automation License Manager finden Sie auf der Produkt-CD von STEP 7.

Sie können den Automation License Manager im Zusammenhang mit S7-SCL oder erst zu einem späteren Zeitpunkt installieren.

Hinweise

- Detaillierte Informationen zur Vorgehensweise beim Installieren des Automation License Managers entnehmen Sie bitte dessen aktueller Liesmich.wri.
 - In der Online-Hilfe zum Automation License Manager erhalten Sie alle benötigten Informationen zur Funktionalität und Handhabung von License Keys.
-

License Keys später installieren

Wenn Sie die Software starten und keine License Keys vorhanden sind, so erhalten Sie eine entsprechende Meldung.

Hinweise

- Sie können die Software zum kurzen Kennen lernen von Bedienoberfläche und Funktionsumfang auch ohne License Key verwenden.
 - Die uneingeschränkte Nutzung unter Berücksichtigung der lizenzrechtlichen Vereinbarungen ist jedoch nur mit installiertem License Key zulässig und möglich.
 - Wenn Sie den License Key **nicht** installiert haben, werden Sie in regelmäßigen Abständen aufgefordert, die Installation vorzunehmen.
-

Zum nachträglichen Installieren von License Keys haben Sie folgende Möglichkeiten:

- Installieren der License Keys von Disketten
- Installieren der License Keys über WebDownLoad (vorherige Bestellung erforderlich)
- Nutzung von im Netzwerk vorhandenen Floating License Keys.

Detaillierte Informationen zur Vorgehensweise entnehmen Sie bitte der Online-Hilfe zum Automation License Manager, die Sie nach der Installation kontextsensitiv über die F1-Taste oder über den Menübefehl **Hilfe > Hilfe zum License Manager** aufrufen können.

Hinweise

- License Keys sind unter Windows 2000/XP nur dann funktionsfähig, wenn sie auf einem Festplattenlaufwerk liegen, auf dem schreibende Zugriffe zugelassen sind.
 - Floating Licenses können auch über ein Netzwerk, also "remote" genutzt werden.
-

2.1.3 Regeln für den Umgang mit License Keys



Vorsicht

Beachten Sie die Hinweise zum Umgang mit License Keys, die in der Online-Hilfe und der Liesmich.wri zum Automation License Manager beschrieben. Bei Nichtbeachtung besteht die Gefahr, dass License Keys unwiderruflich verloren gehen.

Die Online-Hilfe zum Automation License Manager können Sie kontextsensitiv über die F1-Taste oder über den Menübefehl **Hilfe > Hilfe zum Automation License Manager** aufrufen.

In dieser Hilfe erhalten Sie alle benötigten Informationen zur Funktionalität und Handhabung von License Keys.

2.2 Installation

2.2.1 Installationsvoraussetzungen

Systemvoraussetzungen

Das Optionspaket S7-SCL V5.3 SP1 ist ablauffähig auf PG/PC mit einer Installation des Basispakets STEP 7 V5.3 oder höher.

Voraussetzungen bezüglich des Betriebssystems entnehmen Sie bitte der Datei Liesmich.wri.

Hardwarevoraussetzungen

Für S7-SCL bestehen dieselben Einsatzvoraussetzungen wie für das STEP 7-Basispaket. Darüber hinaus benötigt das Optionspaket S7-SCL V5.3 SP1 Speicherkapazität, die Sie der Datei Liesmich.wri entnehmen können.

2.2.2 Installieren von S7-SCL

Installationsprogramm starten

S7-SCL enthält ein Setup-Programm, das die Installation automatisch durchführt. Eingabeaufforderungen auf dem Bildschirm führen Sie Schritt für Schritt durch den gesamten Installationsvorgang.

Gehen Sie folgendermaßen vor:

1. Starten Sie unter Windows den Dialog zur Installation von Software durch Doppelklick auf das Symbol "Software" in "Systemsteuerung".
2. Klicken Sie auf "Installieren".
3. Legen Sie den Datenträger ein und klicken Sie auf "Weiter". Windows sucht nun selbstständig nach dem Installationsprogramm "Setup.exe".
4. Befolgen Sie Schritt für Schritt die Anweisungen, die Ihnen das Installationsprogramm anzeigt.

Zum Installieren von License Keys

Bei der Installation wird überprüft, ob ein entsprechender License Key auf der Festplatte vorhanden ist. Wird kein gültiger License Key erkannt, so erscheint ein Hinweis, dass die Software nur mit vorhandenem License Key benutzt werden kann. Wenn Sie es wünschen, können Sie gleich die License Keys installieren oder aber die Installation fortsetzen und die License Keys zu einem späteren Zeitpunkt nachinstallieren. Im erstgenannten Fall legen Sie die mitgelieferte License Key-Diskette ein, wenn Sie dazu aufgefordert werden.

3 Entwerfen eines S7-SCL-Programms

3.1 Willkommen beim Einsteigerbeispiel "Messwerterfassung"

Was werden Sie lernen

Das Einsteigerbeispiel soll Ihnen zeigen, wie Sie S7-SCL effektiv einsetzen können. Fragen, die am Anfang häufig auftreten sind z. B.:

- Wie kann ich beim Programmwurf mit S7-SCL vorgehen?
- Welche Sprachmittel von S7-SCL bieten sich zur Lösung der Aufgabe an?
- Welche Testfunktionen stehen mir zur Verfügung?

Diese und andere Fragen sollen in diesem Kapitel beantwortet werden.

Verwendete S7-SCL-Sprachmittel

Im Beispiel werden u.a. folgende S7-SCL-Sprachelemente vorgestellt:

- Aufbau und Einsatz der verschiedenen Bausteinarten von S7-SCL
- Aufruf der Bausteine mit Parameter-Übergabe und -Auswertung
- Verschiedene Ein- und Ausgabeformate
- Programmierung mit elementaren Datentypen und Feldern
- Initialisierung von Variablen
- Programmstrukturen mit Verwendung von Verzweigungen und Schleifen.

Benötigte Hardware

Sie können das Beispielprogramm mit SIMATIC S7-300 oder SIMATIC S7-400 zum Ablauf bringen und benötigen dazu folgende Peripherie:

- eine Eingabebaugruppe mit 16 Kanälen
- eine Ausgabebaugruppe mit 16 Kanälen

Verfügbare Testfunktionen

Das Programm ist so aufgebaut, dass Sie es schnell über die Schalter an der Eingabe und die Anzeigen an der Ausgabe testen können. Für einen ausführlichen Test benutzen Sie am besten die Testfunktionen von S7-SCL.

Weiterhin stehen Ihnen sprachübergreifend alle Möglichkeiten des STEP 7-Basispakets zur Verfügung.

3.2 Aufgabenstellung

Übersicht

Messwerte sollen über eine Eingabebaugruppe erfasst und daraufhin durch ein S7-SCL-Programm sortiert und verarbeitet werden. Die Ergebnisse sollen über einer Ausgabebaugruppe angezeigt werden.

Messwerte erfassen

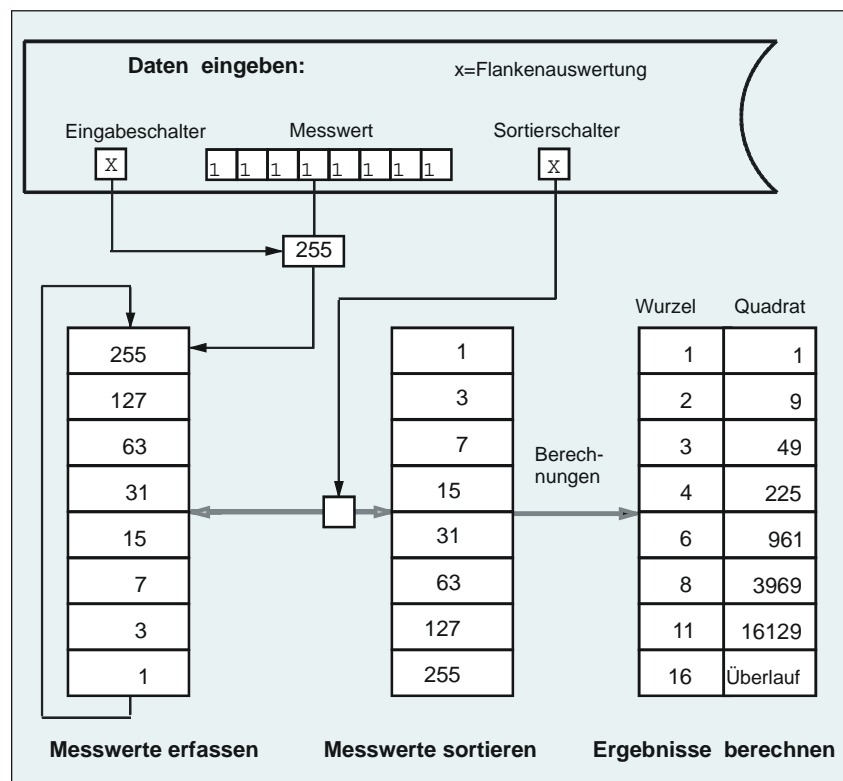
Ein Messwert wird über die 8 Eingabeschalter eingestellt. Er soll genau dann in das Messwertefeld im Speicher übernommen werden, wenn an einem Eingabeschalter eine Flanke erkannt wird (siehe auch folgendes Bild).

Der geforderte Wertebereich der Messwerte beträgt 0 bis 255. Für die Eingabe wird daher ein Byte benötigt.

Messwerte verarbeiten

Das Messwertefeld soll als Ringpuffer mit maximal 8 Einträgen organisiert sein.

Wenn an einem Sortierschalter eine Flanke erkannt wird, so sind die im Messwertefeld gespeicherten Werte aufsteigend zu sortieren. Danach sollen für jeden Wert Berechnungen von Wurzel und Quadrat durchgeführt werden. Für die Verarbeitungsfunktionen wird ein Wort benötigt.



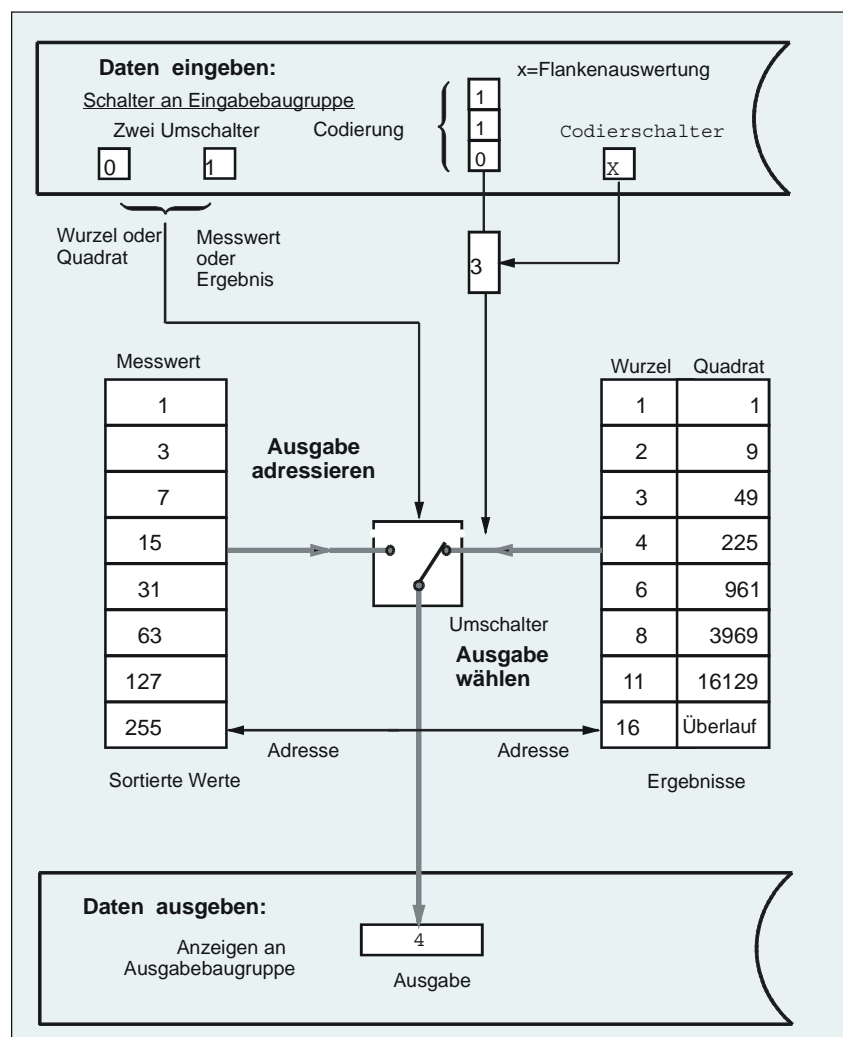
Einstellbare Ausgänge

An der Ausgabebaugruppe kann immer nur ein Wert angezeigt werden. Deshalb soll es folgende Auswahlmöglichkeiten geben:

- Auswahl eines Elements innerhalb einer Liste
- Auswahl zwischen Messwert, Wurzel und Quadrat

Die Auswahl des angezeigten Wertes wird wie folgt realisiert:

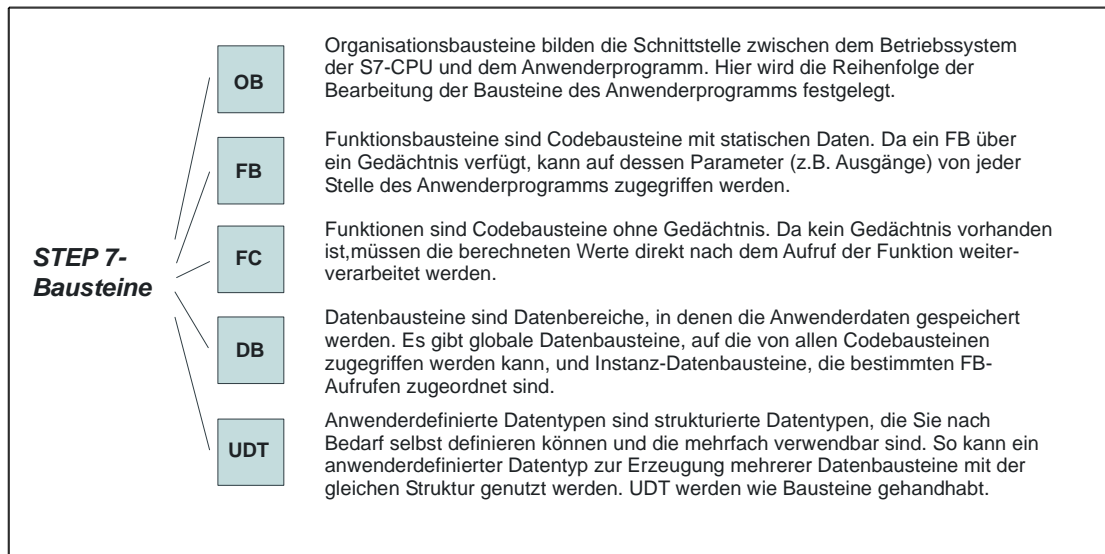
- Mit drei Schaltern wird eine Codierung eingestellt, die übernommen wird, wenn an einem vierten Schalter, dem Codierschalter, eine Flanke erkannt wird. Daraus wird die Adresse berechnet, mit der die Ausgabe adressiert wird.
- Mit der gleichen Adresse werden drei Werte, Messwert, Wurzel und Quadrat, für die Ausgabe bereitgestellt. Um daraus einen Wert auszuwählen, sind zwei Umschalter vorzusehen.



3.3 Aufbau eines strukturierten Programms mit S7-SCL

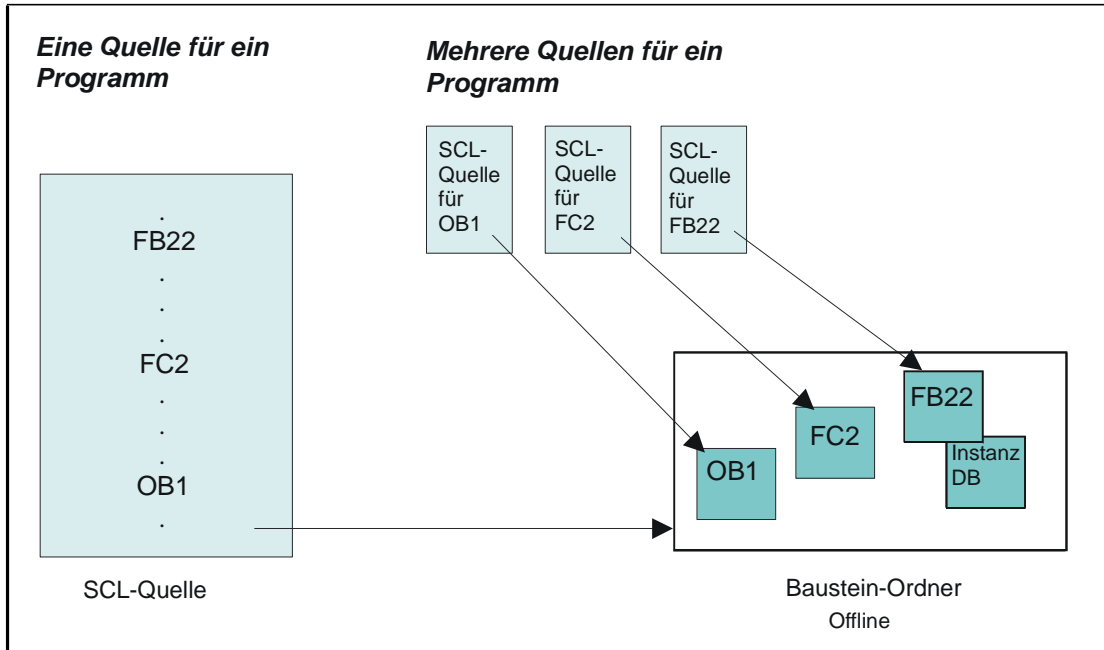
Bausteinarten

Die beschriebene Aufgabe lösen Sie am besten in Form eines **strukturierten S7-SCL-Programms**. Ein solches Programm ist modular aufgebaut, d.h. in Bausteine gegliedert, die jeweils eine bestimmte Teilaufgabe übernehmen. Bei S7-SCL stehen Ihnen wie bei den Programmiersprachen von STEP 7 folgende Bausteinarten zur Verfügung.



Anordnen der Bausteine in S7-SCL-Quellen

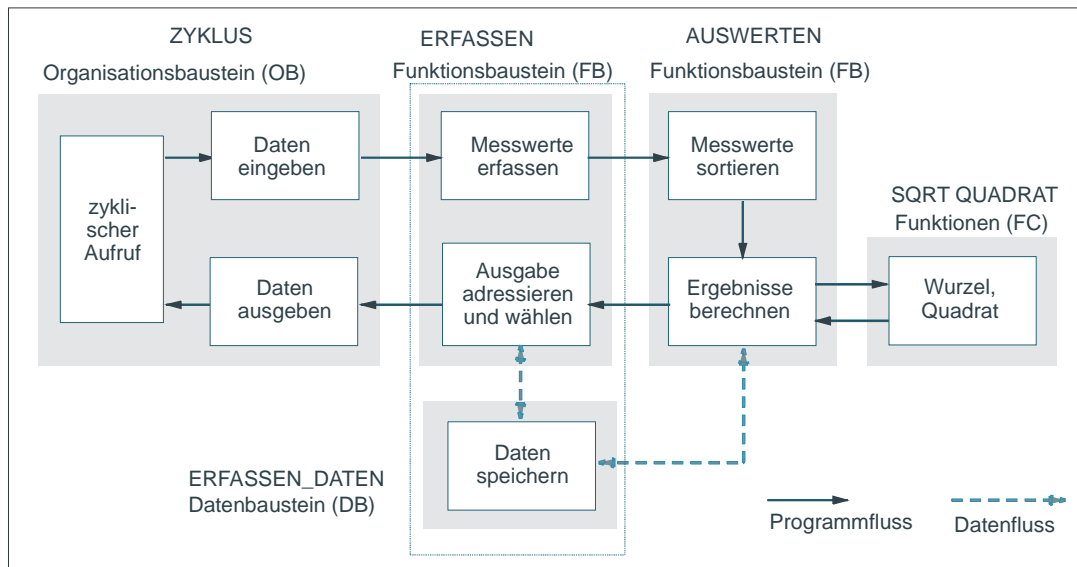
Ein S7-SCL-Programm besteht aus einer oder mehreren S7-SCL-Quellen. Eine Quelle kann einen einzelnen Baustein oder ein komplettes, aus verschiedenen Bausteinen bestehendes Programm enthalten.



3.4 Festlegen der Teilaufgaben

Teilaufgaben

Die Teilaufgaben sind im folgenden Bild als Kästen dargestellt. Die grau unterlegten, rechteckigen Bereiche repräsentieren die Bausteine. Die Anordnung der Codebausteine von links nach rechts entspricht der Aufruffreihenfolge.



Auswahl und Zuordnung der möglichen Bausteinarten

Die einzelnen Bausteine wurden nach folgenden Kriterien ausgewählt:

Funktion		Bausteinname
Anwenderprogramme können nur in einem OB angestoßen werden. Da die Messwerte zyklisch erfasst werden sollen, ist ein OB für <i>zyklischen Aufruf</i> (OB1) erforderlich. Ein Teil der Bearbeitung - <i>Daten eingeben</i> und <i>Daten ausgeben</i> - wird dabei im OB programmiert.	⇒	OB "Zyklus"
Für die Teilaufgabe <i>Messwerte erfassen</i> ist ein Baustein mit Gedächtnis, also ein FB erforderlich, da bestimmte bausteinlokale Daten (z.B. der Ringpuffer) von einem Programmzyklus zum nächsten erhalten bleiben müssen. Der Ort zum <i>Daten speichern</i> (Gedächtnis) ist der Instanz-Datenbaustein <code>ERFASSEN_DATEN</code> . Derselbe FB kann auch die Teilaufgabe <i>Ausgabe adressieren und wählen</i> übernehmen, da die benötigten Daten hier zur Verfügung stehen.	⇒	FB "Erfassen"
Bei der Wahl der Bausteinart zur Lösung der Teilaufgaben <i>Messwerte sortieren</i> und <i>Ergebnisse berechnen</i> ist zu berücksichtigen, dass ein Ausgabepuffer angelegt werden muss, der zu jedem Messwert die Berechnungsergebnisse Wurzel und Quadrat enthält. Deshalb kommt als Baustein nur ein FB in Frage. Da der FB von einem übergeordneten FB aufgerufen wird, benötigt er keinen eigenen DB. Seine Instanzdaten können im Instanz-Datenbaustein des aufrufenden FB abgelegt werden.	⇒	FB "Auswerten"
Zur Lösung der Teilaufgabe <i>Wurzel oder Quadrat berechnen</i> eignet sich ein FC am besten, weil die Rückgabe des Ergebnisses als Funktionswert erfolgen kann. Außerdem werden zur Berechnung keine Daten benötigt, die länger als einen Zyklus der Programmbearbeitung erhalten bleiben müssen. Zur Wurzelberechnung kann die S7-SCL-Standardfunktion <code>SQRT</code> benutzt werden. Zur Quadratberechnung soll eine Funktion <code>QUADRAT</code> erstellt werden, die auch eine Grenzprüfung des Wertebereichs durchführt.	⇒ ⇒	FC "SQRT" (Wurzel) und FC "Quadrat"

3.5 Festlegen der Schnittstellen zwischen den Bausteinen

Übersicht

Die Schnittstelle eines Bausteins wird durch Parameter gebildet, auf die von anderen Bausteinen aus zugegriffen werden kann.

Die im Baustein deklarierten Parameter sind Platzhalter, deren Werte erst bei der konkreten Verwendung (Aufruf) des Bausteins festgelegt werden. Diese Platzhalter bezeichnet man als Formalparameter, die beim Aufruf des Bausteins zugewiesenen Werte als Aktualparameter. Wenn ein Baustein aufgerufen wird, werden ihm Eingabedaten als Aktualparameter übergeben. Nach der Rückkehr zum aufrufenden Baustein werden die Ausgabedaten zur Übernahme bereit gestellt. Eine Funktion (FC) kann ihr Ergebnis als Funktionswert übergeben.

Die Bausteinparameter lassen sich in folgende Kategorien einteilen:

Bausteinparameter	Bedeutung	Deklaration mit
Eingangsparameter	Eingangsparameter nehmen beim Aufruf des Bausteins die aktuellen Eingangswerte auf. Sie können nur gelesen werden.	VAR_INPUT
Ausgangsparameter	Ausgangsparameter übergeben die aktuellen Ausgangswerte an den aufrufenden Baustein. Sie können beschrieben aber auch gelesen werden.	VAR_OUTPUT
Durchgangsparameter	Durchgangsparameter übernehmen beim Aufruf den aktuellen Wert einer Variablen, verarbeiten diesen und legen anschließend die Ergebnisse wieder in der gleichen Variablen ab.	VAR_IN_OUT

OB Zyklus

Der OB ZYKLUS hat selbst keine Formalparameter. Er ruft den FB ERFASSEN auf und übergibt ihm den Messwert und die Steuerungsdaten für dessen Formalparameter.

FB Erfassen

Parametername	Datentyp	Deklarations- typ	Beschreibung
messwert_ein	INT	VAR_INPUT	Messwert
neuwert	BOOL	VAR_INPUT	Schalter, um Messwert in Ringpuffer zu übernehmen
neusort	BOOL	VAR_INPUT	Schalter, um Messwerte zu sortieren und auszuwerten
funktionswahl	BOOL	VAR_INPUT	Umschalter, um Wurzel oder Quadrat zu wählen
auswahl	WORD	VAR_INPUT	Codierung, um Ausgabewert auszuwählen
neuwahl	BOOL	VAR_INPUT	Schalter, um Codierung zu übernehmen
ergebnis_aus	DWORD	VAR_OUTPUT	Ausgabe des berechneten Ergebnisses
messwert_aus	DWORD	VAR_OUTPUT	Ausgabe des zugehörigen Messwerts

Auswerten

Der FB `ERFASSEN` ruft den FB `AUSWERTEN` auf. Als gemeinsame Daten haben die beiden FBs das zu sortierende Messwertefeld. Deshalb wird dieses als Durchgangparameter deklariert. Für die Rechenergebnisse Wurzel und Quadrat wird ein strukturiertes Feld als Ausgangsparameter angelegt. Folgende Tabelle zeigt die Formalparameter:

Name	Datentyp	Deklarations- typ	Beschreibung
sortierpuffer	ARRAY[..] OF REAL	VAR_IN_OUT	Messwertefeld, entspricht dem Ringpuffer
rechenpuffer	ARRAY[..] OF STRUCT	VAR_OUTPUT	Feld für Ergebnisse: Struktur mit den Komponenten "wurzel" und "quadrat" vom Typ INT

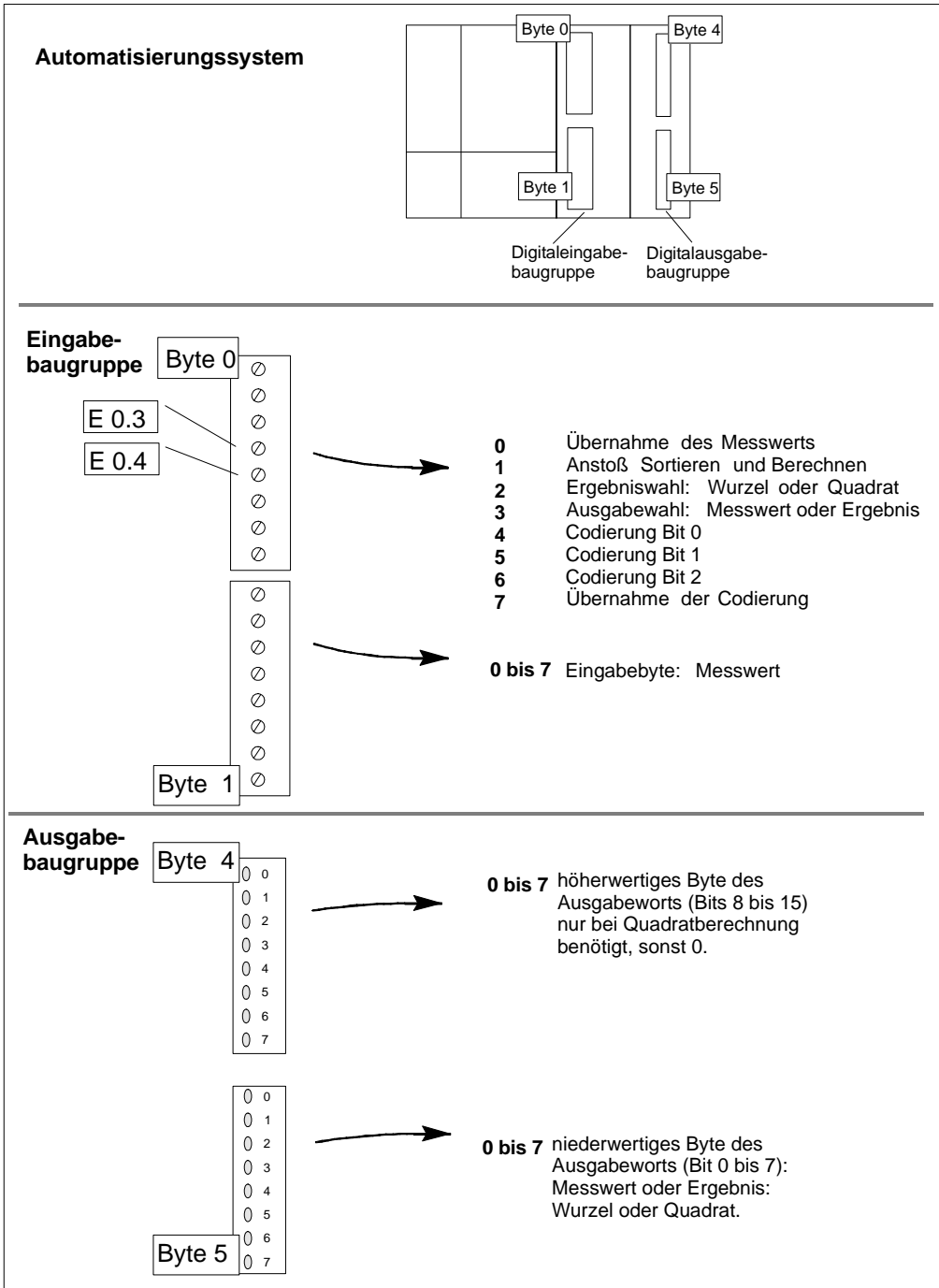
SQRT und Quadrat

Die Funktionen werden von `AUSWERTEN` aufgerufen. Sie benötigen einen Eingabewert und liefern ihr Ergebnis als Funktionswert.

Name	Datentyp	Deklarations- typ	Beschreibung
wert	REAL	VAR_INPUT	Eingabe für SQRT
SQRT	REAL	Funktionswert	Wurzel des Eingabewerts
wert	INT	VAR_INPUT	Eingabe für QUADRAT
QUADRAT	INT	Funktionswert	Quadrat des Eingabewerts

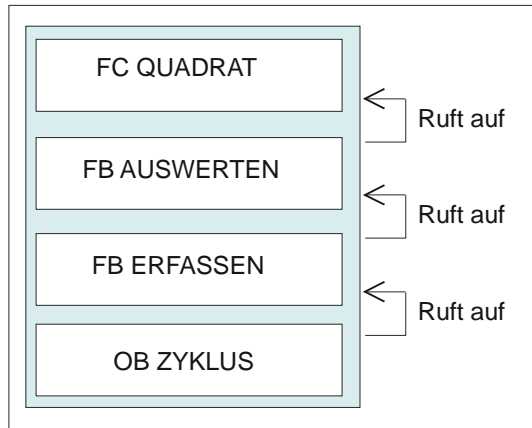
3.6 Festlegen der Ein-/Ausgabe Schnittstelle

Folgendes Bild zeigt die Ein-/Ausgabeschnittstelle. Beachten Sie bitte, dass bei der byteweisen Ein-/Ausgabe oben das niederwertige Byte und unten das höherwertige Byte ist. Bei der wortweisen Ein-/Ausgabe dagegen ist es umgekehrt.



3.7 Festlegen der Reihenfolge der Bausteine in der Quelle

Bezüglich der Reihenfolge der Bausteine in der S7-SCL-Quelle müssen Sie berücksichtigen, dass ein Baustein vorhanden sein muss, bevor Sie ihn benutzen, d.h. von einem anderen Baustein aus aufrufen. In der S7-SCL-Quelle müssen die Bausteine also wie folgt angeordnet sein:



3.8 Festlegen der Symbole

Die Verständlichkeit des Programms verbessert sich, wenn Sie symbolische Namen für Baugruppenadressen und Bausteine vergeben. Dazu müssen Sie Einträge in der Symboltabelle machen.

Im folgenden Bild sehen Sie die Symboltabelle des Beispiels. Sie beschreibt die symbolischen Namen, die Sie in der Symboltabelle vereinbaren müssen, damit die Quelle fehlerfrei übersetzt werden kann:

	Symbol	Adresse	Datentyp	▲
1	Ergebnisse	VAT 3		
2	Messwerte	VAT 2		
3	Überblick SCL Bsp	VAT 1		
4	ZYKLUS	OB 1	OB 1	
5	QUADRAT	FC 41	FC 41	
6	AUSWERTEN	FB 20	FB 20	
7	ERFASSEN	FB 10	FB 10	
8	Codierung	EW 0	WORD	
9	Eingabe	EB 1	BYTE	
10	Codierschalter	E 0.7	BOOL	
11	Ausgabeschalter	E 0.3	BOOL	
12	Funktionsschalter	E 0.2	BOOL	
13	Sortierschalter	E 0.1	BOOL	
14	Eingang 0.0	E 0.0	BOOL	
15	ERFASSEN_DATEN	DB 10	FB 10	
16	Ausgabe	AW 4	INT	

3.9 Erstellen der Funktion QUADRAT

3.9.1 Anweisungsteil der Funktion QUADRAT

Anweisungsteil

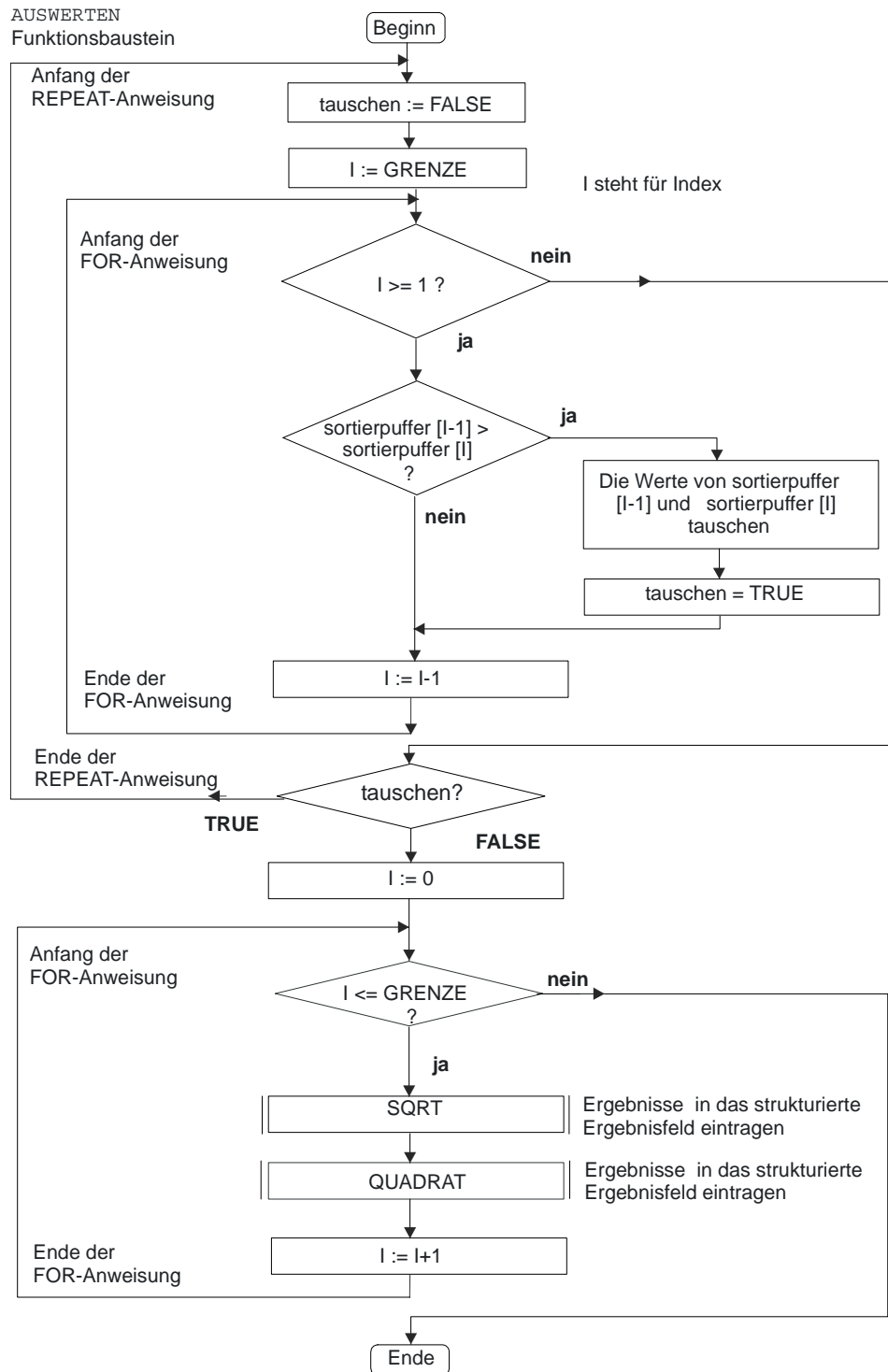
Zuerst wird geprüft, ob der Eingabewert die Grenze überschreitet, bei der das Ergebnis über den Zahlenbereich für Integer hinaus geht. Ist dies der Fall, wird der Maximalwert für Integer eingetragen. Ansonsten wird die Quadrierung durchgeführt. Das Ergebnis wird als Funktionswert übergeben.

```
FUNCTION QUADRAT : INT
(*****
Diese Funktion liefert als Funktionswert das Quadrat des
Eingangswertes der bei überlauf den maximalen Wert, der mit
Integer darstellbar ist.
*****)
VAR_INPUT
    wert : INT;
END_VAR
BEGIN
IF wert <= 181 THEN
    QUADRAT := wert * wert; //Berechnung des Funktionswerts
ELSE
    QUADRAT := 32_767; // bei Überlauf Maximalwert setzen
END_IF;
END FUNCTION
```


3.10 Erstellen des Funktionsbausteins AUSWERTEN

3.10.1 Flussdiagramm von AUSWERTEN

Das Bild stellt den Algorithmus in Form eines Flussdiagramms dar:



3.10.2 Vereinbarungsteil des FB AUSWERTEN

Aufbau des Vereinbarungsteils

Der Vereinbarungsteil dieses Bausteins besteht aus folgenden Teilen:

- Konstantendeklaration: Zwischen CONST und END_CONST
- Durchgangparameter: Zwischen VAR_IN_OUT und END_VAR,
- Ausgangsparameter: Zwischen VAR_OUTPUT und END_VAR
- Deklaration der temporären Variablen: Zwischen VAR_TEMP und END_VAR

```
CONST
    GRENZE := 7;
END_CONST

VAR_IN_OUT
    sortierpuffer : ARRAY[0..GRENZE] OF INT;
END_VAR

VAR_OUTPUT
    rechenpuffer : ARRAY[0..GRENZE] OF
        STRUCT
            wurzel : INT;
            quadrat : INT;
        END_STRUCT;
END_VAR

VAR_TEMP
    tauschen      : BOOL;
    index, hilf   : INT;
    wertr, ergebnr : REAL;
END_VAR
```

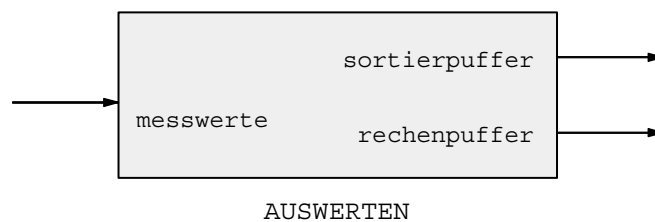
3.10.3 Anweisungsteil des FB AUSWERTEN

Programmablauf

Der Durchgangparameter "sortierpuffer" wird mit dem Ringpuffer "messwerte" verknüpft, d.h. der ursprüngliche Pufferinhalt wird mit den sortierten Messwerten überschrieben.

Für die Rechenergebnisse wird das neue Feld "rechenpuffer" als Ausgangsparameter angelegt. Seine Elemente sind so strukturiert, dass sie zu jedem Messwert die Wurzel und das Quadrat enthalten.

Im folgenden Bild sehen Sie den Zusammenhang zwischen den beschriebenen Feldern:



Diese Schnittstelle zeigt den Kern des Datenaustauschs zur Verarbeitung der Messwerte. Die Werte werden im Instanz-Datenbaustein `ERFASSEN_DATEN` gespeichert, da im aufrufenden FB `ERFASSEN` eine lokale Instanz für den FB `AUSWERTEN` angelegt wurde.

Anweisungsteil von AUSWERTEN

Zuerst werden die Messwerte im Ringpuffer sortiert und danach die Berechnungen durchgeführt:

- **Methode des Algorithmus zum Sortieren**
Hier wird die Methode des permanenten Tauschens von Werten zur Sortierung des Messwertepuffers verwendet, d.h. zwei aufeinanderfolgende Werte werden miteinander verglichen und solange getauscht, bis die gewünschte Sortierreihenfolge erreicht ist. Der verwendete Puffer ist der Durchgangparameter "sortierpuffer".
- **Anstoß der Berechnung**
Wenn die Sortierung abgeschlossen ist, wird eine Schleife zur Berechnung durchlaufen, in der die Funktionen `QUADRAT` zur Quadrierung und `SQRT` zur Wurzelberechnung aufgerufen werden. Ihre Ergebnisse werden in dem strukturierten Feld "rechenpuffer" gespeichert.

Anweisungsteil von AUSWERTEN

Der Anweisungsteil des Codebausteins sieht wie folgt aus:

```

BEGIN
(*****
Teil 1 Sortierung : nach dem "Bubble Sort" Verfahren: Werte
solange paarweise tauschen, bis Messwertpuffer sortiert ist.
*****)
REPEAT
    tauschen := FALSE;
    FOR index := GRENZE TO 1 BY -1 DO
        IF sortierpuffer[index-1] > sortierpuffer[index]
            THEN hilf
:=sortierpuffer[index];
            sortierpuffer[index] :=
sortierpuffer[index-1];
            sortierpuffer[index-1] := hilf;
            tauschen := TRUE;
        END_IF;
    END_FOR;
    UNTIL NOT tauschen
END_REPEAT;
(*****
Teil 2 Berechnung : Wurzelberechnung mit Standardfunktion
SQRT und Quadrierung mit Funktion QUADRAT durchführen.
*****)
FOR index := 0 TO GRENZE BY 1 DO
    wertr := INT_TO_REAL(sortierpuffer[index]);
    ergebnr := SQRT(wertr);
    rechenpuffer[index].wurzel := REAL_TO_INT(ergebnr);
    rechenpuffer[index].quadrat :=
QUADRAT(sortierpuffer[index]);
END_FOR;
END FUNCTION BLOCK

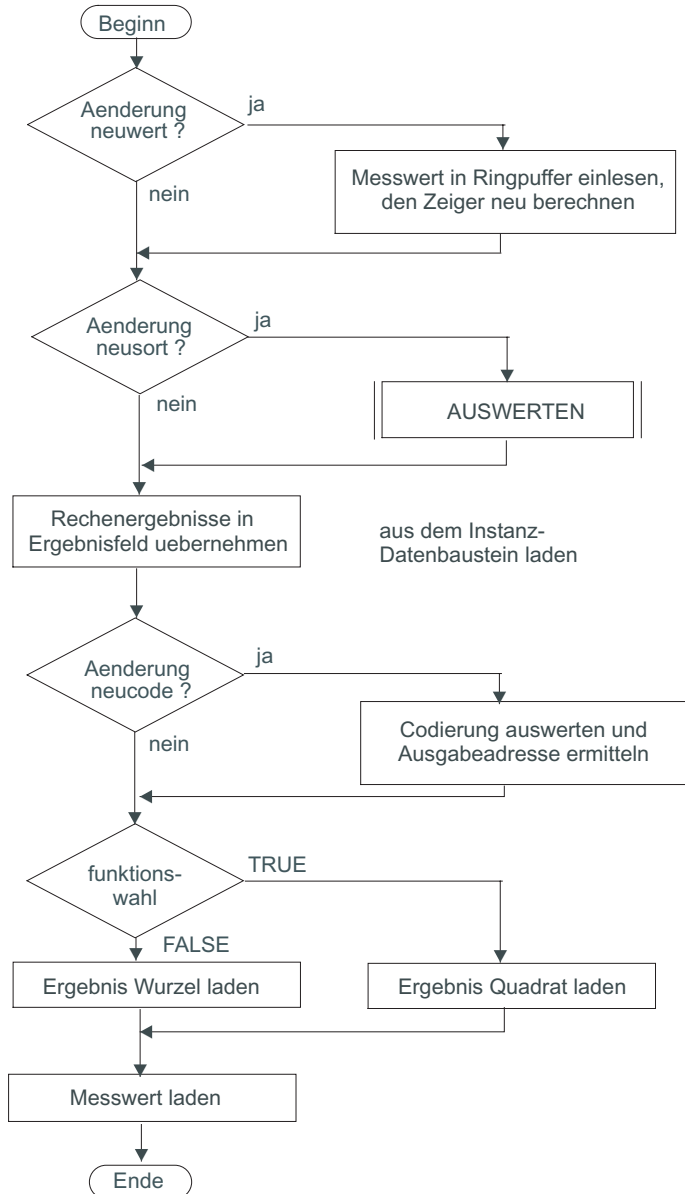
```

3.11 Erstellen des Funktionsbausteins ERFASSEN

3.11.1 Flussdiagramm von Erfassen

Folgendes Bild stellt den Algorithmus in Form eines Flussdiagramms dar:

ERFASSEN
Funktionsbaustein



Mit der MOD-Operation wird der Ringpuffer realisiert: Wenn Grenze erreicht, Beginn von vorne

Ringpuffer sortieren und Berechnungen durchfuehren (dafuer Ergebnisfeld anlegen)

aus dem Instanz-Datenbaustein laden

Zuerst die relevanten Bits an den rechten Rand schieben, dann die nicht benoetigten Stellen mit AND ausblenden

Laden:
Listenelemente mit Ausgabeadresse in die Ausgangsparameter schreiben, um danach ihre Werte anzuzeigen

3.11.2 Vereinbarungsteil des FB ERFASSEN

Aufbau des Vereinbarungsteils

Der Vereinbarungsteil in diesem Baustein besteht aus folgenden Vereinbarungsblocken:

- Konstantendeklaration: Zwischen `CONST` und `END_CONST`.
- Eingangsparameter: Zwischen `VAR_INPUT` und `END_VAR`.
- Ausgangsparameter: Zwischen `VAR_OUTPUT` und `END_VAR`.
- statischen Variablen: Zwischen `VAR` und `END_VAR`. Hierzu zählt auch die Deklaration der lokalen Instanz für den Baustein `AUSWERTEN`.

```

CONST
  GRENZE := 7;
  ANZAHL := GRENZE + 1;
END_CONST
VAR_INPUT
  messwert_ein : INT ; // Neuer Messwert
  neuwert      : BOOL; // Messwert in Umlaufpuffer
"messwerte"
  neusort      : BOOL; // Messwerte sortieren
  funktionswahl: BOOL; // Wahl der Berechnungsfunktion
Wurzel/Quadrat
  neuwahl     : BOOL; // Ausgabeadresse übernehmen
  auswahl     : WORD; // Ausgabeadresse
END_VAR
VAR_OUTPUT
  ergebnis_aus : INT; // berechneter Wert
  messwert_aus  : INT; // zugehöriger Messwert
END_VAR
VAR
  messwerte          : ARRAY[0..GRENZE] OF INT := 8(0);
  ergebnispuffer   : ARRAY[0..GRENZE] OF
  STRUCT
    wurzel          : INT;
    quadrat         : INT;
  END_STRUCT;
  zeiger            : INT := 0;
  altwert          : BOOL := TRUE;
  altsort          : BOOL := TRUE;
  altwahl          : BOOL := TRUE;
  adresse          : INT := 0;
//Konvertierte Ausgabeadresse
  auswerte_instanz: AUSWERTEN; //Lokale Instanz
definieren
END VAR

```

Statische Variablen

Der Bausteintyp FB wurde gewählt, weil es Daten gibt, die von einem Programmzyklus zum nächsten gespeichert werden müssen. Diese sind die statischen Variablen, die im Vereinbarungsblock "VAR, END_VAR" deklariert werden.

Statische Variablen sind lokale Variablen, deren Werte über alle Bausteindurchläufe hinweg erhalten bleiben. Sie dienen der Speicherung von Werten eines Funktionsbausteins und werden im Instanz-Datenbaustein abgelegt.

Initialisierung von Variablen

Beachten Sie die Vorbesetzungswerte, die bei der Initialisierung des Bausteins (nach dem Laden in die CPU) in die Variablen eingetragen werden. Im Vereinbarungsblock "VAR, END_VAR" wird auch die lokale Instanz für den FB AUSWERTEN deklariert. Der Name wird später zum Aufruf und Zugriff auf die Ausgangsparameter verwendet. Als Datenspeicher wird die globale Instanz ERFASSEN_DATEN benutzt.

Name	Datentyp	Vorbesetzung	Beschreibung
messwerte	ARRAY [..] OF INT	8(0)	Ringpuffer für Messwerte
ergebnispuffer	ARRAY [..] OF STRUCT	-	Feld für Strukturen mit den Komponenten "wurzel" und "quadrat" vom Typ INT
zeiger	INT	0	Index für Ringpuffer, dort Eintrag des nächsten Messwerts
altwert	BOOL	FALSE	Vorgängerwert für Messwertübernahme mit "neuwert"
altsort	BOOL	FALSE	Vorgängerwert für Sortieren mit "neusort"
altwahl	BOOL	FALSE	Vorgängerwert für Übernahme der Codierung mit "neuwahl"
adresse	INT	0	Adresse für Messwert- oder Ergebnisausgabe
auswerte_instanz	lokale Instanz	-	Lokale Instanz für den FB AUSWERTEN

3.11.3 Anweisungsteil des FB ERFASSEN

Aufbau des Anweisungsteils

Der Anweisungsteil von ERFASSEN gliedert sich in 3 Teile:

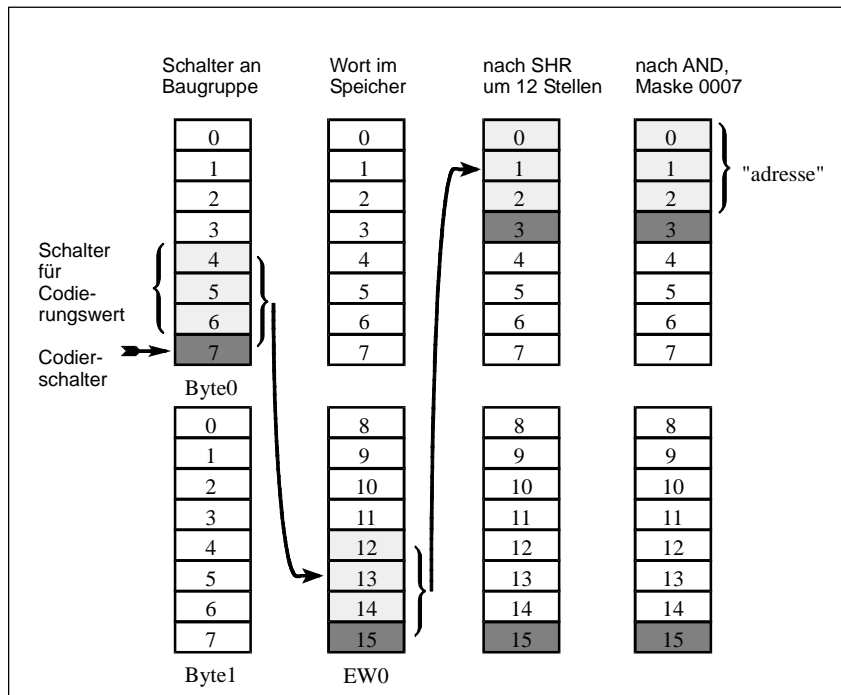
- Messwerte erfassen:
Wenn sich der Eingangsparameter "neuwert" gegenüber "altwert" verändert hat, wird ein neuer Messwert in den Ringpuffer eingelesen.
- Sortierung und Berechnung anstoßen
Sortierung und Berechnung werden durch Aufruf des Funktionsbausteins AUSWERTEN angestoßen, wenn sich der Eingangsparameter "neusort" gegenüber "altsort" verändert hat.
- Codierung auswerten und Ausgabe vorbereiten
Die Codierung wird wortweise eingelesen: Nach der SIMATIC-Konvention bedeutet dies, dass die obere Schaltergruppe (Byte0) die höherwertigen 8 Bit des Eingabeworts enthält und die untere Schaltergruppe (Byte1) die niederwertigen. Unten stehendes Bild zeigt, wo die Schalter liegen, an denen Sie die Codierung einstellen:

Berechnung der Adresse

Folgendes Bild zeigt die Berechnung der Adresse: Das Eingabewort EW0 enthält im 12. bis 14. Bit die Codierung, die übernommen wird, wenn am Codierschalter (15. Bit) eine Flanke erkannt wird. Durch Verschieben nach rechts mit der Standardfunktion SHR und Ausblenden der relevanten Bits mit einer AND-Maske wird "adresse" ermittelt.

Mit dieser Adresse werden die Feldelemente (Rechenergebnis und zugehöriger Messwert) in die Ausgangsparameter geschrieben. Ob Wurzel oder Quadrat ausgegeben wird, hängt von "funktionswahl" ab.

Eine Flanke am Codierschalter wird dadurch erkannt, dass sich "neuwahl" gegenüber "altwahl" verändert hat.



Anweisungsteil

Der Anweisungsteil des Codebausteins sieht wie folgt aus:

```

BEGIN
(*****
Teil 1 : Erfassung der Messwerte. Bei Änderung von "neuwert"
erfolgt Eingabe des Messwerts. Mit Operation MOD wird ein
Umlaufpuffer für Messwerte realisiert.
*****)
IF neuwert <> altwert THEN
    zeiger                := zeiger MOD ANZAHL;
    messwerte[zeiger] := messwert_ein;
    zeiger                := zeiger + 1;
END_IF;
altwert := neuwert;
(*****
Teil 2 : Sortierung und Berechnung anstoßen
Bei Änderung von "neusort" Anstoß zum Sortieren des
Umlaufpuffers und zur Ausführung von Berechnungen mit den
Messwerten. Ergebnisse werden in einem neuen Feld,
"rechenpuffer", gespeichert.
*****
*)
IF neusort <> altsort THEN
    zeiger := 0;                //Umlaufpufferzeiger rücksetzen
    auswerte_instanz(sortierpuffer := messwerte); //AUSWERTEN
    aufrufen
END_IF;
altsort := neusort;
ergebnispuffer := auswerte_instanz.rechenpuffer; //Quadrat
und Wurzel

(*****
*
Teil 3 : Codierung auswerten und Ausgabe vorbereiten: Bei
Änderung von "neuwahl" wird die Codierung für die
Adressierung
des Feldelements für die Ausgabe neu ermittelt: Die
relevanten
Bits von "auswahl" werden ausgeblendet und in Integer
gewandelt.
Je nach Schalterstellung von "funktionswahl" wird "wurzel"
oder
"quadrat" für die Ausgabe bereitgestellt.
*****
*****)
IF neuwahl <> altwahl THEN
    adresse := WORD_TO_INT(SHR(IN := auswahl, N := 12) AND
16#0007);
END_IF;
altwahl := neuwahl;

```

```
IF funktionswahl THEN
    ergebnis_aus := ergebnispuffer[adresse].quadrat;
ELSE
    ergebnis_aus := ergebnispuffer[adresse].wurzel;
END_IF;
messwert_aus := messwerte[adresse]; //Messwertanzeige
END FUNCTION BLOCK
```

3.12 Erstellen des Organisationsbausteins ZYKLUS

Aufgaben des OB ZYKLUS

Ein OB1 wurde gewählt, weil er zyklisch aufgerufen wird. Mit ihm werden folgende Aufgaben für das Programm realisiert:

- Aufruf und Versorgung des Funktionsbausteins ERFASSEN mit Eingabedaten und Steuerungsdaten
- Übernahme der Ergebnisdaten des Funktionsbausteins ERFASSEN
- Ausgabe der Werte zur Anzeige

Am Anfang des Vereinbarungsteils steht das temporäre Datenfeld mit 20 Byte "systemdaten".

Programmcode des OB ZYKLUS

```

ORGANIZATION_BLOCK ZYKLUS
(*****
ZYKLUS entspricht einem OB1, d.h. er wird vom S7-System
zyklisch aufgerufen.
Teil 1 : Aufruf des Funktionsbausteins und Übergabe der
Eingabewerte Teil 2 : Übernahme der Ausgabenwerte und
Ausgabe
mit Ausgabeumschaltung
*****)
VAR_TEMP
    systemdaten : ARRAY[0..20] OF BYTE; // Bereich für OB1
END_VAR
BEGIN
(* Teil 1 :
*****)
ERFASSEN.ERFASSEN_DATEN(
    messwert_ein:= WORD_TO_INT(Eingabe),
    neuwert      := "Eingang 0.0", //Eingabeschalter als
Signalkennzeichen
    neusort      := Sortierschalter,
    funktionswahl:= Funktionsschalter,
    neuwahl      := Codierschalter,
    auswahl      := Codierung);

(* Teil 2 :
*****)
IF Ausgabeschalter THEN
    //Ausgabeumschaltung
    Ausgabe      := ERFASSEN_DATEN.ergebnis_aus; //Wurzel
oder Quadrat
ELSE
    Ausgabe      := ERFASSEN_DATEN.messwert_aus; //Messwert
END_IF;
END ORGANIZATION_BLOCK
    
```

Datentyp-Konvertierung

Der Messwert liegt in der Eingabe als Typ BYTE vor. Er muss nach INT konvertiert werden: Dazu müssen Sie ihn von WORD nach INT konvertieren (die vorherige Konvertierung von BYTE nach WORD erfolgt implizit durch den Compiler). Keine Konvertierung wird dagegen für die Ausgabe benötigt, da diese in der Symboltabelle als INT deklariert wurde.

3.13 Testdaten

Voraussetzungen

Für den Test benötigen Sie eine Eingabebaugruppe mit der Adresse 0 und eine Ausgabebaugruppe mit der Adresse 4.

Stellen Sie vor dem Test die oberen 8 Schalter der Eingabebaugruppe nach links ("0") und die unteren 8 Schalter der Eingabebaugruppe nach rechts ("1").

Laden Sie die Bausteine neu in die CPU, da auch die Initialwerte der Variablen getestet werden.

Testschritte

Führen Sie nun die Testschritte nach der Tabelle durch.

Test	Aktion	Folge
1	Stellen Sie die Codierung "111" ein (E0.4, E0.5 und E0.6) und übernehmen Sie diese mit dem Codierschalter (E0.7).	Alle Ausgänge der Ausgabe (niederwertiges Byte) werden angesteuert und die Anzeigen leuchten.
2	Zeigen Sie die zugehörige Wurzel an, indem Sie den Ausgabeschalter (E0.3) auf "1"schalten.	Die Anzeigen am Ausgang entsprechen dem Binärwert "10000" (=16).
3	Zeigen Sie das zugehörige Quadrat an, indem Sie den Funktionsschalter (E0.2) auf "1"schalten.	Am Ausgang leuchten 15 Anzeigen. Das bedeutet, dass Überlauf auftritt, da 255 x 255 einen zu großen Wert für den Integerbereich ergibt.
4a	Stellen Sie den Ausgabeschalter (E0.3) wieder auf "0" zurück.	Der Messwert wird wieder angezeigt: Alle Anzeigen an den Ausgängen des niederwertigen Ausgabebytes sind gesetzt.
4b	Stellen Sie als neuen Messwert an der Eingabe den Wert 3, d.h. Binärwert "11", ein.	Die Ausgabe verändert sich noch nicht.
5a	Einlesen des Messwerts beobachten: Stellen Sie die Codierung auf "000" ein und übernehmen Sie diese mit dem Codierschalter (E0.7), sodass Sie später die Werteingabe beobachten können.	Am Ausgang wird 0 angezeigt, d.h. keine der Anzeigen leuchtet.
5b	Schalten Sie den Eingabeschalter "Eingang 0.0" (E0.0) um. Dadurch wird der im 4. Testschritt eingestellte Wert eingelesen.	Am Ausgang wird der Messwert 3, Binärwert "11", angezeigt.
6	Stoßen Sie Sortieren und Berechnen an, indem Sie den Sortierschalter (E0.1) umschalten.	Am Ausgang erscheint wieder 0, da durch den Sortiervorgang der Messwert weiter nach oben im Feld verschoben wurde.
7	Messwert nach der Sortierung anzeigen: Stellen Sie die Codierung "110" ein (E0.6 = 1, E0.5 = 1, E0.4 = 0 von EBO, entspricht Bit 14, Bit 13, Bit12 von EW0) und übernehmen Sie diese durch Umschalten des Codierschalters.	Am Ausgang wird nun wieder der Messwert "11" angezeigt, da er der zweithöchste Wert im Feld ist.

Test	Aktion	Folge
8a	Die zugehörigen Ergebnisse anzeigen: Durch Umschalten des Ausgabeschalters (E0.3) wird das Quadrat zum Messwert aus dem 7. Testschritt angezeigt.	Der Ausgangswert 9 bzw. Binärwert "1001" wird angezeigt.
8b	Durch Umschalten des Funktionsschalters (E0.2) erhalten Sie auch die Wurzel.	Der Ausgangswert 2 bzw. Binärwert "10" wird angezeigt.

Zusatztest

Folgende Tabellen erläutern die Schalter an der Eingabebaugruppe sowie Testmuster für Wurzel und Quadrat und helfen Ihnen, eigene Testschritte zu definieren:

- Die Eingabe erfolgt über Schalter: Über die oberen 8 Schalter können Sie das Programm steuern, über die unteren 8 können Sie Messwert einstellen.
- Die Ausgabe erfolgt über Anzeigen: An der oberen Gruppe erscheint das höherwertige Ausgabebyte, an der unteren Gruppe das niederwertige.

Bedienschalter	Name	Erklärung
Kanal 0	Eingabeschalter	Umschalten zur Messwertübernahme
Kanal 1	Sortierschalter	Umschalten zur Sortierung/Auswertung
Kanal 2	Funktionsschalter	Schalter nach links ("0"): Wurzel, Schalter nach rechts ("1"): Quadrat
Kanal 3	Ausgabeschalter	Schalter nach links ("0"): Messwert, Schalter nach rechts ("1"): Ergebnis
Kanal 4	Codierung	Ausgabeadresse Bit 0
Kanal 5	Codierung	Ausgabeadresse Bit 1
Kanal 6	Codierung	Ausgabeadresse Bit 2
Kanal 7	Codierschalter	Umschalten zur Codierungsübernahme

Folgende Tabelle enthält 8 beispielhafte Messwerte in bereits sortierter Reihenfolge.

Geben Sie die Werte in beliebiger Reihenfolge ein. Stellen Sie dazu die jeweilige Bitkombination ein und übernehmen Sie den Wert, indem Sie den Eingabeschalter umschalten. Nachdem alle Werte eingegeben sind, stoßen Sie durch Umschalten des Sortierschalters die Sortierung und Auswertung an. Danach können Sie die sortierten Messwerte oder die Ergebnisse - Wurzel oder Quadrat - anschauen.

Messwert	Wurzel	Quadrat
0000 0001 = 1	0, 0000 0001 = 1	0000 0000, 0000 0001 = 1
0000 0011 = 3	0, 0000 0010 = 2	0000 0000, 0000 1001 = 9
0000 0111 = 7	0, 0000 0011 = 3	0000 0000, 0011 0001 = 49
0000 1111 = 15	0, 0000 0100 = 4	0000 0000, 1110 0001 = 225
0001 1111 = 31	0, 0000 0110 = 6	0000 0011, 1100 0001 = 961
0011 1111 = 63	0, 0000 1000 = 8	0000 1111, 1000 0001 = 3969
0111 1111 = 127	0, 0000 1011 = 11	0011 1111, 0000 0001 = 16129
1111 1111 = 255	0, 0001 0000 = 16	0111 111, 1111 1111 = Überlaufanzeige!

4 Bedienen von S7-SCL

4.1 Starten der S7-SCL-Software

Starten von der Windows-Oberfläche

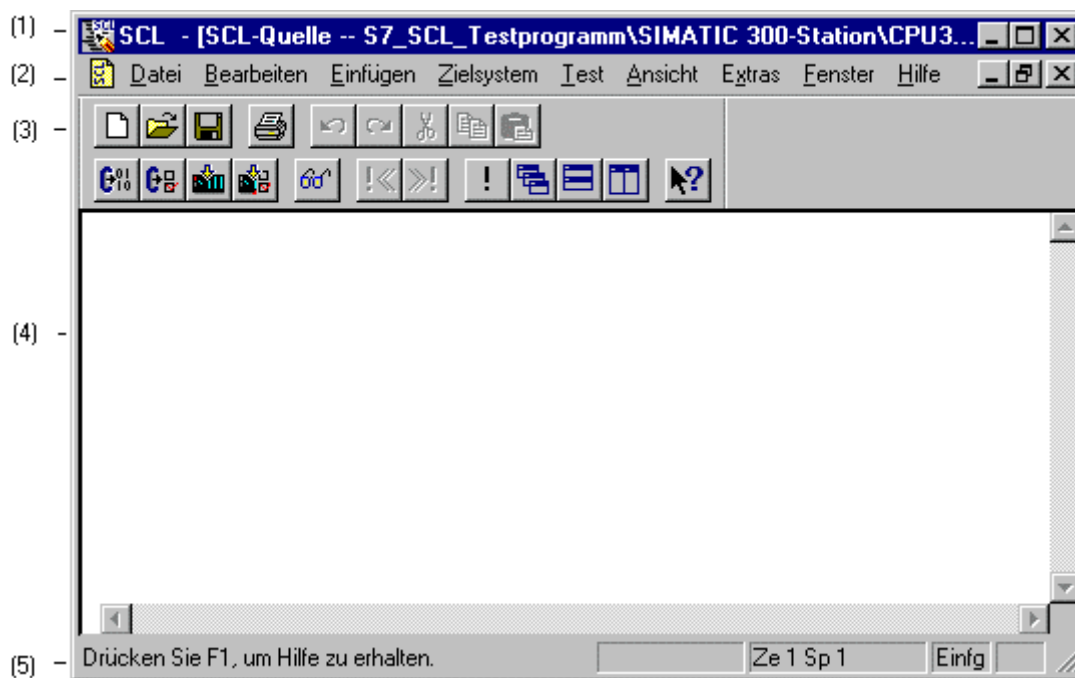
Nachdem Sie die S7-SCL-Software auf Ihrem PG installiert haben, können Sie S7-SCL über die Schaltfläche "Start" auf der Windows-Task-Leiste starten (Eintrag unter "SIMATIC/STEP 7").

Starten aus dem SIMATIC Manager

Am schnellsten starten Sie S7-SCL, indem Sie im SIMATIC Manager den Mauszeiger auf einer S7-SCL-Quelle positionieren und doppelklicken.

4.2 Bedienoberfläche

Die S7-SCL-Fenster bestehen aus den folgenden Standardkomponenten:



1. **Titelzeile:**
enthält den Fenstertitel und Symbole für die Fenstersteuerung.
2. **Menüleiste:**
enthält alle Menüs, die im Fenster zur Verfügung stehen.
3. **Funktionsleiste:**
enthält Symbole, über die Sie häufig verwendete Befehle schnell ausführen können.
4. **Arbeitsbereich:**
enthält die verschiedenen Fenster, in denen Sie den Programmtext editieren oder Compiler- und Testinformationen ablesen können.
5. **Statuszeile:**
zeigt den Status und weitere Informationen zum ausgewählten Objekt an.

4.3 Anpassen der Bedienoberfläche

Anpassen des Editors

Um Einstellungen für den Editor vorzunehmen, wählen Sie den Menübefehl **Extras > Einstellungen** und klicken im Dialogfeld "Einstellungen" das Register "Editor" an. Dort können Sie folgende Einstellungen machen:

Optionen im Register "Editor"	Bedeutung
Schrift	Legt die Schriftart für den gesamten Quelltext fest.
Tabulatorweite	Legt die Spaltenbreite von Tabulatoren fest.
Zeilennummern anzeigen	Zeigt Zeilennummern am beginn der Zeilen an.
Vor Übersetzen speichern	Fragt vor dem Übersetzungsvorgang , ob Sie die Quelle zuvor Speichern möchten.
Vor Sichern rückfragen	Fragt vor dem Speichern nach einer Bestätigung

Anpassen von Schriftart und -farbe

Um Schriftart und -farbe der verschiedenen Sprachelemente zu ändern, wählen Sie den Menübefehl **Extras > Einstellungen** und klicken im Dialogfeld "Einstellungen" das Register "Format" an. Dort können Sie folgende Einstellungen machen:

Optionen im Register "Format"	Bedeutung
Schlüsselwörter in Großbuchstaben	Formatiert S7-SCL-Schlüsselwörter wie FOR, WHILE, FUNCTION_BLOCK, VAR oder END_VAR bei der Eingabe in Großbuchstaben.
Einrücken bei Schlüsselwörtern	Rückt Zeilen in den einzelnen Deklarationsabschnitten und innerhalb der Kontrollanweisungen IF, CASE, FOR, WHILE und REPEAT bei der Eingabe ein.
Automatisch einrücken	Rückt nach einem Zeilenumbruch die neue Zeile automatisch um denselben Wert ein wie die vorhergehende Zeile. Die Einstellung gilt nur für neu eingegebene Zeilen.
Schriftstil/Farbe	Bestimmt Schriftstil und -farbe der einzelnen Sprachelemente

Die Einstellungen in diesem Register sind nur wirksam, wenn Sie im Register "Format" die Option "Folgende Formate verwenden" aktiviert haben und die gewünschten Einstellungen vorgenommen haben.

Funktionsleiste, Haltepunkteleiste, Statuszeile

Die Anzeige von Funktionsleiste, Haltepunkteleiste und Statuszeile können Sie getrennt ein- oder ausschalten. Aktivieren bzw. deaktivieren Sie dazu den entsprechenden Befehl im Menü **Ansicht**. Ein Häkchen hinter dem Menübefehl zeigt an, ob der Befehl gerade aktiviert ist oder nicht.

Fenster "Ausgaben"

Das Fenster "Ausgaben" zeigt Fehler und Warnungen, die beim Übersetzen einer Quelle auftreten. Sie können es mit dem Menübefehl **Ansicht > Ausgaben** ein- und ausschalten.

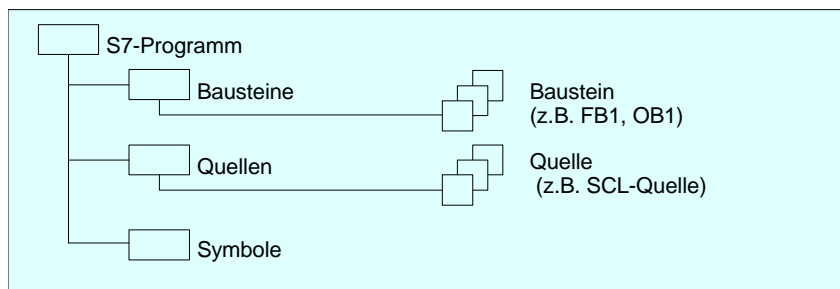
4.4 Anlegen und Hantieren einer S7-SCL-Quelle

4.4.1 Anlegen einer neuen S7-SCL-Quelle

Bevor Sie ein neues S7-SCL-Programm schreiben können, müssen Sie zunächst eine neue S7-SCL-Quelle anlegen. Diese Quelle legen Sie im Ordner "Quellen" unter einem S7-Programm an.

Struktur eines S7-Programms im SIMATIC Manager

Quellen, die Sie mit S7-SCL anlegen, lassen sich folgendermaßen in die Struktur eines S7-Programms einordnen:



Gehen Sie folgendermaßen vor:

1. Öffnen Sie das Dialogfeld "Neu", indem Sie
 - das Symbol "Neu" in der Funktionsleiste anklicken oder
 - den Menübefehl **Datei > Neu** wählen.
2. Wählen Sie im Dialogfeld "Neu"
 - ein Projekt,
 - die Filtereinstellung "SCL-Quelle" und
 - den Ordner "Quellen" innerhalb des S7-Programms aus.
3. Tragen Sie den Namen des Quellobjekts in das entsprechende Textfeld ein. Der Name kann bis zu 24 Zeichen lang sein.
4. Bestätigen Sie mit "OK".

Das Quellobjekt wird unter dem von Ihnen vergebenen Namen angelegt und in einem Arbeitsfenster zur weiteren Bearbeitung angezeigt.

Hinweis

Sie können auch mit dem SIMATIC Manager eine S7-SCL-Quelle erstellen, indem Sie einen Quell-Ordner auswählen und den Menübefehl **Einfügen > S7-Software > S7-SCL-Quelle** wählen.

4.4.2 Öffnen einer S7-SCL-Quelle

Sie können eine S7-SCL-Quelle öffnen, um sie zu übersetzen oder zu editieren.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie das Dialogfeld "Öffnen", indem Sie
 - das Symbol "Öffnen" anklicken oder
 - den Menübefehl **Datei > Öffnen** wählen.
2. Wählen Sie im Dialogfeld:
 - das gewünschte Projekt,
 - das gewünschte S7-Programm und
 - den zugehörigen Quell-Ordner aus
3. Wählen Sie die S7-SCL-Quelle aus.
4. Klicken Sie die Schaltfläche "OK" an.

Hinweis

SCL-Quellen können Sie auch im SIMATIC Manager öffnen, indem Sie auf ihr Symbol doppelklicken oder den Menübefehl **Bearbeiten > Objekt öffnen** wählen, nachdem Sie das gewünschte Objekt markiert haben.

4.4.3 Schließen einer S7-SCL-Quelle

Gehen Sie folgendermaßen vor:

- Wählen Sie den Menübefehl **Datei > Schließen**, oder
- klicken das Symbol "Schließen" in der Titelzeile des Fensters an.

Hinweis

Wenn Sie die Quelle geändert haben, werden Sie gefragt, ob Sie die Änderungen vor dem Schließen speichern wollen. Wenn Sie die Änderungen nicht speichern, gehen sie verloren.

4.4.4 Öffnen von Bausteinen

Das Öffnen von Bausteinen ist mit der Applikation S7-SCL nicht möglich. Sie können immer nur die zugehörige Quelle öffnen. Mit S7-SCL erstellte Bausteine können aber mit dem KOP/AWL/FUP-Editor geöffnet und in der Programmiersprache AWL angezeigt und bearbeitet werden. Nehmen Sie in der AWL-Darstellung keine Änderungen am Baustein vor, da

- die angezeigten MC7-Befehle keinen gültigen AWL-Baustein darstellen müssen,
- eine fehlerfreie Übersetzung mit dem AWL-Compiler Änderungen voraussetzt, die tiefe Kenntnisse sowohl in AWL als auch in S7-SCL erfordern,
- der mit AWL übersetzte Baustein die Sprachkennung AWL und nicht mehr S7-SCL trägt,
- die S7-SCL-Quelle und der MC7-Code nicht mehr konsistent sind.

Weitere Informationen finden Sie in der Online-Hilfe zu STEP 7.

Hinweis

Eine einfachere Pflege Ihrer CPU-Programme erreichen Sie, indem Sie eventuelle Änderungen in den S7-SCL-Quellen vornehmen und diese erneut übersetzen.

4.4.5 Festlegen der Objekteigenschaften

Die Objekteigenschaften können Sie festlegen, indem Sie den Bausteinen Attribute zuweisen. Die Eigenschaften der S7-SCL-Quelle (z.B. Autor) können Sie im Dialogfeld "Eigenschaften" bestimmen.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Datei > Eigenschaften**.
2. Tragen Sie im Dialogfeld "Eigenschaften" die Optionen ein.
3. Bestätigen Sie mit "OK".

4.4.6 Erzeugen von S7-SCL-Quellen mit einem Standard-Editor

Sie können auch einen ASCII-Standardeditor zum Editieren Ihrer S7-SCL-Quelle verwenden. Dabei stehen Ihnen jedoch die hilfreichen Editierfunktionen und die integrierte Online-Hilfe von S7-SCL nicht zur Verfügung.

Nachdem Sie die Quelle erstellt und abgespeichert haben, müssen Sie diese mit dem SIMATIC Manager in den Ordner "Quellen" eines S7-Programms importieren (siehe STEP 7-Dokumentation). Anschließend können Sie die Quelle in S7-SCL öffnen, um sie dann weiterzubearbeiten oder zu übersetzen.

4.4.7 Bausteinschutz einrichten

Sie können für Bausteine einen Bausteinschutz einrichten, indem Sie das Attribut `KNOW_HOW_PROTECT` bei der Programmierung des Bausteins in der Quelle angeben.

Der Bausteinschutz hat folgende Konsequenzen:

- Wenn Sie später einen übersetzten Baustein mit dem inkrementellen AWL-Editor öffnen, kann der Anweisungsteil des Bausteins nicht eingesehen werden.
- Im Vereinbarungsteil des Bausteins werden nur die Variablen der Deklarationstypen `VAR_IN`, `VAR_OUT` und `VAR_IN_OUT` angezeigt. Die Variablen der Vereinbarungsböcke `VAR` und `VAR_TEMP` bleiben verborgen.

Bei der Eingabe des Bausteinschutzes gilt:

- Das Schlüsselwort lautet `KNOW_HOW_PROTECT`. Es wird vor allen anderen Bausteinattributen eingegeben.
- Auf diese Weise lassen sich `OB`, `FB`, `FC` und `DB` schützen.

4.5 Richtlinien für S7-SCL-Quellen

4.5.1 Allgemeine Regeln für S7-SCL-Quellen

S7-SCL-Quellen müssen den folgenden Regeln genügen:

- In einer S7-SCL-Quelle können beliebig viele Codebausteine (`FB`, `FC`, `OB`), Datenbausteine (`DB`) und anwenderdefinierte Datentypen (`UDT`) editiert werden.
- Jede Bausteinart ist typisch aufgebaut.
- Jede Anweisung und jede Variablendeklaration endet mit einem Semikolon (;).
- Groß- und Kleinschreibung wird nicht unterschieden.
- Kommentare dienen lediglich der Dokumentation des Programms. Sie beeinflussen den Programmablauf nicht.
- Instanz-Datenbausteine werden beim Aufruf eines Funktionsbausteins automatisch erzeugt. Sie müssen nicht editiert werden.
- Der `DB 0` ist vorbelegt. Sie können also keinen `DB` dieses Namens erzeugen.

4.5.2 Reihenfolge der Bausteine

Bezüglich der Reihenfolge der Bausteine müssen Sie bei der Erstellung der S7-SCL-Quelle Folgendes beachten:

- Aufgerufene Bausteine stehen vor den aufrufenden Bausteinen.
- Anwenderdefinierte Datentypen (UDT) stehen vor den Bausteinen, in denen sie verwendet werden.
- Datenbausteine mit zugeordnetem anwenderdefinierten Datentyp (UDT) stehen nach dem UDT.
- Datenbausteine stehen vor allen Bausteinen, die auf sie zugreifen.

4.5.3 Verwendung symbolischer Adressen

In einem S7-SCL-Programm arbeiten Sie mit Operanden wie E/A-Signalen, Merkern, Zählern, Zeiten und Bausteinen. Sie können diese Operanden in Ihrem Programm absolut adressieren (z.B. E1.1, M2.0, FB11), jedoch steigt die Lesbarkeit von S7-SCL-Quellen erheblich, wenn Sie dafür Symbole (z.B. Motor_EIN) benutzen. In Ihrem Anwenderprogramm lässt sich dann ein Operand über dieses Symbol ansprechen.

Lokale und globale Symbole

- Globale Symbole verwenden Sie für Speicherbereiche der CPU und Bausteinbezeichnungen. Sie sind im gesamten Anwendungsprogramm bekannt und müssen deshalb eindeutig bezeichnet werden. Die Symboltabelle können Sie mit STEP 7 anlegen.
- Lokale Symbole sind nur in dem Baustein bekannt, in dessen Vereinbarungsteil Sie sie definiert haben. Sie können Namen für Variablen, Parameter, Konstanten und Sprungmarken vergeben, wobei Sie in verschiedenen Bausteinen dieselben Namen für unterschiedliche Zwecke verwenden können.

Hinweis

Achten Sie darauf, dass die symbolischen Namen eindeutig und nicht mit Schlüsselwörtern identisch sind.

4.6 Editieren in S7-SCL-Quellen

4.6.1 Rückgängigmachen der letzten Editieraktion

Mit dem Menübefehl **Bearbeiten > Rückgängig** können Sie eine oder mehrere Aktionen rückgängig machen.

Nicht alle Operationen können rückgängig gemacht werden. So kann z. B. der Menübefehl **Datei > Speichern** nicht rückgängig gemacht werden.

4.6.2 Wiederherstellen einer Editieraktion

Nachdem Sie eine oder mehrere Aktionen rückgängig gemacht haben, können Sie diese mit dem Menübefehl **Bearbeiten > Wiederherstellen** wiederherstellen.

4.6.3 Suchen und Ersetzen von Textobjekten

Wenn Sie eine S7-SCL-Quelle überarbeiten oder ändern möchten, können Sie oft wertvolle Zeit sparen, indem Sie Textobjekte suchen und gegebenenfalls ersetzen. Ziele des Suchlaufs können z.B. Schlüsselwörter, absolute Bezeichner, symbolische Bezeichner etc. sein.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Bearbeiten > Suchen/Ersetzen...**
2. Tragen Sie im Dialogfeld "Suchen/Ersetzen" die Optionen ein.
3. Starten Sie den Suchlauf, indem Sie
 - die Schaltfläche "Suchen" anklicken, um ein Textobjekt zu suchen und es zu markieren oder
 - die Schaltfläche "Ersetzen" bzw. "Alles Ersetzen" anklicken, um den Text zu suchen und durch den im Textfeld "Ersetzen durch" angegebenen Text auszutauschen.

4.6.4 Markieren von Textobjekten

Textobjekte können Sie markieren, indem Sie die Maustaste gedrückt halten und den Mauszeiger über den gewünschten Bereich des Textes ziehen.

Außerdem können Sie:

- den gesamten Text einer Quelle markieren, indem Sie den Menübefehl **Bearbeiten > Alles Markieren** auswählen,
- ein Wort markieren, indem Sie es doppelt anklicken oder
- eine ganze Zeile markieren, indem Sie in den Rand links neben der Zeile klicken.

Mit dem Menübefehl **Bearbeiten > Markierung aufheben** können Sie die Markierung wieder entfernen.

4.6.5 Kopieren von Textobjekten

Mit dieser Funktion können Sie Programmabschnitte oder ganze Programme kopieren. Der kopierte Text wird in die Zwischenablage gespeichert, und lässt sich daraus beliebig oft an einer anderen Stelle einfügen.

Gehen Sie folgendermaßen vor:

1. Markieren Sie das Textobjekt, das kopiert werden soll.
2. Kopieren Sie das Objekt, indem Sie
 - das Symbol "Kopieren" in der Funktionsleiste anklicken oder
 - den Menübefehl **Bearbeiten > Kopieren** wählen.
3. Bewegen Sie die Einfügemarke an die Stelle (in derselben oder in einer anderen Applikation), an der das Objekt eingefügt werden soll.
4. Fügen Sie das Objekt ein, indem Sie
 - das Symbol "Einfügen" in der Funktionsleiste anklicken oder
 - den Menübefehl **Bearbeiten > Einfügen** wählen.

4.6.6 Ausschneiden von Textobjekten

Mit dieser Funktion verschieben Sie ausgewählten Text in die Zwischenablage. Üblicherweise wird dieser Menübefehl zusammen mit dem Menübefehl **Bearbeiten > Einfügen** verwendet, der den Inhalt der Zwischenablage an der aktuellen Position der Einfügemarke einfügt.

Gehen Sie folgendermaßen vor:

1. Markieren Sie das Objekt, das ausgeschnitten werden soll.
2. Schneiden Sie das Objekt aus, indem Sie
 - das Symbol "Ausschneiden" in der Funktionsleiste anklicken oder
 - den Menübefehl **Bearbeiten > Ausschneiden** wählen .

Hinweis

- Das ausgewählte Objekt kann nicht ausgeschnitten werden, wenn der Menübefehl **Bearbeiten > Ausschneiden** nicht aktiviert ist (grau unterlegt).
 - Mit dem Menübefehl **Bearbeiten > Einfügen** können Sie ausgeschnittenen Text an einer anderen Stelle (desselben oder eines anderen Programms) wieder einfügen.
 - Der Inhalt der Zwischenablage bleibt solange erhalten, bis Sie wieder einen der Menübefehle **Bearbeiten > Ausschneiden** oder **Bearbeiten > Kopieren** ausführen.
-

4.6.7 Löschen von Textobjekten

Sie können ein markiertes Textobjekt aus dem Quelltext entfernen.

Gehen Sie folgendermaßen vor:

1. Markieren Sie den zu löschenden Text.
2. Wählen Sie den Menübefehl **Bearbeiten > Löschen**.

Der gelöschte Text wird nicht in die Zwischenablage eingefügt. Das gelöschte Objekt kann mit dem Menübefehl **Bearbeiten > Rückgängig** oder **Bearbeiten > Wiederherstellen** wieder eingefügt werden.

4.6.8 Positionieren der Einfügemarke in einer bestimmten Zeile

Folgende Funktionen ermöglichen es Ihnen, die Einfügemarke gezielt zu positionieren.

Positionieren in einer bestimmten Zeilennummer

Sie können die Einfügemarke an den Anfang einer bestimmten Zeile positionieren:

1. Wählen Sie den Menübefehl **Bearbeiten > Gehe zu Zeile**.
Das Dialogfeld "Gehe zu" öffnet sich.
2. Tragen Sie in das Dialogfeld "Gehe zu" die Zeilennummer ein.
3. Bestätigen Sie mit "OK".

Positionieren auf dem nächsten / vorherigen Lesezeichen

Wenn Lesezeichen in der Quelle gesetzt wurden, können Sie zwischen diesen navigieren:

- Wählen Sie den Menübefehl **Bearbeiten > Gehe zu > Nächstes Lesezeichen / vorheriges Lesezeichen**.

Positionieren auf dem nächsten / vorherigen Fehler im Programmcode

Im Fenster "Ausgaben" werden nach dem Übersetzen alle syntaktischen Fehler mit Angabe der Zeilen- und Spaltennummer angezeigt.

S7-SCL bietet die Möglichkeit, zwischen den einzelnen Fehlerstellen im Programm zu navigieren, und so alle Fehler aus dem vorherigen Übersetzungslauf nacheinander zu beheben:

1. Positionieren Sie die Einfügemarke an einer beliebigen Stelle im Quelltext.
2. Wählen Sie den Menübefehl **Bearbeiten > Gehe zu > Vorheriger Fehler / Nächster Fehler**.

4.6.9 Syntaxgerechtes Einrücken von Zeilen

Folgende Funktionen ermöglichen es Ihnen, S7-SCL-Quellen durch Zeileneinrückung zu strukturieren:

- **Automatisch Einrücken**
Wenn die Funktion aktiv ist, wird nach einem Zeilenumbruch die neue Zeile automatisch um denselben Wert eingerückt wie die vorhergehende Zeile.
- **Einrücken bei Schlüsselwörtern**
Wenn die Funktion aktiv ist, werden Zeilen in den einzelnen Deklarationsabschnitten und innerhalb der Kontrollstrukturen IF, CASE, FOR, WHILE und REPEAT eingerückt.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Extras > Einstellungen**.
2. Wählen Sie im nachfolgenden Dialogfeld das Register "Format".
3. Stellen Sie sicher, dass die Option "Folgende Formate verwenden" aktiv ist.
4. Aktivieren Sie die Option "Automatisch einrücken" oder "Einrücken nach Schlüsselwörtern".

4.6.10 Einstellen von Schriftstil und -farbe

Durch die Verwendung unterschiedlicher Schriftstile und -farben für die verschiedenen Sprachelemente wird eine S7-SCL-Quelle leichter lesbar und erhält ein professionelles Aussehen. Zur Formatierung des Quelltextes können Sie folgende Funktionen einsetzen:

- **Schlüsselwörter in Großbuchstaben:**
Wenn die Funktion aktiv ist, werden definierte Schlüsselwörter wie FOR, WHILE, FUNCTION_BLOCK, VAR oder END_VAR in Großbuchstaben geschrieben.
- **Schriftstil und -farbe definieren:**
Per Voreinstellung sind für die einzelnen Sprachelemente wie Operationen, Kommentare oder Konstanten bereits verschiedene Schriftstile und -farben definiert. Diese Einstellungen können Sie ändern.
Folgende Schriftfarben sind voreingestellt:

Schriftfarbe	Sprachelement	Beispiel
Blau	Schlüsselwörter	ORGANIZATION_BLOCK
	Vordefinierte Datentypen	INT
	Vordefinierte Bezeichner	ENO
	Standardfunktionen	BOOL_TO_WORD
Ocker	Operationen	NOT
Rosa	Konstanten	TRUE
Blaugrün	Kommentar	//... oder (*.*)
Violett	Globale Symbole (Symboltabelle), die in Anführungszeichen stehen	"Motor"
Schwarz	normaler Text	Variablen

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Extras > Einstellungen**.
2. Wählen Sie im nachfolgenden Dialogfeld das Register "Format".
3. Stellen Sie sicher, dass die Option "Folgende Formate beim Drucken verwenden" aktiv ist.
4. Machen Sie nun die gewünschten Einstellungen. Genaue Anweisungen zum Dialogfeld erhalten Sie, wenn Sie die Schaltfläche "Hilfe" anwählen, während das Dialogfeld geöffnet ist.

4.6.11 Platzieren von Lesezeichen im Quelltext

Mit Hilfe von Lesezeichen können Sie schnell innerhalb einer Quelle navigieren. Lesezeichen sind beispielsweise nützlich, um Änderungen durchzuführen, die sich an verschiedenen Stellen einer Quelle auswirken. Sie können in eine Quelle an beliebiger Stelle Lesezeichen einfügen. Sind mehrere Lesezeichen vorhanden, können Sie vor- und rückwärts zwischen den einzelnen Lesezeichen navigieren.

Gültigkeit

Lesezeichen gelten, solange die Quelle geöffnet ist. Sie werden nicht mit der Quelle gespeichert.

Einfügen von Lesezeichen

1. Platzieren Sie die Einfügemarke in der Zeile, die Sie markieren wollen.
2. Wählen Sie den Menübefehl **Bearbeiten > Lesezeichen ein/aus**.

Navigieren zwischen verschiedenen Lesezeichen

Wählen Sie den Menübefehl **Bearbeiten > Gehe zu > vorheriges/nächstes Lesezeichen**.

Löschen von Lesezeichen

Wählen Sie den Menübefehl **Bearbeiten > Alle Lesezeichen löschen**.

Hinweis

Die Funktionen, die Sie zum Arbeiten mit Lesezeichen benötigen, sind schnell über die Lesezeichenleiste verfügbar. Die Lesezeichenleiste blenden Sie mit dem Menübefehl **Ansicht > Lesezeichenleiste ein**.

4.6.12 Einfügen von Vorlagen

4.6.12.1 Einfügen von Bausteinvorlagen

Eine Editierfunktion von S7-SCL ist das Einfügen von Bausteinvorlagen für OB, FB, FC, DB, Instanz-DB, DB aus UDT und UDT. Durch die Bausteinvorlage wird Ihnen die Eingabe und Einhaltung der Syntax erleichtert.

Gehen Sie folgendermaßen vor:

1. Positionieren Sie die Einfügemarke an der Stelle, nach der die Bausteinvorlage eingefügt werden soll.
2. Wählen Sie den entsprechenden Menübefehl **Einfügen > Bausteinvorlage > OB/FB/FC/DB/IDB/DB aus UDT/UDT**.

4.6.12.2 Einfügen von Bausteinaufrufen

S7-SCL unterstützt Sie beim Programmieren von Bausteinaufrufen. Aufrufbare Bausteine sind:

- Systemfunktionsbausteine (SFB) und Systemfunktionen (SFC) aus den SIMATIC-Bibliotheken,
- in S7-SCL erstellte Funktionsbausteine und Funktionen,
- in anderen STEP 7-Sprachen erstellte Funktionsbausteine und Funktionen.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Einfügen > Bausteinaufruf**.
2. Wählen Sie im nachfolgenden Dialogfeld den gewünschten SFC, SFB, FC oder FB aus und bestätigen Sie mit "OK".
S7-SCL trägt den Bausteinaufruf sowie die Formalparameter des Bausteins syntaktisch korrekt in die Quelle ein.
3. Wenn Sie einen Funktionsbaustein aufgerufen haben, ergänzen Sie die Angabe des Instanz-DB.
4. Geben Sie nun die Aktualparameter zur Versorgung des Bausteins ein. Um Ihnen die Auswahl eines Aktualparameters zu erleichtern, gibt S7-SCL den geforderten Datentyp als Kommentar an.

4.6.12.3 Einfügen von Vorlagen für Kommentar

Eine Editierfunktion von S7-SCL ist das Einfügen von Vorlagen für Kommentare. Durch diese Vorlage wird Ihnen die Eingabe und Einhaltung der Syntax erleichtert.

Gehen Sie folgendermaßen vor:

1. Positionieren Sie die Einfügemarke nach dem Bausteinkopf des gewünschten Bausteins.
2. Wählen Sie den entsprechenden Menübefehl **Einfügen > Bausteinvorlage > Kommentar**.

4.6.12.4 Einfügen von Parametervorlagen

Eine Editierfunktion von S7-SCL ist das Einfügen von Vorlagen für die Vereinbarungsböcke der Parameter. Durch diese Vorlagen wird Ihnen die Eingabe und Einhaltung der Syntax erleichtert. Parameter können Sie in Funktionsbausteinen und in Funktionen vereinbaren.

Gehen Sie folgendermaßen vor:

1. Positionieren Sie die Einfügemarke im Vereinbarungsteil eines FB oder FC.
2. Wählen Sie den Menübefehl **Einfügen > Bausteinvorlage > Parameter**.

4.6.12.5 Einfügen von Kontrollstrukturen

Eine Editierfunktion von S7-SCL ist das Einfügen von Kontrollstruktur-Vorlagen für Codebausteine. Durch diese Vorlagen wird Ihnen die Eingabe und Einhaltung der Syntax erleichtert.

Gehen Sie folgendermaßen vor:

1. Positionieren Sie die Einfügemarke an der Stelle, nach der die Vorlage eingefügt werden soll.
2. Wählen Sie den entsprechenden Menübefehl **Einfügen > Kontrollstruktur > IF/CASE/FOR/WHILE/REPEAT**.

4.7 Übersetzen eines S7-SCL-Programms

4.7.1 Wissenswertes zum Übersetzen

Bevor Sie Ihr Programm ablaufen lassen oder testen können, müssen Sie es übersetzen. Durch Anstoßen des Übersetzungsvorgangs aktivieren Sie den Compiler. Der Compiler hat folgende Eigenschaften:

- Sie können eine S7-SCL-Quelle in einem Übersetzungslauf als Ganzes bearbeiten oder einzelne Bausteine aus der Quelle selektiv übersetzen.
- Alle syntaktischen Fehler, die während der Übersetzung gefunden werden, werden anschließend in einem Fenster angezeigt.
- Bei jedem Aufruf eines Funktionsbausteins wird ein zugehöriger Instanz-Datenbaustein generiert, sofern dieser nicht schon existiert.
- Sie haben auch die Möglichkeit, mehrere S7-SCL-Quellen in einem Lauf zu übersetzen, indem Sie eine S7-SCL-Übersetzungssteuerdatei erstellen.
- Über den Menübefehl **Extras > Einstellungen** können Sie Optionen für den Compiler einstellen.

Nachdem Sie ein Anwenderprogramm fehlerfrei erstellt und übersetzt haben, können Sie davon ausgehen, dass das Programm korrekt ist. Trotzdem können Probleme auftreten, wenn das Programm im AS läuft. Nutzen Sie die Testfunktionen von S7-SCL, um solche Fehler zu ermitteln.

4.7.2 Einstellen des Compilers

Sie haben die Möglichkeit, den Übersetzungsvorgang an Ihre individuellen Anforderungen anzupassen.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Extras > Einstellungen**, um das Dialogfeld "Einstellungen" zu öffnen.
2. Wählen Sie das Register "Compiler" bzw. das Register "Baustein erzeugen".
3. Tragen Sie in das Register die Optionen ein.

Optionen im Register "Compiler"

Objektcode erstellen	Mit dieser Option legen Sie fest, ob Sie einen ablauffähigen Code erzeugen möchten oder nicht. Der Übersetzungsvorgang ohne diese Option dient lediglich einer Syntaxprüfung.
Objektcode optimieren	Wenn Sie diese Option wählen, werden die erzeugten Bausteine bezüglich ihres Speicherbedarfs und ihrer Laufzeit im AS optimiert. Es ist empfehlenswert, diese Option immer zu wählen, da durch die Optimierung keine Nachteile entstehen, die die Funktionalität des Bausteins beeinträchtigen.
Feldgrenzen überwachen	Wenn Sie diese Option wählen, wird zur Laufzeit des S7-Programms überprüft, ob Feldindizes in dem - laut Vereinbarung für das Feld (ARRAY) - zulässigen Bereich liegen. Wenn ein Feldindex den zulässigen Bereich überschreitet, wird das OK-Flag auf FALSE gesetzt.
Debug Info erstellen	Diese Option ermöglicht es Ihnen, einen Testlauf mit dem Debugger durchzuführen, nachdem Sie das Programm übersetzt und in die CPU geladen haben. Der Speicherbedarf des Programms und die Ablaufzeiten im AS erhöhen sich jedoch durch diese Option.
OK Flag setzen	Diese Option ermöglicht Ihnen das Abfragen des OK Flags in Ihren S7-SCL-Quelltexten.
Geschachtelte Kommentare zulassen	Wählen Sie diese Option, wenn Sie in Ihrer S7-SCL-Quelle mehrere Kommentare ineinander schachteln möchten.
Maximale Stringlänge	Hier können Sie die Standardlänge des Datentyps STRING reduzieren. In der Grundeinstellung beträgt sie 254 Zeichen. Die Einstellung bezieht sich auf alle Ausgangs- und Durchgangparameter sowie auf Rückgabewerte von Funktionen. Beachten Sie, dass der eingestellte Wert nicht kleiner sein darf als die im Programm verwendeten STRING-Variablen.

Optionen im Register "Baustein erzeugen"

Bausteine überschreiben	Überschreibt bereits existierende Bausteine im Ordner "Bausteine" eines S7-Programms, falls beim Übersetzungsvorgang Bausteine mit der gleichen Bezeichnung erzeugt werden. Ebenso werden beim Laden Bausteine mit gleichem Namen, die bereits im Zielsystem vorhanden sind, überschrieben. Wenn Sie diese Option nicht gewählt haben, müssen Sie eine Meldung bestätigen, bevor der Baustein überschrieben wird.
Warnungen anzeigen	Legt fest, ob nach einem Übersetzungslauf zusätzlich zu den Fehlern auch Warnungen gemeldet werden sollen.
Fehler vor Warnungen	Legt fest, ob im Ausgabefenster Fehler vor Warnungen aufgelistet werden.
Referenzdaten erzeugen	Wählen Sie diese Option, wenn bei der Erzeugung eines Bausteins automatisch Referenzdaten erzeugt werden sollen. Über den Menübefehl Extras > Referenzdaten haben Sie jedoch die Möglichkeit, die Referenzdaten später zu erzeugen oder zu aktualisieren.
Systemattribut "S7_server" berücksichtigen	Wählen Sie diese Option, wenn bei der Erzeugung eines Bausteins das Systemattribut für Parameter "S7-server" berücksichtigt werden soll. Dieses Attribut vergeben Sie, wenn der Parameter für die Verbindungs- oder Meldungsprojektierung relevant ist. Er enthält die Verbindungs- bzw. die Meldungsnummer. Diese Option verlängert den Übersetzungsvorgang.

4.7.3 Übersetzen des Programms

Bevor Sie ein Programm testen oder ablaufen lassen können, müssen Sie es übersetzen. Um sicherzugehen, dass Sie immer die neuste Version Ihrer S7-SCL-Quelle übersetzen, ist es ratsam, den Menübefehl **Extras > Einstellungen** zu wählen und im Register "Editor" die Option "Sichern vor Übersetzen" anzuklicken. Der Menübefehl **Datei > Übersetzen** speichert die S7-SCL-Quelle dadurch implizit.

Gehen Sie folgendermaßen vor:

1. Speichern Sie die zu übersetzende S7-SCL-Quelle ab.
2. Um ein ablauffähiges Programm zu erzeugen, ist es unbedingt notwendig, die Option "Objektcode erstellen" im Dialogfeld "Einstellungen", Register "Compiler" zu wählen.
3. Modifizieren Sie eventuell weitere Einstellungen des Compilers.
4. Stellen Sie fest, ob die zugehörige Symboltabelle sich im selben Programmverzeichnis befindet.
5. Um den Übersetzungsvorgang zu starten, haben Sie folgende Möglichkeiten:
 - Der Menübefehl **Datei > Übersetzen** übersetzt die Quelle als Ganzes.
 - Der Menübefehl **Datei > Teil-Übersetzen** öffnet ein Dialogfeld, in dem Sie einzelne Bausteine zur Übersetzung auswählen können.
6. Im Dialogfeld "Ausgaben" werden alle syntaktischen Fehler und Warnungen angezeigt, die während der Übersetzung des Programms aufgetreten sind.

Korrigieren Sie nach dem Übersetzungsvorgang eventuell gemeldete Fehler und wiederholen Sie die oben geschilderte Vorgehensweise.

4.7.4 Erstellen einer Übersetzungssteuerdatei

Eine Übersetzungssteuerdatei gibt Ihnen die Möglichkeit, mehrere S7-SCL-Quellen innerhalb eines Quell-Ordners in einem Lauf zu übersetzen. In der Übersetzungssteuerdatei tragen Sie die Namen der S7-SCL-Quellen in der Reihenfolge ein, in der sie übersetzt werden sollen.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie das Dialogfeld "Neu", indem Sie den Menübefehl **Datei > Neu** wählen.
2. Wählen Sie im Dialogfeld "Neu"
 - einen Quell-Ordner innerhalb eines S7-Programms und
 - den Objekttyp "SCL-Übersetzungssteuerdatei" aus.
3. Tragen Sie den Namen der Steuerdatei in das entsprechende Textfeld ein (max. 24 Zeichen) und bestätigen Sie mit "OK".
4. Die Datei wird angelegt und in einem Arbeitsfenster zur weiteren Bearbeitung angezeigt.
Geben Sie in das Arbeitsfenster die Namen der zu übersetzenden S7-SCL-Quellen in der gewünschten Reihenfolge ein und speichern Sie die Datei ab.
5. Starten Sie anschließend den Übersetzungsvorgang, indem Sie den Menübefehl **Datei > Übersetzen** wählen.

4.7.5 Beheben von Fehlern nach dem Übersetzen

Alle syntaktischen Fehler und Warnungen, die beim Übersetzen auftreten, werden nach dem Vorgang im Fenster "Ausgaben" angezeigt. Das Auftreten eines Fehlers verhindert die Erzeugung des jeweiligen Bausteins, während beim Auftreten von Warnungen dennoch ein ablauffähiger Baustein erzeugt wird. Der Ablauf im AS kann jedoch fehlerhaft sein.

Gehen Sie folgendermaßen vor, um einen Fehler zu beheben:

1. Markieren Sie den Fehler und drücken Sie die Taste F1, um eine Beschreibung des Fehlers und Maßnahmen zur Fehlerbehebung anzuzeigen.
2. Falls in der Meldung eine Zeilen- (Zxx) und Spaltennummer (Sxx) angegeben wird, können Sie die fehlerhafte Stelle des Quelltextes lokalisieren, indem Sie
 - die Fehlermeldung im Fenster "Ausgaben" anklicken, mit der rechten Maustaste das Kontext-Menü aufrufen und dort den Befehl **Fehler anzeigen** anwählen.
 - auf die Fehlermeldung doppelklicken, um die Einfügemarke an der gemeldeten Stelle (Zeile, Spalte) zu positionieren.

3. Informieren Sie sich über die korrekte Syntax in der S7-SCL-Sprachbeschreibung.
4. Korrigieren Sie den Quelltext dementsprechend.
5. Speichern Sie die Quelle ab.
6. Übersetzen Sie die Quelle erneut.

4.8 Speichern und Drucken einer S7-SCL-Quelle

4.8.1 Speichern einer S7-SCL-Quelle

Unter dem Begriff "Speichern" wird in S7-SCL immer das Speichern der Quelldateien verstanden. Bausteine werden in S7-SCL beim Übersetzen der Quelldatei erzeugt und automatisch im zugehörigen Programmverzeichnis abgelegt. Sie können eine S7-SCL-Quelle jederzeit im aktuellen Zustand abspeichern. Das Objekt wird dabei nicht übersetzt. Syntaktische Fehler werden mit abgespeichert.

Gehen Sie folgendermaßen vor:

- Wählen Sie den Menübefehl **Datei > Speichern**, oder klicken Sie das Symbol "Speichern" in der Funktionsleiste an.
Die S7-SCL-Quelle wird aktualisiert.
- Wenn Sie eine Kopie der aktuellen S7-SCL-Quelle erstellen möchten, wählen Sie den Menübefehl **Datei > Speichern unter**. Es erscheint das Dialogfeld "Speichern unter", in dem Sie den Namen und den Pfad des Duplikats angeben können.

4.8.2 Einstellen des Seitenformats

Das Erscheinungsbild eines Ausdrucks können Sie wie folgt beeinflussen:

- Mit dem Menübefehl **Datei > Seite einrichten** können Sie Seitenformat und -ausrichtung Ihres Ausdrucks festlegen.
- Kopf- und Fußzeilen für Ihre zu druckenden Dokumente können Sie eingeben, indem Sie den Menübefehl **Datei > Seite einrichten** anwählen und im nachfolgenden Dialogfeld das Register "Schriftfelder" selektieren.
- Mit dem Menübefehl **Datei > Drucker einrichten** können Sie weitere druckerabhängige Einstellungen machen.

Achtung

Angaben zur Seitenausrichtung müssen Sie im Dialogfeld "Seite einrichten" machen. Die Angaben im Dialogfeld "Drucker einrichten" sind für den Ausdruck von S7-SCL-Quellen nicht relevant.

- Mit dem Menübefehl **Datei > Druckvorschau** können Sie die vorgenommenen Einstellungen in der Seitenansicht überprüfen, bevor Sie das Dokument zum Drucker senden.

4.8.3 Drucken einer S7-SCL-Quelle

Gedruckt wird die S7-SCL-Quelle des aktiven Arbeitsfensters, d.h. um eine S7-SCL-Quelle drucken zu können, muss sie geöffnet sein.

Gehen Sie folgendermaßen vor:

1. Aktivieren Sie das Arbeitsfenster der S7-SCL-Quelle, deren Inhalt Sie ausdrucken möchten.
2. Öffnen Sie das Dialogfeld "Drucken", indem Sie
 - das Symbol "Drucken" anklicken, oder
 - den Menübefehl **Datei > Drucken** wählen.
3. Wählen Sie im Dialogfeld "Drucken" die Druckoptionen aus, wie z.B. den Druckbereich oder die Anzahl der Kopien.
4. Bestätigen Sie mit "OK".

4.8.4 Einstellen der Druckoptionen

Zur Formatierung des Ausdrucks können Sie folgende Funktionen verwenden:

- **Seitenumbruch am Bausteinbeginn/Bausteinende**
Wenn die Funktion aktiv ist, beginnt jeder Baustein beim Ausdruck mit einer neuen Seite oder endet mit einem Seitenumbruch.
- **Ausdruck mit Zeilennummern**
Wenn die Funktion aktiv ist, werden zu Beginn jeder Zeile Zeilennummern ausgedruckt.
- **Ausdruck in Farbe**
Wenn die Funktion aktiv ist, werden beim Druck die in der Quelle verwendeten Farben ausgegeben.
- **Schrift beim Drucken**
Per Voreinstellung ist für den gesamten Text die Schriftart Courier New in Schriftgröße 8 definiert. Diese Schriftart liefert optimale Druckergebnisse.
- **Schriftstil**
Sie können verschiedene Schriftstile für einzelne Sprachelemente definieren. Folgende Elemente sind einzeln auswählbar:

Sprachelement	Beispiel
Normaler Text	
Schlüsselwörter	ORGANIZATION_BLOCK
Bezeichner von vordefinierten Datentypen	INT
Vordefinierte Bezeichner	ENO
Bezeichner von Standardfunktionen	BOOL_TO_WORD
Operationen	NOT
Konstanten	TRUE
Kommentarblock	(* *)
Zeilenkommentar	//...
Globale Symbole (Symboltabelle), die in Anführungszeichen stehen	"Motor"

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Extras > Einstellungen**.
2. Wählen Sie im nachfolgenden Dialogfeld das Register "Drucken".
3. Stellen Sie sicher, dass die Option "Folgende Formate verwenden" aktiv ist.
4. Machen Sie nun die gewünschten Einstellungen. Genaue Anweisungen zum Dialogfeld erhalten Sie, wenn Sie die Schaltfläche "Hilfe" anwählen, während das Dialogfeld geöffnet ist.

4.9 Laden der erstellten Programme

4.9.1 Urlöschen des CPU-Speichers

Mit der Funktion Urlöschen können Sie online das gesamte Anwenderprogramm in einer CPU löschen.

Gehen sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Betriebszustand** und schalten Sie die CPU auf STOP.
2. Wählen Sie den Menübefehl **Zielsystem > Urlöschen**.
3. Bestätigen Sie die Aktion im daraufhin erscheinenden Dialogfeld.



Warnung

- Die CPU wird zurückgesetzt.
 - Alle Anwenderdaten werden gelöscht.
 - Die CPU bricht alle bestehenden Verbindungen ab.
 - Wenn eine Memory Card gesteckt ist, kopiert die CPU nach dem Urlöschen den Inhalt der Memory Card in den internen Ladespeicher.
-

4.9.2 Laden von Anwenderprogrammen in die CPU

Voraussetzungen

Bei der Übersetzung einer S7-SCL-Quelle werden aus der Quelle die Bausteine erzeugt und im Ordner "Bausteine" des S7-Programms gespeichert.

Bausteine, die in erster Ebene aus den S7-SCL-Bausteinen heraus aufgerufen werden, werden gesucht, automatisch mit in das Verzeichnis "Bausteine" kopiert und in die Ladeliste übernommen.

Weitere Bausteine aus dem Anwenderprogramm laden Sie mit dem SIMATIC Manager aus dem Programmiergerät in die CPU.

Voraussetzung für das Laden ist jeweils eine bestehende Verbindung zwischen PG und CPU. Der CPU-Baugruppe muss ein Online-Anwenderprogramm im SIMATIC Manager zugeordnet sein.

Vorgehensweise

Nachdem Sie die Quelle übersetzt haben, haben Sie folgende Möglichkeiten, den Ladevorgang zu starten:

- Der Menübefehl **Datei > Laden** lädt alle Bausteine, die in der Quelle enthalten sind, sowie alle Bausteine, die in erster Ebene aufgerufen werden.
- Der Menübefehl **Datei > Teil-Laden** öffnet ein Dialogfeld, in dem Sie einzelne Bausteine zum Laden auswählen können.

Die Bausteine werden in die CPU übertragen. Ist ein Baustein schon im RAM der CPU vorhanden, bestätigen Sie nach Rückfrage, ob der Baustein überschrieben werden soll.

Hinweis

Das Laden des Anwenderprogramms im Betriebszustand STOP ist vorteilhaft, da beim Überschreiben eines alten Programms im Betriebszustand RUN Fehler entstehen können.

4.10 Testen der erstellten Programme

4.10.1 Die Testfunktionen von S7-SCL

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.

SCL-Testfunktionen

In S7-SCL können Sie die folgenden Testfunktionen nutzen:

- **Beobachten**
Mit dieser Funktion können Sie Namen und aktuelle Werte von Variablen in der S7-SCL-Quelle ausgeben. Während des Testlaufs werden die Werte der Variablen und Parameter dieses Bereichs in chronologischer Abfolge angezeigt und zyklisch aktualisiert.
- **Testen mit Haltepunkten/Einzelschritt**
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.



Vorsicht

Ein Test bei laufendem Anlagenbetrieb kann bei Funktionsstörungen oder Programmfehlern schwere Sach- und Personenschäden verursachen!

Vergewissern Sie sich, dass keine gefährlichen Zustände eintreten können, bevor Sie die Testfunktionen ausführen!

Voraussetzungen für den Testlauf

- Das Programm wurde mit den Optionen "Objektcode erstellen" und "Debug Info erstellen" übersetzt.
Die Optionen können Sie im Register "Compiler" des Dialogfelds "Einstellungen" wählen. Das Dialogfeld können Sie über den Menübefehl **Extras > Einstellungen** öffnen.
- Aktuelle Referenzdaten stehen zur Verfügung.
Referenzdaten werden automatisch beim Übersetzen erzeugt, wenn die Option "Debug Info erstellen" aktiv ist.
- Zwischen PG/PC und CPU besteht eine Online-Verbindung.
- Das Programm wurde in die CPU geladen.
Das können Sie mit dem Menübefehl **Zielsystem > Laden** durchführen.

4.10.2 Wissenswertes zur Testfunktion "Beobachten"

Testfunktion "Beobachten"

Beim kontinuierlichen Beobachten eines Programms können Sie eine Gruppe von Anweisungen testen. Diese Gruppe von Anweisungen nennt man Beobachtungsbereich. Während des Testlaufs werden die Werte der Variablen und Parameter dieses Bereichs in chronologischer Abfolge angezeigt und zyklisch aktualisiert. Liegt der Beobachtungsbereich in einem Programmteil, der in jedem Zyklus durchlaufen wird, können die Werte der Variablen in der Regel nicht aufeinander folgenden Zyklen erfasst werden.

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

Beobachtungsbereich

Der Umfang der beobachtbaren Anweisungen ist durch die Leistungsfähigkeit der angeschlossenen CPU begrenzt.

Da die S7-SCL-Anweisungen des Quellcodes in unterschiedlich viele Anweisungen im Maschinencode abgebildet werden, ist die Länge des Beobachtungsbereichs variabel, sie wird vom S7-SCL-Debugger ermittelt und markiert, wenn Sie die erste Anweisung des gewünschten Beobachtungsbereichs auswählen.

Zudem haben Sie die Möglichkeit, einen Beobachtungsbereich gezielt zu definieren. Markieren Sie hierzu in der Quelle die zu beobachtenden Anweisungen.

Testmodi

Das Abfragen dieser Testinformationen bewirkt meist eine Verlängerung der Zykluszeiten. Um diese Verlängerung beeinflussen zu können, bietet S7-SCL zwei verschiedene Modi an.

Betriebsmodus	Erläuterung
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 "Prozess" schränkt der S7-SCL-Debugger den maximalen Beobachtungsbereich so ein, dass die Zykluszeiten beim Testvorgang die realen Ablaufzeiten des Prozesses nicht oder nur unwesentlich überschreiten.

Einschränkungen

Bei der Funktion "Beobachten" müssen Sie folgende Einschränkungen beachten:

- Variablen von einem höheren Datentyp können nicht als Ganzes angezeigt werden. Die Elemente solcher Variablen, sofern sie von einem elementaren Datentyp sind, lassen sich beobachten.
- Variablen vom Typ DATE_AND_TIME und STRING sowie Parameter vom Typ BLOCK_FB, BLOCK_FC, BLOCK_DB, BLOCK_SDB, TIMER und COUNTER werden nicht angezeigt.
- Zugriffe auf Datenbausteine der Form <Symbol>.<absoluteAdresse> werden nicht angezeigt (z.B. Daten.DW4).

4.10.3 Wissenswertes zum "Testen mit Haltepunkten/Einzelschrittmodus"

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.

Nach dem Setzen von Haltepunkten können Sie das Programm zunächst bis zu einem Haltepunkt ausführen lassen und von dort aus mit dem schrittweisen Beobachten beginnen.

Einzelschrittfunktionen:

Wenn die Testfunktion "Testen mit Haltepunkten" gestartet ist, können Sie folgende Funktionen durchführen:

- Nächste Anweisung ausführen
Die aktuell markierte Anweisung wird ausgeführt.
- Fortsetzen
Fortsetzung bis zum nächsten aktiven Haltepunkt.
- Ausführung bis Markierung
Fortsetzung bis zu einer Textmarke in der Quelle, die Sie definieren.
- Aufruf ausführen
Sprung in einen unterlagerten S7-SCL-Baustein bzw. Aufruf eines unterlagerten Bausteins.

Haltepunkte:

Haltepunkte können Sie an beliebigen Stellen im Anweisungsteil des Quelltextes definieren.

Erst wenn Sie den Menübefehl **Test > Haltepunkte aktiv** wählen, werden die Haltepunkte in das Automatisierungssystem übertragen und aktiv geschaltet. Das Programm wird dann solange ausgeführt, bis der erste Haltepunkt erreicht wird.

Die maximale Anzahl aktiver Haltepunkte ist CPU-abhängig.

Voraussetzung:

Die geöffnete Quelle darf nicht im Editor geändert worden sein.

4.10.4 Schritte zum Beobachten

Nachdem Sie das übersetzte Programm in das Zielsystem geladen haben, können Sie es im Modus "Beobachten" testen.

Hinweis

Der Umfang der zu testenden Anweisungen (größtmöglicher Beobachtungsbereich) ist von der Leistungsfähigkeit der angeschlossenen CPU abhängig.

Gehen Sie folgendermaßen vor:

1. Stellen Sie sicher, dass die Voraussetzungen für den Testlauf erfüllt sind und die CPU sich im Betriebszustand RUN oder RUN-P befindet.
2. Wählen Sie das Fenster aus, das die Quelle des zu testenden Programms enthält.
3. Wenn Sie den voreingestellten Betriebsmodus (Prozess) ändern wollen, wählen Sie den Menübefehl **Test > Betrieb > Testbetrieb**.
4. Definieren Sie den Beobachtungsbereich. Hierzu haben Sie zwei Möglichkeiten:
 - Positionieren Sie die Einfügemarke in der Zeile des Quelltextes, die die erste Anweisung des zu testenden Bereichs enthält. S7-SCL wählt nun einen maximalen Beobachtungsbereich ab der Position der Einfügemarke.
 - Markieren Sie im Quelltext gezielt die Anweisungen, die Sie beobachten möchten.
5. Vergewissern Sie sich, dass keine gefährlichen Zustände eintreten können.
6. Wählen Sie den Menübefehl **Test > Beobachten**.
7. Deaktivieren Sie den Menübefehl **Test > Beobachten**, um den Testlauf zu unterbrechen
8. Wählen Sie den Menübefehl **Test > Test beenden**, um den Testlauf zu beenden.

Ergebnis

Der Beobachtungsbereich wird ermittelt und durch einen grauen Balken am linken Rand des Fensters angezeigt. Das Fenster teilt sich, und im rechten Teil des Fensters werden die Namen und aktuellen Werte der im Beobachtungsbereich liegenden Variablen zeilengerecht angezeigt.

Anpassen des Beobachtungsfunktion

Sie haben folgende, Möglichkeiten, die Beobachtungsfunktion an Ihre Bedürfnisse anzupassen:

- Definieren Sie eine Aufrufumgebung für den zu beobachtenden Baustein.
- Wählen Sie den Menübefehl **Ansicht > Symbolische Darstellung**, um die Anzeige der symbolischen Namen im Programm aus- bzw. einzuschalten.
- Wählen Sie den Menübefehl **Extras > Einstellungen** und machen Sie im Register "Format" die gewünschten Einstellungen bezüglich der Farbe, in der die Werte dargestellt werden.

4.10.4.1 Definieren einer Aufrufumgebung für Bausteine

Aufrufumgebung

Um den Beobachtungsbereich noch gezielter zu definieren, können Sie eine Aufrufumgebung für die zu beobachtenden Bausteine definieren. Sie legen dabei fest, dass ein Baustein nur unter einer der folgenden Bedingungen beobachtet wird:

Bedingung	Auswahlmöglichkeit
Der Baustein wird von einem bestimmten übergeordneten Baustein aufgerufen.	Aufrufpfad
Der Baustein wird zusammen mit einem bestimmten Datenbaustein aufgerufen.	Globaler Datenbaustein und / oder Instanz-Datenbaustein

Gehen Sie folgendermaßen vor, um eine Aufrufpfad zu definieren

1. Wählen Sie den Menübefehl **Test > Aufrufumgebung Bausteine**.
Im nachfolgenden Dialogfeld wird eine Liste der vorhandenen Bausteine angezeigt.
2. Wählen Sie einen Baustein aus der Liste aus.
3. Aktivieren Sie die Option "Aufrufpfad aktivieren".
Im unteren Teilfenster werden nun die möglichen Aufrufpfade grafisch dargestellt.
4. Wählen Sie den gewünschten Aufrufpfad.

Gehen Sie folgendermaßen vor, um einen Datenbaustein zu definieren

1. Wählen Sie den Menübefehl **Test > Aufrufumgebung Bausteine**.
Im nachfolgenden Dialogfeld wird eine Liste der vorhandenen Bausteine angezeigt.
2. Wählen Sie einen Baustein aus der Liste aus.
3. Aktivieren Sie die Option "Offene Datenbausteine".
4. Geben Sie die Nummer des gewünschten DB an.

Hinweis

Wenn Sie eine Aufrufumgebung definiert haben, gehen Sie folgendermaßen vor, um die Beobachtung zu starten:

1. Positionieren Sie die Einfügemarke in dem zu beobachtenden Baustein, für den die Aufrufumgebung definiert wurde.
 2. Wählen Sie den Menübefehl **Test > Beobachten**.
Die Beobachtungsfunktion wird gestartet. Der Baustein wird beobachtet, wenn alle von Ihnen definierten Aufrufbedingungen erfüllt sind.
-

4.10.5 Schritte zum Testen mit Haltepunkten

4.10.5.1 Definieren von Haltepunkten

Setzen und definieren Sie Haltepunkte folgendermaßen:

1. Öffnen Sie die zu testende Quelle.
2. Blenden Sie mit dem Menübefehl **Ansicht > Haltepunkteleiste** die Funktionsleiste für Haltepunktbearbeitung ein.
3. Setzen Sie die Einfügemarke an der gewünschten Stelle und wählen Sie den Menübefehl **Test > Haltepunkt setzen** oder die entsprechende Schaltfläche auf der Haltepunkteleiste. Die Haltepunkte werden am linken Fensterrand als roter Kreis dargestellt.
4. Falls gewünscht, wählen Sie den Menübefehl **Test > Haltepunkte bearbeiten** und definieren eine Aufrufumgebung. Die Aufrufumgebung legt fest, ob ein Haltepunkt nur dann aktiv ist, wenn der Baustein, in dem er sich befindet,
 - von einem bestimmten übergeordneten Baustein aufgerufen wird, und/oder
 - zusammen mit einem bestimmten Datenbaustein/Instanz-Datenbaustein aufgerufen wird.

4.10.5.2 Starten des Tests mit Haltepunkten

Nachdem Sie das übersetzte Programm in das Zielsystem geladen und Haltepunkte gesetzt haben, können Sie es im Modus "Testen mit Haltepunkten" testen.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie die S7-SCL-Quelle des zu testenden Programms.
2. Vergewissern Sie sich, dass keine gefährlichen Zustände eintreten können und die CPU sich im Zustand RUN-P befindet.
3. Wählen Sie den Menübefehl **Test > Haltepunkte aktiv**.
Ergebnis: Das Fenster wird vertikal in zwei Hälften geteilt. Das Programm läuft bis zum nächsten Haltepunkt. Wird er erreicht, geht die CPU in den Betriebszustand HALT und der rote Haltepunkt wird mit einem gelben Pfeil markiert.
4. Fahren Sie nun fort, indem Sie
 - den Menübefehl **Test > Fortsetzen** wählen oder das Symbol "Fortsetzen" anklicken.
Die CPU geht in den Betriebszustand RUN über. Beim Erreichen des nächsten aktiven Haltepunkts hält sie erneut und zeigt den Haltepunkt in der linken Fensterhälfte an.
 - den Menübefehl **Test > Nächste Anweisung ausführen** wählen oder das Symbol "Nächste Anweisung ausführen" anklicken.
Die CPU geht in den Betriebszustand RUN über. Nach Bearbeitung der markierten Anweisung hält sie erneut und zeigt die Inhalte der gerade bearbeiteten Variablen in der rechten Fensterhälfte an.

- den Menübefehl **Test > Ausführen bis Markierung** wählen oder das Symbol "Ausführen bis Markierung" anklicken.
Die CPU geht in den Betriebszustand RUN über. Beim Erreichen der markierten Programmstelle hält sie erneut.
- den Menübefehl **Test > Aufruf ausführen** wählen, wenn das Programm in einer Zeile angehalten wird, die einen Bausteinaufruf enthält.
Wenn der unterlagerte Baustein mit S7-SCL erstellt wurde, wird er im Testbetrieb geöffnet und bearbeitet. Nach der Bearbeitung springt das Programm wieder zurück an die Aufrufstelle.
Wenn der Baustein in einer anderen Sprache erstellt wurde, wird der Aufruf übersprungen und die nachfolgende Programmzeile markiert.

Hinweis

Die Menübefehle **Test > Nächste Anweisung ausführen** oder **Test > Ausführen bis Markierung** setzen und aktivieren implizit einen Haltepunkt. Achten Sie darauf, dass die CPU-spezifische maximale Anzahl aktiver Haltepunkte nicht erreicht ist, wenn Sie diese Funktionen wählen.

4.10.5.3 Beenden des Tests mit Haltepunkten

Um zum normalen Programmablauf zurückzukehren, gehen Sie folgendermaßen vor:

- Deaktivieren Sie den Menübefehl **Test > Haltepunkte aktiv**, um den Testvorgang zu unterbrechen, oder
- wählen Sie den Menübefehl **Test > Test beenden**, um den Testvorgang zu beenden.

4.10.5.4 Aktivieren, Deaktivieren und Löschen von Haltepunkten

Sie können gesetzte Haltepunkte einzeln aktivieren / deaktivieren und löschen:

1. Wählen Sie den Menübefehl **Test > Haltepunkte bearbeiten**.
2. Im nachfolgenden Dialogfeld können Sie
 - Haltepunkte selektiv aktivieren und deaktivieren, indem Sie sie mit einem Häkchen markieren.
 - Einzelne Haltepunkte löschen.

Um alle Haltepunkte zu löschen, wählen Sie den Menübefehl **Test > Alle Haltepunkte löschen**.

4.10.5.5 Definieren einer Aufrufumgebung für Haltepunkte

Aufrufumgebung

Durch die Definition einer Aufrufumgebung legen Sie fest, dass ein Haltepunkt nur unter einer der folgenden Bedingungen gültig ist:

Bedingung	Auswahlmöglichkeit
Der Baustein, in dem sich der Haltepunkte befindet, wird von einem bestimmten übergeordneten Baustein aufgerufen.	Aufrufpfad
Der Baustein, in dem sich der Haltepunkt befindet, wird zusammen mit einem bestimmten Datenbaustein aufgerufen.	Globaler Datenbaustein und / oder Instanz-Datenbaustein

Gehen Sie folgendermaßen vor, um einen Aufrufpfad zu definieren

1. Wählen Sie den Menübefehl **Test > Haltepunkte bearbeiten**.
Im nachfolgenden Dialogfeld wird eine Liste der vorhandenen Haltepunkte angezeigt.
2. Wählen Sie einen Haltepunkt aus der Liste aus.
3. Aktivieren Sie die Option "Aufrufpfad aktivieren".
Im unteren Teilfenster werden nun die möglichen Aufrufpfade grafisch dargestellt.
4. Wählen Sie den gewünschten Aufrufpfad.

Gehen Sie folgendermaßen vor, um einen Datenbaustein zu definieren

1. Wählen Sie den Menübefehl **Test > Haltepunkte bearbeiten**.
Im nachfolgenden Dialogfeld wird eine Liste der vorhandenen Haltepunkte angezeigt.
2. Wählen Sie einen Haltepunkt aus der Liste aus.
3. Aktivieren Sie die Option "Offene Datenbausteine".
4. Geben Sie die Nummer des gewünschten DB an.

4.10.5.6 Testen im Einzelschrittmodus

Gehen Sie folgendermaßen vor:

1. Setzen Sie einen Haltepunkt vor die Anweisungszeile, ab der Sie Ihr Programm im Einzelschrittmodus testen wollen.
2. Wählen Sie den Menübefehl **Test > Haltepunkte aktiv**.
3. Lassen Sie das Programm bis zu diesem Haltepunkt ablaufen.
4. Um die nächste Anweisung auszuführen, wählen Sie Menübefehl **Test > Nächste Anweisung ausführen**.
 - Handelt es sich um einen Bausteinaufruf, so wird der Aufruf bearbeitet und es wird zur ersten Anweisung nach dem Bausteinaufruf gesprungen.
 - Mit dem Menübefehl **Test > Aufruf ausführen** springen Sie in den Baustein. Dort können Sie dann im Einzelschritt weitertesten oder Sie können auch Haltepunkte setzen. Am Ende des Bausteins erfolgt ein Rücksprung zur nächsten Anweisung nach dem Bausteinaufruf.

4.10.6 Verwenden der STEP 7-Testfunktionen

4.10.6.1 Erzeugen bzw. Anzeigen der Referenzdaten

Sie können Referenzdaten erzeugen und auswerten, um sich das Testen und Ändern Ihres Anwenderprogramms zu erleichtern.

Folgende Referenzdaten können angezeigt werden:

- die Anwenderprogrammstruktur
- die Querverweisliste
- der Belegungsplan
- die Liste nicht verwendeter Operanden
- die Liste der Operanden ohne Symbol

Erzeugung der Referenzdaten

Zur Erzeugung der Daten haben Sie folgende Möglichkeiten:

- Über den Menübefehl **Extras > Referenzdaten anzeigen** können Sie die Daten bei Bedarf erzeugen bzw. aktualisieren und anzeigen lassen.
- Durch Filtern können Sie die angezeigte Menge von Referenzdaten gezielt einschränken und so die Erzeugung und Anzeige der Daten wesentlich beschleunigen. Wählen Sie hierzu den Menübefehl **Extras > Referenzdaten > Filtern**.
- Über den Menübefehl **Extras > Einstellungen** können Sie festlegen, dass die Referenzdaten automatisch beim Übersetzen einer Quelle erzeugt werden. Markieren Sie dazu im Register "Baustein erzeugen" den Eintrag "Referenzdaten erzeugen".
Das automatische Erzeugen der Referenzdaten verlängert jedoch den Übersetzungsvorgang.

4.10.6.2 Beobachten und Steuern von Variablen

Beim Testen mit der Funktion "Variablen beobachten/steuern", können Sie:

- die aktuellen Werte von globalen Daten aus Ihrem Anwenderprogramm anzeigen lassen (beobachten).
- den Variablen des Anwenderprogramms feste Werte zuweisen (steuern).

Gehen Sie folgendermaßen vor:

- Wählen Sie den Menübefehl **Zielsystem > Variablen beobachten/steuern**.
- Erstellen Sie die Variablentabelle (VAT) im daraufhin angezeigten Fenster. Falls Sie Variablen steuern möchten, geben Sie zusätzlich die gewünschten Werte an.
- Legen Sie die Triggerpunkte und -bedingungen fest.



Vorsicht

Ein Verändern der Variablenwerte bei laufendem Anlagenbetrieb kann bei Funktionsstörungen oder Programmfehlern schwere Sach- und Personenschäden verursachen! Vergewissern Sie sich, dass keine gefährlichen Zustände eintreten können, bevor Sie diese Funktion ausführen!

4.10.6.3 Prüfen der Bausteinkonsistenz

Hinweis

Diese Funktion ist ab STEP 7 V5.3 SP2 verfügbar.

Wird eine S7-SCL-Quelle geändert, so müssen eventuell darin referenzierte Bausteine ebenfalls angepasst werden. Ansonsten können Inkonsistenzen im Programm auftreten. Auch wenn die Zeitstempel von Quellen und Bausteinen nicht übereinstimmen, können Inkonsistenzen entstehen.

Durch die STEP 7-Funktion "Bausteinkonsistenz prüfen" können Sie solche Inkonsistenzen aufdecken und Fehler schneller beheben.

Nach Programmänderungen stoßen Sie eine Konsistenzprüfung über alle S7-SCL-Quellen im Projekt an. Bei Inkonsistenzen, die nicht automatisch bereinigt werden können, führt Sie die Funktion an die zu ändernden Positionen in der Quelle. Dort nehmen Sie die notwendigen Änderungen vor. Schritt für Schritt werden alle Inkonsistenzen bereinigt.

Voraussetzungen

- Auf Ihrem Gerät ist STEP 7 V5.3 SP2 oder höher installiert.
- Eine Quelle, die an der Funktion "Bausteinkonsistenz prüfen" teilnehmen soll, muss einmal mit S7-SCL V5.3 SP1 oder höher übersetzt worden sein.
- S7-SCL muss auf dem Gerät, auf dem Sie die Konsistenzprüfung durchführen, installiert sein.

Gehen Sie folgendermaßen vor

1. Öffnen Sie den SIMATIC Manager.
2. Wählen Sie den Ordner "Bausteine" an.
3. Wählen Sie den Menübefehl **Bearbeiten > Bausteinkonsistenz prüfen**.
4. Wählen Sie den Menübefehl **Ansicht > S7-SCL Quell-Referenzen anzeigen**.

Ergebnis

STEP 7 prüft die Zeitstempel und Schnittstellen aller Bausteine und zugehöriger Quellen im aktuellen Ordner und meldet folgende Konflikte:

- Zeitstempelkonflikte zwischen S7-SCL-Quellen und Bausteinen.
- Referenzen zwischen verschiedenen Bausteinen und daraus resultierende Schnittstellenkonflikte.

Wenn eine Inkonsistenz entdeckt wird, wird eine neue Übersetzung der zugehörigen Quelle angestoßen. Enthält die Quelle mehrere Bausteinquellen, werden alle aus dieser Quelle entstandenen Bausteine neu erzeugt. Bei der Übersetzung gelten die aktuell eingestellten Übersetzungsoptionen.

Hinweis

Weitere Informationen zu dieser Funktion erhalten Sie in der Hilfe zu STEP 7.

4.11 Anzeigen und Ändern von CPU-Eigenschaften

4.11.1 Anzeigen und Ändern des Betriebszustands der CPU

Sie können den aktuellen Betriebszustand einer CPU abfragen und modifizieren. Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Betriebszustand**.
2. Wählen Sie im folgenden Dialogfeld einen der folgenden Betriebszustände:
 - Neustart (Warmstart)
 - Kaltstart
 - Wiederanlauf
 - STOP



Warnung

Ein Verändern des Betriebszustands bei laufendem Anlagenbetrieb kann bei Funktionsstörungen oder Programmfehlern schwere Sach- und Personenschäden verursachen!

Vergewissern Sie sich, dass keine gefährlichen Zustände eintreten können, bevor Sie diese Funktion ausführen!

4.11.2 Anzeigen und Einstellen von Datum und Uhrzeit der CPU

Sie können die momentane Uhrzeit einer CPU abfragen und ändern. Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Uhrzeit stellen**.
2. Geben Sie im folgenden Dialogfeld das Datum und die Uhrzeit für die Echtzeituhr der CPU ein.

Falls die CPU nicht mit einer Echtzeituhr ausgestattet ist, werden im Dialogfeld "00:00:00" für die Uhrzeit und "00.00.00" für das Datum angezeigt. Sie können keine Änderungen vornehmen.

4.11.3 Anzeigen der CPU-Daten

Folgende Informationen über eine CPU können Sie sich anzeigen lassen:

- Systemfamilie, CPU-Typ, Bestellnummer und Ausgabestand der CPU.
- Größe des Arbeitsspeichers und des Ladespeichers und möglicher Maximalausbau des Ladespeichers.
- Anzahl und Adressbereich von Ein- und Ausgängen, Zeiten, Zählern und Merkern.
- Anzahl der Lokaldaten, mit denen diese CPU arbeiten kann.
- Anzeige, ob eine CPU mehrprozessorfähig ist oder nicht (CPU-abhängig).

Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Allgemein".

4.11.4 Auslesen des Diagnosepuffers der CPU

Durch das Auslesen des Diagnosepuffers können Sie die Ursache für den Betriebszustand STOP feststellen oder das Auftreten einzelner Diagnoseereignisse zurückverfolgen.

Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Diagnosepuffer".

4.11.5 Anzeigen/Komprimieren des Anwenderspeichers der CPU

Damit können Sie sich die Speicherauslastung der CPU anzeigen lassen und ggf. den Speicher der CPU reorganisieren. Dies ist notwendig, wenn die Größe des größten zusammenhängenden, freien Speicherbereichs nicht ausreicht, um ein neues Objekt vom PG aus in die CPU zu laden.

Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Speicher".

4.11.6 Anzeigen der Zykluszeit der CPU

Folgende Zeiten werden innerhalb der beiden parametrisierten Grenzwerte dargestellt:

- Dauer des längsten Zyklus seit dem letzten Übergang von STOP nach RUN
- Dauer des kürzesten Zyklus seit dem letzten Übergang von STOP nach RUN
- Dauer des letzten Zyklus

Wenn die Dauer des letzten Zyklus nahe an die Überwachungszeit heranreicht, so ist es möglich, dass diese überschritten wird und die CPU in Betriebszustand STOP geht. Die Zykluszeit kann sich zum Beispiel verlängern, wenn Sie Bausteine im Programmstatus testen. Voraussetzung für das Anzeigen der Zykluszeiten Ihres Programms ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Zykluszeit".

4.11.7 Anzeigen des Zeitsystems der CPU

Das Zeitsystem der Zentralbaugruppe (CPU) umfasst Informationen zur internen Uhr und zur Uhrzeitsynchronisation zwischen mehreren CPUs.

Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Zeitsystem".

4.11.8 Anzeigen der Bausteine in der CPU

Sie können sich für die CPU die online verfügbaren Bausteine anzeigen lassen.

Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Leistungsdaten/Bausteine".

4.11.9 Anzeigen der Informationen zur Kommunikation der CPU

Sie können sich für jede CPU online die Informationen über eingestellte und maximale Baudraten, Verbindungen sowie die Kommunikationslast anzeigen lassen.

Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Kommunikation".

4.11.10 Anzeigen der Stacks der CPU

Mit diesem Register können Sie für jede Zentralbaugruppe (CPU) online die Informationen über den Inhalt der Stacks anzeigen. Die CPU muss sich dabei im Betriebszustand STOP befinden oder einen Haltepunkt erreicht haben.

Das Anzeigen der Stacks ist eine wichtige Hilfe zum Finden von Fehlern, z.B. beim Testen Ihrer Bausteine. Ist die CPU dabei in den Betriebszustand STOP gegangen, können Sie sich im Unterbrechungs-Stack (U-Stack) die Unterbrechungsstelle mit den dort aktuellen Anzeigen und Akku-Inhalten anzeigen lassen, um so die Ursache (z.B. Programmierfehler) herauszufinden.

Voraussetzung ist eine bestehende Verbindung zur CPU.

Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Zielsystem > Baugruppenzustand**.
2. Wählen Sie im folgenden Dialogfeld das Register "Stacks".

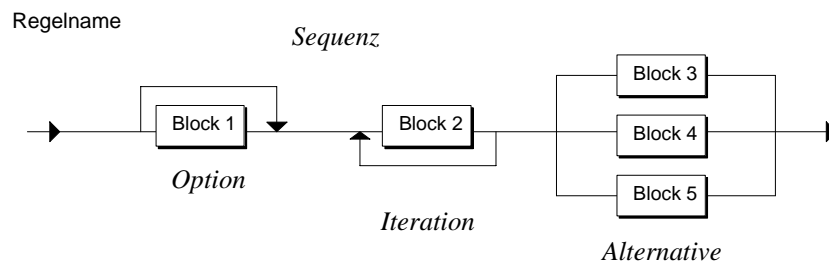
5 S7-SCL-Grundbegriffe

5.1 Interpretation der Syntaxdiagramme

Basis für die Sprachbeschreibung in den einzelnen Kapiteln sind Syntaxdiagramme. Sie geben Ihnen einen guten Einblick in den syntaktischen Aufbau von S7-SCL. Eine vollständige Zusammenstellung aller Diagramme mit den Sprachelementen finden Sie im Kapitel "Sprachbeschreibung".

Was ist ein Syntaxdiagramm?

Das Syntaxdiagramm ist eine grafische Darstellung der Struktur der Sprache. Die Struktur wird durch eine Folge von Regeln beschrieben. Dabei kann eine Regel auf bereits eingeführten Regeln aufbauen.

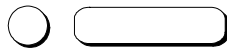


Das Syntaxdiagramm wird von links nach rechts gelesen. Dabei sind die folgenden Regelstrukturen zu beachten:

- Sequenz: Folge von Blöcken
- Option: Überspringbarer Zweig
- Iteration: Wiederholung von Zweigen
- Alternative: Verzweigung

Welche Arten von Blöcken gibt es?

Ein Block ist ein Grundelement oder ein Element, das wiederum aus Blöcken zusammengesetzt ist. Folgendes Bild zeigt die Symbolarten, die den Blöcken entsprechen:



Grundelement, das nicht weiter erklärt werden muss.
Hier handelt es sich um druckbare Zeichen und Spezialzeichen, Schlüsselwörter und vordefinierte Bezeichner.
Die Angaben zu diesen Blöcken sind unverändert zu übernehmen.



Zusammengesetztes Element, das durch weitere Syntaxdiagramme beschrieben wird.

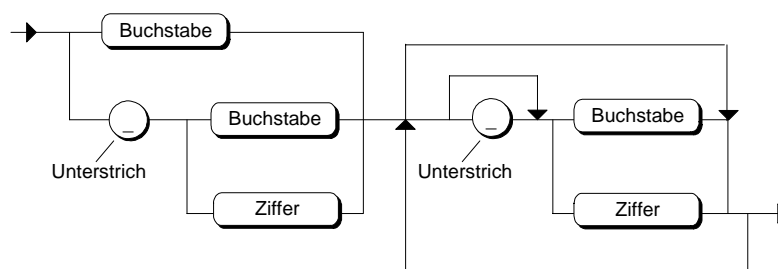
Was bedeutet Formatfreiheit?

Bei der Eingabe von Quelltexten sind sowohl die **syntaktischen Regeln** als auch **lexikalischen Regeln** zu beachten.

Die lexikalischen und syntaktischen Regeln sind im Kapitel "Sprachbeschreibung" detailliert beschrieben. Formatfreiheit bedeutet, dass Sie zwischen den Regelblöcken Formatierungszeichen wie Leerzeichen, Tabulatoren und Seitenwechsel sowie Kommentare einfügen können.

Bei den lexikalischen Regeln haben Sie keine Formatfreiheit. Wenn Sie eine lexikalische Regel anwenden, müssen Sie die Angaben unverändert übernehmen.

Lexikalische Regel



Gültige Beispiele nach dieser dargestellten Regel wären:

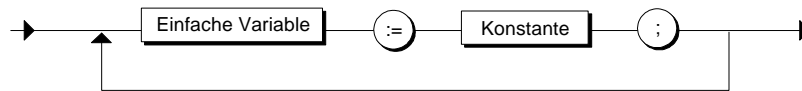
```
R_REGLER3
_A_FELD
_100_3_3_10
```

Ungültige Beispiele aus den oben genannten Gründen sind:

```
1_1AB
RR_20
*#AB
```

Syntaktische Regel

Bei syntaktischen Regeln haben Sie Formatfreiheit.



Gültige Beispiele nach dieser dargestellten Regel wären:

```
VARIABLE_1      := 100;      SCHALTER := FALSE;  
VARIABLE_2      := 3.2;
```

5.2 Zeichensatz

Buchstaben und Ziffern

S7-SCL benutzt aus dem Teilbereich des ASCII-Zeichensatzes:

- die (Klein- und Groß-) Buchstaben von A bis Z.
- die arabischen Ziffern 0 bis 9.
- Leerzeichen - das Leerzeichen selbst (ASCII-Wert 32) und alle Steuerzeichen (ASCII 0-31) einschließlich des Zeilenendezeichens (ASCII 13).

Sonstige Zeichen

Diese Zeichen haben in S7-SCL festgelegte Bedeutung:

+	-	*	/	=	<	>	[]	()
:	;	\$	#	"	'	{	}	%	.	,

Hinweis

Im Kapitel "Sprachbeschreibung" finden Sie eine detaillierte Auflistung aller verwendbaren Zeichen sowie Angaben zur Interpretation dieser Zeichen in S7-SCL.

5.3 Reservierte Wörter

Reservierte Wörter sind Schlüsselwörter, die Sie nur wie vorbestimmt benutzen dürfen. Eine Unterscheidung zwischen Groß- und Kleinschreibung findet nicht statt.

Schlüsselwörter in S7-SCL

AND	END_CASE	ORGANIZATION_BLOCK
ANY	END_CONST	POINTER
ARRAY	END_DATA_BLOCK	PROGRAM
AT	END_FOR	REAL
BEGIN	END_FUNCTION	REPEAT
BLOCK_DB	END_FUNCTION_BLOCK	RETURN
BLOCK_FB	END_IF	S5TIME
BLOCK_FC	END_LABEL	STRING
BLOCK_SDB	END_TYPE	STRUCT
BLOCK_SFB	END_ORGANIZATION_BLOCK	THEN
BLOCK_SFC	END_REPEAT	TIME
BOOL	END_STRUCT	TIMER
BY	END_VAR	TIME_OF_DAY
BYTE	END_WHILE	TO
CASE	ENO	TOD
CHAR	EXIT	TRUE
CONST	FALSE	TYPE
CONTINUE	FOR	VAR
COUNTER	FUNCTION	VAR_TEMP
DATA_BLOCK	FUNCTION_BLOCK	UNTIL
DATE	GOTO	VAR_INPUT
DATE_AND_TIME	IF	VAR_IN_OUT
DINT	INT	VAR_OUTPUT
DIV	LABEL	VOID
DO	MOD	WHILE
DT	NIL	WORD
DWORD	NOT	XOR
ELSE	OF	Namen der Standardfunktionen
ELSIF	OK	
EN	OR	

Schlüsselwörter von STEP 7

Darüber hinaus gelten die STEP 7-Schlüsselwörter als reserviert. Genauere Informationen hierzu entnehmen Sie bitte der Dokumentation zu STEP 7.

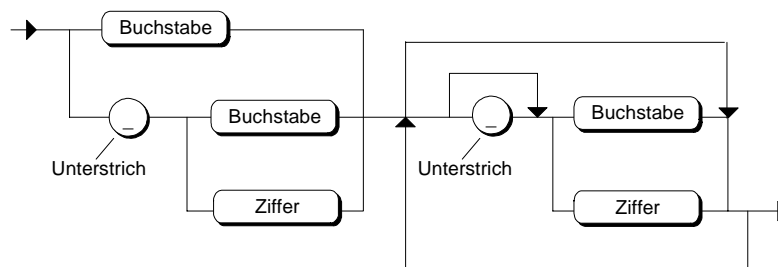
5.4 Bezeichner

Definition

Ein Bezeichner ist ein Name, den Sie für ein Sprachobjekt von S7-SCL, also eine Konstante, eine Variable oder einen Baustein selbst vergeben können.

Regeln

Bezeichner können aus maximal 24 Buchstaben (keine Umlaute) oder Ziffern in jeder beliebigen Reihenfolge zusammengesetzt sein, wobei nur das erste Zeichen ein Buchstabe oder ein Unterstrich sein muss. Sowohl Groß- als auch Kleinbuchstaben sind erlaubt. Eine Unterscheidung zwischen Groß- und Kleinschreibung findet jedoch nicht statt (AnNa und AnnA sind also zum Beispiel identisch).



Beispiele

Die folgenden Namen sind Beispiele für gültige Bezeichner.

X	y12
Summe	Temperatur
Namen	Flaeche
Regler	Tabelle

Die folgenden Namen sind aus den angegebenen Gründen keine gültigen Bezeichner.

4ter	//Das erste Zeichen muss ein Buchstabe oder //ein Unterstrich sein
Array	//ARRAY ist ein Schlüsselwort
S Wert	//Leerstellen sind nicht erlaubt (denken Sie //daran, dass eine Leerstelle ein Zeichen ist).

Hinweise

- Achten Sie darauf, dass der Name nicht schon durch Schlüsselwörter oder Standardbezeichner belegt ist.
- Wählen Sie am besten eindeutige und aussagekräftige Namen, die zur Verständlichkeit des Quelltextes beitragen.

5.5 Standardbezeichner

In S7-SCL sind eine Reihe von Bezeichnern bereits vordefiniert, für die deshalb der Name Standardbezeichner verwendet wird. Diese Standardbezeichner sind:

- die Baustein-Bezeichnungen,
- die Operandenkennzeichen für das Ansprechen von Speicherbereichen der CPU,
- die TIMER-Bezeichnungen, und
- die Zähler-Bezeichnungen.

5.6 Baustein-Bezeichnung

Definition

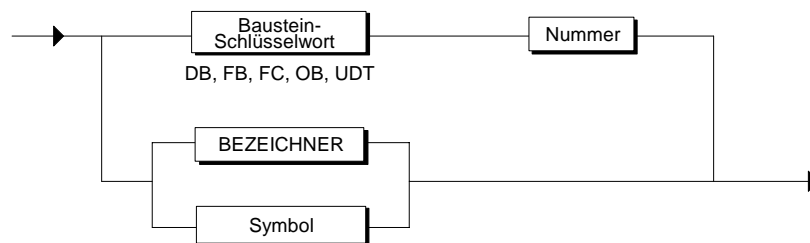
Diese Standardbezeichner werden für die absolute Adressierung von Bausteinen verwendet.

Regeln

Die Tabelle ist nach der deutschen Mnemonik sortiert, die entsprechende internationale Mnemonik wird dazu angegeben. Der Buchstabe x steht hier als Platzhalter für eine Nummer zwischen 0 und 65533 bzw. zwischen 0 und 65535 bei Timer und Counter.

Mnemonik (SIMATIC)	Mnemonik (IEC)	Kennzeichnet
DBx	DBx	Datenbaustein (Data-Block). Die Bausteinbezeichnung DB0 ist von S7-SCL reserviert.
FBx	FBx	Funktionsbaustein (Function-Block)
FCx	FCx	Funktion (Function)
OBx	OBx	Organisationsbaustein (Organization-Block)
SDBx	SDBx	Systemdatenbaustein (System-Data-Block)
SFCx	SFCx	Systemfunktion (System-Function)
SFBx	SFBx	Systemfunktionsbaustein (System-Function-Block)
Tx	Tx	Zeitglied (Timer)
UDTx	UDTx	Globaler bzw. anwenderdefinierter Datentyp (Userdefined Data-Type)
Zx	Cx	Zähler (Counter)

Für die Angabe der Bausteinbezeichnung haben Sie in S7-SCL mehrere Möglichkeiten. Als Nummer des Bausteins können Sie eine ganze Dezimalzahl angeben.



Beispiel

Gültige Bezeichnungen sind:

FB10
DB100
T141

5.7 Operandenkennzeichen

Definition

Die Speicherbereiche einer CPU können Sie von jeder Programmstelle aus mit ihrem Operandenkennzeichen ansprechen.

Regeln

Die Tabelle ist nach der deutschen Mnemonik sortiert, die entsprechende internationale Mnemonik wird dazu angegeben. Die Operandenkennzeichen für Datenbausteine gelten nur zusammen mit der Angabe des Datenbausteins.

Mnemonik (deutsch)	Mnemonik (internat.)	adressiert	Datentyp
Ax.y	Qx.y	Ausgang (über Prozessabbild)	Bit
ABx	QBx	Ausgang (über Prozessabbild)	Byte
ADx	QDx	Ausgang (über Prozessabbild)	Doppelwort
AWx	QWx	Ausgang (über Prozessabbild)	Wort
AXx.y	QXx.y	Ausgang (über Prozessabbild)	Bit
Dx.y	Dx.y	Datenbaustein	Bit
DBx	DBx	Datenbaustein	Byte
DDx	DDx	Datenbaustein	Doppelwort
DWx	DWx	Datenbaustein	Wort
DXx.y	DXx.y	Datenbaustein	Bit
Ex.y	Ix.y	Eingang (über Prozessabbild)	Bit
EBx	IBx	Eingang (über Prozessabbild)	Byte
EDx	IDx	Eingang (über Prozessabbild)	Doppelwort
EWx	IWx	Eingang (über Prozessabbild)	Wort
EXx.y	IXx.y	Eingang (über Prozessabbild)	Bit
Mx.y	Mx.y	Merker	Bit
MBx.y	MBx.y	Merker	Byte
MDx	MDx	Merker	Doppelwort
MWx.y	MWx	Merker	Wort
MXx	MXx	Merker	Bit
PABx	PQBx	Ausgang (Peripherie direkt)	Byte
PADx	PQDx	Ausgang (Peripherie direkt)	Doppelwort
PAWx	PQWx	Ausgang (Peripherie direkt)	Wort
PEBx	PIBx	Eingang (Peripherie direkt)	Byte
PEDx	PIDx	Eingang (Peripherie direkt)	Doppelwort
PEWx	PIWx	Eingang (Peripherie direkt)	Wort

x = Zahl zwischen 0 und 65535 (absolute Adresse)

y = Zahl zwischen 0 und 7 (Bitnummer)

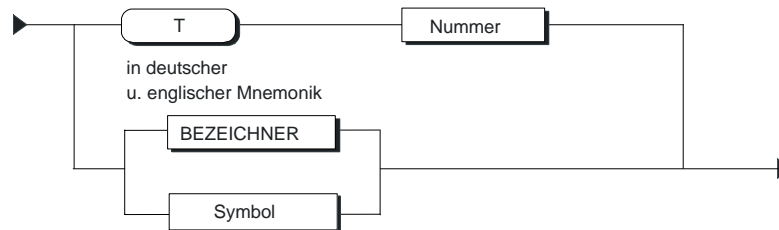
Beispiel

E1.0 MW10 PAW5 DB20.DW3

5.8 Timer-Bezeichnung

Regeln

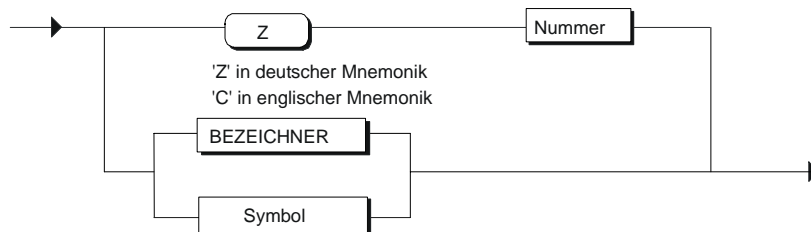
Für die Angabe eines Zeitglieds haben Sie in S7-SCL mehrere Möglichkeiten. Als Nummer des Zeitglieds können Sie eine ganze Dezimalzahl angeben.



5.9 Zähler-Bezeichnung

Regeln

Für die Angabe eines Zählers haben Sie in S7-SCL mehrere Möglichkeiten. Als Nummer des Zählers können Sie eine ganze Dezimalzahl angeben.



5.10 Zahlen

SCL kennt verschiedene Schreibweisen für Zahlen. Die folgenden Regeln gelten für alle Zahlen:

- Eine Zahl kann wahlweise ein Vorzeichen, einen Dezimalpunkt und einen Exponenten beinhalten.
- Kommata und Leerzeichen dürfen nicht innerhalb einer Zahl vorhanden sein.
- Zur optischen Trennung ist der Unterstrich () erlaubt.
- Der Zahl kann wahlweise ein plus (+) oder minus (-) vorangestellt werden. Falls kein Vorzeichen bei der Zahl steht, wird sie als positiv angenommen.
- Zahlen dürfen bestimmte Maximal- und Minimalwerte nicht über- bzw. unterschreiten.

Integerzahlen

Eine Integerzahl enthält weder einen Dezimalpunkt noch einen Exponenten. Somit ist eine Integerzahl einfach eine Folge von Ziffern, die wahlweise mit einem Vorzeichen beginnt. In S7-SCL sind 2 Integer-Typen realisiert, die jeweils unterschiedliche Wertebereiche haben, INT und DINT.

Einige gültige Integerzahlen :

0	1	+1	-1
743	-5280	600 00	-32 211

Die folgenden Integerzahlen sind aus den angeführten Gründen **falsch**:

123,456	Kommata sind nicht erlaubt.
36.	In einer Integerzahl darf kein Dezimalpunkt stehen.
10 20 30	Leerzeichen sind nicht erlaubt.

In S7-SCL können Sie Integerzahlen in unterschiedlichen Zahlensystemen darstellen. Das erfolgt durch Vorstellen eines Schlüsselwortes für das Zahlensystem. Dabei steht 2# für das Binärsystem, 8# für das Oktalsystem und 16# für das Hexadezimalsystem.

Gültige Integerzahlen für Dezimal 15:

2#1111 8#17 16#F

Realzahl

Eine Realzahl muss entweder einen Dezimalpunkt oder einen Exponenten (oder beides) enthalten. Ein Dezimalpunkt muss zwischen zwei Ziffern stehen. Somit kann eine Realzahl nicht mit einem Dezimalpunkt anfangen oder enden.

Einige gültige Realzahlen :

0.0	1.0	-0.2	827.602
50000.0	-0.000743	12.3	-315.0066

Die folgenden Realzahlen sind **falsch**:

1.	Auf beiden Seiten des Dezimalpunktes muss eine Ziffer stehen.
1,000.0	Kommata sind nicht erlaubt.
.3333	Auf beiden Seiten des Dezimalpunktes muss eine Ziffer stehen.

Ein Exponent kann enthalten sein, um die Lage des Dezimalpunktes festzulegen. Falls kein Dezimalpunkt vorhanden ist, wird angenommen, dass er auf der rechten Seite der Ziffer steht. Der Exponent selbst muss entweder eine positive oder negative Integerzahl sein. Die Basis 10 wird durch den Buchstaben E ersetzt.

Die Größe 3×10 Exponent 10 kann in S7-SCL durch folgende Realzahlen dargestellt werden:

3.0E+10	3.0E10	3e+10	3E10
0.3E+11	0.3e11	30.0E+9	30e9

Die folgenden Realzahlen sind **falsch**:

3.E+10	Auf beiden Seiten des Dezimalpunkts muss eine Ziffer stehen.
8e2.3	Der Exponent muss eine Integerzahl sein.
.333e-3	Auf beiden Seiten des Dezimalpunkts muss eine Ziffer stehen.
30 E10	Leerzeichen sind nicht erlaubt.

5.11 Zeichenketten

Definition

Eine Zeichenkette ist eine Folge von Zeichen (d.h. Buchstaben, Ziffern und Sonderzeichen), die in Anführungszeichen stehen.

Einige gültige Zeichenketten:

```
'ROT'           '76181 Karlsruhe'       '270-32-3456'
'DM19.95'      'Die richtige Antwort ist:'
```

Regeln

Spezielle Formatierungszeichen, das Anführungszeichen (') oder ein \$-Zeichen können Sie mit Fluchtsymbol \$ eingeben.

Quelltext	nach Kompilierung
'SIGNAL\$'ROT\$''	SIGNAL' ROT'
'50.0\$\$'	50.0\$
'WERT\$P'	WERT <i>Seitenumbruch</i>
'REG-\$L'	REG- <i>Zeilenumbruch</i>
'REGEL\$R	REGLER <i>Wagenrücklauf</i>
'SCHRITT\$T'	SCHRITT <i>Tabulator</i>

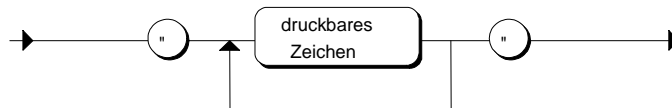
Für die nicht druckbaren Zeichen geben Sie die Ersatzdarstellung im Hexacode mit \$hh an, wobei hh stellvertretend für den hexadezimal ausgedrückten Wert des ASCII-Zeichens steht.

Zur Unterbrechung einer Zeichenkette (für Kommentare, die nicht ausgedruckt oder angezeigt werden sollen) stehen Ihnen in S7-SCL die Zeichen \$> und \$< für eine Stringunterbrechung zur Verfügung.

5.12 Symbol

Symbole können Sie in S7-SCL nach folgender Syntax angeben. Die Anführungszeichen sind nur dann erforderlich, wenn das Symbol nicht der Regel BEZEICHNER entspricht.

Syntax:



Hinweis

Achten Sie darauf, dass die symbolischen Namen eindeutig und nicht mit Schlüsselwörtern identisch sind.

5.13 Kommentarblock

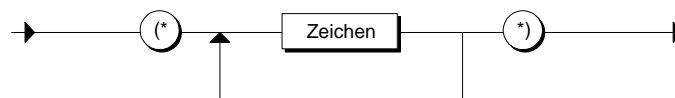
Regeln

- Der Kommentarblock kann sich über mehrere Zeilen erstrecken, und wird als Block mit "(" eingeleitet und durch ")" abgeschlossen.
- Die Schachtelung von Kommentarblöcken ist standardmäßig erlaubt. Sie können diese Einstellung jedoch ändern und die Schachtelung von Kommentarblöcken unzulässig machen.
- Ein Kommentar darf weder einen symbolischen Namen noch eine Konstante unterbrechen. Die Unterbrechung von Strings ist jedoch möglich.

Syntax

Der Kommentarblock wird durch das folgende Syntaxdiagramm formal dargestellt:

Kommentarblock



Beispiel

(* Dies ist ein Beispiel eines Kommentarblocks,
der sich über mehrere Zeilen erstrecken kann.*)

```
SCHALTER := 3 ;  
END_FUNCTION_BLOCK
```

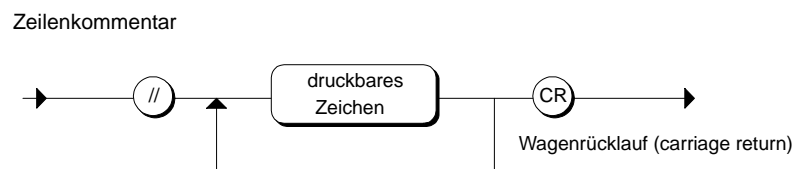
5.14 Zeilenkommentar

Regeln

- Der Zeilenkommentar wird mit "//" eingeleitet und erstreckt sich bis zum Ende der Zeile.
- Die Länge des Kommentars ist begrenzt auf max. 254 Zeichen einschließlich der Einleitungszeichen '//'.
• Ein Kommentar darf weder einen symbolischen Namen noch eine Konstante unterbrechen. Die Unterbrechung von Strings ist jedoch möglich.

Syntax

Der Zeilenkommentar wird durch das folgende Syntaxdiagramm formal dargestellt:



Beispiel

```
VAR  
    SCHALTER : INT ; // Zeilenkommentar  
END_VAR
```

Hinweise

- Kommentare innerhalb des Vereinbarungsteils, die mit // beginnen, werden in die Schnittstelle des Bausteins übernommen, und können somit auch im KOP/AWL/FUP-Editor angezeigt werden.
 - Die druckbaren Zeichen entnehmen Sie bitte dem Kapitel "Sprachbeschreibung".
 - Innerhalb des Zeilenkommentars sind die Zeichenpaare "(" und ")" bedeutungslos.
-

5.15 Variablen

Ein Bezeichner, dessen Wert während der Programmdurchführung geändert werden kann, wird Variable genannt. Jede Variable muss einzeln erklärt (d.h. vereinbart) werden, bevor sie innerhalb eines Codebausteins oder Datenbausteins benutzt werden kann. Die Variablenvereinbarung legt fest, dass ein Bezeichner eine Variable ist (und keine Konstante etc.) und spezifiziert durch die Zuordnung zum Datentyp den Variablentyp.

Je nach Gültigkeit der Variablen wird unterschieden zwischen:

- Lokaldaten
- Globalen Anwenderdaten
- erlaubten vordefinierten Variablen (Speicherbereichen einer CPU)

Lokaldaten

Lokaldaten sind Daten, die innerhalb eines Codebausteins (FC, FB, OB) vereinbart werden und nur für diesen Codebaustein Gültigkeit haben. Im Einzelnen sind das:

Variable	Bedeutung
Statische Variablen	Eine statische Variable ist eine lokale Variable, deren Wert über alle Bausteindurchläufe hinweg erhalten bleibt (Bausteingedächtnis). Sie dient der Speicherung von Werten eines Funktionsbausteins.
Temporäre Variablen	Temporäre Variablen gehören lokal zu einem Codebaustein und belegen keinen statischen Speicherbereich. Ihr Wert bleibt nur während eines Bausteinablaufs erhalten. Auf temporäre Variablen kann außerhalb des Bausteins, in dem die Variablen deklariert wurden, nicht zugegriffen werden.
Bausteinparameter	Bausteinparameter sind formale Parameter eines Funktionsbausteins oder einer Funktion. Es sind lokale Variablen die dazu dienen, die beim Aufruf angegebenen aktuellen Parameter zu übergeben.

Globale Anwenderdaten

Globale Anwenderdaten sind Daten bzw. Datenbereiche die Sie von jeder Programmstelle aus nutzen können. Dazu müssen Sie Datenbausteine (DB) erstellen.

Wenn Sie einen DB erstellen, legen Sie in einer Strukturvereinbarung seinen Aufbau fest. Anstelle einer Strukturvereinbarung kann auch ein anwenderdefinierter Datentyp (UDT) verwendet werden. Die Reihenfolge, in der Sie die Strukturkomponente angeben, bestimmt die Reihenfolge der Daten in dem DB.

Speicherbereiche einer CPU

Auf die Speicherbereiche einer CPU können Sie über die Operandenkennzeichen direkt von jeder Programmstelle aus zugreifen, ohne diese Variablen vereinbaren zu müssen.

Sie haben darüber hinaus immer die Möglichkeit, diese Datenbereiche auch symbolisch anzusprechen. Die Symbolzuordnung erfolgt in diesem Fall global über die Symboltabelle in STEP 7.

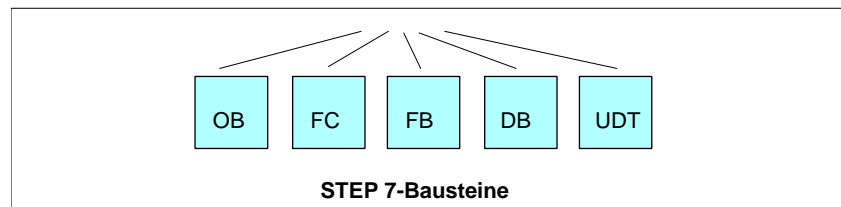
6 S7-SCL-Programmstruktur

6.1 Bausteine in S7-SCL-Quellen

In einer S7-SCL-Quelldatei können Sie 1 bis n Bausteine programmieren. Die STEP 7-Bausteine sind durch ihre Funktion, ihre Struktur oder ihren Verwendungszweck abgegrenzte Teile eines Anwenderprogramms.

Bausteinarten

Folgende Bausteine stehen zur Verfügung:



Vorgefertigte Bausteine

Nicht jede Funktion müssen Sie selbst programmieren. Sie können auch auf vorgefertigte Bausteine zurückgreifen. Sie sind im Betriebssystem der Zentralbaugruppen oder in Bibliotheken (*S7lib*) des STEP 7-Basispakets vorhanden und können z. B. für die Programmierung von Kommunikationsfunktionen genutzt werden.

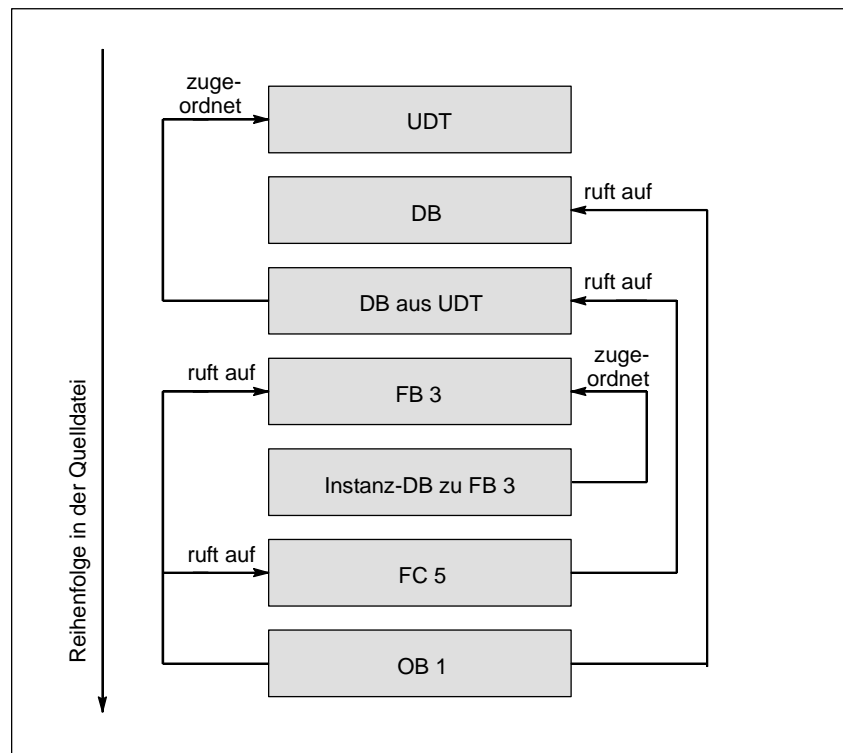
6.2 Reihenfolge der Bausteine

Grundsätzlich gilt die Regel:

Aufgerufene Bausteine stehen vor den aufrufenden Bausteinen.

Das heißt im Einzelnen:

- Anwenderdefinierte Datentypen (UDT) stehen vor den Bausteinen, in denen sie verwendet werden.
- Datenbausteine mit einem zugeordneten anwenderdefiniertem Datentyp (UDT) stehen hinter dem UDT.
- Datenbausteine, auf die von allen Codebausteinen aus zugegriffen werden kann, stehen vor den Bausteinen, aus denen sie aufgerufen werden.
- Datenbausteine mit zugeordnetem Funktionsbaustein stehen hinter dem Funktionsbaustein.
- Der Organisationsbaustein OB 1, der andere Bausteine aufruft, steht zuletzt. Bausteine, die wiederum von den aus OB 1 aufgerufenen Bausteinen aufgerufen werden, müssen vor diesen stehen.
- Bausteine, die Sie in der Quelldatei aufrufen, aber nicht in derselben Quelldatei programmieren, müssen bereits zum Zeitpunkt der Dateiübersetzung im entsprechenden Anwenderprogramm vorhanden sein.
- Neben den Bausteinen können S7-SCL-Quellen auch Angaben über die Compilereinstellungen enthalten, mit denen die einzelnen Bausteine übersetzt werden sollen. Die Compileroptionen stehen außerhalb der Bausteingrenzen.



6.3 Allgemeiner Aufbau eines Bausteins

Ein Baustein besteht aus folgenden Bereichen:

- Bausteinanfang, gekennzeichnet durch ein Schlüsselwort und eine Bausteinnummer oder einen symbolischen Bausteinnamen, also z.B. "ORGANIZATION_BLOCK OB1" für einen Organisationsbaustein. Bei Funktionen wird zusätzlich der Funktionstyp angegeben. Dieser bestimmt den Datentyp des Rückgabewertes. Soll kein Wert zurückgegeben werden, ist das Schlüsselwort VOID anzugeben.
- Optionaler Bausteintitel, eingeleitet mit dem Schlüsselwort "TITLE ="
- Optionaler Bausteinkommentar. Der Bausteinkommentar kann sich über mehrere Zeilen erstrecken, wobei jede Zeile mit "///" beginnt.
- Eintragung der Bausteinattribute (optional)
- Eintragung der Systemattribute für Bausteine (optional)
- Vereinbarungsteil (unterschiedlich je nach Bausteinart)
- Anweisungsteil in Codebausteinen bzw. Zuweisung von Aktualwerten in Datenbausteinen (optional)
- In Codebausteinen: Anweisungen
- Bausteinende, gekennzeichnet durch END_ORGANIZATION_BLOCK, END_FUNCTION_BLOCK oder END_FUNCTION

6.4 Bausteinanfang und -ende

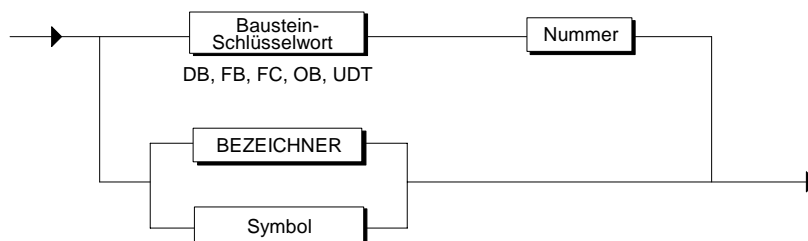
Der Quelltext für einen einzelnen Baustein wird abhängig von der Bausteinart mit einem Standardbezeichner für den Anfang des Bausteins und der Bausteinbezeichnung eingeleitet. Abgeschlossen wird er mit einem Standardbezeichner für das Ende des Bausteins.

Die Syntax für die verschiedenen Bausteinarten zeigt folgende Tabelle:

Bezeichnung	Bausteinart	Syntax
Funktionsbaustein	FB	FUNCTION_BLOCK fb_name ... END_FUNCTION_BLOCK
Funktion	FC	FUNCTION fc_name : funktionstyp ... END_FUNCTION
Organisationsbaustein	OB	ORGANIZATION_BLOCK ob_name ... END_ORGANIZATION_BLOCK
Datenbaustein	DB	DATA_BLOCK db_name ... END_DATA_BLOCK
Globaler Datentyp	UDT	TYPE udt_name ... END_TYPE

Bausteinbezeichnung

In der Tabelle steht *xx_name* für die Bausteinbezeichnung nach der folgenden Syntax:



Die Bausteinnummer kann einen Wert von 0 bis 65533 einnehmen, der Datenbaustein-Bezeichner DB0 ist jedoch reserviert.

Beachten Sie auch, dass Sie einen Bezeichner oder ein Symbol in der Symboltabelle von STEP 7 definieren müssen.

Beispiel

```
FUNCTION_BLOCK FB10  
FUNCTION_BLOCK Reglerbaustein  
FUNCTION_BLOCK "Regler.B1&U2"
```


6.5 Bausteinattribute

Definition

Ein Bausteinattribut ist eine Bausteineigenschaft, die Sie z.B. zur Angabe der Bausteinüberschrift, der Version, des Bausteinschutzes oder des Autors verwenden können. Die Eigenschaften können Sie sich bei der Auswahl von Bausteinen für Ihren Anwendungsfall in einem Eigenschaftsfenster von STEP 7 anzeigen lassen.

Folgende Attribute können Sie vergeben:

Schlüsselwort / Attribut	Bedeutung	Beispiele
TITLE='druckbares Zeichen'	Überschrift des Bausteins	TITLE='SORTIEREN'
VERSION : 'Dezimalziffernfolge. Dezimalziffernfolge'	Versionsnummer des Bausteins (0..15) Hinweis: Bei Datenbausteinen (DB) wird das Attribut VERSION nicht in Hochkommata angegeben.	VERSION : '3.1' //Bei DB: VERSION : 3.1
KNOW_HOW_PROTECT	Bausteinschutz; ein Baustein, der mit dieser Option übersetzt wurde, lässt sich mit STEP 7 nicht öffnen.	KNOW_HOW_PROTECT
AUTHOR :	Name des Autors: Firmenname, Abteilungsname od. andere Namen (BEZEICHNER und 'druckbare Zeichen')	AUTHOR : Siemens AUTHOR : 'A&D AS'
NAME :	Bausteinname (BEZEICHNER und 'druckbare Zeichen')	NAME : PID NAME : 'A&D AS'
FAMILY :	Name der Bausteinfamilie: z.B. Motoren. Speichert den Baustein in einer Bausteingruppe, sodass er schneller wiedergefunden werden kann (BEZEICHNER und 'druckbare Zeichen').	FAMILY : Beispiel FAMILY : 'A&D AS'

Regeln

- Die Bausteinattribute vereinbaren Sie mit Hilfe von Schlüsselwörtern direkt nach der Anweisung für den Bausteinanfang.
- Die Bezeichner dürfen maximal 8 Zeichen lang sein.

Die Eingabe von Bausteinattributen richtet sich nach folgender Syntax:

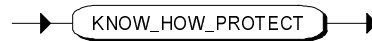
Überschrift



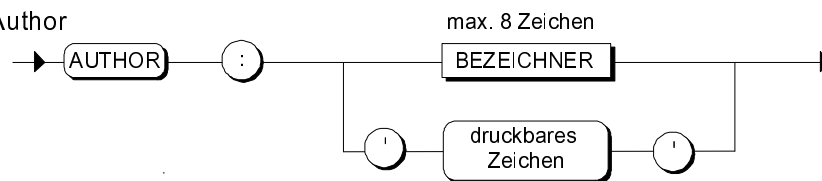
Version



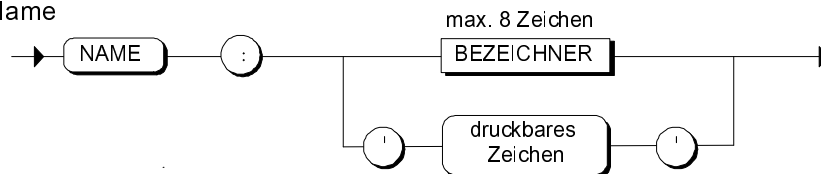
Bausteinschutz



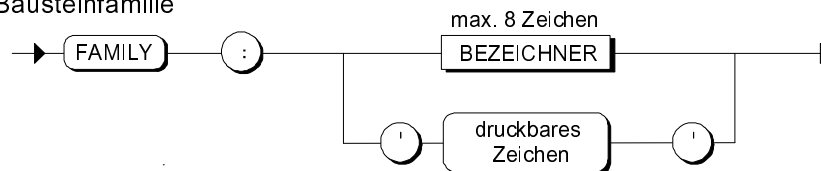
Author



Name



Bausteinfamilie



Beispiele

```
FUNCTION_BLOCK FB10
TITLE = 'Mittelwert'
VERSION : '2.1'
KNOW_HOW_PROTECT
AUTHOR : AUT_1
```

6.6 Bausteinkommentar

Kommentare zum gesamten Baustein geben Sie im Bausteinkopf nach der Zeile "TITLE:" ein. Wählen Sie hierzu die Zeilenkommentar-Notation. Wenn sich der Kommentar über mehrere Zeilen erstreckt, beginnt jede Zeile mit //.

Der Bausteinkommentar wird z.B. im Eigenschaftsfenster des Bausteins im SIMATIC Manager oder im KOP/AWL/FUP-Editor angezeigt.

Beispiel

```
FUNCTION_BLOCK FB15

TITLE=MEIN_BAUSTEIN
//Dies ist ein Bausteinkommentar.

//Er wird als Folge von Zeilenkommentaren eingegeben
//und kann z.B. im SIMATIC Manager angezeigt werden.

AUTHOR...

FAMILY...
```

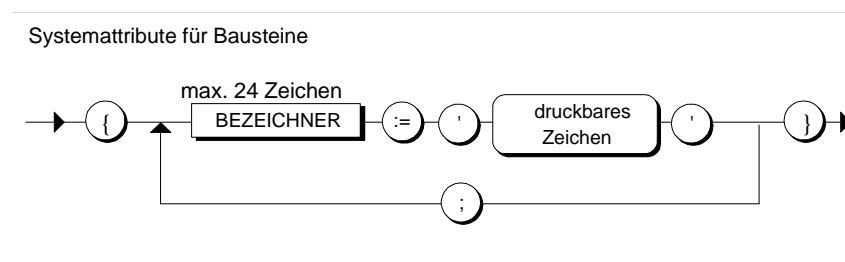
6.7 Systemattribute für Bausteine

Definition

Systemattribute sind applikationsübergreifende Attribute im Leitsystem. Systemattribute für Bausteine gelten für den gesamten Baustein.

Regeln

- Systemattribute vereinbaren Sie direkt nach der Anweisung für den Bausteinanfang.
- Die Eingabe richtet sich nach folgender Syntax:



Beispiele

```

FUNCTION_BLOCK FB10
{S7_m_c := 'true' ;
S7_blockview := 'big'}
  
```

6.8 Vereinbarungsteil

Definition

Der Vereinbarungsteil dient zur Vereinbarung der lokalen Variablen, Parameter, Konstanten und Sprungmarken.

- Die lokalen Variablen, Parameter, Konstanten und Sprungmarken, die nur innerhalb eines Bausteins Gültigkeit haben sollen, definieren Sie im Vereinbarungsteil des Codebausteins.
- Die Datenbereiche, die von jedem Codebaustein aus ansprechbar sein sollen, definieren Sie im Vereinbarungsteil von Datenbausteinen.
- Im Vereinbarungsteil eines UDT legen Sie einen anwenderdefinierten Datentyp fest.

Aufbau

Ein Vereinbarungsteil gliedert sich in unterschiedliche Vereinbarungsblöcke, die jeweils durch ein eigenes Schlüsselwortpaar gekennzeichnet sind. Jeder Block enthält eine Deklarationsliste für gleichartige Daten. Die Reihenfolge dieser Blöcke ist beliebig. Folgende Tabelle zeigt die möglichen Vereinbarungsblöcke:

Daten	Syntax	FB	FC	OB	DB	UDT
Konstanten	CONST Deklarationsliste END CONST	X	X	X		
Sprungmarken	LABEL Deklarationsliste END LABEL	X	X	X		
Temporäre Variable	VAR_TEMP Deklarationsliste END VAR	X	X	X		
Statische Variable	VAR Deklarationsliste END VAR	X	X *)		X **)	X **)
Eingangsparameter	VAR_INPUT Deklarationsliste END VAR	X	X			
Ausgangsparameter	VAR_OUTPUT Deklarationsliste END VAR	X	X			
Durchgangsparameter	VAR_IN_OUT Deklarationsliste END VAR	X	X			

*) Die Vereinbarung von Variablen innerhalb des Schlüsselwortpaars VAR und END_VAR ist in Funktionen zwar erlaubt, die Vereinbarungen werden aber beim Übersetzen einer Quelle in den temporären Bereich verschoben.

**) In DB und UDT wird das Schlüsselwort VAR und END_VAR durch STRUCT und END_STRUCT ersetzt.

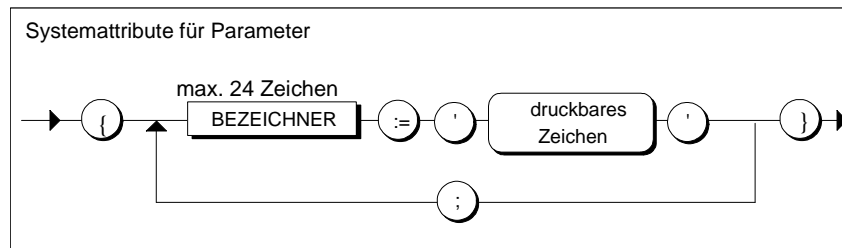
6.9 Systemattribute für Parameter

Definition

Systemattribute sind applikationsübergreifende Attribute im Leitsystem. Sie dienen z. B. für die Meldungs- oder Verbindungsprojektierung. Systemattribute für Parameter gelten jeweils nur für den einzelnen projektierten Parameter. Sie können Eingangs-, Ausgangs- und Durchgangsparemtern Systemattribute zuweisen.

Regeln

- Sie vergeben die Systemattribute für Parameter in den Vereinbarungsblocken Eingangsparemeter, Ausgangsparemeter bzw. Durchgangsparemeter.
- Ein Bezeichner darf max. 24 Zeichen lang sein.
- Die Eingabe richtet sich nach folgender Syntax:



Beispiel

```

VAR_INPUT
    in1    {S7_server:='alarm_archiv';
           S7_a_type:='ar_send'}: DWORD ;
END_VAR
  
```

Die Hilfe zu Systemattributen können Sie aus der S7-SCL Online-Dokumentation aufrufen. Wählen Sie hierzu das Kapitel "Aufruf von Referenzhilfen".

6.10 Anweisungsteil

Definition

Der Anweisungsteil beinhaltet Anweisungen, die nach dem Aufruf eines Codebausteins zur Ausführung kommen sollen. Diese Anweisungen dienen zur Verarbeitung von Daten oder Adressen.

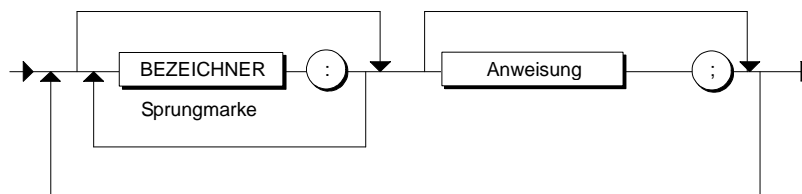
Der Anweisungsteil von Datenbausteinen enthält Anweisungen zur Vorbesetzung seiner Variablen.

Regeln

- Der Anweisungsteil kann optional mit dem Schlüsselwort BEGIN eingeleitet werden. Bei Datenbausteinen ist BEGIN zwingend erforderlich. Der Anweisungsteil endet mit dem Schlüsselwort für das Bausteinende.
- Jede Anweisung endet mit einem Semikolon.
- Alle im Anweisungsteil verwendeten Bezeichner müssen zuvor vereinbart werden.
- Vor jeder Anweisung können optional Sprungmarken stehen

Die Eingabe richtet sich nach folgender Syntax:

Anweisungsteil



Beispiel

```

BEGIN
    ANFANGSWERT      := 0 ;
    ENDWERT          := 200 ;
    .
    .
    SPEICHERN:      ERGEBNIS      := SOLLWERT ;
    .
    .
END_FUNCTION_BLOCK

```


6.11 Anweisungen

Definition

Eine Anweisung ist die kleinste selbstständige Einheit des Anwenderprogramms. Sie stellt eine Arbeitsvorschrift für den Prozessor dar.

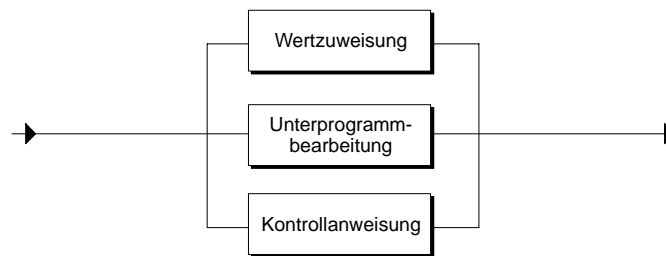
S7-SCL kennt folgende Arten von Anweisungen:

- Wertzuweisungen, die dazu dienen, einer Variablen einen Wert, das Ergebnis eines Ausdrucks, oder den Wert einer anderen Variablen zuzuweisen.
- Kontrollanweisungen, die dazu dienen, Anweisungen oder Gruppen von Anweisungen zu wiederholen oder innerhalb eines Programms zu verzweigen.
- Unterprogrammbearbeitungen, die zum Aufrufen von Funktionen und Funktionsbausteinen dienen.

Regeln

Die Eingabe richtet sich nach folgender Syntax:

Anweisung



Beispiel

Die folgenden Beispiele sollen die verschiedenen Varianten der Anweisungen veranschaulichen:

```
// Beispiel für eine Wertzuweisung
MESSWERT:= 0 ;
```

```
// Beispiel für eine Unterprogrammbearbeitung
FB1.DB11 (UEBERGABE:= 10) ;
```

```
// Beispiel für eine Kontrollanweisung
WHILE ZAEHLER < 10 DO..
.
.
END_WHILE;
```

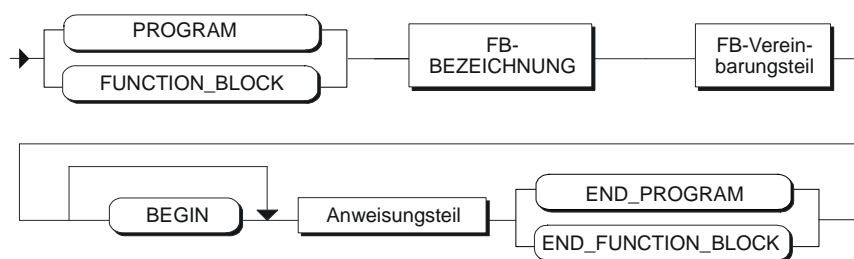
6.12 Aufbau eines Funktionsbausteins (FB)

Definition

Ein Funktionsbaustein (FB) ist ein Codebaustein, der einen Teil eines Programms enthält und über einen zugeordneten Speicherbereich verfügt. Immer wenn ein FB aufgerufen wird, muss ihm ein Instanz-DB zugeordnet werden. Den Aufbau dieses Instanz-DB bestimmen Sie mit der Definition des FB-Vereinbarungsteils.

Syntax

Funktionsbaustein



FB-Bezeichnung

Geben Sie nach dem Schlüsselwort `FUNCTION_BLOCK` bzw. `PROGRAM` als FB-Bezeichnung das Schlüsselwort `FB` und dahinter die Bausteinnummer oder den symbolischen Namen des FB an. Die Bausteinnummer kann einen Wert von 0 bis 65533 einnehmen.

Beispiele:

```
FUNCTION_BLOCK FB10
FUNCTION_BLOCK MOTOR1
```

FB-Vereinbarungsteil

Der FB-Vereinbarungsteil dient zur Festlegung der bausteinspezifischen Daten. Die zulässigen Vereinbarungsböcke entnehmen Sie bitte dem Kapitel "Vereinbarungsteil". Beachten Sie, daß der Vereinbarungsteil auch den Aufbau des zugeordneten Instanz-DB bestimmt.

Beispiel

Folgendes Beispiel zeigt den Quellcode für einen Funktionsbaustein. Die Eingangs- und Ausgangsparameter (hier V1, V2) sind hier mit Anfangswerten vorbelegt.

```
FUNCTION_BLOCK FB11
VAR_INPUT
    V1 : INT := 7 ;
END_VAR
VAR_OUTPUT
    V2 : REAL ;
END_VAR
VAR
    FX1, FX2, FY1, FY2 : REAL ;

END_VAR
BEGIN
    IF V1 = 7 THEN
        FX1 := 1.5 ;
        FX2 := 2.3 ;
        FY1 := 3.1 ;
        FY2 := 5.4 ;
        //Aufruf der Funktion FC11 mit
Parameterversorgung
        //durch die statischen Variablen.
        V2 := FC11 (X1:= FX1, X2 := FX2, Y1 := FY1,
Y2 := FY2) ;
    END_IF ;
END_FUNCTION_BLOCK
```

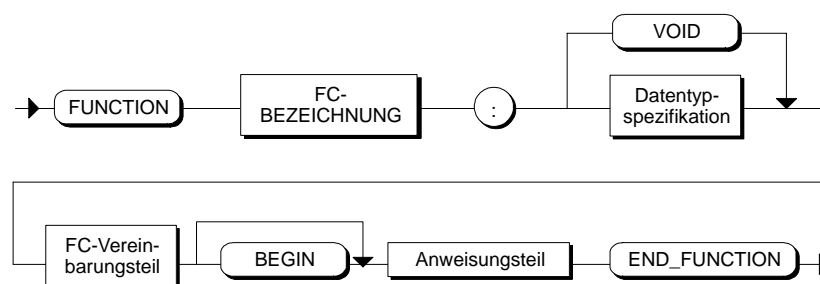
6.13 Aufbau einer Funktion (FC)

Definition

Eine Funktion (FC) ist ein Codebaustein, dem kein eigener Speicherbereich zugeordnet ist. Sie benötigt keinen Instanz-DB. Im Gegensatz zu einem FB kann eine Funktion ein Funktionsergebnis (Rückgabewert) an den Aufrufpunkt zurückliefern. Die Funktion kann daher wie eine Variable in einem Ausdruck verwendet werden. Funktionen vom Typ VOID haben keinen Rückgabewert.

Syntax

Funktion



FC-Bezeichnung

Geben Sie nach dem Schlüsselwort "FUNCTION" als FC-Bezeichnung das Schlüsselwort FC und dahinter die Bausteinnummer oder den symbolischen Namen des FC an. Die Bausteinnummer kann einen Wert von 0 bis 65533 einnehmen.

Beispiel:

```

FUNCTION FC17 : REAL
FUNCTION FC17 : VOID
  
```

Datentypspezifikation

Die Datentypspezifikation bestimmt den Datentyp des Rückgabewerts. Hierbei sind mit Ausnahme der Datentypen STRUCT und ARRAY alle Datentypen zulässig. Die Angabe des Datentyps entfällt, wenn mit VOID auf den Rückgabewert verzichtet wird.

FC-Vereinbarungsteil

Der FC-Vereinbarungsteil dient zur Deklaration der lokalen Daten (Temporäre Variable, Eingangsparameter, Ausgangsparameter, Durchgangsparameter, Konstanten, Sprungmarken).

FC-Anweisungsteil

Im Anweisungsteil muss dem Funktionsnamen das Funktionsergebnis zugeordnet werden. Bei Funktionen mit dem Funktionstyp VOID entfällt diese Zuweisung. Eine gültige Anweisung innerhalb einer Funktion mit der Bezeichnung FC31 ist z.B.:

```
FC31:= WERT;
```

Beispiel

```
FUNCTION FC11: REAL
VAR_INPUT
    x1: REAL ;
    x2: REAL ;
    x3: REAL ;
    x4: REAL ;
END_VAR
VAR_OUTPUT
    Q2: REAL ;
END_VAR
BEGIN
    // Rückgabe des Funktionswerts
    FC11:= SQRT( (x2 - x1)**2 + (x4 - x3) **2 ) ;
    Q2:= x1 ;
END_FUNCTION
```

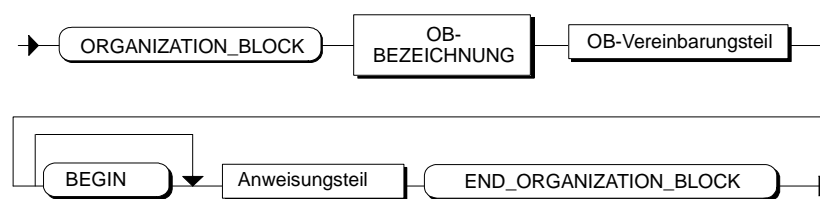
6.14 Aufbau eines Organisationsbausteins (OB)

Definition

Der Organisationsbaustein ist wie ein FB oder FC Teil des Anwenderprogramms und wird vom Betriebssystem zyklisch oder bei bestimmten Ereignissen aufgerufen. Er bildet die Schnittstelle zwischen Anwenderprogramm und Betriebssystem.

Syntax

Organisationsbaustein



OB-Bezeichnung

Geben Sie nach dem Schlüsselwort "ORGANIZATION_BLOCK" als OB-Bezeichnung das Schlüsselwort OB und dahinter die Bausteinnummer oder den symbolischen Namen des OB an. Die Bausteinnummer kann einen Wert von 1 bis 65533 einnehmen.

Beispiele:

```
ORGANIZATION_BLOCK OB1
ORGANIZATION_BLOCK ALARM
```

OB-Vereinbarungsteil

Der OB-Vereinbarungsteil dient zur Deklaration der lokalen Daten (temporäre Variablen, Konstanten, Sprungmarken).

Jeder OB benötigt für seinen Ablauf grundsätzlich 20 Byte Lokaldaten für das Betriebssystem (BESY). Sie müssen dafür ein Feld mit einem beliebigen Bezeichner vereinbaren. Wenn Sie die Bausteinvorlage für OB einfügen, ist diese Vereinbarung bereits enthalten.

Beispiel

```
ORGANIZATION_BLOCK OB1
VAR_TEMP
    HEADER : ARRAY [1..20] OF BYTE ; //20 Byte für Besy
END_VAR
BEGIN
    FB17.DB10 (V1 := 7) ;
END_ORGANIZATION_BLOCK
```

6.15 Aufbau eines Datenbausteins (DB)

Definition

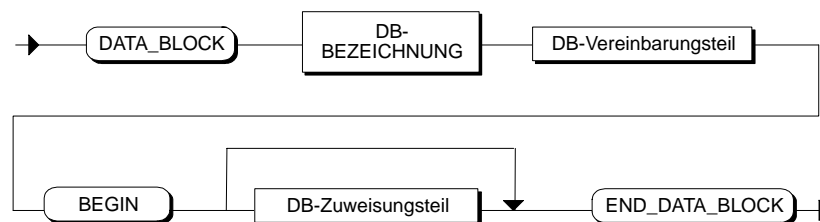
Globale anwenderspezifische Daten, auf die alle Bausteine eines Programms zugreifen sollen, werden in Datenbausteinen abgelegt. Jeder FB, FC oder OB kann diese Datenbausteine lesen oder beschreiben.

Es gibt zwei Typen von Datenbausteinen:

- **Datenbausteine**
Datenbausteine, auf die alle Codebausteine des CPU-Programms zugreifen können. Jeder FB, FC oder OB kann die in diesen Datenbausteinen enthaltenen Daten lesen bzw. schreiben.
- **Datenbausteine, die einem FB zugeordnet sind (Instanz-DB)**
Instanz-Datenbausteine sind Datenbausteine, die einem bestimmten Funktionsbaustein (FB) zugeordnet sind. Sie enthalten die Lokaldaten für diesen zugeordneten Funktionsbaustein. Diese Datenbausteine werden vom S7-SCL-Compiler automatisch erzeugt, sobald der FB im Anwenderprogramm aufgerufen wird.

Syntax

Datenbaustein



DB-Bezeichnung

Geben Sie nach dem Schlüsselwort "DATA_BLOCK" als DB-Bezeichnung das Schlüsselwort DB und dahinter die Bausteinnummer oder den symbolischen Namen des DB an. Die Bausteinnummer kann einen Wert von 1 bis 65533 einnehmen.

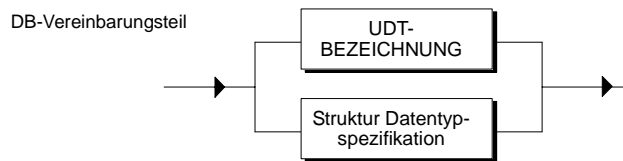
Beispiele:

```
DATA_BLOCK DB20
DATA_BLOCK MESSBEREICH
```

DB-Vereinbarungsteil

Im DB-Vereinbarungsteil definieren Sie die Datenstruktur des DB. Dafür gibt es zwei Möglichkeiten:

- **Durch Zuordnung eines Anwenderdefinierten Datentyps**
Hier können Sie die Bezeichnung eines zuvor im Programm definierten anwenderdefinierten Datentyps angeben. Der Datenbaustein nimmt dann die Struktur dieses UDT an. Anfangswerte für die Variablen können Sie im Zuweisungsteil des DB vergeben.
- **Durch Definition eines STRUCT-Datentyps**
Innerhalb der STRUCT-Datentypspezifikation legen Sie den Datentyp für jede im DB zu speichernde Variable und ggf. Anfangswerte fest.



Beispiel:

```
DATA_BLOCK DB20
    STRUCT // Vereinbarungsteil
        WERT:ARRAY [1..100] OF INT;
    END_STRUCT

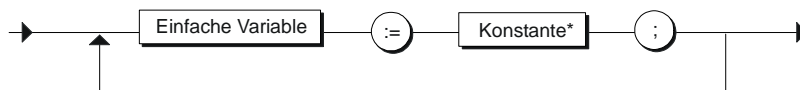
BEGIN // Anfang Zuweisungsteil
:
END_DATA_BLOCK // Ende Datenbaustein
```

DB-Zuweisungsteil

Im Zuweisungsteil können Sie die Daten, die Sie im Vereinbarungsteil vereinbart haben, für Ihren speziellen Anwendungsfall mit einzelnen DB-spezifischen Werten anpassen.

Der Zuweisungsteil wird mit dem Schlüsselwort BEGIN eingeleitet und besteht dann aus einer Folge von Wertzuweisungen.

DB-Zuweisungsteil



* in AWL-Schreibweise

Bei der Vergabe von Anfangswerten (Initialisierung), der Angabe von Attributen und der Angabe von Kommentaren gilt die Syntax von AWL. Informationen über die Schreibweisen der Konstanten, Attribute und Kommentare können Sie der Online-Hilfe zu AWL oder der STEP 7-Dokumentation entnehmen.

Beispiel

```
// Datenbaustein mit zugeordnetem STRUCT-Datentyp
DATA_BLOCK DB10
    STRUCT // Datenvereinbarung mit Vorbelegung
        WERT :      ARRAY [1..100] OF INT := 100 (1) ;
        SCHALTER :   BOOL      := TRUE ;
        S_WORT :     WORD      := W#16#FFAA ;
        S_BYTE :     BYTE      := B#16#FF ;
        S_TIME :     S5TIME     := S5T#1h30m10s
    ;
    END_STRUCT

BEGIN // Zuweisungsteil
    // Wertzuweisung für bestimmte Feldelemente
    WERT [1] := 5;
    WERT [5] := -1 ;

END_DATA_BLOCK

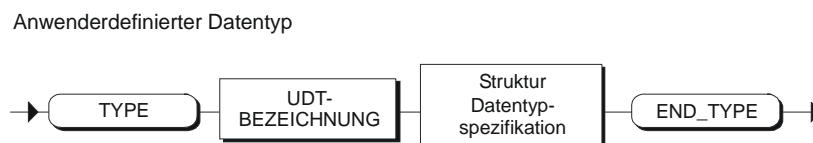
// Datenbaustein mit zugeordnetem anwenderdefiniertem
Datentyp
DATA_BLOCK DB11
    UDT 51
BEGIN
END_DATA_BLOCK
```

6.16 Aufbau eines anwenderdefinierten Datentyps

Anwenderdefinierte Datentypen UDT sind von Ihnen erzeugte spezielle Datenstrukturen. Da anwenderdefinierte Datentypen einen Namen haben, sind sie mehrfach einsetzbar. Sie sind nach Ihrer Definition im gesamten Anwenderprogramm verwendbar und somit globale Datentypen. Sie können deshalb diese Datentypen:

- wie elementare oder zusammengesetzte Datentypen in Bausteinen verwenden oder
- als Vorlage für die Erstellung von Datenbausteinen mit gleicher Datenstruktur benutzen.

Bei der Eingabe von anwenderdefinierten Datentypen müssen Sie beachten, dass sie in der S7-SCL-Quelle vor den Bausteinen stehen, in denen sie verwendet werden.



UDT-Bezeichnung

Geben Sie nach dem Schlüsselwort `TYPE` das Schlüsselwort `UDT` und dahinter eine Nummer oder einfach den symbolischen Namen des UDT an. Die Bausteinnummer kann einen Wert von 0 bis 65533 annehmen.

Beispiele:

```

TYPE UDT10
TYPE VERSORGUNGSBLOCK
  
```

Spezifikation des Datentyps

Die Spezifikation des Datentyps erfolgt immer mit Hilfe einer **STRUCT-Datentypspezifikation**. Der Datentyp UDT kann in den Vereinbarungsböcken von Codebausteinen oder in Datenbausteinen verwendet bzw. DB zugeordnet werden.

Beispiel für eine UDT-Definition

```
TYPE MESSWERTE
STRUCT
// UDT Definition mit symbolischer Bezeichnung
    BIPOL_1 : INT := 5;
    BIPOL_2 : WORD := W#16#FFAA ;
    BIPOL_3 : BYTE := B#16#F1 ;
    BIPOL_4 : WORD := B#(25,25) ;
    MESSUNG : STRUCT
        BIPOLAR_10V : REAL ;
        UNIPOLAR_4_20MA : REAL ;
    END_STRUCT ;
END_STRUCT ;
END_TYPE

//Verwendung des UDT in einem FB
FUNCTION_BLOCK FB10
VAR
    MESS_BEREICH : MESSWERTE;
END_VAR
BEGIN
    // . . .
    MESS_BEREICH.BIPOL_1 := -4 ;
    MESS_BEREICH.MESSUNG.UNIPOLAR_4_20MA := 2.7 ;
    // . . .
END_FUNCTION_BLOCK
```

6.17 Compileroptionen in S7-SCL-Quellen

Definition

Neben den Bausteinen können S7-SCL-Quellen auch Angaben über die CompilerEinstellungen enthalten, mit denen die einzelnen Bausteine übersetzt werden sollen.

Compileroptionen steuern den Übersetzungslauf für einzelne Bausteine oder die gesamte Quelle, unabhängig von den Einstellungen im Register "Compiler (Einstellungen)".

Compileroptionen können Sie in S7-SCL-Quellen oder Übersetzungssteuerdateien verwenden.

Gültigkeit

Die Optionen gelten nur für die Übersetzung der Quelle, für die sie definiert wurden. Die Gültigkeit einer Compileroption beginnt mit ihrer Nennung und endet am Ende der Quelle oder der Übersetzungssteuerdatei. Bei mehreren gleichen Compileroptionen gilt die zeitlich letzte.

Wurden für einen Baustein Compileroptionen definiert, sind diese höher prior als die Einstellungen aus dem Register "Compiler (Einstellungen)". Die Einstellungen im Register bleiben jedoch global gültig.

Regeln

Für Compileroptionen gelten folgende Regeln:

- Die Optionen stehen in der Quelle außerhalb der Bausteingrenzen.
- Die Optionen stehen in einer eigenen Zeile.
- Groß- und Kleinschreibung wird nicht unterschieden.

Verfügbare Optionen

Die Tabelle zeigt die verfügbaren Optionen:

Option	Wert	Bedeutung
[Scl_]ResetOptions	Keine Wertangabe möglich	Vorbesetzung der Compilereinstellungen herstellen (Einstellungen aus Dialog)
[Scl_]OverwriteBlocks	'y[es]' oder 'n[o]'	Bausteine überschreiben
[Scl_]GenerateReferenceData	'y[es]' oder 'n[o]'	Referenzdaten erzeugen
[Scl_]S7ServerActive	'y[es]' oder 'n[o]'	Systemattribut "S7_server" berücksichtigen
[Scl_]CreateObjectCode	'y[es]' oder 'n[o]'	Objektcode erstellen
[Scl_]OptimizeObjectCode	'y[es]' oder 'n[o]'	Objektcode optimieren
[Scl_]MonitorArrayLimits	'y[es]' oder 'n[o]'	Feldgrenzen überwachen
[Scl_]CreateDebugInfo	'y[es]' oder 'n[o]'	Debug Info erstellen
[Scl_]SetOKFlag	'y[es]' oder 'n[o]'	OK Flag setzen
[Scl_]SetMaximumStringLength	'1 .. 254'	Maximale Stringlänge

Beispiel

```
{SCL_OverwriteBlocks := 'y' ; SCL_CreateDebugInfo := 'y'}
{SetOKFlag := 'y' ; OptimizeObjectCode := 'y'}
```


7 Datentypen

7.1 Übersicht über die Datentypen in S7-SCL

Ein Datentyp ist die Zusammenfassung von Wertebereichen und Operationen zu einer Einheit.

Die Datentypen bestimmen:

- die Art und Bedeutung der Datenelemente
- die zulässigen Bereiche der Datenelemente
- die zulässige Menge der Operationen, die mit einem Operanden eines Datentyps ausgeführt werden können
- die Schreibweise der Konstanten dieses Datentyps.

Elementare Datentypen

Elementare Datentypen definieren die Struktur von Daten, die nicht in kleinere Einheiten zerlegt werden können. Sie entsprechen der Definition der Norm DIN EN 1131-3. Ein elementarer Datentyp beschreibt einen Speicherbereich mit fester Länge und steht für Bit-, Integer, Real, Zeitdauer, Uhrzeit und Zeichengrößen. Folgende Datentypen sind in S7-SCL vordefiniert.

Gruppe	Datentypen	Bedeutung
Bitdatentypen	BOOL BYTE WORD DWORD	Daten dieses Typs belegen entweder 1Bit, 8 Bit, 16 Bit oder 32 Bit
Zeichentypen	CHAR	Daten dieses Typs belegen genau 1 Zeichen des ASCII-Zeichensatzes
Numerische Typen	INT DINT REAL	Daten dieses Typs stehen für die Verarbeitung numerischer Werte zur Verfügung
Zeittypen	TIME DATE TIME_OF_DAY S5TIME	Daten dieses Typs repräsentieren die unterschiedlichen Zeit- und Datumswerte in STEP 7.

Zusammengesetzte Datentypen

SCL unterstützt folgende zusammengesetzte Datentypen:

Datentyp	Bedeutung
DATE_AND_TIME DT	Definiert einen Bereich mit 64 Bit (8 Byte). Dieser Datentyp speichert (in binärcodiertem Dezimalformat) Datum und Uhrzeit und ist in S7-SCL bereits vordefiniert.
STRING	Definiert einen Bereich für eine Zeichenfolge von maximal 254 Zeichen (Datentyp CHAR).
ARRAY	Definiert ein Feld aus Elementen eines Datentyps (entweder elementar oder zusammengesetzt).
STRUCT	Definiert eine Gruppierung von beliebig kombinierten Datentypen. Sie können ein Feld aus Strukturen oder auch eine Struktur aus Strukturen und Feldern definieren.

Anwenderdefinierte Datentypen

Anwenderdefinierte Datentypen können Sie mit der Datentypdeklaration selbst schaffen. Sie haben einen eigenen Namen und sind mehrfach verwendbar. So kann ein anwenderdefinierter Datentyp zur Erzeugung mehrerer Datenbausteine mit der gleichen Struktur genutzt werden.

Parametertypen

Parametertypen sind spezielle Datentypen für Zeiten, Zähler und Bausteine, die als Formalparameter eingesetzt werden können.

Datentyp	Bedeutung
TIMER	Dient dazu, Zeitfunktionen als Parameter zu vereinbaren
COUNTER	Dient dazu, Zählfunktionen als Parameter zu vereinbaren
BLOCK_xx	Dient dazu, FC, FB, DB und SDB als Parameter zu vereinbaren
ANY	Dient dazu, einen Operanden eines beliebigen Datentyps als Parameter zuzulassen
POINTER	Dient dazu, einen Speicherbereich als Parameter zuzulassen

Datentyp ANY

In S7-SCL können Sie Variablen vom Datentyp ANY als Formalparameter eines Bausteins einsetzen. Darüber hinaus können Sie auch temporäre Variablen dieses Typs anlegen und diese in Wertzuweisungen verwenden.

7.2 Elementare Datentypen

7.2.1 Bitdatentypen

Daten dieses Typs sind Bitkombinationen, die entweder 1 Bit (Datentyp BOOL), 8 Bit, 16 Bit oder 32 Bit belegen. Ein numerischer Wertebereich ist für die Datentypen Byte, Wort und Doppelwort nicht angebar. Es handelt sich um Bitkombinationen, mit denen nur boolesche Ausdrücke gebildet werden können.

Typ	Schlüsselwort	Bitbreite	Ausrichtung	Wertebereich
Bit	BOOL	1 Bit	beginnt am "least significant" Bit im Byte	0, 1 oder FALSE, TRUE
Byte	BYTE	8 Bit	beginnt am "least significant" Byte im Word	-
Wort	WORD	16 Bit	beginnt an einer WORD-Grenze	-
Doppelwort	DWORD	32	beginnt an einer WORD-Grenze	-

7.2.2 Zeichentypen

Daten dieses Typs belegen genau 1 Zeichen des ASCII-Zeichensatzes.

Typ	Schlüsselwort	Bitbreite	Wertebereich
Einzelzeichen	CHAR	8	erweiterter ASCII-Zeichensatz

7.2.3 Numerische Datentypen

Sie stehen für die Verarbeitung numerischer Werte zur Verfügung (z.B. zum Berechnen von arithmetischen Ausdrücken).

Typ	Schlüsselwort	Bitbreite	Ausrichtung	Wertebereich
Integer (Ganzzahl)	INT	16	beginnt an einer WORD-Grenze	-32_768 bis 32_767
Doppelinteger	DINT	32	beginnt an einer WORD-Grenze	-2_147_483_648 bis 2_147_483_647
Gleitpunktzahl (IEEE-Gleitpunktzahl)	REAL	32	beginnt an einer WORD-Grenze	-3.402822E+38 bis -1.175495E-38 +/- 0 1.175495E-38 bis 3.402822E+38

7.2.4 Zeittypen

Daten dieses Typs repräsentieren die unterschiedlichen Zeit-/und Datumswerte innerhalb STEP 7 (z.B. zum Einstellen des Datums oder zum Eingeben des Zeitwerts für eine Zeit).

Typ	Schlüsselwort	Bitbreite	Ausrichtung	Wertebereich
S5-Zeit	S5TIME S5T	16	Beginnt an einer WORD-Grenze	T#0H_0M_0S_10MS bis T#2H_46M_30S_0MS
Zeitdauer: IEC-Zeit in Schritten von 1 ms.	TIME T	32	Beginnt an einer WORD-Grenze	-T#24D_20H_31M_23S_647MS bis T#24D_20H_31M_23S_647MS
Datum: IEC-Datum in Schritten von 1 Tag.	DATE D	16	Beginnt an einer WORD-Grenze	D#1990-01-01 bis D#2168-12-31
Tageszeit: Uhrzeit in Schritten von 1 ms.	TIME_OF_DAY TOD	32	Beginnt an einer WORD-Grenze	TOD#0:0:0.0 bis TOD#23:59:59.999

Bei Variablen vom Datentyp S5TIME ist die Auflösung begrenzt, d.h. es sind lediglich die Zeitbasen 0.01s, 0.1s, 1s, 10s verfügbar. Der Compiler rundet die Werte entsprechend. Wenn der eingestellte Wert größer ist als es der Wertebereich zulässt, wird der obere Grenzwert verwendet.

7.3 Zusammengesetzte Datentypen

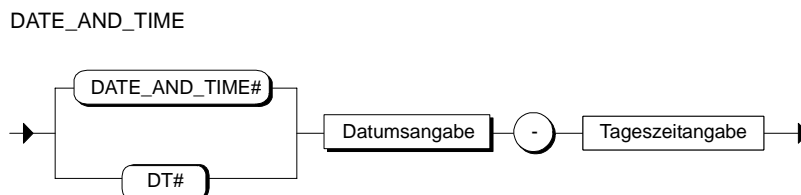
7.3.1 Datentyp DATE_AND_TIME

Definition

Dieser Datentyp definiert einen Bereich mit 64 Bits (8 Byte) für die Angabe von Datum und Uhrzeit. Der Datenbereich speichert die folgenden Informationen:

Jahr, Monat, Tag, Stunden, Minuten, Sekunden, Millisekunden.

Syntax



Die genaue Syntax für die Datums- und Tageszeitangabe finden Sie im Abschnitt "Vereinbarung von Konstanten".

Wertebereich

Typ	Schlüsselwort	Bitbreite	Ausrichtung	Wertebereich
Datum und Uhrzeit	DATE_AND_TIME DT	64	Beginnt und endet an einer WORD-Grenze	DT#1990-01-01-0:0:0.0 bis DT#2089-12-31-23:59:59.999

Der Datentyp Date_And_Time wird im BCD-Format gespeichert:

Bytes	Inhalt	Bereich
0	Jahr	1990 ... 2089
1	Monat	01 ... 12
2	Tag	1 ... 31
3	Stunde	0 ... 23
4	Minute	0 ... 59
5	Sekunde	0 ... 59
6	2 MSD (most significant decade) von ms	00 ... 99

Bytes	Inhalt	Bereich
7 (4 MSB)	LSD (least significant decade) von ms	0 ... 9
7 (4 LSB)	Wochentag	1 ... 7 (1 = Sonntag)

Beispiel

Eine gültige Definition für 20.10.1995 12 Uhr 20 Minuten 30 Sekunden und 10 Millisekunden ist:

```
DATE_AND_TIME#1995-10-20-12:20:30.10  
DT#1995-10-20-12:20:30.10
```

Hinweis

Um gezielt auf die Komponenten DATE oder TIME zuzugreifen, stehen Ihnen Standard-Funktionen (in der STEP 7-Bibliothek) zur Verfügung.

7.3.2 Datentyp STRING

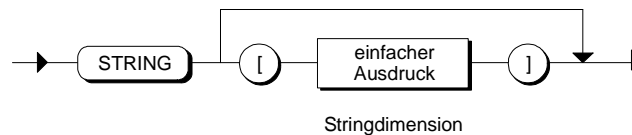
Definition

Ein STRING-Datentyp definiert eine Zeichenkette von maximal 254 Zeichen. Der Standardbereich, der für eine Zeichenkette reserviert ist, besteht aus 256 Byte. Dieser Speicherbereich wird benötigt, um 254 Zeichen und einen Kopf von 2 Byte zu speichern.

Sie können den Speicherplatz für eine Zeichenkette verringern, indem Sie bei der Definition auch die maximale Anzahl der Zeichen angeben, die in der Kette gespeichert werden sollen. Ein Nullstring, d.h. ein String ohne Inhalt, stellt den kleinstmöglichen Wert dar.

Syntax

STRING-Datentypspezifikation



Der einfache Ausdruck steht für die maximale Anzahl der Zeichen im STRING. In einer Zeichenkette sind alle Zeichen des ASCII-Codes zugelassen. Ein String kann auch Sonderzeichen, z.B. Steuerzeichen, und nichtdruckbare Zeichen enthalten. Diese können Sie über die Syntax \$hh eingeben, wobei hh stellvertretend für den hexadezimal ausgedrückten Wert des ASCII-Zeichens steht (Beispiel: '\$0D\$0AText')

Bei der Deklaration von Speicherplatz für Zeichenketten kann die maximale Anzahl der Zeichen, die in der Zeichenkette gespeichert werden sollen, definiert werden. Wird hier keine Maximallänge angegeben, wird ein String der Länge 254 angelegt.

Beispiel:

```
VAR
    Text1    : String [123];
    Text2    : String;
END_VAR
```

Die Konstante "123" bei der Deklaration der Variablen "Text1" steht für die maximale Anzahl der Zeichen in der Zeichenkette. Bei der Variablen "Text2" wird eine Länge von 254 Zeichen reserviert.

Hinweis

Bei Ausgangs- und Durchgangsparemtern sowie bei Rückgabewerten von Funktionen können Sie den standardmäßig reservierten Bereich von 254 Zeichen reduzieren, um die Ressourcen Ihrer CPU besser zu nutzen. Wählen Sie dazu den Menübefehl **Extras > Einstellungen** und im folgenden Dialogfeld das Register "Compiler".

Tragen Sie dann bei der Option "Maximale Stringlänge" die gewünschte Anzahl ein. Beachten Sie, daß die Einstellung sich auf alle STRING-Variablen in der Quelle bezieht. Der eingestellte Wert darf daher nicht kleiner sein als die im Programm verwendeten STRING-Variablen.

Initialisierung von Zeichenketten

Stringvariablen lassen sich, wie andere Variablen auch, bei der Deklaration der Parameter von Funktionsbausteinen (FBs) mit konstanten Zeichenfolgen initialisieren. Bei den Parametern von Funktionen (FCs) ist keine Initialisierung möglich.

Ist die vorbelegte Zeichenkette kürzer als die deklarierte Maximallänge, werden die restlichen Zeichenstellen nicht belegt. Bei der Weiterbearbeitung der Variablen werden nur die aktuell belegten Zeichenstellen berücksichtigt.

Beispiel:

```
x : STRING[7]:='Adresse';
```

Werden temporäre Variablen vom Typ String z. B. zur Zwischenspeicherung von Ergebnissen benötigt, dann müssen diese in jedem Fall vor ihrer erstmaligen Verwendung mit einer Stringkonstanten entweder in der Variablendeklaration oder in einer nachfolgenden Wertzuweisung mit einem Vorbelegungswert beschrieben werden.

Hinweis

Liefert eine Funktion aus einer Standardbibliothek einen Rückgabewert, der vom Datentyp STRING ist, und soll dieser Wert einer temporären Variablen zugewiesen werden, so muss die Variable vorher initialisiert werden.

Beispiel:

```
FUNCTION Test : STRING[45]
VAR_TEMP
  x : STRING[45];
END_VAR
x := 'a';
x := concat (in1 := x, in2 := x);
Test := x;
END_FUNCTION
```

Ohne die Initialisierung **x := 'a'**; würde die Funktion ein falsches Ergebnis liefern.

Ausrichtung

Variablen vom Typ STRING beginnen und enden an einer WORD-Grenze.

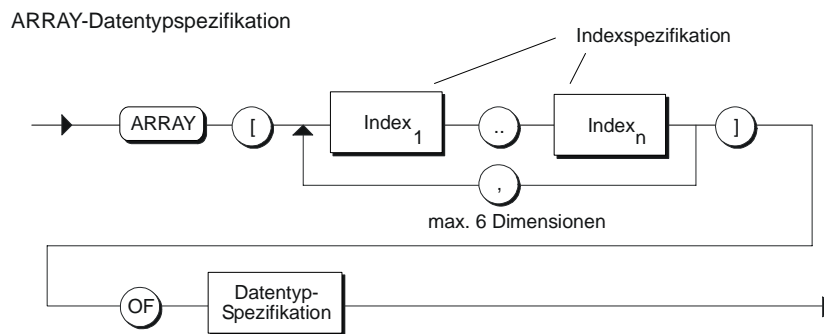
7.3.3 Datentyp ARRAY

Definition

ARRAYs haben eine festgelegte Anzahl von Komponenten eines einzigen Datentyps. S7-SCL unterscheidet:

- den eindimensionalen ARRAY-Typ. Das ist eine Liste von Datenelementen, die in aufsteigender Reihenfolge angeordnet sind.
- den zweidimensionalen ARRAY-Typ. Das ist eine Tabelle von Daten, die aus Zeilen und Spalten besteht. Die erste Dimension bezieht sich auf die Zeilennummer und die zweite auf die Spaltennummer.
- den höherdimensionalen ARRAY-Typ. Das ist die Erweiterung des zweidimensionalen ARRAY-Typs um weitere Dimensionen. Die Anzahl der maximal zulässigen Dimensionen beträgt 6.

Syntax



Indexspezifikation

Sie beschreibt die Dimensionen des ARRAY-Datentyps mit

- dem kleinst- und größtmöglichen Index (Indexbereich) für jede Dimension. Der Index kann ein beliebiger ganzzahliger Wert (-32768 bis 32767) sein.
- Die Grenzen müssen durch zwei Punkte getrennt angegeben werden. Die einzelnen Index-Bereiche werden durch Kommas getrennt.
- Die gesamte Indexspezifikation wird in eckigen Klammern eingeschlossen.

Datentypspezifikation

Mit der Datentypspezifikation deklarieren Sie den Datentyp der Komponenten. Alle Datentypen sind für die Spezifikation zugelassen. Der Datentyp eines Arrays kann z.B. auch eine STRUCT-Typ sein. Parametertypen dürfen nicht als Elementtyp für ein Feld verwendet werden.

Beispiel

```
VAR
    REGLER1 :
        ARRAY[1..3,1..4] OF INT:=-54, 736, -83, 77,
        -1289,          10362, 385, 2,
        60,              -37, -7, 103 ;
    REGLER2 : ARRAY[1..10] OF REAL ;
END_VAR
```

Ausrichtung

Variablen vom Typ ARRAY werden Zeile für Zeile angelegt. Jede Dimension einer Variablen vom Typ BOOL, BYTE oder CHAR enden an einer BYTE-Grenze, alle anderen an einer WORD-Grenze.

7.3.4 Datentyp STRUCT

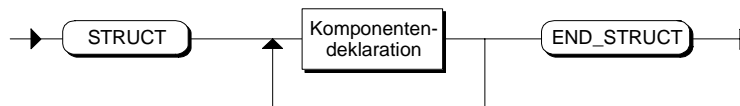
Definition

Der Datentyp STRUCT beschreibt einen Bereich, der aus einer festen Anzahl von Komponenten besteht, die in ihren Datentypen verschieden sein dürfen. Diese Datenelemente werden unmittelbar nach dem Schlüsselwort STRUCT in der Komponentendeklaration angegeben.

Insbesondere kann ein Datenelement des Datentyps STRUCT selbst wieder zusammengesetzt sein. Damit ist die Schachtelung der STRUCT-Datentypen zulässig.

Syntax

STRUCT-Datentypspezifikation

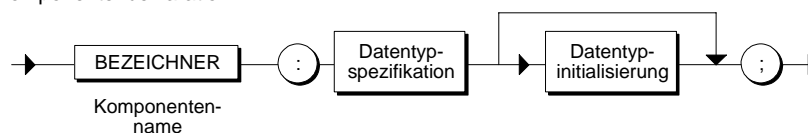


Komponentendeklaration

Die Komponentendeklaration ist eine Auflistung der verschiedenen Komponenten des Datentyps STRUCT. Sie besteht aus:

- 1 bis n Bezeichnern mit dem zugeordneten Datentyp und
- einer optionalen Vorbesetzung mit Anfangswerten.

Komponentendeklaration



Der Bezeichner ist der Name eines Strukturelements für das die nachfolgende Datentypspezifikation gelten soll.

Für die Datentypspezifikation sind alle Datentypen, mit Ausnahme von Parametertypen zugelassen.

Sie können optional nach der Datentypspezifikation ein einzelnes Strukturelement mit einem Anfangswert belegen. Diese Zuordnung erfolgt über eine Wertzuweisung.

Beispiel

```
VAR
    MOTOR : STRUCT
        DATEN : STRUCT
            LASTSTROM : REAL ;
            SPANNUNG  : INT  := 5 ;
        END_STRUCT ;
    END_STRUCT ;
END_VAR
```

Ausrichtung

Variablen vom Typ STRUCT beginnen und enden an einer WORD-Grenze.

Achtung

Falls Sie eine Struktur definieren, die nicht an einer WORD-Grenze endet, füllt S7-SCL automatisch die fehlenden Bytes auf und passt somit die Größe der Struktur an.

Das Anpassen der Strukturgröße kann zu Konflikten beim Zugriff auf Datentypen mit ungerader Byte-Länge führen.

7.4 Anwenderdefinierte Datentypen

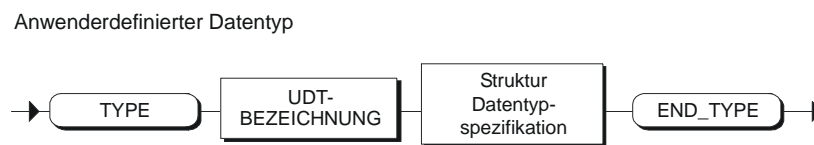
7.4.1 Anwenderdefinierte Datentypen (UDT)

Definition

Einen anwenderdefinierten Datentyp (UDT) definieren Sie als Baustein. Er ist nach seiner Definition im gesamten Anwenderprogramm verwendbar und somit ein globaler Datentyp. Sie können diese Datentypen mit seiner UDT-Bezeichnung UDTx (x steht für eine Nummer) oder unter einem zugeordneten symbolischen Namen im Vereinbarungsteil eines Bausteins oder Datenbausteins verwenden.

Der anwenderdefinierte Datentyp kann zur Vereinbarung von Variablen, Parametern, Datenbausteinen und weiteren anwenderdefinierten Datentypen verwendet werden. Es können auch Komponenten von Feldern oder Strukturen mit Hilfe des anwenderdefinierten Datentyps vereinbart werden.

Syntax



UDT-Bezeichnung

Eine Vereinbarung eines anwenderdefinierten Datentyps wird durch das Schlüsselwort TYPE eingeleitet, gefolgt vom Namen des anwenderdefinierten Datentyps (UDT-Bezeichner). Der Name des anwenderdefinierten Datentyps kann entweder absolut, d.h. durch einen Standardnamen der Form UDTx (x ist eine Nummer) angegeben werden, oder es kann ein symbolischer Name verwendet werden.

Beispiele:

```
TYPE UDT10  
TYPE MESSWERTE
```

Datentypspezifikation

Nach der UDT-Bezeichnung folgt die Spezifikation des Datentyps. Hier ist nur die STRUCT-Datentypspezifikation zugelassen:

```
STRUCT
:
END_STRUCT
```

Hinweis

Innerhalb eines anwenderdefinierten Datentyps gilt die Syntax von AWL. Dies betrifft z.B. die Konstanten-Schreibweise und die Vergabe von Anfangswerten (Initialisierung). Informationen über die Schreibweisen der Konstanten können Sie der Online-Hilfe zu AWL entnehmen.

Beispiel

```
// UDT Definition mit symbolischer Bezeichnung
TYPE
MESSWERTE:      STRUCT
                BIPOL_1 : INT := 5;
                BIPOL_2 : WORD := W#16#FFAA ;
                BIPOL_3 : BYTE := B#16#F1 ;
                BIPOL_4 : WORD := W#16#1919 ;
                MESSUNG : STRUCT
                    BIPOLAR_10V : REAL ;
                    UNIPOLAR_4_20MA : REAL ;
                END_STRUCT;
            END_STRUCT;
END_TYPE

//Verwendung des UDT in einem FB
FUNCTION_BLOCK FB10
VAR
    MESS_BEREICH : MESSWERTE;
END_VAR
BEGIN
    // . . .
    MESS_BEREICH.BIPOL_1 := -4 ;
    MESS_BEREICH.MESSUNG.UNIPOLAR_4_20MA := 2.7 ;
    // . . .
END_FUNCTION_BLOCK
```

7.5 Datentypen für Parameter

Für die Festlegung der formalen Baustein-Parameter von FB und FC können neben den bereits eingeführten Datentypen sogenannte Parametertypen verwendet werden.

Parameter	Größe	Beschreibung
TIMER	2 Byte	Kennzeichnet ein bestimmtes Zeitglied, das von dem Programm in dem aufgerufenen Codebaustein verwendet werden soll. Aktualparameter: z. B. T1
COUNTER	2 Byte	Kennzeichnet einen bestimmten Zähler, der von dem Programm in dem aufgerufenen Codebaustein verwendet werden soll. Aktualparameter: z. B. Z10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 Byte	Kennzeichnet einen bestimmten Baustein, der von einem aufgerufenen AWL-Codebaustein verwendet werden soll. Aktualparameter: z. B. FC101, DB42 Hinweise: Auf den Datentyp BLOCK_DB können Sie absolut zugreifen (myDB.dw10). Ebenso können Sie ihn mit BLOCK_DB_TO_WORD() weiter verarbeiten. Die Datentypen BLOCK_SDB, BLOCK_FB und BLOCK_FC sind von S7-SCL-Programmen nicht auswertbar. Sie können sie lediglich dazu verwenden, Parameter dieses Typs beim Aufruf von AWL-Bausteinen zu versorgen.
ANY	10 Byte	Wird verwendet, wenn als Datentyp des Aktualparameters ein beliebiger Datentyp erlaubt sein soll.
POINTER	6 Byte	Kennzeichnet einen bestimmten Speicherbereich, der von dem Programm verwendet werden soll. Aktualparameter: z. B. M50.0

7.5.1 Datentypen TIMER und COUNTER

Sie legen ein bestimmtes Zeitglied oder ein bestimmtes Zählglied fest, das bei der Bearbeitung eines Bausteins verwendet werden soll. Die Datentypen TIMER und COUNTER sind nur für Eingangsparameter (VAR_INPUT) erlaubt.

7.5.2 BLOCK-Datentypen

Sie legen einen bestimmten Baustein fest, der als Eingangsparameter verwendet werden soll. Die Deklaration des Eingangsparameters bestimmt die Bausteinart (FB, FC, DB). Bei der Parameterversorgung geben Sie den Bausteinbezeichner an. Sowohl der absolute als auch der symbolische Bezeichner ist erlaubt.

Auf den Datentyp BLOCK_DB können Sie absolut zugreifen (`myDB.dw10`). Für die anderen Block-Datentypen stellt S7-SCL keine Operationen zur Verfügung. Es können lediglich Parameter dieses Typs bei Bausteinaufrufen versorgt werden. Bei Funktionen ist das Durchreichen eines Eingangsparameters nicht möglich.

In S7-SCL können Sie Operanden der folgenden Datentypen als Aktualparameter zuordnen:

- Funktionsbausteine ohne Formalparameter.
- Funktionen ohne Formalparameter und Rückgabewert (VOID-Funktionen).
- Datenbausteine und Systemdatenbausteine.

7.5.3 Datentyp POINTER

Dem Datentyp POINTER können Sie Variablen zuweisen, die Sie als Formalparameter eines Bausteins vereinbaren. Beim Aufruf eines solchen Bausteins können diese Parameter mit Operanden beliebigen Datentyps (außer ANY) versorgt werden.

SCL bietet jedoch nur eine Anweisung zur Verarbeitung des Datentyps POINTER an: das Weiterreichen an unterlagerte Bausteine.

Sie können folgende Arten von Operanden als Aktualparameter zuordnen:

- Absolute Adressen
- Symbolischen Namen
- Operanden vom Datentyp POINTER:
Dies ist nur möglich, wenn der Operand ein Formalparameter verträglicher Parameterart ist.
- Konstante NIL:
Sie geben einen Nullpointer an.

Einschränkungen

- Der Datentyp POINTER ist für formale Eingangsparameter, Durchgangparameter von FB und FC und für Ausgangsparameter von FC erlaubt. Konstanten sind als Aktualparameter nicht zulässig (mit Ausnahme er Konstanten NIL).
- Wenn Sie beim Aufruf eines FB oder FC einen Formalparameter vom Typ POINTER mit einer temporären Variable versorgen, können Sie diesen Parameter nicht an einen weiteren Baustein durchreichen. Die temporären Variablen verlieren beim Durchreichen ihre Gültigkeit.

- Beim Aufruf eines FC oder FB können Sie Prozess-Eingänge (%PEW) nur Formalparametern vom Typ Pointer zuweisen, wenn der Formalparameter als Eingangsparameter deklariert wurde.
- Beim Aufruf eines FB können Sie Prozess-Ausgänge (%PAW) nur Formalparametern vom Typ Pointer zuweisen, wenn der Formalparameter als Ausgangsparameter deklariert wurde.

Beispiel

```
FUNCTION FC100 : VOID
VAR_IN_OUT
    N_out : INT;
    out   : POINTER;
END_VAR
VAR_TEMP
    ret   : INT;
END_VAR
BEGIN
    // ...
    ret := SFC79(N := N_out, SA := out);
    // ...
END_FUNCTION

FUNCTION_BLOCK FB100
VAR
    ii : INT;
    aa : ARRAY[1..1000] OF REAL;
END_VAR

BEGIN
    // ...
    FC100( N_out := ii, out := aa);
    // ...
END_FUNCTION_BLOCK
```


7.6 Datentyp ANY

In S7-SCL können Sie Variablen vom Datentyp ANY wie folgt deklarieren:

- Als Formalparameter eines Bausteins, wobei diese Parameter beim Aufruf des Bausteins mit Aktualparametern beliebigen Datentyps versorgt werden dürfen
- Als temporäre Variablen, wobei Sie diesen Variablen Werte beliebigen Datentyps zuweisen können.

Folgende Daten können Sie als Aktualparameter einsetzen bzw. auf der rechten Seite einer Wertzuweisung verwenden:

- Lokale und globale Variablen
- Variablen im DB (absolut oder symbolisch adressiert)
- Variablen in der lokalen Instanz (absolut oder symbolisch adressiert)
- Konstante NIL:
Sie geben einen Nullpointer an.
- Datentyp ANY
- Zeiten, Zähler und Bausteine:
Sie geben den Bezeichner an (z.B. T1, Z20 oder FB6).

Einschränkungen

- Der Datentyp ANY ist für formale Eingangsparameter, Durchgangsparameter von FB und FC und für Ausgangsparameter von FC erlaubt. Konstanten sind als Aktualparameter bzw. auf der rechten Seite einer Wertzuweisung nicht zulässig (mit Ausnahme der Konstanten NIL).
- Wenn Sie beim Aufruf eines FB oder FC einen Formalparameter vom Typ ANY mit einer temporären Variable versorgen, können Sie diesen Parameter nicht an einen weiteren Baustein durchreichen. Die temporären Variablen verlieren beim Durchreichen ihre Gültigkeit.
- Variablen dieses Typs dürfen nicht als Komponententyp in einer Struktur oder als Elementtyp für ein Feld verwendet werden.
- Beim Aufruf eines FC oder FB können Sie Prozess-Eingänge (%PEW) nur Formalparametern vom Typ ANY zuweisen, wenn der Formalparameter als Eingangsparameter deklariert wurde.
- Beim Aufruf eines FB können Sie Prozess-Ausgänge (%PAW) nur Formalparametern vom Typ ANY zuweisen, wenn der Formalparameter als Ausgangsparameter deklariert wurde.

7.6.1 Beispiel zum Datentyp ANY

```
VAR_INPUT
    iANY : ANY;
END_VAR

VAR_TEMP
    pANY : ANY;
END_VAR

CASE ii OF
1:
    pANY := MW4;           // pANY enthält die Adresse
                          // von MW4

3..5:
    pANY:= aINT[ii];      // pANY enthält die Adresse
                          // des ii-ten
                          // Elements von Feld aINT;

100:
    pANY := iANY;         // pANY enthält den Wert der
                          // Eingangsvariablen iANY ELSE
    pANY := NIL;          // pANY enthält den Wert
                          // des NIL-Zeigers
END_CASE;

SFCxxx(IN := pANY);
```

8 Deklaration lokaler Variablen und Parameter

8.1 Lokale Variablen und Bausteinparameter

Kategorien von Variablen

Folgende Tabelle zeigt, in welche Kategorien sich die lokalen Variablen einteilen lassen:

Variable	Bedeutung
Statische Variablen	Statische Variablen sind lokale Variablen, deren Wert über alle Bausteindurchläufe hinweg erhalten bleibt (Bausteingedächtnis). Sie dienen der Speicherung von Werten eines Funktionsbausteins und werden im Instanz-Datenbaustein abgelegt.
Temporäre Variablen	Temporäre Variablen gehören lokal zu einem Codebaustein und belegen keinen statischen Speicherbereich, da sie im Stack der CPU abgelegt werden. Ihr Wert bleibt nur während eines Bausteinablaufs erhalten. Auf temporäre Variablen kann außerhalb des Bausteins, in dem die Variablen deklariert wurden, nicht zugegriffen werden.

Kategorien von Bausteinparametern

Bausteinparameter sind Platzhalter, die erst bei der konkreten Verwendung (Aufruf) des Bausteins festgelegt werden. Die im Baustein stehenden Platzhalter bezeichnet man als Formalparameter, die beim Aufruf des Bausteins zugewiesenen Werte als Aktualparameter. Die Formalparameter eines Bausteins können wie lokale Variablen betrachtet werden.

Die Bausteinparameter lassen sich in folgende Kategorien einteilen:

Bausteinparameter	Bedeutung
Eingangsparameter	Eingangsparameter nehmen beim Aufruf des Bausteins die aktuellen Eingangswerte auf. Sie können nur gelesen werden.
Ausgangsparameter	Ausgangsparameter übergeben die aktuellen Ausgangswerte an den aufrufenden Baustein. Sie können beschrieben aber auch gelesen werden.
Durchgangsparameter	Durchgangsparameter übernehmen beim Aufruf eines Bausteins aktuelle Eingangswerte. Nach der Bearbeitung des Werts nehmen sie das Ergebnis auf und geben es zurück an den aufrufenden Baustein.

Flags (OK-Flag)

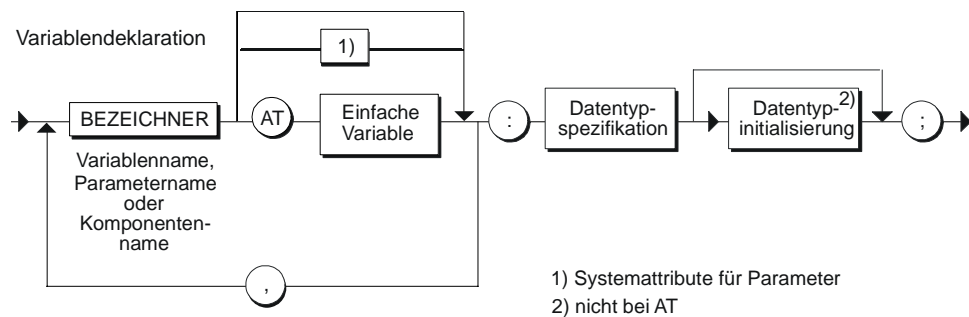
Der S7-SCL-Compiler bietet ein Flag an, das zur Erkennung von Fehlern bei der Ausführung von Programmen in der CPU dient. Es ist eine lokale Variable vom Typ BOOL mit dem vordefinierten Namen "OK".

8.2 Allgemeine Syntax einer Variablen- oder Parameterdeklaration

Variablen und Bausteinparameter müssen einzeln deklariert werden, bevor sie innerhalb eines Codebausteins oder Datenbausteins benutzt werden können. Die Deklaration legt fest, dass ein Bezeichner als Bausteinparameter oder Variable verwendet wird und ordnet ihm einen Datentyp zu.

Eine Variablen- oder Parameterdeklaration besteht aus einem frei wählbaren Bezeichner und einer Datentypangabe. Die allgemeine Form zeigt das Syntaxdiagramm.

Syntax der Variablen- oder Parameterdeklaration



Beispiele

```

WERT1      :      REAL;
falls es mehrere Variablen desselben Typs gibt:

WERT2, WERT3, WERT4, .....: INT;
FELD       :      ARRAY[1..100, 1..10] OF REAL;
SATZ       :      STRUCT
                                MESSFELD:ARRAY[1..20] OF
REAL;
                                SCHALTER:BOOL;
                                END_STRUCT
    
```

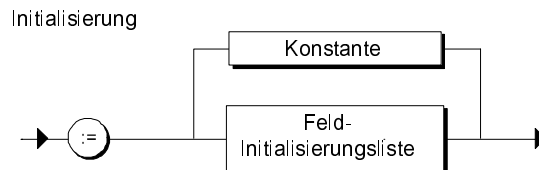
Hinweis

Um reservierte Wörter als Bezeichner zu verwenden, müssen Sie das Zeichen "#" voranstellen (z.B. #FOR).

8.3 Initialisierung

Statische Variablen sowie Eingangs- und Ausgangsparameter eines FB können bei der Vereinbarung mit einem Wert vorbesetzt werden. Auch Durchgangparameter können vorbesetzt werden, jedoch nur, wenn sie elementaren Datentyps sind. Diese Vorbesetzung erfolgt bei einfachen Variablen durch die Zuweisung (:=) einer Konstanten nach der Datentypangabe.

Syntax



Beispiel

```
WERT :REAL := 20.25;
```

Hinweis

Eine Initialisierung einer Variablenliste (A1, A2, A3,...: INT:=...) ist nicht möglich. Sie müssen die Variablen in diesem Fall einzeln initialisieren.

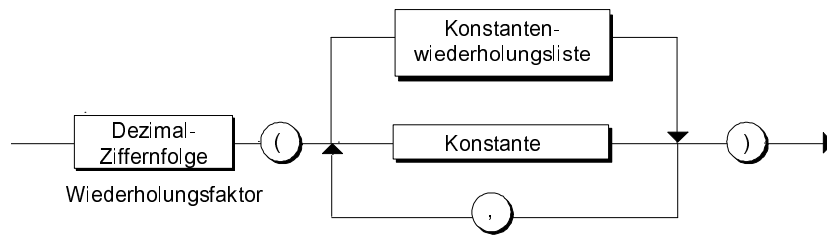
Feld-Initialisierung

Bei der Initialisierung von ARRAYS können Sie entweder für jede Feldkomponente einen Wert durch Komma getrennt angeben, oder durch Voranstellen eines Wiederholungsfaktors (Integer) mehrere Komponenten mit dem gleichen Wert initialisieren.

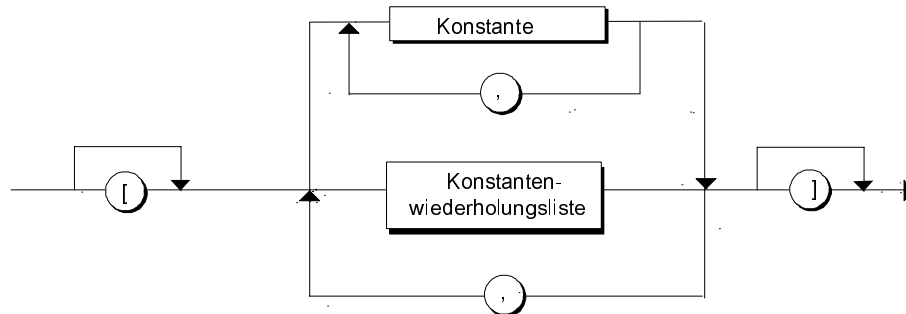
Die Initialwerte können optional in eine Klammer eingeschlossen sein. Auch bei mehr-dimensionalen Feldern wird nur ein Klammerpaar angegeben.

Syntax der Feld-Initialisierung

Konstantenwiederholungsliste



Feld-Initialisierungsliste



Beispiele

```

VAR
// Initialisierung statischer Variablen:
INDEX1 : INT := 3 ;
// Feldinitialisierung:
REGLER1 : ARRAY [1..2, 1..2] OF INT := -54, 736, -83, 77;
REGLER2 : ARRAY[1..10] OF REAL := 10(2.5);
REGLER1 : ARRAY [1..2, 1..2] OF INT := [-54, 736, -83, 77];
REGLER2 : ARRAY[1..10] OF REAL := [10(2.5)];
// Strukturinitialisierung:
GENERATOR: STRUCT
DAT1 : REAL := 100.5;
A1 : INT := 10 ;
A2 : STRING[6] := 'FAKTOR';
A3 : ARRAY[1..12] OF REAL := 0.0, 10(100.0), 1.0;
END_STRUCT ;
END_VAR
    
```

8.4 Deklarieren von Sichten auf Variablenbereiche

Um mit einem weiteren Datentyp auf eine deklarierte Variable zuzugreifen, können Sie Sichten auf die Variable oder auf Bereiche innerhalb der Variablen definieren. Hierzu verwenden Sie das Schlüsselwort "AT". Eine Sicht ist nur lokal im Baustein sichtbar, sie wird nicht in die Schnittstelle übernommen. Eine Sicht darf wie jede andere Variable im Baustein verwendet werden. Sie erbt alle Eigenschaften der Variablen, auf die sie zeigt, nur der Datentyp ist neu.

Beispiel

Im folgenden Beispiel werden mehrere Sichten auf einen Eingangsparameter ermöglicht:

```
VAR_INPUT
    Buffer : ARRAY[0..255] OF BYTE;
    Telegram1 AT Buffer : UDT100 ;
    Telegram2 AT Buffer : UDT200 ;
END_VAR
```

Der aufrufende Baustein versorgt den Parameter Buffer, die Namen Telegram1 und Telegram2 sieht er nicht. Dem aufgerufenen Baustein stehen nun 3 Möglichkeiten zur Verfügung, die Daten zu interpretieren, nämlich als Feld unter dem Namen Buffer oder anders strukturiert unter Telegram1 oder Telegram2.

Regeln

- Die Deklaration einer weiteren Sicht auf eine Variable muss nach der Deklaration der Variablen, auf die sie zeigen soll, im selben Deklarationsblock erfolgen.
- Eine Initialisierung ist nicht möglich.
- Der Datentyp der Sicht muss mit dem Datentyp der Variablen kompatibel sein. Die Variable gibt die Größe der Speicherbereichs vor. Der Speicherbedarf der Sicht darf gleich oder kleiner sein. Darüber hinaus gelten folgende Kombinationsregeln:

		Datentyp der Sicht:	Datentyp der Variablen:		
			Elementar	Zusammen- gesetzt	ANY/POINTER
FB	Deklaration einer Sicht in VAR, VAR_TEMP, VAR_IN oder VAR_OUT	Elementar Zusammen- gesetzt ANY/POINTER	x x	x x x (1)	x (1)
	Deklaration einer Sicht in VAR_IN_OUT	Elementar Zusammen- gesetzt ANY/POINTER	x	x	
FC	Deklaration einer Sicht in VAR oder VAR_TEMP	Elementar Zusammen- gesetzt ANY/POINTER	x x	x x x	x
	Deklaration einer Sicht in VAR_IN, VAR_OUT oder VAR_IN_OUT	Elementar Zusammen- gesetzt ANY/POINTER	x	x	

(1) Any_Pointer in VAR_OUT nicht erlaubt.

Elementar = BOOL, BYTE, WORD, DWORD, INT, DINT, DATE, TIME, S5TIME, CHAR

Zusammengesetzt = ARRAY, STRUCT, DATE_AND_TIME, STRING

8.5 Verwendung von Multiinstanzen

Möglicherweise wollen oder können Sie aufgrund der Leistungsdaten (z.B. Speicherplatz) der verwendeten S7-CPU nur eine beschränkte Anzahl von Datenbausteinen für Instanzdaten spendieren. Wenn in Ihrem Anwenderprogramm in einem FB weitere, bereits vorhandene Funktionsbausteine aufgerufen werden (Aufrufhierarchie von FB), dann können Sie diese weiteren Funktionsbausteine ohne eigene (d. h. zusätzliche) Instanz-DB aufrufen.

Dazu bietet sich folgende Lösung an:

- Nehmen Sie die aufzurufenden FB als statische Variablen in die Variablendeklaration des aufrufenden FB auf.
- In diesem Funktionsbaustein rufen Sie weitere Funktionsbausteine auf ohne eigene Instanz-DB.
- Damit erreichen Sie eine Konzentrierung der Instanzdaten in einem Instanz-Datenbaustein, d.h. Sie können die verfügbare Anzahl der DB besser ausnutzen.

8.6 Instanzdeklaration

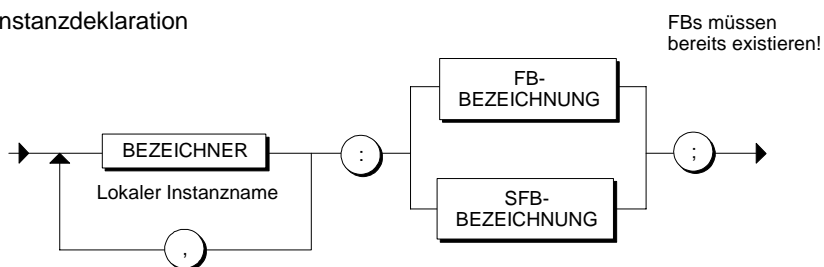
Bei Funktionsbausteinen können Sie im Vereinbarungsblock für statische Variablen (VAR; END_VAR) außer den Variablen mit elementarem, zusammengesetztem oder anwenderdefiniertem Datentyp, auch Variablen vom Typ FB oder SFB vereinbaren. Diese Variablen werden lokale Instanzen des FB oder SFB genannt.

Die lokalen Instanzdaten werden im Instanzdatenbaustein des aufrufenden Funktionsbausteins gespeichert. Eine lokale instanzspezifische Initialisierung ist nicht möglich.

Bausteine, die als lokale Instanz aufgerufen werden, dürfen nicht die Länge 0 haben. Mindestens eine statische Variable oder ein Parameter muss in einem solchen Baustein deklariert sein.

Syntax

Instanzdeklaration



Beispiel

```
Versorgung1      : FB10;
Versorgung2, Versorgung3, Versorgung4      : FB100;
Motor1          : Motor ;
```

Dabei ist `Motor` ein in der Symboltabelle eingetragenes Symbol für einen FB.

8.7 Flags (OK-Flag)

Das OK-Flag dient dazu, die korrekte oder inkorrekte Ausführung eines Bausteins zu vermerken. Es ist eine lokale Variable vom Typ BOOL mit dem vordefinierten Namen "OK".

Zu Beginn einer Programmausführung hat das OK-Flag den Wert TRUE. Es kann an beliebiger Stelle im Baustein mit S7-SCL-Anweisungen abgefragt oder auf TRUE / FALSE gesetzt werden. Tritt während der Ausführung einer Operation ein Fehler auf (z.B. eine Division durch Null), wird das OK-Flag auf FALSE gesetzt. Beim Verlassen des Bausteins wird der Wert des OK-Flags in den Ausgangsparameter ENO gespeichert und kann dadurch vom aufrufenden Baustein ausgewertet werden.

Deklaration

Das OK-Flag ist eine systemvereinbarte Variable. Es ist keine Vereinbarung erforderlich. Sie müssen aber die Compileroption "OK-Flag setzen" vor dem Übersetzen wählen, wenn Sie das OK-Flag in Ihrem Anwenderprogramm verwenden möchten.

Beispiel

```
// OK-Flag auf TRUE setzen,  
// damit überprüft werden kann,  
// ob die Aktion korrekt abläuft.  
OK:= TRUE;  
Division:= 1 / IN;  
IF OK THEN  
    // Die Division verlief korrekt.  
  
    // :  
    // :  
  
ELSE    // Die Division verlief fehlerhaft.  
  
    // :  
  
END_IF;
```

8.8 Deklarationsabschnitte

8.8.1 Übersicht der Deklarationsabschnitte

Jeder Kategorie lokaler Variablen oder Parameter ist ein eigener Vereinbarungsblock zugeordnet, der jeweils durch ein eigenes Schlüsselwortpaar gekennzeichnet ist. Jeder Block enthält die Deklarationen, die für diesen Vereinbarungsblock erlaubt sind. Die Reihenfolge dieser Blöcke ist beliebig.

Folgende Tabelle zeigt, welche Variablen- oder Parameterart Sie in den verschiedenen Codebausteinen vereinbaren dürfen:

Daten	Syntax	FB	FC	OB
Variablen als: Statische Variable	VAR ... END_VAR	X	X *)	
Temporäre Variable	VAR_TEMP ... END_VAR	X	X	X
Bausteinparameter als: Eingangsparameter	VAR_INPUT ... END_VAR	X	X	
Ausgangsparameter	VAR_OUTPUT ... END_VAR	X	X	
Durchgangsparameter	VAR_IN_OUT ... END_VAR	X	X	

*) Die Vereinbarung von Variablen innerhalb des Schlüsselwortpaars VAR und END_VAR ist in Funktionen zwar erlaubt, die Vereinbarungen werden aber beim Übersetzen einer Quelle im temporären Bereich angelegt.

8.8.2 Statische Variablen

Statische Variablen sind lokale Variablen, deren Wert über alle Bausteindurchläufe hinweg erhalten bleibt (Bausteingedächtnis). Sie dienen der Speicherung von Werten eines Funktionsbausteins und werden in einem zugehörigen Instanzdatenbaustein gespeichert.

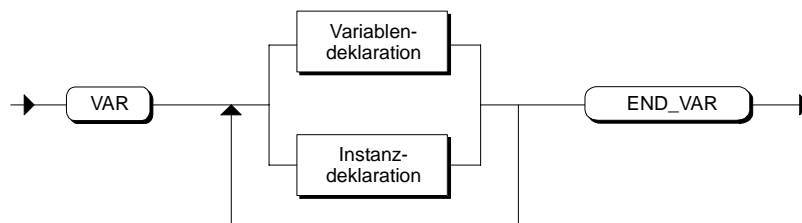
Syntax

Statische Variablen werden im Deklarationsabschnitt VAR / END_VAR deklariert. Dieser Vereinbarungsblock ist Bestandteil des FB-Vereinbarungsteils. Nach dem Übersetzen bestimmt dieser Block zusammen mit den Blöcken für die Bausteinparameter den Aufbau des zugeordneten Instanz-Datenbausteins.

Sie können in diesem Block:

- Variablen anlegen, den Variablen Datentypen zuweisen und die Variablen initialisieren.
- Einen aufzurufenden FB als statische Variable deklarieren, wenn Sie diesen im aktuellen FB als lokale Instanz aufrufen möchten.

Statischer Variablenblock



Beispiel

```

VAR
DURCHLAUF      :INT;
MESSFELD       :ARRAY [1..10] OF REAL;
SCHALTER       :BOOL;
MTOR_1,MOTOR_2 :FB100;      //Instanzdeklaration

END_VAR
  
```

Zugriff

Der Zugriff auf die Variablen erfolgt im Anweisungsteil:

- **Zugriff innerhalb des Bausteines:** Im Anweisungsteil des Funktionsbausteins, in dessen Vereinbarungsteil eine Variable deklariert wurde, können Sie auf die Variable zugreifen. Die genaue Vorgehensweise ist im Kapitel "Wertzuweisung" erklärt.
- **Zugriff von außen über den Instanz-DB:** Von anderen Bausteinen aus können Sie die Variable durch indizierten Zugriff ansprechen, z.B. *DBx.variable*.

8.8.3 Temporäre Variablen

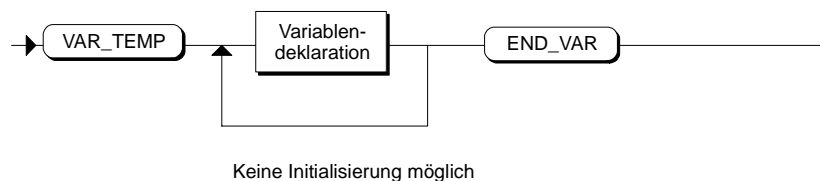
Temporäre Variablen gehören lokal zu einem Codebaustein und belegen keinen statischen Speicherbereich. Sie werden im Stack der CPU abgelegt. Ihr Wert bleibt nur während eines Bausteinablaufs erhalten. Auf temporäre Variablen kann außerhalb des Bausteins, in dem die Variablen deklariert wurden, nicht zugegriffen werden. Bei Beginn der Ausführung eines OB, FB oder FC ist der Wert der temporären Daten nicht definiert. Eine Initialisierung ist nicht möglich.

Sie sollten Daten dann als temporäre Daten vereinbaren, wenn Sie diese nur zur Speicherung von Zwischenergebnissen bei der Bearbeitung Ihres OB, FB oder FC benötigen.

Syntax

Die Vereinbarung der temporären Variablen geschieht im Deklarationsabschnitt VAR_TEMP / END_VAR. Dieser Vereinbarungsblock ist Bestandteil eines FB, FC, oder OB. Dabei werden innerhalb der Variablendeklaration Variablenamen und Datentypen angegeben.

Temporärer Variablenblock



Beispiel

```
VAR_TEMP
    PUFFER 1      : ARRAY [1..10] OF INT ;
    HILF1, HILF2 : REAL ;
END_VAR
```

Zugriff

Der Zugriff auf die Variablen erfolgt immer im Anweisungsteil des Codebausteins, in dessen Vereinbarungsteil die Variable deklariert wurde (Zugriff von innen), siehe Kapitel "Wertzuweisung".

8.8.4 Bausteinparameter

Parameter sind Platzhalter, die erst bei der konkreten Verwendung (Aufruf) des Bausteins festgelegt werden. Die im Baustein stehenden (vereinbarten) Platzhalter bezeichnet man als Formalparameter, die beim Aufruf des Bausteins zugewiesenen Werte als Aktualparameter. Parameter bilden somit einen Mechanismus zum Informationsaustausch zwischen den Bausteinen.

Arten von Bausteinparametern

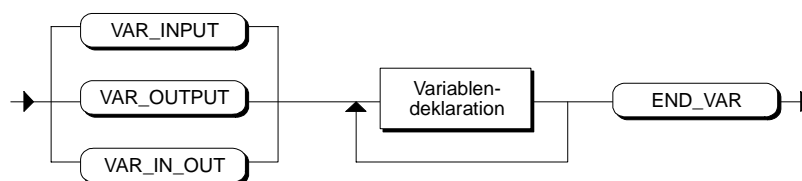
- Formale Eingangparameter nehmen die aktuellen Eingangswerte auf (Datenfluss von außen nach innen).
- Formale Ausgangparameter dienen der Übergabe von Ausgangswerten (Datenfluss von innen nach außen).
- Formale Durchgangparameter haben sowohl die Funktion eines Eingangs- als auch eines Ausgangsparameters.

Syntax

Die Vereinbarung der Formalparameter geschieht im Vereinbarungsteil eines Funktionsbausteins bzw. einer Funktion getrennt nach Parametertyp in den drei Vereinbarungsblocken für Parameter. Innerhalb der Variablendeklaration legen Sie den Parameternamen und den Datentyp fest. Eine Initialisierung ist nur bei Eingangs- und Ausgangsparametern eines FB möglich.

Bei der Vereinbarung der Formalparameter können Sie außer elementaren, zusammengesetzten und anwenderdefinierten Datentypen auch die Datentypen für Parameter verwenden.

Parameterblock



Initialisierung nur möglich für VAR_INPUT und VAR_OUTPUT

Beispiel

```
VAR_INPUT    // Eingangsparameter
    MEIN_DB: BLOCK_DB ;
    REGLER : DWORD ;
    UHRZEIT: TIME_OF_DAY ;
END_VAR

VAR_OUTPUT  // Ausgangsparameter
    SOLLWERTE: ARRAY [1..10] of INT ;
END_VAR

VAR_IN_OUT  // Durchgangsparameter
    EINSTELLUNG : INT ;
END_VAR
```

Zugriff

Der Zugriff auf die Bausteinparameter erfolgt im Anweisungsteil eines Codebausteins:

- **Zugriff von innen:** d.h. im Anweisungsteil des Bausteins, in dessen Vereinbarungsteil der Parameter deklariert wurde. Dies ist im Kapitel "Wertzuweisung" und im Kapitel "Ausdrücke, Operationen und Operanden" erklärt.
- **Zugriff von außen über Instanz-DB:** Auf Bausteinparameter von Funktionsbausteinen können Sie über den zugeordneten Instanz-DB zugreifen.

9 Vereinbarung von Konstanten und Sprungmarken

9.1 Konstanten

Konstanten sind Daten, die einen festen Wert besitzen, der sich zur Programmlaufzeit nicht ändern kann.

SCL kennt folgende Gruppen von Konstanten.

- Bitkonstanten
- Numerische Konstanten
 - Ganzzahl-Konstanten
 - Realzahl-Konstanten
- Zeichen-Konstanten
 - Char-Konstanten
 - String-Konstanten
- Zeitangabe
 - Datum-Konstanten
 - Zeitdauer-Konstanten
 - Tageszeit-Konstanten
 - Datum und Zeit-Konstanten

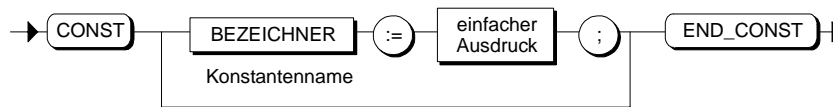
9.1.1 Vereinbarung symbolischer Namen für Konstanten

Eine Vereinbarung von Konstanten ist nicht notwendig. Sie haben aber die Möglichkeit, im Vereinbarungsteil symbolische Namen für die Konstanten zu vergeben.

Die Vereinbarung symbolischer Namen für Konstanten erfolgt mit der CONST-Anweisung im Vereinbarungsteil Ihres Code-Bausteins. Sie ist für alle Konstanten eines Bausteins empfehlenswert. Dadurch erreichen Sie eine bessere Lesbarkeit und eine einfachere Pflege des Bausteins bei Änderung von konstanten Werten.

Syntax

Konstantenblock



In einfachen Ausdrücken sind nur die arithmetischen Basisoperationen zulässig (*, /, +, -, DIV, MOD).

Beispiel

```

CONST
    Zahl      := 10 ;
    UHRZEIT1  := TIME#1D_1H_10M_22S_2MS ;
    NAME      := 'SIEMENS' ;
    ZAHL2     := 2 * 5 + 10 * 4 ;
    ZAHL3     := 3 + ZAHL2 ;
END_CONST
  
```

9.1.2 Datentypen von Konstanten

Die Zuweisung von Datentypen an Konstanten unterscheidet sich von der Vorgehensweise bei AWL:

Eine Konstante erhält ihren Datentyp immer erst mit der arithmetischen oder logischen Verknüpfung, in der sie verwendet wird, z.B.:

```
Int1:=Int2 + 12345           //"12345" erhält den Datentyp INT
Real1:=Real2 + 12345       //"12345" erhält den Datentyp REAL
```

Dabei wird der Konstanten der Datentyp zugewiesen, dessen Wertebereich gerade noch ausreicht, um die Konstante ohne Wertverlust aufzunehmen. Zum Beispiel hat die Konstante "12345" nicht wie bei AWL immer den Datentyp INT, sondern sie hat die Datentypklasse ANY_NUM, je nach Verwendung also INT, DINT, oder REAL.

Typisierte Konstanten

Durch die typisierte Konstantenschreibweise können Sie bei folgenden numerischen Datentypen auch explizit einen Datentyp angeben.

Beispiele:

Datentyp	Typisierte Schreibweise	
BOOL	BOOL#1 Bool#false	bool#0 BOOL#TRUE
BYTE	BYTE#0 Byte#'ä'	B#2#101 b#16#f
WORD	WORD#32768 W#2#1001_0100	word#16#f WORD#8#177777
DWORD	DWORD#16#f000_0000 DW#2#1111_0000_1111_0000	dword#32768 DWord#8#3777777777
INT	INT#16#3f_ff Int#2#1111_0000	int#-32768 inT#8#77777
DINT	DINT#16#3fff_ffff DInt#2#1111_0000	dint#-65000 dinT#8#1777777777
REAL	REAL#1 real#2e4	real#1.5 real#3.1
CHAR	CHAR#A	CHAR#49

9.1.3 Schreibweisen von Konstanten

Für den Wert einer Konstanten gibt es je nach Datentyp und Datenformat eine bestimmte Schreibweise (Format). Typ und der Wert der Konstanten gehen direkt aus der Schreibweise hervor und müssen deshalb nicht deklariert werden.

Beispiele:

15	WERT 15	als Integer-Konstante in Dezimaldarstellung
2#1111	WERT 15	als Integer-Konstante in Binärdarstellung
16#F	WERT 15	als Integer-Konstante in Hexadarstellung

Übersicht über die möglichen Schreibweisen

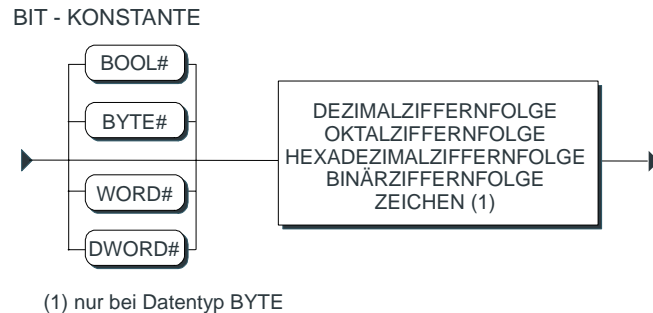
Datentyp	Beschreibung	Beispiele in S7-SCL	Beispiele in AWL, falls abweichend
BOOL	1 Bit	FALSE TRUE BOOL#0 BOOL#1 BOOL#FALSE BOOL#TRUE	
BYTE	8 Bit-Hexazahl	B#16#00 B#16#FF BYTE#0 B#2#101 Byte#'ä' b#16#f	
CHAR	8 Bit (1 ASCII-Zeichen)	'A' CHAR#49	
STRING	Maximal 254 ASCII-Zeichen	'Adresse'	
WORD	16 Bit- Hexazahl	W#16#0000 W#16#FFFF word#16#f	
	16 Bit Oktalzahl	WORD#8#177777 8#177777	
	6 Bit Binärzahl	W#2#1001_0100 WORD#32768	
DWORD	32 Bit Hexazahl	DW#16#0000_0000 DW#16#FFFF_FFFF	
	32 Bit Oktalzahl	Dword#8#3777777777 8#3777777777	
	32 Bit Binärzahl	DW#2#1111_0000_1111_0000 dword#32768	

Datentyp	Beschreibung	Beispiele in S7-SCL	Beispiele in AWL, falls abweichend
INT	16 Bit Festpunktzahl	-32768 +32767 INT#16#3f_ff int#-32768 Int#2#1111_0000 inT#8#77777	
DINT	32 Bit Festpunktzahl	-2147483648 +2147483647 DINT#16#3fff_ffff dint#-65000 Dint#2#1111_0000 dinT#8#17777777777	L#-2147483648 L#+2147483647
REAL	32 Bit Gleitpunktzahl	Dezimaldarstellung 123.4567 REAL#1 real#1.5 Exponentialdarstellung real#2e4 +1.234567E+02	
S5TIME	16 Bit Zeitwert im SIMATIC-Format	T#0ms TIME#2h46m30s T#0.0s TIME#24.855134d	S5T#0ms S5TIME#2h46m30s
TIME	32 Bit Zeitwert im IEC-Format	-T#24d20h31m23s647ms TIME#24d20h31m23s647ms T#0.0s TIME#24.855134d	
Date	16 Bit Datumswert	D#1990-01-01 DATE#2168-12-31	
TIME_OF_DAY	32 Bit Tageszeit	TOD#00:00:00 TIME_OF_DAY#23:59:59.999	
DATE_AND_ TIME	Datums- und Zeitwert	DT#1995-01-01-12:12:12.2	

9.1.3.1 Bit-Konstanten

Bit-Konstanten enthalten Werte der Länge 1 Bit, 8 Bit, 16 Bit oder 32 Bit. Diese können im S7-SCL-Programm (je nach Länge) Variablen mit den Datentypen BOOL, BYTE, WORD oder DWORD zugewiesen werden.

Syntax



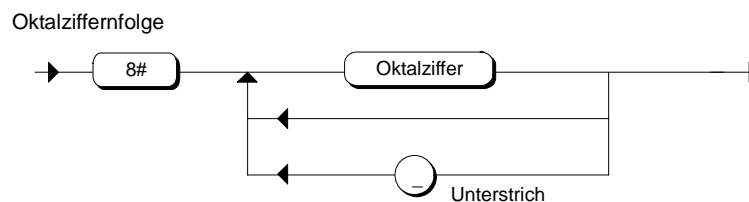
Dezimalziffernfolge

Die Dezimalziffernfolge innerhalb von Konstanten kann optional durch Unterstriche getrennt werden. Die Unterstriche unterstützen die bessere Lesbarkeit bei größeren Zahlen. Die folgenden Beispiele sind zulässige Schreibweisen für eine Dezimalziffernfolge innerhalb von Konstanten:

```
DW#2#1111_0000_1111_0000
dword#32768
```

Binäre, oktale, hexadezimale Werte

Die Angabe einer Ganzzahl-Konstante in einem anderen Zahlensystem als dem dezimalen, erfolgt durch das Voranstellen der Präfixe **2#**, **8#** oder **16#**, gefolgt von der Zahl in der Darstellung des jeweiligen Systems. Dies ist im folgenden Bild am Beispiel einer Ziffernfolge für eine Oktalzahl erklärt:



Beispiel

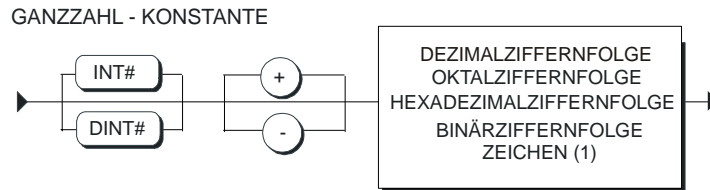
Die folgenden Beispiele sind mögliche Schreibweisen für Bit-Konstanten:

```
Bool#false
8#177777
DW#16#0000_0000
```

9.1.3.2 Ganzzahl-Konstante

Ganzzahl-Konstanten enthalten ganzzahlige Werte der Länge 16 Bit oder 32 Bit. Diese können im S7-SCL-Programm (je nach Länge) Variablen mit den Datentypen INT oder DINT zugewiesen werden.

Syntax



(1) nur bei Datentyp INT

Dezimalziffernfolge

Die Dezimalziffernfolge innerhalb von Konstanten kann optional durch Unterstriche getrennt werden. Die Unterstriche unterstützen die bessere Lesbarkeit bei größeren Zahlen. Die folgenden Beispiele sind zulässige Schreibweisen für eine Dezimalziffernfolge innerhalb von Konstanten:

```

1000
1_120_200
666_999_400_311
  
```

Binäre, oktale, hexadezimale Werte

Die Angabe einer Ganzzahl-Konstante in einem anderen Zahlensystem als dem dezimalen, erfolgt durch das Voranstellen der Präfixe **2#**, **8#** oder **16#**, gefolgt von der Zahl in der Darstellung des jeweiligen Systems.

Beispiel

Die folgenden Beispiele sind mögliche Schreibweisen für Ganzzahl-Konstanten:

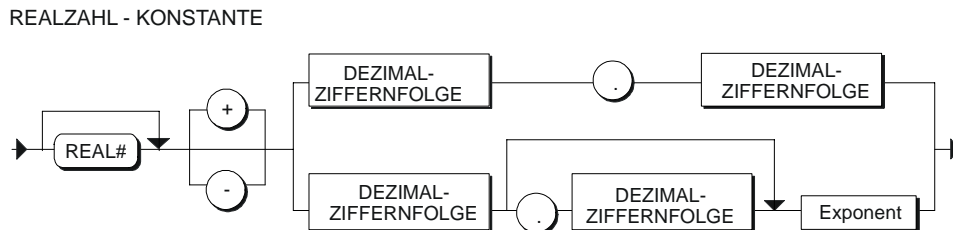
```

Wert_2:=2#0101           // Binärzahl, Dezimalwert 5
Wert_3:=8#17;           // Oktalzahl, Dezimalwert 14
Wert_4:=16#F;           // Hexadezimalzahl, Dezimalwert 15
Wert_5:=INT#16#3f_ff    // Hexadezimalzahl,
                        // typisierte Schreibweise
  
```

9.1.3.3 Realzahl-Konstante

Realzahl-Konstanten sind Werte mit Nachkommastellen. Sie können Variablen mit dem Datentyp REAL zugewiesen werden.

Syntax



Die Angabe des Vorzeichens ist optional. Wird kein Vorzeichen angegeben wird die Zahl als positiv interpretiert.

Die Dezimalziffernfolge innerhalb von Konstanten kann optional durch Unterstriche getrennt werden. Die Unterstriche unterstützen die bessere Lesbarkeit bei größeren Zahlen. Die folgenden Beispiele sind zulässige Schreibweisen für eine Dezimalziffernfolge innerhalb von Konstanten:

```
1000
1_120_200
666_999_400_311
```

Exponent

Bei der exponentiellen Schreibweise können Sie zur Angabe von Gleitpunktzahlen einen Exponenten verwenden. Die Angabe des Exponenten erfolgt durch das Vorstellen des Buchstabens "E" oder "e" gefolgt von dem Integerwert.

Die Größe 3×10^{10} kann in S7-SCL durch folgende Realzahlen dargestellt werden:

```
3.0E+10  3.0E10      3e+10      3E10
0.3E+11  0.3e11      30.0E+9    30e9
```

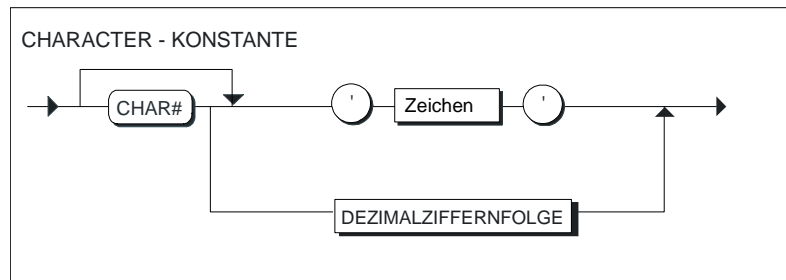
Beispiele

```
ZAHL4 := -3.4 ;
ZAHL5 := 4e2 ;
ZAHL6 := real#1.5;
```


9.1.3.4 Char-Konstante (Einzelzeichen)

Die Char-Konstante enthält genau ein Zeichen. Das Zeichen wird durch einfache Hochkommata (') eingeschlossen. Char-Konstanten können nicht in Ausdrücken verwendet werden.

Syntax



Beispiel

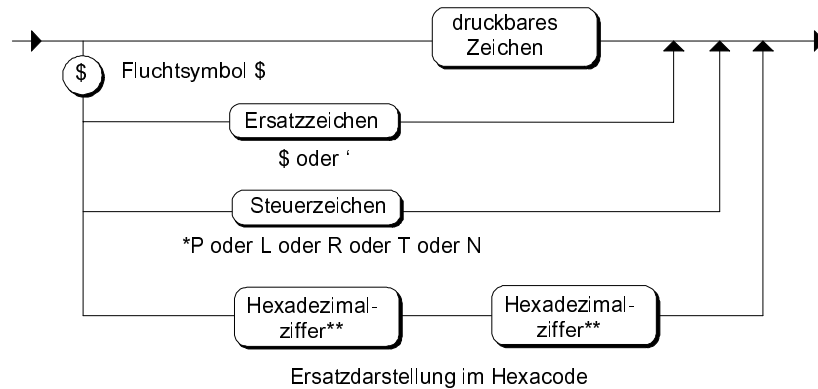
```
Zeichen_1 := 'B';  
Zeichen_2 := char#43;  
Zeichen_3 := char#'B';
```

Syntax des Zeichens

Das Zeichen kann dem vollständigen erweiterten ASCII-Zeichensatz entstammen. Spezielle Formatierungszeichen, das Anführungszeichen (') oder ein \$-Zeichen können Sie mit Hilfe des Fluchtsymbols \$ eingeben.

Sie können auch nichtdruckbare Zeichen aus dem vollständigen, erweiterten ASCII-Zeichensatz verwenden. Sie müssen dazu die Ersatzdarstellung im Hexacode angeben.

Zeichen



* P=Seitenvorschub
L=Zeilenvorschub
R=Wagenruecklauf
T=Tabulator
N=neue Zeile

** \$00 nicht zulaessig

Beispiel für die Angabe eines Zeichens im Hexacode:

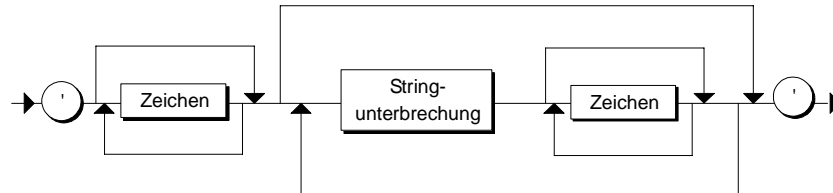
```
ZEICHEN := '$41' ; //entspricht dem Zeichen 'A'
Blank   := '$20' ; //entspricht dem Zeichen □
```

9.1.3.5 String-Konstante

Eine String-Konstante ist eine Zeichenkette von maximal 254 Zeichen. Die Zeichen werden durch einfache Hochkommata eingeschlossen. String-Konstanten können nicht in Ausdrücken verwendet werden.

Syntax

STRING - KONSTANTE

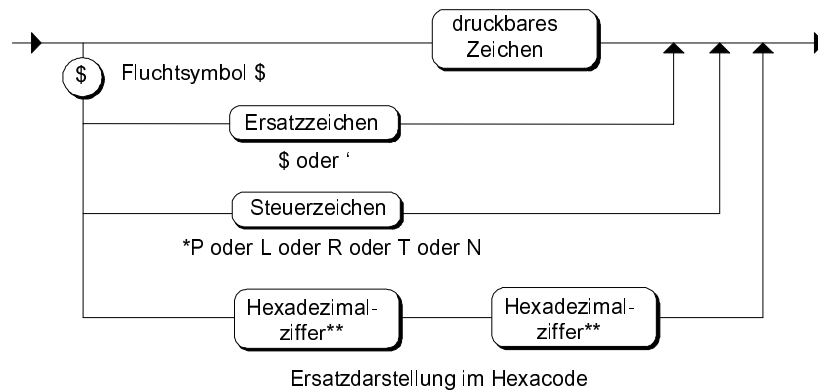


Syntax des Zeichens

Das Zeichen kann dem vollständigen erweiterten ASCII-Zeichensatz entstammen. Spezielle Formatierungszeichen, das Anführungszeichen (') oder ein \$-Zeichen können Sie mit Hilfe des Fluchtsymbols \$ eingeben.

Sie können auch nicht druckbare Zeichen aus dem vollständigen, erweiterten ASCII-Zeichensatz verwenden. Sie müssen dazu die Ersatzdarstellung im Hexadezimalcode angeben.

Zeichen



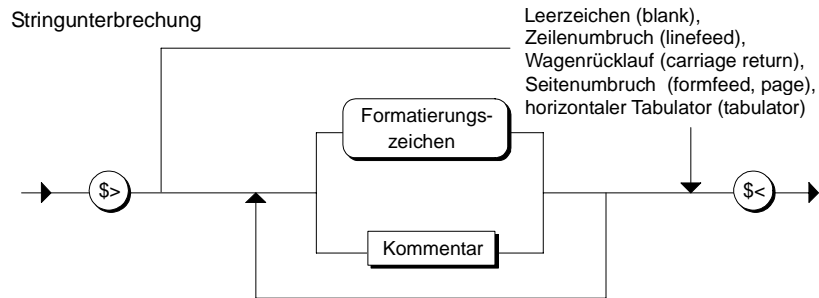
* P=Seitenvorschub
L=Zeilenvorschub
R=Wagenruecklauf
T=Tabulator
N=neue Zeile

** \$00 nicht zulaessig

Stringunterbrechung

Eine String-Konstante dürfen Sie mehrmals unterbrechen und fortsetzen.

Ein String steht entweder in einer Zeile eines S7-SCL-Bausteins oder wird durch spezielle Kennungen auf mehrere Zeilen aufgeteilt. Zur Unterbrechung eines Strings dient die Kennung \$>, zur Fortsetzung in einer Folgezeile dient die Kennung \$<. Der Raum zwischen der Unterbrechungs- und der Fortsetzungskennung darf sich auch auf mehrere Zeilen erstrecken und neben Leerzeichen lediglich Kommentar enthalten.



Beispiele

```
// String-Konstante:
NAME:= 'SIEMENS';
//Unterbrechung einer String-Konstanten
MELDUNG1:= 'MOTOR- $>
$< Steuerung';
// String in hexadezimaler Darstellung:
MELDUNG1:= '$41$4E' (*Zeichenfolge AN*);
```

9.1.3.6 Datumskonstante

Ein Datum wird durch die Präfixe DATE# oder D# eingeleitet. Die Datumsangabe erfolgt durch Integerzahlen für die Jahreszahl (4-stellig), die Monatsangabe und die Tagesangabe, die durch Bindestriche zu trennen sind.

Syntax

Datumsangabe



Beispiel

```
ZEITVARIABLE1 := DATE#1995-11-11 ;
ZEITVARIABLE2 := D#1995-05-05 ;
```

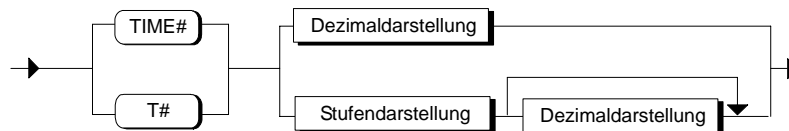
9.1.3.7 Zeitdauer-Konstante

Eine Zeitdauer wird durch die Präfixe TIME# oder T# eingeleitet. Die Angabe der Zeitdauer kann auf zwei Arten erfolgen:

- Dezimaldarstellung
- Stufendarstellung

Syntax

ZEITDAUER



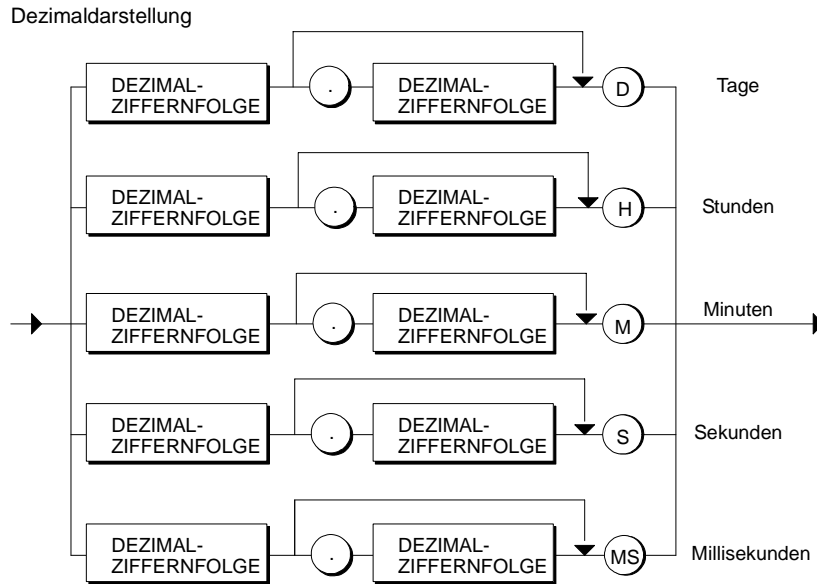
- Jede Zeiteinheit (z. B. Stunden, Minuten) darf nur 1 x angegeben werden.
- Die Reihenfolge - Tage, Stunden, Minuten, Sekunden, Millisekunden - ist einzuhalten.

Der Wechsel von der Stufendarstellung zur Dezimaldarstellung ist nur bei noch nicht angegebenen Zeiteinheiten möglich.

Nach den einleitenden Präfix T# oder TIME# müssen Sie mindestens eine Zeiteinheit angeben.

Dezimaldarstellung

Die Dezimaldarstellung verwenden Sie, wenn Sie eine Zeitkomponente, wie Stunden oder Minuten, als Dezimalzahl angeben möchten.

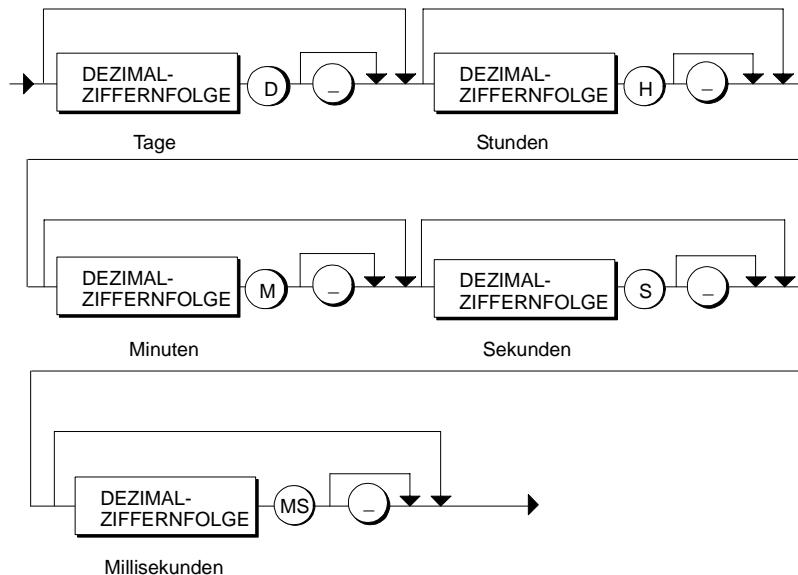


Der Einstieg in die Dezimaldarstellung ist nur bei noch nicht definierten Zeiteinheiten möglich.

Stufendarstellung

Die Stufendarstellung ist eine Sequenz der einzelnen Zeitkomponenten. Es werden erst Tage, gefolgt von Stunden usw., durch Unterstrich getrennt angegeben, wobei das Weglassen von Komponenten erlaubt ist. Eine Komponente muss aber mindestens angegeben werden.

Stufendarstellung



Beispiel

```
// Dezimaldarstellung
Intervall1:= TIME#10.5S ;

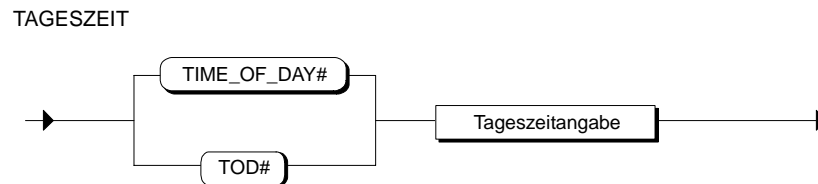
// Stufendarstellung
Intervall2:= T#3D_2S_3MS ;

// Stufendarstellung und Dezimaldarstellung
Intervall3 := T#2D_2.3s ;
```

9.1.3.8 Tageszeit-Konstanten

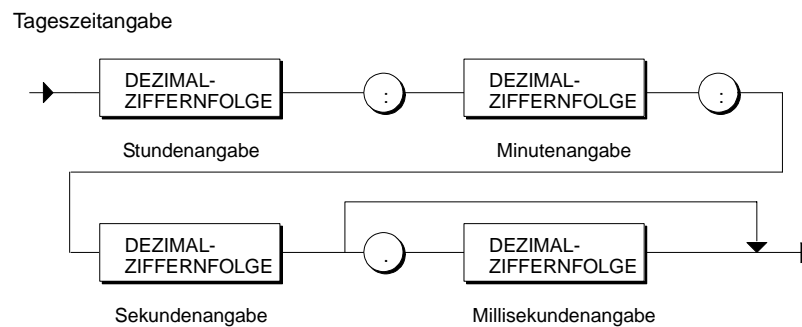
Eine Tageszeit wird durch die Präfixe TIME_OF_DAY# oder TOD# eingeleitet.

Syntax



Die Tageszeitangabe erfolgt durch die Angabe der Stunden, Minuten und der Sekunden, die durch Doppelpunkt zu trennen sind. Die Angabe der Millisekunden ist optional. Sie wird durch einen Punkt von den anderen Angaben getrennt.

Tageszeitangabe



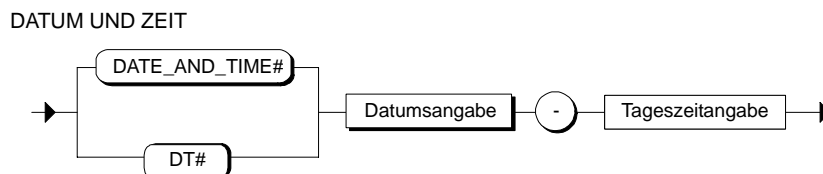
Beispiel

```
UHRZEIT1:= TIME_OF_DAY#12:12:12.2 ;  
UHRZEIT2:= TOD#11:11:11 ;
```


9.1.3.9 Datum- und Zeitkonstante

Eine Datums- und Zeitangabe wird durch die Präfixe DATE_AND_TIME# oder DT# eingeleitet. Sie ist eine aus der Datumsangabe und der Tageszeitangabe zusammengesetzte Konstante.

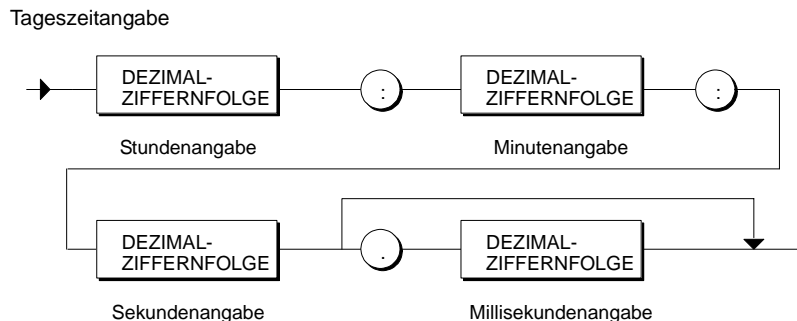
Syntax



Datumsangabe



Tageszeitangabe



Beispiel

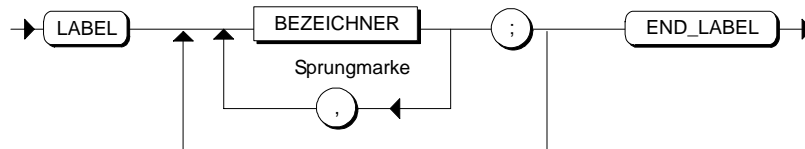
```
UHRZEIT1 := DATE_AND_TIME#1995-01-01-12:12:12.2 ;
UHRZEIT2 := DT#1995-02-02-11:11:11;
```

9.2 Vereinbarung von Sprungmarken

Sprungmarken (Labels) dienen dazu, das Ziel einer GOTO-Anweisung zu definieren. Sie werden im Vereinbarungsteil eines Codebausteins mit ihrem symbolischen Namen vereinbart.

Syntax

Sprungmarkenblock



Beispiel

```
LABEL  
    MARKE1, MARKE2, MARKE3 ;  
END_LABEL
```

10 Globale Daten

10.1 Übersicht über globale Daten

Sie haben in S7-SCL die Möglichkeit, globale Daten anzusprechen. Es gibt zwei Arten von globalen Daten:

- **CPU-Speicherbereiche**

Diese Speicherbereiche sind systemvereinbarte Daten: z. B. Eingänge, Ausgänge und Merker. Die Anzahl der zur Verfügung stehenden CPU-Speicherbereiche ist durch die jeweilige CPU festgelegt.

- **Globale Anwenderdaten als ladbare Datenbausteine**

Diese Datenbereiche liegen innerhalb von Datenbausteinen. Um sie benutzen zu können, müssen Sie vorher die Datenbausteine erstellen und darin die Daten vereinbaren. Im Fall von Instanz-Datenbausteinen werden sie von Funktionsbausteinen abgeleitet und automatisch erzeugt.

Zugriff auf globale Daten

Der Zugriff auf die globalen Daten kann auf folgende Arten erfolgen:

- **absolut:** Über Operandenkennzeichen und absolute Adresse.
- **symbolisch:** Über ein Symbol, das Sie vorher in der Symboltabelle definiert haben.
- **indiziert:** Über Operandenkennzeichen und Feldindex.
- **strukturiert:** Über eine Variable.

Zugriffsart	CPU-Speicherbereiche	Globale Anwenderdaten
absolut	Ja	ja
symbolisch	Ja	ja
indiziert	Ja	ja
strukturiert	Nein	ja

10.2 Speicherbereiche der CPU

10.2.1 Übersicht der Speicherbereiche der CPU

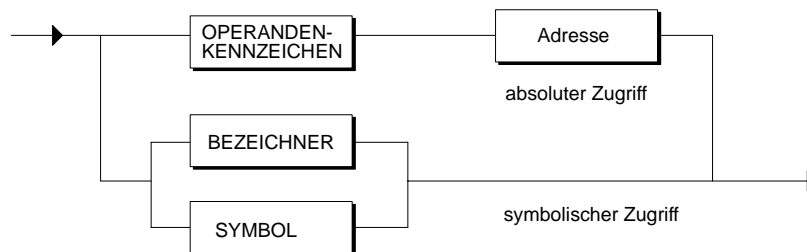
Die Speicherbereiche einer CPU sind systemvereinbarte Bereiche. Deshalb müssen diese Bereiche in Ihrem Codebaustein nicht vereinbart werden. Jede CPU stellt folgende Speicherbereiche mit einem eigenen Adressraum zur Verfügung:

- Ein- / Ausgänge im Prozessabbild (z.B. A1.0)
- Peripherie Ein- / Ausgänge (z.B. PA1.0)
- Merker (z.B. M1.0)
- Zeitglieder, Zähler (C1)

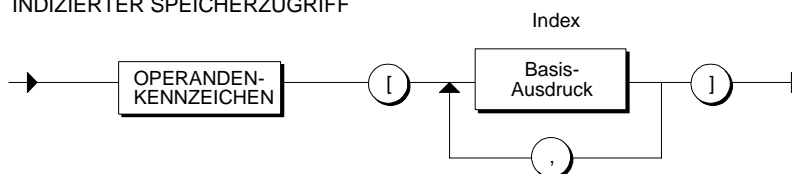
Syntax für den Zugriff

- Der Zugriff auf einen CPU-Speicherbereich erfolgt in einer Wertzuweisung im Anweisungsteil eines Codebausteins: mit einem einfachen Zugriff, den Sie absolut oder symbolisch angeben können, oder
- mit einem indizierten Zugriff.

EINFACHER SPEICHERZUGRIFF

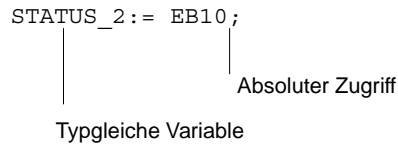


INDIZIERTER SPEICHERZUGRIFF



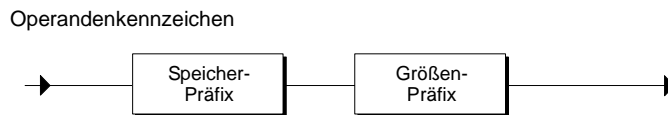
10.2.2 Absoluter Zugriff auf Speicherbereiche der CPU

Der absolute Zugriff auf einen Speicherbereich der CPU erfolgt z.B. über eine Wertzuweisung eines Absolutbezeichners an eine typgleiche Variable.



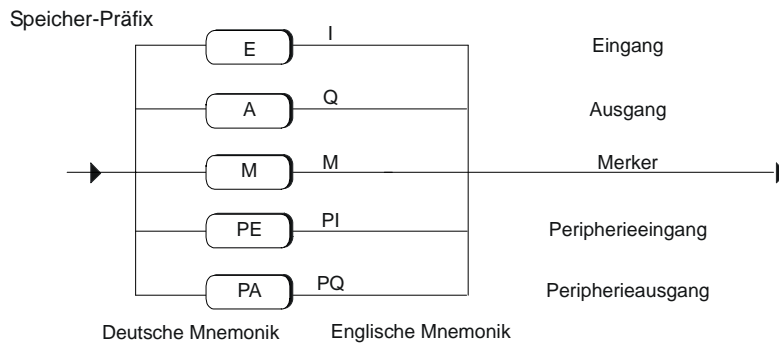
Der Absolutbezeichner verweist auf einen Speicherbereich innerhalb der CPU. Sie spezifizieren diesen Bereich, indem Sie das Operandenkennzeichen (hier EB) gefolgt von der Adresse (hier 10) angeben.

Syntax des Absolutbezeichners



Speicher-Präfix

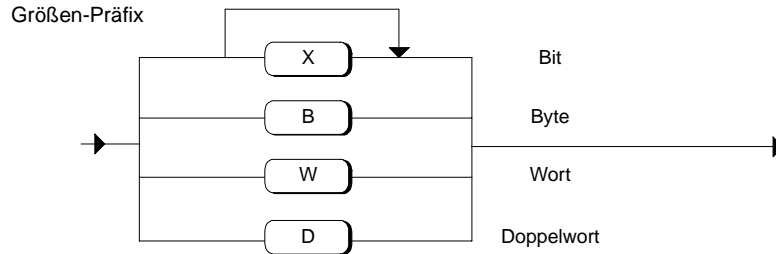
Mit dem Speicher-Präfix legen Sie die Art der zu adressierenden Speicherbereiche fest.



Abhängig von der eingestellten Mnemonik haben entweder die deutschen oder die englischen-Operandenkennzeichen eine reservierte Bedeutung.

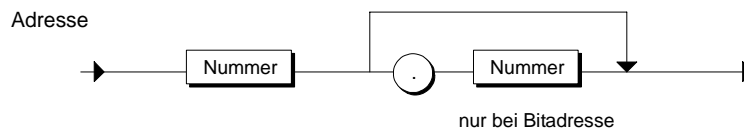
Größen-Präfix

Durch Angabe des Größen-Präfix spezifizieren Sie die Länge des Speicherbereiches, der aus der Peripherie gelesen werden soll. Sie können z.B. ein Byte oder ein Wort lesen. Die Angabe des Größen-Präfix ist optional, wenn Sie lediglich ein Bit spezifizieren möchten.



Adresse

Bei der Angabe der Adresse wird zunächst die absolute Byteadresse angegeben und anschließend, durch Punkt getrennt, die Bitadresse des betreffenden Bytes. Die Angabe der Bitadresse ist optional.



Beispiele

```
STATUSBYTE      :=EB10;
STATUS_3        :=E1.1;
MESSWERT        :=EW20;
```

10.2.3 Symbolischer Zugriff auf Speicherbereiche der CPU

Die symbolische Adressierung ermöglicht es Ihnen, anstelle eines Absolutbezeichners symbolische Namen für die Adressierung der CPU-Speicherbereiche zu nutzen.

Die Zuordnung der symbolischen Namen zu den jeweiligen Operanden Ihres Anwenderprogrammes nehmen Sie in der Symboltabelle vor. Diese können Sie mit dem Menübefehl **Extras > Symboltabelle** jederzeit aus S7-SCL öffnen, um weitere Symbole hinzuzufügen.

Als Datentypangabe sind alle elementaren Datentypen zugelassen, sofern sie die Zugriffsbreite aufnehmen können. Folgende Tabelle stellt beispielhaft eine Symboltabelle dar.

Symbol	Absolute Adresse	Datentyp	Kommentar
Motorkontakt_1	E 1.7	BOOL	Kontaktschalter 1 für Motor A
Eingang1	EW10	INT	Statuswort

Zugriff

Der Zugriff erfolgt über eine Wertzuweisung an eine typgleiche Variable mit der vereinbarten Symbolik.

Beispiel

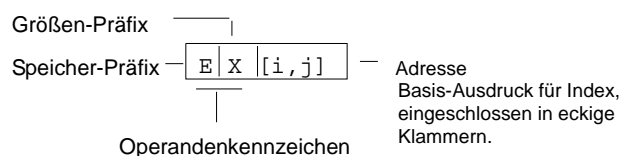
```
MESSWERT_1 := Motorkontakt_1;  
Status_Motor1 := Eingang1 ;
```

10.2.4 Indizierter Zugriff auf Speicherbereiche der CPU

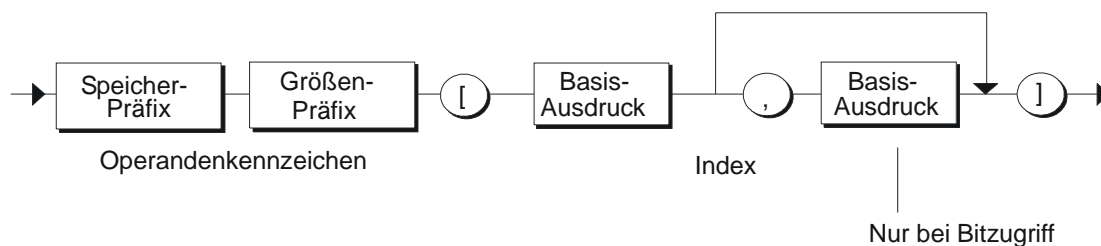
Sie haben auch die Möglichkeit, auf Speicherbereiche der CPU indiziert zuzugreifen. Dies hat gegenüber der absoluten Adressierung den Vorteil, dass Sie die Zugriffe durch die Verwendung von variablen Indizes dynamisch adressieren können. So können Sie z.B. die Laufvariable einer FOR-Schleife als Adresse verwenden.

Der indizierte Zugriff auf einen Speicherbereich geschieht ähnlich wie der absolute Zugriff. Er unterscheidet sich nur in der Angabe der Adresse. Anstelle der Adresse wird ein Index spezifiziert, der eine Konstante, eine Variable oder ein arithmetischer Ausdruck sein kann.

Der Absolutbezeichner setzt sich beim indizierten Zugriff aus dem Operandenkennzeichen (Speicher-Präfix und Größen-Präfix) sowie einem Basis-Ausdruck für das Indizieren zusammen.



Syntax des Absolutbezeichners



Die Indizierung (Basisausdruck) muss den folgenden Regeln entsprechen:

- Jeder Index muss dabei ein arithmetischer Ausdruck vom Datentyp INT sein
- Bei einem Zugriff, der vom Datentyp BYTE, WORD oder DWORD ist, müssen Sie genau einen Index verwenden. Der Index wird als Byteadresse interpretiert. Die Zugriffsbreite wird durch das Größen-Präfix festgelegt.
- Bei einem Zugriff, der vom Datentyp BOOL ist, müssen Sie zwei Indizes benutzen. Der erste Index spezifiziert die Byteadresse, der zweite Index die Bitposition innerhalb des Bytes.

Beispiel

```
MESSWERT_1 :=EW[ZAHELER] ;
AUSMARKE :=E[BYTENR, BITNR] ;
```


10.3 Datenbausteine

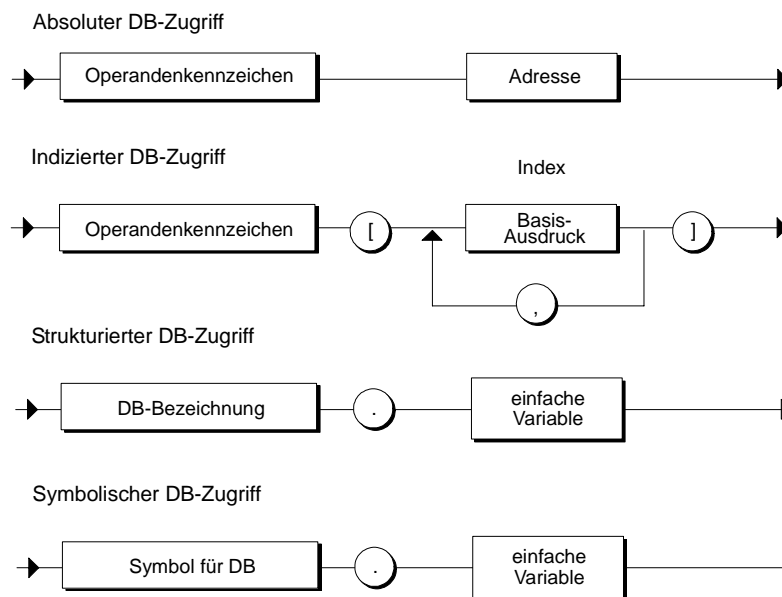
10.3.1 Übersicht der Datenbausteine

Innerhalb von Datenbausteinen können Sie für Ihren Anwendungsfall alle Daten speichern und verarbeiten, deren Gültigkeitsbereich sich auf das ganze Programm bzw. auf das ganze Projekt beziehen sollen. Jeder Codebaustein kann auf die globalen Anwenderdaten lesend oder schreibend zugreifen.

Zugriff

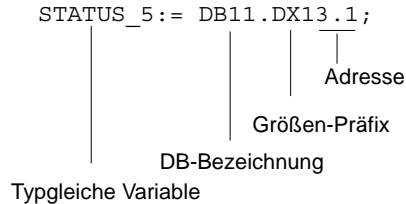
Auf die Daten eines globalen Datenbausteins können Sie auf folgende Arten zugreifen:

- absolut bzw. einfach,
- strukturiert,
- indiziert.



10.3.2 Absoluter Zugriff auf Datenbausteine

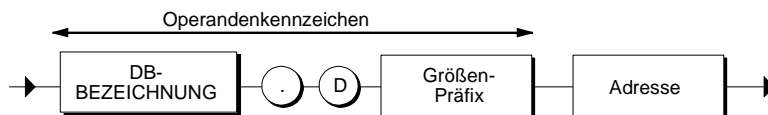
Der absolute Zugriff auf einen Datenbaustein erfolgt wie bei den Speicherbereichen der CPU über eine Wertzuweisung an eine typgleiche Variable. Nach der Angabe der DB-Bezeichnung folgt das Schlüsselwort "D" mit Angabe des Größen-Präfixes (z.B. X für Bit) und der Byteadresse (z.B. 13.1).



Syntax

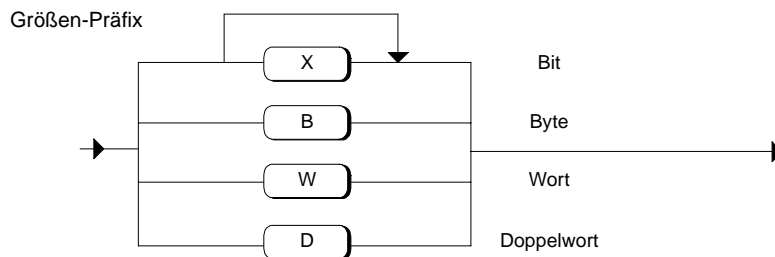
Sie spezifizieren den Zugriff, indem Sie die DB-Bezeichnung zusammen mit dem Größen-Präfix und der Adresse angeben

Absoluter DB-Zugriff



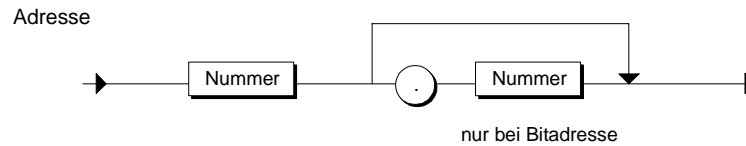
Größen-Präfix

Der Größen-Präfix gibt die Länge des Speicherbereiches im Datenbaustein an, der angesprochen werden soll. Sie können z.B. ein Byte oder ein Wort aus dem DB lesen. Die Angabe des Größen-Präfix ist optional, wenn Sie lediglich ein Bit spezifizieren möchten.



Adresse

Bei der Angabe der Adresse wird zunächst die absolute Byteadresse angegeben und anschließend, durch einen Punkt getrennt, die Bitadresse (nur bei Bitzugriff) des betreffenden Bytes.



Beispiel

Einige Beispiele für den absoluten Zugriff auf einen Datenbaustein. Der Datenbaustein selbst wird im ersten Teil absolut und im zweiten symbolisch angegeben:

```
STATUSBYTE      :=DB101.DB10;
STATUS_3 :=DB30.D1.1;
MESSWERT :=DB25.DW20;
```

```
STATUSBYTE      :=Statusdaten.DB10;
STATUS_3 :="Neue Daten".D1.1;
MESSWERT :=Messdaten.DW20.DW20;
```

```
STATUS_1 :=WORD_TO_BLOCK_DB (INDEX).DW10;
```

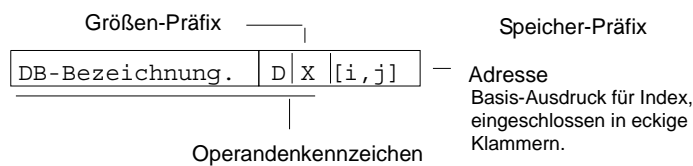
10.3.3 Indizierter Zugriff auf Datenbausteine

Sie haben auch die Möglichkeit, auf Datenbausteine indiziert zuzugreifen. Dies hat gegenüber der absoluten Adressierung den Vorteil, dass Sie Operanden adressieren können, deren Adresse erst zur Laufzeit feststeht. Sie können also zum Beispiel die Laufvariable einer FOR-Schleife als Adresse verwenden.

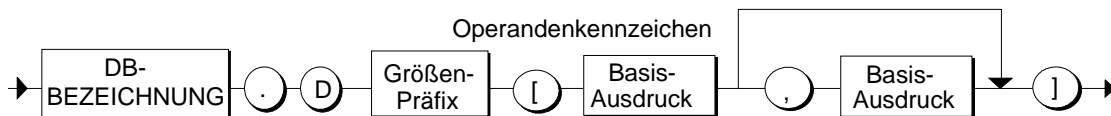
Der indizierte Zugriff auf einen Datenbaustein geschieht ähnlich wie der absolute Zugriff. Er unterscheidet sich nur in der Angabe der Adresse.

Anstelle der Adresse wird ein Index spezifiziert, der eine Konstante, eine Variable oder ein arithmetischer Ausdruck sein kann.

Der indizierte Zugriff setzt sich aus der DB-Bezeichnung, dem Operandenkennzeichen (Schlüsselwort "D" und Größen-Präfix) sowie einem Basis-Ausdruck für das Indizieren zusammen.



Syntax



Die Indizierung muss den folgenden Regeln entsprechen:

- Bei einem Zugriff, der vom Datentyp BYTE, WORD oder DWORD ist, müssen Sie genau einen Index verwenden. Der Index wird als Byteadresse interpretiert. Die Zugriffsbreite wird durch das Größen-Präfix festgelegt.
- Bei einem Zugriff, der vom Datentyp BOOL ist, müssen Sie zwei Indizes benutzen. Der erste Index spezifiziert die Byteadresse, der zweite Index die Bitposition innerhalb des Bytes.
- Jeder Index muss dabei ein arithmetischer Ausdruck vom Datentyp INT (0 - 32767) sein

Beispiel

```

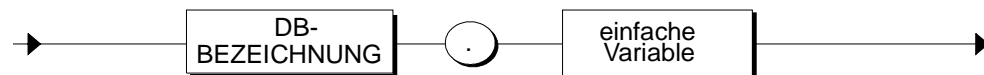
STATUS_1 := DB11.DW[ZAEHLER];
STATUS_2 := DB12.DX[WNR, BITNR];
STATUS_1 := Datenbasis1.DW[ZAEHLER];
STATUS_2 := Datenbasis2.DX[WNR, BITNR];
STATUS_1 := WORD_TO_BLOCK_DB(INDEX).DW[ZAEHLER];
    
```

10.3.4 Strukturierter Zugriff auf Datenbausteine

Der strukturierte Zugriff erfolgt über den Bezeichner der im Datenbaustein vereinbarten Variablen. Sie können die Variable jeder typgleichen Variablen zuweisen.

Die Variable in dem Datenbaustein referenzieren Sie, indem Sie den DB-Namen und, getrennt durch einen Punkt, den Namen der einfachen Variablen angeben.

Syntax



Die einfache Variable steht hierbei für eine Variable, der Sie bei der DB-Vereinbarung einen elementaren oder einen zusammengesetzten Datentyp zugeordnet haben.

Wird ein Parameter vom Typ BLOCK_DB oder das Ergebnis der Konvertierungsfunktion WORD_TO_BLOCK_DB als Einleitung für einen Zugriff auf einen Datenbaustein benutzt, so ist kein strukturierter, sondern nur ein absoluter oder indizierter Zugriff möglich.

Beispiel

Im Vereinbarungsteil des FB10:

```

VAR
Ergebnis:   STRUCT ERG1 : INT;
ERG2 : WORD;
END_STRUCT
END_VAR
  
```

Anwenderdefinierter Datentyp UDT1

```

TYPE UDT1   STRUCT ERG1 : INT;
ERG2 : WORD;
END_STRUCT
  
```

DB20 mit anwenderdefiniertem Datentyp:

```

DB20
UDT1
BEGIN ...
  
```

DB30 ohne anwenderdefinierten Datentyp:

```

DB30   STRUCT ERG1 : INT;
ERG2 : WORD;
END_STRUCT
BEGIN ...
  
```

Funktionsbaustein mit den Zugriffen:

```

..
FB10.DB10();
ERGWORT_A := DB10.Ergebnis.ERG2;
ERGWORT_B := DB20.ERG2;
ERGWORT_C := DB30.ERG2;
  
```


11 Ausdrücke, Operationen und Operanden

11.1 Übersicht der Ausdrücke, Operationen und Operanden

Ein Ausdruck steht für einen Wert, der beim Übersetzungsvorgang oder zur Laufzeit des Programms berechnet wird und besteht aus Operanden (z.B. Konstanten, Variablen oder Funktionsaufrufe) und Operationen (z.B. *, /, + oder -).

Die Datentypen der Operanden und die verwendeten Operationen bestimmen den Typ des Ausdrucks. S7-SCL unterscheidet:

- arithmetische Ausdrücke
- Vergleichsausdrücke
- logische Ausdrücke

Die Auswertung des Ausdrucks geschieht in einer bestimmten Reihenfolge. Sie ist festgelegt durch:

- die Priorität der beteiligten Operationen und
- die links-rechts Reihenfolge oder
- bei Operationen gleicher Priorität durch die vorgenommene Klammerung.

Das Ergebnis eines Ausdrucks können Sie:

- einer Variablen zuweisen.
- als Bedingung für eine Kontrollanweisung verwenden.
- als Parameter für den Aufruf einer Funktion oder eines Funktionsbausteins verwenden.

11.2 Operationen

Ausdrücke bestehen aus Operationen und Operanden. Die meisten Operationen von S7-SCL verknüpfen zwei Operanden und werden deshalb als *binär* bezeichnet. Andere Operationen arbeiten mit nur einem Operanden, daher bezeichnet man sie als *unär*.

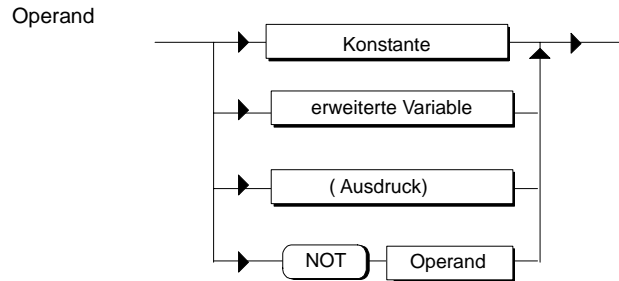
Binäre Operationen werden zwischen die Operanden geschrieben (z. B. A + B). Eine unäre Operation steht immer unmittelbar vor seinem Operand (z. B. -B).

Die in folgender Tabelle gezeigte Priorität der Operationen regelt die Reihenfolge der Berechnung. Die Ziffer "1" entspricht dabei der höchsten Priorität.

Klasse	Operation	Darstellung	Priorität
Zuweisungsoperation:	Zuweisung	: =	11
Arithmetische Operationen:	Potenz	**	2
	Unäre Operationen		
	unäres Plus	+	3
	unäres Minus	-	3
	Arithmetische Basisoperationen		
	Multiplikation	*	4
	Division	/	4
	Modulo-Funktion	MOD	4
	Ganzzahlige Division	DIV	4
	Addition	+	5
Subtraktion	-	5	
Vergleichsoperationen:	Kleiner	<	6
	Größer	>	6
	Kleiner gleich	<=	6
	Größer gleich	>=	6
	Gleichheit	=	7
	Ungleichheit	<>	7
Logische Operationen:	Negation	NOT	3
Logische Basisoperationen			
	Und	AND oder &	8
	Exklusiv-Oder	XOR	9
	Oder	OR	10
Klammerungszeichen:	Klammerung	()	1

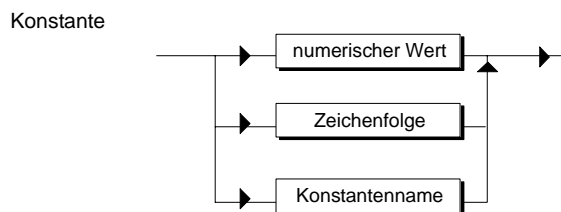
11.3 Operanden

Operanden sind Objekte, mit denen ein Ausdruck gebildet werden kann. Zur Bildung von Operanden sind folgende Elemente erlaubt:



Konstanten

Es können Konstanten mit ihrem numerischen Wert oder mit einem symbolischen Namen oder Zeichenfolgen verwendet werden.



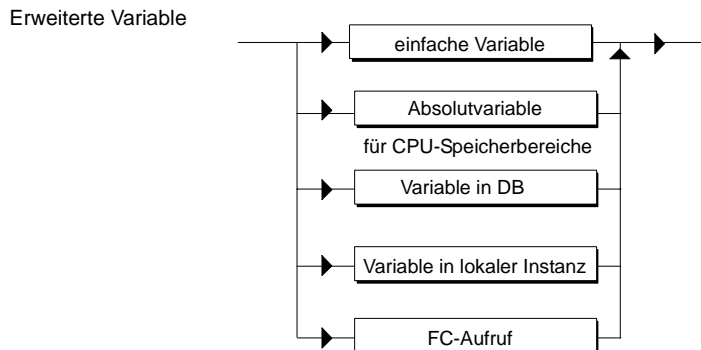
Beispiele für gültige Konstanten sind:

```

4_711
4711
30.0
' ZEICHEN'
FAKTOR
  
```

Erweiterte Variable

Die erweiterte Variable ist ein Oberbegriff für eine Reihe von Variablen, die in Kapitel "Wertzuweisungen" genauer behandelt werden.



Beispiele für gültige Variablen sind:

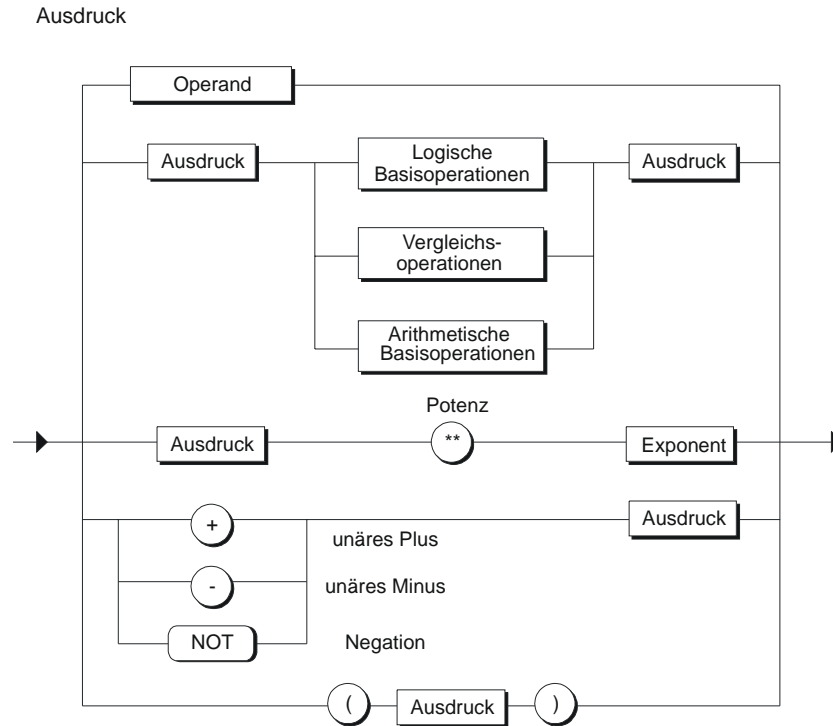
SOLLWERT	einfache Variable
EW10	absolute Variable
E100.5	absolute Variable
DB100.DW [INDEX]	Variable im DB
MOTOR.DREHZAHL	Variable in lokaler Instanz
SQR (20)	Standard-Funktion
FC192 (SOLLWERT)	Funktionsaufruf

Hinweis

Bei einem Funktionsaufruf wird das errechnete Ergebnis, der Rückgabewert, anstelle des Funktionsnamens in den Ausdruck eingefügt. Aus diesem Grund sind VOID-Funktionen, die keinen Rückgabewert haben, als Operanden für einen Ausdruck nicht zulässig.

11.4 Syntax eines Ausdrucks

Syntax



Ergebnis eines Ausdrucks

Das Ergebnis eines Ausdrucks können Sie:

- einer Variablen zuweisen.
- als Bedingung für eine Kontrollanweisung verwenden.
- als Parameter für den Aufruf einer Funktion oder eines Funktionsbausteins verwenden.

Auswertungsreihenfolge

Die Auswertungsreihenfolge eines Ausdrucks ist abhängig von:

- der Priorität der beteiligten Operationen
- der Links-Rechts-Reihenfolge
- der vorgenommenen Klammerung (bei Operationen gleicher Priorität).

Regeln

Die Verarbeitung der Ausdrücke erfolgt nach den folgenden Regeln:

- Ein Operand zwischen zwei Operationen von unterschiedlicher Priorität ist immer an die höherrangige gebunden.
- Die Operationen werden entsprechend der hierarchischen Rangfolge bearbeitet.
- Operationen gleicher Priorität werden von links nach rechts bearbeitet.
- Das Voranstellen eines Minuszeichens vor einen Bezeichner ist gleichbedeutend mit der Multiplikation mit -1.
- Arithmetische Operationen dürfen nicht direkt aufeinander folgen. Deshalb ist der Ausdruck $a * - b$ ungültig, aber $a * (-b)$ erlaubt.
- Das Setzen von Klammerpaaren kann den Operationenvorrang außer Kraft setzen, d.h. die Klammerung hat die höchste Priorität.
- Ausdrücke in Klammern werden als einzelner Operand betrachtet und immer als Erstes ausgewertet.
- Die Anzahl von linken Klammern muss mit der von rechten Klammern übereinstimmen.
- Arithmetische Operationen können nicht in Verbindung mit Zeichen oder logischen Daten angewendet werden. Deshalb sind Ausdrücke wie 'A' + 'B' und $(n \leq 0) + (m > 0)$ falsch.

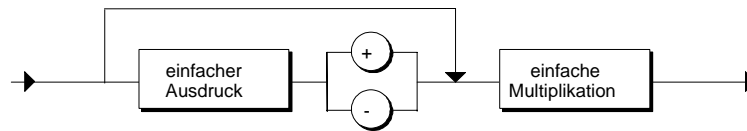
Beispiele für Ausdrücke

```
EB10           // Operand
A1 AND (A2)    // logischer Ausdruck
(A3) < (A4)    // Vergleichsausdruck
3+3*4/2        //arithmetischer Ausdruck
```

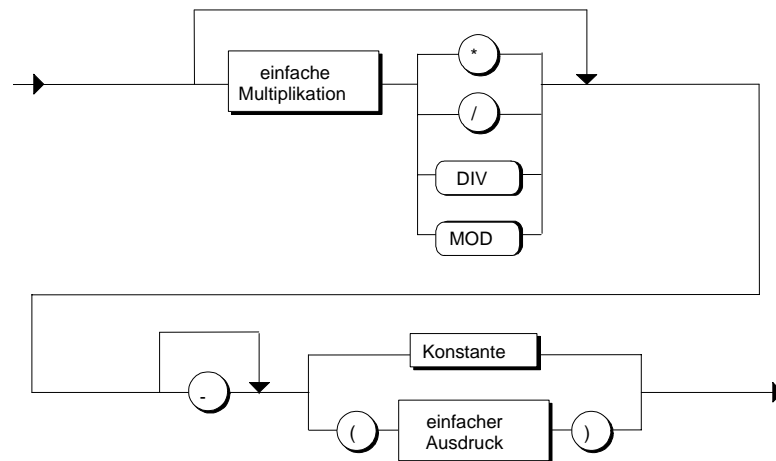
11.5 Einfacher Ausdruck

Unter einem einfachen Ausdruck versteht man in S7-SCL einfache arithmetische Ausdrücke. Sie haben die Möglichkeit, konstante Werte paarweise zu multiplizieren oder zu dividieren und diese Paare durch Addition oder Subtraktion zu verknüpfen.

Syntax des einfachen Ausdrucks:



Syntax der einfachen Multiplikation:



Beispiel:

```
EINF_AUSDRUCK= A * B + D / C - 3 * WERT1;
```

11.6 Arithmetische Ausdrücke

Ein arithmetischer Ausdruck ist ein mit arithmetischen Operationen gebildeter Ausdruck. Diese Ausdrücke ermöglichen die Verarbeitung von numerischen Datentypen.

Die folgende Tabelle fasst die möglichen Operationen zusammen und zeigt, welchem Typ das Ergebnis in Abhängigkeit der Operanden zuzuordnen ist. Dabei werden folgende Abkürzungen verwendet:

ANY_INT	für die Datentypen	INT, DINT
ANY_NUM	für die Datentypen	ANY_INT und REAL

Operation	Kennzeichen	1.Operand	2. Operand	Ergebnis	Priorität
Potenz	**	ANY_NUM	ANY_NUM	REAL	2
unäres Plus	+	ANY_NUM	-	ANY_NUM	3
		TIME	-	TIME	3
unäres Minus	-	ANY_NUM	-	ANY_NUM	3
		TIME	-	TIME	3
Multiplikation	*	ANY_NUM	ANY_NUM	ANY_NUM	4
		TIME	ANY_INT	TIME	4
Division	/	ANY_NUM	ANY_NUM	ANY_NUM	4
		TIME	ANY_INT	TIME	4
Integer-Division	DIV	ANY_INT	ANY_INT	ANY_INT	4
		TIME	ANY_INT	TIME	4
Modulo-Division	MOD	ANY_INT	ANY_INT	ANY_INT	4
Addition	+	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	5
		TOD	TIME	TOD	5
		DT	TIME	DT	5
Subtraktion	-	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	5
		TOD	TIME	TOD	5
		DATE	DATE	TIME	5
		TOD	TOD	TIME	5
		DT	TIME	DT	5
		DT	DT	TIME	5

Hinweis

Der Datentyp des Ergebnisses einer Division (/) wird durch den Datentyp des höchstwertigen Operanden bestimmt.

Werden z.B. zwei Operanden vom Datentyp INT dividiert, so ist das Ergebnis auch vom Datentyp INT (d.h. $10/3=3$ während $10.0/3=3.33$).

Regeln

Operationen in arithmetischen Ausdrücken werden in der Reihenfolge ihrer Priorität behandelt.

- Es empfiehlt sich, negative Zahlen wegen der Übersichtlichkeit in Klammern zu setzen, auch dort wo es nicht notwendig ist.
- Bei Divisionen mit zwei ganzzahligen Operanden vom Datentyp INT liefern die Operationen „DIV“ und „/“ dasselbe Ergebnis (siehe folgendes Beispiel).
- Die Divisionsoperationen „/“, „MOD“ und „DIV“ erfordern, dass der zweite Operand ungleich Null ist.
- Wenn ein Operand vom Typ INT (Ganzzahl) und der andere vom Typ REAL (Gleitpunktzahl) ist, ist das Ergebnis immer vom Typ REAL.
- Bei der Subtraktion von Daten des Typs DATE_AND_TIME und TIME muss der Operand mit dem Datentyp TIME immer rechts vom Operator „-“ stehen.

Beispiele

```
// Das Ergebnis (11) des arithmetischen Ausdrucks wird
// der Variablen "WERT" zugewiesen
WERT1 := 3 + 3 * 4 / 2 - (7+3) / (-5) ;
// Der WERT des folgenden Ausdrucks ist 1
WERT2 := 9 MOD 2 ;
```

11.7 Logische Ausdrücke

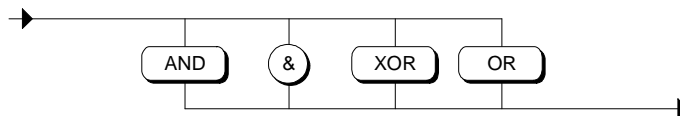
Ein logischer Ausdruck ist ein mit logischen Operationen gebildeter Ausdruck.

Logische Basisoperationen

Mit den Operationen AND, &, XOR und OR können logische Operanden (Typ BOOL) oder Variablen vom Datentyp BYTE, WORD oder DWORD verknüpft werden, um logische Ausdrücke zu bilden. Zur Negation eines logischen Operanden wird die Operation NOT verwendet.

Logischer Basisoperator

NOT ist kein Basisoperator!
Der Operator wirkt wie Vorzeichen.



Logische Operationen

Das Ergebnis des Ausdrucks ist entweder TRUE oder FALSE bei der Verknüpfung von booleschen Operanden oder ein Bitmuster nach der Bitverknüpfung zwischen den beiden Operanden.

Die folgende Tabelle gibt Auskunft über die verfügbaren logischen Ausdrücke und die Datentypen für Ergebnis und Operanden. Dabei werden folgende Abkürzungen verwendet:

ANY_BIT	für die Datentypen	BOOL, BYTE, WORD, DWORD
---------	--------------------	-------------------------

Operation	Kennzeichen	1.Operand	2. Operand	Ergebnis	Priorität
Negation	NOT	ANY_BIT	-	ANY_BIT	3
Konjunktion	AND	ANY_BIT	ANY_BIT	ANY_BIT	8
Exklusiv Disjunktion	XOR	ANY_BIT	ANY_BIT	ANY_BIT	9
Disjunktion	OR	ANY_BIT	ANY_BIT	ANY_BIT	10

Ergebnis

Das Ergebnis eines logischen Ausdrucks ist entweder

- 1 (*true*) oder 0 (*false*) bei der Verknüpfung von booleschen Operanden oder
- ein Bitmuster nach der Bitverknüpfung zwischen den beiden Operanden.

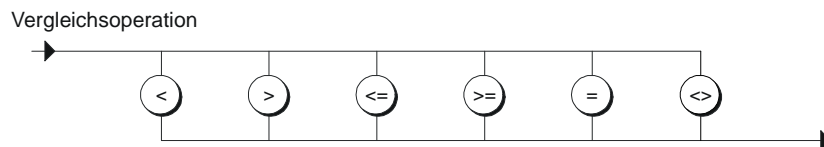
Beispiele

```
// Das Ergebnis des Vergleichsausdrucks wird negiert.  
    IF NOT (ZAEHLER > 5) THEN . . . ;  
// Das Ergebnis des ersten Vergleichsausdrucks wird negiert  
// und mit dem Ergebnis des zweiten konjugiert  
    A := NOT (ZAEHLER1 = 4) AND (ZAEHLER2 = 10) ;  
// Disjunktion zweier Vergleichsausdrücke  
    WHILE (A >= 9) OR (ABFRAGE <> "n") DO.... ;  
// Maskierung eines Eingangsbytes (Bitverknüpfung)  
    Ergebnis := EB10 AND 2#11110000 ;
```

11.8 Vergleichsausdrücke

Die Vergleichsoperationen vergleichen die Werte zweier Operanden und liefern einen booleschen Wert. Das Ergebnis ist TRUE, falls der Vergleich erfüllt ist, und FALSE, falls er nicht erfüllt ist.

Syntax



Regeln

Folgende Tabelle zeigt die vergleichbaren Datentypen und die beim Vergleich geltenden Regeln:

Datentyp	=	<>	>0	<0	>	<	Regeln
INT	✓	✓	✓	✓	✓	✓	Innerhalb der Klasse der numerischen Datentypen sind alle Vergleichsoperatoren erlaubt. Der Operator mit dem mächtigeren Typ bestimmt den Typ der Operation.
DINT	✓	✓	✓	✓	✓	✓	
REAL	✓	✓	✓	✓	✓	✓	
BOOL	✓	✓					Innerhalb der Klasse der Bitdatentypen sind als Vergleichsoperatoren nur GLEICH und UNGLEICH erlaubt. Der Operator mit dem mächtigeren Typ bestimmt den Typ der Operation.
BYTE	✓	✓					
WORD	✓	✓					
DWORD	✓	✓					

Datentyp	=	<>	>0	<0	>	<	Regeln
CHAR	✓	✓	✓	✓	✓	✓	Bei den Zeichen und Zeichenketten wird die Länge der Variablen und der numerische Wert eines jeden ASCII-Zeichen zum Vergleich herangezogen. Der Vergleich bei STRING wird intern durch die Funktionen EQ_STRNG, GE_STRNG, LE_STRNG, GT_STRNG und LT_STRNG der IEC Bibliothek durchgeführt. Folgende Operanden sind für diese Funktionen nicht zulässig: <ul style="list-style-type: none"> • Parameter einer FC. • IN_OUT Parameter eines FB vom Typ STRUCT oder ARRAY.
STRING	✓	✓	✓	✓	✓	✓	
DATE	✓	✓	✓	✓	✓	✓	
TIME	✓	✓	✓	✓	✓	✓	
DT	✓	✓	✓	✓	✓	✓	Der Vergleich bei DT wird intern durch die Funktionen EQ_DT, GE_DT, LE_DT, GT_STRNG und LT_DT der IEC Bibliothek durchgeführt. Folgende Operanden sind für diese Funktionen nicht zulässig: <ul style="list-style-type: none"> • Parameter einer FC. • IN_OUT Parameter eines FB vom Typ STRUCT oder ARRAY.
TOD	✓	✓	✓	✓	✓	✓	
S5-TIME							S5-TIME-Variablen sind als Vergleichsoperanden nicht erlaubt. Eine explizite Konvertierung von S5TIME nach TIME mit Hilfe von IEC-Funktionen ist notwendig.

Verknüpfung von Vergleichen

Vergleichsausdrücke können nach dem Gesetz der booleschen Logik verknüpft werden. So lassen sich Abfragen wie "wenn $a < b$ und $b < c$ dann..." realisieren.

Hierbei werden die einzelnen Operationen in der Reihenfolge ihrer Priorität ausgeführt. Die Priorität kann durch Klammerung beeinflusst werden.

Beispiele

```
// Vergleich 3 KLEINER GLEICH 4. Das Ergebnis
// ist "TRUE"
      A := 3 <= 4
// Vergleich 7 UNGLEICH 7. Das Ergebnis
// ist "FALSE"
      7 <> 7
// Auswertung eines Vergleichsausdrucks in
// einer IF-Verzweigung
      IF ZAEHLER < 5 THEN ....
// Verknüpfung zweier Vergleichsausdrücke
      Wert_A > 20 AND Wert_B < 20
// Verknüpfung zweier Vergleichsausdrücke mit Klammerung
      A<>(B AND C)
```

12 Anweisungen

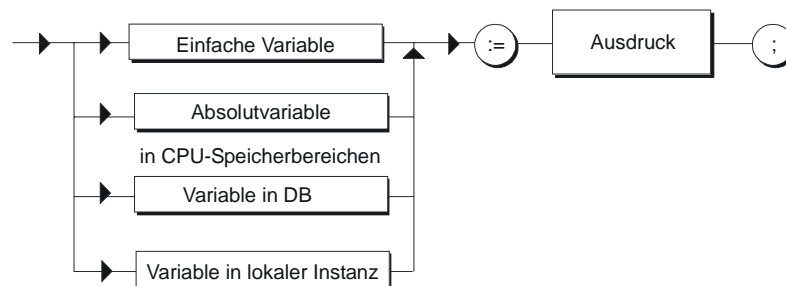
12.1 Wertzuweisungen

Wertzuweisungen ersetzen den momentanen Wert einer Variablen mit einem neuen Wert, der über einen Ausdruck angegeben wird. Dieser Ausdruck kann auch Bezeichner von Funktionen enthalten, die dadurch aktiviert werden und entsprechende Werte zurückliefern (Rückgabewert).

Nach untenstehendem Bild wird der Ausdruck auf der rechten Seite der Wertzuweisung ausgewertet und das Ergebnis in der Variablen abgespeichert, deren Namen auf der linken Seite des Zuweisungszeichens steht. Die dabei zugelassenen Variablen zeigt das Bild.

Syntax der Wertzuweisung:

Wertzuweisung



Der Typ eines Zuweisungsausdrucks ist der Typ des linken Operanden. Die einfache Variable kann eine Variable elementaren oder zusammengesetzten Datentyps sein.

12.1.1 Wertzuweisungen mit Variablen eines elementaren Datentyps

Jeder Ausdruck und jede Variable mit elementarem Datentyp können einer anderen typgleichen Variablen zugewiesen werden.

```
Bezeichner := Ausdruck ;  
Bezeichner := Variable ;
```

Beispiel

```
FUNCTION_BLOCK FB12  
VAR  
    SCHALTER_1      : INT ;  
    SCHALTER_2      : INT ;  
    SOLLWERT_1      : REAL ;  
    SOLLWERT_2      : REAL ;  
    ABFRAGE_1       : BOOL ;  
    ZEIT_1           : S5TIME ;  
    ZEIT_2           : TIME ;  
    DATUM_1          : DATE ;  
    TAGESZEIT_1     : TIME_OF_DAY ;  
END_VAR  
BEGIN  
  
    // Zuweisung einer Konstanten zu einer Variablen  
    SCHALTER_1      := -17 ;  
    SOLLWERT_1      := 100.1 ;  
    ABFRAGE_1       := TRUE ;  
    ZEIT_1           := T#1H_20M_10S_30MS ;  
    ZEIT_2           := T#2D_1H_20M_10S_30MS ;  
    DATUM_1          := D#1996-01-10 ;  
  
    // Zuweisung einer Variablen zu einer Variable  
    SOLLWERT_1      := SOLLWERT_2 ;  
    SCHALTER_2      := SCHALTER_1 ;  
    // Zuweisung eines Ausdrucks zu einer Variablen  
  
    SCHALTER_2      := SCHALTER_1 * 3 ;  
END_FUNCTION_BLOCK
```

12.1.2 Wertzuweisungen mit Variablen vom Typ STRUCT und UDT

Variablen vom Typ STRUCT und UDT sind strukturierte Variablen, die entweder für eine komplette Struktur oder eine Komponente dieser Struktur stehen.

Gültige Angaben für eine Strukturvariable sind:

```
Abbild                //Bezeichner für eine Struktur
Abbild.element       //Bezeichner für Strukturkomponente
Abbild.feld          //Bezeichner eines einfachen Feldes
                    //innerhalb einer Struktur
Abbild.feld[2,5]     //Bezeichner einer Feldkomponente
                    //innerhalb einer Struktur
```

Zuweisung einer kompletten Struktur

Eine gesamte Struktur ist einer anderen Struktur nur dann zuweisbar, wenn die Strukturkomponenten sowohl in ihren Datentypen als auch in ihren Namen übereinstimmen. Gültige Zuweisungen sind:

```
structname_1 := structname_2 ;
```

Zuweisung von Strukturkomponenten

Sie können jeder Strukturkomponente eine typverträgliche Variable, einen typverträglichen Ausdruck oder eine andere Strukturkomponente zuweisen.

Eine Strukturkomponente können Sie referenzieren, indem Sie den Bezeichner der Struktur und den Bezeichner der Strukturkomponente durch einen Punkt getrennt angeben. Gültige Zuweisungen sind:

```
structname_1.element1 := Wert ;
structname_1.element1 := 20.0 ;
structname_1.element1 := structname_2.element1 ;
structname_1.feldname1 := structname_2.feldname2 ;
structname_1.feldname[10] := 100 ;
```

Beispiel

```
FUNCTION_BLOCK FB3
VAR
    HILFSVAR : REAL ;
    MESSWERT : STRUCT //Zielstruktur
        SPANNUNG:REAL ;
        WIDERSTAND:REAL ;
        EINFACHFELD : ARRAY
            [1..2, 1..2] OF INT ;
            END_STRUCT ;
    ISTWERT : STRUCT //Quellstruktur
        SPANNUNG : REAL ;
        WIDERSTAND : REAL ;
        EINFACHFELD : ARRAY
            [1..2, 1..2] OF INT ;
            END_STRUCT ;
END_VAR

BEGIN
//Zuweisung einer kompletten Struktur an eine
//komplette Struktur
    MESSWERT := ISTWERT ;
//Zuweisung einer Strukturkomponente
//zu einer Strukturkomponente
    MESSWERT.SPANNUNG := ISTWERT.SPANNUNG ;
//Zuweisung einer Strukturkomponente zu einer
//typengleichen Variablen
    HILFSVAR := ISTWERT.WIDERSTAND ;
//Zuweisung einer Konstanten zu einer Strukturkomponente
    MESSWERT.WIDERSTAND := 4.5;
//Zuweisung einer Konstanten zu einem einfachen Feld-Element
    MESSWERT.EINFACHFELD[1,2] := 4;
END_FUNCTION_BLOCK
```


12.1.3 Wertzuweisungen mit Variablen vom Typ ARRAY

Ein Feld besteht aus 1 bis max. 6 Dimensionen und enthält Feldelemente, die alle vom selben Typ sind. Für die Zuweisung von Feldern an eine Variable gibt es zwei Zugriffsvarianten: Sie können komplette Felder oder Teilfelder referenzieren.

Zuweisung eines kompletten Feldes

Ein komplettes Feld ist einem anderen Feld zuweisbar, wenn sowohl die Datentypen der Komponenten als auch die Feldgrenzen (kleinst- und größtmögliche Feldindizes) übereinstimmen. Ist dies der Fall, geben Sie nach dem Zuweisungszeichen den Bezeichner des Feldes an. Gültige Zuweisungen sind:

```
feldname_1 := feldname_2 ;
```

Hinweis

Beachten Sie, dass kompletten Feldern keine Konstanten zugewiesen werden dürfen.

Zuweisung einer Feldkomponente

Ein einzelnes Feldelement wird mit dem Feldnamen gefolgt von geeigneten Indexwerten in eckigen Klammern angesprochen. Für jede Dimension steht ein Index zur Verfügung. Sie werden durch Komma getrennt und ebenfalls in eckige Klammern eingeschlossen. Ein Index muss ein arithmetischer Ausdruck vom Datentyp INT sein.

Eine Wertzuweisung für ein zulässiges Teilfeld erhalten Sie, indem Sie in den eckigen Klammern hinter dem Namen des Feldes von rechts beginnend Indizes weglassen. Damit sprechen Sie einen Teilbereich an, dessen Dimensionsanzahl gleich der Anzahl der weggelassenen Indizes ist. Gültige Zuweisungen sind:

```
feldname_1[ i ] := feldname_2[ j ] ;  
feldname_1[ i ] := Ausdruck ;  
bezeichner_1   := feldname_1[ i ] ;
```

Beispiel

```
FUNCTION_BLOCK FB3
VAR
  SOLLWERTE      :ARRAY [0..127] OF INT ;
  ISTWERTE       :ARRAY [0..127] OF INT ;
  // Vereinbarung einer Matrix (=zweidimensionales Feld)
  // mit 3 Zeilen und 4 Spalten
  REGLER : ARRAY [1..3, 1..4] OF INT ;
  // Vereinbarung eines Vektors (=eindimensionales Feld)
  // mit 4 Komponenten
  REGLER_1 : ARRAY [1..4] OF INT ;
END_VAR

BEGIN
  // Zuweisung eines kompletten Feldes zu einem Feld
  SOLLWERTE := ISTWERTE ;
  // Zuweisung eines Vektors zu der zweiten Zeile
  // des Feldes REGLER
  REGLER[2] := REGLER_1 ;
  // Zuweisung einer Feldkomponente zu einer Feldkomponente
  // des Feldes REGLER
  REGLER [1,4] := REGLER_1 [4] ;
END_FUNCTION_BLOCK
```

12.1.4 Wertzuweisungen mit Variablen vom Typ STRING

Eine Variable vom Datentyp STRING enthält eine Zeichenkette von maximal 254 Zeichen. Jeder Variablen vom Datentyp STRING kann eine andere typgleiche Variable zugewiesen werden. Gültige Zuweisungen sind:

```
stringvariable_1 := String-Konstante;  
stringvariable_1 := stringvariable_2 ;
```

Beispiel

```
FUNCTION_BLOCK FB3  
VAR  
    ANZEIGE_1 : STRING[50] ;  
    STRUKTUR1 : STRUCT  
        ANZEIGE_2 : STRING[100] ;  
        ANZEIGE_3 : STRING[50] ;  
    END_STRUCT ;  
END_VAR  
  
BEGIN  
    // Zuweisung einer Konstanten zu einer STRING-Variablen  
    ANZEIGE_1 := 'Fehler in Modul 1' ;  
    // Zuweisung einer Strukturkomponente zu einer  
    // STRING-Variablen.  
    ANZEIGE_1 := STRUKTUR1.ANZEIGE_3 ;  
    // Zuweisung einer STRING-Variable zu einer STRING-Variablen  
    If ANZEIGE_1 <> STRUKTUR1.ANZEIGE_3 THEN  
        ANZEIGE_1 := STRUKTUR1.ANZEIGE_3 ;  
    END_IF ;  
END_FUNCTION_BLOCK
```

12.1.5 Wertzuweisungen mit Variablen vom Typ DATE_AND_TIME

Der Datentyp DATE_AND_TIME definiert einen Bereich mit 64 Bits (8 Byte) für die Angabe von Datum und Uhrzeit. Jeder Variablen vom Datentyp DATE_AND_TIME kann eine andere typgleiche Variable oder Konstante zugewiesen werden. Gültige Zuweisungen sind:

```
dtvariable_1 := Datum und Zeit-Konstante;  
dtvariable_1 := dtvariable_2 ;
```

Beispiel

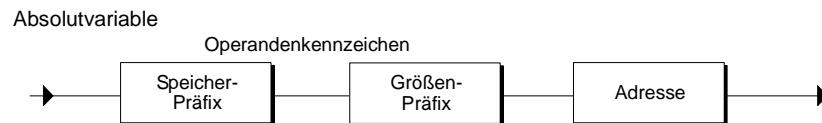
```
FUNCTION_BLOCK FB3  
VAR  
    ZEIT_1      : DATE_AND_TIME ;  
    STRUKTUR1   : STRUCT  
        ZEIT_2 : DATE_AND_TIME ;  
        ZEIT_3 : DATE_AND_TIME ;  
    END_STRUCT ;  
END_VAR  
  
BEGIN  
    // Zuweisung einer Konstanten zu einer  
    // DATE_AND_TIME-Variablen  
    ZEIT_1 := DATE_AND_TIME#1995-01-01-12:12:12.2 ;  
    STRUKTUR1.ZEIT_3 := DT#1995-02-02-11:11:11 ;  
    // Zuweisung einer Strukturkomponente zu einer  
    // DATE_AND_TIME-Variablen.  
    ZEIT_1 := STRUKTUR1.ZEIT_2 ;  
    // Zuweisung einer DATE_AND_TIME-Variable zu einer  
    DATE_AND_TIME-Variablen  
    If ZEIT_1 < STRUKTUR1.ZEIT_3 THEN  
        ZEIT_1 := STRUKTUR1.ZEIT_3 ;  
    END_IF ;  
END_FUNCTION_BLOCK
```

12.1.6 Wertzuweisungen mit Absolutvariablen für Speicherbereiche

Die Absolutvariable referenziert die global gültigen Speicherbereiche einer CPU. Sie haben drei Möglichkeiten, diese Bereiche anzusprechen:

- Absoluter Zugriff
- Indizierter Zugriff
- Symbolischer Zugriff

Syntax der Absolutvariablen



Beispiel

```

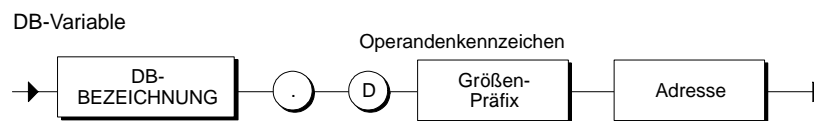
FUNCTION_BLOCK FB3
VAR
    STATUSWORT1    : WORD ;
    STATUSWORT2    : BOOL ;
    STATUSWORT3    : BYTE ;
    STATUSWORT4    : BOOL ;
    ADRESSE        : INT ;
END_VAR
BEGIN
    ADRESSE        := 10 ;
    // Zuweisung eines Eingangsworts an eine Variable
    // (einfacher Zugriff)
    STATUSWORT1    := EW4 ;
    // Zuweisung einer Variable an ein Ausgangsbit
    // (einfacher Zugriff)
    a1.1          := STATUSWORT2 ;
    // Zuweisung eines Eingangsbytes an eine Variable
    // (indizierter Zugriff)
    STATUSWORT3    := EB[ADRESSE] ;
    // Zuweisung eines Eingangsbits an eine Variable
    // (indizierter Zugriff)
    FOR ADRESSE := 0 TO 7 BY 1 DO
        STATUSWORT4 := e[1, ADRESSE] ;
    END_FOR ;
END_FUNCTION_BLOCK
  
```

12.1.7 Wertzuweisungen mit globalen Variablen

Auch der Zugriff auf Daten in Datenbausteinen erfolgt mittels einer Wertzuweisung an typgleiche Variablen oder umgekehrt. Sie können jeder globalen Variablen eine typgleiche Variable oder einen typgleichen Ausdruck zuweisen. Sie haben mehrere Möglichkeiten, diese Daten anzusprechen:

- Strukturierter Zugriff
- Absoluter Zugriff
- Indizierter Zugriff

Syntax der DB-Variable



Beispiel

```
FUNCTION_BLOCK FB3
VAR
  REGLER_1      : ARRAY [1..4] OF INT ;
  STATUSWORT1   : WORD ;
  STATUSWORT2   : ARRAY [0..10] OF WORD ;
  STATUSWORT3   : INT ;
  STATUSWORT4   : WORD ;
  ADRESSE       : INT ;
END_VAR
VAR_INPUT
  ADRESSWORT    : WORD ;
END_VAR
BEGIN
  // Zuweisen des Worts 1 aus DB11
  // an eine Variable (einfacher Zugriff)
  STATUSWORT1 := DB11.DW1 ;
  // Der Feldkomponente in der 1. Zeile und
  // der 1. Spalte der Matrix wird der Wert
  // der Variablen "ZAHL" zugewiesen (strukturiertes Zugriff):
  REGLER_1[1] := DB11.ZAHL ;
  // Zuweisen der Strukturkomponente "ZAHL2"
  // der Struktur "ZAHL1" an die Variable Statuswort3
  STATUSWORT3 := DB11.ZAHL1.ZAHL2 ;
  // Zuweisen eines Worts mit Indexadresse
  // aus DB11 an eine Variable (indizierter Zugriff)
  FOR
    ADRESSE := 1 TO 10 BY 1 DO
      STATUSWORT2[ADRESSE] := DB11.DW[ADRESSE] ;
      // Hier werden der Eingangsparameter ADRESSWORT
      // als Nummer des DBs und
      // der Index ADRESSE zur Angabe der Wortadresse
      // innerhalb des DBs verwendet.
      STATUSWORT4 :=
WORD_TO_BLOCK_DB(ADRESSWORT).DW[ADRESSE] ;
    END_FOR ;
END_FUNCTION_BLOCK
```

12.2 Kontrollanweisungen

12.2.1 Übersicht der Kontrollanweisungen

Auswahanweisungen

Mit einer Auswahanweisung haben Sie die Möglichkeit, den Programmfluss in Abhängigkeit von Bedingungen in verschiedene Anweisungsfolgen zu verzweigen.

Verzweigungsart	Funktion
IF-Anweisung	Mit der IF-Anweisung können Sie den Programmfluss in Abhängigkeit von einer Bedingung, die entweder TRUE oder FALSE ist, in eine von zwei Alternativen verzweigen.
CASE-Anweisung	Mit einer CASE-Anweisung können Sie den Programmfluss im Sinne einer 1:n-Verzweigung steuern, indem Sie eine Variable einen Wert aus n möglichen annehmen lassen.

Schleifenbearbeitung

Die Schleifenbearbeitung können Sie mit Hilfe von Wiederholungsanweisungen steuern. Eine Wiederholungsanweisung gibt an, welche Teile eines Programms, in Abhängigkeit von Bedingungen, wiederholt werden soll.

Verzweigungsart	Funktion
FOR-Anweisung	dient zur Wiederholung einer Folge von Anweisungen, solange die Laufvariable innerhalb des angegebenen Wertebereichs liegt.
WHILE-Anweisung	dient zur Wiederholung einer Folge von Anweisungen, solange eine Durchführungsbedingung erfüllt ist.
REPEAT-Anweisung	dient zur Wiederholung einer Folge von Anweisungen, bis eine Abbruchbedingung erfüllt ist.

Programmsprung

Ein Programmsprung bewirkt den sofortigen Sprung zu einem angegebenen Sprungziel und damit zu einer anderen Anweisung innerhalb desselben Bausteins.

Verzweigungsart	Funktion
CONTINUE-Anweisung	dient zum Abbruch der Ausführung des momentanen Schleifendurchlaufes.
EXIT-Anweisung	dient zum Verlassen einer Schleife an beliebiger Stelle und unabhängig vom Erfülltsein der Abbruchbedingung.
GOTO -Anweisung	bewirkt den sofortigen Sprung zu einer angegebenen Sprungmarke.
RETURN-Anweisung	bewirkt das Verlassen eines aktuellen bearbeiteten Bausteins.

12.2.2 Bedingungen

Die Bedingung ist entweder ein Vergleichsausdruck, ein logischer Ausdruck oder ein arithmetischer Ausdruck. Sie ist vom Typ BOOL und kann die Werte TRUE oder FALSE annehmen. Arithmetische Ausdrücke sind TRUE, falls sie einen Wert ungleich Null annehmen und FALSE, wenn sie Null ergeben. Folgende Tabelle zeigt Beispiele für Bedingungen:

Typ	Beispiel
Vergleichsausdruck	TEMP > 50 ZAEHLER <= 100 CHAR1 < 'S'
Vergleichs- und logischer Ausdruck	(ALPHA <> 12) AND NOT BETA
Boolescher Operand	E1.1
Arithmetischer Ausdruck	ALPHA = (5 + BETA)

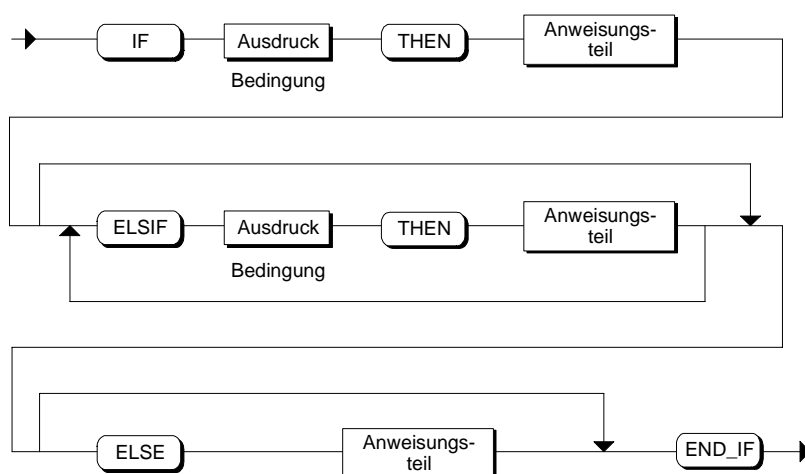
12.2.3 IF-Anweisung

Die IF-Anweisung ist eine bedingte Anweisung. Sie bietet eine oder mehrere Optionen und wählt eine (gegebenenfalls auch keine) ihrer Anweisungsteile zur Ausführung an.

Die Ausführung der bedingten Anweisung bewirkt die Auswertung der angegebenen logischen Ausdrücke. Ist der Wert eines Ausdrucks TRUE, so gilt die Bedingung erfüllt, bei FALSE als nicht erfüllt.

Syntax

IF-Anweisung



Die IF-Anweisung wird nach den folgenden Regeln bearbeitet:

- Die erste Anweisungsfolge, deren logischer Ausdruck = TRUE ist, kommt zur Ausführung. Die restlichen Anweisungsfolgen kommen nicht zur Ausführung.
- Falls kein boolescher Ausdruck = TRUE ist, wird die Anweisungsfolge bei ELSE ausgeführt (oder keine Anweisungsfolge, falls der ELSE-Zweig nicht vorhanden ist).
- Es dürfen beliebig viele ELSIF-Anweisungen vorhanden sein.

Hinweis

Die Verwendung eines oder mehrerer ELSIF-Zweige bietet gegenüber einer Sequenz von IF-Anweisungen den Vorteil, dass die einem gültigen Ausdruck folgenden logischen Ausdrücke nicht mehr ausgewertet werden. Die Laufzeit eines Programms lässt sich so verkürzen.

Beispiel

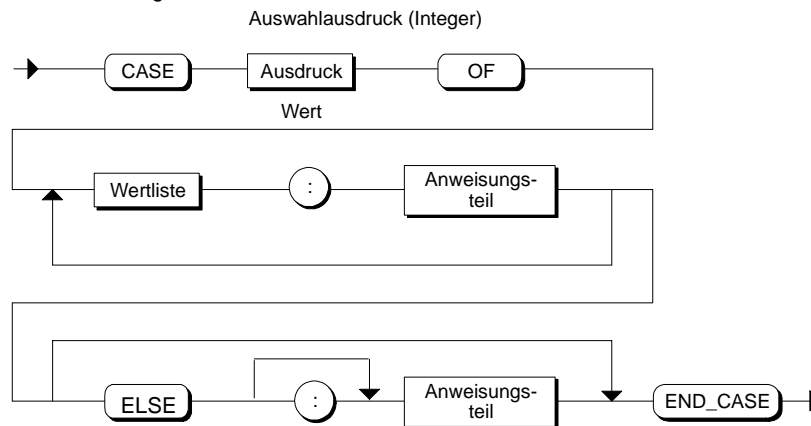
```
IF E1.1 THEN
    N := 0 ;
    SUM := 0 ;
    OK := FALSE ; // OK-Flag auf FALSE setzen
ELSIF START = TRUE THEN
    N := N + 1 ;
    SUM := SUM + N ;
ELSE
    OK := FALSE ;
END_IF ;
```

12.2.4 CASE-Anweisung

Die CASE-Anweisung dient der Auswahl unter 1-n alternativen Programmteilen. Diese Auswahl beruht auf dem laufenden Wert eines Auswahl-Ausdrucks.

Syntax

CASE-Anweisung

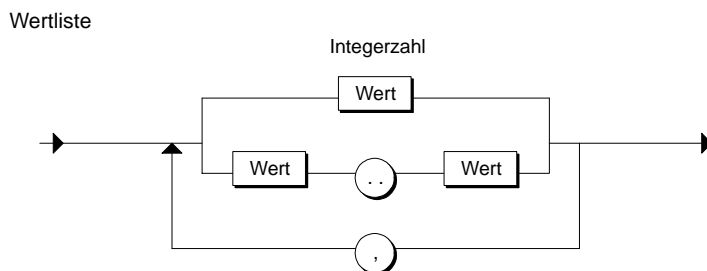


Die CASE-Anweisung wird nach folgenden Regeln bearbeitet

- Der Auswahl-Ausdruck muss einen Wert vom Typ INTEGER liefern.
- Bei der Abarbeitung der CASE-Anweisung wird überprüft, ob der Wert des Auswahl-Ausdrucks in einer angegebenen Wertliste enthalten ist. Bei Übereinstimmung wird der der Liste zugeordnete Anweisungsteil ausgeführt.
- Ergibt der Vergleichsvorgang keine Übereinstimmung, so wird der Anweisungsteil nach ELSE ausgeführt oder keine Anweisung, falls der ELSE-Zweig nicht vorhanden ist.

Wertliste

Die Wertliste enthält die erlaubten Werte für den Auswahl-Ausdruck.

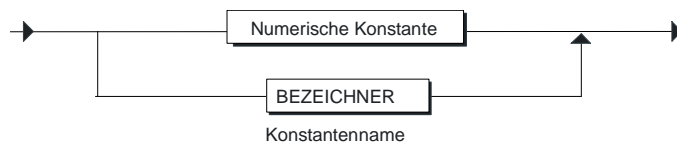


Es gelten folgende Regeln:

- Jede Wertliste beginnt mit einer Konstanten, einer Konstantenliste oder einem Konstantenbereich.
- Die Werte innerhalb der Wertliste müssen vom Typ INTEGER sein.
- Jeder Wert darf nur einmal vorkommen.

Wert

Der Wert folgt untenstehender Syntax:



Beispiel

```

CASE TW OF
  1 :      DISPLAY:= OVEN_TEMP;
  2 :      DISPLAY:= MOTOR_SPEED;
  3 :      DISPLAY:= GROSS_TARE;
          AW4:= 16#0003;
  4..10:   DISPLAY:= INT_TO_DINT (TW);
          AW4:= 16#0004;
  11,13,19: DISPLAY:= 99;
          AW4:= 16#0005;
ELSE:
  DISPLAY:= 0;
  TW_ERROR:= 1;
END_CASE ;

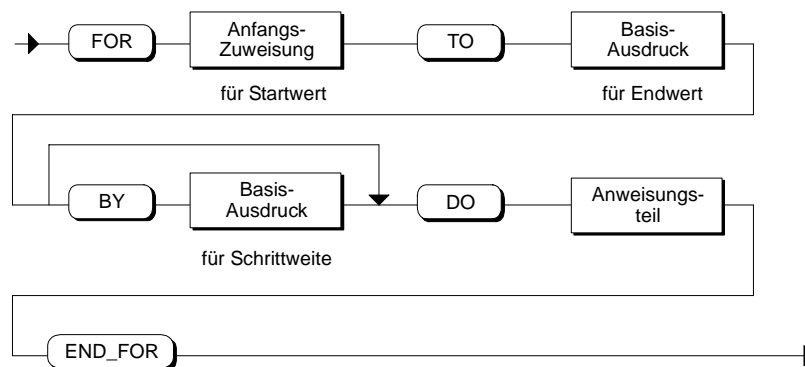
```

12.2.5 FOR-Anweisung

Eine FOR-Anweisung dient zur Wiederholung einer Anweisungsfolge, solange eine Laufvariable innerhalb des angegebenen Wertebereichs liegt. Die Laufvariable muss der Bezeichner einer lokalen Variablen vom Typ INT oder DINT sein. Die Definition einer Schleife mit FOR schließt die Festlegung eines Start- und eines Endwertes mit ein. Beide Werte müssen typgleich mit der Laufvariablen sein.

Syntax

FOR-Anweisung



Die FOR-Anweisung wird wie folgt bearbeitet:

- Beim Start der Schleife wird die Laufvariable auf den Startwert (Anfangszuweisung) gesetzt und nach jedem Schleifendurchlauf um die angegebene Schrittweite erhöht (positive Schrittweite) oder erniedrigt (negative Schrittweite), solange bis der Endwert erreicht ist.
- Nach jedem Durchlauf wird überprüft, ob die Bedingung (Variable liegt innerhalb des Wertebereichs) erfüllt ist. Bei ja wird die Anweisungsfolge ausgeführt, andernfalls wird die Schleife und damit die Anweisungsfolge übersprungen.

Regeln

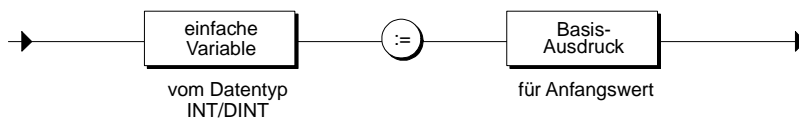
Folgende Regeln sind bei der Formulierung von FOR-Anweisungen zu beachten:

- Die Laufvariable darf nur vom Datentyp INT oder DINT sein.
- Die Angabe von BY [Schrittweite] kann entfallen. Ist keine Schrittweite spezifiziert, dann beträgt sie +1.

Anfangszuweisung

Die Bildung des Startwerts der Laufvariablen muss der folgenden Syntax entsprechen. Die einfache Variable auf der linken Seite der Zuweisung muss vom Datentyp INT oder DINT sein.

Anfangszuweisung

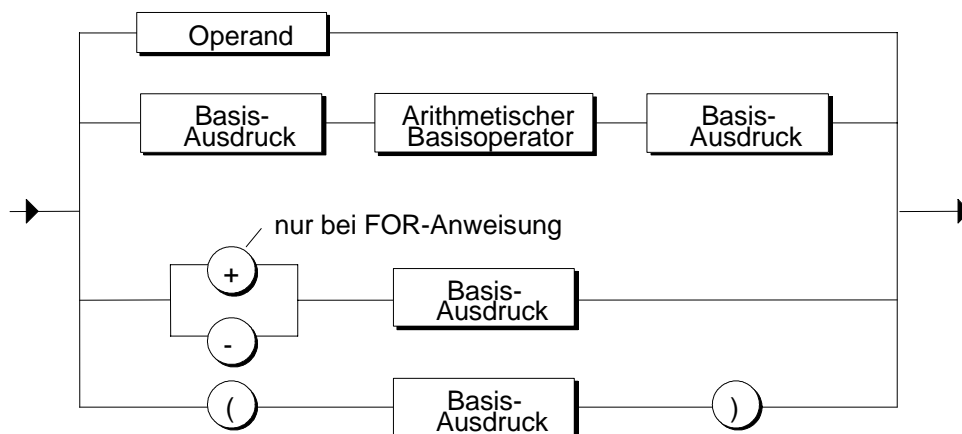


Beispiele für gültige Anfangszuweisungen sind:

```
FOR I := 1 TO 20
FOR I := 1 TO (ANFANG + J)
```

Endwert und Schrittweite

Für die Bildung des Endwertes und der gewünschten Schrittweite können Sie jeweils einen Basis-Ausdruck bilden. Die Bildung des Basis-Ausdrucks muss der folgenden Syntax entsprechen:



- Die Angabe von *BY [Schrittweite]* kann entfallen. Ist keine Schrittweite spezifiziert, dann beträgt sie +1.
- Anfangswert, Endwert und Schrittweite sind Ausdrücke (siehe Kapitel "Ausdrücke, Operationen, Operanden"). Die Auswertung erfolgt einmalig am Beginn der Ausführung der FOR-Anweisung.
- Eine Veränderung der beiden Werte für Endwert und Schrittweite während der Ausführung der Schleife ist nicht erlaubt.

Beispiel

```
FUNCTION_BLOCK FOR_BSP
VAR
    INDEX: INT ;
    KENNWORT: ARRAY [1..50] OF STRING;
END_VAR
BEGIN
    FOR INDEX := 1 TO 50 BY 2 DO
        IF KENNWORT [INDEX] = 'KEY' THEN
            EXIT;
        END_IF;
    END_FOR;

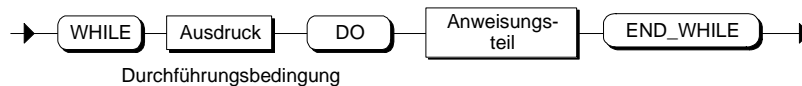
END_FUNCTION_BLOCK
```


12.2.6 WHILE-Anweisung

Die WHILE-Anweisung erlaubt die wiederholte Ausführung einer Anweisungsfolge unter der Kontrolle einer Durchführungsbedingung. Die Durchführungsbedingung wird nach den Regeln eines logischen Ausdrucks gebildet.

Syntax

WHILE-Anweisung



Die WHILE-Anweisung wird nach folgenden Regeln bearbeitet:

- Vor jeder Ausführung des Anweisungsteils wird die Durchführungsbedingung ausgewertet (abweisende Schleife).
- Der auf DO folgende Anweisungsteil wird solange wiederholt, wie die Durchführungsbedingung den Wert TRUE liefert.
- Tritt der Wert FALSE auf, wird die Schleife übersprungen und die der Schleife folgende Anweisung ausgeführt.

Beispiel

```

FUNCTION_BLOCK WHILE_BSP
VAR
    INDEX: INT ;
    KENNWORT: ARRAY [1..50] OF STRING ;
END_VAR
BEGIN
    INDEX := 1 ;
    WHILE INDEX <= 50 AND KENNWORT[INDEX] <> 'KEY' DO
        INDEX := INDEX + 2;
    END_WHILE ;
END_FUNCTION_BLOCK

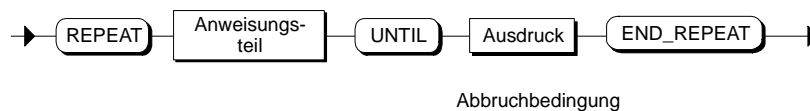
```

12.2.7 REPEAT-Anweisung

Eine REPEAT-Anweisung bewirkt die wiederholte Ausführung einer zwischen REPEAT und UNTIL stehenden Anweisungsfolge bis zum Eintreten einer Abbruchbedingung. Die Abbruchbedingung wird nach den Regeln eines logischen Ausdrucks gebildet.

Syntax

REPEAT-Anweisung



Die Bedingung wird jeweils nach der Ausführung des Rumpfes überprüft. Dies bedeutet, dass der Rumpf mindestens einmal ausgeführt wird, auch wenn die Abbruchbedingung von Anfang an erfüllt ist.

Beispiel

```
FUNCTION_BLOCK REPEAT_BSP
VAR
    INDEX: INT ;
    KENNWORT: ARRAY [1..50] OF STRING ;
END_VAR

BEGIN
    INDEX := 0 ;
    REPEAT
        INDEX := INDEX + 2 ;
    UNTIL INDEX > 50 OR KENNWORT[INDEX] = 'KEY'
    END_REPEAT ;

END_FUNCTION_BLOCK
```

12.2.8 CONTINUE-Anweisung

Eine CONTINUE-Anweisung dient zum Abbruch der Ausführung des momentanen Schleifendurchlaufes einer Wiederholungsanweisung (FOR, WHILE oder REPEAT).

Syntax

CONTINUE-Anweisung



Die CONTINUE-Anweisung wird nach folgenden Regeln bearbeitet:

- Diese Anweisung bewirkt den sofortigen Abbruch der Ausführung der Anweisungsfolge.
- Abhängig davon, ob die Bedingung für die Wiederholung der Schleife erfüllt ist oder nicht, wird der Rumpf noch einmal ausgeführt oder die Wiederholungsanweisung verlassen und die unmittelbar nachfolgende Anweisung ausgeführt.
- In einer FOR-Anweisung wird direkt nach einer CONTINUE-Anweisung die Laufvariable um die angegebene Schrittweite erhöht.

Beispiel

```

FUNCTION_BLOCK CONTINUE_BSP
VAR
    INDEX      :INT ;
    FELD :ARRAY[1..100] OF INT ;
END_VAR

BEGIN
    INDEX := 0 ;
    WHILE INDEX <= 100 DO
        INDEX := INDEX + 1 ;
        // Wenn FELD[INDEX] gleich INDEX ist,
        // dann wird FELD [INDEX] nicht verändert:
        IF FELD[INDEX] = INDEX THEN
            CONTINUE ;

            END_IF ;
            FELD[INDEX] := 0 ;
            // Weitere Anweisungen
        END_WHILE ;
    END_FUNCTION_BLOCK
  
```

12.2.9 EXIT-Anweisung

Eine EXIT-Anweisung dient zum Verlassen einer Schleife (FOR, WHILE oder REPEAT) an beliebiger Stelle und unabhängig vom Erfülltsein der Abbruchbedingung.

Syntax

EXIT-Anweisung



Die EXIT-Anweisung wird nach folgenden Regeln bearbeitet:

- Diese Anweisung bewirkt das sofortige Verlassen derjenigen Wiederholungsanweisung, die die EXIT-Anweisung unmittelbar umgibt.
- Die Ausführung des Programms wird nach dem Ende der Wiederholungsschleife (z.B. nach END_FOR) fortgesetzt.

Beispiel

```
FUNCTION_BLOCK EXIT_BSP
VAR
    INDEX_1      : INT ;
    INDEX_2      : INT ;
    INDEX_GESUCHT : INT ;
    KENNWORT     : ARRAY[1..51] OF STRING ;
END_VAR

BEGIN
    INDEX_2 := 0 ;
    FOR INDEX_1 := 1 TO 51 BY 2 DO
        // Verlassen der FOR-Schleife, wenn
        // KENNWORT[INDEX_1] gleich 'KEY' ist:
        IF KENNWORT[INDEX_1] = 'KEY' THEN
            INDEX_2 := INDEX_1 ;
            EXIT ;
        END_IF ;
    END_FOR ;
    // Die folgende Wertzuweisung kommt nach
    // der Ausführung von EXIT oder nach dem
    // regulären Ende der FOR-Schleife zur Ausführung:
    INDEX_GESUCHT := INDEX_2 ;
END_FUNCTION_BLOCK
```

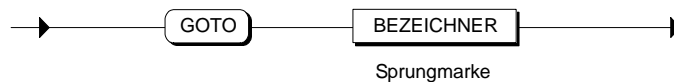
12.2.10 GOTO-Anweisung

Mit einer GOTO-Anweisung können Sie einen Programmsprung realisieren. Sie bewirkt den sofortigen Sprung zu einer angegebenen Sprungmarke und damit zu einer anderen Anweisung innerhalb desselben Bausteins.

GOTO-Anweisungen sollten nur in Sonderfällen, z. B. Fehlerbearbeitung eingesetzt werden. Nach den Regeln der strukturierten Programmierung sollte die GOTO-Anweisung nicht verwendet werden.

Syntax

GOTO-Anweisung



Dabei bezeichnet Sprungmarke eine Marke im LABEL/END_LABEL Vereinbarungsblock. Diese Marke ist der Anweisung vorangestellt, die nach dem GOTO als nächste ausgeführt werden soll.

Bei der Verwendung der GOTO-Anweisung sind folgende Regeln zu beachten:

- Das Ziel einer Sprunganweisung muss innerhalb desselben Bausteins liegen.
- Das Sprungziel muss eindeutig sein.
- Einsprung in einen Schleifenblock ist nicht zulässig. Aussprung aus einem Schleifenblock ist möglich.

Beispiel

```

FUNCTION_BLOCK GOTO_BSP
VAR
    INDEX      : INT ;
    A          : INT ;
    B          : INT ;
    C          : INT ;
    KENNWORT   : ARRAY[1..51] OF STRING ;
END_VAR
LABEL
    MARKE1, MARKE2, MARKE3 ;
END_LABEL

BEGIN
    IF A > B THEN
        GOTO MARKE1 ;
    ELSIF A > C THEN
        GOTO MARKE2 ;
    END_IF ;
    // . . .
    MARKE1: INDEX := 1 ;
                GOTO MARKE3 ;
    MARKE2: INDEX := 2 ;
    // . . .
    MARKE3:
    // . . .
  
```

12.2.11 RETURN-Anweisung

Eine RETURN-Anweisung bewirkt das Verlassen des aktuell bearbeiteten Bausteins (OB, FB, FC) und die Rückkehr zum aufrufenden Baustein bzw. zum Betriebssystem, wenn ein OB verlassen wird.

Syntax

RETURN-Anweisung



Hinweis

Eine RETURN-Anweisung am Ende des Ausführungsteils eines Codebausteins bzw. des Vereinbarungsteils eines Datenbausteins ist redundant, da diese automatisch ausgeführt wird.

12.3 Aufruf von Funktionen und Funktionsbausteinen

12.3.1 Aufruf und Parameterübergabe

Aufruf von FC und FB

Um die Lesbarkeit und die leichte Pflege von Anwenderprogrammen zu gewährleisten, unterteilt man die gesamte Funktionalität des Programms in Teilaufgaben, die von Funktionsbausteinen (FB) und Funktionen (FC) übernommen werden. Sie können von einem S7-SCL-Baustein aus andere FC oder FB aufrufen. Aufrufbare Bausteine sind:

- in S7-SCL erstellte Funktionsbausteine und Funktionen
- in anderen STEP 7-Sprachen (KOP, FUP, AWL) erstellte Funktionsbausteine und Funktionen
- Systemfunktionen (SFC) und Systemfunktionsbausteine (SFB), die im Betriebssystem der CPU verfügbar sind.

Prinzip der Parameterübergabe

Beim Aufruf von Funktionen oder Funktionsbausteinen findet ein Datenaustausch zwischen dem aufrufenden und dem aufgerufenen Baustein statt. In der Schnittstelle des aufgerufenen Bausteins sind Parameter definiert, mit denen der Baustein arbeitet. Diese Parameter werden als Formalparameter bezeichnet. Sie sind lediglich "Platzhalter" für die Parameter, die dem Baustein beim Aufruf übergeben werden. Die beim Aufruf übergebenen Parameter werden als Aktualparameter bezeichnet.

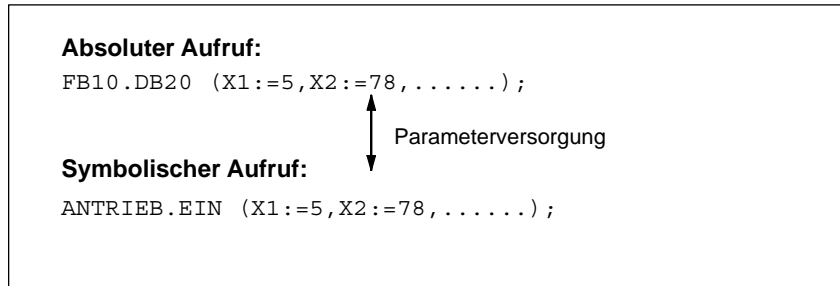
Syntax der Parameterübergabe

Die Parameter, die übergeben werden sollen, müssen im Aufruf als Parameterliste angegeben werden. Die Parameter werden in Klammern geschrieben. Mehrere Parameter werden durch Komma getrennt.

Im folgenden Beispiel eines Funktionsaufrufs werden je ein Eingangs-, Durchgangs-, und Ausgangsparameter angegeben.

Aktualparameter		Formalparameter
3	→	E_Par
LAENGE	↔	D_Par
Quersumme	←	A_Par

Die Angabe der Parameter hat die Form einer Wertzuweisung. Durch diese Wertzuweisung weisen Sie den Parametern, die Sie im Vereinbarungsteil des aufgerufenen Bausteins definiert haben (Formalparametern), einen Wert (Aktualparameter) zu.



12.3.2 Aufruf von Funktionsbausteinen

12.3.2.1 Aufruf von Funktionsbausteinen (FB oder SFB)

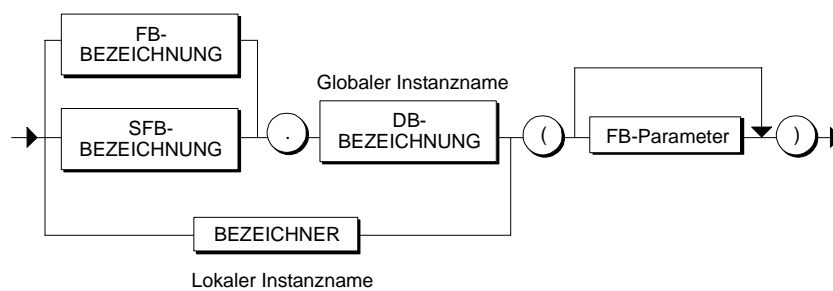
Beim Aufruf eines Funktionsbausteins können Sie sowohl globale Instanzdatenbausteine als auch lokale Instanzbereiche des aktuellen Instanzdatenbausteins benutzen.

Der Aufruf eines FB als lokale Instanz unterscheidet sich vom Aufruf als globale Instanz in der Speicherung der Daten. Die Daten werden hier nicht in einem gesonderten DB abgelegt, sondern in dem Instanzdatenbaustein des aufrufenden FB.

Syntax

FB-Aufruf

FB: Funktionsbaustein
SFB: Systemfunktionsbaustein

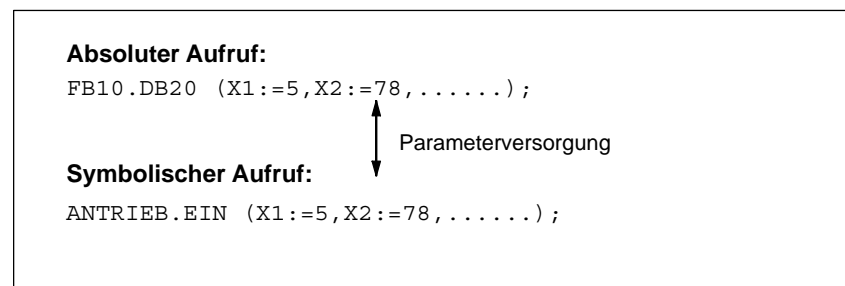


Aufruf als globale Instanz

Der Aufruf erfolgt in einer Aufrufanweisung unter Angabe:

- des Namens des Funktionsbausteins bzw. Systemfunktionsbausteins (FB- oder SFB-Bezeichnung),
- des Instanz-Datenbausteins (DB-Bezeichnung),
- sowie der Parameterversorgung (FB-Parameter).

Ein Aufruf einer globalen Instanz kann absolut oder symbolisch definiert sein.



Aufruf als lokale Instanz

Der Aufruf erfolgt in einer Aufrufanweisung unter Angabe:

- des lokalen Instanznamens (BEZEICHNER),
- sowie der Parameterversorgung (FB-Parameter),

Ein Aufruf einer lokalen Instanz ist immer symbolisch. Den symbolischen Namen müssen Sie im Vereinbarungsteil des aufrufenden Bausteins vereinbaren.

```
MOTOR (X1:=5,X2:=78,.....);
```



Parameterversorgung

12.3.2.2 Versorgung der FB-Parameter

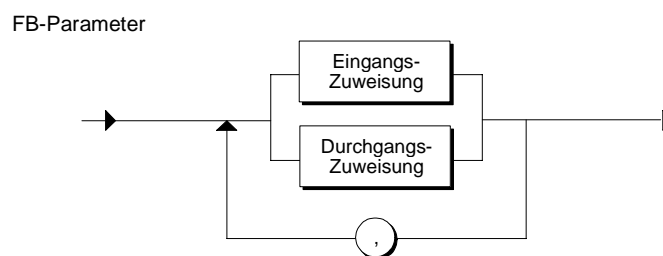
Beim Aufruf eines Funktionsbausteins - als globale oder lokale Instanz - müssen Sie in der Parameterliste folgende Parameter versorgen:

- Eingangsparameter
- Durchgangparameter

Die Ausgangsparameter werden nicht beim Aufruf eines FB angegeben, sondern nach dem Aufruf versorgt.

Syntax einer Wertzuweisung zur Definition der FB-Parameter:

Die Syntax der FB-Parameterangaben ist beim Aufruf globaler und lokaler Instanzen gleich.



Zur Parameterversorgung gelten folgende Regeln:

- Die Reihenfolge der Zuweisungen ist beliebig.
- Datentyp von Formal- und Aktualparameter müssen übereinstimmen.
- Die einzelnen Zuweisungen sind durch Komma getrennt.
- Ausgangszuweisungen sind in FB-Aufrufen nicht möglich. Der Wert eines vereinbarten Ausgangsparameters ist in den Instanzdaten gespeichert. Dort ist er für den Zugriff von allen FBs aus verfügbar. Um ihn zu lesen, müssen Sie von einem FB aus einen Zugriff definieren.
- Beachten Sie die Besonderheiten bei Parametern vom Datentyp ANY und vom Datentyp POINTER.

Ergebnis nach dem Bausteindurchlauf

Nach dem Bausteindurchlauf

- sind die übergebenen aktuellen Eingangsparameter unverändert.
- sind die übergebenen aber veränderten Werte der Durchgangsparameter aktualisiert. Eine Ausnahme bilden die Durchgangsparameter eines elementaren Datentyps.
- können die Ausgangsparameter vom aufrufenden Baustein aus dem globalen Instanz-Datenbaustein oder dem lokalen Instanzbereich gelesen werden.

Beispiel

Ein Aufruf mit je einer Zuweisung eines Eingangs- und eines Durchgangsparameters könnte z. B. wie folgt aussehen:

```
FB31.DB77 (E_Par:=3, D_Par:=LAENGE);
```

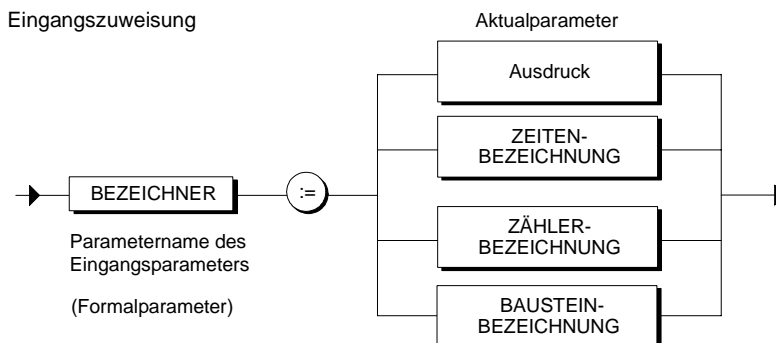
12.3.2.3 Eingangszuweisung (FB)

Durch Eingangszuweisungen werden den formalen Eingangsparametern Aktualparameter zugewiesen. Der FB kann diese Aktualparameter nicht verändern. Die Zuweisung von aktuellen Eingangsparametern ist optional. Wird kein Aktualparameter angegeben, bleiben die Werte des letzten Aufrufs erhalten.

Mögliche Aktualparameter sind:

Aktualparameter	Erläuterung
Ausdruck	<ul style="list-style-type: none"> • Arithmetischer, logischer oder Vergleichs-Ausdruck • Konstante • Erweiterte Variable
ZEITEN-/ZÄHLER-Bezeichnung	Sie legen eine bestimmte Zeit oder einen bestimmten Zähler fest, der bei der Bearbeitung eines Bausteins verwendet werden soll.
BAUSTEIN-Bezeichnung	<p>Sie legen einen bestimmten Baustein fest, der als Eingangsparameter verwendet werden soll. Die Bausteinart (FB, FC, DB) wird in der Deklaration des Eingangsparameters festgelegt.</p> <p>Bei der Parameterversorgung geben Sie die Bausteinnummer des Bausteins an. Dieses kann sowohl die absolute als auch die symbolische sein.</p>

Syntax



12.3.2.4 Durchgangszuweisung (FB)

Durchgangszuweisungen dienen dazu, den formalen Durchgangsparemtern des aufgerufenen FB Aktualparameter zuzuweisen. Der aufgerufene FB kann die Durchgangsparemter verändern. Der neue Wert des Parameters, der bei der Abarbeitung des FB entsteht, wird in den Aktualparameter zurückgeschrieben. Der ursprüngliche Wert wird dabei überschrieben.

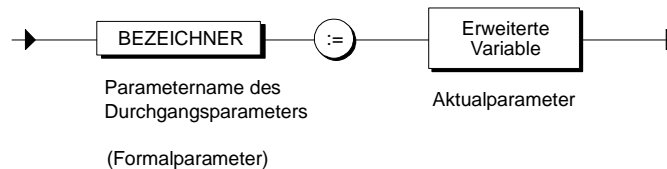
Wenn im aufgerufenen FB Durchgangsparemter vereinbart sind, müssen diese beim ersten Aufruf versorgt werden. Bei weiteren Durchläufen ist die Angabe von Aktualparametern optional. Bei Durchgangsparemtern eines elementaren Datentyps erfolgt jedoch keine Aktualisierung des Aktualparameters, wenn die Formalparameter beim Aufruf nicht versorgt werden.

Da der zugewiesene Aktualparameter während des FB-Durchlaufs als Durchgangsparemter verändert werden kann, muss er eine Variable sein:

Aktualparameter	Erläuterung
Erweiterte Variable	Folgende Typen der erweiterten Variablen stehen zur Verfügung: <ul style="list-style-type: none"> • einfache Variablen und Parameter • Zugriff auf Absolutvariablen • Zugriff auf Datenbausteine • Funktions-Aufrufe

Syntax

Durchgangszuweisung



Hinweis

- Bei den Datentypen ANY und POINTER gelten besondere Regeln bei der Versorgung.
- Bei Durchgangsparemtern nicht-elementaren Datentyps sind als Aktualparameter nicht erlaubt:
 - FB-Durchgangsparemter
 - FC-Parameter

12.3.2.5 Ausgangswerte lesen (FB-Aufruf)

Ausgangsparameter können nach der Bearbeitung des aufgerufenen Bausteins aus dem globalen Instanzbaustein oder dem lokalen Instanzbereich mittels einer Wertzuweisung gelesen werden.

Beispiel

```
ERGEBNIS := DB10.CONTROL ;
```

12.3.2.6 Beispiel zum Aufruf als globale Instanz

Ein Funktionsbaustein mit einer FOR-Schleife könnte wie in folgenden Beispielen aussehen. In den Beispielen wird angenommen, dass in der Symboltabelle für den FB17 das Symbol TEST vereinbart ist.

Funktionsbaustein

```
FUNCTION_BLOCK TEST

VAR_INPUT
    ENDWERT: INT; //Eingangsparameter
END_VAR
VAR_IN_OUT
    IQ1 : REAL; //Durchgangsparameter
END_VAR
VAR_OUTPUT
    CONTROL: BOOL; //Ausgangsparameter
END_VAR
VAR
    INDEX: INT;
END_VAR

BEGIN
    CONTROL :=FALSE;
    FOR INDEX := 1 TO ENDWERT DO
        IQ1 :=IQ1*2;
        IF IQ1 > 10000 THEN
            CONTROL := TRUE;
        END_IF;
    END_FOR;
END_FUNCTION_BLOCK
```

Aufruf

Zum Aufrufen dieses FB können Sie eine der folgenden Varianten wählen.
Vorausgesetzt wird, dass VARIABLE1 im aufrufenden Baustein als REAL-Variable vereinbart ist

```
//Absoluter Aufruf, globale Instanz:  
FB17.DB10 (ENDWERT:=10, IQ1:=VARIABLE1);  
  
//Symbolischer Aufruf, globale Instanz:  
TEST.TEST_1 (ENDWERT:=10, IQ1:= VARIABLE1);
```

Ergebnis

Nach der Ausführung des Bausteins steht der für den Durchgangparameter IQ1 ermittelte Wert in VARIABLE1 zur Verfügung.

Ausgangswert lesen

Die möglichen Varianten, den Ausgangsparameter CONTROL zu lesen, sollen anhand zweier Beispiele erläutert werden:

```
//Der Zugriff auf den Ausgangsparameter  
//erfolgt durch:  
    ERGEBNIS:= DB10.CONTROL;  
  
//Sie können den Ausgangsparameter aber auch  
//bei einem anderen FB-Aufruf direkt zur  
//Versorgung eines Eingangsparameters heranziehen:  
    FB17.DB12 (EIN_1:=DB10.CONTROL);
```


12.3.2.7 Beispiel zum Aufruf als lokale Instanz

Ein Funktionsbaustein mit einer einfachen FOR-Schleife könnte wie in Beispiel "Aufruf als globale Instanz" programmiert sein, wobei angenommen wird, dass in der Symboltabelle für den FB17 das Symbol TEST vereinbart ist.

Diesen FB können Sie unter der Voraussetzung dass VARIABLE1 im aufrufenden Baustein als REAL-Variable deklariert ist, so aufrufen:

Aufruf

```
FUNCTION_BLOCK CALL
VAR
// Lokale Instanzvereinbarung
    TEST_L : TEST ;
    VARIABLE1 : REAL ;
    ERGEBNIS : BOOL ;
END_VAR
BEGIN
. . .

// Aufruf, lokale Instanz:
TEST_L (ENDWERT:= 10, IQ1:= VARIABLE1) ;
```

Ausgangswert lesen

Der Ausgangsparameter CONTROL kann wie folgt gelesen werden:
//Der Zugriff auf den Ausgangsparameter
//erfolgt durch:
ERGEBNIS := TEST_L.CONTROL ;
END_FUNCTION_BLOCK

12.3.3 Aufruf von Funktionen

12.3.3.1 Aufruf von Funktionen (FC)

Der Aufruf einer Funktion erfolgt unter Angabe des Funktionsnamens (FC-, SFC-BEZEICHNUNG oder BEZEICHNER), sowie der Parameterliste. Der Funktionsname, der den Rückgabewert bezeichnet, kann absolut oder symbolisch angegeben werden:

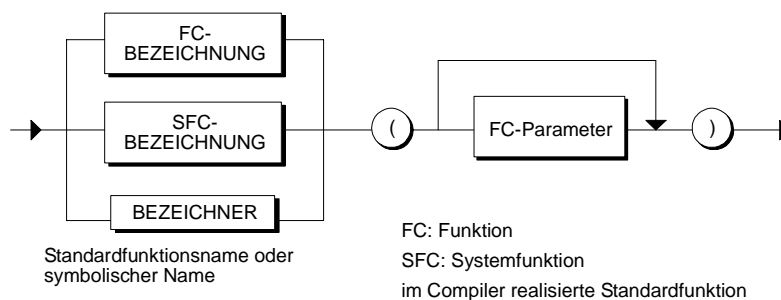
```
FC31 (X1:=5, Q1:=Quersumme) ; // Absolut
```

```
ABSTAND (X1:=5, Q1:=Quersumme) ; // Symbolisch
```

Nach dem Aufruf stehen die Ergebnisse der Funktion als Rückgabewert oder als Ausgangs- und Durchgangsparameter (Aktualparameter) zur Verfügung.

Syntax

Funktionsaufruf



Hinweis

Wird in S7-SCL eine Funktion aufgerufen, deren Rückgabewert nicht versorgt wurde, kann es zu einer fehlerhaften Ausführung des Anwenderprogramms kommen:

- Bei einer in S7-SCL programmierten Funktion kann dieser Fall eintreten, wenn der Rückgabewert zwar versorgt wurde, die entsprechende Anweisung aber nicht durchlaufen wird.
- Bei einer in AWL/KOP/FUP programmierten Funktion kann dieser Fall eintreten, wenn die Funktion ohne Versorgung des Rückgabewertes programmiert wurde oder die entsprechende Anweisung nicht durchlaufen wird.

12.3.3.2 Rückgabewert (FC)

Im Gegensatz zu den Funktionsbausteinen liefern Funktionen als Ergebnis den Rückgabewert. Aus diesem Grund können Funktionen wie Operanden behandelt werden (Ausnahme sind Funktionen vom Typ VOID).

Die Funktion berechnet den Rückgabewert, der den gleichen Namen trägt wie die Funktion, und gibt ihn an den aufrufenden Baustein zurück. Dort ersetzt der Wert den Funktionsaufruf.

In der folgenden Wertzuweisung wird z.B. die Funktion `ABSTAND` aufgerufen und das Ergebnis der Variablen `LAENGE` zugewiesen:

```
LAENGE := ABSTAND (X1 := -3, Y1 := 2);
```

Der Rückgabewert kann in folgenden Elementen eines FC oder FB verwendet werden:

- in einer Wertzuweisung,
- in einem logischen, arithmetischen oder Vergleichsausdruck oder
- als Parameter für einen weiteren Funktionsbaustein- oder Funktions-Aufruf

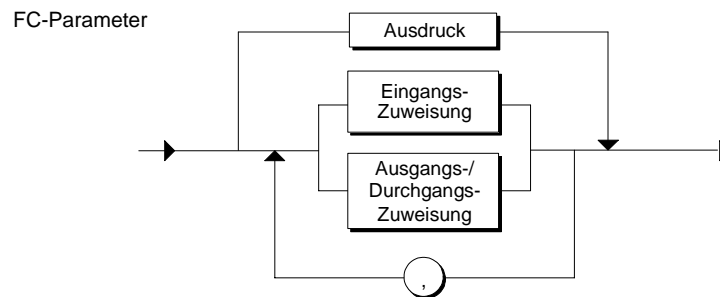
Hinweis

- Bei Funktionen mit dem Rückgabewert ANY muss mindestens ein Eingangs- oder Durchgangsparameter ebenfalls vom Typ ANY sein. Falls Sie mehrere ANY-Parameter definiert haben, müssen Sie diese mit Aktualparametern der gleichen Typklasse (z.B. INT, DINT und REAL) versorgen. Der Rückgabewert ist dann automatisch vom größten verwendeten Datentyp in dieser Typklasse.
 - Die maximale Länge des Datentyps STRING kann von 254 Zeichen auf eine beliebige Anzahl Zeichen reduziert werden.
-

12.3.3.3 FC-Parameter

Im Gegensatz zu Funktionsbausteinen haben Funktionen kein Gedächtnis, in das sie die Werte der Parameter speichern könnten. Lokale Daten werden während des Durchlaufs der Funktion nur temporär gespeichert. Aus diesem Grund müssen Sie beim Aufruf einer Funktion allen formalen Eingangs-, Durchgangs- und Ausgangsparametern, die im Vereinbarungsteil einer Funktion definiert werden, Aktualparameter zuweisen

Syntax



Regeln

Zur Parameterversorgung gelten folgende Regeln:

- Die Reihenfolge der Zuweisungen ist beliebig.
- Datentyp von Formal- und Aktualparameter müssen übereinstimmen.
- Die einzelnen Zuweisungen sind durch Komma getrennt.

Beispiel

Ein Aufruf mit je einer Zuweisung eines Ein-, Aus- und Durchgangsparameters könnte z. B. wie folgt aussehen:

```
FC32 (E_Param1:=5,D_Param1:=LAENGE,  
      A_Param1:=Quersumme)
```

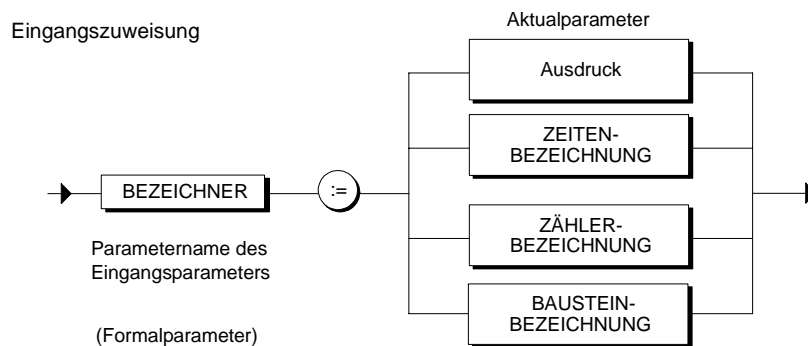
12.3.3.4 Eingangszuweisung (FC)

Durch Eingangszuweisungen werden den formalen Eingangsparametern der aufgerufenen FC Werte (Aktualparameter) zugewiesen. Der FC kann mit diesen Aktualparametern arbeiten, sie jedoch nicht verändern. Im Gegensatz zum FB-Aufruf ist diese Zuweisung beim FC-Aufruf nicht optional.

In Eingangszuweisungen sind folgende Aktualparameter zuweisbar:

Aktualparameter	Erläuterung
Ausdruck	Ein Ausdruck steht für einen Wert und besteht aus Operanden und Operationen. Folgende Arten von Ausdrücken stehen zur Verfügung: <ul style="list-style-type: none"> • Arithmetischer, logischer oder Vergleichs-Ausdruck • Konstante • Erweiterte Variable
ZEITEN-/ZÄHLER-Bezeichnung	Sie legen eine bestimmte Zeit oder einen bestimmten Zähler fest, der bei der Bearbeitung eines Bausteins verwendet werden soll.
BAUSTEIN-Bezeichnung	Sie legen einen bestimmten Baustein fest, der als Eingangsparameter verwendet werden soll. Die Bausteinart (FB, FC, DB) wird in der Deklaration des Eingangsparameters festgelegt. Bei der Parameterzuweisung geben Sie die Bausteinadresse des Bausteins an. Diese kann sowohl die absolute als auch die symbolische sein.

Syntax



Hinweis

Bei formalen Eingangsparametern von einem nicht-elementaren Datentyp sind FB-Durchgangparameter und FC-Parameter als Aktualparameter nicht erlaubt. Beachten Sie bitte auch die Besonderheiten bei den Datentypen ANY und POINTER.

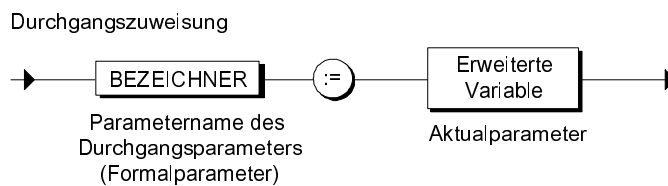
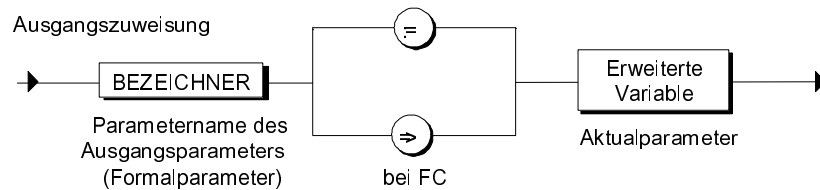
12.3.3.5 Ausgangs-/Durchgangszuweisung (FC)

In einer Ausgangszuweisung legen Sie fest, in welche Variable des aufrufenden Bausteins die Ausgangswerte, die bei der Abarbeitung einer Funktion entstehen, geschrieben werden. Mit einer Durchgangszuweisung weisen Sie einem Durchgangparameter einen Aktualwert zu.

Die Aktualparameter in Ausgangs- und Durchgangszuweisungen müssen eine Variable sein, da die Funktion Werte in die Parameter schreiben soll. Ein Eingangsparameter ist in Durchgangsanweisungen aus diesem Grund nicht zuweisbar (der Wert könnte nicht geschrieben werden). In Ausgangs- und Durchgangszuweisungen ist also nur die erweiterte Variable zuweisbar:

Aktualparameter	Erläuterung
Erweiterte Variable	Folgende Typen der erweiterten Variablen stehen zur Verfügung: <ul style="list-style-type: none"> • einfache Variablen und Parameter • Zugriff auf Absolutvariablen • Zugriff auf Datenbausteine • Funktions-Aufrufe

Syntax



Hinweis

Folgende Aktualparameter sind bei formalen Ausgangs-/Durchgangsparemtern nicht erlaubt:

- FC/FB-Eingangsparemter
 - FB-Durchgangsparemter von nicht elementarem Datentyp
 - FC-Durchgangs- und Ausgangsparemter von nicht elementarem Datentyp
 - Beachten Sie bitte auch die Besonderheiten bei den Datentypen ANY und POINTER.
 - Die maximale Länge des Datentyps STRING kann von 254 Zeichen auf eine beliebige Anzahl Zeichen reduziert werden.
-

12.3.3.6 Beispiel für einen Funktionsaufruf

Aufzurufende Funktion

Eine Funktion ABSTAND zur Berechnung des Abstandes zweier Punkte (X1,Y1) und (X2,Y2) in der Ebene bei Verwendung kartesischer Koordinaten könnte folgendermaßen aussehen (In den Beispielen ist immer angenommen, dass in einer Symboltabelle für den FC37 das Symbol ABSTAND vereinbart ist.):

```
FUNCTION ABSTAND: REAL // symbolisch
VAR_INPUT
    X1: REAL;
    X2: REAL;
    Y1: REAL;
    Y2: REAL;
END_VAR
VAR_OUTPUT
    Q2: REAL;
END_VAR
BEGIN
    ABSTAND:= SQRT( (X2-X1)**2 + (Y2-Y1)**2 );
    Q2:= X1+X2+Y1+Y2;
END_FUNCTION
```

Aufrufender Baustein

Zur weiteren Verwendung eines Funktionswertes stehen Ihnen unter anderem folgende Möglichkeiten offen:

```
FUNCTION_BLOCK CALL
VAR
    LAENGE : REAL ;
    QUERSUMME : REAL ;
    RADIUS : REAL;
    Y : REAL;
END_VAR
BEGIN
    . . .
    //Aufruf in einer Wertzuweisung:
    LAENGE := ABSTAND (X1:=3, Y1:=2, X2:=8.9, Y2:= 7.4,
        Q2:=QUERSUMME) ;
    //Aufruf in einem arithmetischen oder logischen Ausdruck,
    //z.B.
    Y := RADIUS + ABSTAND (X1:=-3, Y1:=2, X2:=8.9, Y2:=7.4,
        Q2:=QUERSUMME) ;
    //Verwendung in der Parameterversorgung eines weiteren
    //aufgerufenen Bausteins
    FB32.DB32 (DISTANZ:= ABSTAND (X1:=-3, Y1:=2, X2:=8.9,
        Y2:=7.4),
        Q2:=QUERSUMME) ;
    . . .
END_FUNCTION_BLOCK
```


12.3.4 Implizit definierte Parameter

12.3.4.1 Eingangsparameter EN

Jeder Funktionsbaustein und jede Funktion besitzt den implizit definierten Eingangsparameter EN. EN ist vom Datentyp BOOL und ist im Bereich bausteintemporäre Daten abgelegt. Wenn EN gleich TRUE ist, wird der aufgerufene Baustein ausgeführt, andernfalls nicht. Die Versorgung des Parameters EN ist optional. Es ist jedoch zu beachten, dass er nicht im Vereinbarungsteil eines Bausteines bzw. einer Funktion deklariert werden darf.

Da EN ein Eingangsparameter ist, können Sie EN innerhalb eines Bausteines nicht verändern.

Hinweis

Der Rückgabewert einer Funktion ist nicht definiert, falls sie nicht aufgerufen wurde (EN : FALSE).

Beispiel

```
FUNCTION_BLOCK FB57
VAR
    MEIN_ENABLE: BOOL ;
    Ergebnis : REAL;
END_VAR
// . . .
BEGIN
// . . .
MEIN_ENABLE:= FALSE ;

// Aufruf einer Funktion, wobei der EN - Parameter
// versorgt wird:
Ergebnis := FC85 (EN:= MEIN_ENABLE, PAR_1:= 27) ;
// FC85 wurde nicht ausgeführt, da MEIN_ENABLE
// oben gleich FALSE gesetzt
wurde....

END_FUNCTION_BLOCK
```

12.3.4.2 Ausgangsparameter ENO

Jeder Funktionsbaustein und jede Funktion besitzt den implizit definierten Ausgangsparameter ENO, der vom Datentyp BOOL ist. Er ist im Bereich bausteintemporäre Daten abgelegt. Am Ende der Ausführung eines Bausteines wird der aktuelle Wert des OK-Flags in ENO abgelegt.

Unmittelbar nach dem Aufruf eines Bausteines können Sie anhand des Wertes von ENO überprüfen, ob alle Operationen im Baustein richtig abgelaufen sind oder ob es zu Fehlern gekommen ist.

Beispiel

```
// Aufruf eines Funktionsbausteines:  
FB30.DB30 ([Parameterversorgung]);  
  
// Überprüfung, ob im aufgerufenen Baustein alles in Ordnung  
// abgelaufen ist:  
IF ENO THEN  
// alles in Ordnung  
// . . .  
ELSE  
// Fehler aufgetreten daher Fehlerbehandlung  
// . . .  
END_IF;
```

13 Zähler und Zeiten

13.1 Zähler

13.1.1 Zählerfunktionen

STEP 7 stellt eine Reihe von Standard-Zählerfunktionen zur Verfügung. Diese Zähler können Sie in Ihrem S7-SCL-Programm verwenden, ohne sie zuvor vereinbaren zu müssen. Sie müssen sie lediglich mit den erforderlichen Parametern versorgen. STEP 7 bietet folgende Zählerfunktionen an:

Zählerfunktion	Bedeutung
S_CU	Aufwärtszähler (Counter Up)
S_CD	Abwärtszähler (Counter Down)
S_CUD	Auf- und Abwärtszähler (Counter Up Down)

13.1.2 Aufruf von Zählerfunktionen

Zählerfunktionen werden aufgerufen wie Funktionen. Die Funktionsbezeichnung kann überall anstelle eines Operanden in einen Ausdruck eingesetzt werden, solange der Typ des Funktionswerts mit dem des ersetzten Operanden kompatibel ist.

Der Funktionswert (Rückgabewert), der an die Aufrufstelle zurückgegeben wird, ist der aktuelle Zählwert (BCD-Format) im Datentyp WORD.

Absoluter oder dynamischer Aufruf

Beim Aufruf können Sie als Zählernummer einen absoluten Wert (z. B. C_NO:=Z10) eingeben. Ein solcher Wert kann allerdings zur Laufzeit nicht mehr verändert werden.

Anstelle der absoluten Zähler-Nummer können Sie auch eine Variable oder eine Konstante vom Datentyp INT angeben. Das hat den Vorteil, dass Sie den Zähleraufruf dynamisch gestalten können, indem Sie der Variablen bei jedem Aufruf eine andere Nummer zuweisen.

Eine weitere Möglichkeit des dynamischen Aufrufs ist die Angabe einer Variablen vom Datentyp COUNTER.

Beispiele

```
//Beispiel für einen absoluten Aufruf:
S_CD (C_NO:=Z12,
      CD:=E0.0,
      CU:=E0.1,
      S:=E0.2 & E0.3,
      PV:=120,
      R:=FALSE,
      CV:=binVal,
      Q:=actFlag);

//Beispiel für einen dynamischen Aufruf: Bei jedem Durchlauf
//einer FOR-Schleife wird ein anderer Zähler aufgerufen:
FUNCTION_BLOCK ZAEHL
VAR_INPUT
    Zaehl: ARRAY [1..4] of STRUCT
        C_NO: INT;
        PV  : WORD;
    END_STRUCT;
.
.
END_VAR
.
.
FOR I:= 1 TO 4 DO
    S_CD(C_NO:=Zaehl[I].C_NO, S:=true, PV:= Zaehl[I].PV);
END_FOR;

//Beispiel für einen dynamischen Aufruf bei Verwendung
//einer Variablen vom Datentyp COUNTER:
FUNCTION_BLOCK ZAEHLER
VAR_INPUT
    MeinZaehler:COUNTER;
END_VAR
.
.
CurrVal:=S_CD (C_NO:=MeinZaehler,.....);
```

Hinweis

Die Namen der Funktionen und Parameter sind in deutscher und englischer Mnemonik gleich. Lediglich die Zählerbezeichnung ist von der Mnemonik abhängig (deutsch: Z, englisch: C).

13.1.3 Parameterversorgung bei Zählerfunktionen

Folgende Tabelle zeigt eine Übersicht der Parameter für Zählerfunktionen:

Parameter	Datentyp	Beschreibung
C_NO	COUNTER INT	Zählerkennnummer (ZAEHLER-BEZEICHNUNG); Bereich ist von der CPU abhängig.
CD	BOOL	Eingang CD: Abwärtszählen
CU	BOOL	Eingang CU: Aufwärtszählen
S	BOOL	Eingang für Voreinstellung des Zählers
PV	WORD	Wert im Bereich zwischen 0 und 999 für das Setzen des Zählers (eingegeben als 16#<Wert>, wobei Wert im BCD-Format)
R	BOOL	Rücksetzeingang
Q	BOOL	Ausgang: Status des Zählers
CV	WORD	Ausgang: Zählerstand binär
RET_VAL	WORD	Ergebnis im BCD-Format

Regeln

Da die Parameterwerte (z. B. CD:=E0.0) global gespeichert sind, ist ihre Angabe in bestimmten Fällen optional. Bei der Parameterversorgung sind folgende allgemeine Regeln zu beachten:

- Der Parameter für die Zählerbezeichnung C_NO muss beim Aufruf versorgt werden. Anstelle der absoluten Zählernummer (z. B. Z12) können Sie auch eine Variable oder eine Konstante mit dem Datentyp INT oder einen Eingangsparameter vom Datentyp COUNTER beim Aufruf angeben.
- Es muss entweder der Parameter CU (Aufwärtszähler) oder der Parameter CD (Abwärtszähler) versorgt werden.
- Die Angabe der Parameter PV (Vorbewert) und S (Setzen) kann paarweise entfallen.
- Der Ergebniswert im BCD-Format ist immer der Funktionswert.

Beispiel

```
FUNCTION_BLOCK FB1
VAR
    CurrVal, binVal: word;
    actFlag: bool;
END_VAR

BEGIN
CurrVal :=S_CD (C_NO:= Z10, CD:=TRUE, S:=TRUE, PV:=100,
                R:=FALSE,CV:=binVal,Q:=actFlag);
CurrVal :=S_CU (C_NO:= Z11, CU:=M0.0, S:=M0.1, PV:=16#110,
                R:=M0.2,CV:=binVal,Q:=actFlag);
CurrVal :=S_CUD(C_NO:= Z12, CD:=E0.0, CU:=E0.1,
                S:=E0.2 &E0.3, PV:=120,R:=FALSE,
                CV:=binVal,Q:=actFlag);
CurrVal :=S_CD (C_NO:= Z10, CD:=FALSE, S:=FALSE, PV:=100,
                R:=TRUE, CV:=binVal,Q:=actFlag);
END_FUNCTION_BLOCK
```

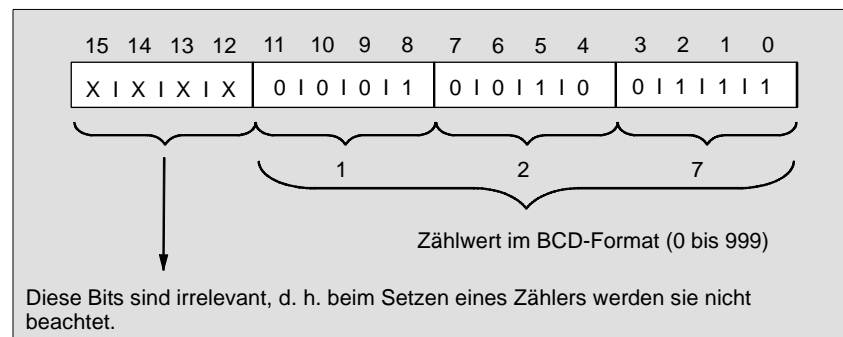
13.1.4 Eingabe und Auswertung des Zählerwerts

Für die Eingabe des Vorbesetzungswertes bzw. für die Auswertung des Funktionsergebnisses benötigen Sie die interne Darstellung des Zählerwerts. Der Zählerwert ist vom Datentyp WORD, wobei die Bits 0-11 den Zählerwert im BCD-Code enthalten. Die Bits 12-15 werden nicht berücksichtigt.

Beim Setzen des Zählers wird der von Ihnen festgelegte Wert in den Zähler geschrieben. Der Wertebereich liegt zwischen 0 und 999. Sie können den Zählerwert innerhalb dieses Bereichs verändern, indem Sie die Operationen Aufwärts-/Abwärtszähler (S_CUD), Aufwärtszähler (S_CU) und Abwärtszähler (S_CD) angeben.

Format

Folgendes Bild veranschaulicht die Bit-Konfiguration des Zählerwerts:



Eingabe

- Dezimal als Integer-Wert: z.B. 295, sofern dieser Wert einem gültigen BCD-Format entspricht.
- Im BCD-Format (Eingabe als Hexadezimalkonstante): z.B. 16#127

Auswertung

- Als Funktionsergebnis (Typ WORD): im BCD-Format
- Als Ausgangsparameter CV (Typ WORD): binär

13.1.5 Aufwärtszähler (S_CU)

Mit dem Zähler Counter Up (S_CU) können Sie nur Aufwärts-Zähloperationen durchführen. Die Tabelle veranschaulicht die Funktionsweise des Zählers:

Operation	Funktionsweise
Aufwärts zählen	Der Wert des Zählers wird um "1" erhöht, wenn der Signalzustand an Eingang CU von "0" auf "1" wechselt und der Zählwert kleiner als 999 ist.
Zähler setzen	Wenn der Signalzustand an Eingang S von "0" auf "1" wechselt, wird der Zähler mit dem Wert des Eingangs PV gesetzt. Ein solcher Signalwechsel ist immer erforderlich, um einen Zähler zu setzen.
Rücksetzen	Der Zähler wird rückgesetzt, wenn Eingang R = 1 ist. Das Rücksetzen des Zählers setzt den Zählwert auf "0".
Zähler abfragen	Eine Signalzustandsabfrage an Ausgang Q ergibt "1", wenn der Zählwert größer als "0" ist. Die Abfrage ergibt "0", wenn der Zählwert gleich "0" ist.

13.1.6 Abwärtszähler (S_CD)

Mit dem Zähler Counter Down (S_CD) können Sie nur Abwärts-Zähloperationen durchführen. Die Tabelle veranschaulicht die Funktionsweise des Zählers:

Operation	Funktionsweise
Abwärts zählen	Der Wert des Zählers wird um "1" vermindert, wenn der Signalzustand an Eingang CD von "0" auf "1" wechselt (steigende Flanke) und der Zählwert größer als "0" ist.
Zähler setzen	Wenn der Signalzustand an Eingang S von "0" auf "1" wechselt, wird der Zähler mit dem Wert des Eingangs PV gesetzt. Ein solcher Signalwechsel ist immer erforderlich, um einen Zähler zu setzen.
Rücksetzen	Der Zähler wird rückgesetzt, wenn Eingang R = 1 ist. Das Rücksetzen des Zählers setzt den Zählwert auf "0".
Zähler abfragen	Eine Signalzustandsabfrage an Ausgang Q ergibt "1", wenn der Zählwert größer als "0" ist. Die Abfrage ergibt "0", wenn der Zählwert gleich "0" ist.

13.1.7 Auf- und Abwärtszähler (S_CUD)

Mit dem Zähler Counter Up Down (S_CUD) können Sie sowohl Auf- als auch Abwärts-Zähloperationen ausführen. Bei Gleichzeitigkeit von Vorwärts- und Rückwärts-Zählimpulsen werden beide Operationen bearbeitet. Der Zählwert bleibt unverändert. Die Tabelle veranschaulicht die Funktionsweise des Zählers:

Operation	Funktionsweise
Aufwärts zählen	Der Wert des Zählers wird um "1" erhöht, wenn der Signalzustand an Eingang CU von "0" auf "1" wechselt und der Zählwert kleiner als 999 ist.
Abwärts zählen	Der Wert des Zählers wird um "1" vermindert, wenn der Signalzustand an Eingang CD von "0" auf "1" wechselt und der Zählwert größer als "0" ist.
Zähler setzen	Wenn der Signalzustand an Eingang S von "0" auf "1" wechselt, wird der Zähler mit dem Wert des Eingangs PV gesetzt. Ein solcher Signalwechsel ist immer erforderlich, um einen Zähler zu setzen.
Rücksetzen	Der Zähler wird rückgesetzt, wenn Eingang R = 1 ist. Das Rücksetzen des Zählers setzt den Zählwert auf "0".
Zähler abfragen	Eine Signalzustandsabfrage an Ausgang Q ergibt "1", wenn der Zählwert größer als "0" ist. Die Abfrage ergibt "0", wenn der Zählwert gleich "0" ist.

13.1.8 Beispiel zu Zählfunktionen

Parameterbelegung

Die Tabelle zeigt die Parameterbelegung der Beispielfunktion S_CD.

Parameter	Beschreibung
C_NO	MEINZAEHLER
CD	EINGANG E0.0
S	SETZEN
PV	VORBESETZUNG 16#0089
R	RUECKSETZEN
Q	A0.7
CV	BIN WERT

Beispiel

```
FUNCTION_BLOCK ZAEHLEN
VAR_INPUT
    MEINZAEHLER      : COUNTER ;
END_VAR
VAR_OUTPUT
    ERGEBNIS        : INT ;
END_VAR
VAR
    SETZEN           : BOOL ;
    RUECKSETZEN     : BOOL ;
    BCD_WERT        : WORD ;    // Zählerstand BCD codiert
    BIN_WERT        : WORD ;    // Zählerstand binär
    VORBESETZUNG    : WORD ;
END_VAR
BEGIN
    A0.0             := 1 ;
    SETZEN           := E0.2 ;
    RUECKSETZEN     := E0.3 ;
    VORBESETZUNG    := 16#0089 ;
//abwärts zählen
    BCD_WERT := S_CD (C_NO := MEINZAEHLER,
                    CD    := E0.0 ,
                    S     := SETZEN ,
                    PV    := VORBESETZUNG,
                    R     := RUECKSETZEN ,
                    CV    := BIN_WERT ,
                    Q     := A0.7) ;
//Weiterverarbeitung als Ausgangsparameter
    ERGEBNIS        := WORD_TO_INT (BIN_WERT) ;
    AW4             := BCD_WERT ;
END_FUNCTION_BLOCK
```

13.2 Zeiten

13.2.1 Zeitfunktionen

Zeiten sind Funktionselemente in Ihrem Programm, die zeitgesteuerte Abläufe ausführen und überwachen. STEP 7 stellt eine Reihe von Standard-Zeitfunktionen zur Verfügung, auf die Sie mit S7-SCL zugreifen können:

Zeitfunktion	Bedeutung
S_PULSE	Zeit als Impuls starten
S_PEXT	Zeit als verlängerter Impuls starten
S_ODT	Zeit als Einschaltverzögerung starten
S_ODTS	Zeit als speichernde Einschaltverzögerung starten
S_OFFDT	Zeit als Ausschaltverzögerung starten

13.2.2 Aufruf von Zeitfunktionen

Zeitfunktionen werden aufgerufen wie Funktionen. Die Funktionsbezeichnung kann überall anstelle eines Operanden in einen Ausdruck eingesetzt werden, solange der Typ des Funktionsergebnisses mit dem des ersetzten Operanden kompatibel ist.

Der Funktionswert (Rückgabewert) der an die Aufrufstelle zurückgegeben wird, ist ein Zeitwert vom Datentyp S5TIME.

Absoluter oder dynamischer Aufruf

Beim Aufruf können Sie als Nummer der Zeitfunktion einen absoluten Wert (z. B. T_NO:=T10) vom Datentyp TIMER eingeben. Ein solcher Wert kann allerdings zur Laufzeit nicht mehr verändert werden.

Anstelle der absoluten Nummer können Sie auch eine Variable oder eine Konstante vom Datentyp INT angeben. Das hat den Vorteil, dass Sie den Aufruf dynamisch gestalten können, indem Sie der Variablen bei jedem Aufruf eine andere Nummer zuweisen.

Eine weitere Möglichkeit des dynamischen Aufrufs ist die Angabe einer Variablen vom Datentyp TIMER.

Beispiele

```
//Beispiel für einen absoluten Aufruf:
CurrTime:=S_ODT (T_NO:=T10,
                S:=TRUE,
                TV:=T#1s,
                R:=FALSE,
                BI:=biVal,
                Q:=actFlag);

//Beispiel für einen dynamischen Aufruf: Bei jedem Durchlauf
//einer FOR-Schleife wird eine andere Zeitfunktion
//aufgerufen:
FUNCTION_BLOCK ZEIT
VAR_INPUT
    MEINE_ZEIT: ARRAY [1..4] of STRUCT
        T_NO: INT;
        TV  : WORD;
    END_STRUCT;
.
.
END_VAR
.
.
FOR I:= 1 TO 4 DO
CurrTime:= S_ODT(T_NO:=MEINE_ZEIT[I].T_NO, S:=true,
                TV:= MEINE_ZEIT[I].TV);
END_FOR;

//Beispiel für einen dynamischen Aufruf bei Verwendung einer
//Variablen vom Datentyp TIMER:
FUNCTION_BLOCK ZEITGEBER
VAR_INPUT
    meineZeit:TIMER;
END_VAR
.
.
CurrTime:=S_ODT (T_NO:=meineZeit,.....);
```

Hinweis

Die Namen der Funktionen sind in deutscher und englischer Mnemonik gleich.

13.2.3 Parameterversorgung bei Zeitfunktionen

Folgende Tabelle zeigt eine Übersicht der Parameter für Zeitfunktionen:

Parameter	Datentyp	Beschreibung
T_NO	TIMER INTEGER	Kennnummer der Zeit; Bereich ist von der CPU abhängig
S	BOOL	Starteingang
TV	S5TIME	Voreinstellung Zeitwert (BCD-Format)
R	BOOL	Rücksetzeingang
Q	BOOL	Status der Zeit
BI	WORD	Rest-Zeitwert (Binär)
RET_VAL	S5TIME	Timer-Wert

Regeln

Da die Parameterwerte global gespeichert sind, ist ihre Angabe in bestimmten Fällen optional. Die folgenden allgemeinen Regeln sind bei der Parameterversorgung zu beachten:

- Der Parameter für die Zeitgliedbezeichnung T_NO muss beim Aufruf versorgt werden. Anstelle der absoluten Zeitglied-Nummer (z.B. T10), können Sie auch eine Variable mit dem Datentyp INT oder einen Eingangsparemeter von Datentyp TIMER beim Aufruf angeben.
- Die Angabe der Parameter PV (Vorbeseztwert) und S (Setzen) kann paarweise entfallen.
- Der Ergebniswert im S5TIME-Format ist immer der Funktionswert.

Beispiel

```
FUNCTION_BLOCK FB2
VAR
    CurrTime    : S5time;
    BiVal       : word;
    ActFlag     : bool;
END_VAR

BEGIN
    CurrTime :=S_ODT (T_NO:= T10, S:=TRUE, TV:=T#1s, R:=FALSE,
                    BI:=biVal,Q:=actFlag);
    CurrTime :=S_ODTS (T_NO:= T11, S:=M0.0, TV:= T#1s, R:=M0.1,
                    BI:=biVal,Q:=actFlag);
    CurrTime :=S_OFFDT(T_NO:= T12, S:=E0.1 & actFlag, TV:= T#1s,
                    R:=FALSE,
                    BI:=biVal,Q:=actFlag);
    CurrTime :=S_PEXT (T_NO:= T13, S:=TRUE, TV:= T#1s, R:=E0.0,
                    BI:=biVal,Q:=actFlag);
    CurrTime :=S_PULSE(T_NO:= T14, S:=TRUE, TV:= T#1s, R:=FALSE,
                    BI:=biVal,Q:=actFlag);
END_FUNCTION_BLOCK
```

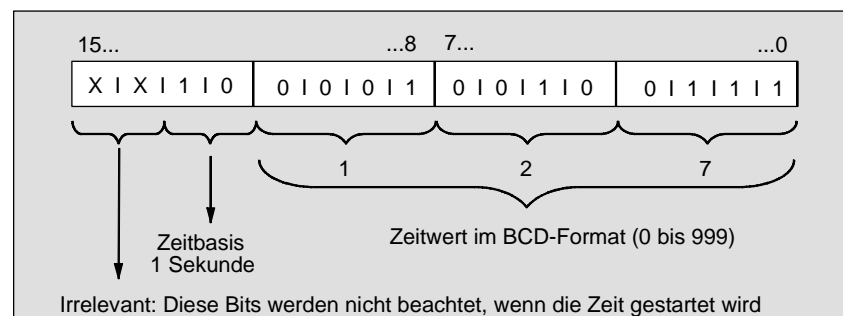
13.2.4 Eingabe und Auswertung des Zeitwerts

Für die Eingabe des Vorbesetzungswertes bzw. für die Auswertung des Funktionsergebnisses im BCD-Code benötigen Sie die interne Darstellung des Zeitwerts. Der Zeitwert ist vom Datentyp WORD, wobei die Bits 0-11 den Zeitwert im BCD-Format und die Bits 12 und 13 die Zeitbasis enthalten. Die Bits 14 und 15 werden nicht berücksichtigt.

Das Aktualisieren der Zeit vermindert den Zeitwert um jeweils eine Einheit in einem Intervall, der von der Zeitbasis festgelegt wurde. Der Zeitwert wird solange vermindert, bis er gleich "0" ist. Der Zeitbereich umfasst 0 bis 9990 Sekunden.

Format

Folgendes Bild veranschaulicht die Bit-Konfiguration des Zeitwerts:



Eingabe

Mit folgenden Formaten können Sie einen vordefinierten Zeitwert laden:

- In Stufendarstellung
- In Dezimaldarstellung

Die Zeitbasis wird in beiden Fällen automatisch gewählt und der Wert zur nächstniederen Zahl mit dieser Zeitbasis gerundet.

Auswertung

Das Ergebnis können Sie in zwei verschiedenen Formaten auswerten:

- Als Funktionsergebnis (Typ S5TIME): im BCD-Format
- Als Ausgangsparameter (Zeitwert ohne Zeitbasis vom Typ WORD): binär

Zeitbasis für Zeitwerte

Für die Eingabe und Auswertung des Zeitwertes benötigen Sie die Zeitbasis (Bits 12 und 13 des Timerworts). Die Zeitbasis definiert das Intervall, in dem der Zeitwert um eine Einheit vermindert wird (siehe Tabelle). Die kleinste Zeitbasis beträgt 10 ms; die größte 10 s.

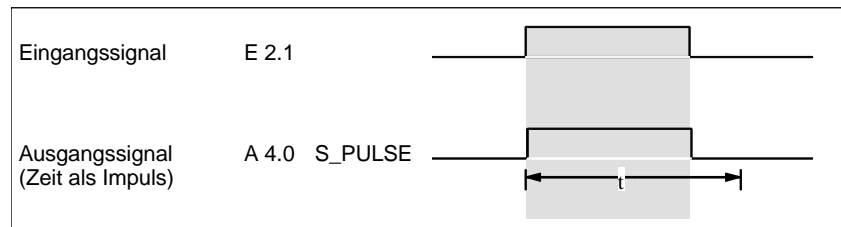
Zeitbasis	Binärcode für Zeitbasis
10 ms	00
100 ms	01
1 s	10
10 s	11

Hinweis

Da Zeitwerte nur in einem Zeitintervall gespeichert werden, werden Werte, die keine genauen Vielfachen des Zeitintervalls sind, abgeschnitten. Werte, deren Auflösung für den gewünschten Bereich zu groß ist, werden abgerundet, sodass der gewünschte Bereich erzielt wird, nicht jedoch die gewünschte Auflösung.

13.2.5 Zeit als Impuls starten (S_PULSE)

Die maximale Zeit, in der das Ausgangssignal auf "1" bleibt, ist gleich dem programmierten Zeitwert. Tritt während der Laufzeit des Zeitgliedes am Eingang der Signalzustand 0 auf, wird das Zeitglied auf "0" gesetzt. Dies bedeutet eine vorzeitige Beendigung der Laufzeit.



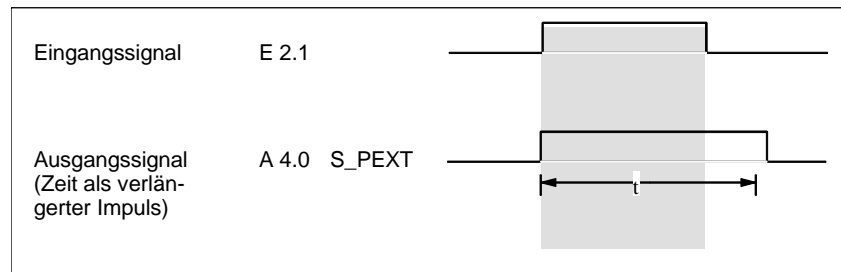
Funktionsweise

Die Tabelle zeigt die Funktionsweise von "Zeit als Impuls starten":

Operation	Funktionsweise
Zeit starten	Die Operation "Zeit als Impuls starten" startet eine angegebene Zeit, wenn der Signalzustand am Starteingang (S) von "0" auf "1" wechselt. Um die Zeit freizugeben, ist immer ein Signalwechsel erforderlich.
Laufzeit festlegen	Die Zeit läuft solange mit dem Wert weiter, der an Eingang TV angegeben ist, bis die programmierte Zeit abgelaufen ist und der Eingang S = 1 ist.
Laufzeit vorzeitig beenden	Wechselt Eingang S von "1" auf "0", bevor der Zeitwert abgelaufen ist, wird die Zeit angehalten.
Rücksetzen	Die Zeit wird zurückgesetzt, wenn der Rücksetzeingang (R) von "0" auf "1" wechselt, während die Zeit läuft. Durch diesen Wechsel werden auch der Zeitwert und die Zeitbasis auf Null zurückgesetzt. Der Signalzustand "1" an Eingang R hat keinen Einfluss, wenn die Zeit nicht läuft.
Signalzustand abfragen	Solange die Zeit läuft, ergibt eine Signalzustandsabfrage nach "1" an Ausgang Q das Ergebnis "1". Bei vorzeitiger Beendigung der Laufzeit ergibt eine Signalzustandsabfrage an Ausgang Q das Ergebnis "0".
Aktuellen Zeitwert abfragen	Der aktuelle Zeitwert kann am Ausgang BI und durch den Funktionswert S_PULSE abgefragt werden.

13.2.6 Zeit als verlängerter Impuls starten (S_PEXT)

Das Ausgangssignal bleibt für die programmierte Zeit (t) auf "1", unabhängig davon, wie lange das Eingangssignal auf "1" bleibt. Eine erneute Auslösung des Startimpulses bewirkt einen erneuten Ablauf der Zeitdauer, sodass der Ausgangsimpuls zeitlich verlängert wird (Nachtriggerung).



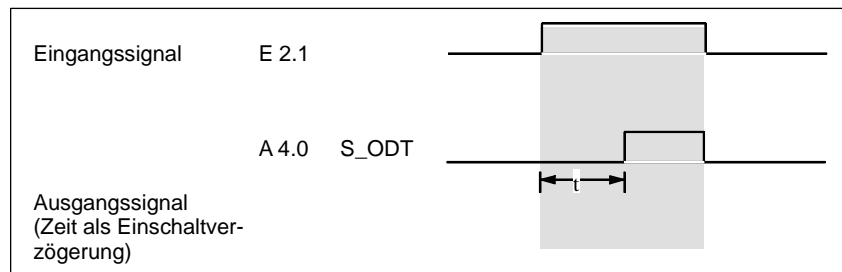
Funktionsweise

Die Tabelle zeigt die Funktionsweise von "Zeit als verlängerter Impuls starten":

Operation	Funktionsweise
Zeit starten	Die Operation "Zeit als verlängerten Impuls starten" (S_PEXT) startet eine angegebene Zeit, wenn der Signalzustand am Starteingang (S) von "0" auf "1" wechselt. Um die Zeit freizugeben, ist immer ein Signalwechsel erforderlich.
Laufzeit erneut starten	Wechselt der Signalzustand am Eingang S noch während der Laufzeit erneut auf "1", wird die Zeit mit dem angegebenen Zeitwert neu gestartet
Laufzeit voreinstellen	Die Zeit läuft solange mit dem Wert weiter, der an Eingang TV angegeben ist, bis die programmierte Zeit abgelaufen ist
Rücksetzen	Die Zeit wird zurückgesetzt, wenn der Rücksetzeingang (R) von "0" auf "1" wechselt, während die Zeit läuft. Durch diesen Wechsel werden auch der Zeitwert und die Zeitbasis auf Null zurückgesetzt. Der Signalzustand "1" an Eingang R hat keinen Einfluss, wenn die Zeit nicht läuft.
Signalzustand abfragen	Solange die Zeit läuft, ergibt eine Signalzustandsabfrage nach "1" an Ausgang Q das Ergebnis "1", unabhängig von der Länge des Eingangssignals.
Aktuellen Zeitwert abfragen	Der aktuelle Zeitwert kann am Ausgang BI und durch den Funktionswert S_PEXT abgefragt werden.

13.2.7 Zeit als Einschaltverzögerung starten (S_ODT)

Das Ausgangssignal wechselt nur von "0" auf "1", wenn die programmierte Zeit abgelaufen ist, und das Eingangssignal noch immer "1" beträgt. D.h. der Ausgang wird verzögert eingeschaltet. Eingangssignale, deren Zeitdauer kürzer als die der programmierten Zeit sind, erscheinen am Ausgang nicht.



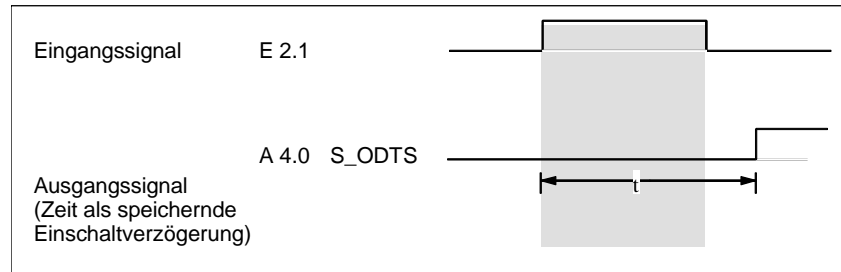
Funktionsweise

Die Tabelle zeigt die Funktionsweise von "Zeit als Einschaltverzögerung starten":

Operation	Funktionsweise
Zeit starten	Die Operation "Zeit als Eingangsverzögerung starten" startet eine angegebene Zeit, wenn der Signalzustand am Starteingang (S) von "0" auf "1" wechselt. Um die Zeit freizugeben, ist immer ein Signalwechsel erforderlich.
Zeit anhalten	Wechselt der Signalzustand an Eingang S von "1" auf "0", während die Zeit läuft, wird sie angehalten.
Laufzeit festlegen	Die Zeit läuft mit dem Wert weiter, der an Eingang TV angegeben ist, solange der Signalzustand an Eingang S = 1 ist.
Rücksetzen	Die Zeit wird zurückgesetzt, wenn der Rücksetzeingang (R) von "0" auf "1" wechselt, während die Zeit läuft. Durch diesen Wechsel werden auch der Zeitwert und die Zeitbasis auf Null zurückgesetzt. Die Zeit wird auch dann zurückgesetzt, wenn R = 1 ist, während die Zeit nicht läuft.
Signalzustand abfragen	Eine Signalzustandsabfrage nach "1" an Ausgang Q ergibt "1", wenn die Zeit fehlerfrei abgelaufen ist und Eingang S noch immer "1" ist. Wurde die Zeit angehalten, ergibt eine Signalzustandsabfrage nach "1" immer "0". Eine Signalzustandsabfrage nach "1" an Ausgang Q ergibt auch dann "0", wenn die Zeit nicht läuft und der Signalzustand an Eingang S noch immer "1" beträgt.
Aktuellen Zeitwert abfragen	Der aktuelle Zeitwert kann am Ausgang BI und durch den Funktionswert S_ODT abgefragt werden.

13.2.8 Zeit als speichernde Einschaltverzögerung starten (S_ODTS)

Das Ausgangssignal wechselt nur von "0" auf "1", wenn die programmierte Zeit abgelaufen ist, unabhängig davon, wie lange das Eingangssignal auf "1" bleibt.



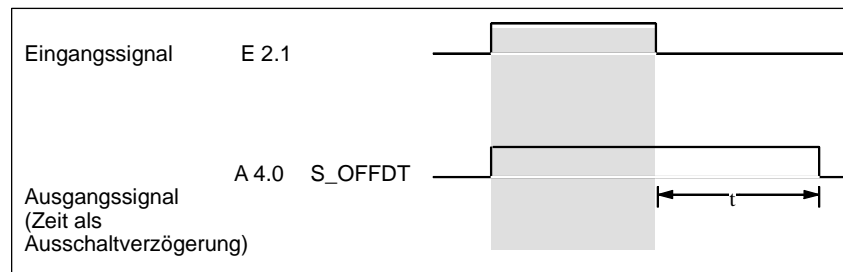
Funktionsweise

Die Tabelle zeigt die Funktionsweise von "Zeit als speichernde Einschaltverzögerung starten":

Operation	Funktionsweise
Zeit starten	Die Operation "Zeit als speichernde Eingangsverzögerung starten" startet eine angegebene Zeit, wenn der Signalzustand am Starteingang (S) von "0" auf "1" wechselt. Um die Zeit freizugeben, ist immer ein Signalwechsel erforderlich.
Zeit neu starten	Die Zeit wird mit dem angegebenen Wert neu gestartet, wenn Eingang S von "0" auf "1" wechselt, während die Zeit läuft.
Laufzeit festlegen	Die Zeit läuft auch dann mit dem Wert weiter, der an Eingang TV angegeben ist, wenn der Signalzustand an Eingang S noch vor Ablauf der Zeit auf "0" wechselt.
Rücksetzen	Wechselt der Rücksetzeingang (R) von "0" auf "1", wird die Zeit unabhängig vom Signalzustand an Eingang S zurückgesetzt.
Signalzustand abfragen	Eine Signalzustandsabfrage nach "1" an Ausgang Q ergibt nach Ablauf der Zeit unabhängig vom Signalzustand an Eingang S das Ergebnis "1".
Aktuellen Zeitwert abfragen	Der aktuelle Zeitwert kann am Ausgang BI und durch den Funktionswert S_ODTS abgefragt werden.

13.2.9 Zeit als Ausschaltverzögerung starten (S_OFFDT)

Bei einem Signalzustandswechsel von "0" nach "1" am Starteingang S erscheint am Ausgang Q der Zustand "1". Wechselt der Zustand am Starteingang von "1" nach "0", wird die Zeit gestartet. Erst nach Ablauf der Zeitdauer nimmt der Ausgang den Signalzustand "0" an. Der Ausgang wird also verzögert abgeschaltet.



Funktionsweise

Die Tabelle zeigt die Funktionsweise von "Zeit als Ausschaltverzögerung starten":

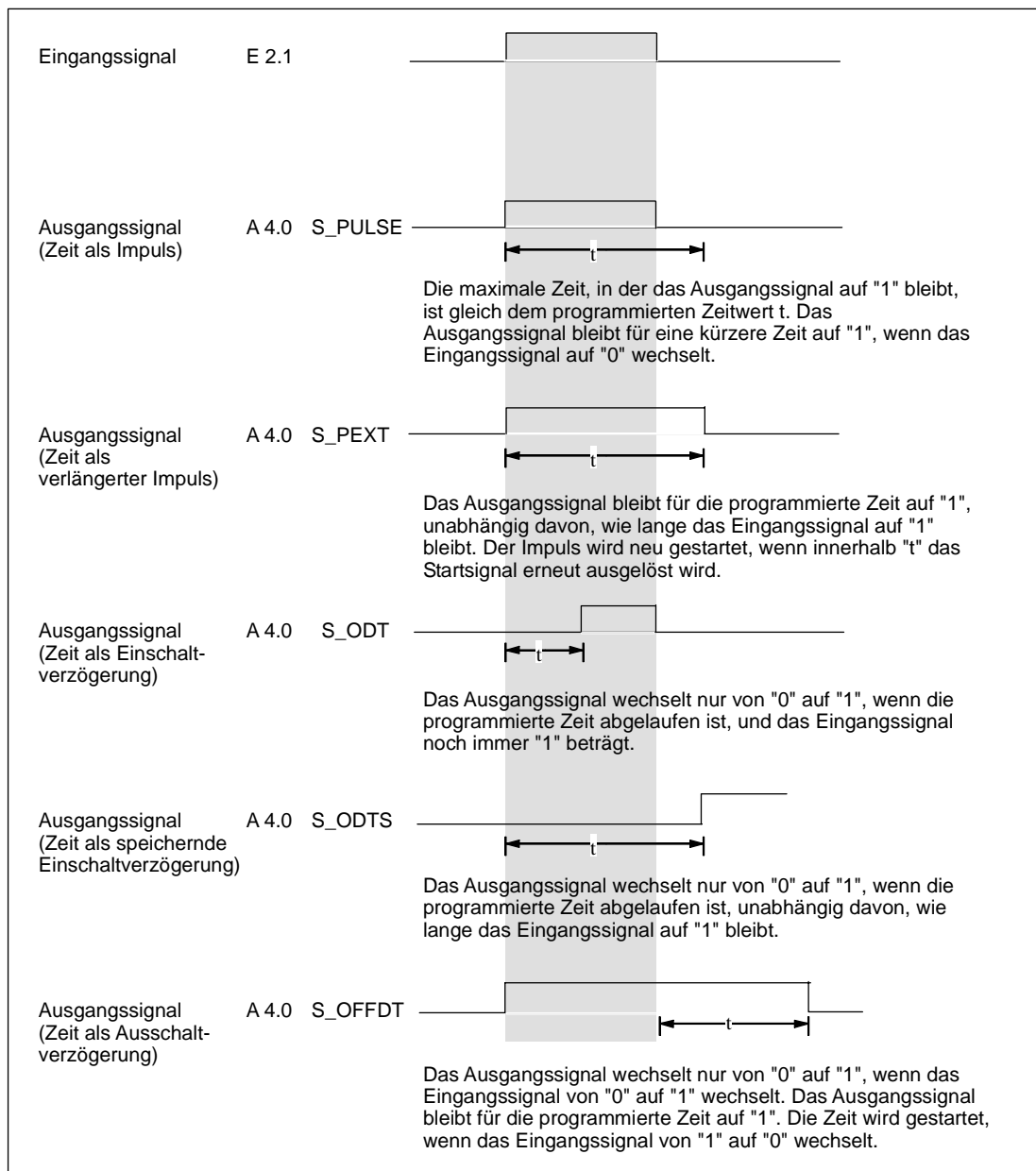
Operation	Funktionsweise
Zeit starten	Die Operation "Zeit als Ausschaltverzögerung starten" startet die angegebene Zeit, wenn der Signalzustand am Starteingang (S) von "1" auf "0" wechselt. Es ist immer ein Signalwechsel erforderlich, um die Zeit freizugeben.
Zeit neu starten	Die Zeit wird wieder neu gestartet, wenn der Signalzustand an Eingang S erneut von "1" auf "0" wechselt (z.B. nach dem Rücksetzen).
Laufzeit festlegen	Die Zeit läuft mit dem Wert, der an Eingang TV angegeben ist.
Rücksetzen	Wechselt der Rücksetzeingang (R) von "0" auf "1", während die Zeit läuft, wird die Zeit zurückgesetzt.
Signalzustand abfragen	Eine Signalzustandsabfrage nach "1" an Ausgang Q ergibt "1", wenn der Signalzustand an Eingang S = 1 ist oder die Zeit läuft.
Aktuellen Zeitwert abfragen	Der aktuelle Zeitwert kann am Ausgang BI und durch den Funktionswert S_OFFDT abgefragt werden.

13.2.10 Beispiel zu Zeitfunktionen

```
FUNCTION_BLOCK ZEITGEBER
VAR_INPUT
    meineZeit : TIMER ;
END_VAR
VAR_OUTPUT
    ergebnis : S5TIME ;
END_VAR
VAR
    setzen      : BOOL ;
    ruecksetzen : BOOL ;
    bcdWert     : S5TIME ; //Zeitbasis u. Restwert BCD
                    //codiert
    binWert     : WORD ; //Zeitwert binär
    Vorbesetzung : S5TIME ;
END_VAR
BEGIN
    A0.0      := 1;
    setzen    := E0.0 ;
    ruecksetzen := E0.1;
    Vorbesetzung := T#25S ;
    bcdWert   := S_PEXT (T_NO := meineZeit ,
                        S     := setzen ,
                        TV    := Vorbesetzung ,
                        R     := ruecksetzen ,
                        BI    := binWert ,
                        Q     := A0.7) ;
    //Weiterverarbeitung als Ausgangsparameter
    ergebnis := bcdWert ;
    //An Ausgabe zur Anzeige
    AW4 := binWert ;
END_FUNCTION_BLOCK
```

13.2.11 Auswahl des richtigen Zeitglieds

Folgendes Bild bietet einen Überblick über die fünf verschiedenen Zeiten, die in diesem Abschnitt beschrieben wurden. Diese Übersicht soll Ihnen helfen, die für Ihre Zwecke adäquaten Zeitgeber auszuwählen.



14 Standardfunktionen von S7-SCL

14.1 Datentyp-Konvertierungsfunktionen

14.1.1 Konvertierung von Datentypen

Wenn Sie zwei Operanden in einer Operation verknüpfen, müssen Sie die Verträglichkeit ihrer Datentypen beachten. Sind die Operanden ungleichen Datentyps, muss eine Konvertierung durchgeführt werden. S7-SCL kennt die folgenden Arten der Datentyp-Konvertierung:

- Implizite Datentyp-Konvertierung
Die Datentypen sind in Klassen eingeteilt. Innerhalb der Klassen führt S7-SCL eine implizite Datentyp-Konvertierung durch. Die dabei vom Compiler verwendeten Funktionen sind in den "Konvertierungsfunktionen Klasse A" zusammengefasst.
- Explizite Datentyp-Konvertierung
Bei Operanden, die nicht derselben Klasse angehören, müssen Sie selbst eine Konvertierungsfunktion aufrufen. Zur expliziten Datentyp-Konvertierung stellt S7-SCL zahlreiche Standardfunktionen zur Verfügung, die sich in folgende Klassen aufteilen lassen:
 - Konvertierungsfunktionen Klasse B
 - Funktionen zum Runden und Abschneiden

14.1.2 Implizite Datentypkonvertierung

Innerhalb der in der Tabelle definierten Klassen von Datentypen wird vom Compiler eine implizite Datentyp-Konvertierung entsprechend der angegebenen Reihenfolge durchgeführt. Als gemeinsames Format zweier Operanden ist jeweils der größere der beiden Datentypen definiert - so ist z.B. das gemeinsame Format von BYTE und WORD - WORD.

Beachten Sie bitte, dass bei einer Datentyp-Konvertierung innerhalb der Klasse ANY_BIT führende Bits auf 0 gesetzt werden.

Klassen	Reihenfolge der Konvertierung
ANY_BIT	BOOL > BYTE > WORD > DWORD
ANY_NUM	INT > DINT > REAL

Beispiel zur impliziten Konvertierung von Datentypen

```

VAR
    PID_REGLER_1 : BYTE ;
    PID_REGLER_2 : WORD ;
END_VAR
BEGIN
    IF (PID_REGLER_1 <> PID_REGLER_2) THEN ...
    (* In obiger IF-Anweisung wird PID_REGLER_1 implizit von BYTE
    zu WORD konvertiert. *)
    
```

14.1.2.1 Konvertierungsfunktionen Klasse A

Die Tabelle stellt die Datentyp-Konvertierungs-Funktionen der Klasse A dar. Diese Funktionen setzt der Compiler implizit ab, Sie können sie aber auch explizit angeben. Das Ergebnis ist immer definiert.

Funktionsname	Konvertierungsregel
BOOL_TO_BYTE	Ergänzung führender Nullen
BOOL_TO_DWORD	Ergänzung führender Nullen
BOOL_TO_WORD	Ergänzung führender Nullen
BYTE_TO_DWORD	Ergänzung führender Nullen
BYTE_TO_WORD	Ergänzung führender Nullen
CHAR_TO_STRING	Transformation in einen String (der Länge 1), der das gleiche Zeichen enthält.
DINT_TO_REAL	Transformation in REAL entsprechend IEEE-Norm. Der Wert kann sich - wegen der anderen Genauigkeit bei REAL - ändern.
INT_TO_DINT	Das höherwertige Wort des Funktionswertes wird bei einem negativen Eingangsparameter mit 16#FFFF, sonst mit Nullen aufgefüllt. Der Wert bleibt gleich.
INT_TO_REAL	Transformation in REAL entsprechend IEEE-Norm. Der Wert bleibt gleich.
WORD_TO_DWORD	Ergänzung führender Nullen

14.1.3 Standardfunktionen zur expliziten Datentyp-Konvertierung

Die allgemeine Beschreibung des Funktionsaufrufs finden Sie im Abschnitt "Aufruf von Funktionen".

Folgendes ist beim Aufruf von Konvertierungsfunktionen zu beachten:

- **Eingangsparameter:**
Jede Funktion zur Konvertierung eines Datentyps hat genau einen Eingangsparameter mit dem Namen IN. Da es sich um eine Funktion mit nur einem Parameter handelt, muss er nicht angegeben werden.
- **Funktionswert**
Das Ergebnis ist immer der Funktionswert.
- **Namensgebung**
Da die Datentypen des Eingangsparameters und des Funktionswertes aus dem jeweiligen Funktionsnamen hervorgehen, sind sie in den Übersichten (Klasse A und Klasse B) nicht gesondert aufgelistet: z.B. bei Funktion `BOOL_TO_BYTE` ist der Datentyp des Eingangsparameters `BOOL`, der Datentyp des Funktionswertes `BYTE`.

14.1.3.1 Konvertierungsfunktionen Klasse B

Die Tabelle stellt die Datentyp-Konvertierungsfunktionen der Klasse B dar. Diese Funktionen müssen Sie explizit angeben. Das Ergebnis kann auch undefiniert sein, wenn die Größe des Zieldatentyps unzureichend ist.

Sie können diesen Fall entweder selbst überprüfen, indem Sie eine Grenzprüfung vorschalten, oder die Überprüfung vom System durchführen lassen, indem Sie vor der Compilierung die Option "OK-Flag" wählen. Das System setzt dann in den Fällen, in denen das Ergebnis undefiniert ist, das OK-Flag auf FALSE.

Funktionsname	Konvertierungsregel	OK
<code>BOOL_TO_INT</code>	<code>WORD_TO_INT(BOOL_TO_WORD(x))</code>	N
<code>BOOL_TO_DINT</code>	<code>DWORD_TO_DINT(BOOL_TO_DWORD(x))</code>	N
<code>BYTE_TO_BOOL</code>	Kopieren des niederwertigsten Bits	J
<code>BYTE_TO_CHAR</code>	Übernahme des Bitstrings	N
<code>BYTE_TO_INT</code>	<code>WORD_TO_INT(BYTE_TO_WORD(x))</code>	N
<code>BYTE_TO_DINT</code>	<code>DWORD_TO_DINT(BYTE_TO_DWORD(x))</code>	N
<code>CHAR_TO_BYTE</code>	Übernahme des Bitstrings	N
<code>CHAR_TO_INT</code>	Der im Eingangsparameter vorhandene Bitstring wird in das niederwertige Byte des Funktionswertes eingetragen. Das höherwertige Byte wird mit Nullen aufgefüllt.	N
<code>DATE_TO_DINT</code>	Übernahme des Bitstrings	N
<code>DINT_TO_DATE</code>	Übernahme des Bitstrings	J
<code>DINT_TO_DWORD</code>	Übernahme des Bitstrings	N
<code>DINT_TO_INT</code>	Kopieren des Bits für das Vorzeichen. Der im Eingangsparameter vorhandene Wert wird im Datentyp INT interpretiert. Wenn der Wert kleiner als -32_768 oder größer als 32_767 ist, dann wird die OK-Variable auf FALSE gesetzt.	J

Funktionsname	Konvertierungsregel	OK
DINT_TO_TIME	Übernahme des Bitstrings	N
DINT_TO_TOD	Übernahme des Bitstrings	J
DINT_TO_BOOL	DWORD_TO_BOOL(DINT_TO_DWORD(x))	J
DINT_TO_BYTE	DWORD_TO_BYTE(DINT_TO_DWORD(x))	J
DINT_TO_STRING	DI_STRNG	N
DINT_TO_WORD	DWORD_TO_WORD(DINT_TO_DWORD(x))	J
DWORD_TO_BOOL	Kopieren des niederwertigsten Bits	J
DWORD_TO_BYTE	Kopieren der 8 niederwertigsten Bits	J
DWORD_TO_DINT	Übernahme des Bitstrings	N
DWORD_TO_REAL	Übernahme des Bitstrings	N
DWORD_TO_WORD	Kopieren der 16 niederwertigste Bits	J
DWORD_TO_INT	DINT_TO_INT (DWORD_TO_DINT(x))	J
INT_TO_CHAR	Übernahme des Bitstrings	J
INT_TO_WORD	Übernahme des Bitstrings	N
INT_TO_BOOL	WORD_TO_BOOL(INT_TO_WORD(x))	J
INT_TO_BYTE	WORD_TO_BYTE(INT_TO_WORD(x))	J
INT_TO_DWORD	WORD_TO_DWORD(INT_TO_WORD(x))	N
INT_TO_STRING	I_STRNG(x)	N
REAL_TO_DINT	Runden des IEEE-REAL-Wertes auf DINT. Wenn der Wert kleiner als -2_147_483_648 oder größer als 2_147_483_647 ist, dann wird die OK-Variable gleich FALSE gesetzt.	J
REAL_TO_DWORD	Übernahme des Bitstrings	N
REAL_TO_INT	Runden des IEEE-REAL-Wertes auf INT. Wenn der Wert kleiner als -32_768 oder größer als 32_767 ist, dann wird die OK-Variable gleich FALSE gesetzt.	J
REAL_TO_STRING	R_STRNG(x)	N
STRING_TO_CHAR	Kopieren des ersten Zeichens des Strings. Wenn der STRING nicht eine Länge von 1 hat, dann wird die OK-Variable gleich FALSE gesetzt.	J
STRING_TO_INT	STRNG_I(x)	N
STRING_TO_DINT	STRNG_DI(x)	N
STRING_TO_REAL	STRNG_R(x)	N
TIME_TO_DINT	Übernahme des Bitstrings	N
TOD_TO_DINT	Übernahme des Bitstrings	N
WORD_TO_BOOL	Kopieren des niederwertigsten Bits	J
WORD_TO_BYTE	Kopieren der niederwertigsten 8 Bits	J
WORD_TO_INT	Übernahme des Bitstrings	N
WORD_TO_DINT	INT_TO_DINT(WORD_TO_INT(x))	N
WORD_TO_BLOCK_DB	Das Bitmuster von WORD wird als Datenbausteinnummer interpretiert.	N
BLOCK_DB_TO_WORD	Die Datenbausteinnummer wird als Bitmuster von WORD interpretiert.	N

Funktionsname	Konvertierungsregel	OK
BCD_TO_INT(x) WORD_BCD_TO_INT(x)	Der Ausdruck x hat den Typ WORD und wird als BCD-codierter Wert zwischen -999 und +999 angenommen. Das Ergebnis liegt nach der Konvertierung als Ganzzahl (Binärdarstellung) vom Typ INT vor. Tritt während der Umwandlung einer Fehler auf, geht das Automatisierungssystem in STOP. Eine Bearbeitung der Stopursache ist im OB121 möglich.	N
INT_TO_BCD(x) INT_TO_BCD_WORD(x)	Der Ausdruck x hat den Typ INT und wird als Ganzzahl mit einem Wert zwischen -999 und +999 angenommen. Das Ergebnis liegt nach der Konvertierung als BCD-codierte Zahl vom Typ WORD vor. Außerhalb des Wertebereichs ist das Ergebnis undefiniert. Bei angewählter Option "OK-Flag setzen" erhält in diesem Fall das OK-Flag den Wert false.	J
BCD_TO_DINT(x) DWORD_BCD_TO_DINT(x)	Der Ausdruck x hat den Typ DWORD und wird als BCD-codierter Wert zwischen -9999999 und +9999999 angenommen. Das Ergebnis liegt nach der Konvertierung als Ganzzahl (Binärdarstellung) vom Typ DINT vor. Tritt während der Umwandlung einer Fehler geht das Automatisierungssystem in STOP. Eine Bearbeitung der Stopursache ist im OB121 möglich.	N
DINT_TO_BCD(x) DINT_TO_BCD_DWORD(x)	Der Ausdruck x hat den Typ DINT und wird als Ganzzahl mit einem Wert zwischen -9999999 und +9999999 angenommen. Das Ergebnis liegt nach der Konvertierung als BCD-codierte Zahl vom Typ DWORD vor. Außerhalb des Wertebereichs ist das Ergebnis undefiniert. Bei angewählter Option "OK-Flag setzen" erhält in diesem Fall das OK-Flag den Wert false	J

Achtung

Wird eine Konstante von einem höherwertigen Datentyp in einen niedrigeren Datentyp konvertiert, erhalten Sie beim Übersetzen eine Fehlermeldung, falls die Konstante außerhalb des Bereichs des niedrigeren Datentyps liegt.

Beispiele:

```

M0.0      :=WORD_TO_BOOL(W#16#FFFF) ;
MW0       :=DINT TO INT(35000) ;

```

Hinweis

Sie haben außerdem die Möglichkeit, weitere IEC-Funktionen zur Datentyp-Konvertierung zu nutzen. Informationen zu den einzelnen Funktionen finden Sie im STEP 7-Referenzhandbuch "System- und Standardfunktionen für S7-300/400".

14.1.3.2 Funktionen zum Runden und Abschneiden

Zu den Datentyp-Konvertierungen zählen auch die Funktionen zum Runden und Abschneiden von Zahlen. Die Tabelle zeigt die Namen, Datentypen (für den Eingangsparameter und den Funktionswert) und Aufgaben dieser Funktionen:

Funktionsname	Datentyp Eingangsparameter	Datentyp Funktionswert	Aufgabe
ROUND	REAL	DINT	Runden (Bilden einer DINT-Zahl). Gemäß DIN EN 61131-3 wird immer zum nächsten geraden Integer-Wert hin gerundet, d.h. 1.5 wird nach 2 gerundet, 2.5 wird ebenfalls nach 2 gerundet.
TRUNC	REAL	DINT	Abschneiden (Bilden einer DINT-Zahl)

Hinweis

Sie haben außerdem die Möglichkeit, weitere IEC-Funktionen zur Datentyp-Konvertierung zu nutzen. Informationen zu den einzelnen Funktionen finden Sie im STEP 7-Referenzhandbuch "System- und Standardfunktionen für S7-300/400".

Beispiel

```
// Hier wird abgerundet (Ergebnis: 3)
    ROUND (3.14) ;

// Hier wird aufgerundet (Ergebnis: 4)
    ROUND (3.56) ;

// Hier wird abgeschnitten (Ergebnis: 3)
    TRUNC (3.14) ;

// Hier wird abgeschnitten (Ergebnis: 3)
    TRUNC (3.56) ;
```

14.1.3.3 Beispiele zur Konvertierung mit Standardfunktionen

Im folgenden Beispiel ist eine explizite Konvertierung notwendig, da der Zieldatentyp weniger mächtig ist als der Quelldatentyp.

```
FUNCTION_BLOCK FB10
VAR
    SCHALTER : INT;
    REGLER   : DINT;
END_VAR

(* INT ist weniger mächtig als DINT *)
SCHALTER := DINT_TO_INT (REGLER) ;
// . . .
END_FUNCTION_BLOCK
```

Im folgenden Beispiel ist eine explizite Datentyp-Konvertierung notwendig, da der Datentyp REAL für einen arithmetischen Ausdruck mit der Operation MOD nicht zulässig ist:

```
FUNCTION_BLOCK FB20
VAR
    SCHALTER : REAL
    INTWERT   : INT := 17;
    KONV2     : INT ;
END_VAR

(* MOD darf nur auf Daten vom Typ INT oder DINT angewendet
werden *)
KONV2 := INTWERT MOD REAL_TO_INT (SCHALTER);
// . . .
END_FUNCTION_BLOCK
```

Im folgenden Beispiel ist eine Konvertierung notwendig, da nicht der richtige Datentyp für eine logische Operation vorliegt. Die Operation NOT darf nur auf Daten vom Typ BOOL, BYTE, WORD oder DWORD angewendet werden.

```
FUNCTION_BLOCK FB30
VAR
    INTWERT : INT := 17;
    KONV1   : WORD ;
END_VAR

(* NOT darf nicht auf Daten vom Typ INT angewendet werden *)
KONV1 := NOT INT_TO_WORD (INTWERT);
// . . .
END_FUNCTION_BLOCK
```

Das folgende Beispiel zeigt die Konvertierung bei Ein- und Ausgaben an die Peripherie:

```
FUNCTION_BLOCK FB40
VAR
    Radius_ein    : WORD ;
    Radius        : INT;
END_VAR

    Radius_ein    := %EB0;
    Radius        := WORD_TO_INT (radius_ein);
(* Konvertierung bei Wechsel in eine andere Typklasse. Wert
kommt von der Eingabe und wird zur Weiterberechnung
konvertiert.*)

    Radius        := Radius (flaeche:= Kreisdaten.flaeche)
    %AB0          :=WORD_TO_BYTE (INT_TO_WORD(RADIUS));
(*Radius wird rückgerechnet aus der Fläche und liegt in
Integer vor. Zur Ausgabe wird der Wert zuerst in eine andere
Typklasse (INT_TO_WORD) und dann in einen weniger mächtigen
Typ (WORD_TO_BYTE) konvertiert.*)
// . . .
END_FUNCTION_BLOCK
```


14.2 Numerische Standardfunktionen

14.2.1 Allgemeine arithmetische Standardfunktionen

Das sind die Funktionen zur Berechnung des absoluten Betrages, des Quadrats oder der Quadratwurzel einer Größe.

Der Datentyp ANY_NUM steht für INT, DINT oder REAL. Beachten Sie, dass Eingangsparameter vom Typ INT oder DINT intern in REAL-Variablen umgewandelt werden, wenn der Funktionswert vom Typ REAL ist.

Funktionsname	Datentyp Eingangsparameter	Datentyp Funktionswert	Beschreibung
ABS	ANY_NUM	ANY_NUM	Betrag
SQR	ANY_NUM	REAL	Quadrat
SQRT	ANY_NUM	REAL	Wurzel

Hinweis

Sie haben außerdem die Möglichkeit, weitere IEC-Funktionen zur Datentyp-Konvertierung zu nutzen. Informationen zu den einzelnen Funktionen finden Sie im STEP 7-Referenzhandbuch "System- und Standardfunktionen für S7-300/400".

14.2.2 Logarithmische Funktionen

Das sind die Funktionen zur Berechnung eines Exponentialwertes oder eines Logarithmus einer Größe.

Der Datentyp ANY_NUM steht für INT, DINT oder REAL. Beachten Sie, dass Eingangsparameter vom Typ ANY_NUM intern in REAL-Variablen umgewandelt werden.

Funktionsname	Datentyp Eingangsparameter	Datentyp Funktionswert	Beschreibung
EXP	ANY_NUM	REAL	e hoch IN
EXPD	ANY_NUM	REAL	10 hoch IN
LN	ANY_NUM	REAL	Natürlicher Logarithmus
LOG	ANY_NUM	REAL	Dekadischer Logarithmus

Hinweis

Sie haben außerdem die Möglichkeit, weitere IEC-Funktionen zur Datentyp-Konvertierung zu nutzen. Informationen zu den einzelnen Funktionen finden Sie im STEP 7-Referenzhandbuch "System- und Standardfunktionen für S7-300/400".

14.2.3 Trigonometrische Funktionen

Die in der Tabelle dargestellten trigonometrischen Funktionen erwarten und berechnen Größen von Winkeln im Bogenmaß.

Der Datentyp ANY_NUM steht für INT, DINT oder REAL. Beachten Sie, dass Eingangsparameter vom Typ ANY_NUM intern in REAL-Variablen umgewandelt werden.

Funktionsname	Datentyp Eingangsparameter	Datentyp Funktionswert	Beschreibung
ACOS	ANY_NUM	REAL	Arcus-Cosinus
ASIN	ANY_NUM	REAL	Arcus-Sinus
ATAN	ANY_NUM	REAL	Arcus-Tangens
COS	ANY_NUM	REAL	Cosinus
SIN	ANY_NUM	REAL	Sinus
TAN	ANY_NUM	REAL	Tangens

Hinweis

Sie haben außerdem die Möglichkeit, weitere IEC-Funktionen zur Datentyp-Konvertierung zu nutzen. Informationen zu den einzelnen Funktionen finden Sie im STEP 7-Referenzhandbuch "System- und Standardfunktionen für S7-300/400".

14.2.4 Beispiele zu numerischen Standardfunktionen

Aufruf	ERGEBNIS
ERGEBNIS := ABS (-5) ;	// 5
ERGEBNIS := SQRT (81.0) ;	// 9
ERGEBNIS := SQR (23) ;	// 529
ERGEBNIS := EXP (4.1) ;	// 60.340 ...
ERGEBNIS := EXPD (3) ;	// 1_000
ERGEBNIS := LN (2.718281) ;	// 1
ERGEBNIS := LOG (245) ;	// 2.389_166 ...
PI := 3.141592 ; ERGEBNIS := SIN (PI / 6) ;	// 0.5
ERGEBNIS := ACOS (0.5) ;	// 1.047_197 (=PI / 3)

14.3 Bitstring Standardfunktionen

Jede Bitstring Standardfunktion hat zwei Eingangsparameter, die durch IN bzw. N bezeichnet werden. Das Ergebnis ist immer der Funktionswert. Folgende Tabelle zeigt die Funktionsnamen und die Datentypen der zwei Eingangsparameter und des Funktionswertes. Es bedeuten:

- Eingangsparameter IN: Puffer, in dem die Bitschiebeoperationen durchgeführt werden. Der Datentyp dieses Eingangsparameters bestimmt den Datentyp des Funktionswertes.
- Eingangsparameter N: Anzahl der Rotationen bei den Umlaufpufferfunktionen ROL und ROR bzw. die Anzahl der zu schiebenden Stellen bei SHL und SHR.

Die Tabelle zeigt die möglichen Bitstring-Standardfunktionen:

Funktionsname	Datentyp Eingangsparameter IN	Datentyp Eingangsparameter N	Datentyp Funktionswert	Aufgabe
ROL	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	Der im Parameter IN vorhandene Wert wird um so viele Bitstellen nach links rotiert, wie der Inhalt des Parameters N angibt.
ROR	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	Der im Parameter IN vorhandene Wert wird um so viele Bitstellen nach rechts rotiert, wie der Inhalt des Parameters N angibt.
SHL	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	In dem im Parameter IN vorhandenen Wert werden so viele Bitstellen nach links geschoben und so viele Bitstellen auf der rechten Seite durch 0 ersetzt, wie der Inhalt des Parameters N angibt.
SHR	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	In dem im Parameter IN vorhandenen Wert werden so viele Bitstellen nach rechts geschoben und so viele Bitstellen auf der linken Seite durch 0 ersetzt, wie der Inhalt des Parameters N angibt.

Hinweis

Sie haben außerdem die Möglichkeit, weitere IEC-Funktionen zur Datentyp-Konvertierung zu nutzen. Informationen zu den einzelnen Funktionen finden Sie im STEP 7-Referenzhandbuch "System- und Standardfunktionen für S7-300/400".

14.3.1 Beispiele zu Bitstring Standardfunktionen

Aufruf	Ergebnis
ERGEBNIS := ROL (IN:=BYTE#2#1101_0011, N:=5);	// 2#0111_1010 // (= 122 dezimal)
ERGEBNIS := ROR (IN:=BYTE#2#1101_0011, N:=2);	// 2#1111_0100 // (= 244 dezimal)
ERGEBNIS := SHL (IN:=BYTE#2#1101_0011, N:=3);	// 2#1001_1000 // (= 152 dezimal)
ERGEBNIS := SHR (IN:=BYTE#2#1101_0011, N:=2);	// 2#0011_0100 // (= 52 dezimal)

14.4 Funktionen zur Verarbeitung von Zeichenketten

14.4.1 Funktionen zur Stringmanipulation

LEN

Die Funktion LEN (FC 21) gibt die aktuelle Länge einer Zeichenkette (Anzahl der gültigen Zeichen) als Rückgabewert aus. Ein Leerstring (") hat die Länge Null. Die Funktion meldet keine Fehler.

Beispiel: `LEN (S:= XYZ)`

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S	INPUT	STRING	D, L	Eingangsvariable im Format STRING
Rückgabewert		INT	E, A, M, D, L	Anzahl der aktuellen Zeichen

CONCAT

Die Funktion CONCAT fasst maximal 32 STRING-Variablen zu einer Zeichenkette zusammen. Ist die Ergebniszeichenkette länger als die am Ausgangsparameter angelegte Variable, wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt. Bei Verwendung der S7-SCL-Funktion CONCAT, wird implizit die FC2 aus der Bibliothek "IEC-Funktionen" aufgerufen.

Beispiel: `CONCAT (IN1:= Ventil, IN2:= Zustand)`

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	STRING CHAR	D, L	Eingangsvariable im Format STRING oder CHAR
IN2	INPUT	STRING CHAR	D, L	Eingangsvariable im Format STRING oder CHAR
INn	INPUT	STRING CHAR	D, L	Eingangsvariable im Format STRING oder CHAR
Rückgabewert		STRING CHAR	D, L	Zusammengefasste Zeichenkette

LEFT bzw. RIGHT

Die Funktionen LEFT bzw. RIGHT (FC 20 bzw. FC 32) liefern die ersten bzw. letzten L Zeichen einer Zeichenkette. Ist L größer als die aktuelle Länge der STRING-Variablen, wird der komplette String zurückgeliefert. Bei L = 0 oder wenn L negativ ist, wird ein Leerstring zurückgeliefert.

Beispiel: LEFT (IN:= 'Ventil', L:= 4)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	STRING	D, L	Eingangsvariable im Format STRING
L	INPUT	INT	E, A, M, D, L, Konst.	Länge der linken bzw. rechten Zeichenkette
Rückgabewert		STRING	D, L	Ausgangsvariable im Format STRING

MID

Die Funktion MID (FC 26) liefert einen Teil einer Zeichenkette. L ist die Länge der Zeichenkette, die ausgelesen werden soll, P ist die Position des ersten auszulesenden Zeichens.

Geht die Summe aus L und (P-1) über die aktuelle Länge der STRING-Variablen hinaus, wird eine Zeichenkette ab dem P. Zeichen bis zum Ende des Eingangswerts geliefert. In allen anderen Fällen (P liegt außerhalb der aktuellen Länge, P und/oder L gleich Null oder negativ) wird ein Leerstring ausgegeben.

Beispiel: MID (IN:= Temperatur, L:= 2, P:= 3)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	STRING	D, L	Eingangsvariable im Format STRING
L	INPUT	INT	E, A, M, D, L, Konst.	Länge der mittleren Zeichenkette
P	INPUT	INT	E, A, M, D, L, Konst.	Position des ersten Zeichens
Rückgabewert		STRING	D, L	Ausgangsvariable im Format STRING

INSERT

Die Funktion INSERT (FC 17) fügt die Zeichenkette am Parameter IN2 in die Zeichenkette am Parameter IN1 nach dem P. Zeichen ein. Ist P gleich Null, wird die zweite Zeichenkette vor der ersten Zeichenkette eingefügt. Ist P größer als die aktuelle Länge der ersten Zeichenkette, wird die zweite Zeichenkette an die erste angehängt. Ist P negativ, wird ein Leerstring ausgegeben. Wenn die Ergebniszeichenkette länger ist als die am Ausgangsparameter angegebene Variable; wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt.

Beispiel: INSERT (IN1:= 'Teilnehmer eingetroffen',
IN2:=Name, P:= 11)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	STRING	D, L	STRING-Variable, in die eingefügt wird
IN2	INPUT	STRING	D, L	einzufügende STRING-Variable
P	INPUT	INT	E, A, M, D, L, Konst.	Einfügeposition
Rückgabewert		STRING	D, L	Ergebniszeichenkette

DELETE

Die Funktion DELETE (FC 4) löscht in einer Zeichenkette L Zeichen ab dem P. Zeichen (einschließlich). Ist L und/oder P gleich Null oder ist P größer als die aktuelle Länge der Eingangszeichenkette, wird die Eingangszeichenkette zurückgeliefert. Ist die Summe aus L und P größer als die Eingangszeichenkette, wird bis zum Ende der Zeichenkette gelöscht. Ist L und/oder P negativ, wird ein Leerstring ausgegeben.

Beispiel: DELETE (IN:= 'Temperatur ok', L:= 6, P:= 5)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	STRING	D, L	STRING-Variable, in der gelöscht wird
L	INPUT	INT	E, A, M, D, L, KONST	Anzahl der zu löschenden Zeichen
P	INPUT	INT	E, A, M, D, L, KONST	Position des ersten zu löschenden Zeichens
Rückgabewert		STRING	D, L	Ergebniszeichenkette

REPLACE

Die Funktion REPLACE (FC 31) ersetzt L Zeichen der ersten Zeichenkette (IN1) ab dem P. Zeichen (einschließlich) durch die zweite Zeichenkette (IN2). Ist L gleich Null, wird die erste Zeichenkette zurückgeliefert. Ist P gleich Null oder Eins, wird ab dem 1. Zeichen (einschließlich) ersetzt. Liegt P außerhalb der ersten Zeichenkette, wird die zweite Zeichenkette an die erste Zeichenkette angehängt. Ist L und/oder P negativ, wird ein Leerstring ausgegeben. Wenn die Ergebniszeichenkette länger ist als die am Ausgangsparameter angegebene Variable; wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt.

Beispiel: REPLACE (IN1:= Temperatur, IN2:= Wert, L:= 6, P:= 5)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	STRING	D, L	STRING-Variable, in die eingesetzt wird
IN2	INPUT	STRING	D, L	einzusetzende STRING-Variable
L	INPUT	INT	E, A, M, D, L, Konst.	Anzahl der zu ersetzenden Zeichen
P	INPUT	INT	E, A, M, D, L, Konst.	Position des 1. ersetzten Zeichens
Rückgabewert		STRING	D, L	Ergebniszeichenkette

FIND

Die Funktion FIND (FC 11) liefert die Position der zweiten Zeichenkette (IN2) innerhalb der ersten Zeichenkette (IN1). Die Suche beginnt links; es wird das erste Auftreten der Zeichenkette gemeldet. Ist die zweite Zeichenkette in der ersten nicht vorhanden, wird Null zurückgemeldet. Die Funktion meldet keine Fehler.

Beispiel: FIND (IN1:= Bearbeitungsschritt, IN2:='station')

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	STRING	D, L	STRING-Variable, in der gesucht wird
IN2	INPUT	STRING	D, L	zu suchende STRING-Variable
Rückgabewert		INT	E, A, M, D, L	Position der gefundenen Zeichenkette

14.4.2 Funktionen zum Stringvergleich

Vergleiche mit Zeichenketten als Operanden sind mit den S7-SCL-Vergleichsoperationen =, <>, <, >, <= bzw. >= möglich. Der Compiler baut den entsprechenden Funktionsaufruf automatisch ein. Die folgenden Funktionen sind hier nur der Vollständigkeit halber aufgeführt.

EQ_STRNG und NE_STRNG

Die Funktion EQ_STRNG (FC 10) bzw. (FC 29) vergleicht die Inhalte zweier Variablen im Format STRING auf gleich (FC10) oder ungleich (FC29) und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter S1 gleich (ungleich) der Zeichenkette am Parameter S2 ist. Die Funktion meldet keine Fehler.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2	INPUT	STRING	D, L	Eingangsvariable im Format STRING
Rückgabewert		BOOL	E, A, M, D, L	Vergleichsergebnis

GE_STRNG und LE_STRNG

Die Funktion GE_STRNG (FC 13) bzw. (FC 19) vergleicht die Inhalte zweier Variablen im Format STRING auf größer (kleiner) oder gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter S1 größer (kleiner) oder gleich der Zeichenkette am Parameter S2 ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist 'a' größer als 'A'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die längere Zeichenkette als größer. Die Funktion meldet keine Fehler.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2	INPUT	STRING	D, L	Eingangsvariable im Format STRING
Rückgabewert		BOOL	E, A, M, D, L	Vergleichsergebnis

GT_STRNG und LT_STRNG

Die Funktion GT_STRNG (FC 15) bzw. (FC 24) vergleicht die Inhalte zweier Variablen im STRING-Format auf größer (kleiner) und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter S1 größer (kleiner) als die Zeichenkette am Parameter S2 ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist 'a' größer als 'A'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die längere Zeichenkette als größer. Die Funktion meldet keine Fehler.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2	INPUT	STRING	D, L	Eingangsvariable im Format STRING
Rückgabewert		BOOL	E, A, M, D, L	Vergleichsergebnis

14.4.3 Funktionen zur Wandlung des Datenformats

INT_TO_STRING und STRING_TO_INT

Die Funktionen INT_TO_STRING und STRING_TO_INT wandeln eine Variable im INT-Format in eine Zeichenkette oder eine Zeichenkette in eine INT-Variable. Implizit werden dabei die Funktionen I_STRNG (FC16) bzw. STRNG_I (FC38) aus der mitgelieferten Bibliothek "IEC-Funktionen" verwendet. Die Zeichenkette wird mit einem führenden Vorzeichen dargestellt. Ist die am Rückgabeparameter angegebene Variable zu kurz, findet keine Konvertierung statt.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
INT_TO_STRING				
I	INPUT	INT	E, A, M, D, L, Konst.	Eingangswert
Rückgabewert		STRING	D, L	Ergebniszeichenkette
STRING_TO_INT				
S	INPUT	STRING	D, L	Eingangszeichenkette
Rückgabewert		INT	E, A, M, D, L	Ergebnis

DINT_TO_STRING und STRING_TO_DINT

Die Funktionen DINT_TO_STRING und STRING_TO_DINT wandeln eine Variable im DINT-Format in eine Zeichenkette oder eine Zeichenkette in eine DINT-Variable. Implizit werden dabei die Funktionen DI_STRNG (FC5) bzw. STRNG_DI (FC37) aus der mitgelieferten Bibliothek "IEC-Funktionen" verwendet. Die Zeichenkette wird mit einem führenden Vorzeichen dargestellt. Ist die am Rückgabeparameter angegebene Variable zu kurz, findet keine Konvertierung statt.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DINT_TO_STRING				
I	INPUT	DINT	E, A, M, D, L, Konst.	Eingangswert
Rückgabe-wert		STRING	D, L	Ergebniszeichenkette
STRING_TO_DINT				
S	INPUT	STRING	D, L	Eingangszeichenkette
Rückgabe-wert		DINT	E, A, M, D, L	Ergebnis

REAL_TO_STRING und STRING_TO_REAL

Die Funktionen REAL_TO_STRING und STRING_TO_REAL wandeln eine Variable im REAL-Format in eine Zeichenkette oder eine Zeichenkette in eine REAL-Variable. Implizit werden dabei die Funktionen R_STRNG (FC30) bzw. STRNG_R (FC39) aus der mitgelieferten Bibliothek "IEC-Funktionen" verwendet.

Die Zeichenkette muss in folgendem Format vorliegen:

$\pm v.nnnnnnnE\pm xx$

(\pm = Vorzeichen, v = Vorkommastellen, n = Nachkommastellen, x = Exponentenstellen)

Ist die Länge der Zeichenkette kleiner als 14, oder ist sie nicht wie oben gezeigt aufgebaut, findet keine Wandlung statt

Ist die am Rückgabeparameter angegebene Variable zu kurz oder liegt am Parameter IN keine gültige Gleitpunktzahl an, findet ebenfalls keine Konvertierung statt.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REAL_TO_STRING				
IN	INPUT	REAL	E, A, M, D, L, Konst.	Eingangswert
RET_VAL	OUTPUT	STRING	D, L	Ergebniszeichenkette
STRING_TO_REAL				
S	INPUT	STRING	D, L	Eingangszeichenkette
Rückgabe-wert		REAL	E, A, M, D, L	Ergebnis

14.4.4 Beispiel zur Verarbeitung von Zeichenketten

Meldetexte zusammensetzen

```
//Meldetexte prozessgesteuert zusammensetzen und abspeichern

////////////////////////////////////
//Der Baustein enthält die benötigten Meldetexte und           //
//die letzten 20 generierten Meldungen                          //
////////////////////////////////////

DATA_BLOCK Meldetexte

    STRUCT
        Index      : int;
        textpuffer  : array [0..19] of string[34];
        HW          : array [1..5] of string[16]; //5
    verschiedene Geräte
        stati       : array [1..5] of string[12]; // 5
    verschiedene Zustände
    END_STRUCT
BEGIN
    Index :=0;
    HW[1] := 'Motor ';
    HW[2] := 'Ventil ';
    HW[3] := 'Presse ';
    HW[4] := 'Schweisstation ';
    HW[5] := 'Brenner ';
    Stati[1] := ' gestört';
    Stati[2] := ' gestartet';
    Stati[3] := ' Temperatur';
    Stati[4] := ' repariert';
    Stati[5] := ' gewartet';
END_DATA_BLOCK

////////////////////////////////////
//Die Funktion setzt Meldetexte zusammen und trägt sie in     //
//den DB Meldetexte ein. Die Meldetexte werden in einem       //
//Umlaufpuffer abgelegt. Der nächste freie Index des          //
//Textpuffers steht ebenfalls im DB Meldetexte und wird       //
//von der Funktion aktualisiert.                               //
////////////////////////////////////

FUNCTION Textgenerator : bool
VAR_INPUT
    unit      : int; //Index des Gerätetextes
    nr        : int; // IDNr. des Gerätes
    status    : int;
    wert      : int;
END_VAR
VAR_TEMP
    text      : string[34];
    i         : int;
END_VAR
```

```

//initialisierung der temporären Variablen
text := '';
Textgenerator := true;
Case unit of
  1..5 : case status of
    1..5 : text := concat( in1 := Meldetexte.HW[unit],
                          in2 := right
                              l:=2,in:=I_STRNG(nr));
    text := concat( in1 := text,
                   in2 := Meldetexte.stati[status]);
    if wert <> 0 then
      text := concat( in1 := text,
                     in2 := I_STRNG(wert));
    end_if;
  else Textgenerator := false;
end_case;
else Textgenerator := false;
end_case;
i := Meldetexte.index;
  Meldetexte.textpuffer[i] := text;
  Meldetexte.index := (i+1) mod 20;
END_FUNCTION

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Die Function wird im zyklischen Programm bei //
//Flankenwechsel im %M10.0 aufgerufen und bewirkt den //
//einmaligen Eintrag einer Meldung, falls sich ein //
//Parameter ändert. //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Organization_block zyklus
Var_temp
  Besy_ifx : array [0..20] of byte;
  fehler: BOOL;
End_var;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Der folgende Aufruf bewirkt den Eintrag //
//"Motor 12 gestartet" im Textpuffer des DB Meldetexte, //
//wobei über %MW0 eine 1, über %EW2 die 12 und über %MW2 //
//eine 2 übergeben werden muß. *) //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

if %M10.0 <> %M10.1 then
  fehler := Textgenerator (unit := word_to_int(%MW0),
                          nr := word_to_int(%EW2),
                          status := word_to_int(%MW2),
                          wert := 0);
  %M10.1:=M10.0;
end_if;
end_organization_block

```

14.5 Funktionen zur Selektion von Werten

14.5.1 Funktionen zur Selektion von Werten

Folgende Funktionen zur Selektion von Werten stehen als interne S7-SCL-Funktionen zur Verfügung. Sie sind konform zu IEC 61131-3.

Hinweis

Einige der Funktionen sind auch in der STEP 7-Standardbibliothek enthalten. Die Funktionen aus der Bibliothek entsprechen jedoch nicht in allen Punkten den Forderungen der IEC.

SEL

Die Funktion SEL selektiert einen von zwei Eingangswerten.

Als Eingangswerte sind alle Datentypen zulässig, mit Ausnahme der Datentypen ARRAY und STRUCT und der Parameter-Datentypen. Alle parametrisierten Variablen müssen vom Datentyp der selben Klasse sein.

Beispiel:

```
A := SEL (G := SELECT, IN0 := X, IN1 := Y);
```

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
G	INPUT	BOOL	E, A, M, D, L	Selektionskriterium
IN0	INPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	Erster Eingangswert
IN1	INPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	Zweiter Eingangswert
Rückgabewert	OUTPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	selektierter Eingangswert (optional)

MAX

Die Funktion MAX wählt aus einer Anzahl von Variablenwerten den größten aus.

Als Eingangswerte sind numerische Datentypen und Zeitdatentypen zugelassen. Alle parametrisierten Variablen müssen vom Datentyp der selben Klasse sein. Der Ausdruck nimmt den höchstwertigen Datentyp an.

Beispiel: A:= MAX (IN1:=a, IN2:=b, IN3:=c, IN4:=d);

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	erster Eingangswert
IN2	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	zweiter Eingangswert
INn (n=3...32)	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	letzter Eingangswert (optional)
Rückgabewert	OUTPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	größter der Eingangswerte (optional)

MIN

Die Funktion MIN wählt aus einer Anzahl von Variablenwerten den kleinsten aus.

Als Eingangswerte sind numerische Datentypen und Zeitdatentypen zugelassen. Alle parametrisierten Variablen müssen vom gleichen Datentyp sein. Der Ausdruck nimmt den höchstwertigen Datentyp an.

Beispiel: A:= MIN (IN1:=a, IN2:=b, IN3:=c, IN4:=d);

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	erster Eingangswert
IN2	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	zweiter Eingangswert
INn (n=3...32)	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	letzter Eingangswert (optional)
Rückgabewert	OUTPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	kleinster der Eingangswerte (optional)

LIMIT

Die Funktion LIMIT begrenzt den Zahlenwert einer Variablen auf parametrierbare Grenzwerte. Als Eingangswerte sind alle numerischen Datentypen und die Zeit-Datentypen zulässig. Alle Parameter müssen vom selben Datentyp sein. Der Ausdruck nimmt den höchstwertigen Datentyp an. Der untere Grenzwert (MN) darf nicht größer sein als der obere Grenzwert (MX).

Beispiel: A:= LIMIT (MN:=5, IN:= Bearbeitungsschritte, MX:= 10);

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MN	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	Untergrenze
IN	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	Eingangsvariable
MX	INPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	Obergrenze
Rückgabewert	OUTPUT	ANY_NUM Zeitdatentypen außer S5TIME	E, A, M, D, L	begrenzte Ausgangsvariable (optional)

MUX

Die Funktion MUX selektiert einen Eingangswert aus einer Anzahl von Eingangswerten. Die Auswahl erfolgt anhand des Eingangsparameters K. Als Eingangswerte sind alle Datentypen zulässig. Der Ausdruck nimmt den höchstwertigen Datentyp an.

Beispiel:

A:= MUX (K:=SELECT, IN0:= Steps, IN1:=Number, IN2:=Total);

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
K	INPUT	INT	E, A, M, D, L	Selektionskriterium
IN0	INPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	Erster Eingangswert
IN1	INPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	Zweiter Eingangswert
INn (n=2...31)	INPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	Letzter Eingangswert (optional)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
INELSE	INPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	Alternativer Eingangswert (optional) Liegt K außerhalb von 0...n, so wird der aktuelle Wert von INELSE verwendet. Ist INELSE nicht belegt, wird der aktuelle Wert von IN0 verwendet.
Rückgabewert	OUTPUT	Alle Datentypen außer ARRAY und STRUCT	E, A, M, D, L	Ausgewählter Wert (optional)

14.6 SFCs, SFBs und Standardbibliothek

14.6.1 Systemfunktionen/-funktionsbausteine und Standardbibliothek

Die S7-CPU's enthalten in das Betriebssystem integrierte System- und Standardfunktionen, die Sie bei der Programmierung in S7-SCL nutzen können. Im Einzelnen sind das:

- Organisationsbausteine (OBs)
- Systemfunktionen (SFC)
- Systemfunktionsbausteine (SFB)

Aufrufschnittstelle (SFC/SFB)

Sie können Bausteine symbolisch oder absolut adressieren. Dazu benötigen Sie entweder den symbolischen Namen, der in der Symboltabelle vereinbart sein muss, oder die Nummer für die absolute Bezeichnung des Bausteins.

Beim Aufruf müssen Sie den Formalparametern, deren Namen und Datentypen bei der Erstellung des parametrierbaren Bausteins festgelegt wurden, die Aktualparameter zuordnen, mit dessen Werten der Baustein zur Laufzeit Ihres Programms arbeitet.

S7-SCL durchsucht folgende Verzeichnisse und Bibliotheken nach dem aufzurufenden Baustein:

- den Ordner "Programme"
- die Simatic-Standardbibliotheken
- die IEC-Standardbibliothek

Findet S7-SCL einen Baustein, so wird dieser in das Anwenderprogramm kopiert. Ausnahmen bilden die Bausteine, die aufgrund ihres Namens mit der Schreibweise (" ... ") gerufen werden müssen und absolut aufgerufene Bausteine. Diese Namen werden nur in der Symboltabelle des Anwenderprogramms gesucht. Sie müssen diese Funktionen selbst mit dem SIMATIC Manager ins Anwenderprogramm kopieren.

Bedingter Aufruf(SFB/SFC)

Für den bedingten Aufruf müssen Sie den vordefinierten Eingangsparameter EN mit 0 belegen (z.B. über den Eingang E0.3), dann wird der Baustein nicht aufgerufen. Wird EN mit 1 belegt, wird die Funktion aufgerufen. Der Ausgangsparameter ENO wird in diesem Fall auch auf "1" gesetzt (sonst auf "0"), falls während der Bearbeitung des Bausteins kein Fehler auftritt.

Ein bedingter Aufruf von SFC ist nicht empfehlenswert, da die Variable, die den Rückgabewert der Funktion aufnehmen soll, undefiniert ist, wenn die Funktion nicht aufgerufen wird.

Hinweis

Wenn Sie folgende Operationen für die Datentypen TIME, DATE_AND_TIME und STRING in Ihrem Programm verwenden, ruft S7-SCL implizit die entsprechenden Standardbausteine auf.

Deshalb sind die Symbole und Bausteinnummern dieser Standardbausteine reserviert und dürfen nicht für andere Bausteine benutzt werden. Ein Verstoß gegen diesen Hinweis wird von S7-SCL nicht in allen Fällen überprüft und kann zu einem Compilerfehler führen.

Die folgende Tabelle gibt eine Übersicht der von S7-SCL implizit benutzten IEC-Standardfunktionen.

Operation	DATE_AND_TIME	STRING
==	EQ_DT (FC9)	EQ_STRING (FC10)
<>	NE_DT (FC28)	NE_STRING (FC29)
>	GT_DT (FC14)	GT_STRING (FC15)
>=	GE_DT (FC12)	GE_STRING (FC13)
<=	LE_DT (FC18)	LE_STRING (FC19)
<	LT_DT (FC23)	LT_STRING (FC24)
DATE_AND_TIME + TIME	AD_DT_TM (FC1)	
DATE_AND_TIME + TIME	SB_DT_TM (FC35)	
DATE_AND_TIME + DATE_AND_TIME	SB_DT_DT (FC34)	
TIME_TO_S5TIME(TIME)	TIM_S5TI (FC40)	
S5TIME_TO_TIME(S5TIME)	S5TI_TIM (FC33)	

Alle weiteren Informationen über verfügbare SFB, SFC und OBs sowie eine detaillierte Schnittstellenbeschreibung finden Sie im STEP 7-Referenzhandbuch "System- und Standardfunktionen für S7-300/400".

14.6.2 Übergabeschnittstelle zu OB

Organisationsbausteine

Organisationsbausteine bilden die Schnittstelle zwischen dem Betriebssystem der CPU und dem Anwenderprogramm. Mit Hilfe von OB können Programmteile gezielt zur Ausführung gebracht werden:

- beim Anlauf der CPU
- in zyklischer oder auch zeitlich getakteter Ausführung
- zu bestimmten Uhrzeiten oder Tagen
- nach Ablauf einer vorgegebenen Zeitdauer
- beim Auftreten von Fehlern
- beim Auftreten von Prozess- oder Kommunikationsalarmen

Organisationsbausteine werden entsprechend der ihnen zugeordneten Priorität bearbeitet.

Verfügbare OB

Nicht alle CPUs können alle in S7 verfügbaren OB bearbeiten. Sie entnehmen den Datenblättern zu Ihrer CPU, welche OB Ihnen zur Verfügung stehen.

15 Sprachbeschreibung

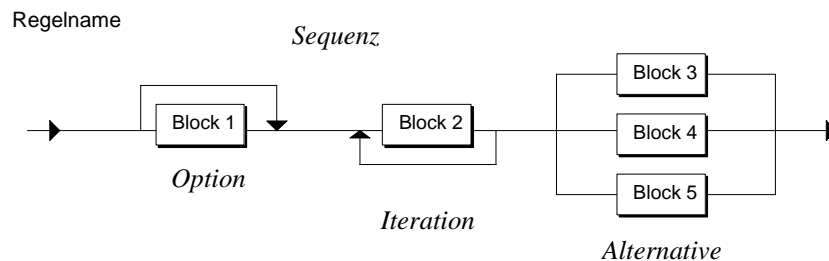
15.1 Formale Sprachbeschreibung

15.1.1 Übersicht Syntaxdiagramme

Basis für die Sprachbeschreibung in den einzelnen Kapiteln sind Syntaxdiagramme. Sie geben Ihnen einen guten Einblick in den syntaktischen Aufbau von S7-SCL. Eine vollständige Zusammenstellung aller Diagramme mit den Sprachelementen finden Sie in den Kapiteln "Lexikalische Regeln" und "Syntaktische Regeln".

Was ist ein Syntaxdiagramm?

Das Syntaxdiagramm ist eine grafische Darstellung der Struktur der Sprache. Die Struktur wird durch eine Folge von Regeln beschrieben. Dabei kann eine Regel auf bereits eingeführten Regeln aufbauen.

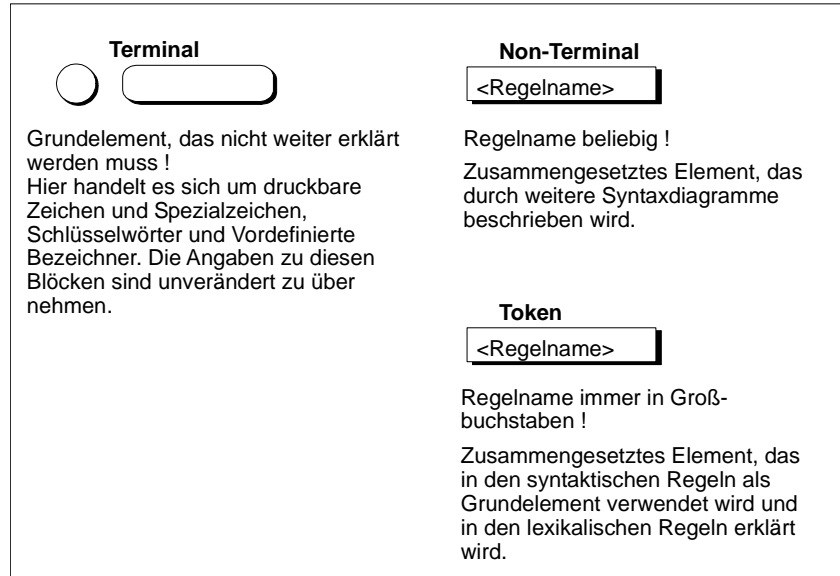


Das Syntaxdiagramm wird von links nach rechts gelesen. Dabei sind die folgenden Regelstrukturen zu beachten:

- Sequenz: Folge von Blöcken
- Option: Überspringbarer Zweig
- Iteration: Wiederholung von Zweigen
- Alternative: Verzweigung

Welche Arten von Blöcken gibt es?

Ein Block ist ein Grundelement oder ein Element, das wiederum aus Blöcken zusammengesetzt ist. Folgendes Bild zeigt die Symbolarten, die den Blöcken entsprechen:



15.1.2 Regeln

Die Regeln, die Sie für den Aufbau Ihres S7-SCL-Programms verwenden können, sind in die Stufen **lexikalische** und **syntaktische** Regeln eingeteilt.

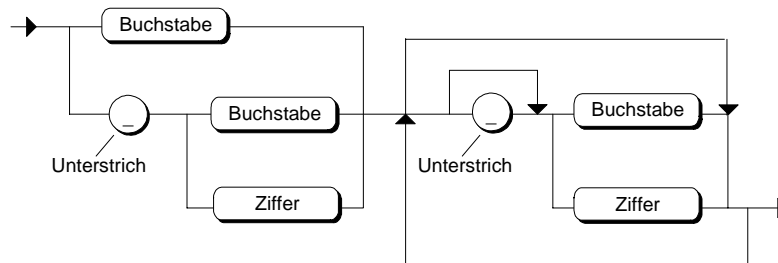
Lexikalische Regeln

Die lexikalischen Regeln beschreiben die Struktur der Elemente (Token), die bei der Lexikalanalyse des Compilers bearbeitet werden. Daher ist die Schreibweise nicht formatfrei und die Regeln sind streng einzuhalten. Das bedeutet insbesondere:

- Einfügen von Formatierungszeichen ist nicht erlaubt.
- Block- und Zeilenkommentare können nicht eingefügt werden.
- Attribute zu Bezeichnern können nicht eingefügt werden.

Das Beispiel zeigt die lexikalische Regel BEZEICHNER. Sie beschreibt den Aufbau eines Bezeichners (Namen), z. B.:

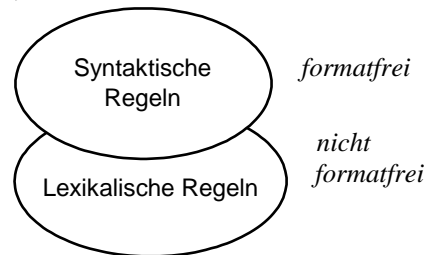
MESS_FELD_12
SOLLWERT_B_1



Syntaktische Regeln

Aufbauend auf den lexikalischen Regeln wird in den syntaktischen Regeln die Struktur von S7-SCL beschrieben. Im Rahmen dieser Regeln können Sie Ihr S7-SCL-Programm formatfrei erstellen:

SCL-Programm



Formales

Jede Regel hat einen Regelnamen, der vorangestellt ist. Wenn die Regel in einer übergeordneten Regel verwendet wird, so taucht der Regelname in einem Rechteck auf.

Ist der Regelname in Großbuchstaben geschrieben, so handelt es sich um ein Token, das in den lexikalischen Regeln beschrieben wird.

Semantik

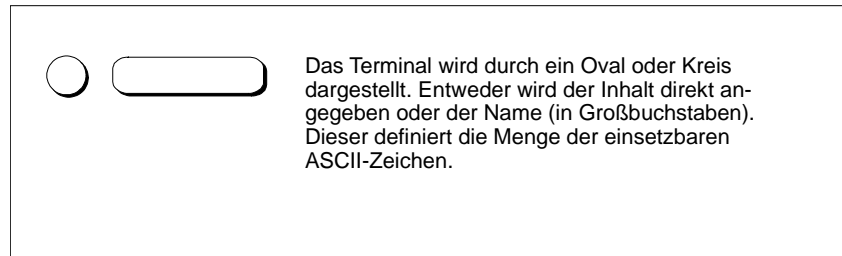
In den Regeln kann nur der formale Aufbau der Sprache dargestellt werden. Die Bedeutung, d.h. Semantik, geht nicht immer daraus hervor. Deshalb wird an wichtigen Stellen Zusatzinformation neben die Regeln geschrieben. Beispiele dafür sind:

- bei gleichartigen Elementen mit unterschiedlicher Bedeutung wird ein Zusatzname angegeben: z. B. in Regel Datumsangabe bei DEZIMALZIFFERNFOLGE Jahr, Monat oder Tag. Der Name deutet auf die Verwendung hin.
- Wichtige Einschränkungen werden neben den Regeln vermerkt: z. B. finden Sie bei der Regel Symbol den Hinweis, dass ein Symbol in der Symboltabelle definiert werden muss.

15.1.3 Terminale der lexikalischen Regeln

Definition

Ein Terminal ist ein Grundelement, das nicht durch eine weitere Regel, sondern verbal erklärt wird. In den Syntaxdiagrammen finden Sie es als folgendes Symbol:



In den folgenden Tabellen sind die Terminale spezifiziert durch die Angabe der Menge der Zeichensatzelemente aus dem ASCII-Zeichensatz.

Buchstaben und Ziffern

Buchstaben und Ziffern sind die hauptsächlich verwendeten Zeichen. Der BEZEICHNER besteht z. B. aus Buchstaben, Ziffern und Unterstrich.

Zeichen	Untergruppe	Zeichensatzelemente
Buchstabe	Großbuchstabe Kleinbuchstabe	A.. Z a.. z
Ziffer	Dezimalziffer	0.. 9
Oktalziffer	Oktalziffer	0.. 7
Hexadezimalziffer	Hexadezimalziffer	0.. 9, A.. F, a.. f
Bit	Binärziffer	0, 1

Druckbare Zeichen und Spezialzeichen

Der vollständige, erweiterte ASCII Zeichensatz ist benutzbar in Strings, Kommentaren und Symbolen.

Zeichen	Untergruppe	Zeichensatzelemente
druckbares Zeichen	abhängig vom verwendeten Zeichencode. Bei ASCII-Code z. B. ab Dezimaläquivalent 31, ohne DEL und ohne die folgenden Ersatzzeichen:	alle druckbaren Zeichen
Ersatzzeichen	Dollarzeichen Apostroph	\$,
Steuerzeichen	\$P oder \$p \$L oder \$l \$R oder \$r \$T oder \$t \$N oder \$n	Seitenumbruch (formfeed, page) Zeilenumbruch (linefeed) Wagenrücklauf (carriage return) Tabulator neue Zeile
Ersatzdarstellung im Hexacode	\$hh	beliebige Zeichen, aufnehmbar als Hexacode (hh)

15.1.4 Formatierungs-, Trennzeichen und Operationen

In den lexikalischen Regeln

In folgender Tabelle finden Sie die Verwendung einzelner Zeichen des ASCII-Zeichensatzes als Formatierungs- und Trennzeichen im Bereich der lexikalischen Regeln.

Zeichen	Beschreibung
:	Trennzeichen zwischen Stunden, Minuten und Sekunden Attribute
.	Trennzeichen für Absolute Adressierung Realzahl- und Zeitintervalldarstellung
' '	Zeichen und Zeichenkette
" "	Einleitungszeichen für Symbol nach den Regeln der Symboltabelle
_ Unterstrich	Trennzeichen für Zahlenwerte in Konstanten kann in BEZEICHNERN vorkommen
\$	Fluchtsymbol zur Angabe von Steuerzeichen oder Ersatzzeichen
\$> \$<	Stringunterbrechung, falls der String nicht in eine Zeile passt oder falls Kommentare eingefügt werden sollen.

Für Konstanten

In folgender Tabelle finden Sie die Verwendung einzelner Zeichen und Zeichenketten für Konstanten im Bereich der lexikalischen Regeln. Die Tabelle gilt für deutsche und englische Mnemonik.

Präfix	Kennzeichen für	Lexikalische Regel
BOOL#	Typisierte Konstante vom Typ BOOL	BIT-Konstante
BYTE#	Typisierte Konstante vom Typ BYTE	BIT-Konstante
WORD#	Typisierte Konstante vom Typ WORD	BIT-Konstante
DWORD#	Typisierte Konstante vom Typ DWORD	BIT-Konstante
INT#	Typisierte Konstante vom Typ INT	Ganzzahl-Konstante
DINT#	Typisierte Konstante vom Typ DINT	Ganzzahl-Konstante
REAL#	Typisierte Konstante vom Typ REAL	REAL-Konstante
CHAR#	Typisierte Konstante vom Typ CHAR	CHAR-Konstante
2#	Numerische Konstante	Binärziffernfolge
8#	Numerische Konstante	Oktalziffernfolge
16#	Numerische Konstante	Hexadezimalziffernfolge
D#	Zeitangabe	DATUM
DATE#	Zeitangabe	DATUM
DATE_AND_TIME#	Zeitangabe	DATUM UND ZEIT
DT#	Zeitangabe	DATUM UND ZEIT
E	Trennzeichen für REALZAHL-Konstante	Exponent
e	Trennzeichen für REALZAHL-Konstante	Exponent

Präfix	Kennzeichen für	Lexikalische Regel
D	Trennzeichen für Zeitintervall (Day)	Tage (Regel: Stufendarstellung)
H	Trennzeichen für Zeitintervall (Hour)	Stunden: (Regel: Stufendarstellung)
M	Trennzeichen für Zeitintervall (Minutes)	Minuten : (Regel: Stufendarstellung)
MS	Trennzeichen für Zeitintervall (Milliseconds)	Millisekunden: (Regel: Stufendarstellung)
S	Trennzeichen für Zeitintervall (Seconds)	Sekunden: (Regel: Stufendarstellung)
T#	Zeitangabe	ZEITDAUER
TIME#	Zeitangabe	ZEITDAUER
TIME_OF_DAY#	Zeitangabe	TAGESZEIT
TOD#	Zeitangabe	TAGESZEIT

In den syntaktischen Regeln

In folgender Tabelle finden Sie die Verwendung einzelner Zeichen als Formatierungs- und Trennzeichen im Bereich der syntaktischen Regeln sowie bei Kommentar und Attributen.

Zeichen	Beschreibung	Syntaktische Regel, Kommentar oder Attribut
:	Trennzeichen zur Typangabe, bei Anweisung nach Sprungmarke	Variablendeklaration, Instanzdeklaration, Funktion, Anweisungsteil, CASE-Anweisung
;	Abschluss einer Vereinbarung oder Anweisung	Variablendeklaration, Anweisungsteil, DB-Zuweisungsteil, Konstantenblock, Sprungmarkenblock, Komponentendeklaration
,	Trennzeichen für Listen und Sprungmarkenblöcke	Variablendeklaration, Array-Datentyp Spezifikation, Feld-Initialisierungsliste, FB-Parameter, FC-Parameter, Wertliste, Instanzdeklaration
..	Bereichsangabe	Array-Datentyp Spezifikation, Wertliste
.	Trennzeichen für FB- und DB-Name, Absolute Adressierung	FB-Aufruf, strukturierte Variable
()	Aufruf Funktion und Funktionsbaustein Klammerung in Ausdrücken, Initialisierungsliste für Arrays	Funktionsaufruf, FB-Aufruf, Ausdruck, Feld-Initialisierungsliste, Einfache Multiplikation, Potenzausdruck
[]	Array-Vereinbarung, strukturierte Variable Teil Array, Indizierung bei globalen Variablen und Strings	Array-Datentyp Spezifikation, STRING-Datentypspezifikation
(* *)	Kommentarblock	siehe "Lexikalische Regeln"
//	Zeilenkommentar	siehe "Lexikalische Regeln"
{ }	Attributblock	Angabe von Attributen
%	Einleitung für Direktbezeichner	Um IEC-konform zu programmieren, kann %M4.0 anstatt M4.0 verwendet werden.
#	Einleitung für ein Nicht-Schlüsselwort	Kennzeichnet einen Bezeichner als Nicht-Schlüsselwort, z.B. #FOR.

Operationen

In folgender Tabelle sind alle S7-SCL Operationen dargestellt, Schlüsselworte, z. B. AND und die üblichen Operationen als einzelne Zeichen. Diese Tabelle gilt für deutsche und englische Mnemonik.

Operation	Beschreibung	Syntaktische Regel
:=	Zuweisungsoperation, Anfangszuweisung, Datentypinitialisierung	Wertzuweisung, DB-Zuweisungsteil, Konstantenblock, Ausgangs- /Durchgangs-Zuweisung, Eingangs- Zuweisung, Durchgangs-Zuweisung
+, -	Arithmetische Operationen: unäre Operationen, Vorzeichen	Ausdruck, einfacher Ausdruck, Potenzausdruck
+, -, *, / MOD; DIV	Arithmetische Basisoperationen	Arithmetischer Basisoperation, einfache Multiplikation
**	Arithmetische Operationen: Potenzoperation	Ausdruck
NOT	Logische Operationen: Negation	Ausdruck, Operand
AND, &, OR; XOR,	Logische Basisoperationen	Logische Basisoperation
<, >, <=, >=, =, <>	Vergleichsoperation	Vergleichsoperation

15.1.5 Schlüsselwörter und vordefinierte Bezeichner

In folgender Tabelle finden Sie Schlüsselwörter von S7-SCL und vordefinierte Bezeichner alphabetisch aufgelistet. Dazu wird eine Beschreibung sowie die syntaktische Regel angegeben, in der sie als Terminale verwendet werden. Schlüsselwörter sind generell unabhängig von der Mnemonik.

Schlüsselwörter	Beschreibung	Syntaktische Regel
AND	Logische Operation	Logische Basisoperation
ANY	Bezeichnung für ANY Datentyp	Parameter-Datentyp Spezifikation
ARRAY	Einleitung der Spezifikation eines Arrays, danach folgt zwischen "[" und "]" die Indexliste	Array-Datentyp Spezifikation
AT	Deklariert eine Sicht auf eine Variable	Variablendeklaration
BEGIN	Einleitung Anweisungsteil bei Codebausteinen oder Initialisierungsteil bei Datenbaustein	Organisationsbaustein, Funktion, Funktionsbaustein, Datenbaustein
BLOCK_DB	Bezeichnung für Datentyp BLOCK_DB	Parameter-Datentyp Spezifikation
BLOCK_FB	Bezeichnung für Datentyp BLOCK_FB	Parameter-Datentyp Spezifikation
BLOCK_FC	Bezeichnung für Datentyp BLOCK_FC	Parameter-Datentyp Spezifikation
BLOCK_SDB	Bezeichnung für Datentyp BLOCK_SDB	Parameter-Datentyp Spezifikation
BOOL	Elementarer Datentyp für binäre Daten	Bitdatentyp
BY	Einleitung der Schrittweite	FOR-Anweisung
BYTE	Elementarer Datentyp	Bitdatentyp
CASE	Einleitung Kontrollanweisung zur Selektion	CASE-Anweisung
CHAR	Elementarer Datentyp	Zeichentyp
CONST	Einleitung für Definition von Konstanten	Konstantenblock
CONTINUE	Steueranweisung für FOR-, WHILE und REPEAT-Schleife	CONTINUE-Anweisung
COUNTER	Datentyp für Zähler, nur verwendbar in Parameterblock	Parameter-Datentyp Spezifikation
DATA_BLOCK	Einleitung des Datenbausteins	Datenbaustein
DATE	Elementarer Datentyp für Datum	Zeittyp
DATE_AND_TIME	Zusammengesetzter Datentyp für Datum und Uhrzeit	DATE_AND_TIME
DINT	Elementarer Datentyp für Ganzzahl (Integer) doppelter Genauigkeit	Numerischer Datentyp
DIV	Operation für Division	Arithmetischer Basisoperation, einfache Multiplikation
DO	Einleitung des Anweisungsteils bei FOR-Anweisung	FOR-Anweisung, WHILE-Anweisung
DT	Elementarer Datentyp für Datum und Uhrzeit	DATE_AND_TIME
DWORD	Elementarer Datentyp Doppelwort	Bitdatentyp
ELSE	Einleitung des Falls, wenn keine Bedingung erfüllt ist	IF-Anweisung, CASE-Anweisung
ELSIF	Einleitung der Alternativ-Bedingung	IF-Anweisung
EN	Flag zur Bausteinfreigabe	

Schlüsselwörter	Beschreibung	Syntaktische Regel
ENO	Fehlerflag des Bausteins	
END_CASE	Abschluss der CASE-Anweisung	CASE-Anweisung
END_CONST	Abschluss bei Definition von Konstanten	Konstantenblock
END_DATA_BLOCK	Abschluss des Datenbausteins	Datenbaustein
END_FOR	Abschluss der FOR-Anweisung	FOR-Anweisung
END_FUNCTION	Abschluss der Funktion	Funktion
END_FUNCTION_BLOCK	Abschluss des Funktionsbausteins	Funktionsbaustein
END_IF	Abschluss der IF-Anweisung	IF-Anweisung
END_LABEL	Abschluss des Sprungmarkenblocks	Sprungmarkenblock
END_TYPE	Abschluss des UDT	Anwenderdefinierter Datentyp
END_ORGANIZATION_BLOCK	Abschluss des Organisationsbausteins	Organisationsbaustein
END_REPEAT	Abschluss der REPEAT-Anweisung	REPEAT-Anweisung
END_STRUCT	Abschluss der Spezifikation einer Struktur	Struktur-Datentyp Spezifikation
END_VAR	Abschluss eines Deklarationsblocks	temporärer Variablenblock, statischer Variablenblock, Parameterblock
END_WHILE	Abschluss der WHILE-Anweisung	WHILE-Anweisung
EXIT	Direkter Ausstieg aus der Schleifenbearbeitung	EXIT
FALSE	Vordefinierte boolesche Konstante: Logische Bedingung nicht erfüllt, Wert gleich 0	
FOR	Einleitung der Kontrollanweisung zur Schleifenbearbeitung	FOR-Anweisung
FUNCTION	Einleitung der Funktion	Funktion
FUNCTION_BLOCK	Einleitung des Funktionsbausteins	Funktionsbaustein
GOTO	Anweisung zur Durchführung eines Sprungs zu einer Sprungmarke	Programmsprung
IF	Einleitung Kontrollanweisung für Selektion	IF-Anweisung
INT	Elementarer Datentyp für Ganzzahl (Integer) einfacher Genauigkeit	Numerischer Datentyp
LABEL	Einleitung des Sprungmarkenblocks	Sprungmarkenblock
MOD	Arithmetische Operation für Divisionsrest	Arithmetische Basisoperation, einfache Multiplikation
NIL	Nullpointer	
NOT	Logische Operation, gehört zu den Unären Operationen	Ausdruck, Operand
OF	Einleitung der Datentypspezifikation	Array-Datentyp-Spezifikation, CASE-Anweisung
OK	Flag, das aussagt, ob die Anweisungen eines Bausteins fehlerfrei abgearbeitet wurden	
OR	Logische Operation	Logische Basisoperation

Schlüsselwörter	Beschreibung	Syntaktische Regel
ORGANIZATION_BLOCK	Einleitung des Organisationsbausteins	Organisationsbaustein
POINTER	Zeiger-Datentyp, nur erlaubt in Parameterdeklaration im Parameterblock, wird nicht in S7-SCL bearbeitet	siehe Kapitel "Globale Daten".
PROGRAM	Einleitung des Anweisungsteils eines FB (Synonym zu FUNCTION_BLOCK)	Funktionsbaustein
REAL	Elementarer Datentyp	Numerischer Datentyp
REPEAT	Einleitung der Kontrollanweisung zur Schleifenbearbeitung	REPEAT-Anweisung
RET_VAL	Rückgabewert einer Funktion	Funktion
RETURN	Steueranweisung zur Rückkehr von Unterprogramm	RETURN-Anweisung
S5TIME	Elementarer Datentyp für Zeitangaben, spezielles S5-Format	Zeittyp
STRING	Datentyp für Zeichenkette	STRING-Datentyp Spezifikation
STRUCT	Einleitung der Spezifikation einer Struktur, danach folgt Liste der Komponenten	STRUCT-Datentyp Spezifikation
THEN	Einleitung der Folgeaktionen, wenn Bedingung erfüllt	IF-Anweisung
TIME	Elementarer Datentyp für Zeitangaben	Zeittyp
TIMER	Datentyp für Zeitglied, nur verwendbar in Parameterblock	Parameter-Datentyp Spezifikation
TIME_OF_DAY	Elementarer Datentyp für Tageszeit	Zeittyp
TO	Einleitung des Endwerts	FOR-Anweisung
TOD	Elementarer Datentyp für Tageszeit	Zeittyp
TRUE	Vordefinierte Boolesche Konstante: Logische Bedingung erfüllt, Wert ungleich 0	
TYPE	Einleitung des UDT	Anwenderdefinierter Datentyp
VAR	Einleitung eines Deklarationsblocks	statischer Variablenblock
VAR_TEMP	Einleitung eines Deklarationsblocks	temporärer Variablenblock
UNTIL	Einleitung der Abbruchbedingung für REPEAT-Anweisung	REPEAT-Anweisung
VAR_INPUT	Einleitung eines Deklarationsblocks	Parameterblock
VAR_IN_OUT	Einleitung eines Deklarationsblocks	Parameterblock
VAR_OUTPUT	Einleitung eines Deklarationsblocks	Parameterblock
WHILE	Einleitung der Kontrollanweisung zur Schleifenbearbeitung	WHILE-Anweisung
WORD	Elementarer Datentyp Wort	Bitdatentyp
VOID	Kein Rückgabewert bei einem Funktionsaufruf	Funktion
XOR	Logische Operation	Logische Basisoperation

15.1.6 Operandenkennzeichen und Bausteinschlüsselwörter

Globale Systemdaten

In folgender Tabelle finden Sie die deutsche Mnemonik der S7-SCL Operandenkennzeichen mit Beschreibung alphabetisch aufgelistet:

- Angabe des Operandenkennzeichens:
Speicher-Präfix (A, E, M, PA, PE) oder Datenbaustein (D)
- Angabe der Größe des Datenelements:
Größen-Präfix (optional oder B, D, W, X)

Die Mnemonik stellt eine Kombination zwischen dem Operandenkennzeichen (Speicher-Präfix oder D für Datenbaustein) und Größen-Präfix dar. Beides sind lexikalische Regeln. Die Tabelle ist nach der deutschen Mnemonik sortiert, die entsprechende englische Mnemonik wird dazu angegeben.

Deutsche Mnemonik	Englische Mnemonik	Speicher-Präfix oder Datenbaustein	Größen-Präfix
A	Q	Ausgang (über Prozessabbild)	Bit
AB	QB	Ausgang (über Prozessabbild)	Byte
AD	QD	Ausgang (über Prozessabbild)	Doppelwort
AW	QW	Ausgang (über Prozessabbild)	Wort
AX	QX	Ausgang (über Prozessabbild)	Bit
D	D	Datenbaustein	Bit
DB	DB	Datenbaustein	Byte
DD	DD	Datenbaustein	Doppelwort
DW	DW	Datenbaustein	Wort
DX	DX	Datenbaustein	Bit
E	I	Eingang (über Prozessabbild)	Bit
EB	IB	Eingang (über Prozessabbild)	Byte
ED	ID	Eingang (über Prozessabbild)	Doppelwort
EW	IW	Eingang (über Prozessabbild)	Wort
EX	IX	Eingang (über Prozessabbild)	Bit
M	M	Merker	Bit
MB	MB	Merker	Byte
MD	MD	Merker	Doppelwort
MW	MW	Merker	Wort
MX	MX	Merker	Bit
PAB	PQB	Ausgang (Peripherie direkt)	Byte
PAD	PQD	Ausgang (Peripherie direkt)	Doppelwort
PAW	PQW	Ausgang (Peripherie direkt)	Wort
PEB	PIB	Eingang (Peripherie direkt)	Byte
PED	PID	Eingang (Peripherie direkt)	Doppelwort
PEW	PIW	Eingang (Peripherie direkt)	Wort

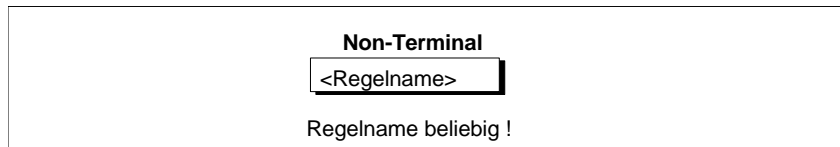
Bausteinschlüsselwörter

Werden für die absolute Adressierung von Bausteinen verwendet. Die Tabelle ist nach der deutschen Mnemonik sortiert, die entsprechende englische Mnemonik wird dazu angegeben.

Deutsche Mnemonik	Englische Mnemonik	Speicher-Präfix oder Datenbaustein
DB	DB	Datenbaustein (Data_Block)
FB	FB	Funktionsbaustein (Function_Block)
FC	FC	Funktion (Function)
OB	OB	Organisationsbaustein (Organization_Block)
SDB	SDB	Systemdatenbaustein (System_Data_Block)
SFC	SFC	Systemfunktion (System_Function)
SFB	SFB	Systemfunktionsbaustein (System_Function_Block)
T	T	Zeitglied (Timer)
UDT	UDT	globaler bzw. anwenderdefinierter Datentyp (Userdefined_Data_Type)
Z	C	Zähler (Counter)

15.1.7 Übersicht Non-Terminals

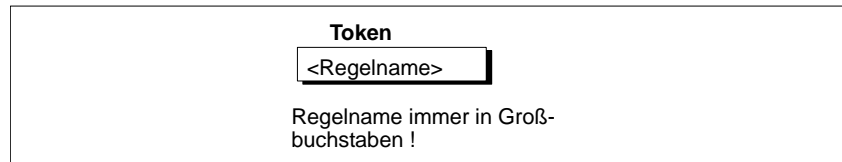
Ein Non-Terminal ist ein zusammengesetztes Element, das durch eine weitere Regel beschrieben wird. Das Non-Terminal wird durch einen Kasten dargestellt. Der Name im Kasten entspricht dem Regelnamen der weiterführenden Regel.



Das Element kommt in den lexikalischen und den syntaktischen Regeln vor.

15.1.8 Übersicht Token

Ein Token ist ein zusammengesetztes Element, das in den syntaktischen Regeln als Grundelement verwendet wird und in den lexikalischen Regeln erklärt wird. Das Token wird durch ein Rechteck dargestellt. Der NAME, in Großbuchstaben, entspricht dem Regelnamen der weiterführenden lexikalischen Regeln (ohne Rechteck).



Die definierten Token stellen Bezeichner dar, die als Ergebnisse der lexikalischen Regeln ermittelt wurden. Diese Token beschreiben die:

- Bezeichner
- Namensvergabe bei S7-SCL
- Vordefinierten Konstanten und Flags

15.1.9 Bezeichner

Bezeichner

Mit Bezeichnern können Sie Sprachobjekte von S7-SCL ansprechen. Folgende Tabelle informiert Sie über die Klassen von Bezeichnern.

Bezeichnerart	Bemerkungen, Beispiele
Schlüsselwörter	z. B. Steueranweisungen <code>BEGIN, DO, WHILE</code>
Vordefinierte Namen	Namen von <ul style="list-style-type: none"> • Standard-Datentypen (z. B. <code>BOOL, BYTE, INT</code>) • vordefinierte Standardfunktionen z. B. <code>ABS</code> • Standardkonstanten <code>TRUE</code> und <code>FALSE</code>
Operandenkennzeichen bei Absolutbezeichnern	für globale Systemdaten und Datenbausteine: z. B. <code>E1.2, MW10, FC20, T5, DB30, DB10.D4.5</code>
freie wählbare Namen nach der Regel <code>BEZEICHNER</code>	Namen von <ul style="list-style-type: none"> • vereinbarten Variablen • Strukturkomponenten • Parametern • vereinbarten Konstanten • Sprungmarken
Symbole der Symboltabelle	Erfüllen entweder die lexikalische Regel <code>BEZEICHNER</code> oder die lexikalische Regel <code>Symbol</code> , d.h. in Hochkommata eingeschlossen, z. B. <code>"xyz"</code>

Groß- und Kleinschreibung

Bei den Schlüsselwörtern ist Groß- und Kleinschreibung nicht relevant. Seit S7-SCL Version 4.0 wird auch bei den vordefinierten Namen, sowie bei den frei wählbaren Namen, z. B. für Variablen, und bei den Symbolen aus der Symboltabelle Groß- und Kleinschreibung nicht mehr unterschieden. Folgende Tabelle gibt Ihnen einen Überblick.

Bezeichnerart	case-sensitive?
Schlüsselwörter	nein
vordefinierte Namen bei Standard-Datentypen	nein
Namen bei vordefinierten Standardfunktionen	nein
vordefinierte Namen bei Standardkonstanten	nein
Operandenkennzeichen bei Absolutbezeichnern	nein
freie Namen	nein
Symbole der Symboltabelle	nein

Die Namen für Standardfunktionen, z. B. BYTE_TO_WORD und ABS können also auch in Kleinbuchstaben geschrieben werden. Genauso die Parameter für Zeit- und Zählfunktionen, z. B. SV, se oder ZV.

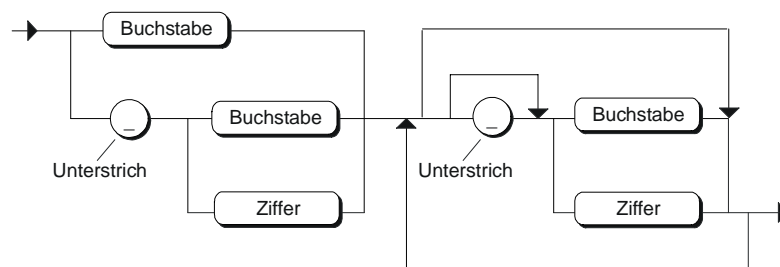
15.1.10 Namensvergabe bei S7-SCL

Vergabe von wählbaren Namen

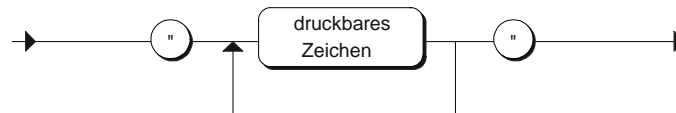
Für die Vergabe von Namen haben Sie generell 2 Möglichkeiten:

- Sie können Namen innerhalb S7-SCL selbst vergeben. Diese Namen müssen der Regel BEZEICHNER entsprechen. Die Regel BEZEICHNER können Sie für jeden Namen in S7-SCL benutzen.
- Sie können Namen über STEP 7 mit Hilfe der Symboltabelle einführen. Die Regel für diese Namen ist ebenfalls BEZEICHNER oder als erweiterte Möglichkeit SYMBOL. Durch die Angabe in Hochkommata kann das Symbol mit allen druckbaren Zeichen (z.B. Leerzeichen) gebildet werden.

BEZEICHNER



SYMBOL



Symbole müssen in der Symboltabelle definiert werden.

Regeln bei der Namensvergabe

Beachten Sie bitte Folgendes:

- Bei der Namensvergabe wählen Sie am besten eindeutige und aussagekräftige Namen, die zur Verständlichkeit des Programms beitragen.
- Achten Sie darauf, ob der Name schon vom System belegt ist, z. B. durch Bezeichner für Datentypen oder Standardfunktionen.
- Gültigkeitsbereich: Bei Namen, die global gültig sind, erstreckt sich der Gültigkeitsbereich über das gesamte Programm. Lokal gültige Namen gelten nur innerhalb eines Bausteins. Sie haben somit die Möglichkeit gleiche Namen in verschiedenen Bausteinen zu benutzen. Die folgende Tabelle informiert Sie über die Möglichkeiten, die Sie haben.

Einschränkungen

Bei der Vergabe von Namen müssen Sie einige Einschränkungen beachten.

Die Namen müssen in ihrem Gültigkeitsbereich eindeutig sein, d.h. Namen, die bereits innerhalb eines Bausteins vergeben wurden, dürfen nicht nochmals im selben Baustein benutzt werden. Weiterhin dürfen folgende vom System belegte Namen nicht benutzt werden:

- Namen von Schlüsselwörtern: z. B. CONST, END_CONST, BEGIN
- Namen von Operationen: z. B. AND, XOR
- Namen von vordefinierten Bezeichnern: z. B. Namen für Datentypen wie BOOL, STRING, INT
- Namen der vordefinierten Konstanten TRUE und FALSE
- Namen von Standardfunktionen: z. B. ABS, ACOS, ASIN, COS, LN
- Namen von Absolut- bzw. Operandenkennzeichen für globale Systemdaten: z. B. EB, EW, ED, AB, AW, AD MB, MD

Verwendung von BEZEICHNER

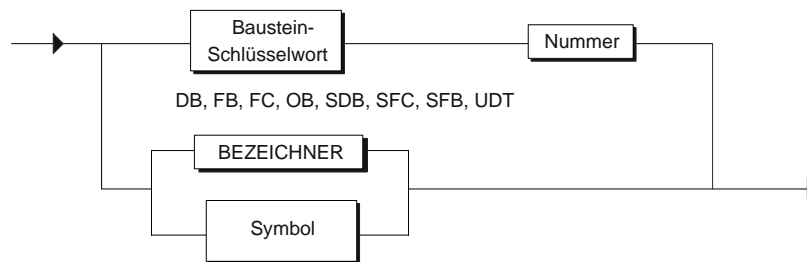
Folgende Tabelle zeigt Ihnen, für welche Fälle Sie Namen, die der Regel für BEZEICHNER entsprechen, angeben können.

BEZEICHNER	Beschreibung	Regel
Bausteinname	Symbolischer Name für Baustein	BAUSTEIN-BEZEICHNUNG, Funktionsaufruf
Name für Zeitglied und Zähler	Symbolischer Name für Zeitglied und Zähler	TIMER-BEZEICHNUNG, ZÄHLER-BEZEICHNUNG
Attributname	Name für ein Attribut	Attributzuweisung
Konstantenname	Vereinbarung symbolischer Konstante, Verwendung	Konstantenblock, Konstante
Sprungmarke	Vereinbarung Sprungmarke, Verwendung Sprungmarke	Sprungmarkenblock-Anweisungsteil, GOTO-Anweisung
Variablenname	Vereinbarung temporärer oder statischer Variable	Variablendeklaration, Einfache Variable, Strukturierte Variable
Lokaler Instanzname	Vereinbarung lokaler Instanzen	Instanzdeklaration, FB-Aufrufname

BAUSTEIN-BEZEICHNUNG

In der Regel BAUSTEIN-BEZEICHNUNG können Sie BEZEICHNER und Symbol alternativ einsetzen:

BAUSTEIN-BEZEICHNUNG



Analog zu BAUSTEIN-BEZEICHNUNG gelten auch die Regeln TIMER-BEZEICHNUNG und COUNTER-BEZEICHNUNG.

15.1.11 Vordefinierte Konstanten und Flags

Beide Tabellen gelten für deutsche und englische Mnemonik.

Konstanten

Mnemonik	Beschreibung
FALSE	Vordefinierte boolesche Konstante (Standardkonstante) mit dem Wert 0. Sie hat die logische Bedeutung, dass eine Bedingung nicht erfüllt ist.
TRUE	Vordefinierte boolesche Konstante (Standardkonstante) mit dem Wert 1. Sie hat die logische Bedeutung, dass eine Bedingung erfüllt ist.

Flags

Mnemonik	Beschreibung
EN	Flag zur Baustein-Freigabe
ENO	Fehlerflag des Bausteins
OK	Flag wird auf FALSE gesetzt, wenn eine Anweisung fehlerhaft bearbeitet wurde.

15.2 Lexikalische Regeln

15.2.1 Übersicht: Lexikalische Regeln

Die lexikalischen Regeln beschreiben die Struktur der Elemente (Token), die bei der Lexikalanalyse des Compilers bearbeitet werden. Daher ist die Schreibweise nicht formatfrei und die Regeln sind streng einzuhalten. Das bedeutet insbesondere:

- Einfügen von Formatierungszeichen ist nicht erlaubt.
- Block- und Zeilenkommentare können nicht eingefügt werden.
- Attribute zu Bezeichnern können nicht eingefügt werden.

15.2.2 Bezeichnungen

Regel	Syntaxdiagramm
Bezeichner	<p>BEZEICHNER</p>
Baustein-bezeichnung	<p>BAUSTEIN-BEZEICHNUNG</p>

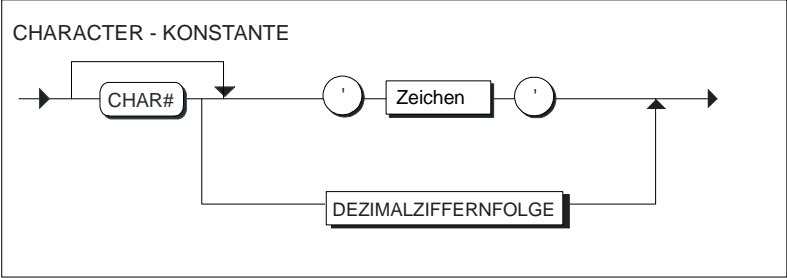
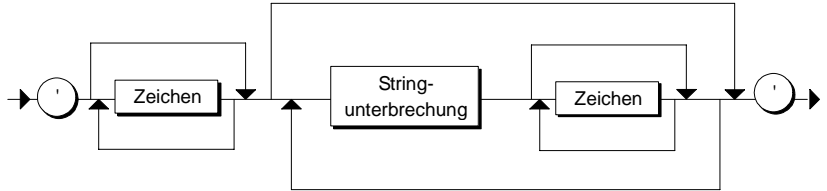
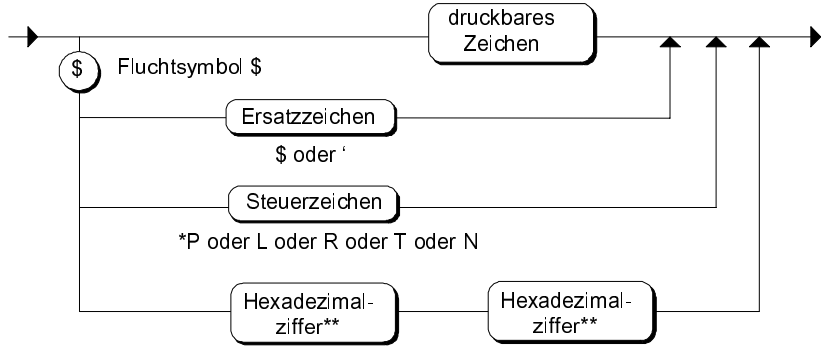
Regel	Syntaxdiagramm
Timerbezeichnung	
Zählerbezeichnung	
Baustein-Schlüsselwort	
Symbol	

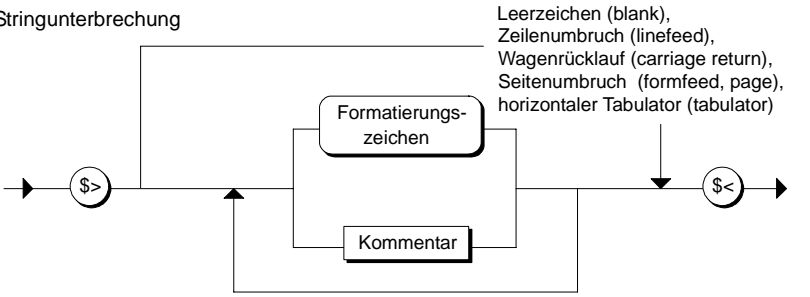
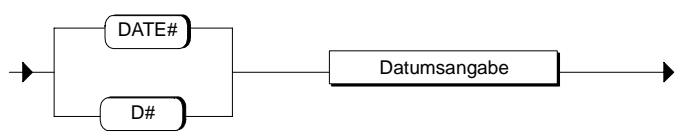
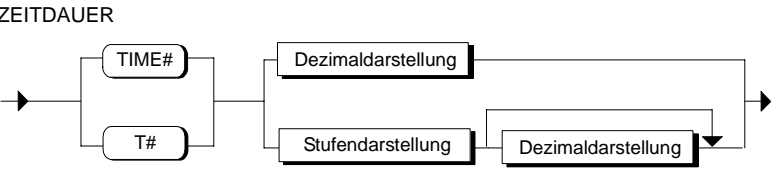
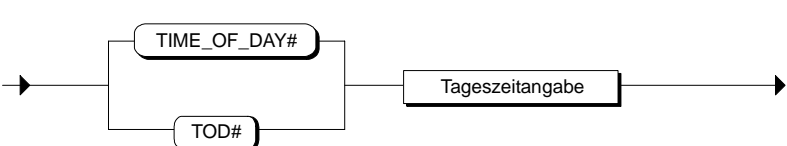
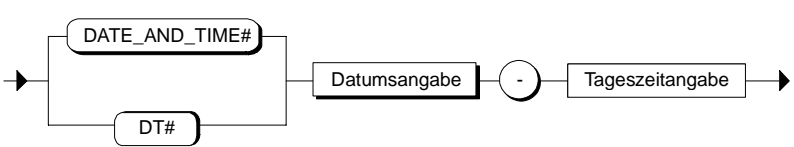
Regel	Syntaxdiagramm
Nummer	<p>The diagram shows a horizontal arrow representing the sequence of a number. A box labeled 'Ziffer' is positioned above the arrow. A line goes from the top of the 'Ziffer' box down, then left, then up, and then right, forming a loop that connects back to the start of the arrow, indicating that 'Ziffer' can be repeated.</p>


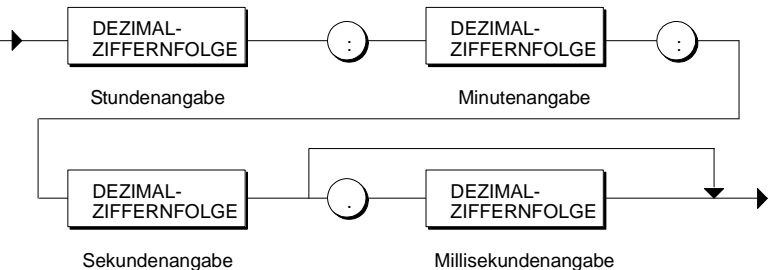
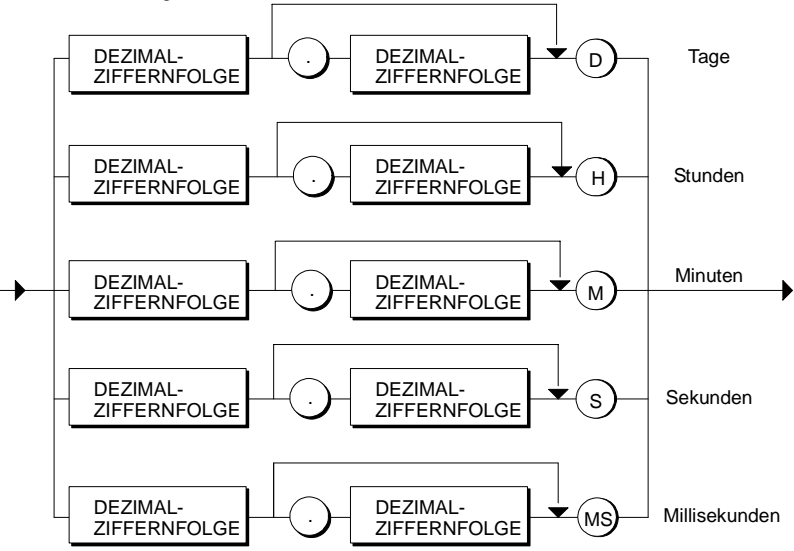
15.2.3 Konstanten

Regel	Syntaxdiagramm
Bit-Konstante	<p>BIT - KONSTANTE</p> <p>The diagram shows a vertical stack of four boxes: 'BOOL#', 'BYTE#', 'WORD#', and 'DWORD#'. Lines from these boxes converge into a single horizontal arrow pointing to a large box containing the following text: DEZIMALZIFFERNFOLGE OKTALZIFFERNFOLGE HEXADEZIMALZIFFERNFOLGE BINÄRZIFFERNFOLGE ZEICHEN (1)</p> <p>(1) nur bei Datentyp BYTE</p>
Ganzzahl-Konstante	<p>GANZZAHL - KONSTANTE</p> <p>The diagram shows a vertical stack of two boxes: 'INT#' and 'DINT#'. Lines from these boxes converge into a horizontal arrow pointing to a box with a '+' sign above it and a '-' sign below it. This box then leads to a large box containing the following text: DEZIMALZIFFERNFOLGE OKTALZIFFERNFOLGE HEXADEZIMALZIFFERNFOLGE BINÄRZIFFERNFOLGE ZEICHEN (1)</p> <p>(1) nur bei Datentyp INT</p>
Realzahl-Konstante	<p>REALZAHL - KONSTANTE</p> <p>The diagram shows a box 'REAL#' leading to a box with '+' and '-' signs. This box then branches into two paths. The top path goes through a box 'DEZIMAL-ZIFFERNFOLGE', then a box with a decimal point '.', then another 'DEZIMAL-ZIFFERNFOLGE' box. The bottom path goes through a 'DEZIMAL-ZIFFERNFOLGE' box, then a box with a decimal point and a dot '·', then another 'DEZIMAL-ZIFFERNFOLGE' box, and finally an 'Exponent' box. Both paths then converge into a final arrow.</p>

Regel	Syntaxdiagramm
Dezimalziffernfolge	<p>Dezimalziffernfolge</p> <p>Dezimalziffer: 0-9</p>
Binärziffernfolge	<p>Binärziffernfolge</p> <p>Binärziffer: 0 oder 1</p>
Oktalziffernfolge	<p>Oktalziffernfolge</p> <p>Oktalziffer</p>
Hexadezimalziffernfolge	<p>HEXADEZIMALZIFFER</p> <p>Hexadezimalziffer: 0-9 A-F</p>
Exponent	<p>Exponent</p> <p>DEZIMAL-ZIFFERNFOLGE</p>

Regel	Syntaxdiagramm
Character-Konstante	<p>CHARACTER - KONSTANTE</p> 
String-Konstante	<p>STRING - KONSTANTE</p> 
Zeichen	<p>Zeichen</p>  <p>Ersatzdarstellung im Hexacode</p> <p>* P=Seitenvorschub L=Zeilenvorschub R=Wagenruecklauf T=Tabulator N=neue Zeile</p> <p>** \$00 nicht zulaessig</p>

Regel	Syntaxdiagramm
Stringunterbrechung	<p>Stringunterbrechung</p> <p>Leerzeichen (blank), Zeilenumbruch (linefeed), Wagenrücklauf (carriage return), Seitenumbruch (formfeed, page), horizontaler Tabulator (tabulator)</p> 
Datum	<p>DATUM</p> 
zeitdauer	<p>ZEITDAUER</p>  <p>- Jede Zeiteinheit (z. B. Stunden, Minuten) darf nur 1 x angegeben werden. - Die Reihenfolge - Tage, Stunden, Minuten, Sekunden, Millisekunden - ist einzuhalten.</p>
Tageszeit	<p>TAGESZEIT</p> 
Datum und Zeit	<p>DATUM UND ZEIT</p> 

Regel	Syntaxdiagramm
Datumsgabe	<p>Datumsgabe</p>  <p style="text-align: center;"> Jahr Monat Tag </p>
Tageszeitangabe	<p>Tageszeitangabe</p>  <p style="text-align: center;"> Stundenangabe Minutenangabe </p> <p style="text-align: center;"> Sekundenangabe Millisekundenangabe </p>
Dezimaldarstellung	<p>Dezimaldarstellung</p>  <p style="text-align: right;"> Tage Stunden Minuten </p> <p style="text-align: right;"> Sekunden Millisekunden </p> <p>Der Einstieg in die Dezimaldarstellung ist nur bei noch nicht definierten Zeiteinheiten möglich.</p>

Regel	Syntaxdiagramm
Stufendarstellung	<p>Stufendarstellung</p> <p>The diagram illustrates the hierarchical structure of a time value. It is composed of three main parts, each representing a different unit of time:</p> <ul style="list-style-type: none"> Tage (Days): A box labeled 'DEZIMAL-ZIFFERNFOLGE' followed by a circle containing 'D' and a circle containing '-'. An arrow points from the start to the 'D' box, and another from the '-' box to the start of the 'Stunden' part. Stunden (Hours): A box labeled 'DEZIMAL-ZIFFERNFOLGE' followed by a circle containing 'H' and a circle containing '-'. An arrow points from the start of the 'Tage' part to the 'H' box, and another from the '-' box to the start of the 'Minuten' part. Minuten (Minutes): A box labeled 'DEZIMAL-ZIFFERNFOLGE' followed by a circle containing 'M' and a circle containing '-'. An arrow points from the start of the 'Stunden' part to the 'M' box, and another from the '-' box to the start of the 'Sekunden' part. Sekunden (Seconds): A box labeled 'DEZIMAL-ZIFFERNFOLGE' followed by a circle containing 'S' and a circle containing '-'. An arrow points from the start of the 'Minuten' part to the 'S' box, and another from the '-' box to the start of the 'Millisekunden' part. Millisekunden (Milliseconds): A box labeled 'DEZIMAL-ZIFFERNFOLGE' followed by a circle containing 'MS' and a circle containing '-'. An arrow points from the start of the 'Sekunden' part to the 'MS' box, and another from the '-' box to the final end arrow.

15.2.4 Absolutadressierung

Regel	Syntaxdiagramm
Einfacher Speicherzugriff	
Indizierter Speicherzugriff	
Operandenkennzeichen für Speicher	
Absoluter DB-Zugriff	
Indizierter DB-Zugriff	
Strukturierter DB-Zugriff	

Regel	Syntaxdiagramm
Operanden-Kennzeichen DB	
Speicher-Präfix	<p>Speicher-Präfix</p>
Größen-Präfix für Speicher und DB	<p>Größen-Präfix</p>
Adresse für Speicher und DB	<p>Adresse</p>
Zugriff auf lokale Instanz	

15.2.5 Kommentare

Folgende sind die wichtigsten Punkte, die beim Einbau von Kommentaren zu beachten sind:

- Schachtelung von Kommentaren ist zulässig, wenn die Option "Geschachtelte Kommentare zulassen" aktiviert ist.
- Der Einbau ist an beliebigen Stellen in den syntaktischen Regeln möglich, nicht aber in den lexikalischen Regeln.

Regel	Syntaxdiagramm
Kommentar	<p>The diagram shows a horizontal arrow starting from the left. A vertical line branches off upwards to a box labeled 'Zeilenkommentar'. Another vertical line branches off downwards to a box labeled 'Kommentarblock'. Both boxes have arrows pointing to the right, which then merge back into a single horizontal arrow pointing to the right.</p>
Zeilenkommentar	<p>Zeilenkommentar</p> <p>The diagram shows a horizontal arrow starting from the left. It passes through a circle containing two slashes '//'. Then it enters a box labeled 'druckbares Zeichen'. After the box, it passes through a circle containing 'CR'. Below the 'CR' circle is the text 'Wagenrücklauf (carriage return)'. An arrow loops from the bottom of the 'druckbares Zeichen' box back to the top of the 'CR' circle, indicating a loop.</p>
Kommentarblock	<p>Kommentarblock</p> <p>The diagram shows a horizontal arrow starting from the left. It passes through a circle containing '(*'. Then it enters a box labeled 'Zeichen'. After the box, it passes through a circle containing '*)'. An arrow loops from the bottom of the 'Zeichen' box back to the top of the '(*' circle, indicating a loop.</p>

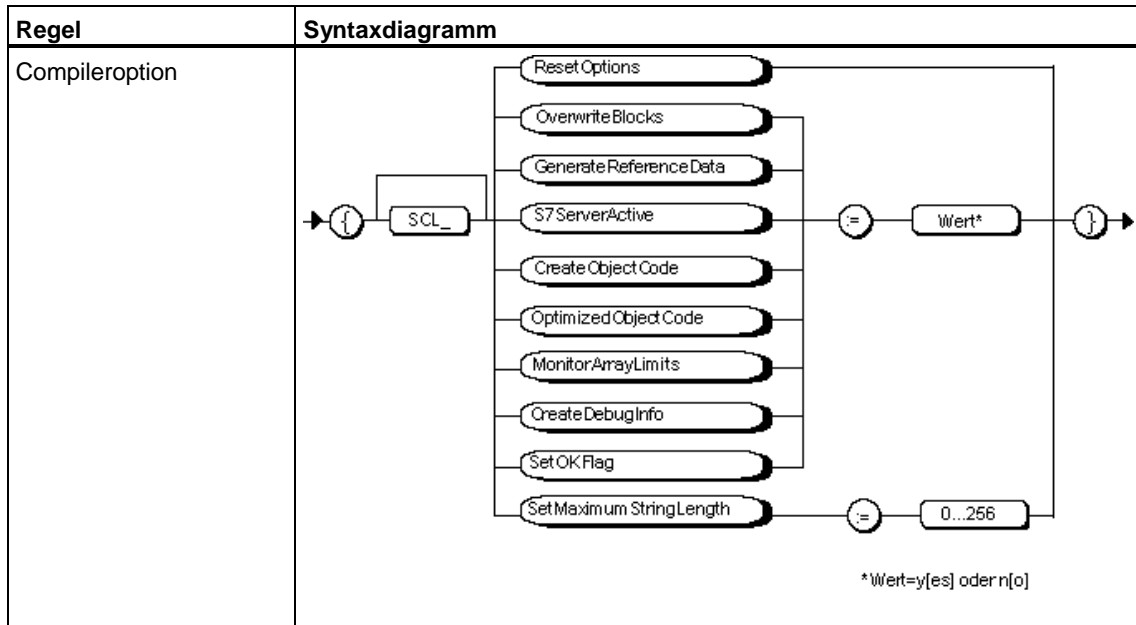
15.2.6 Bausteinattribute

Bausteinattribute können mit folgender Syntax nach der BAUSTEIN-BEZEICHNUNG und vor der Vereinbarung des ersten Variablen- oder Parameterblocks stehen.

Regel	Syntaxdiagramm
Überschrift	
Version	
Bausteinschutz	
Author	
Name	
Bausteinfamilie	
Systemattribute für Bausteine	<p>Systemattribute für Bausteine</p>

15.2.7 Compileroptionen

Compileroptionen stehen in der Quelle außerhalb der Bausteingrenzen in einer eigenen Zeile. Groß- und Kleinschreibung wird nicht unterschieden.



15.3 Syntaktische Regeln

15.3.1 Übersicht: Syntaktische Regeln

Aufbauend auf den lexikalischen Regeln wird in den syntaktischen Regeln die Struktur von S7-SCL beschrieben. Im Rahmen dieser Regeln können Sie Ihr S7-SCL-Programm formatfrei erstellen.

Jede Regel hat einen Regelnamen, der vorangestellt ist. Wenn die Regel in einer übergeordneten Regel verwendet wird, so taucht der Regelname in einem Rechteck auf.

Ist der Name im Rechteck in Großbuchstaben geschrieben, so handelt es sich um ein Token, das in den lexikalischen Regeln beschrieben wird.

Über Regelnamen in abgerundeten Rahmen oder Kreisen finden Sie Informationen im Kapitel "Formale Sprachbeschreibung".

Formatfreiheit

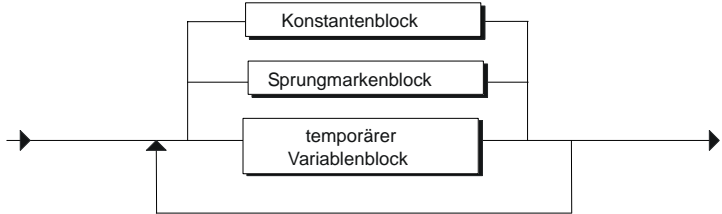
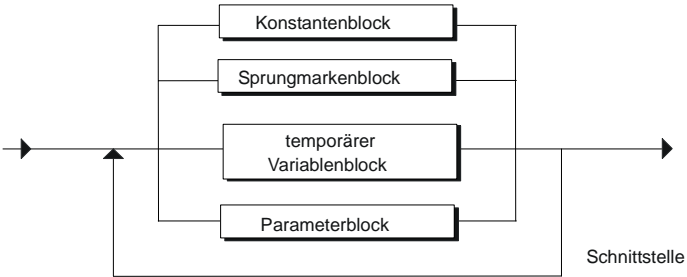
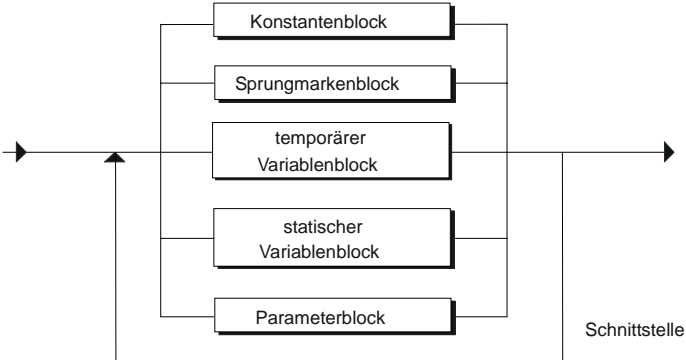
Formatfreiheit bedeutet:

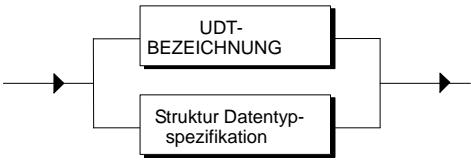
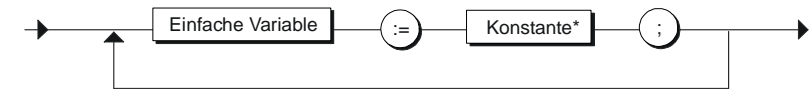
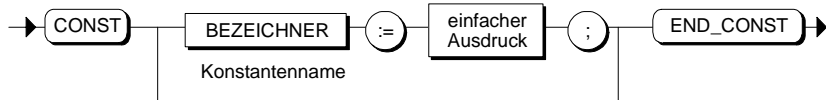
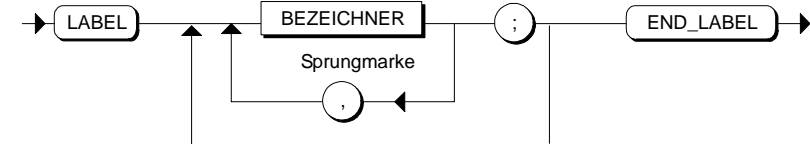
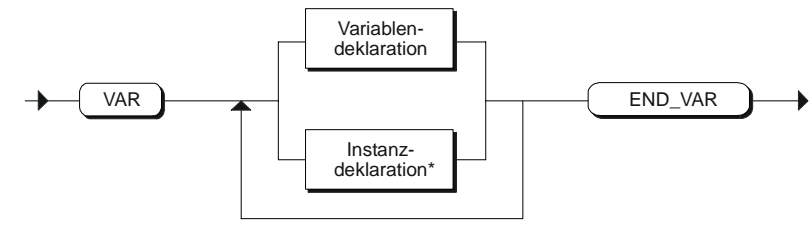
- Einfügen von Formatierungszeichen ist überall möglich
- Zeilenkommentare Kommentarblöcke können eingefügt werden

15.3.2 Gliederungen von S7-SCL-Quellen

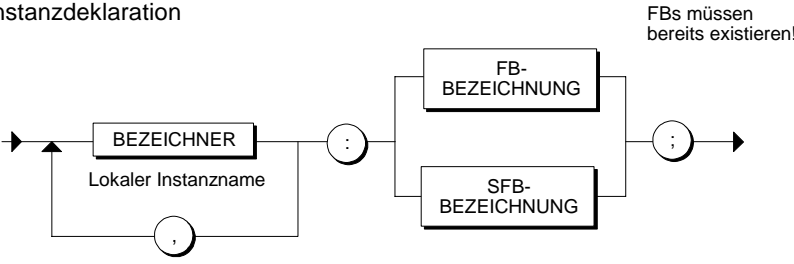
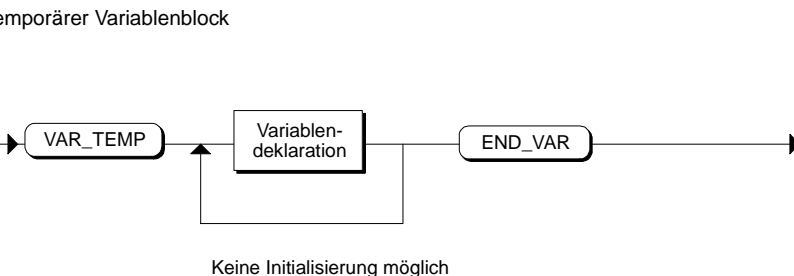
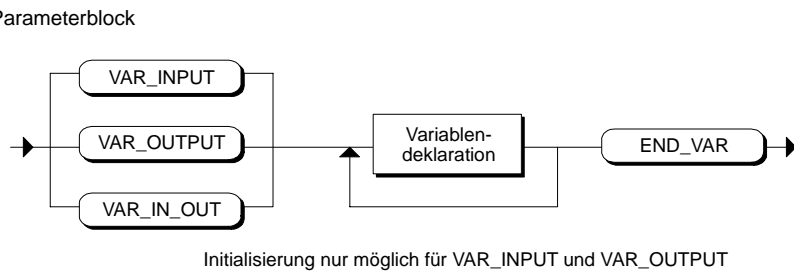
Regel	Syntaxdiagramm
S7-SCL-Programm	
S7-SCL-Programmeinheit	
Organisationsbaustein	<p>Organisationsbaustein</p>
<p>Funktion Beachten Sie, dass bei Funktionen ohne VOID im Anweisungsteil der Rückgabewert dem Funktionsnamen zugewiesen werden muss!</p>	<p>Funktion</p>

15.3.3 Aufbau der Vereinbarungsteile

Regel	Syntaxdiagramm
OB-Vereinbarungsteil	 <p>The diagram shows a sequence of three blocks: 'Konstantenblock', 'Sprungmarkenblock', and 'temporärer Variablenblock'. An input arrow enters from the left and splits to enter each block. The outputs of the first two blocks are connected to the input of the third block. A feedback loop arrow starts from the output of the 'temporärer Variablenblock' and returns to the input of the 'Konstantenblock'.</p>
FC-Vereinbarungsteil	 <p>The diagram shows a sequence of four blocks: 'Konstantenblock', 'Sprungmarkenblock', 'temporärer Variablenblock', and 'Parameterblock'. An input arrow enters from the left and splits to enter each block. The outputs of the first three blocks are connected to the input of the fourth block. A feedback loop arrow starts from the output of the 'Parameterblock' and returns to the input of the 'Konstantenblock'. The output of the 'Parameterblock' is also labeled 'Schnittstelle'.</p>
FB-Vereinbarungsteil	 <p>The diagram shows a sequence of five blocks: 'Konstantenblock', 'Sprungmarkenblock', 'temporärer Variablenblock', 'statischer Variablenblock', and 'Parameterblock'. An input arrow enters from the left and splits to enter each block. The outputs of the first four blocks are connected to the input of the fifth block. A feedback loop arrow starts from the output of the 'Parameterblock' and returns to the input of the 'Konstantenblock'. The output of the 'Parameterblock' is also labeled 'Schnittstelle'.</p>

Regel	Syntaxdiagramm
DB-Vereinbarungsteil	
DB-Zuweisungsteil	<p>DB-Zuweisungsteil</p>  <p style="text-align: right;">* in AWL-Schreibweise</p>
Konstantenblock	<p>Konstantenblock</p> 
Sprungmarkenblock	<p>Sprungmarkenblock</p> 
Statischer Variablenblock	<p>Statischer Variablenblock</p>  <p style="text-align: right;">* nur bei FB</p>

Regel	Syntaxdiagramm
Variablendeklaration	<p>1) Systemattribute für Parameter</p>
Datentyp-Initialisierung	<p>Initialisierung</p>
Feld-Initialisierungsliste	<p>Konstantenwiederholungsliste</p> <p>Feld-Initialisierungsliste</p>

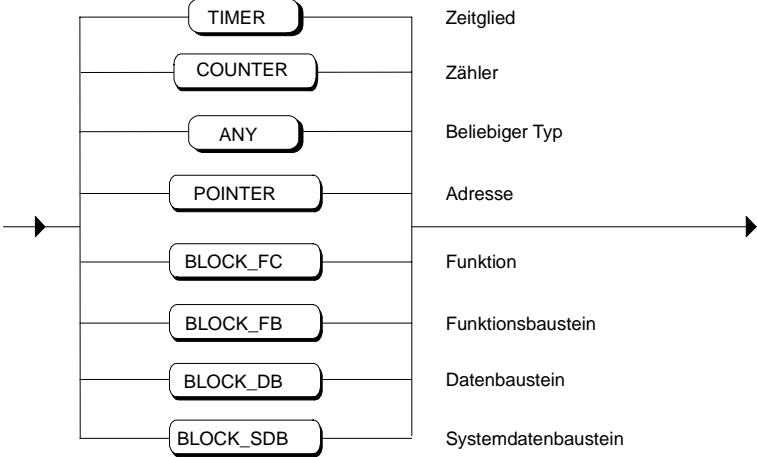
Regel	Syntaxdiagramm
<p>Instanzdeklaration (nur im Abschnitt VAR eines FB möglich)</p>	<p>Instanzdeklaration</p>  <p>FBs müssen bereits existieren!</p>
<p>Temporärer Variablenblock</p>	<p>Temporärer Variablenblock</p>  <p>Keine Initialisierung möglich</p>
<p>Parameterblock</p>	<p>Parameterblock</p>  <p>Initialisierung nur möglich für VAR_INPUT und VAR_OUTPUT</p>

Regel	Syntaxdiagramm
Datentyp-Spezifikation	<pre>graph LR; A[Elementarer Datentyp] --- B[DATE_AND_TIME]; B --- C[STRING-Datentyp Spezifikation]; C --- D[ARRAY-Datentyp Spezifikation]; D --- E[STRUCT-Datentyp Spezifikation]; E --- F[UDT-BEZEICHNUNG]; F --- G[Parameter-Datentyp Spezifikation];</pre> <p>The diagram shows a vertical list of seven boxes, each representing a different data type specification. From top to bottom, they are: 'Elementarer Datentyp', 'DATE_AND_TIME', 'STRING-Datentyp Spezifikation', 'ARRAY-Datentyp Spezifikation', 'STRUCT-Datentyp Spezifikation', 'UDT-BEZEICHNUNG', and 'Parameter-Datentyp Spezifikation'. A horizontal arrow on the left points to the middle of the list, and a horizontal arrow on the right points away from the middle of the list, indicating that the list is a choice of options for the 'Datentyp-Spezifikation' rule.</p>

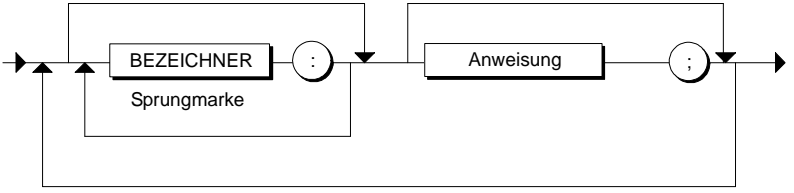
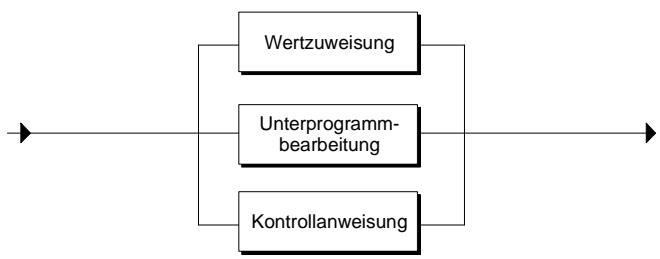
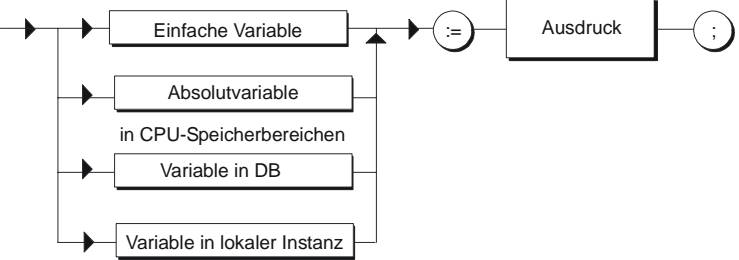
15.3.4 Datentypen in S7-SCL

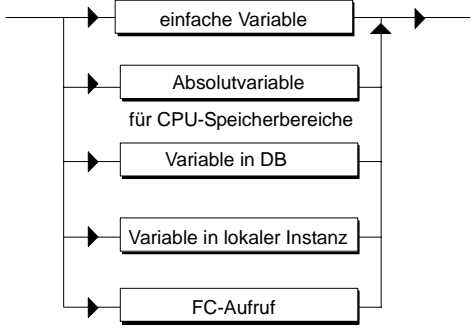
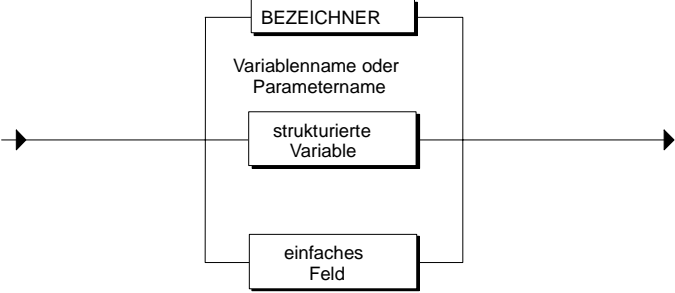
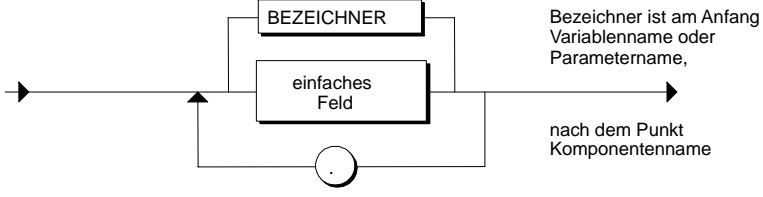
Regel	Syntaxdiagramm
Elementarer Datentyp	
Bitdatentyp	
Zeichentyp	
STRING-Datentyp Spezifikation	<p>STRING-Datentypspezifikation</p>
Numerischer Datentyp	

Regel	Syntaxdiagramm
Zeityp	<p>The diagram shows five terminal symbols in rounded rectangles: S5TIME, TIME, TIME_OF_DAY, TOD, and DATE. Lines from S5TIME and TIME converge to a label 'Zeit, S5-Format'. Lines from TIME and TIME_OF_DAY converge to a label 'Zeit'. Lines from TOD and DATE converge to a label 'Datum'. A line from TIME_OF_DAY and TOD converges to a label 'Tageszeit'. An arrow enters from the left and an arrow exits to the right.</p>
DATE_AND_TIME	<p>The diagram shows two terminal symbols: DATE_AND_TIME# and DT#. Lines from both converge to a box labeled 'Datumsangabe'. This is followed by a circle containing a minus sign '-', and then a box labeled 'Tageszeitangabe'. An arrow enters from the left and an arrow exits to the right.</p>
ARRAY-Datentyp Spezifikation	<p>The diagram shows a terminal symbol 'ARRAY' followed by a box 'Index₁', an ellipsis '...', and a box 'Index_n'. Lines from 'Index₁' and 'Index_n' converge to a circle containing a comma ','. Below this is the text 'max. 6 Dimensionen'. The sequence is enclosed in square brackets '[' and ']'. Below the brackets is a circle containing 'OF' and a box 'Datentyp-Spezifikation'. An arrow enters from the left and an arrow exits to the right.</p>
<p>STRUCT-Datentyp Spezifikation Vergessen Sie nicht, das Schlüsselwort END_STRUCT mit Semikolon abzuschließen !</p>	<p>The diagram shows a terminal symbol 'STRUCT', a box 'Komponentendeklaration', and a terminal symbol 'END_STRUCT'. Lines from 'Komponentendeklaration' and 'END_STRUCT' converge to a semicolon ';'. An arrow enters from the left and an arrow exits to the right.</p>
Komponentendeklaration	<p>The diagram shows a terminal symbol 'BEZEICHNER' (with 'Komponentennamen' below it), a circle containing a colon ':', a box 'Datentypspezifikation', a box 'Dateninitialisierung', and a terminal symbol ';' (with 'Komponentennamen innerhalb von Strukturen' below it). Lines from 'Datentypspezifikation' and 'Dateninitialisierung' converge to the semicolon. An arrow enters from the left and an arrow exits to the right.</p>

Regel	Syntaxdiagramm
Parametertyp Spezifikation	

15.3.5 Anweisungsteil

Regel	Syntaxdiagramm
Anweisungsteil	<p>Anweisungsteil</p> 
Anweisung	<p>Anweisung</p> 
Wertzuweisung	<p>Wertzuweisung</p> 

Regel	Syntaxdiagramm
Erweiterte Variable	<p>Erweiterte Variable</p>  <pre> graph LR Start(()) --> EV[Erweiterte Variable] EV --> EV1[einfache Variable] EV --> EV2[Absolutvariable für CPU-Speicherbereiche] EV --> EV3[Variable in DB] EV --> EV4[Variable in lokaler Instanz] EV --> EV5[FC-Aufruf] EV1 --> End(()) EV2 --> End EV3 --> End EV4 --> End EV5 --> End </pre>
Einfache Variable	 <pre> graph LR Start(()) --> EV[Einfache Variable] EV --> BE[BEZEICHNER Variablenname oder Parametername] EV --> SV[strukturierte Variable] EV --> EF[einfaches Feld] BE --> End(()) SV --> End EF --> End </pre>
Strukturierte Variable	 <pre> graph LR Start(()) --> SV[Strukturierte Variable] SV --> BE[BEZEICHNER] SV --> EF[einfaches Feld] BE --> End(()) EF --> End subgraph Note direction TB N1[Bezeichner ist am Anfang Variablenname oder Parametername, nach dem Punkt Komponentenname] end </pre>

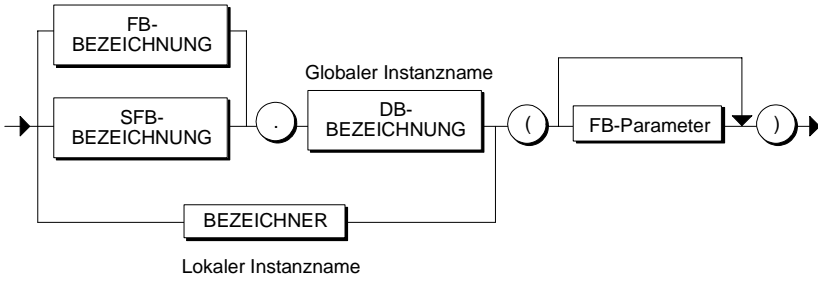
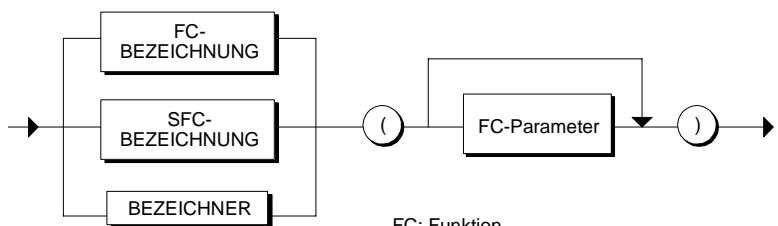
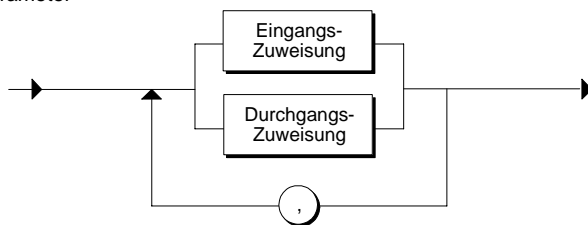
15.3.6 Wertzuweisungen

Regel	Syntaxdiagramm
Ausdruck	<p>Ausdruck</p> <p>The diagram for 'Ausdruck' shows a central box labeled 'Ausdruck' with an arrow pointing to it from the left and an arrow pointing away to the right. The diagram is enclosed in a larger box with an arrow pointing to it from the left and an arrow pointing away to the right. Inside this box, there are several paths:</p> <ul style="list-style-type: none"> A path through a box labeled 'Operand'. A path through a box labeled 'Ausdruck', then a box containing 'Logische Basisoperationen', 'Vergleichsoperationen', and 'Arithmetische Basisoperationen', and finally another box labeled 'Ausdruck'. A path through a box labeled 'Ausdruck', then a circle containing '**' (Potenz), then a box labeled 'Exponent', and finally another box labeled 'Ausdruck'. A path through a box containing '+', '-', and 'NOT' (unäres Plus, unäres Minus, Negation), then a box labeled 'Ausdruck', and finally another box labeled 'Ausdruck'. A path through a circle containing '(', then a box labeled 'Ausdruck', and finally a circle containing ')'. This path is shown as a separate line at the bottom of the diagram.
Einfacher Ausdruck	<p>The diagram for 'Einfacher Ausdruck' shows a central box labeled 'einfacher Ausdruck' with an arrow pointing to it from the left and an arrow pointing away to the right. The diagram is enclosed in a larger box with an arrow pointing to it from the left and an arrow pointing away to the right. Inside this box, there are two paths:</p> <ul style="list-style-type: none"> A path through a box labeled 'einfacher Ausdruck', then a box containing '+' and '-' (addition and subtraction), and finally another box labeled 'einfache Multiplikation'. A path through a box labeled 'einfache Multiplikation'.

Regel	Syntaxdiagramm
Einfache Multiplikation	
Operand	
Erweiterte Variable	<p>Erweiterte Variable</p>

Regel	Syntaxdiagramm
Konstante	<p>Konstante</p>
Exponent	<p>Exponent</p>
Logische Basisoperation	
Arithmetische Basisoperation	<p>Arithmetische Basisoperation</p>
Vergleichsoperation	<p>Vergleichsoperation</p>

15.3.7 Aufruf von Funktionen und Funktionsbausteinen

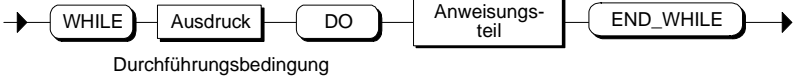
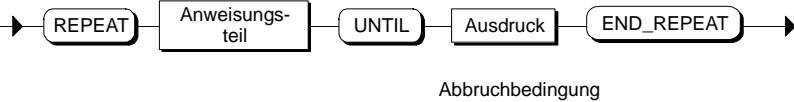



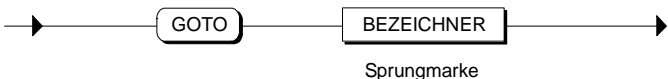
Regel	Syntaxdiagramm
<p>FB-Aufruf</p>	<p>FB-Aufruf</p> <p>FB: Funktionsbaustein SFB: Systemfunktionsbaustein</p>  <p>The diagram shows a call to a function block (FB). It starts with an arrow pointing to a box containing 'FB-BEZEICHNUNG' and 'SFB-BEZEICHNUNG'. Below this box is 'BEZEICHNER'. These three elements are connected to a dot operator (·). This dot operator is connected to a box containing 'DB-BEZEICHNUNG'. Above this box is 'Globaler Instanzname'. This box is connected to an opening parenthesis '('. This parenthesis is connected to a box containing 'FB-Parameter'. This box is connected to a closing parenthesis ')'. An arrow points out from the closing parenthesis.</p> <p>Lokaler Instanzname</p>
<p>Funktionsaufruf</p>	<p>Funktionsaufruf</p>  <p>The diagram shows a call to a function (FC). It starts with an arrow pointing to a box containing 'FC-BEZEICHNUNG' and 'SFC-BEZEICHNUNG'. Below this box is 'BEZEICHNER'. These three elements are connected to an opening parenthesis '('. This parenthesis is connected to a box containing 'FC-Parameter'. This box is connected to a closing parenthesis ')'. An arrow points out from the closing parenthesis.</p> <p>Standardfunktionsname oder symbolischer Name</p> <p>FC: Funktion SFC: Systemfunktion im Compiler realisierte Standardfunktion</p>
<p>FB-Parameter</p>	<p>FB-Parameter</p>  <p>The diagram shows the structure of an FB parameter. It starts with an arrow pointing to a box containing 'Eingangs-Zuweisung' and 'Durchgangs-Zuweisung'. Below this box is a comma operator (,). This comma operator is connected to another arrow pointing to the right.</p>

Regel	Syntaxdiagramm
FC-Parameter	<p>FC-Parameter</p>
Eingangszuweisung	<p>Eingangszuweisung</p>
Durchgangszuweisung	<p>Durchgangszuweisung</p>
Ausgangszuweisung	<p>Ausgangszuweisung</p>

15.3.8 Kontrollanweisungen

Regel	Syntaxdiagramm
<p>IF-Anweisung</p> <p>Vergessen Sie nicht, das Schlüsselwort END_IF mit Semikolon abzuschließen !</p>	<p>IF-Anweisung</p>
<p>Case-Anweisung</p> <p>Vergessen Sie nicht, das Schlüsselwort END_CASE mit Semikolon abzuschließen !</p>	<p>CASE-Anweisung</p> <p>Auswahlausdruck (Integer)</p>
<p>Wertliste</p>	<p>Wertliste</p> <p>Integerzahl</p>

Regel	Syntaxdiagramm
Wert	
Wiederholungsanweisungen und Sprunganweisungen	
<p>FOR-Anweisung</p> <p>Vergessen Sie nicht, das Schlüsselwort END_FOR mit Semikolon abzuschließen !</p>	<p>FOR-Anweisung</p>
Anfangszuweisung	<p>Anfangszuweisung</p>

Regel	Syntaxdiagramm
<p>WHILE-Anweisung</p> <p>Vergessen Sie nicht, das Schlüsselwort END_WHILE mit Semikolon abzuschließen !</p>	<p>WHILE-Anweisung</p> 
<p>REPEAT-Anweisung</p> <p>Vergessen Sie nicht, das Schlüsselwort END_REPEAT mit Semikolon abzuschließen !</p>	<p>REPEAT-Anweisung</p> 
<p>CONTINUE-Anweisung</p>	<p>CONTINUE-Anweisung</p> 
<p>RETURN-Anweisung</p>	<p>RETURN-Anweisung</p> 
<p>EXIT-Anweisung</p>	<p>EXIT-Anweisung</p> 
<p>Programmsprung</p>	<p>GOTO-Anweisung</p> 

16 Tipps und Tricks

Division zweier Integer-Werte mit Ergebnis im REAL-Format

Sie programmieren in S7-SCL folgende Anweisung:

```
Fraction:=Dividend/Divisor;
```

wobei `Fraction` ein Real-Wert ist, während `Dividend` und `Divisor` Integer-Werte sind.

Beachten Sie, dass der S7-SCL-Compiler bei solchen Operationen eine implizite Datentyp-Konvertierung durchführt und somit die obige Anweisung folgendermaßen übersetzt:

```
Fraction:=INT_TO_REAL(Dividend/Divisor);
```

Daraus folgt, dass die Division immer einen gerundeten Wert als Ergebnis liefert, z.B. $1/3 = 0$ oder $3/2 = 1$.

Laufzeitoptimaler Code bei Zugriff auf Strukturen in Datenbausteinen

Müssen Sie mehr als einmal auf eine Struktur in einem Datenbaustein zugreifen, so empfiehlt sich folgende Vorgehensweise:

1. Legen Sie eine lokale Variable mit dem Typ der Struktur an.
2. Weisen Sie der Variablen einmal die Struktur aus dem Datenbaustein zu.
3. Anschließend können Sie die Variable im Code mehrmals verwenden, ohne erneut auf den DB zugreifen zu müssen.

Beispiel:

```
DB100.feld[i].value :=  
DB100.feld[i].wert1 * DB100.feld[i].wert2 /  
DB100.feld[i].wert3 ;
```

Dieses Beispiel benötigt weniger Speicher und weniger Laufzeit, wenn Sie es folgendermaßen programmieren:

```
VAR_TEMP  
  tmp : STRUCT  
      value : REAL;  
      wert1 : REAL;  
      wert2 : REAL;  
      wert3 : REAL;  
  END_STRUCT;  
END_VAR  
tmp := DB100.feld[i];  
DB100.feld[i].value := tmp.wert1 * tmp.wert2 / tmp.wert3;
```

Hinweis

Mit VAR_TEMP legen Sie Variablen im Stack der CPU ab. Bei kleinen CPUs kann das zur Stacküberschreitung führen. Gehen Sie daher mit temporären Variablen sparsam um!

Probleme beim Allokieren des L-Stacks bei kleinen CPUs

Probleme beim Allokieren des L-Stacks sind auf die geringere Stackgröße der kleinen CPU's zurückzuführen. In den meisten Fällen kann dies leicht umgangen werden, wenn Sie folgende Hinweise beachten:

- Gehen Sie mit temporären Variablen (Abschnitt VAR_TEMP bzw. VAR) sparsam um.
- Deklarieren Sie keine Variablen von einem höheren Datentyp, und reduzieren Sie die Anzahl der Variablen eines elementaren Datentyps auf ein absolutes Minimum.
- Benutzen Sie statische Variablen:
 - Wenn Sie einen FB programmieren, können Sie den Abschnitt VAR anstelle von VAR_TEMP benutzen.
 - Wenn Sie einen OB oder FC programmieren, müssen Sie auf einen globalen Datenbaustein oder auf Merker ausweichen.
- Vermeiden Sie umfangreiche Ausdrücke. Bei der Bearbeitung umfangreicher Ausdrücke speichert der Compiler Zwischenergebnisse auf den Stack. Je nach Typ und Anzahl der Zwischenergebnisse kann die verfügbare Stackgröße überschritten werden.
Abhilfe:
Zerlegen Sie den Ausdruck in mehrere kleinere Ausdrücke, und weisen Sie die Zwischenergebnisse expliziten Variablen zu.

Ausgabe von REAL-Zahlen beim Beobachten

Die Testfunktion "Beobachten" kann bei der Ausgabe nicht darstellbarer REAL-Zahlen folgende Muster liefern:

Wert	Darstellung
+ infinity	1.#INRandom-digits
- infinity	-1.#INRandom-digits
Indefinite	digit.#INDrandom-digits
NaN digit.	#NANrandom-digits

Anzeigen von S7-SCL-Programmen in AWL-Darstellung

Sie können mit dem AWL/KOP/FUP-Editor einen S7-SCL-Baustein öffnen und somit die generierten MC7-Befehle anzeigen. Führen Sie jedoch keine Änderungen in AWL durch, da:

- die angezeigten MC7-Befehle nicht in jedem Fall einen gültigen AWL-Baustein darstellen müssen.
- eine fehlerfreie Übersetzung mit dem AWL-Compiler in der Regel Änderungen voraussetzt, die tiefe Kenntnisse sowohl in AWL als auch in S7-SCL erfordern.
- der mit AWL übersetzte Baustein dann die Sprachkennung AWL und nicht mehr S7-SCL trägt.
- die S7-SCL-Quelle und der MC7-Code nicht mehr konsistent sind.

Behandlung der Zeitstempel von Quelle, Schnittstelle und Code

Zeitstempel der Quelle

Eine S7-SCL-Quelle hat immer den Zeitstempel der letzten Änderung.

Zeitstempel des Bausteincodes

Bausteine (FB, FC und OB) erhalten immer den Zeitstempel des Übersetzungszeitpunktes.

Zeitstempel der Bausteinschnittstelle

Eine Änderung des Zeitstempels einer Schnittstelle ergibt sich nur, wenn die Struktur der Schnittstelle geändert wird, d.h.

- Der Zeitstempel einer Schnittstelle bleibt erhalten wenn Änderungen im Anweisungsteil, in den Attributen, im Kommentar, im Abschnitt VAR_TEMP (bei FC auch VAR) oder in der Schreibweise der Namen von Parametern bzw. Variablen vorgenommen werden. Dies gilt auch für unterlagerte Schnittstellen.
- Der Zeitstempel einer Schnittstelle wird aktualisiert, wenn sich der Datentyp oder eine ggf. vorhandene Initialisierung eines Parameters bzw. einer Variablen ändert, oder aber wenn Parameter bzw. Variablen entfernt oder hinzugefügt werden und wenn sich bei Multi-Instanzen der Name des FB ändert. Dies gilt auch für unterlagerte Schnittstellen.

Rückgabewert von STEP 7 Standard- und Systemfunktionen

Viele STEP7 Standard- und Systemfunktionen haben einen Funktionswert vom Typ INT, der den Fehlercode enthält. Im Referenzhandbuch für diese Funktionen sind die möglichen Fehlercodes als WORD-Konstanten in der Art "W#16#8093" angegeben.

S7-SCL ist eine streng auf Typengleichheit prüfende Sprache, daher können INT und WORD nicht gemischt werden. So bringt z.B. folgende Abfrage nicht das erwünschte Ergebnis.

```
IF SFCxx(..) = 16#8093 THEN ...
```

Sie können dem S7-SCL-Compiler aber folgendermaßen mitteilen, dass eine WORD-Konstante als INT anzusehen ist.

- Durch Typisierung der Konstanten. In diesem Falle lautet die obige Abfrage wie folgt:
IF SFCxx(..) = INT#16#8093 THEN ...
- Durch die Konvertierungsfunktion WORD_TO_INT() . Die obige Abfrage müssen Sie dann wie folgt formulieren:
IF SFCxx(..) = WORD_TO_INT(16#8093) THEN ...

Umverdrahten von Bausteinen

Sie können die Bausteinaufrufe in den S7-SCL-Bausteinen nicht mit der SIMATIC Manager Funktion **Extras > Umverdrahten** umverdrahten. Sie müssen in der S7-SCL-Quelle die Aufrufe der betroffenen Bausteine manuell bearbeiten.

Empfehlungen:

- Definieren Sie für die Bausteine symbolische Namen in der Symboltabelle und rufen Sie die Bausteine symbolisch auf.
- Definieren Sie für absolute Adressen (E, M, A etc.) symbolische Namen in der Symboltabelle und verwenden Sie die symbolischen Namen in Ihrem Programm.

Wenn Sie anschließend einen Baustein umverdrahten möchten, so müssen Sie nur die Zuordnung in der Symboltabelle ändern und keine Änderungen in der S7-SCL-Quelle vornehmen.

Zuweisen von Strukturen mit ungerader Byte-Länge

Die Länge einer Struktur wird immer auf Wortgrenzen aufgefüllt. Um eine Struktur über eine ungerade Byte-Anzahl zulegen, bietet S7-SCL das AT Konstrukt:

Beispiel:

```
VAR
theStruct : STRUCT
                twoBytes : ARRAY [0..1] OF BYTE;
                oneInt   : INT
                oneByte  : BYTE;
            END_STRUCT;
fiveBytes AT theStruct : ARRAY[0..4] OF BYTE;
END_VAR
```

Dort, wo 5 BYTES verlangt werden, benutzen Sie den Bezeichner fiveBytes. Über den Bezeichner theStruct kann dann strukturiert zugegriffen werden.

Grenzwerte für FOR-Anweisungen

Um "sichere" FOR-Anweisungen, die nicht endlos laufen, zu programmieren, beachten Sie folgende Regel und Grenzwerte:

Regel

FOR ii := anfang TO ende BY schritt DO

Wenn...	...dann	Bemerkung
anfang < ende	ende < (PMAX - schritt)	Laufvariable ii läuft in positiver Richtung.
anfang > ende AND schritt < 0	ende > (NMAX - schritt)	Laufvariable ii läuft in negativer Richtung.

Grenzwerte

Für die beiden möglichen Datentypen gelten unterschiedliche Grenzwerte:

Datentyp	PMAX	NMAX
ii vom Typ INT	32_767	-32_768
ii vom Typ DINT	2_147_483_647	-2_147_483_648

Glossar

Adressierung

Zuweisung einer Adresse im Anwenderprogramm. Adressen können bestimmten Operanden oder Operandenbereichen zugewiesen werden (Beispiele: Eingang E 12.1, Merkerwort MW25)

Adressierung, absolut

Bei der absoluten Adressierung wird die Adresse des zu bearbeitenden Operanden angegeben. Beispiel: Die Adresse A 4.0 bezeichnet das Bit 0 im Byte 4 des Prozessabbilds der Ausgänge.

Adressierung, symbolisch

Bei der symbolischen Adressierung wird der zu bearbeitende Operand symbolisch angegeben. Die Zuordnung zwischen Symbolen und Adressen erfolgt über die Symboltabelle oder über eine Symboldatei.

Aktualparameter

Aktualparameter ersetzen beim Aufruf eines Funktionsbausteins (FB) oder einer Funktion (FC) die Formalparameter.

Beispiel: Der Formalparameter "Start" wird ersetzt durch den Aktualparameter "E 3.6"

Anfangswert

Wert, der einer Variablen beim Systemanlauf zugewiesen wird.

Anweisung

Eine Anweisung ist die kleinste selbstständige Einheit eines in einer textuellen Sprache erstellten Anwenderprogramms. Sie stellt eine Arbeitsvorschrift für den Prozessor dar.

Anwenderprogramm

Das Anwenderprogramm enthält alle Anweisungen und Deklarationen für die Signalverarbeitung, durch die eine Anlage oder ein Prozess gesteuert werden können. Es ist einer programmierbaren Baugruppe (z. B. CPU, FM) zugeordnet und kann in kleinere Einheiten (Bausteine) strukturiert werden.

Attribut

Ein Attribut ist eine Eigenschaft, die z.B. an eine Bausteinbezeichnung oder einen Variablennamen angehängt werden kann. Bei S7-SCL gibt es z. B. Attribute für folgende Angaben: Bausteinüberschrift, Ausgabestand, Bausteinschutz, Autor, Bausteinname, Bausteinfamilie.

Aufrufhierarchie

Alle Bausteine müssen erst aufgerufen werden, ehe sie bearbeitet werden können. Die Reihenfolge und Schachtelung dieser Aufrufe wird Aufrufhierarchie genannt.

Aufrufschnittstelle

Die Aufrufschnittstelle wird definiert durch die Eingangs-, Ausgangs- und Durchgangparameter (Formalparameter) eines Bausteins im Anwenderprogramm. Bei Aufruf des Bausteins werden diese Parameter durch die Aktualparameter ersetzt.

Ausdruck

Ein Ausdruck dient in S7-SCL zur Verarbeitung von Daten. Es wird unterschieden zwischen arithmetischen, logischen Ausdrücken und Vergleichsausdrücken.

Ausgangsparameter (A-Parameter)

Mit den Ausgangsparametern eines Bausteins im Anwenderprogramm werden Ergebnisse an den aufrufenden Baustein übergeben.

Baustein

Bausteine sind durch ihre Funktion, ihre Struktur oder ihren Verwendungszweck abgegrenzte Teile des Anwenderprogramms. Es gibt bei STEP 7 Codebausteine (FB, FC, OB, SFC, SFB) Datenbausteine (DB, SDB) und anwenderdefinierte Datentypen (UDT).

Bausteinkommentar

Zusatzinformationen zu einem Baustein (z.B. Erläuterungen zum automatisierten Prozess), die nicht in den Arbeitsspeicher von SIMATIC S7-Automatisierungssystemen geladen werden.

Bausteinschutz

Als Bausteinschutz bezeichnet man die Möglichkeit, einzelne Bausteine gegen Rückübersetzung zu schützen, wenn die Übersetzung der Bausteinquelle mit dem Schlüsselwort "KNOW_HOW_PROTECTED" vorgenommen wurde.

Bausteinart

Die Bausteinarchitektur von STEP 7 kennt folgende Bausteinarten: Organisationsbausteine, Funktionsbausteine, Funktionen, Datenbausteine sowie Systemfunktionsbausteine, Systemfunktionen, Systemdatenbausteine und anwenderdefinierte Datentypen.

Bausteinaufruf

Starten eines Bausteins im STEP 7-Anwenderprogramm: Organisationsbausteine werden grundsätzlich vom Betriebssystem aufgerufen, alle anderen Bausteine werden vom STEP 7-Anwenderprogramm aufgerufen.

Bausteinklasse

Bausteine unterteilt man, ihrem Inhalt entsprechend, in zwei Klassen: Codebausteine und Datenbausteine.

BCD

Binär codierte Dezimalzahl. Bei STEP 7 erfolgt die CPU-interne Angabe von Zeiten und Zählern nur im BCD-Format.

Beobachten

Beim Beobachten eines Programms können Sie ein Programm in seinem Ablauf in der CPU kontrollieren. Dabei werden z.B. Namen und aktuelle Werte von Variablen und Parametern in chronologischer Abfolge angezeigt und zyklisch aktualisiert.

Bezeichner

Kombination von Buchstaben, Zahlen und Unterstrichen, die ein Sprachelement bezeichnet.

CASE-Anweisung

Diese Anweisung ist eine Verzweigungsanweisung. Sie dient der Auswahl aus 1-n Programmteilen, abhängig vom Wert eines Auswahlausdrucks.

Codebaustein

Ein Codebaustein ist bei SIMATIC S7 ein Baustein, der einen Teil des STEP 7-Anwenderprogramms enthält. Im Gegensatz dazu enthält ein Datenbaustein nur Daten. Es gibt folgende Codebausteine: Organisationsbausteine (OB), Funktionsbausteine (FB), Funktionen (FC), Systemfunktionsbausteine (SFB) und Systemfunktionen (SFC).

CONTINUE-Anweisung

Eine CONTINUE-Anweisung dient bei S7-SCL zum Abbruch der Ausführung des momentanen Schleifendurchlaufes einer Wiederholungsanweisung (FOR, WHILE oder REPEAT).

Datenbaustein (DB)

Datenbausteine sind Bausteine, die Daten und Parameter enthalten, mit denen das Anwenderprogramm arbeitet. Sie enthalten im Gegensatz zu allen anderen Bausteinen keine Anweisungen.

Daten, statisch

Statische Daten sind Lokaldaten eines Funktionsbausteins, die im Instanz-Datenbaustein gespeichert werden und deshalb bis zur nächsten Bearbeitung des Funktionsbausteins erhalten bleiben.

Daten, temporär

Temporäre Daten sind Lokaldaten eines Bausteins, die während der Bearbeitung eines Bausteins im L-Stack abgelegt werden und nach der Bearbeitung nicht mehr verfügbar sind.

Datentyp

Datentypen bestimmen

- die Art und Bedeutung von Datenelementen
- die zulässigen Speicher- und Wertebereiche von Datenelementen
- die zulässige Menge der Operationen, die mit einem Operanden eines Datentyps ausgeführt werden können.
- die Schreibweise von Datenelementen

Datentyp, anwenderdefiniert

Anwenderdefinierte Datentypen (UDT) werden vom Anwender mit der Datentypdeklaration geschaffen. Sie haben einen eigenen Namen und sind mehrfach verwendbar. So kann ein anwenderdefinierter Datentyp zur Erzeugung mehrerer Datenbausteine mit der gleichen Struktur (z.B. Regler) genutzt werden.

Datentyp, elementar

Elementare Datentypen sind vordefinierte Datentypen gemäß IEC 1131-3. Beispiele: Datentyp "BOOL" definiert eine binäre Variable ("Bit"); Datentyp "INT" definiert eine 16-Bit-Festpunkt-Variable.

Datentyp, zusammengesetzt

Komplexe Datentypen, die sich aus Datenelementen elementaren Datentyps zusammensetzen. Man unterscheidet zwischen Strukturen und Feldern. Auch die Datentypen STRING und DATE_AND_TIME zählen hierzu.

Datentyp-Konvertierung

Eine Datentypkonvertierung ist notwendig, wenn zwei Operanden ungleichen Datentyps miteinander verknüpft werden sollen.

Deklaration

Mechanismus zur Definition eines Sprachelements. Eine Deklaration umfasst die Verbindung eines Bezeichners mit dem Sprachelement, sowie die Zuordnung von Attributen und Datentypen.

Deklarationsabschnitt

Die Variablendeklaration eines Bausteins teilt sich in verschiedene Deklarationsabschnitte zur Deklaration der unterschiedlichen Bausteinparameter. Der Deklarationsabschnitt IN enthält z.B. die Deklaration der Eingangsparameter, der Deklarationsabschnitt OUT die Deklaration der Ausgangsparameter.

Durchgangparameter

Durchgangparameter gibt es bei Funktionen und Funktionsbausteinen. Mit Durchgangsparemtern werden Daten an den aufgerufenen Baustein übergeben, dort verarbeitet und die Ergebnisse vom aufgerufenen Baustein wieder in der gleichen Variablen abgelegt.

Eingangsparameter

Eingangsparameter gibt es nur bei Funktionen und Funktionsbausteinen. Mit Hilfe der Eingangsparameter werden Daten zur Verarbeitung an den aufgerufenen Baustein übergeben.

Einzelstschritt

Der Einzelschritt ist ein Testschritt innerhalb der Einzelschrittfunktion des Debuggers von S7-SCL. Mit der Einzelschrittfunktion können Sie das Programm Anweisung für Anweisung ausführen und im Ergebnisfenster beobachten.

Enable (EN)

Bei STEP 7 hat jeder Funktionsbaustein und jede Funktion den implizit definierten Eingangsparameter "Enable" (EN), der beim Aufruf des Bausteins gesetzt werden kann. Wenn EN gleich TRUE ist, wird der angerufene Baustein ausgeführt, andernfalls nicht.

Enable out (ENO)

Bei STEP 7 hat jeder Baustein einen Ausgang "Enable Output" (ENO). Am Ende der Ausführung eines Bausteins wird der aktuelle Wert des OK-Flags in ENO abgelegt. Unmittelbar nach dem Aufruf eines Bausteins können Sie anhand des Wertes von ENO überprüfen, ob alle Operationen im Baustein richtig abgelaufen sind oder ob es zu Fehlern gekommen ist.

EXIT-Anweisung

Sprachkonstruktion innerhalb eines Programms, die zum Abbruch einer Schleife an beliebiger Stelle und unabhängig von Bedingungen dient.

Feld

Ein Feld (ARRAY) ist ein zusammengesetzter Datentyp bestehend aus Datenelementen gleichen Typs. Diese Datenelemente können wiederum elementar oder zusammengesetzt sein.

FOR-Anweisung

Sprachkonstruktion innerhalb eines Programms. Die FOR-Anweisung führt eine Anweisungsfolge in einer Schleife aus, wobei einer Variablen (Laufvariablen) fortlaufende Werte zugewiesen werden.

Formalparameter

Ein Formalparameter ist ein Platzhalter für den "tatsächlichen" Parameter (Aktualparameter) bei parametrierbaren Codebausteinen. Bei FB und FC werden die Formalparameter vom Anwender deklariert, bei SFB und SFC sind sie bereits vorhanden. Beim Aufruf des Bausteins wird dem Formalparameter ein Aktualparameter zugeordnet, sodass der angerufene Baustein mit diesem aktuellen Wert arbeitet. Die Formalparameter zählen zu den Lokaldaten des Bausteins und unterteilen sich nach Eingangs-, Ausgangs-, und Durchgangsparemtern.

Funktion (FC)

Eine Funktion (FC) ist gemäß IEC 1131-3 ein Codebaustein ohne statische Daten. Eine Funktion bietet die Möglichkeit der Übergabe von Parametern im Anwenderprogramm. Dadurch eignen sich Funktionen zur Parametrierung von häufig wiederkehrenden komplexen Funktionen, z.B. Berechnungen.

Funktionsbaustein (FB)

Ein Funktionsbaustein (FB) ist gemäß IEC 1131-3 ein Codebaustein mit statischen Daten (Daten, statisch). Da ein FB über ein Gedächtnis (Instanz-Datenbaustein) verfügt, kann auf seine Parameter (z. B. Ausgänge) zu jeder Zeit an jeder beliebigen Stelle im Anwenderprogramm zugegriffen werden.

Globale Daten

Globale Daten sind Daten, die von jedem Codebaustein (FC, FB, OB) aus ansprechbar sind. Im Einzelnen sind das Merker M, Eingänge E, Ausgänge A, Zeiten, Zähler und Elemente von Datenbausteinen DB. Auf globale Daten kann entweder absolut oder symbolisch zugegriffen werden.

GOTO-Anweisung

Sprachkonstruktion innerhalb eines Programms. Eine GOTO-Anweisung bewirkt den sofortigen Sprung zu einer angegebenen Sprungmarke und damit zu einer anderen Anweisung innerhalb desselben Bausteins.

HALT

Der Betriebszustand HALT wird aus dem Betriebszustand RUN durch Anforderung vom Programmiergerät erreicht. In diesem Betriebszustand sind spezielle Testfunktionen möglich.

Haltepunkt

Mit dieser Funktion kann die Zentralbaugruppe (CPU) an definierten Programmstellen in den Betriebszustand HALT versetzt werden. Beim Erreichen eines Haltepunktes können die Testfunktionen wie z.B. schrittweise Befehlsbearbeitung oder Variablen beobachten/steuern durchgeführt werden.

Instanz

Mit "Instanz" wird der Aufruf eines Funktionsbausteins bezeichnet. Dabei ist ihm ein Instanz-Datenbaustein oder eine Lokale Instanz zugeordnet. Wird ein Funktionsbaustein im STEP 7-Anwenderprogramm n-mal mit jeweils unterschiedlichen Parametern und Instanz-Datenbausteinnamen aufgerufen, so existieren n Instanzen.

Instanz-Datenbaustein (Instanz-DB)

Ein Instanz-Datenbaustein speichert die Formalparameter und statischen Lokaldaten von Funktionsbausteinen. Ein Instanz-Datenbaustein kann einem FB-Aufruf oder einer Aufrufhierarchie von Funktionsbausteinen zugeordnet sein.

Integer (INT)

Integer (INT) ist einer der elementaren Datentypen. Die Darstellung erfolgt als 16-bit Ganzzahl.

Kommentar

Sprachkonstruktion zur Einbindung von erläuterndem Text in ein Programm, der keinen Einfluss auf die Programmausführung hat.

Konstante

Platzhalter für konstante Werte bei Codebausteinen. Konstanten werden verwendet, um die Lesbarkeit eines Programms zu erhöhen. Beispiel: Anstatt einen Wert (z.B. 10) direkt anzugeben, wird der Platzhalter "Max_Schleifendurchläufe" angegeben. Bei dessen Aufruf wird dafür der Wert der Konstanten (z.B. 10) ersetzt.

Konstante (symbolisch)

Konstanten mit symbolischem Namen sind Platzhalter für konstante Werte bei Codebausteinen. Symbolische Konstanten werden verwendet, um die Lesbarkeit eines Programms zu erhöhen.

Laden in Zielsystem

Transferieren von ladbaren Objekten (z.B. Codebausteinen) vom Programmiergerät in den Ladespeicher einer CPU.

Lesezeichen

Lesezeichen sind temporäre Textmarken, die eine beliebige Stelle innerhalb einer Quelle kennzeichnen. Sie erleichtern die Navigation in der Quelle.

Lexikalische Regel

Die untere Regelstufe der formalen S7-SCL-Sprachbeschreibung besteht aus den lexikalischen Regeln. Bei ihrer Anwendung besteht keine Formattfreiheit, d.h. die Ergänzung von Leerzeichen und Steuerzeichen, ist nicht erlaubt.

Literal

Formale Schreibweise, die den Wert und den Typ einer Konstanten bestimmt.

Lokaldaten

Lokaldaten sind die einem Codebaustein zugeordneten Daten, die in seinem Deklarationsteil bzw. in seiner Variablendeklaration deklariert werden. Sie umfassen (bausteinabhängig): Formalparameter, statische Daten, temporäre Daten.

Merker (M)

Speicherbereich im Systemspeicher einer SIMATIC S7-CPU. Auf ihn kann schreibend und lesend zugegriffen werden (bit-, byte-, wort- und doppelwortweise). Der Merkerbereich kann vom Anwender zum Speichern von Zwischenergebnissen verwendet werden.

Mnemonik

Die Mnemonik ist eine abgekürzte Darstellung der Operanden und der Programmieroperationen im Programm. STEP 7 unterstützt die englische Darstellung (in der z.B. "I" für Eingang steht) und die deutsche Darstellung (hier wird z. B. "E" für Eingang verwendet).

Multiinstanz

Bei der Verwendung von Multiinstanzen enthält der Instanz-Datenbaustein die Daten für mehrere Funktionsbausteine einer Aufrufhierarchie.

Nonterminal

Ein Nonterminal ist ein zusammengesetztes Element in einer Syntaxbeschreibung, das durch eine weitere lexikalische oder syntaktische Regel beschrieben wird.

Nutzdaten

Nutzdaten werden zwischen einer Zentralbaugruppe und Signalbaugruppe, Funktionsbaugruppe und Kommunikationsbaugruppen über das Prozessabbild oder über Direktzugriffe ausgetauscht. Nutzdaten können sein: Digitale und analoge Ein-/Ausgangssignale von Signalbaugruppen, Steuer- und Statusinformationen von Funktionsbaugruppen.

Offline

Offline bezeichnet den Betriebszustand, bei dem das Programmiergerät keine Verbindung mit dem Automatisierungssystem hat (physikalisch, logisch).

OK-Flag

Das OK-Flag dient dazu, die korrekte oder inkorrekte Ausführung einer Bausteinbefehlsfolge zu vermerken. Sie ist global vom Typ BOOL.

Online

Online bezeichnet den Betriebszustand, bei dem das Programmiergerät mit dem Automatisierungssystem verbunden ist (physikalisch, logisch).

Online-Hilfe

STEP 7 bietet Ihnen die Möglichkeit, sich während des Arbeitens mit der Programmiersoftware kontextabhängige Hilfetexte am Bildschirm anzeigen zu lassen.

Operand

Ein Operand ist ein Teil einer Anweisung und sagt aus, womit der Prozessor etwas tun soll. Er kann sowohl absolut als auch symbolisch adressiert werden.

Operandenkennzeichen

Ein Operandenkennzeichen ist der Teil des Operanden einer Operation, in dem Informationen enthalten sind, wie z. B. der Speicherbereich, in dem die Operation einen Wert (Datenobjekt) findet, mit dem sie eine Verknüpfung ausführt oder die Größe eines Werts (Datenobjekt), mit dem sie eine Verknüpfung ausführt. In der Anweisung "Wert := EB10" ist "EB" das Operandenkennzeichen ("E" steht für den Eingangsbereich des Speichers, "B" steht für ein Byte in diesem Bereich).

Operation

Eine Operation ist Teil einer Anweisung und sagt aus, was der Prozessor tun soll.

Organisationsbaustein (OB)

Organisationsbausteine bilden die Schnittstelle zwischen dem Betriebssystem der CPU und dem Anwenderprogramm. In den Organisationsbausteinen wird die Reihenfolge der Bearbeitung des Anwenderprogramms festgelegt.

Parametertyp

Ein Parametertyp ist ein spezieller Datentyp für Zeiten, Zähler und Bausteine. Er kann bei Eingangsparametern von Funktionsbausteinen und Funktionen, bei Durchgangsparametern nur von Funktionsbausteinen verwendet werden, um Zeiten, Zähler und Bausteine an den aufgerufenen Baustein zu übergeben.

Programmierung, strukturiert

Zur Lösung komplexer Automatisierungsaufgaben wird das Anwenderprogramm in einzelne abgeschlossene Programmteile (Bausteine) unterteilt. Die Gliederung des Anwenderprogramms erfolgt funktional oder entsprechend der technologischen Anlagenstruktur.

Programmierung, symbolisch

Die Programmiersprache S7-SCL ermöglicht das Verwenden von symbolischen Zeichenfolgen anstelle von Operanden: z. B. der Operand A1.1 kann ersetzt werden durch "Ventil_17". Die Symboltabelle stellt die Verbindung zwischen Operand und der zugeordneten symbolischen Zeichenfolge her.

Projekt

Ein Ordner für alle Objekte einer Automatisierungslösung unabhängig von der Anzahl der Stationen, Baugruppen und deren Vernetzung.

Prozessabbild

Die Signalzustände der digitalen Ein- und Ausgabebaugruppen werden in der CPU in einem Prozessabbild hinterlegt. Man unterscheidet das Prozessabbild der Eingänge (PAE) und das der Ausgänge (PAA).

Prozessabbild der Ausgänge (PAA)

Das Prozessabbild der Ausgänge wird am Ende des Anwenderprogramms vom Betriebssystem auf die Ausgangsbaugruppen übertragen.

Prozessabbild der Eingänge (PAE)

Das Prozessabbild der Eingänge wird vor der Bearbeitung des Anwenderprogramms vom Betriebssystem von den Eingangsbaugruppen gelesen.

Quelle

Teil eines Programms, der mit einem grafischen oder textuellen Editor erstellt wird und aus dem durch Übersetzung ein ablauffähiges Anwenderprogramm entsteht.

Realzahl

Eine Realzahl ist eine positive oder negative Zahl, die einen Dezimalwert darstellt z.B. 0.339 oder -11.1.

REPEAT-Anweisung

Sprachkonstruktion innerhalb eines Programms, die zur Wiederholung einer Folge von Anweisungen bis zu einer Abbruchbedingung dient.

Rückgabewert (RET_VAL)

Im Gegensatz zu Funktionsbausteinen liefern Funktionen als Ergebnis einen Rückgabewert.

RETURN-Anweisung

Sprachkonstruktion innerhalb eines Programms, die das Verlassen des aktuellen Bausteins bewirkt.

RUN

Im Betriebszustand RUN wird das Anwenderprogramm bearbeitet, das Prozessabbild wird zyklisch aktualisiert. Alle digitalen Ausgänge sind freigegeben.

RUN-P

Der Betriebszustand RUN-P entspricht dem Betriebszustand RUN, mit dem Unterschied, dass bei dem Betriebszustand RUN-P sämtliche Programmiergerätefunktionen ohne Einschränkungen erlaubt sind.

S7-Anwenderprogramm

Ein Ordner für Bausteine, die auf eine programmierbare S7-Baugruppe (z. B. CPU, FM) geladen werden und dort lauffähig sind, um eine Anlage oder einen Prozess zu steuern.

S7-SCL

PASCAL-ähnliche Hochsprache nach der Norm DIN EN-61131-3 (int. IEC 1131-3) zur Programmierung von komplexen Aufgaben in einer SPS, z. B. Algorithmen, Datenverarbeitungsaufgaben. Abkürzung für "Structured Control Language".

SCL-Compiler

Der S7-SCL-Compiler ist ein Batch-Compiler, mit dem das zuvor editierte Programm (SCL-Quelle) in den MC7-Maschinencode übersetzt wird. Die dadurch erzeugten Bausteine werden im S7-Programm im Ordner "Bausteine" abgelegt.

SCL-Debugger

Der S7-SCL-Debugger ist ein Hochsprachendebugger, mit dem logische Programmierfehler in mit S7-SCL erstellten Anwenderprogrammen gefunden werden können.

SCL-Editor

Der S7-SCL-Editor ist ein auf S7-SCL zugeschnittener Editor, mit dem die S7-SCL-Quelle erstellt werden kann.

SCL-Quelle

Die S7-SCL-Quelle ist die Datei, in der das Programm in S7-SCL erstellt wird. Die Quelldatei wird anschließend mit dem S7-SCL-Compiler übersetzt.

Schlüsselwort

Lexikalische Einheit, die ein Sprachelement charakterisiert, z.B. "IF".

Schlüsselwörter werden bei S7-SCL verwendet, um den Beginn eines Bausteins zu kennzeichnen, um Sektionen im Deklarations- bzw. Vereinbarungsteil zu markieren und um Anweisungen zu kennzeichnen. Außerdem werden sie für Kommentare und Attribute benutzt.

Semantik

Beziehung zwischen den symbolischen Elementen einer Programmiersprache und ihren Bedeutungen, ihrer Auslegung und Anwendung.

Sicht

Um mit einem weiteren Datentyp auf eine deklarierte Variable zuzugreifen, können Sie Sichten auf die Variable oder auf Bereiche innerhalb der Variablen definieren. Eine Sicht kann wie jede andere Variable im Baustein verwendet werden. Sie erbt alle Eigenschaften der Variablen, auf die sie zeigt, nur der Datentyp ist neu.

Speicherbereich

Eine Zentralbaugruppe hat bei SIMATIC S7 drei Speicherbereiche: Den Ladebereich, den Arbeitsbereich und den Systembereich.

Statuswort

Das Statuswort ist Bestandteil der Register der Zentralbaugruppe. Im Statuswort befinden sich Statusinformationen und Fehlerinformationen, die im Zusammenhang mit der Bearbeitung von STEP 7-Befehlen auftreten. Die Statusbits können vom Anwender gelesen und beschrieben werden; die Fehlerbits können nur gelesen werden.

Struktur (STRUCT)

Zusammengesetzter Datentyp, der aus beliebigen Datenelementen unterschiedlichen Datentyps besteht. Die Datentypen innerhalb von Strukturen können elementar oder höher sein.

Symbol

Ein Symbol ist ein vom Anwender unter Berücksichtigung bestimmter Syntaxvorschriften definierter Name. Dieser Name kann nach der Festlegung, wofür er stehen soll (z.B. Variable, Datentyp, Baustein) bei der Programmierung und beim Bedienen und Beobachten verwendet werden. Beispiel: Operand: E 5.0, Datentyp: Bool, Symbol: Taster_Notaus.

Symboltabelle

Tabelle zur Zuordnung von Symbolen (=Name) zu Adressen für globale Daten und Bausteine. Beispiele: Notaus (Symbol) - E 1.7 (Adresse) oder Regler (Symbol) - SFB 24 (Baustein).

Syntaktische Regel

Die obere Regelstufe der formalen S7-SCL-Sprachbeschreibung besteht aus den syntaktischen Regeln. Bei ihrer Anwendung besteht Formattfreiheit, d.h. z. B. Leerzeichen und Steuerzeichen dürfen ergänzt werden.

Systemfunktion (SFC)

Eine Systemfunktion (SFC) ist eine im Betriebssystem der CPU integrierte Funktion, die bei Bedarf im STEP 7-Anwenderprogramm aufgerufen werden kann.

Systemfunktionsbaustein (SFB)

Ein Systemfunktionsbaustein (SFB) ist ein im Betriebssystem der CPU integrierter Funktionsbaustein, der bei Bedarf im STEP 7-Anwenderprogramm aufgerufen werden kann.

Systemdatenbaustein (SDB)

System-Datenbausteine sind Datenbereiche in der S7-CPU, die Systemeinstellungen und Baugruppenparameter enthalten. Die Systemdatenbausteine werden mit der STEP 7 Basissoftware erzeugt und geändert.

Systemspeicher (Systembereich)

Der Systemspeicher ist in die S7-CPU integriert und als RAM-Speicher ausgeführt. Im Systemspeicher sind die Operandenbereiche (z. B. Zeiten, Zähler, Merker) sowie vom Betriebssystem intern benötigte Datenbereiche (z. B. Puffer für Kommunikation) abgelegt.

Terminal

Ein Terminal ist ein Grundelement einer lexikalischen oder syntaktischen Regel, das nicht durch eine weitere Regel, sondern verbal erklärt wird. Ein Terminal kann z.B. ein Schlüsselwort oder nur ein einzelnes Zeichen sein.

Übersetzen

Erzeugen eines lauffähigen Anwenderprogramms aus einer Quelle.

Übersetzung, quellorientiert

Bei der quellorientierten Eingabe findet die Übersetzung in ein ablauffähiges Anwenderprogramm erst dann statt, wenn sämtliche Anweisungen eingegeben worden sind. Bei der Übersetzung wird auf eventuelle Eingabefehler geprüft.

UDT

Siehe: Datentyp anwenderdefiniert.

Variable

Eine Variable definiert ein Datum mit variablem Inhalt, das im STEP 7-Anwenderprogramm verwendet werden kann. Eine Variable besteht aus einem Operanden (z. B. M 3.1) und einem Datentyp (z. B. Bool) und kann mit einem Symbol (z. B. BAND_EIN) gekennzeichnet werden. Die Variable wird im Vereinbarungsteil deklariert.

Variablendeklaration

Die Variablendeklaration umfasst die Angabe eines symbolischen Namens, eines Datentyps und evtl. eines Vorbelegungswerts und eines Kommentars.

Variablentabelle

In der Variablentabelle werden die Variablen inkl. der zugehörigen Formatangaben zusammengestellt, die beobachtet und gesteuert werden sollen.

Vereinbarungsteil

Im Vereinbarungsteil werden die Lokaldaten eines Codebausteins deklariert, wenn die Programmerstellung mit einem Texteditor erfolgt.

Zähler

Zähler sind Bestandteile des Systemspeichers der CPU. Der Inhalt dieser Zähler wird asynchron zum Anwenderprogramm vom Betriebssystem aktualisiert. Mit STEP 7-Anweisungen wird die genaue Funktion der Zählerzelle (z. B. Aufwärtszähler) festgelegt und ihre Bearbeitung (Start) angestoßen.

Zeiten

Zeiten sind Bestandteile des Systemspeichers der CPU. Der Inhalt dieser Zeiten wird asynchron zum Anwenderprogramm vom Betriebssystem aktualisiert. Mit STEP 7-Anweisungen wird die genaue Funktion der Zeitzelle (z. B. Einschaltverzögerung) festgelegt und ihre Bearbeitung (Start) angestoßen.

Zuweisung

Mechanismus, um einer Variable einen Wert zu geben.

Zyklusüberwachungszeit

Überschreitet die Bearbeitungszeit des Anwenderprogramms die eingestellte Zyklusüberwachungszeit, so erzeugt das Betriebssystem eine Fehlermeldung und die CPU geht in den STOP-Zustand.

Zykluszeit

Die Zykluszeit ist die Zeit, die die CPU für die einmalige Bearbeitung des Anwenderprogramms benötigt.

Index

- ***
- * 11-8, 11-9
- ** 11-8
- /**
- / 11-8
- +**
- + 11-8, 11-9
- <**
- < 11-13, 11-14
- <= 11-14
- <> 11-12, 11-14
- =**
- = 11-12
- >**
- > 11-14
- >= 11-12
- A**
- Abbruchbedingung 12-22, 12-24
- ABS 14-9
- Absolutadressierung
 - Lexikalische Regeln 15-29
- Absoluter Zugriff
 - auf Datenbausteine 10-8
 - auf Speicherbereiche der CPU 10-3
- Abwärtszähler (S_CD) 13-6
- ACOS 14-10
- Addition 11-2
- Adressen 10-2
- Aktualparameter 8-1
 - Definition 8-1
 - Eingangszuweisung 12-41
- AND 11-10, 11-11
- Anfangswerte 8-4
- Anlegen einer neuen S7-SCL-Quelle 4-4
- Anweisungen 12-1, 12-14, 12-16, 12-18, 12-21 - 12-26
- Anweisungsteil
 - Aufbau 6-12
 - Syntaktische Regeln 15-45
- Anwenderdaten 10-1
 - global 10-1
- Anwenderdefinierte Datentypen (UDT) 6-22, 7-14, 12-3
- Anwenderprogramm 3-4, 6-1
- Anwenderspeicher 4-41
- Anwendungsbereich 1-1
- ANY 7-19, 7-20
- Anzeigen der Bausteine in der CPU 4-42
- Anzeigen der Informationen zur Kommunikation der CPU 4-43
- Anzeigen der Stacks der CPU 4-43
- Anzeigen der Zykluszeit der CPU 4-42
- Anzeigen des Zeitsystems der CPU 4-42
- Anzeigen und Ändern des Betriebszustands der CPU 4-40
- Anzeigen und Einstellen von Datum und Uhrzeit der CPU 4-40
- Anzeigen/Komprimieren des Anwenderspeichers der CPU 4-41
- Arbeitsbereich 4-2
- Arbeitsweise von S7-SCL 1-2
- Arithmetische Ausdrücke 11-8
- ARRAY 7-11, 8-5, 12-6
 - Wertzuweisungen mit Variablen vom Datentyp ARRAY 12-5
- ASIN 14-10
- AT 8-6
- ATAN 14-10
- Attribute 6-5, 6-9, 6-11
- Auf- und Abwärtszähler (S_CUD) 13-7
- Aufbau des Vereinbarungsteils 6-10
- Aufbau einer Funktion (FC) 6-16
- Aufbau einer S7-SCL-Quelle 6-10, 6-13, 6-14, 6-16, 6-19
- Aufbau einer S7-SCL-Quelle 6-12, 6-18
- Aufbau eines Datenbausteins (DB) 6-19
- Aufbau eines Funktionsbausteins (FB) 6-14
- Aufbau eines Organisationsbausteins (OB) 6-18

- Aufruf von Funktionen (FC) 12-38
 - Ausgangs-/Durchgangszuweisung 12-42
 - Ausgangsparameter ENO 12-46
 - Eingangsparameter EN 12-45
 - Eingangszuweisung 12-41
 - Parameterversorgung 12-40
 - Rückgabewert 12-39
 - Syntax 12-38
 - Vorgehensweise 12-38
 - Aufruf von Funktionsbausteinen (FB oder SFB) 12-29
 - Aufruf als globale Instanz 12-29
 - Aufruf als lokale Instanz 12-29
 - Ausgangswerte lesen 12-35
 - Durchgangszuweisung 12-34
 - Eingangszuweisung 12-33
 - Syntax 12-29
 - Versorgung der FB-Parameter 12-31
 - Vorgehensweise 12-29
 - Aufruf von Zählfunktionen 13-1
 - Aufruf von Zeitfunktionen 13-9
 - Aufrufen von Bausteinen 4-16
 - Aufrufumgebung 4-32
 - Aufwärtszähler (S_CU) 13-6
 - Ausdrücke 11-2 - 11-12
 - Ausgangsparameter 8-1
 - Ausgangsparameter ENO 12-46
 - Ausgangswerte lesen 12-35
 - Ausgangszuweisung beim FB-Aufruf 12-35
 - Ausgangszuweisung beim FC-Aufruf 12-42
 - Auslesen der CPU-Daten 4-41
 - Auslesen des Diagnosepuffers der CPU 4-41
 - Ausschneiden von Textobjekten 4-11
 - Auswahl des richtigen Zeitglieds 13-21
 - Auswahlweisung 12-12
 - AUTHORS.EXE 2-6
 - Automation License Manager 2-1
 - Automatisches Einrücken von Zeilen 4-13
 - Autorisierung 2-6
 - Autorisierung installieren 2-4
 - Autorisierungsdiskette 2-4, 2-6
 - Autorisierungsprogramm 2-4
- B**
- Bausteinanfang 6-3
 - Bausteinattribute 6-6, 6-9
 - Definition 6-5
 - Lexikalische Regeln 15-32
 - Systemattribute für Bausteine 6-9
 - Bausteinaufruf 4-16
 - Bausteinbezeichnung 5-7, 5-8, 6-3, 6-4
 - Bausteine 3-4, 3-5, 4-6, 6-1
 - Bausteinende 6-3
 - Bausteinparameter 5-17, 8-13
 - Bausteinschutz 4-7
 - Bausteinstruktur 6-3
 - Bausteinvorlagen 4-16
 - Bearbeiten einer S7-SCL-Quelle 4-9 - 4-17
 - Bedienoberfläche 4-2
 - Bedingungen 12-13
 - Beheben von Fehlern nach dem Übersetzen 4-21
 - Beispiel "Messwerterfassung" 3-1
 - Beispiele 7-20, 12-36, 12-37, 12-44, 13-7, 13-20, 14-7, 14-10, 14-12
 - Beobachten 4-28, 4-29, 4-31, 4-32
 - Betriebszustand 4-40
 - Bezeichner 5-6, 15-17 - 15-19
 - Beispiele 5-6
 - Definition 5-6
 - Formale Sprachbeschreibung 15-15, 15-17
 - Regeln 5-6
 - Bezeichnungen
 - Lexikalische Regeln 15-21
 - BIT 7-3
 - Bitdatentypen 7-3
 - Bit-Konstanten 9-6
 - Bitstring-Standardfunktion 14-11
 - BLOCK_DB_TO_WORD 14-4
 - BLOCK-Datentypen 7-17
 - Blöcke in Syntaxdiagrammen 5-1
 - Blockkommentar 5-15
 - Boolescher Ausdruck 11-12
 - BYTE 7-3
 - BYTE_TO_BOOL 14-3
 - BYTE_TO_CHAR 14-3
- C**
- CASE-Anweisung 12-12, 12-16
 - Certificate of License 2-1, 2-3
 - CHAR 7-3
 - CHAR_TO_BYTE 14-3
 - CHAR_TO_INT 14-3
 - Char-Konstante 9-9
 - Codebausteine 3-4, 4-8, 6-1
 - Compiler 1-2, 4-19
 - Einstellen des Compilers 4-19
 - Compileroptionen 6-24, 15-33
 - CONCAT 14-13
 - CONTINUE-Anweisung 12-12, 12-23
 - COS 14-10
 - COUNTER 7-16, 13-1

D

DATE 7-4
 DATE_AND_TIME 7-5
 DATE_TO_DINT 14-3
 Datenbausteine 6-19, 10-7, 10-8, 10-10, 10-11
 Datentyp ANY 7-19
 Datentyp ARRAY 7-10
 Datentyp COUNTER 7-16
 Datentyp DATE_AND_TIME 7-5
 Datentyp- Konvertierungsfunktionen 14-3, 14-6
 Datentyp POINTER 7-17
 Datentyp STRING 7-7
 Datentyp STRUCT 7-12
 Datentyp TIMER 7-16
 Datentyp UDT 7-14
 Datentypen 7-1 - 7-14
 anwenderdefinierte (UDT) 6-22, 7-14
 Beschreibung 7-1
 elementar 7-2
 zusammengesetzte 7-2
 Datentypen für Parameter 7-16, 7-17
 Datum stellen 4-40
 Datumskonstante 9-13
 Debugger 1-2
 Definieren einer Aufrufumgebung für Bausteine 4-32
 Definieren einer Aufrufumgebung für Haltepunkte 4-35
 Deinstallieren
 der Nutzungsberechtigung 2-5
 Deklaration 6-10
 Deklaration statischer Variablen 8-3
 DELETE 14-15
 DI_STRNG 14-19
 Diagnosepuffer 4-41
 DIN Norm EN-61131-3 1-1
 DINT 7-3
 DINT_TO_DATE 14-3
 DINT_TO_DWORD 14-3
 DINT_TO_INT 14-3, 14-4
 DINT_TO_TIME 14-4
 DINT_TO_TOD 14-4
 DIV 11-8
 Division 11-2
 Doppelwort 7-3
 Download 4-25
 druckbare Zeichen 9-9, 9-11
 Drucken einer S7-SCL-Quelle 4-23
 Durchgangparameter 8-1, 12-34
 Durchgangszuweisung 12-34, 12-42
 Durchgangszuweisung (FB/SFB) 12-34
 Durchgangszuweisung (FC) 12-42

DWORD 7-3
 DWORD_TO_BOOL 14-4
 DWORD_TO_BYTE 14-4
 DWORD_TO_DINT 14-4
 DWORD_TO_REAL 1) 14-3
 DWORD_TO_WORD 14-4

E

Editor 1-2
 Einfacher Ausdruck 11-7
 Einfügen von Bausteinaufrufen 4-16
 Einfügen von Bausteinvorlagen 4-16
 Einfügen von Kontrollstrukturen 4-17
 Einfügen von Parametervorlagen 4-17
 Einfügen von Vorlagen für Kommentar 4-16
 Eingangsparameter 8-1, 12-33, 12-41, 12-45
 Definition 8-1
 Eingangsparameter EN 12-45
 Eingangszuweisung (FB) 12-33
 Eingangszuweisung (FC) 12-41
 Einrücken von Zeilen 4-13
 Einsteigerbeispiel 3-1
 Einstellen des Seitenformats 4-22
 Einstellungen 4-19
 Einteilungen 4-3
 Einzelschritt 4-30
 Elementare Datentypen 7-1, 7-3, 7-4
 EN 12-45
 ENO 12-46
 Entwerfen von S7-SCL-Programmen 3-1
 Entwicklungsumgebung 1-2
 EQ_STRNG 14-17
 Ersetzen von Textobjekten 4-9
 Erstellen einer Übersetzungssteuerdatei 4-21
 Erweiterte Variable 11-4
 Erzeugen bzw. Anzeigen der Referenzdaten 4-37
 Erzeugen von S7-SCL-Quellen mit einem Standard-Editor 4-6
 EXIT-Anweisung 12-12, 12-24
 Exklusiv-Oder 11-2
 EXP 14-9
 EXPD 14-9

F

Farbe und Schriftart des Quelltextes 4-14, 4-24
 FB-Parameter 12-31 - 12-35
 FC 6-16, 12-27, 12-38
 FC-Parameter 12-40 - 12-42

Fehlerbehebung nach der Übersetzung 4-21
Feld (ARRAY)
 Initialisierung 8-4
 Wertzuweisungen mit Variablen vom Datentyp ARRAY 12-5
Felder 7-10
Festlegen der Objekteigenschaften 4-6
FIND 14-16
Flags (OK-Flag) 8-9
Flussdiagramm von AUSWERTEN 3-13
Flussdiagramm von Erfassen 3-17
FOR-Anweisung 12-12, 12-18, 16-5
Formale Sprachbeschreibung 15-1
Formalparameter 8-1
Formatfreiheit 5-2, 5-3
Funktion (FC) 6-16, 12-27, 12-38
Funktionen von S7-SCL 1-4
Funktionen zum Runden und Abschneiden 14-6
Funktionen zur Selektion von Werten 14-23
Funktionsbaustein (FB) 6-14, 12-27, 12-29, 12-31
Funktionsleiste 4-2

G

Ganzzahlige Division 11-2
Ganzzahl-Konstante 9-7
GE_STRNG 14-17
Gehe zu 4-12
Gleichheit 11-2
Globale Daten 10-1
 Übersicht über globale Daten 10-2
Globale Instanz 12-29, 12-35
GOTO-Anweisung 12-25
Grenzwerte für FOR-Anweisungen 16-5
GT_STRNG 14-18

H

Haltepunkte 4-33, 4-34, 4-36
Hantieren einer S7-SCL-Quelle 4-5, 4-22
Hantieren einer S7-SCL-Quelle 4-5, 4-6, 4-22, 4-23

I

I_STRNG 14-19
IF-Anweisung 12-12, 12-14
Indizierter Zugriff auf Datenbausteine 10-10
Indizierter Zugriff auf Speicherbereiche der CPU 10-6
Initialisierung 8-4

INSERT 14-15
Installation 2-6
Installationsvoraussetzungen 2-6
Installieren des Automation License Managers 2-4
Instanzdeklaration 8-8
INT 7-3
INT_TO_CHAR 14-4
INT_TO_WORD 14-4

K

Klammerung 11-2
Kommentar
 Blockkommentar 5-15
 Einfügen von Vorlagen für Kommentar 4-16
 Lexikalische Regeln 15-31
 Zeilenkommentar 5-16
Kommunikation der CPU 4-43
Konstanten 9-2 - 9-17
Kontinuierlich beobachten 4-28
Kontrollanweisungen 4-17, 6-13, 12-14
 Anweisungen 6-13
 CASE-Anweisung 12-16
 CONTINUE-Anweisung 12-23
 Einfügen von Kontrollanweisungen 4-17
 EXIT-Anweisung 12-24
 FOR-Anweisung 12-18, 12-19
 GOTO-Anweisung 12-25
 IF-Anweisung 12-14
 REPEAT-Anweisung 12-22
 Syntaktische Regeln 15-52
 WHILE-Anweisung 12-21
Konvertierungsfunktionen 14-3
 Klasse B 14-3
Kopieren von Textobjekten 4-10

L

Labels (Sprungmarken) 9-18
Laden 4-25, 4-26
LE_STRNG 14-17
LEFT 14-14
LEN 14-13
Lesezeichen 4-15
Lexikalische Regeln 15-21
License Key 2-1, 2-2, 2-5
License Manager 2-1, 2-2
Literale 9-6 - 9-17
 siehe Konstanten 15-23
Lizenz 2-2, 2-3

Lizenz-Typen 2-3
 Enterprise License 2-1
 Floating License 2-3
 Rental License 2-3
 Single License 2-3
 Trial License 2-3
 Upgrade License 2-3
 LN 14-9
 LOG 14-9
 Logarithmische Funktionen 14-9
 Lokale Daten 5-17, 8-1, 8-4, 8-11, 8-12
 Lokale Instanz 12-29, 12-30, 12-37
 Löschen von Textobjekten 4-11
 LT_STRNG 14-18

M

Markieren von Textobjekten 4-10
 Mathematische Standardfunktionen 14-9,
 14-10
 Menüleiste 4-2
 Merker 10-2
 MID 14-14
 MOD 11-8, 11-9
 Modulo-Funktion 11-2
 Multiinstanzen 8-8
 Multiplikation 11-2

N

Namen 5-6
 Beispiele 5-6
 Definition 5-6
 Formale Sprachbeschreibung 15-15,
 15-17
 Regeln 5-6
 NE_STRNG 14-17
 Negation 11-2
 nicht druckbare Zeichen 9-9, 9-11
 Non-Terminal (in Syntaxdiagrammen)
 15-14
 Normerfüllung 1-1
 NOT 11-10, 11-11
 Notautorisierung 2-6
 Nullpointer 7-19
 Numerische Datentypen 7-3
 Numerische Standardfunktionen
 14-9, 14-10
 Nutzungsberechtigung 2-6
 Nutzungsberechtigung durch den
 Automation License Manager 2-1

O

OB 6-18
 Oder 11-2
 Öffnen einer S7-SCL-Quelle 4-5
 Öffnen von Bausteinen 4-6
 OK-Flag 8-1, 8-9
 Operanden 11-3, 11-4
 Operandenkennzeichen 5-9
 Operationen 15-9
 alphabetische Auflistung 15-7
 OR 11-10, 11-11
 Organisationsbaustein 6-18

P

Parameter 6-11, 7-16, 7-17, 8-1, 8-10,
 8-13, 8-14, 12-31, 12-40 - 12-42
 Parameterversorgung 12-27
 Parameterversorgung
 bei Zählfunktionen 13-3
 Parameterversorgung
 bei Zeitfunktionen 13-11
 Parametervorlagen 4-17
 Peripherie Ein- / Ausgänge 10-2
 Platzieren von Lesezeichen
 im Quelltext 4-15
 POINTER 7-17, 7-18
 Positionieren der Einfügemarke in einer
 bestimmten Zeile 4-12
 Potenz 11-2
 Programmwurf 3-1
 Programmiersprache
 höher 1-1, 1-4
 Programmierung
 strukturiert 1-4
 Programmsprung 12-12
 Programmverzweigung 12-12
 Prozessabbild der Ein- und Ausgänge 10-2
 Prüfen der Bausteinkonsistenz 4-38

Q

Quelle 4-4, 4-5 - 4-8, 4-22, 4-23, 6-12, 6-22

R

R_STRNG 14-20
 REAL 7-3
 REAL_TO_DINT 14-4
 REAL_TO_DWORD 2) 14-3
 REAL_TO_INT 14-4
 Realzahl-Konstante 9-8
 Referenzdaten 4-37

- Regeln
 - für den Umgang mit License Keys 2-5
 - Regeln für den Umgang mit License Keys 2-5
 - Regeln für S7-SCL-Quellen 4-7
 - Regelstrukturen 5-1
 - Reihenfolge der Bausteine 4-8
 - REPEAT-Anweisung 12-12, 12-22
 - REPLACE 14-16
 - Reservierte Wörter 5-5
 - RETURN-Anweisung 12-12, 12-26
 - Returnwert 12-39
 - siehe Rückgabewert 12-39
 - RIGHT 14-14
 - ROL 14-11
 - ROR 14-11
 - ROUND 14-6
 - Rückgabewert (FC) 12-39
 - Rückgängigmachen der letzten Editieraktion 4-9
- S**
- S_CD 13-6
 - S_CU 13-6
 - S_CUD 13-7
 - S_ODT 13-17
 - S_ODTS 13-18
 - S_OFFDT 13-19
 - S_PEXT 13-16
 - S_PULSE 13-15
 - S5-Time 13-13
 - S5TIME 7-4
 - S7-SCL 1-2, 1-4, 4-2
 - Arbeitsweise 1-2
 - Bedienoberfläche 4-2
 - Funktionen 1-4
 - Grundbegriffe 5-4 - 5-13
 - Schleifenbearbeitung 12-12
 - Schließen einer S7-SCL-Quelle 4-5
 - Schlüsselwörter 5-5, 15-10
 - Schriftstil und -farbe 4-14, 4-24
 - Seitenformat 4-22
 - Seitenumbruch 4-24
 - SFCs/SFBs 14-27
 - SHL 14-11
 - SHR 14-11
 - Sichten auf Variablenbereiche 8-6
 - SIN 14-10
 - Speicherbereiche der CPU 5-9, 10-1, 10-2, 10-3, 10-5, 10-6
 - Speichern einer S7-SCL-Quelle 4-22
 - Speicherort von Variablen 8-6
 - Sprachbeschreibung 5-1, 15-1
 - Sprunganweisungen 12-12
 - Sprungmarken 9-18
 - SQR 14-9
 - SQRT 14-9
 - Stacks 4-43
 - Standardbezeichner 5-7
 - Standardfunktionen 14-3, 14-6, 14-9, 14-10, 14-11
 - Starten
 - S7-SCL 4-1
 - Statische Variablen 5-17, 8-1, 8-3, 8-8, 8-11
 - Statuszeile 4-2
 - STEP 7-Testfunktionen 4-37, 4-38
 - Steuerdatei für die Übersetzung 4-21
 - STRING 7-7 - 7-9, 9-9, 9-12, 14-13, 14-14 - 14-20
 - STRING_TO_CHAR 14-4
 - STRNG_DI 14-19
 - STRNG_I 14-19
 - STRNG_R 14-20
 - STRUCT 7-12, 7-13
 - Strukturen 7-12
 - Strukturierte Programmierung 3-4, 3-6
 - Strukturierter Zugriff
 - auf Datenbausteine 10-11
 - Subtraktion 11-2
 - Suchen von Textobjekten 4-9
 - Symbolisch programmieren 4-8
 - Symbolische Konstanten 9-2
 - Symbolischer Zugriff auf Speicherbereiche der CPU 10-5
 - Syntaktische Regeln 15-34
 - Syntaxdiagramme 5-1, 15-1
 - Syntaxgerechtes Formatieren des Quelltextes 4-14
 - Systemattribute 6-9, 6-11
 - für Bausteine 6-9
 - für Parameter 6-11
 - Systemfunktionen/-funktionsbausteine und Standardbibliothek 14-27
- T**
- Tageszeit-Konstante 9-16
 - TAN 14-10
 - Temporäre Variablen 5-17, 8-1, 8-12
 - Terminale der lexikalischen Regeln (Syntaxdiagramme) 15-5
 - Testen in Einzelschritten 4-30
 - Testen mit Haltepunkten 4-30
 - Testfunktionen S7-SCL 4-27 - 4-32
 - Testfunktionen STEP 7 4-37, 4-38
 - Textmarke 4-15
 - TIME 7-4

TIME_OF_DAY 7-4
 TIME_TO_DINT 14-4
 Timer 7-4, 7-16, 13-9 - 13-21
 Titelzeile 4-2
 TOD_TO_DINT 14-4
 Trigonometrische Funktionen 14-10
 TRUNC 14-6

U

Übersetzen 4-18 - 4-21, 6-24
 Übersetzungsteuerdatei 4-21, 6-24
 UDT 7-15
 Aufruf 6-22
 Definition 6-22, 6-23
 Elemente 6-22
 Uhrzeit stellen 4-40
 unäres Minus 11-2
 unäres Plus 11-2
 Und 11-2
 Ungleichheit 11-2
 Unterprgrammbearbeitung 6-13
 Urlöschen des CPU-Speichers 4-25

V

VAR 8-10
 VAR_IN_OUT 8-10
 VAR_INPUT 8-10
 VAR_OUTPUT 8-10
 VAR_TEMP 8-10
 Variablen
 Allgemeine Syntax einer Variablen- oder
 Parameterdeklaration 8-3
 Initialisierung 8-4, 8-5
 Instanzdeklaration 8-8
 Lokale Variablen und
 Bausteinparameter 8-1
 statische Variablen 5-17, 8-1
 temporäre Variablen 5-17, 8-1
 Übersicht der Vereinbarungsblöcke 8-10
 Variablen beobachten/steuern 4-38
 Vereinbarungsteil 6-10, 8-4, 8-10 - 8-14
 Aufbau 6-10
 Bausteinparameter 8-14
 Definition 6-10
 Initialisierung 8-4
 Statische Variablen 8-11
 Syntaktische Regeln 15-37
 Temporäre Variablen 8-12
 Übersicht der Vereinbarungsblöcke 8-10
 Vergleichsausdrücke 11-13, 11-14
 Vordefinierte Konstanten und Flags
 Formale Sprachbeschreibung 15-20
 Vorlagen 4-16
 für Bausteine 4-16

für Kommentar 4-16
 für Kontrollstrukturen 4-17
 für Parameter 4-17

W

Warnungen 4-21
 Was ist neu? 1-6
 Wertzuweisung 6-13, 12-2, 12-3, 12-7,
 12-8, 12-9, 12-10
 Syntaktische Regeln 15-47
 Wertzuweisung mit Variablen
 vom Typ STRUCT und UDT 12-3
 Wertzuweisungen mit Absolutvariablen
 für Speicherbereiche 12-9
 Wertzuweisungen mit globalen Variablen
 12-10
 Wertzuweisungen mit Variablen eines
 elementaren Datentyps 12-2
 Wertzuweisungen mit Variablen vom
 Datentyp ARRAY 12-5
 Wertzuweisungen mit Variablen
 vom Typ DATE_AND_TIME 12-8
 Wertzuweisungen mit Variablen
 vom Typ STRING 12-7
 WHILE-Anweisung 12-12, 12-21
 Wiederherstellen einer Editieraktion 4-9
 WORD 7-3
 WORD_TO_BLOCK_DB 14-4
 WORD_TO_BOOL 14-4
 WORD_TO_BYTE 14-4
 WORD_TO_INT 14-4

X

XOR 11-10

Z

Zahlen 5-11
 Zähler 13-2 - 13-7
 Abwärtszählen (S_CD) 13-6
 Auf- und Abwärtszähler (S_CUD) 13-7
 Aufruf von Zählerfunktionen 13-1
 Aufwärtszähler (S_CU) 13-6
 Beispiel zu Zählerfunktionen 13-7
 Eingabe und Auswertung des
 Zählerwerts 13-5
 Parameterversorgung
 bei Zählerfunktionen 13-3
 Zeichenketten 5-13
 Zeichensatz 5-4
 Zeichentypen 7-3
 Zeilenkommentar 5-16
 Zeilennummern 4-3, 4-24
 Zeitdauer-Konstante 9-13

- Zeiten 7-4, 13-9 - 13-21
 - Aufruf von Zeitfunktionen 13-9
 - Beispiele 13-20
 - Eingabe und Auswertung des Zeitwerts 13-13
 - Parameterversorgung bei Zeitfunktionen 13-11
 - Zeit als Ausschaltverzögerung starten (S_OFFDT) 13-19
 - Zeit als Einschaltverzögerung starten (S_ODT) 13-17
 - Zeit als Impuls starten (S_PULSE) 13-15
 - Zeit als speichernde Einschaltverzögerung starten (S_ODTS) 13-18
 - Zeit als verlängerter Impuls starten (S_PEXT) 13-16
- Zeitsystem der CPU 4-42
- Zeitwert 13-13
- Zusammengesetzte Datentypen 7-2, 7-5, 7-7
- Zuweisen von Strukturen mit ungerader Byte-Länge 16-4
- Zykluszeit 4-42