3 Die Sprache ST bzw. SCL im Detail

3.1 Unterschiede in den einzelnen Sprachen ST/SCL

Obwohl gerade die Übertragbarkeit des ST in der Industrie gelobt wird, haben die Tests in diesem Buch auch gravierende Unterschiede gezeigt, auf die in diesem Buch immer wieder eingegangen werden muss. An dieser Stelle sollen einige markante Punkte zusammengefasst werden.

3.1.1 Unterschiede in der Variablendarstellung

Durch die Unterschiede in der Variablendarstellung innerhalb des Programm-Codes wird die Möglichkeit, mittels Kopieren der Inhalte der Programme oder von Baustein-Teilen diese zu übertragen stark eingeschränkt. Hier hilft die Funktion <Suchen><Ersetzen> oder <Suchen><Löschen> vielfach weiter. Das wichtigste Element ist das # und die „"-Zeichen in den SCL-Programmen.

Statement SCL

```
(* DB_Waschm1 = Name des Datenbausteins,  x_X5VerschlussZu = Name
     der Variablen *)
#x_Motor := #x_Start  AND "DB_Waschm1".x_X5VerschlussZu ;
```

Anmerkung für SCL:

Kennung für eine lokale Variable

„Datenbaustein".Variable

Statement ST

```
x_Motor := x_Start  AND x_X5VerschlussZu ;
```

Hinweis: Wenn ST-Code zum TIA Portal/SCL eingefügt wird, ergänzt der SCL-Editor (TIA Portal) die Variablen automatisch.

Beispiel:

PC-Worx-Programm

```
(* Einfache Logik Test der Übertragung *)
x_Motor := x_IN1 AND x_IN2 OR (i_Anzahl =0 ) ;
```

Nach dem Kopieren zum TIA Portal

```
(* Einfache Logik Test der Übertragung *)
x_Motor := "x_IN1" AND "x_IN2" OR (i_Anzahl = 0);
```

Nach der Bekanntgabe der Variablen im TIA Portal

```
(* Einfache Logik Test der Übertragung *)
#x_Motor := "x_IN1" AND "x_IN2" OR (#i_Anzahl = 0);
```

= Variable Lokal, " " = Variable Global

3.1.2 Unterschiede in der Programmierung

Beim Aufruf von Funktionen oder Funktionsbausteinen zeigen sich deutliche Unterschiede. Diese Unterschiede sollten bekannt sein, bilden aber keine Nachteile, da jedes Problem auf unterschiedlichen Wegen gelöst werden kann. Im Weiteren werden einige Hinweise dazu gegeben.

3.1.2.1 PC Worx – Ansprache eines Arrays (zweidimensional)

Die Unterschiede bestehen (insbesondere bei PC Worx) in der Bekanntgabe eines mehrdimensionalen Felds und auch hinsichtlich der Ansprache der einzelnen Array-Felder.

Vergleich:

Ansprache eines Arrays (2D) im PC Worx:

```
i_Erg  := Arri_Messwerte[5][7];
```

Ansprache eines Arrays (2D) bei Codesys 2.3/TIA Portal/logi.CAD 3

Sonst ST oder SCL:

```
i_Erg  := Arri_Messwerte[5,7];
```

Wenn Softwareinhalte übertragen werden sollen, können die notwendigen Korrekturen mit <Suchen><Ersetzen> durchgeführt werden.

3.1.2.2 Unterschiede bei der Bekanntgabe von Startwerten (Initialisierung)

Bei Codesys 2.3 und dem TIA Portal können die Start- oder Initialwerte direkt bei der Variablendeklaration vorgegeben werden. Bei PC Worx oder logi.CAD 3 geschieht diese Initialisierung als Zuweisung im Programm, oder in getrennten, speziellen Funktionsbausteinen. Dieser Weg kann sehr hilfreich sein, da damit auch Umschaltungen (Sprache, Rezepturen, Nockenschaltwerke) ohne Neustarts während des laufenden Betriebs möglich sind. Ebenso können durch Import neuer Funktionsbausteine neue Sprachen, Rezepturen, Programmsteuerungen u. v. m. integriert werden, ohne in die Details der Steuerung einzugreifen (siehe auch Kapitel 4 und 10).

3.1.2.3 PC Worx – Unterschiede Bekanntgabe von Arrays oder Strukturen

Der Weg Arrays bekannt zu geben, differiert am deutlichsten bei PC Worx. Bei dieser Softwareumgebung wird das Array als eigener Datentyp definiert und zur Anwendung in den einzelnen Bausteinen instanziiert, bzw. wie in einer „Mutter-Kind-Beziehung" vererbt (siehe auch Kapitel 2 und 4).

3.1.2.4 PC Worx – Unterschiede bei Zuweisungs- und Testaktionen

Bei allen arithmetischen Funktionen und Vergleichen geht der Compiler von dem Typ INTEGER oder höher aus. Wird der Wert einem USINT, WORD usw. zugewiesen oder dagegen getestet, muss der Typ vorangestellt werden (siehe Beispiele).

Arithmetik PC Worx (beachte usint#1)

```
usi_XNr := usint#1;  (* Schrittzeiger rücksetzen *)
```

Testaktion PC Worx (beachte word#16#1)

```
WHILE w_SK001Copy <> word#16#1 DO
```

3.1.2.5 Ansprache von Bit in Byte – WORD, ...

Allgemein lassen es die getesteten Systeme zu, Elemente einer Variablen direkt abzufragen (Bit in Byte oder WORD bzw. Doppelwort, Byte in WORD oder Doppelwort, WORD in Doppelwort). Die Syntax weicht jedoch teilweise ab.

Aufgabe: Das 5. Bit eines Status-Word (w_Status) wird einem Motor zugewiesen (x_Motor5).

Beispiel Codesys 2.3:

```
x_Motor5 := w_Status.5 ;
```

Beispiel PC Worx:

```
x_Motor5 := w_Status.X5 ;
```

Beispiel TIA Portal (SCL):

eingeben wird: `x_Motor5 := w_Status.X5 ;`
Es erscheint automatisch: `x_Motor5 := w_Status.%X5 ;`

3.1.2.6 Unterschiede bei den Funktionen

Je nachdem wie stark sich die einzelnen Systeme an der IEC 61131-3 ausrichten, fallen die Möglichkeiten der Funktionen unterschiedlich aus. Jede Norm bringt viele Vorteile für den Anwender, schränkt aber auch die Möglichkeiten, Kreativität und Flexibilität der Hersteller ein. Diese Tatsache bestimmt sehr stark die Freiheiten der Funktionen und die Art der Realisierung.

3.1.2.6.1 PC Worx

In PC Worx kann der Anwender den Funktionsausgang nutzen (keine zusätzlichen Ausgänge) und dieser Funktionsausgang kann nur ein Grund-Datentyp sein (kein Array und keine Struktur).

Anmerkung: Darin liegt kein Nachteil, da alle Funktionen auch problemlos als Funktionsbaustein realisiert werden können.

3.1.2.6.2 Codesys 2.3

Codesys 2.3 lässt nur den Funktionsausgang zu, erlaubt aber als Funktionsausgänge Arrays und Strukturen, sowie innerhalb der Funktion den Zugriff auf „globale Variablen". Damit hat der Anwender viele Freiheiten.

3.1.2.6.3 TIA Portal (SCL)

Das TIA Portal (SCL) lässt neben der Datenausgabe über den Funktionsausgang weitere Ausgänge zu, ebenso höhere Datentypen und Strukturen.

3.1.2.7 Zusätzliche Möglichkeiten der Programmsteuerung mit SCL: GOTO

SCL (S7-1200) hat zur Programmsteuerung noch zusätzlich das **GOTO**. Diese Anweisung in einem Programm würde bewirken, dass es kaum noch auf andere Steuerungshersteller übertragbar ist. Gerade das **GOTO** sollte daher kritisch gesehen werden, da es aus der Basic-Welt stammt oder der historischen AWL mit JMP-Befehlen usw. Details hierzu finden sich im Abschnitt zu „Kontroll- und Schleifenanweisungen".

3.1.2.8 Unterschiede bei den CASE-Anweisungen

SCL lässt überschneidende „Schaltziele" oder „CASE-Werte" zu, während PC Worx und Codesys auf derartige Überschneidungen beim anschließenden Kompilieren bzw. Übersetzen mit einer Fehlermeldung reagieren. Details werden unter „CASE-Anweisungen" beschrieben.

3.2 Mittel für den grafischen Vorentwurf

In der Automatisierungstechnik werden häufig zum Erstentwurf Darstellungsmittel benutzt, mit denen die Steuerung noch nicht direkt programmiert werden kann. Diese Entwurfsmittel sind also unabhängig von dem Automatisierungssystem und der verwendeten Programmiersprache. Zu diesen Beschreibungsmitteln gehören Struktogramme (Nassi-Schneidermann-Diagramm), Flussdiagramme, Petri-Netze u. v. m.

3.2.1 Struktogramme (bekannt unter Nassi-Schneidermann-Diagramm)

Die Struktogramme entsprechen sehr stark dem strukturierten Text und sehen keine Rücksprünge vor. Soweit diese notwendig wären, sind sie durch fertige Befehle (Kontrollstrukturen) abgedeckt (Case-, Repeat-, While-Anweisungen usw.).