Variablen: Bezeichner AT %<IQM><XBWDL>Zahl.Zahl : Typ; (direkte Darstellung)
AT %<IQM><XBWDL>Zahl.Zahl : Typ; (symbolische Darstellung)
I: Eingabe, Q: Ausgabe, M: Merker
Bezeichner : Typ ; (symbolische Variable)

Regeln für die Bildung des Variablenbezeichners:
- Es dürfen nur Buchstaben, Ziffern und der Unterstrich verwendet werden. Umlaute sind nicht erlaubt.
- Das erste Zeichen darf keine Ziffer sein.
- Es dürfen nicht mehrere Unterstriche unmittelbar aufeinander folgen.

Beispiel:
```
VAR
  AT%IW7    :WORD
  AT%QX2.5 :BOOL
  AT%MB4    :BYTE
END_VAR
```

Datentypen:

Datentyp			Bits	Bereich	Initialisierung	Bemerkung
ANY_BIT		BOOL	1	0,1	0	Symbol: X
		BYTE	8	0,...,16#FF	0	Symbol: B
		WORD	16	0,...,16#FFFF	0	Symbol: W
		DWORD	32	0,...,16#FFFFFFFF	0	Symbol: D
		LWORD	64	0,...,16#FFFFFFFFFFFFFFFF	0	Symbol: L
ANY_NUM	ANY_INT	SINT	8	-128,...,127	0	
		INT	16	-32768,...,32767	0	
		DINT	32	-2^{31},...,$2^{31}-1$	0	
		LINT	64	-2^{63},...,$2^{63}-1$	0	
		USINT	8	0,...,255	0	
		UINT	16	0,...,65535	0	
		UDINT	32	0,...,$2^{32}-1$	0	
		ULINT	64	0,...,$2^{64}-1$	0	
	ANY_REAL	REAL	32	$1.18 \cdot 10^{-38}$...$3.40 \cdot 10^{38}$	0.0	
		LREAL	64	-	0.0	
ANY_DATE		DATE	-	-	d#0001-01-01	Datum
		TIME_OF_DAY	-	-	tod#00:00:00	Uhrzeit
		DATE_AND_TIME	-	-	dt#0001-01-01-00:00:00	Datum und Uhrzeit
TIME			-	-	t#0s	Zeitdauer
STRING			-	-	' '	Zeichenfolge

Aufruf von Funktionen: Funktionsname parameter1, parameter2, ...
(Als erster Parameter wird das AE geladen, d.h. *parameter1* ist eigentlich der 2.)

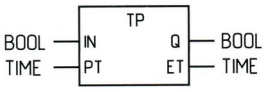
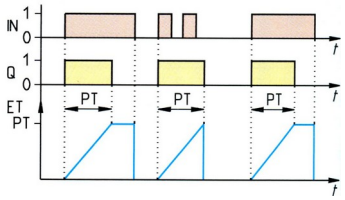
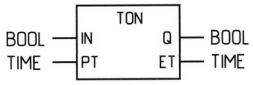
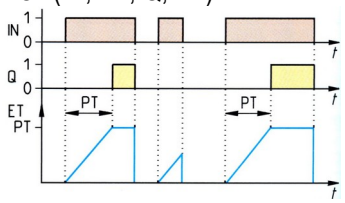
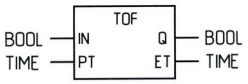
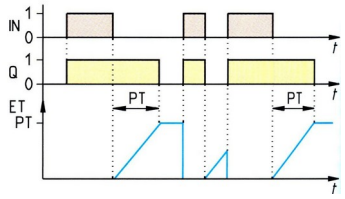
Aufruf von Funktionsbausteinen:
* Liste der Parameter eingeschlossen in runden Klammern oder
* vorausgehendes Laden /Speichern der Eingangsparameter oder
* Benutzung der Eingangsvariablen als Operatoren

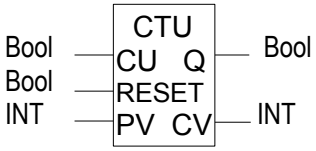
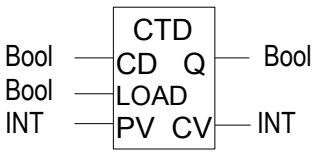
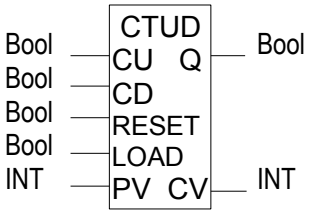
AWL: → Sprungmarke und Kommentar sind optional

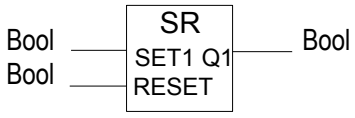
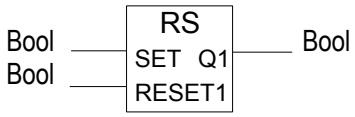
Sprungmarke (Name:)	Operation	Operand	Kommentar (*...*)

Operatoren		Beschreibung
logisch allgemein	(bzw.)	Klammerebene öffnen bzw. schließen
	LD	Aktuelles Ergebnis (AE; current result) mit dem Operanden laden
	ST	STORE, AE im angegebenen Operanden speichern
logisch	LDN	AE mit negierten Operanden laden
	NOT	AE negieren
	AND bzw. ANDN	UND-Verknüpfung des Operanden bzw. negierten Operanden mit dem AE
	OR bzw. ORN	ODER-Verknüpfung des Operanden bzw. negierten Operanden mit dem AE
	XOR bzw. XORN	Exklusiv-ODER-Verknüpfung des Operanden bzw. negierten Operanden mit dem AE
	STN	AE wird negiert im angegebenen Operanden abgespeichert
	S	Wenn AE=”1” so wird der angegebene Operand auf „1” gesetzt
	R	Wenn AE=”1” so wird der angegebene Operand auf „0” gesetzt
allgemein	ADD	Operand wird zum AE addiert
	SUB	Operand wird vom AE subtrahiert
	MUL	Operand wird mit dem AE multipliziert
	DIV	AE wird durch den Operanden dividiert
	GT	AE > Operand?
	GE	AE $\geq$ Operand?
	EQ	AE = Operand?
	NE	AE $\neq$ Operand?
	LE	AE $\leq$ Operand?
	LT	AE < Operand?
Sprung und Aufruf	JMP	unbedingter Sprung
	JMPC	bedingter Sprung wenn AE=”1”
	JMPCN	bedingter Sprung wenn AE=”0”
	CAL	unbedingter Aufruf des genannten Funktionsbausteines
	CALC	bedingter Aufruf des genannten Funktionsbausteines wenn AE=”1”
	CALCN	bedingter Aufruf des genannten Funktionsbausteines wenn AE=”0”
	RET	unbedingter Rücksprung aus einem Funktionsbaustein
	RETC	bedingter Rücksprung aus einem Funktionsbaustein wenn AE=”1”
	RETCN	bedingter Rücksprung aus einem Funktionsbaustein wenn AE=”0”

Standardfunktionen:

<table>
<tr><td colspan="2" rowspan="4">Timer</td><td rowspan="2">Eingangs-variablen</td><td>IN</td><td>Starteingang, Typ BOOL</td></tr>
<tr><td>PT</td><td>Preset Time, Vorgabezeit, Typ TIME</td></tr>
<tr><td rowspan="2">Ausgangs-variablen</td><td>Q</td><td>Typ BOOL, Ist der Timer abgelaufen?</td></tr>
<tr><td>ET</td><td>Effective Time=abgelaufene Zeit, Typ TIME.</td></tr>
<tr><td></td><td>TP</td><td>Timer pulse, Pulsgeber</td><td>FUP:
 
</td><td>TP(IN, PT, Q, ET)
 
</td></tr>
<tr><td></td><td>TON</td><td>Timer on-delay Einschalt-verzögerung</td><td>FUP:
 
</td><td>TON(IN, PT, Q, ET)
 
</td></tr>
<tr><td></td><td>TOF</td><td>Timer off-delay Ausschalt-verzögerung</td><td>FUP:
 
</td><td>TOF(IN, PT, Q, ET)
 
</td></tr>
<tr><td></td><td colspan="2">Beispiel in AWL</td><td colspan="2">AWL (Beispiel):
VAR Timer1: TP; T_ab: TIME; END_VAR
CAL Timer1(IN:=%IX0.1,PT:=T#5s)
LD Timer1.Q
ST %QX1.1
LD Timer1.ET
ST T_ab</td></tr>
<tr><td colspan="2" rowspan="8">Zähler</td><td rowspan="5">Eingangs-variablen</td><td>CU (Count Up)</td><td>Aufwärtszählen bei positiver Flanke, Typ BOOL</td></tr>
<tr><td>CD (Count Down)</td><td>Abwärtszählen bei positiver Flanke, Typ BOOL</td></tr>
<tr><td>RESET</td><td>Rücksetzen (Zählerstand auf 0) , Typ BOOL</td></tr>
<tr><td>LOAD</td><td>Laden des Zählerstandes mit Wert PV, Typ BOOL</td></tr>
<tr><td>PV (Preset Value)</td><td>Vorgabewert mit dem der Zähler geladen wird, Typ INT</td></tr>
<tr><td rowspan="3">Ausgangs-variablen</td><td>CV (Current Value)</td><td>Aktueller Wert (Zählerstand), Typ INT</td></tr>
<tr><td>Q</td><td>Ausgang, Typ BOOL, ist gesetzt, wenn das obere Zählziel PV (beim Aufwärtszähler CTU) bzw. das untere Zählziel (beim Abwärtszähler CTD) erreicht ist.</td></tr>
<tr><td>QU</td><td>Ausgang, Typ BOOL, ist gesetzt, wenn das obere Zählziel PV beim allgemeinen Zähler CTUD erreicht ist.</td></tr>
<tr><td>QD</td><td>Ausgang, Typ BOOL, ist gesetzt, wenn das untere Zählziel 0 beim allgemeinen Zähler CTUD erreicht ist.</td></tr>
</table>

	CTU	Aufwärtszähler	CTU(CU, RESET, PV, Q, CV) 	Wenn RESET=„1" ist, wird die Zählvariable CV mit 0 gesetzt. Wenn an CU eine positive Flanke anliegt, dann wird CV um 1 erhöht, solange CV kleiner als PV ist. Q wird „1" wenn CV größer oder gleich PV ist.
	CTD	Abwärtszähler	CTD(CD, LD, PV, Q, CV) 	Wenn LOAD=„1" ist, wird die Zählvariable CV mit der Obergrenze PV gleichgesetzt. Wenn LOAD=„0" ist und an CD eine positive Flanke anliegt, dann wird CV um 1 erniedrigt, solange CV kleiner als PV ist. Q=„1" wenn CV kleiner oder gleich 0 ist.
	CTUD	Auf-und Abwärtszähler, allgemeiner Zähler	CTUD(CU, CD, RESET, LOAD, PV, QU, QD, CV) 	Wenn RESET=„1" ist, dann wird die Zählvariable CV mit 0 initialisiert. Andernfalls und wenn LOAD=„1" ist, dann wird CV mit PV initialisiert. Wenn RESET=„0" und LOAD= „0" sind und wenn an CU eine positive Flanke anliegt, dann wird CV um 1 erhöht, solange CV keinen Überlauf verursacht. Wenn RESET=„0" und LOAD= „0" sind und wenn an CD eine positive Flanke anliegt, dann wird CV jeweils um 1 erniedrigt, solange CV keinen Unterlauf verursacht. QU liefert „1" wenn CV größer oder gleich PV geworden ist. QD liefert „1" wenn CV kleiner oder gleich 0 geworden ist.
	Beispiel in AWL		VAR Counter1: CTU; END_VAR CAL Counter1(CU:=%IX0.1,RESET:=%IX0.2,PV:=250) LD Counter1.CV ST %MW10 LD Counter1.Q ST %QX1.1	

SR	Bistabilen Funktionsblock dominant setzen Q1 = SR (SET1, RESET) 	Q1 = (NOT RESET AND Q1) OR SET1 Q1, SET1 und RESET sind vom Typ BOOL.
RS	Bistabilen Funktionsblock dominant zurücksetzen; Q1 = RS (SET, RESET1) 	Q1 = NOT RESET1 AND (Q1 OR SET) Q1, SET und RESET1 sind vom Typ BOOL.
	Beispiel in AWL:	VAR FF1: RS; END_VAR CAL FF1(SET:=%IX0.1,RESET1:=%IX0.2) LD FF1.Q1 ST %QX1.1
R_TRIG (F_TRIG)	Erkennen einer positiven (negativen) Flanke 	R_TRIG: Bei einem Wechsel des Eingangssignals CLK von 0 nach 1 wird Q = 1 gesetzt. F_TRIG: Bei einem Wechsel des Eingangssignals CLK von 1 nach 0 wird Q = 1 gesetzt. Bei der nächsten Ausführung wird Q auf 0 zurück gesetzt.

Ablaufsprache AS (früher: Schrittkettenprogrammierung mittels FBS, AWL oder KOP)

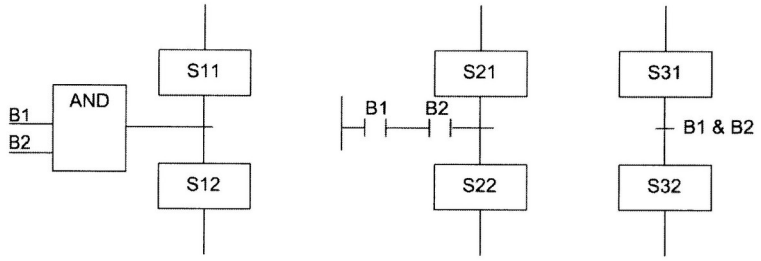
Die grundlegenden Elemente sind Schritte (Anfangs- und Endschritt, normale Schritte, Sprünge) und Transitionen (Weiterschaltbedingungen).
Der Zustand eines Schrittes wird im zugehörigen Schrittmerker gespeichert und ist entweder aktiv oder inaktiv. Die Schrittmerker müssen nicht als Variable deklariert werden, sie werden intern vom System erzeugt. Es kann immer nur ein Schritt aktiv sein. Einem Schritt können ein oder mehrere Aktionsblöcke zugewiesen werden.

Bedeutung der Bestimmungszeichen für die Aktionen des Schrittes:

Zeichen	Bedeutung	Funktionsweise
N	nicht gespeichert	Die Aktion ist solange aktiv wie der Schritt.
S0	Setzen (gespeichert)	Die Aktion wird ausgeführt sobald der Schritt aktiv wird und wird weiter ausgeführt, auch wenn der Schritt schon deaktiviert wurde, bis sie einen Reset erhält.
R0	vorrangiges Rücksetzen	Die Aktion wird deaktiviert.
L	zeitbegrenzt	Die Aktion wird ausgeführt sobald der Schritt aktiv wird und wird solange ausgeführt, bis der Schritt inaktiv wird oder die gegebene Zeitspanne abgelaufen ist.
D	zeitverzögert	Die Ausführung der Aktion startet erst, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist und der Schritt immer noch aktiv ist. Die Aktion wird ausgeführt, bis der Schritt deaktiviert wird.
P	Impuls (Flanke)	Die Aktion wird genau einmal ausgeführt, sobald der Schritt aktiv wird.
SD	gespeichert und zeitverzögert	Die Ausführung der Aktion startet, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist. Sie wird solange ausgeführt, bis sie einen Reset erhält.
DS	zeitverzögert und gespeichert	Die Ausführung der Aktion startet erst, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist und der Schritt immer noch aktiv ist. Die Aktion wird dann ausgeführt, bis sie einen Reset erhält.
SL	gespeichert und zeitbegrenzt	Die Aktion wird ausgeführt, sobald der Schritt aktiviert wird. Sie wird solange ausgeführt, bis die gegebene Zeit abgelaufen ist oder sie einen Reset erhält.
C	bedingt	Der Code der Aktion wird ausgeführt, wenn die angegebene Bedingung erfüllt ist.
F	freigabebedingt	Der Code der Aktion wird ausgeführt, sobald eine Freigabe erfolgt ist.

Transition (Weiterschaltbedingung)

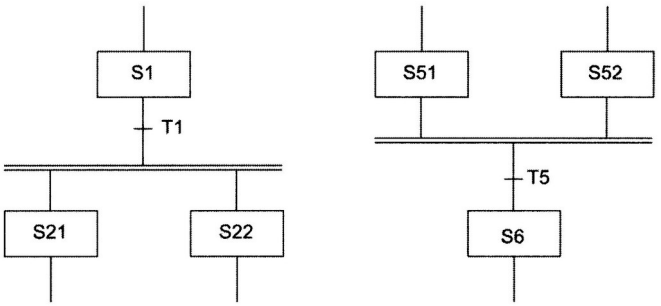
Das Weiterschalten von einem Vorgängerschritt zu einem Nachfolgeschritt läuft nach folgenden Regeln ab:
① Wenn der Vorgängerschritt aktiv ist, wird die Transition freigegeben.
② Bei freigegebener Transition wird die Transitionsbedingung berechnet.
③ Ist die Transitionsbedingung erfüllt, schaltet die Transition zunächst den Vorgängerschritt ab und aktiviert anschließend den Nachfolgeschritt.



Ablaufregeln

Simultanverzweigung

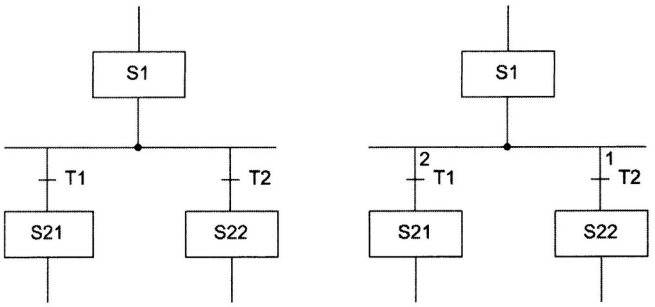
Wenn mehrere Schritte vor einer Transition liegen, so wird die Weiterschaltbedingung erst dann freigegeben, wenn alle Vorgängerschritte aktiv sind.



Kettenauswahl

Die Transitionen regeln den Zugang zu einem Pfad.

Die SPS beginnt mit der Prüfung der Transitionen im linken Zweig (T1). Ist die Bedingung erfüllt, so wird dieser Zweig abgearbeitet. Nur wenn eine Bedingung nicht erfüllt ist, wird die nächste Transition geprüft (T2).



Durch Angabe von Prioritäten kann die Reihenfolge festgelegt werden.

Fehlt der Verzweigungspunkt, so müssen sich die Transitionen gegenseitig ausschließen.