

Automatisierungstechnik nach internationaler Norm programmieren (8)

Autor: Dr. Ulrich Becker

Fachzentrum Automatisierungstechnik und vernetzte Systeme im BTZ Rohr-Kloster

Mail: Ulrich.Becker@BTZ-Rohr.de

Folge 8: Variablendeklaration und Fehlersuche

In den Folgen 6 und 7 wurde ein Programm nach den Forderungen der Aufgabenstellung 2 erarbeitet. Als letzte Handlung wurde dazu ein weiterer Funktionsbaustein mit Zeitgliedern nach IEC 61131-3 entwickelt. Der vor dem Laden des Programms in den Controller angestoßene Übersetzungslauf ergab zahlreiche Fehler. Die Ursache wurde in der nicht korrekten Deklaration der Variablen vermutet. In dieser Folge werden deshalb die Kenntnisse über Variablen vertieft und Methoden der Fehlersuche vorgestellt.

Lokale und globale Variable

Die 21 gemeldeten Fehler beim Übersetzen des in den Folgen 6 und 7 erarbeiteten Programms sind keineswegs eigenständige Fehler, sondern bedingen sich vielfach gegenseitig. Es zeigt sich, dass Variable, die beispielsweise im Funktionsbaustein „Motorsteuerung“ deklariert wurden, im Baustein „Zaehlerfkt“ nicht bekannt sind. Ein Beispiel dafür ist die Variable „Band_Re“, welche bei Wert TRUE den Bandmotor im Rechtslauf einschaltet.

Was sind Variable? Variable sind Größen, deren Werte während der Programmdurchführung geändert werden. Mitunter werden sie deshalb sehr einfach als Platzhalter bezeichnet. Variable können gelesen und beschrieben werden. Sie treten uns mit Ihrem Namen (Bezeichner) und ihrem Datentyp entgegen. Bevor man eine Variable im Programm nutzen kann, muss man sie dem Programmiersystem „bekannt machen“. Das erfolgt im Deklarationsteil jedes Bausteins oder aber in den Ressourcen eines Projektes.

Beim Programmieren haben wir bisher alle erforderlichen Variablen stets in der aktuellen POE und dort in den Block zwischen die Schlüsselwort VAR und END_VAR geschrieben, und zwar so, wie sie gerade anfielen. Damit haben wir **lokale** Variable deklariert. Lokale Variable gelten nur in der POE, in der sie deklariert wurden. Sie haben keine Verbindung zu anderen Programmteilen und können von dort weder gelesen noch beschrieben werden.

Alle Variablen, die bausteinübergreifend benutzt werden, müssen deshalb **global** deklariert werden. Globale Variable werden durch das Schlüsselwort VAR_GLOBAL gekennzeichnet. Sie sind Bestandteil der Ressourcen eines Projektes (Folge 2 Bild 5). Es ist möglich, eine globale Variable mit dem gleichen Namen wie eine lokale zu definieren. Innerhalb eines Bausteins hat dann stets die lokal definierte Variable Vorrang!

Zur Korrektur des Projektes müssen wir deshalb eine Reihe von Variablen global definieren. Mit den allgemein bekannten Werkzeugen -> *Ausschneiden* und -> *Einfügen* ist dies problemlos möglich. Als Ergebnis zeigt **Bild 44** den Block der globalen Variablen. Alle hier nicht aufgeführten Variablen bleiben weiterhin lokale Variable.

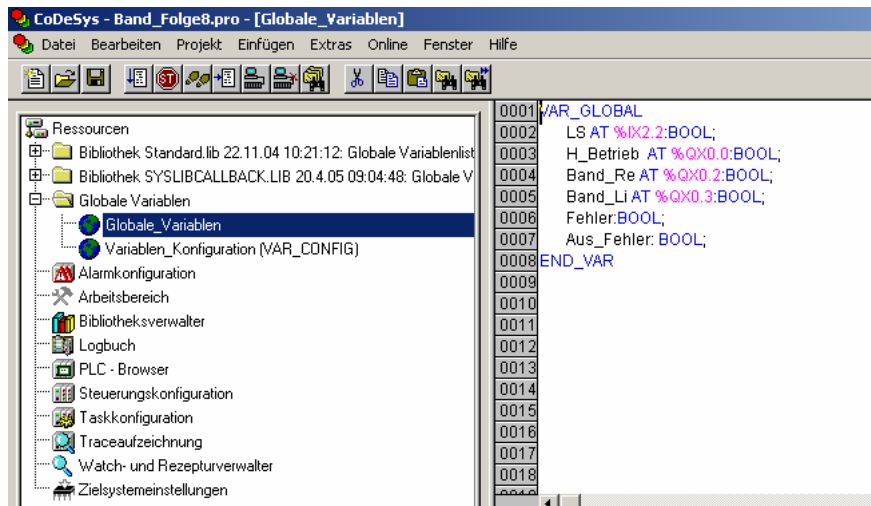


Bild 44: Globale Variable des Projektes nach Aufgabenstellung 2

Mit lokalen und globalen Variablen kennen wir zwei wichtige Arten von Variablen. Darüber hinaus können aber noch vielfältige andere Variable deklariert werden (**Tabelle 9**)

Schlüsselwort	Erläuterung
VAR	Lokale Variable, gilt nur innerhalb einer POE
VAR_GLOBAL	Globale Variable
VAR_CONSTANT	Lokal benutzte Konstante
VAR_GLOBAL_CONSTANT	Global benutzte Konstante
VAR_EXTERNAL	Von außen(von der Konfiguration) gelieferte Variable
VAR_GLOBAL_RETAIN	Nullspannungssichere (gepufferte) Variable
VAR_GLOBAL_PERSISTENT	Nach Programm-Download remanente Variable
VAR_CONFIG	Variable für Hardwarekonfiguration
VAR_INPUT	Gelesener Bausteinparameter, von außen in die POE kommend
VAR_OUTPUT	Geschriebener Bausteinparameter, von der POE nach außen geliefert
VAR_IN_OUT	Gelesener und geschriebener Bausteinparameter (Durchgangparameter), von außen kommend, in der POE veränderbar und nach außen geliefert
END_VAR	Abschluß eines beliebigen Variablenblockes

Tabelle 9: Die wichtigsten Schlüsselwörter für Variable nach IEC 61131-3

Variable vom Typ CONSTANT sind vom Wesen her eigentlich keine „Variable“, werden aber als solche geführt. Die Bausteinparameter vom Typ INPUT, OUTPUT sowie IN_OUT werden für parametrierbare Funktionen und Funktionsbausteine benötigt. Dieses Detail wird in den nächsten Folgen erläutert. Jeder Block dieser Variablen ist mit dem Schlüsselwort END_VAR abzuschließen. Nicht alle genannten Typen werden in gleicher Häufigkeit eingesetzt.

Zum Vergleich: Variablendeklaration in Step7

Step 7 kennt globale Variable als Elemente der Symboltabelle und als Elemente der (globalen) Datenbausteine. Sie werden durch Hochkomma gekennzeichnet, zum Beispiel "Variable_1". Für alle diese Variablen muss der Programmierer Speicherbereiche angeben und die Adressen selbst verwalten. Speicherbereiche liegen in der CPU (Merkerbereich, Timer und Zähler und Prozessabbild der Ein- und Ausgänge), im Peripheriebereich (z.B. in einer Analogbaugruppe) und im Datenbereich. Das Festlegen von Adressen als Bit, Byte, Wort oder Doppelwort ist immer wieder mit Aufwand und Fehlern verbunden. IEC Programmiersysteme verwalten diese Adressen selbst.

Lokale Variable werden im Deklarationsteil von Organisationsbausteinen (OB), Funktionen (FC) und Funktionsbausteinen (FB) als temporäre Variable TEMP deklariert und im Lokaldaten-Stack gespeichert. Für jeden Baustein sind eine begrenzte Menge lokaler Daten reserviert. Auch die Belegung des gesamten Lokaldatenstacks eines vom Baustein OB1 zyklisch abgearbeiteten Programms muss der Programmierer im Auge behalten. Die Adressen dieser Variablen verwaltet Step7 jedoch selbst. Sie werden angezeigt und man könnte mit ihnen absolut programmieren. Dies ist aber nicht üblich, man verwendet stattdessen die Bezeichner (Namen der Variablen)! Gleichfalls im Deklarationsteil der FC werden Bausteinparameter Typ IN, OUT und IN_OUT deklariert.

Anders bei Funktionsbausteinen: Eine Besonderheit von Step7 sind die Instanzdatenbausteine, mit denen FB's instanziiert werden. Hier werden deren Bausteinparameter Typ IN, OUT und IN_OUT sowie die statischen Variablen STAT verwaltet. Deklariert werden diese genau wie bei Funktionen im Deklarationsteil des Bausteins, dann aber vom System in den Instanzdatenbaustein überführt. Die besonderen statische Variablen benutzt man, wenn Lokaldaten ihre Werte über den aktuellen Aufruf des Bausteins hinaus erhalten sollen („Gedächtnis“). **Bild 45** zeigt die Deklarationstabellen für die Bausteintypen FC, FB und OB. Die OB's verfügen über eine Vielzahl eigener temporärer Variablen, der Programmierer kann aber weitere Variablen TEMP hinzufügen.

Im System CoDeSys haben wir bereits mehrfach Funktionsbausteine instanziiert. Dazu haben wir dem Aufruf eine Variable vom Typ eben dieses selbstdefinierten FB zugeordnet. Einen gesonderten Instanzdatenbaustein kennt IEC 61131-3 ebensowenig wie spezielle Datenbausteine oder Variable vom Typ STAT.

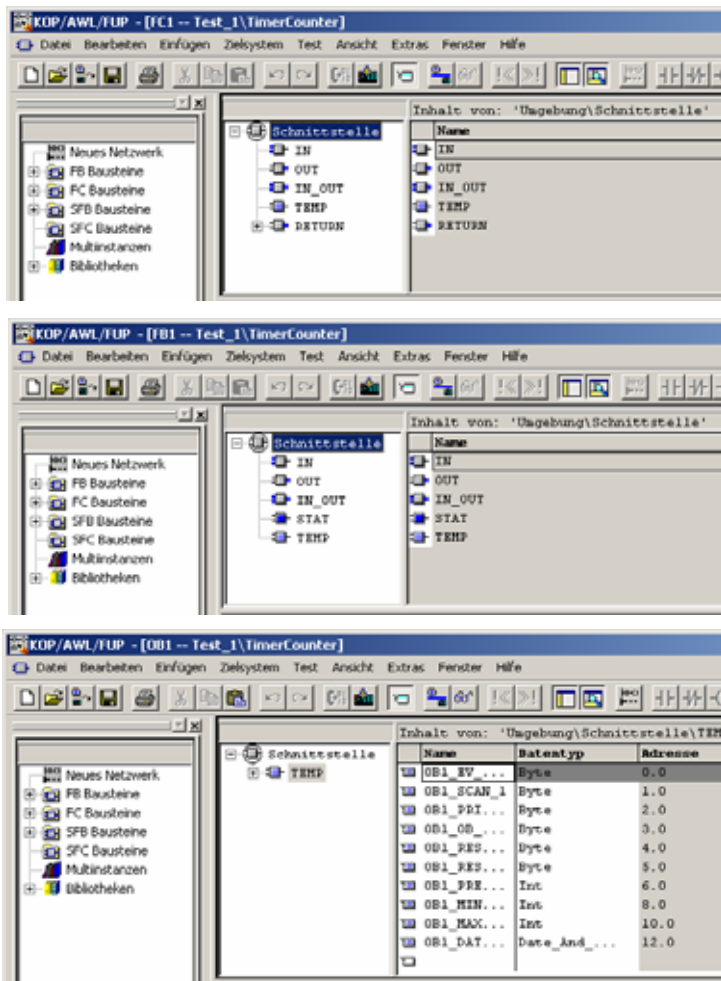


Bild 45: Variablendeklaration in Step7 Bausteinen: oben Funktion, Mitte Funktionsbaustein, unten OB1 als Beispiel eines Organisationsbausteins

Syntaxfehler und Logische Fehler

Auf dem Weg zu einem fehlerfreien Programm sind weitere Mängel auszuräumen. Im einfachsten Falle sind das Syntaxfehler, die beim Übersetzungslauf vom Compiler erkannt werden. Anklicken der Fehlermeldung führt zumeist direkt zur Fehlerstelle im Programm. Schwerwiegender sind Logische Fehler. Diese hat allein der Programmierer zu verantworten, und das System kann sie selbstverständlich nicht erkennen. Ein Beispiel ist die ursprünglich nicht korrekte Programmierung der Betriebsbereitschaft durch Wahl einer falschen Variablen im Netzwerk 1 des Bausteins „Motorsteuerung“ (Bild 36 in Folge 6).

Kenner von Step7 schätzen das weit ausgebaute System für Diagnose und Fehlersuche mit Systemmeldungen, Diagnosepuffer und dem Baustein-, Unterbrechungs- und Lokaldatenstack. Das Diagnosesystem erlaubt insbesondere schnelles Abstellen von Fehlern, die zum Stopp der CPU führen (bekannt als „Stopp-Fehler“). Einige dieser Fehler darf man aber als „Step7 hausgemacht“ bezeichnen, da sie z.B. durch Schreiben in Datenbereiche entstehen, die zuvor nicht angelegt wurden (Peripherie-Zugriffsfehler). Wenn Systeme die Adressierung der Variablen selbst erledigen, kommt es zu solchen Fehlern nicht! Zur Eingrenzung logischer Fehler dienen zuerst die Referenzdaten. Zu ihnen gehören Querverweise,

Belegungslisten und Programmstrukturen. Weitere Möglichkeiten ergeben sich mit den Werkzeugen -> "Variable beobachten und steuern".

Eine Reihe solcher Softwarewerkzeuge können auch im Programmiersystem CoDeSys genutzt werden. Als Beispiel zeigt **Bild 46** das Menu für die Ausgabe von Aufrufbaum und Querverweisliste sowie spezielle Überprüfungen. So sind mit -> "Mehrfaches Schreiben auf OUTPUT" die gefürchteten Mehrfachzuweisungen zu ermitteln. Über die Möglichkeiten von Steuern und Forcen haben wir bereits in Folge 5 berichtet

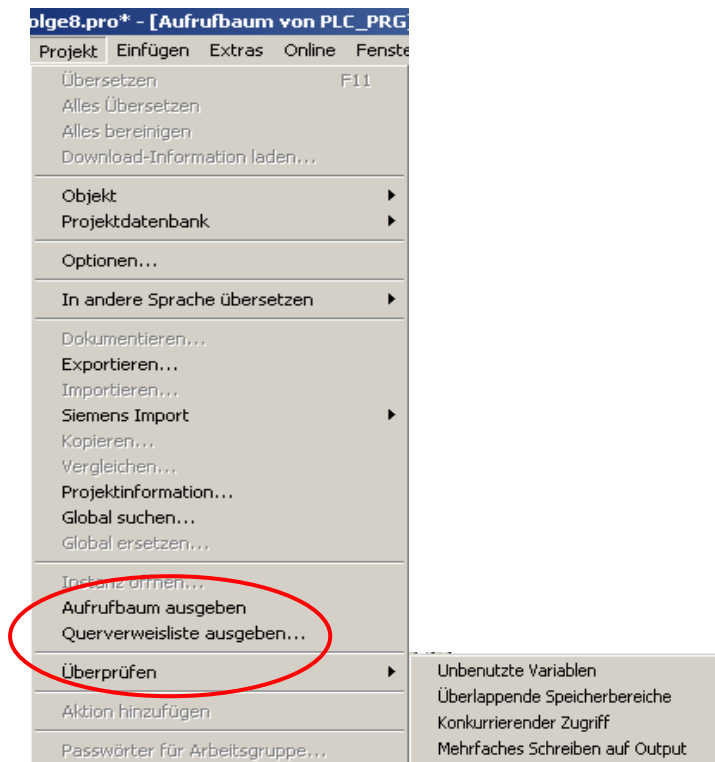


Bild 46: Menu für CoDeSys Softwarewerkzeuge zur Eingrenzung logischer Fehler

Bild 47 zeigt zwei Details aus diesem Menu: Im Aufrufbaum des Bausteins PLC_PRG werden die Bausteine in ihrer gegenseitigen Abhängigkeit dargestellt. Zu erkennen sind die drei von uns erarbeiteten Funktionsbausteine und die in ihnen aufgerufenen Standard-Funktionsbausteine. An einer solchen Abbildung kann man sich in einem Projekt grundsätzlich orientieren. Das zweite Beispiel zeigt die Querverweisliste für die Variable „Band_Re“. Hier ist aufgeführt, an welchen Stellen des Programms diese Variable geschrieben und gelesen wird.

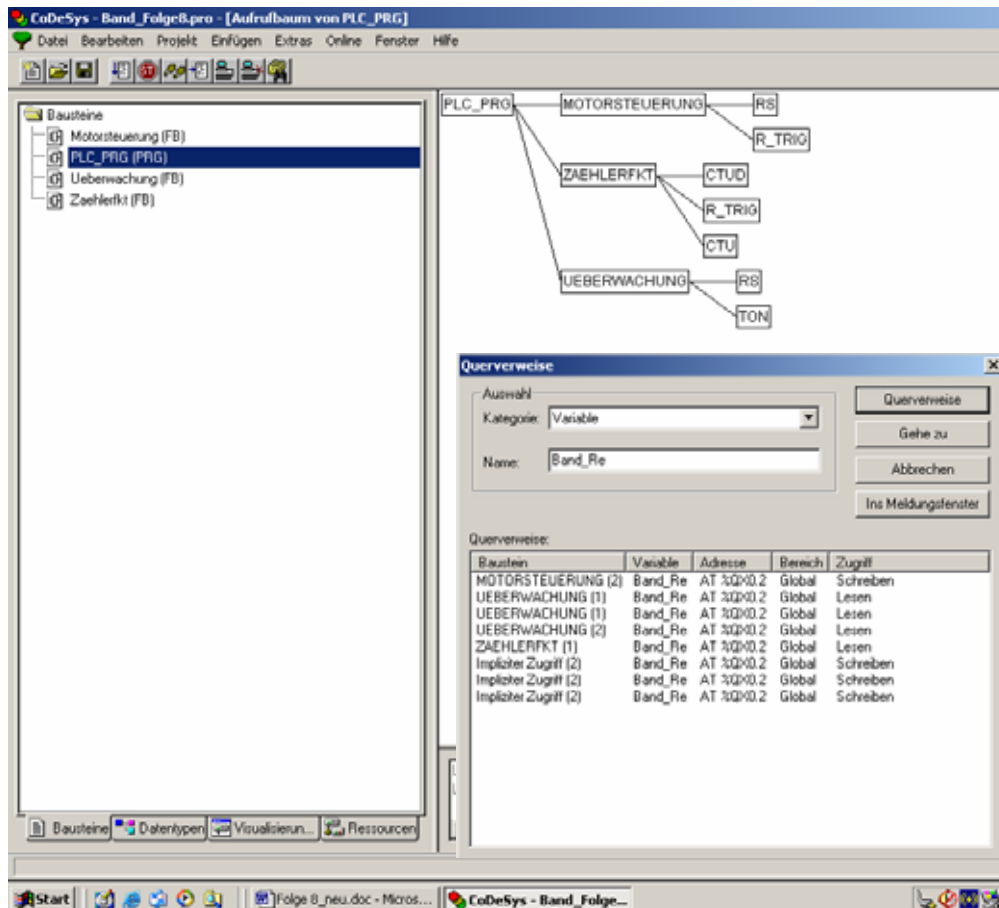


Bild 47: Beispiel für Aufrufbaum und Querverweisliste

Beim Testen des Programms am Trainingsrack zeigt sich ein weiterer unerwarteter Fehler: Die im Lager eintreffenden Teile werden nicht gezählt, obwohl das Programm des Bausteins „Zaehlfunkt“ formal korrekt scheint. Das Ergründen der Ursache ist etwas aufwendiger. Beim Programmieren mit graphischen Sprachen geht mitunter das Bewusstsein für die streng sequentielle Abarbeitung aller Befehle verloren. Dadurch können „Wettlaufprobleme“ derart entstehen, dass Signale „abgeschaltet“ werden, bevor sie ihre Steueraufgabe erfüllen. Eine solche Problematik zeigt **Bild 48**. In der POE „Motorsteuerung“ haben wir das Abschalten Bandes durch positive Flanken des Lichtschrankensignals programmiert. In der Programm-bearbeitung nachfolgend wird die gleiche Flanke zum Zählen der Teile genutzt, aber nur dann, wenn das Band läuft. Den Widerspruch dieses Beispiels kann man bereits dadurch beheben, dass im Programm PLC_PRG die POE „Zaehlerfkt“ vor der POE „Motorsteuerung“ aufgerufen wird.

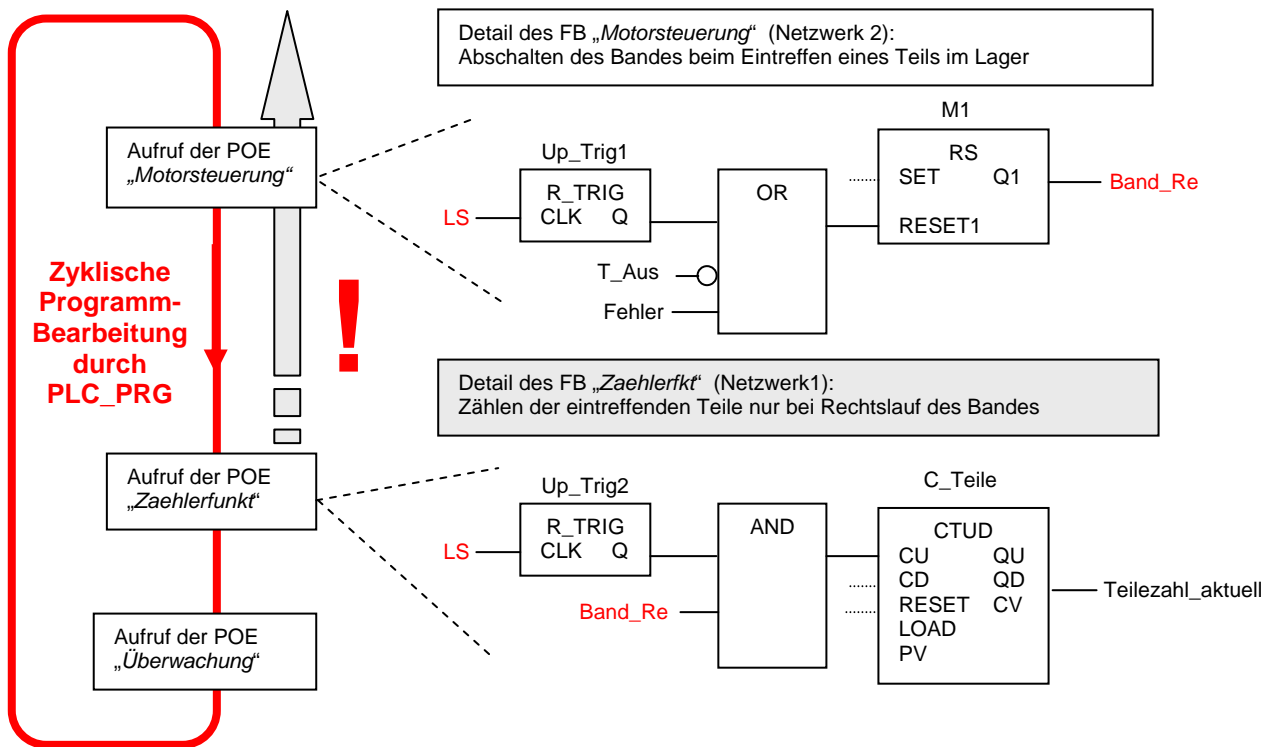


Bild 48: Logischer Fehler durch nicht korrekte Reihenfolge von Anweisungen

Schlussbemerkung:

In dieser Folge wurden die Kenntnisse über die Deklaration von Variablen vertieft und die Merkmale von lokalen und globalen Variablen dargestellt. Mit diesem Wissen und unter Nutzung von Softwarewerkzeugen für die Fehlersuche konnten verbliebene Mängel des Programms behoben werden. In den nächsten Folgen soll die Anwendung von Bausteinparametern als Schnittstelle für den Aufruf parametrierbarer Bausteine untersucht werden. In Folge 9 erfolgt dies zunächst für Funktion, Folge 10 für Funktionsbausteine.

Glossar:

absolute und symbolische Adressierung	In Step7 kann man Programme unter Verwendung von absolut adressierten Operanden wie E12.3 oder DBW 4 oder aber mit Symbolen (Bezeichnern) derselben schreiben (siehe Symboltabelle).
Mehrfach-Zuweisung	Mehrfaches Schreiben einer Variablen durch Zuweisen (= oder ST). Dies führt bei sequentieller zyklischer Programmbearbeitung immer zu Fehlern.
Compiler	Übersetzer. Beim Compilieren wird das in einer Hochsprache erstellte Programm auf Syntaxfehler geprüft und zumeist in eine „maschinennahe“ Sprache übersetzt.
Querverweis	Hinweis auf alle Verwendungsstellen einer bestimmten Größe in einem Programm
Referenzdaten	Referenz: Beziehung. Referenzdaten stellen die Bezüge und Zusammenhänge der Komponenten eines Projektes zueinander
Schlüsselwort	Eindeutige Zeichenkombination zur Festlegung von Datentypen, Operanden oder Operationen. Sie dürfen nicht für andere Zwecke verwendet werden, insbesondere nicht für die Bezeichnung von Variablen.
Sequentielle Programm-Bearbeitung	Sequenz: Folge Unter sequentieller Programmbearbeitung versteht man die Bearbeitung aller Anweisungen (Befehle) in gleichbleibend strenger Reihenfolge, so wie sie der Reihe nach formuliert wurden.
Stack	Stapelspeicher, auch Register
Symboltabelle	In Step7 eine Tabelle, in der den global geltenden Operanden und Bausteinen Symbole zugeordnet werden, ursprünglich aus Gründen der leichteren Lesbarkeit von Programmen. Im Wesen aber sind Symbole globale Variable.
Syntax	Satzlehre, hier für Schreibweise der Anweisungen benutzt