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

ANHANG II

IEC61131

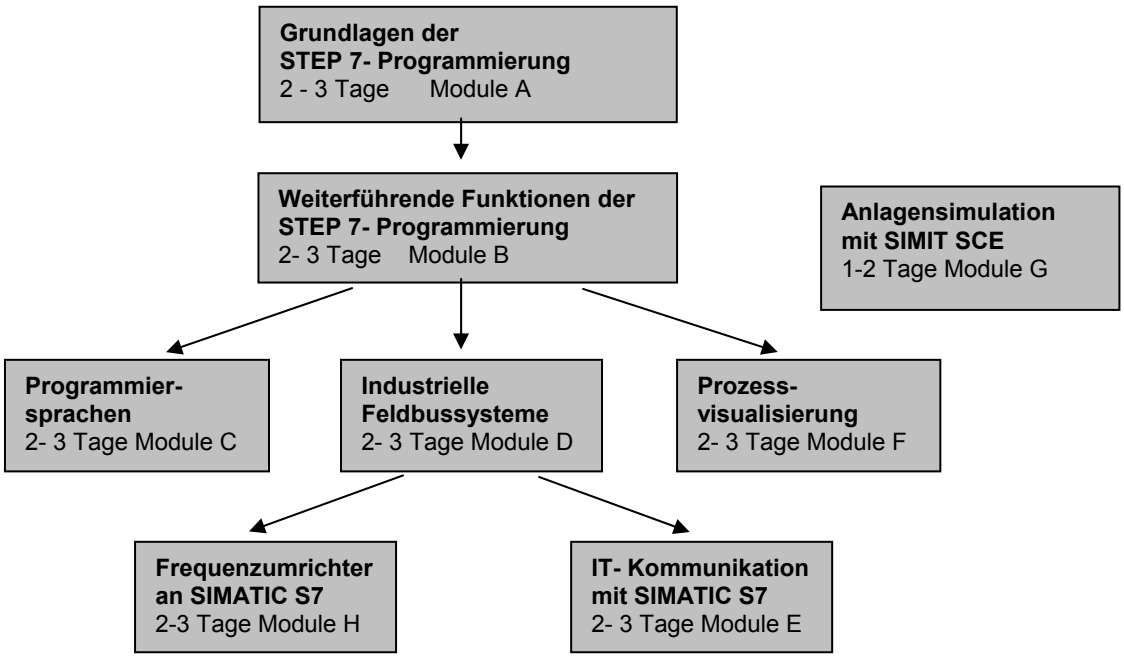
Wir danken der Fa. Michael Dziallas Engineering und den Lehrkräften von beruflichen Schulen sowie weiteren Personen für die Unterstützung bei der Erstellung der Unterlage

SEITE:

1.	**Vorwort**	4
2.	**Hinweise zur Norm IEC 61131**	5
3.	**Einführung in die Norm DIN 661131**	6
3.1.	DIN EN 661131 Teil 1, Allgemeine Information	6
3.2.	DIN EN 661131 Teil 2, Betriebsmittelanforderungen	6
3.3.	DIN EN 661131 Teil 3, Programmiersprachen (IEC 61131-3)	6
3.4.	DIN EN 661131 Teil 4, Anwenderrichtlinien	6
3.5.	DIN EN 661131 Teil 5, Kommunikation (bei IEC in Bearbeitung)	7
4.	**DIN EN 661131 Teil 3, Programmiersprachen**	7
4.1.	Das IEC 61131-3- Software-Modell	7
4.2.	Programme	8
4.3.	Variablen	9
4.4.	Die Anweisungsliste (AWL)	12
4.5.	Kontaktplan (KOP)	15
4.6.	Die Funktionsbaustein-Sprache (FBS)	16
4.7.	Die Ablaufsprache (AS)	16
4.8.	Strukturierter Text (ST)	21
4.9.	Dokumentation	26

1. VORWORT

Anhang II stellt eine theoretische Grundlage für die Bearbeitung sämtlicher Module dar.



Lernziel:

Der Leser erhält mit diesem Anhang Informationen zur internationalen Norm IEC 61131.

Voraussetzungen:

Da dies theoretische Grundlagen sind, werden auch keine speziellen Voraussetzungen benötigt.

2. HINWEISE ZUR NORM IEC 61131

Die Norm EN 661131-3 legt die Syntax und Semantik einer vereinheitlichten Reihe von Programmiersprachen für Speicherprogrammierbare Steuerungen (SPS) fest.
In diesem Modul soll dem Leser ein Einblick in Konventionen der Norm gewährt werden.
Es erfolgt keine komplette Abhandlung der Norm mit Erklärung aller Schaltzeichen, sondern nur eine allgemeine Beschreibung der wichtigsten Elemente der Sprachen

- Anweisungsliste (AWL)
- Funktionsbaustein-Sprache (FBS)
- Kontaktplan (KOP)
- Ablaufsprache (AS)
- Strukturierter Text (ST)

Die Unterlage basiert auf Auszügen und Interpretationen aus der Norm DIN EN 661131: 1993. Eine genauere Beschreibung der einzelnen Definitionen der Programmiersprachen kann den Normungsunterlagen entnommen werden.

Hinweise zur Normerfüllung

Die Programmiersprachen KOP und FUP entsprechen den in der Norm DIN EN 661131-3 (int. IEC 61131-3) festgelegten Sprachen „Kontaktplan“ und „Funktionsbaustein-Sprache“. Genau Aussagen zur Normerfüllung finden Sie in der Normerfüllungstabelle „NORM.TAB“. Die Datei befindet sich im Ordner „Siemens“ unter dem Verzeichnis „STEP 7“.

AWL entspricht der in der Norm DIN EN 661131-3 (int. IEC 61131-3) festgelegten Sprache „Anweisungsliste“, wobei hinsichtlich der Operationen wesentliche Unterschiede bestehen.
Genau Aussagen zur Normerfüllung finden Sie in der Normerfüllungstabelle „NORM.TAB“. Die Datei befindet sich im Ordner „Siemens“ unter dem Verzeichnis „Step 7“.

Die Ablaufsprache S7-GRAPH entspricht der in der Norm DIN EN 661131-3 (int. IEC 61131-3) festgelegten Sprache „Sequential Function Chart“.

3. EINFÜHRUNG IN DIE NORM DIN 661131

Das Dokument stellt einen Leitfaden dar, der sich mit der Anwendung von Speicherprogrammierbaren Steuerungen (SPS) und den zugehörigen Peripheriegeräten befasst.
Diese internationale Norm wurde nach den Regeln der Europäischen Gemeinschaft in Deutschland als DIN EN 661131, in Frankreich als NF EN 661131 und in England als BS EN 661131 übernommen.

3.1. DIN EN 661131 TEIL 1, ALLGEMEINE INFORMATION

Die Norm DIN EN 661131 Teil 1 (IEC 61131-1) gilt für Speicherprogrammierbare Steuerungen, die in Niederspannungsanlagen, in denen die Nennspannung des Netzes 1000V AC (50/60Hz) oder 1500V DC nicht übersteigt, zur Steuerung von Maschinen und diskontinuierlichen Prozessen eingesetzt werden. Im Teil 1 werden die Begriffe zur Verwendung in den anderen Teilen der Norm definiert.

3.2. DIN EN 661131 TEIL 2, BETRIEBSMITTELANFORDERUNGEN

Dieser Teil der Norm DIN EN 661131 Teil 2 (IEC 61131-2) spezifiziert

- die elektrischen, mechanischen und funktionellen Anforderungen für Speicherprogrammierbare Steuerungen und deren zugehörigen Peripheriegeräte sowie die anzuwendenden Betriebs, Lager- und Transportbedingungen;
- die Informationen, die vom Hersteller mitgeliefert werden müssen;
- die Prüfmethoden und Verfahrensweisen, die für die Nachprüfung auf Übereinstimmung mit den Anforderungen für Speicherprogrammierbare Steuerungen und die zugehörigen Peripheriegeräte anzuwenden sind.

3.3. DIN EN 661131 TEIL 3, PROGRAMMIERSPRACHEN (IEC 61131-3)

Das Dokument behandelt die Programmiersprachen für die Speicherprogrammierbaren Steuerungen, wie sie in DIN EN 661131-1 definiert sind. Dabei ist zur Darstellung auf Drucker und Bildschirm der ISO 646 Zeichensatz anzuwenden. Graphische und semigrafische Darstellungen der Sprachelemente, die in diesem Teil definiert sind, sind zwar zulässig, aber nicht in diesem Teil definiert.
Es wurden keine neuen Programmiersprachen definiert, sondern die am meisten genutzten Programmiersprachen AWL, KOP, FBS, AS und ST wurden harmonisiert.

3.4. DIN EN 661131 TEIL 4, ANWENDERRICHTLINIEN (BEI IEC IN BEARBEITUNG)

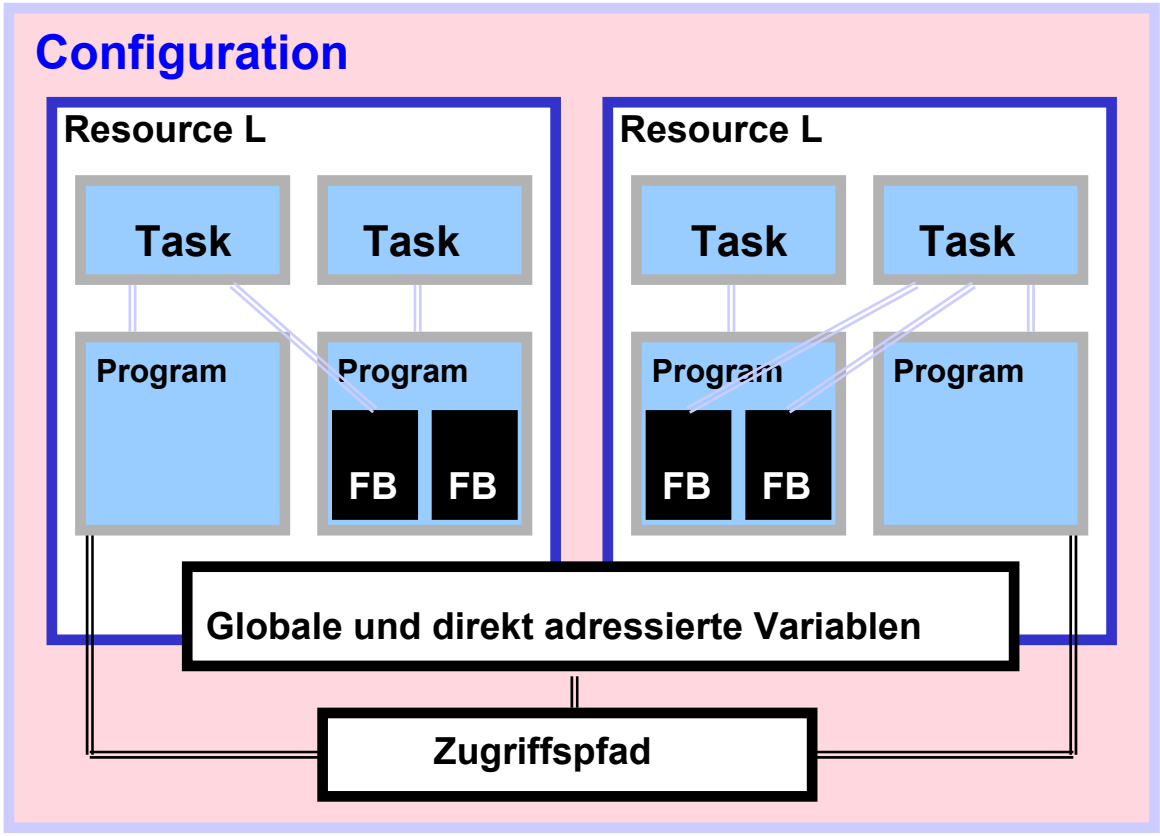
Dieser Teil bearbeitet die Anwenderrichtlinien von SPS-Systemen. Es werden Hinweise für alle Phasen eines Projektes gegeben; beginnend bei der Systemanalyse über die Phasen der Spezifikation, Auswahl der Geräte bis hin zur Anwendung und Wartung der Geräte.

3.5. DIN EN 661131 TEIL 5, KOMMUNIKATION (BEI IEC IN BERABEITUNG)

Dieser Teil bearbeitet die Kommunikation zwischen SPS verschiedener Hersteller und die Kommunikation beliebiger Geräte mit der SPS.
Basierend auf der MAP-Standardisierung werden die Kommunikationsdienste einer SPS als Ergänzungsnorm zu ISO/IEC 9506-1/2 festgelegt. Es werden die Kommunikationsbausteine beschrieben für genormte Lese- und Schreibzugriffe.

4. DIN EN 661131 TEIL 3, PROGRAMMIERSPRACHEN

4.1. DAS IEC 61131-3- SOFTWARE-MODELL



Übersicht	Einer der wichtigsten Punkte in einem Programmiersystem bildet die Möglichkeit, ein größeres Programm in kleine Einheiten zu unterteilen. Diese Einheiten müssen ihre eigenen Daten "kapseln" und dürfen nur über klar definierte Schnittstellen mit anderen Software-Komponenten kommunizieren. Zu diesem Zweck wurde während der Entwicklung der IEC 61131-3 Norm der weiterreichende Kontext eines SPS-Programms mit einbezogen und ein abstraktes Software-Modell mit einzelnen Hierarchieebenen entwickelt.
Configuration= Konfiguration	Die höchste Ebene ist ein PLC-Anwenderprogramm in einer Konfiguration enthalten. Eine Konfiguration entspricht im allgemeinen einer SPS-Steuerung. Reale Automatisierungssysteme sind in der Regel aus mehreren SPSen, d.h. mehreren IEC Konfigurationen aufgebaut. Eine IEC Konfiguration kann deshalb mit anderen IEC Konfigurationen über klar definierte Schnittstellen, die natürlich vorher deklariert werden müssen, kommunizieren.
Resource= Ressource	Innerhalb einer Konfiguration gibt es eine oder mehrere Ressourcen. Eine Ressource liefert die notwendige Unterstützung für die Ausführung von SPS-Anwenderprogrammen. In einem Steuerungssystem entspricht i.a. einer IEC Ressource z.B. eine reale CPU-Baugruppe oder eine virtuelle SPS-CPU auf einem PC (SOFT-PLC).

4.2. PROGRAMME

Ein Programm ist in IEC 61131 definiert als „ eine logische Anordnung von allen Programm-Sprachelementen und –Konstrukten, die für die beabsichtigte Signalverarbeitung zur Steuerung einer Maschine oder eines Prozesses mit einem SPS-System erforderlich sind“.

Ein Programm welches die Norm erfüllt

1. darf nur Eigenschaften benutzen, die in diesem Teil zum Gebrauch der jeweiligen Sprache festgelegt sind,
2. darf keine Eigenschaften benutzen, die als Erweiterungen der Sprache gelten,
3. darf von keiner bestimmten Interpretation implementierungsabhängiger Eigenschaften abhängen.

Ein normgerechtes Programm muss auf jedem normgerechten System die gleichen Ergebnisse liefern.
Ausnahmen bilden nur Programme die Abhängig sind

- vom Zeitverhalten der Programmausführung, der Anwendung
- von implementierungsabhängigen Eigenschaften im Programm
- von der Ausführung von Prozeduren zur Fehlerbehandlung

Auszug aus der Programmdeklaration:

1. Die begrenzenden Schlüsselwörter für die Programmdeklaration müssen PROGRAMM...END-PROGRAMM sein.
2. Ein Programm kann eine Konstruktion VAR_ACCESS...END_VAR enthalten, die ein Mittel zur Festlegung von bezeichneten Variablen bietet, auf die durch einige der Kommunikationsdienste zugegriffen werden kann. Ein Zugriffspfad verknüpft jede derartige Variable mit einem Eingang, Ausgang oder einer internen Variablen eines Programms. Das Format und der Gebrauch müssen der Norm IEC 61131-5 entsprechen.
3. Programme können nur innerhalb von Ressourcen instanziert werden, während Funktionsbausteine nur innerhalb von Programmen instanziert werden können.

Zusammenfassung

Ein normgerechtes Programm

- muss in einer normgerechten Sprache ohne Erweiterungen erstellt sein
- hat über einen Variablendeklarationsteil zu verfügen
- ist durch die Schlüsselwörter PROGRAMM...END-PROGRAMM zu begrenzen
- ist nur innerhalb von Ressourcen ablauffähig
- muss auf jedem normgerechten System die gleichen Ergebnisse liefern, Ausnahmen sind möglich und müssen dokumentiert werden

4.3. VARIABLEN

Eine Variable dient zur Identifizierung von Datenobjekten. Der Inhalt der Variablen ist veränderbar, sie ist verbunden mit Daten, Eingängen, Ausgängen oder Speicherplatz der SPS. Variablen können als elementare oder als abgeleitete Typen deklariert werden.

Darstellung der Einzelelement-Variablen

Eine Einzelelement-Variable ist ein einzelnes Datenelement von einem der elementaren Datentypen, z.B. ein Eingang oder Ausgang. Die direkte Darstellung einer Einzelelement-Variablen erfolgt durch folgende Symbole:

- ein Prozentzeichen
- ein Präfix für den Speicherort
- ein Präfix für die Größe
- eine oder mehrere vorzeichenlose ganze Zahlen

Beispiele:

%QX75 und %Q75	Ausgangsbit 75
%IW215	Eingangswort-Speicherort 15

Eigenschaften der Präfixe

Nr.	Präfix	Bedeutung
1	I	Speicherort Eingang
2	Q	Speicherort Ausgang
3	M	Speicherort Merker
4	X	(Einzel-) Bit-Größe
5	kein	(Einzel-) Bit-Größe
6	B	Byte-(8 Bit) Größe
7	W	Wort-(16 Bit) Größe
8	D	Doppelwort-(32 Bit) Größe
9	L	Langwort-(64 Bit) Größe

Anmerkung: Falls nicht anders deklariert, muss der Datentyp einer direkt adressierten Variablen einer „(Einzel-)Bit"-Größe vom Typ Bool sein.
Nationale Normungsorganisationen können Tabellen mit Übersetzungen der Präfixe herausgeben.

Darstellung der Multielement-Variablen

Die Multielement-Variable besteht aus Feldern (Arrays) und Strukturen (Structures).
Ein Feld ist eine Sammlung von Datenelementen des gleichen Datentyps, die durch einen oder mehrere Feldindizes angesprochen werden. Die Indizes sind in eckigen Klammern eingeschlossen und durch Kommas getrennt.

Beispiel:

OUTARY[%MB6,SYM] := INARY[0] + INARY[7] - INARY[%MB6] * %IW62;

Eine strukturierte Variable ist eine Variable, die als ein Typ deklariert ist, der vorher bereits als eine Datenstruktur festgelegt wurde. Diese Struktur besteht aus einer Sammlung von bezeichneten Elementen. Ein Element einer strukturierten Variable muss durch ein oder mehr Bezeichner oder durch Feldzugriffe dargestellt werden. Die Bezeichner sind durch Punkte getrennt, wobei der erste Bezeichner den Namen des strukturierten Elements darstellt. Die folgenden Bezeichner stellen den Zugriff ein Bestimmtes Datenelement innerhalb der Datenstruktur dar.

Beispiel:

MODULE_5_CONFIG.SIGNAL_TYP := SINGLE_ENDED;

Initialisierung von Variablen

Bei der Initialisierung kann eine Variable verschiedene Werte annehmen:

- den Wert, den Variable hatte, als das Konfigurationselement gestoppt wurde (gepufferter Wert)
- einen anwenderspezifischen Anfangswert
- den voreingestellten Anfangswert für den der Variablen zugeordneten Anfangstyp

Der Anwender kann wählen, ob die Variable gepuffert werden soll indem er das Bestimmungszeichen RETAIN verwendet. Der Anfangswert muss beim Start nach folgenden Regeln ermittelt werden:

- bei einem „Warmstart“ müssen die gepufferten (remanenten) Variablen den gespeicherten Wert annehmen
- bei einem „Kaltstart“ müssen die Variablen den vorgegebenen Wert annehmen oder den spezifischen Anfangswert des zugehörigen Datentyps
- ungepufferte Variablen müssen mit dem vorgegebenen Anfangswert oder mit dem spezifischen Anfangswert des Datentyps initialisiert werden

Deklaration

Jede Typdeklaration einer SPS-Programm-Organisationseinheit muss an seinem Anfang mindestens einen Deklarationsteil enthalten, der die Typen der Variablen festlegt, die in dieser Organisationseinheit verwendet werden. Der Deklarationsteil muss die Textform eines der Schlüsselwörter VAR, VAR_INPUT oder VAR_OUTPUT besitzen. Die Deklaration wird durch VAR_END abgeschlossen.

Beispiel:

Programm Stanze Aufgabe 2

```
VAR
    B1 AT %IX0.0       :BOOL;  (*  Näherungsschalter B1   *)
    B2 AT %IX0.1       :BOOL;  (*  Näherungsschalter B2   *)
    B3 AT %IX0.2       :BOOL;  (*  Näherungsschalter B3   *)
    B4 AT %IX0.3       :BOOL;  (*  Näherungsschalter B4   *)
    Y1 AT %QX4.0       :BOOL;  (*  Zylinder 1.0 ausfahren  *)
END_VAR
```

4.4. DIE ANWEISUNGSLISTE (AWL)

Die Anweisungsliste setzt sich aus einer Abfolge von Anweisungen (Befehlen) zusammen. Folgende Konventionen sind einzuhalten:

- Jede Anweisung muss in einer neuen Zeile beginnen
- Jede Anweisung enthält einen Operator mit zusätzlichen Modifizierern (z.B. I 0.0 für das Eingangsbit 0 im Eingangsbyte 0)
- Mehrere Operanden sind durch Kommas zu trennen (z.B. cal FB1, DB2)
- Der Anweisung kann eine identifizierende Marke vorangehen, der ein Doppelpunkt folgt.
- Ein definierter Kommentar ist das letzte Element einer Zeile
- Leerzeilen können vorhanden sein

Operatoren, Modifizierer und Operanden

Der Wert des Ausdrucks, der berechnet wird, wird durch den aktuell errechneten Wert ersetzt.
Ergebnis := Ergebnis OP Operand

Beispiel: Ergebnis := Ergebnis AND %IX1

Der Modifizierer „N“ zeigt die boolesche Negation des Operanden.
Die Anweisung ANDN%IX2 bedeutet:

Ergebnis := Ergebnis AND NOT %IX2

Der Modifizierer linke Klammer „(“ zeigt an, dass die Auswertung des Operators zurückgestellt wird, bis die rechte Klammer „)“ erscheint.
Die Anweisungsfolge

```
AND( %IX1
OR %IX2
)
```

bedeutet Ergebnis := Ergebnis AND (%IX1 OR %IX2)

Der Modifizierer „C“ zeigt an, dass die zugehörige Anweisung nur erfüllt werden darf, wenn der Wert des gerade ausgewerteten Ergebnisses eine boolesche 1 ist. Ist ein Modifizierer „N“
mit dem Operator verknüpft, so muss das Ergebnis eine boolesche 0 sein.

Operanden der Anweisungsliste

Nr.	Operator	Modifizierer	Operand	Bedeutung
1	LD	N	Anm. 1	Setzt aktuelles Ergebnis dem Operanden gleich
2	ST	N	Anm. 1	Speichert aktuelles Ergebnis auf die Operanden-Adresse
3	S R	Anm. 2 Anm. 2	**BOOL** BOOL	Setzt booleschen Operator auf 1 Setzt booleschen Operator auf 0 zurück
4 5 6 7	AND & OR XOR	N, (N, (N, (N, (	BOOL BOOL BOOL BOOL	Boolesches UND Boolesches UND Boolesches ODER Boolesches Exklusiv-ODER
8 9 10 11	ADD SUB MUL DIV	((((	Anm. 1 Anm. 1 Anm. 1 Anm. 1	Addition Subtraktion Multiplikation Division
12 13 14 15 16 17	GT GE EQ NE LE LT	((((((	Anm. 1 Anm. 1 Anm. 1 Anm. 1 Anm. 1 Anm. 1	Vergleich: > Vergleich: >= Vergleich: = Vergleich: <> Vergleich: <= Vergleich: <
18 19 20	JMP CAL RET	C,N C,N C,N	**MARKE** NAME	Sprung zur Marke Aufruf Funktionsbaustein (Anm. 3) Rücksprung von Funktion oder Funktionsbaustein
21	)			Bearbeitung zurückgestellter Operationen

Anmerkung 1: Die Operatoren müssen entweder überladen oder mit Typ angegeben sein. Das aktuelle Ergebnis und der Operand müssen denselben Typ haben.

Anmerkung 2: Die Operationen erden nur ausgeführt, wenn der Wert des aktuellen Ergebnisses eine boolesche 1 ist.

Anmerkung 4: Dem Namen des Funktionsbausteins folgt eine Liste von Argumenten in Klammern.

Eigenschaften des Funktionsbaustein-Aufrufs

Nr.	Beschreibung/Beispiel
1	CAL mit Liste der Eingangsparameter: CAL C10(CU:=%IX10, PV:=15)
2	CAL mit Laden/Speichern der Eingangsparameter: LD 15 ST C10.PV LD %IX10 ST C10.CU CAL C10
3	Gebrauch der Eingabe-Operatoren: LD 15 PV C10 LD %IX10 CU C10

Anmerkung: Eine Deklaration wie in der Form wie VAR C10: CTU; END_VAR wird in den Beispielen angenommen.

Darstellung der Anweisungsliste in STEP 7

```
FC2 : Beispielbaustein
Netzwerk 1: UND-Verknüpfung in Kontaktplan
      A     I      0.0                "B1"                -- Näherungsschalter B1
      A     I      0.1                "B2"                -- Näherungsschalter B2
      =     Q      4.0                "Y1"                -- Zylinder 1 ausfahren

Netzwerk 2: ODER-Verknüpfung in Kontaktplan
      O     I      0.2                "B3"                -- Näherungsschalter B3
      O     I      0.3                "B4"                -- Näherungsschalter B4
      =     Q      4.0                "Y1"                -- Zylinder 1 ausfahren
```

4.5. KONTAKTPLAN (KOP)

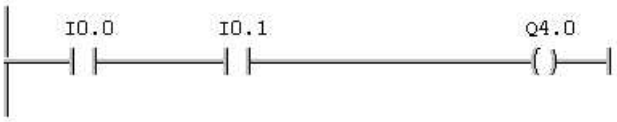
Definition des Netzwerkes

- Ein KOP-Netzwerk wird links und rechts durch jeweils eine vertikale Stromschiene begrenzt
- Ein Verbindungselement wird durch eine horizontale Linie gekennzeichnet.
- Der Zustand des Verbindungselementes wird als „EIN“ und „AUS“ gemäß dem Stromfluss bezeichnet und entspricht den booleschen Werten 0 und 1.
- Der Zustand der linken Verbindungslinie ist immer „EIN“.
- Der Zustand der rechten Verbindungslinie ist undefiniert
- Das horizontale Verbindungselement überträgt den Zustand der unmittelbaren linken Seite auf die unmittelbare rechte Seite des Elements
- Das vertikale Verbindungselement kreuzt sich mit einem oder mehreren horizontalen Elementen. Der Zustand der Vertikalen Verbindung muss „AUS“ sein, wenn alle horizontalen Verbindungen auf der linken Seite des Elements „AUS“ sind. Der Zustand muss „EIN“ sein, wenn einer oder mehrere der Horizontalen Verbindungen auf der linken Seite des Elements „EIN“ sind.
- Der Zustand der vertikalen Verbindung muss auf alle horizontalen Verbindungen der rechten Seite des Elements kopiert werden. Eine Kopie des Zustandes auf die linke Seite ist nicht erlaubt.

Darstellung des Kontaktplans in STEP 7

FC2 : Beispielbaustein

Netzwerk 1: UND-Verknüpfung in Kontaktplan



Symbolinformation:

I0.0	B1	Näherungsschalter B1
I0.1	B2	Näherungsschalter B2
Q4.0	Y1	Zylinder 1 ausfahren

Netzwerk 2: ODER-Verknüpfung in Kontaktplan

Symbolinformation:

I0.2	B3	Näherungsschalter B3
I0.3	B4	Näherungsschalter B4
Q4.0	Y1	Zylinder 1 ausfahren

4.6. DIE FUNKTIONSBAUSTEIN-SPRACHE (FBS)

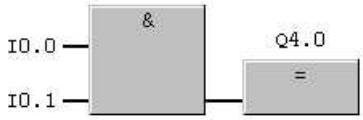
Definition des Netzwerks

- Elemente der Sprache FBS müssen durch Signalfluss-Linien begrenzt werden
- Ausgänge von Funktionsbausteinen dürfen nicht miteinander verbunden werden
- Die Auswertung eines Netzwerkes muss vollendet sein, bevor die Auswertung eines folgenden Netzwerkes beginnen darf, wenn das folgende Netzwerk Ausgänge des vorherigen Netzwerkes benutzt.

Darstellung der FBS in STEP 7

FC2 : Beispielbaustein

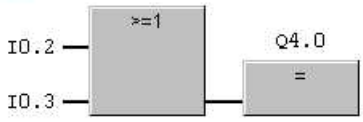
Netzwerk 1 : UND-Verknüpfung in Kontaktplan



Symbolinformation:

I0.0	B1	Näherungsschalter B1
I0.1	B2	Näherungsschalter B2
Q4.0	Y1	Zylinder 1 ausfahren

Netzwerk 2: ODER-Verknüpfung in Kontaktplan



Symbolinformation:

I0.2	B3	Näherungsschalter B3
I0.3	B4	Näherungsschalter B4
Q4.0	Y1	Zylinder 1 ausfahren

4.7. DIE ABLAUFSPRACHE (AS)

Die Elemente der Ablaufsprache bieten Hilfsmittel zur Gliederung eine SPS-Programm-Organisationseinheit in eine Menge von Transitionen und Schritten, die miteinander Verbunden sind. Zu jedem Schritt gehört eine Menge von Aktionen und jede Transition enthält eine Transitionsbedingung. Das erstellte Programm wird in Funktionsbausteinen mit Instanzdatenbaustein gespeichert.

Schritte

Ein Schritte ist graphisch durch einen Block darzustellen. Der Block enthält einen Schrittnamen in Form eines Bezeichners oder in Textform (z.B. Schritt 1 oder S1). Die gerichteten Verbindungen werden durch vertikale Linien dargestellt, welche oben am Schritt angeknüpft ist. Die Verbindungen können auch in Form von Transitionen dargestellt werden.
Zu einem Schritt gehören eine Anzahl verschiedener Aktionen, wie z.B. Setzen oder Rücksetzen von Ein- oder Ausgängen.
Ein Schrittmerker zeigt den Zustand des Schrittes, aktiv oder inaktiv, er kann durch eine boolesche Variable dargestellt werden. Die Variable zeigt den booleschen Wert 1, wenn der Schritt aktiv und eine 0 , wenn der Schritt inaktiv ist.
Die Zeit „Time“ beginnt bei der Aktivierung eines Schrittes mit dem Wert „0“ und stoppt bei der Deaktivierung. Der Zeitwert bleibt bis zur nächsten Aktivierung auf dem erreichten Wert stehen.
Jede Ablaufsteuerung besitzt eine Programm-Organisationseinheit. In ihr werden die Anfangswerte ihrer internen und Ausgangsvariablen abgelegt. Außerdem erfolgt eine Definition der aktiven Anfangsschritte, jedes AS-Netzwerk muss genau einen Anfangsschritt haben.
Die Systeminitialisierung erfolgt über die voreingestellte Anfangszeit für Schritte,
die Aktivierung des nächsten Schrittes muss innerhalb einer vorgegebenen Zeit erfolgen, sonst wechselt das System in den Zustand „Störung“ oder über den Anfangszustand für gewöhnliche Schritte, die Aktivierung erfolgt durch die Erfüllung der Weiterschaltbedingung.

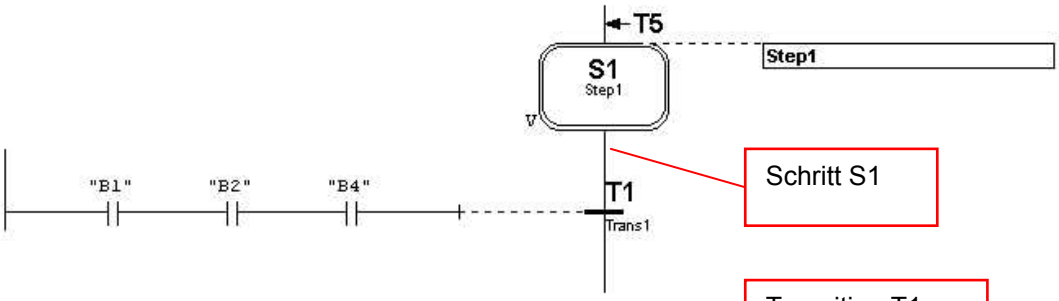
Transitionen

Die Transition wird durch einen horizontalen Strich, welcher durch die vertikale Verbindung zwischen zwei Schritten geht, dargestellt. Eine Transition ist eine Weiterschaltbedingung in der Bedingungen definiert sind, die erfüllt sein müsse, bevor der nächste Schritt aktiv geschaltet werden kann. Sind in einer Transition keine Bedingungen definiert, so gilt diese als erfüllt.
Die Transitionsbedingung kann in der Sprache KOP, ST, AWL oder FBS definiert werden.

Aktionen

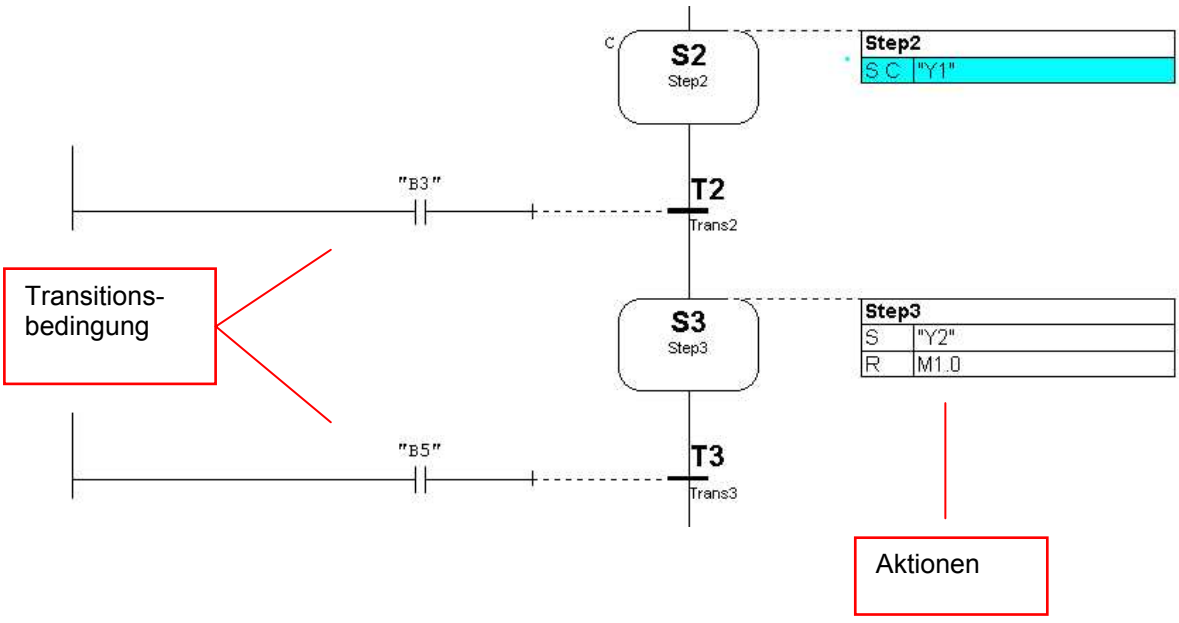
Ein Schritt beinhaltet von 0 bis mehrere Aktionen. Eine Aktion kann eine boolesche Variable oder auch eine Folge von Anweisungen in einer der normgerechten Programmiersprachen sein. Die Funktion der Aktion muss durch Steuerkennzeichen erkennbar sein.

Darstellung einer AS in S7-GRAPH



Adresse	Symbol	Typ	Kommentar
I0.0	B1	BOOL	Näherungsschalter B1
I0.1	B2	BOOL	Näherungsschalter B2
I0.3	B4	BOOL	Näherungsschalter B4
S001.T	Step1.T	DWORD	
T#10S		TIME	

Ausschnitt aus einer Ablaufkette



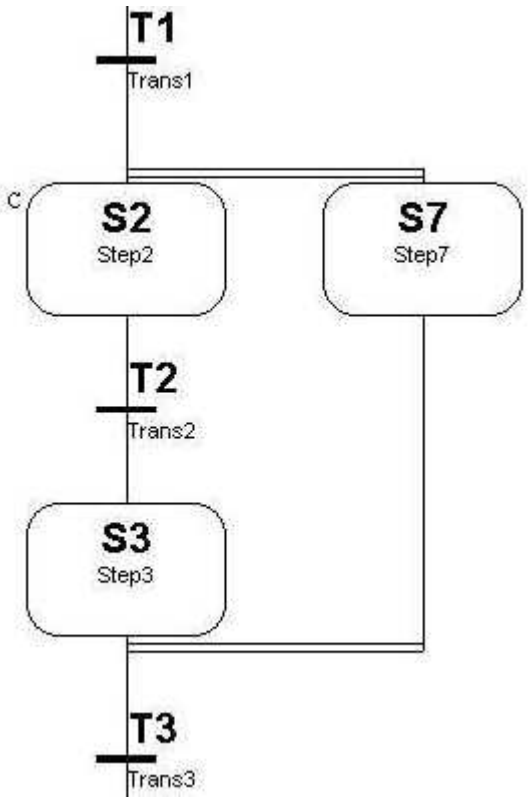
Darstellung der Bestimmungszeichen für Aktionen

Nr.	Bestimmungszeichen	Erläuterung
1	keines	nicht gespeichert (kein Zeichen)
2	N	nicht gespeichert
3	R	vorrangiges Rücksetzen
4	S	Setzen (gespeichert)
5	L	zeitbegrenzt
6	D	zeitverzögert
7	P	Impuls (Flanke)
8	SD	gespeichert und zeitverzögert
9	DS	verzögert und gespeichert
10	SL	gespeichert und zeitbegrenzt

Ablaufregeln

- Der Anfangszustand eines AS-Netzwerkes ist durch einen Anfangsschritt charakterisiert, der bei der ProgramminiT I Alisierung im aktiven Zustand ist.
- Eine wird freigegeben, wenn der vorangehende Schritt und die benötigte Transitionsbedingungen erfüllt sind.
- Das Auslösen einer Transition, deaktiviert alle vorangegangenen Schritte
- Zwei Schritte dürfen nicht direkt verbunden sein, sie sind durch Transitionen zu trennen
- Zwei Transitionen dürfen nicht direkt verbunden sein, sie sind durch einen Schritt zu trennen
- Eine Aktivierung von mehreren Schritten, liegt eine Simultanverzweigung vor, eine Zusammenführung der Simultanketten muss durch eine doppelte horizontale Linie gekennzeichnet werden.

Darstellung einer Simultanverzweigung



4.8. STRUKTURIERTER TEXT (ST)

4.8.1. AUSDRÜCKE

Ausdrücke bestehen aus Operatoren und Operanden. Die Operatoren der Sprache ST sind in der Tabelle 5 dargestellt.

Nr.	Operation	Symbol	Rangfolge
1	Klammerung	(Ausdruck)	höchste
2	Funktionsauswertung Beispiele:	Bezeichner (Argument-Liste) LN(A), MAX(X,Y) usw.	
3	Potenzierung	**	
4 5	Negation Komplement	- NOT	
6 7 8	Multiplikation Division Modulo	* / MOD	
9 10	Addition Subtraktion	+ -	
11	Vergleich	<,>,<=,>=	
12 13	Gleichheit Ungleichheit	= <>	
14 15	Boolesches UND Boolesches UND	& AND	
16	Boolesches Exklusiv ODER	XOR	
17	Boolesches ODER	OR	niedrigste

Die Auswertung eines Ausdrucks besteht aus dem Anwenden der Operatoren auf die Operanden, in der Reihenfolge, die durch die Rangfolge der Operatoren definiert ist. Der Operator mit der höchsten Rangfolge in einem Ausdruck muss zuerst angewendet werden, gefolgt vom Operator der nächst niederen Rangfolge usw., bis die Auswertung vollendet ist. Operatoren mit gleichem Rang müssen angewendet werden, wie sie im Ausdruck von rechts nach links beschrieben sind.
Zum Beispiel, wenn A,B,C und D den Typ INT (Integer) und entsprechend die Werte 1,2,3 und 4 haben, dann muss

A+B-C*ABS(D) als –9 errechnet werden ((1+2)-(3*4)= -9)

und

(A+B-C)*ABS(D) 0 ergeben. ((1+2-3)*4=0)

Wenn ein Operator zwei Operanden hat, muss der linke Operand zuerst ausgewertet werden. Zum Beispiel muss im Ausdruck
SIN(A)*COS(B)
der Ausdruck SIN(A) zuerst ausgewertet werden, gefolgt von COS(B), gefolgt von der Auswertung des Produktes.
Boolesche Ausdrücke brauchen nur in dem Umfang ausgewertet zu werden, der notwendig ist, um das resultierende Ergebnis zu ermitteln.
Wenn zum Beispiel A<=B ist, dann genügt es, nur den Ausdruck (A>B) auszuwerten, um zu entscheiden, dass der Wert des Ausdrucks (A>B)&(C<D) den booleschen Wert 0 hat.
Funktionen müssen als Elemente von Ausdrücken aufgerufen werden, die aus dem Funktionsnamen bestehen, dem eine eingeklammerte Liste von Argumenten folgt.

4.8.2. ANWEISUNGEN

Die Anweisungen der Sprache ST sind in Tabelle 6 zusammengefasst. Anweisungen müssen durch Semikolons abgeschlossen werden.

Nr.	Anweisungstyp/Verweis	Beispiele
1	Zuweisung	A:=B; CV:=CV+1; C:=SIN(X);
2	Funktionsbaustein-Aufruf und Gebrauch des FB-Ausgangs	CMD_TMR(IN:=%IX5, PT:=T#300ms); A:=CMD_TMR.Q;
3	RETURN	RETURN;
4	IF	D:=B*B-4*A*C; IF D< 0.0 THEN NROOTS:=0; ELSEIF D=0.0 THEN NROOTS:= 1; X1:=-B/(2.0*A); ELSE NROOTS:= 2; X1:=(-B+SQRT(D))/(2.0*A); X1:=(-B-SQRT(D))/(2.0*A); END_IF;
5	CASE	TW:=BCD_TO_INT(THUMBWHEEL); TW_ERROR:=0; CASE TW OF 1,5:DISPLAY:=OVEN_TEMP; 2: DISPLAY:=MOTOR_SPEED; 3: DISPLAY:=GROSS_TARE; 4,6..10:DISPLAY:=STATUS(TW-4); ELSE DISPLAY:=0; TW_ERROR:=1; END_CASE; QW100:=INT_TO_BCD(DISPLAY);

6	FOR	J:=101;
		FOR i:=1TO 100BY 2 DO
		IF WORDS[I]='KEY' THEN
		J:=I;
		EXIT;
		END_IF
		END_FOR;

7	WHILE	J:=1; WHILE J<= 100 & WORDS[J] <> ‚KEY' DO J:= J+2; END_WHILE;
8	Wiederholung REPEAT	J:= -1; REPEAT J:= J+2; UNTIL J= 101 OR WORDS[J]= ‚KEY' END_REPEAT;
9	Ausgang EXIT	EXIT;
10	Leer-Anweisung	;

.

4.8.2. ZUWEISUNGEN

Die Zuweisung ersetzt den aktuellen Wert einer Einzel- oder Multielement-Variablen durch das Ergebnis der Auswertung eines Ausdrucks. Eine Zuweisung muss aus einer Variablenangabe auf der linken Seite bestehen, gefolgt von einem Zuweisungsoperator „:=“, gefolgt vom Ausdruck, der auszuwerten ist. Zum Beispiel wird die Anweisung A:=B; benutzt werden, um den einzelnen Datenwert der Variablen A durch den aktuellen Wert der Variablen B zu ersetzen, falls beide den Typ INT haben.

Die Zuweisung muss auch benutzt werden, um den Wert zuzuweisen, der von einer Funktion zurückzugeben ist. Dies geschieht durch Einsetzen des Funktionsnamens auf die linke Seite des Zuweisungsoperators im Rumpf der Funktionsdeklaration. Der Wert, der von der Funktion zurückgegeben wird, muss das Ergebnis der letzten Auswertung einer solchen Funktion sein. Es ist ein Fehler, von der Auswertung einer Funktion mit dem „ENO“-Ausgang ungleich Null zurückzukehren, wenn nicht mindestens eine solche Zuweisung gemacht wurde.

4.8.3. STEUERANWEISUNGEN FÜR FUNKTIONEN UND FUNKTIONSBAUSTEINE

Steueranweisungen für Funktion und Funktionsbaustein bestehen aus Mechanismen zum Aufrufen von Funktionsbausteinen und zum Steuern des Rücksprungs zur aufrufenden Einheit, bevor das physikalische Ende einer Funktion oder eines Funktionsbausteins erreicht ist.
Funktionsbausteine müssen durch eine Anweisung aufgerufen werden, die aus dem Namen des Funktionsbausteins besteht, dem eine eingeklammerte Liste von Wertzuweisungen an bezeichnete Eingangsparameter folgt, wie in Tabelle 6 festgelegt ist. Es ist nicht erforderlich, dass allen Eingangsparametern bei allen Aufrufen eines Funktionsbausteins Werte zugewiesen werden. Falls einem Bestimmten Parameter kein Wert bei einem Funktionsbaustein-Aufruf zugewiesen wird, muss der vorher zugewiesene Wert angewendet werden (oder der Angangswert, falls vorher keine Zuweisung gemacht wurde). Die Anweisung RETURN (Rücksprung) muss ein vorzeitiges Verlassen der Funktion oder eines Funktionsbausteins bewirken.

4.8.4. AUSWAHLANWEISUNGEN

Auswahlanweisungen umfassen die IF- und CASE-Anweisungen. Eine Auswahlanweisung wählt aufgrund einer festgelegten Bedingung eine)oder eine Gruppe) ihrer Anweisungen aus, aus der sie zusammengesetzt sind. Beispiel für Auswahlanweisungen sind in Tabelle 6 angegeben.
Die IF-Anweisung legt fest, dass eine Gruppe von Anweisungen nur ausgeführt wird, wenn der zugehörige boolesche Ausdruck den Wert 1 (wahr) liefert. Falsch die Bedingung falsch ist, darf entweder Keine Anweisung ausgeführt werden oder die Anweisungsgruppe ist auszuführen, die dem SCHLÜSSELWORT ELSE folgt (oder dem Schlüsselwort ELSIF, falls seine zugehörige boolesche Bedingung wahr ist).

Die CASE-Anweisung besteht aus einem Ausdruck, der als eine Variable vom Typ INT ausgewertet werden muss, und einer Liste von Anweisungsgruppen, wobei jede Gruppe mit einer Marke versehen ist, die aus einer oder mehreren ganzen Zahlen oder Bereichen von ganzzahligen Werten besteht. Dies legt fest, dass die erste Gruppe von Anweisungen ausgeführt werden muss, die dem Schlüsselwort ELSE folgt. Andernfalls darf keine Anweisungsfolge ausgeführt werden.

4.8.5. WIEDERHOLUNGSANWEISUNGEN

Wiederholungsanweisungen legen fest, dass eine Gruppe von zugehörigen Anweisungen wiederholt ausgeführt werden muss. Die FOR-Anweisung wird benutzt, falls die Anzahl der Wiederholungen im voraus bestimmt werden kann, andernfalls werden die Konstrukte WHILE und REPEAT benutzt.

Die EXIT-Anweisung muss benutzt werden, um Wiederholungen zu beenden, bevor die Endebedingung erfüllt ist.
Wenn die EXIT-Anweisung innerhalb geschachtelter Wiederholungskonstrukte liegt, muss die innerste Schleife verlassen werden, in der EXIT liegt. Die Steuerung muss zur nächsten Anweisung nach dem ersten Schleifenende übergehen, das der EXIT-Anweisung folgt.

Die FOR-Anweisung zeigt an, dass eine Anweisungsfolge bis zum Schlüsselwort END_FOR wiederholt ausgeführt werden muss, dabei wird der Steuervariablen der FOR-Schleife ein fortschreitender Wert zugewiesen. Die Steuervariable, der Anfangswert und der Endwert müssen Ausdrücke des selben ganz zahligen Datentyps sein und dürfen nicht durch eine der wiederholten Anweisungen verändert werden. Die FOR_Anweisung inkrementiert die Steuervariable von einem Anfangswert auf- oder abwärts zu einem Endwert in Inkrementen, die durch den Wert eines Ausdrucks bestimmt sind, dieser Wert ist auf 1 voreingestellt.
Die Prüfung der Endbedingung wird am Anfang jeder Wiederholung durchgeführt, so dass die Anweisungsfolge nicht ausgeführt wird, falls der Anfangswert den Endwert überschreitet. Der Wert der Steuervariablen nach der Vollendung der FOR-Schleife ist abhängig von der Implementierung.

Die While-Anweisung bewirkt, dass die Folge von Anweisungen bis zum Schlüsselwort END_WHILE wiederholt ausgeführt wird, bis der zugehörige boolesche Ausdruck falsch ist. Falls der Ausdruck von Anfang an falsch ist, wird die Gruppe von Anweisungen überhaupt nicht ausgeführt.

Die REPEAT-Anweisung bewirkt, dass die Folge von Anweisungen bis zum Schlüsselwort UNTIL wiederholt (und mindestens einmal) ausgeführt wird, bis die zugehörige boolesche Bedingung wahr ist.

Die Anweisungen WHILE und REPEAT dürfen nicht benutzt werden, um Synchronisation zwischen Prozessen vorzunehmen, z.B. als eine „Warteschleife“ mit einer extern bestimmten Endebedingung. Zu diesem Zweck müssen die AS-Elemente benutzt werden.
Es muss ein Fehler sein, wenn eine WHILE- oder REPEAT-Anweisung in einem Algorithmus benutzt wird, für den die Erfüllung der Schleifen-Endebedingung oder der Ausführung einer Exit-Anweisung nicht garantiert werden kann

4.9. DOKUMENTATION

Die Dokumentation eines Projektes sollte bestehen aus:

- Beschreibung der Hardware-Konfiguration mit projektspezifischen Bezeichnungen
- Anwendungsprogramm-Dokumentation bestehend aus:
- Ausdruck des Anwendungsprogramms mit möglichen Bezeichnern für die behandelten Signale und Daten
- Querverweislisten für alle bearbeiteten Daten (Ein-/Ausgabe, interne Funktionen wie intern gespeicherte Daten, Zeitgebern, Zählern usw.)
- Kommentare
- Beschreibung der Änderungen
- Wartungshandbuch