11. Ablaufsteuerungen und Ablaufsprache

11.1 Einführung

Der Ablauf vieler Prozesse - insbesondere der von Fertigungsprozessen - kann determiniert als eine Abfolge eindeutiger Zustände beschrieben werden. Zufallsreaktionen spielen keine Rolle. Es ist bekannt, wann ein bestimmter Zustand des Prozesses auftritt und welche Zustände von dort aus nachfolgend eingenommen werden können. Unter Zustand versteht man dabei sehr allgemein eine Situation im Prozess, die sich hinsichtlich ihrer Eingangs-, Ausgangs- und inneren Signalen auch nur in einem Element (!) von einer anderen Situation abgrenzt.

Weil die Steuerung solcher Abläufe seit jeher eine typische Aufgabe der Automatisierungstechnik ist, wurden dafür der Begriff **„Ablaufsteuerung“** geprägt und effektive Regeln entwickelt. Ergebnis ist die **Ablaufsprache (AS)** als eigene „Sprache“ der Automatisierungstechnik. Die Ablaufsprache ist eine der nach IEC 61131-3 genormten Sprachen und vom Grundsatz her eine graphische Sprache wie FUP. Auch vor der Norm gab es bereits Ablaufsprachen, z.B. Graph 5 im System Simatic S5 (heute vorteilhaft ausgebaut als Graph 7 für Simatic S7).

Selbstverständlich kann man derartige determinierte Abläufe auch ohne Ablaufsprache und letztlich auch ohne die Regeln für Ablaufsteuerungen programmieren und automatisieren. Der Einsatz der Ablaufsprache soll

- Programmentwicklung, Inbetriebnahme, Betrieb und Instandhaltung grundsätzlich erleichtern
- die Programme übersichtlich und de facto „standardisiert“ gestalten.

Diese Effekte können jedoch nur eintreten, wenn die Regeln strikt eingehalten werden. Ziel ist es durchaus, den häufig individuellen Stil von Programmen zu unterbinden!

Auch wenn mitunter gegenteilige Meinungen vertreten werden (häufig weil die AS nicht oder nicht konsequent beherrscht wird), so bietet das Prinzip der Ablaufsteuerung viele Vorteile. Es lohnt sich deshalb vor Bearbeitung einer Automatisierungsaufgabe zu prüfen, ob diese mit den Regeln der Ablaufsteuerung zu lösen ist! Dann ist die eigentliche ingenieurtechnische Aufgabe die Analyse des Prozesses und die Ermittlung aller eintretenden Zustände und ihrer Reihenfolge. Diese sind das Abbild der Funktion der Maschine oder Anlage bei der Ausführung ihrer bestimmungsgemäßen Funktion und weiter auch ihrer Start-, Einricht- und Havariesituationen. Liegt diese Analyse vor, so ist die Erstellung des Programms selbst eine qualifizierte Routinearbeit.

11.2 Die Elemente der Ablaufsprache: Schritte, Transitionen und Aktionen

In der Ablaufsprache werden Zustände im Sinne von Beharrungszuständen als **Schritte** bezeichnet. Da sich solche Schritte linear oder mit Verzweigungen aneinander reihen, entstehen **Schrittketten** (mitunter auch als Ablaufkette bezeichnet). Die Schrittkette ist das Kernstück einer Ablaufsteuerung. Da zwischen den Schritten **gerichtete Verbindungen** bestehen, liegt bereits im Wesen der Schrittkette die eindeutige und zwangsweise Aufeinanderfolge der Schritte begründet, was einen eindeutigen und damit sicheren Ablauf erzwingt. Dies gilt auch dann, wenn unterschiedliche Maschinenzustände von gleichen Ausgangsbedingungen her erforderlich werden. Wichtig ist, dass man auch wirklich alle (!) Zustände – z.B. auch Wartezeiten – als Schritt erfasst!

Die Bedingungen, die das Einschalten eines Schrittes (zeitlich nach einem anderen) erlauben, werden als **Weiterschaltbedingungen oder Transitionen** bezeichnet. Transitionen sind Variablen vom Typ BOOL. Sie werden überwiegend durch logische Verknüpfung aus Signalen des Prozesses oder aus inneren Signalen wie Zeiten, Zählerstände, generierten Fehlersignalen oder auch aus Eingaben des Bedienpersonals abgeleitet. Transitionen bestimmen, ob und wann von einem Schritt auf einen anderen gewechselt wird.

Ein Schritt wird gleichfalls durch eine Boolesche Variable dargestellt, bei Simatic häufig durch ein Merkerbit oder eine statische Variable eines Funktionsbausteins. Es ist klar, dass man mit diesem BOOL allein keine Eingriffe im Prozess vornehmen kann. Deshalb legt die Ablaufsprache fest, dass ein Schritt **Befehle oder Aktionen** „veranlasst“ oder „ausgibt“. Art und Wirkung der Befehle ist in der AS genormt und hinterlegt, und allein das erleichtert die Programmierung der Eingriffe in den Prozess in Größenordnungen!

Kernstück einer Ablaufsteuerung ist die Schrittkette. Deren grundlegenden Elemente sind Schritt und Transition. Als Startpunkt der Kette wird ein Initialschritt festgelegt. Schritte sind mit Aktionen verbunden. Schritte ohne Aktionen wirken wie Warte-Funktionen.

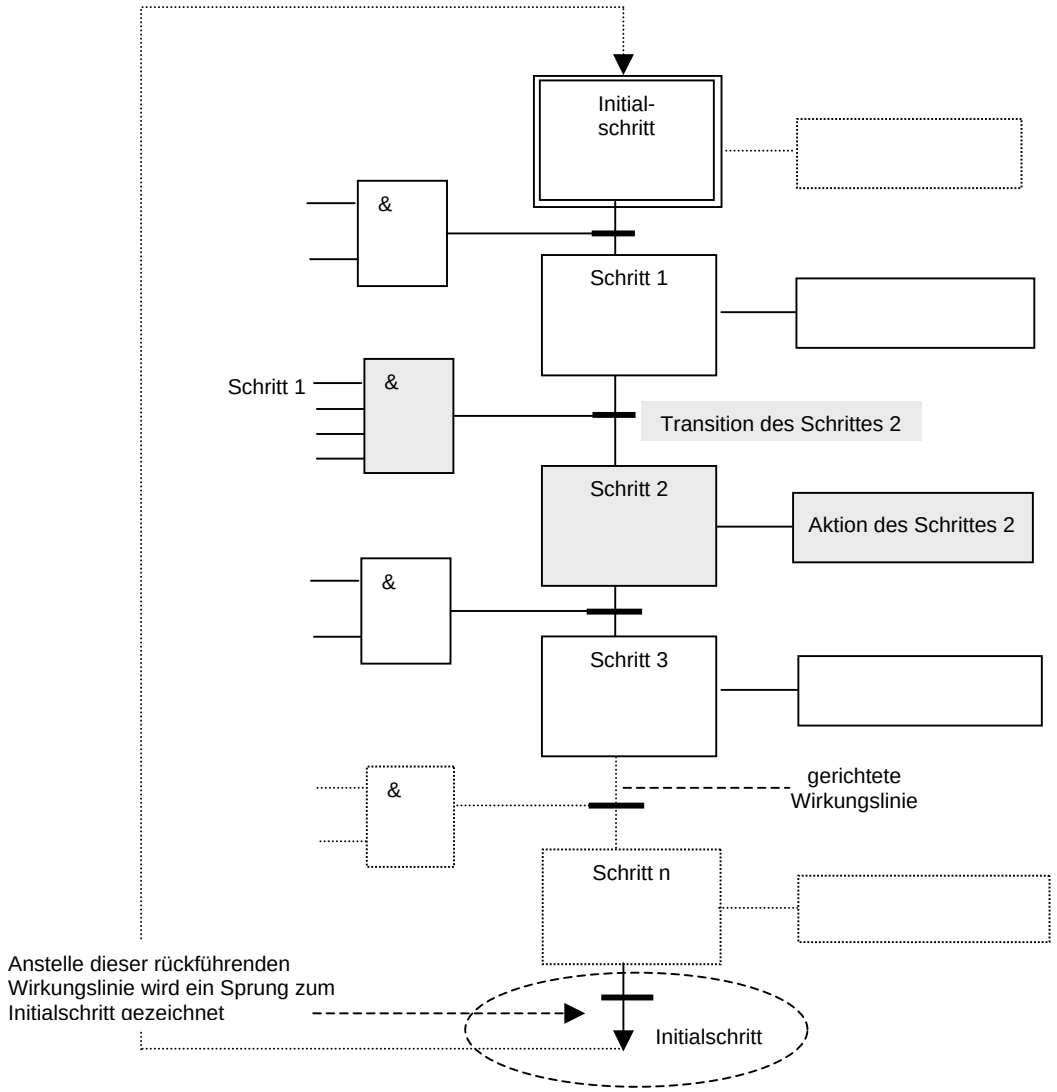
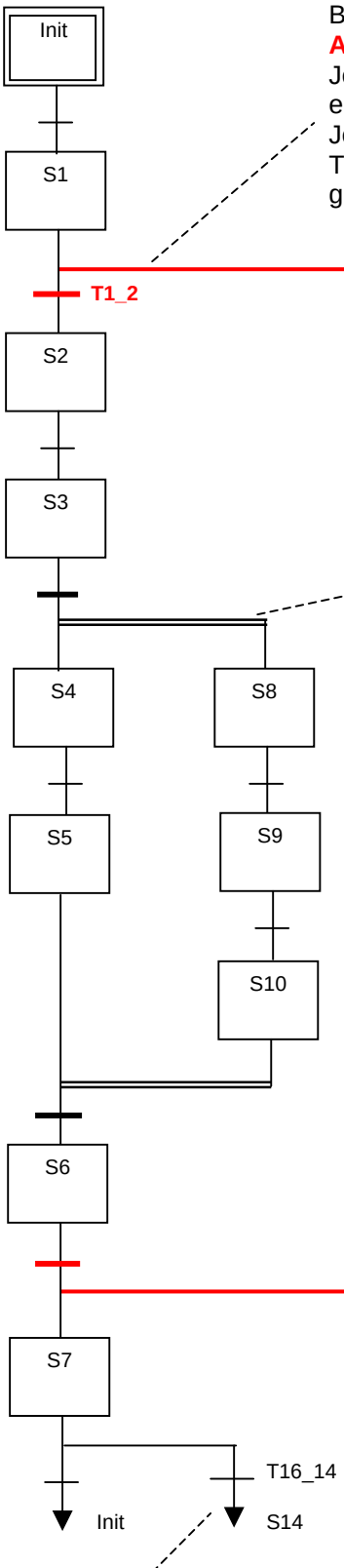


Bild 11-1: Schritt, Transition und Aktion als Elemente der linearen Schrittkette. Bei den Transitionen wurden logische Verknüpfungen angedeutet

Überwiegend werden Schrittketten mit einer Schleife für die Rückkehr zum Initialschritt benötigt. Dies ist im **Bild 11-1** mit dem durch eine Transition bedingten Rücksprung zum Initialschritt dargestellt. Es gibt aber auch Ausnahmen, wo die Kette am Ende verharrt und durch andere Maßnahmen ein erneutes Durchlaufen der Kette veranlasst werden muss

Die lineare Schrittkette ist Kernstück und Grundform der Ablaufsteuerung. **Verzweigungen, Sprünge und Schleifen** in den Schrittketten vervielfältigen die Möglichkeiten der Ablaufsprache und tragen den Anforderungen von Automatisierungsaufgaben Rechnung (**Bild11-2**)



Beispiel einer
Alternativ-Verzweigung (ODER-Verzweigung):
Je nach gültigen Transitionen T1_2 oder T1_11 wird entweder der eine **oder** der andere Zweig durchlaufen, aber stets nur einer!
Jedes Ende der Alternativ-Verzweigung hat seine eigene Transition zum Verlassen der Verzweigung, von denen nur eine (!) gültig sein darf.

Beispiel einer
Simultan-Verzweigung (UND-Verzeigung):
Hier werden die Stränge der Kette parallel bearbeitet. Beide Stränge werden gleichzeitig mit einer (!) Transition aktiviert.
Bei der Zusammenführung der Stränge gibt es nur eine (!) weiterführende Transition.

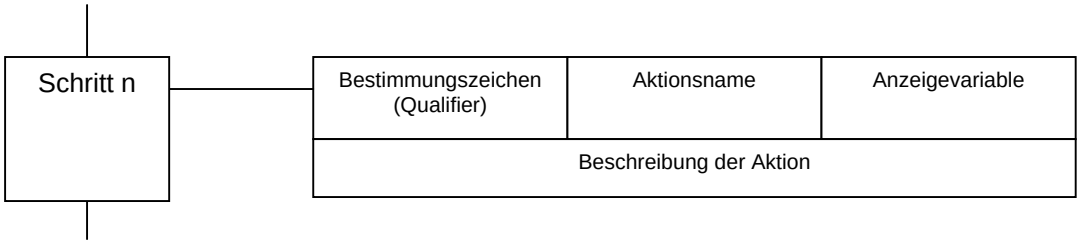
Beispiel einer **Schleife**:
Durch Sprung zu Schritt S11 entsteht eine Schleife, die mehrfach durchlaufen werden kann.
(Die Transitionen T16_11 und T17-7 sollten dann das Signal eines Schleifenzählers auswerten, damit ein Verlassen der Schleife möglich wird.

Beispiel einer **Sprunges** zu einem beliebigen anderen Schritt der Ablaufkette:
Ist die Transition T16_14 TRUE, setzt die Ablaufsteuerung mit Schritt S14 fort.

Bild 11-2: Verzweigungen, Schleifen und Sprünge in Schrittketten

Die Eigenschaften der Aktionen werden im **Aktionsblock** beschrieben. Es sind ausführliche und vereinfachte Darstellungsformen möglich.

Ausführlich kann ein Aktionsblock die im Bild dargestellten Angaben enthalten. Im einfachsten Fall werden nur Bestimmungszeichen und Name benutzt.



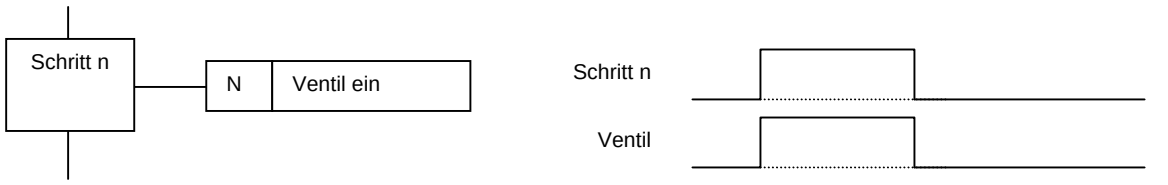
Grundsätzlich wird eine Aktion ausgeführt, wenn der zugehörige Schritt aktiv ist und eine im Hintergrund wirkende **Aktionssteuerung** die Aktion freigibt. Diese Aktionssteuerung ist erforderlich, um die unterschiedlichen Aktionsarten – gekennzeichnet durch das Bestimmungszeichen – umzusetzen. Die Aktionssteuerung ist Bestandteil des Programmiersystems bei Anwendung der Ablaufsprache.

- **Bestimmungszeichen (Qualifier) sind:**

N	Non stored (Nicht gespeichert)	Die Aktion ist solange aktiv wie der Schritt
S	Set (Stored, gespeichert)	Die Aktion wird aktiviert und bleibt bis zu einem Reset aktiv
R	Overriding Reset (zurückgesetzt)	Die Aktion wird deaktiviert
L	Time Limited (zeitbegrenzt)	Die Aktion wird für eine bestimmte Zeit aktiviert, maximal solange der Schritt aktiv ist
D	Time Delayed (zeitverzögert)	Die Aktion wird nach einer bestimmten Zeit aktiv, sofern der Schritt noch aktiv ist und dann solange der Schritt aktiv ist
P	Pulse (Impuls)	Die Aktion wird genau einmal ausgeführt, wenn der Schritt aktiv wird
SD	Stored and Time Delayed (gespeichert und verzögert)	Die Aktion wird nach einer bestimmten Zeit aktiviert und bleibt bis zu einem Reset aktiv
SL	Stored and Limited (gespeichert und zeitbegrenzt)	Die Aktion ist für eine bestimmte Zeit aktiviert
DS	Delayed and Stored (verzögert und gespeichert)	Die Aktion wird nach einer bestimmten Zeit aktiviert, sofern der Schritt noch aktiv ist und bleibt bis zu einem Reset aktiv

Die Bestimmungszeichen L, D, SD, DS und SL benötigen eine Zeitangabe im Format TIME, z.B. L T#5s oder P T#2s300ms.

- **Beispiele für das Wirken der Aktionsteuerung**



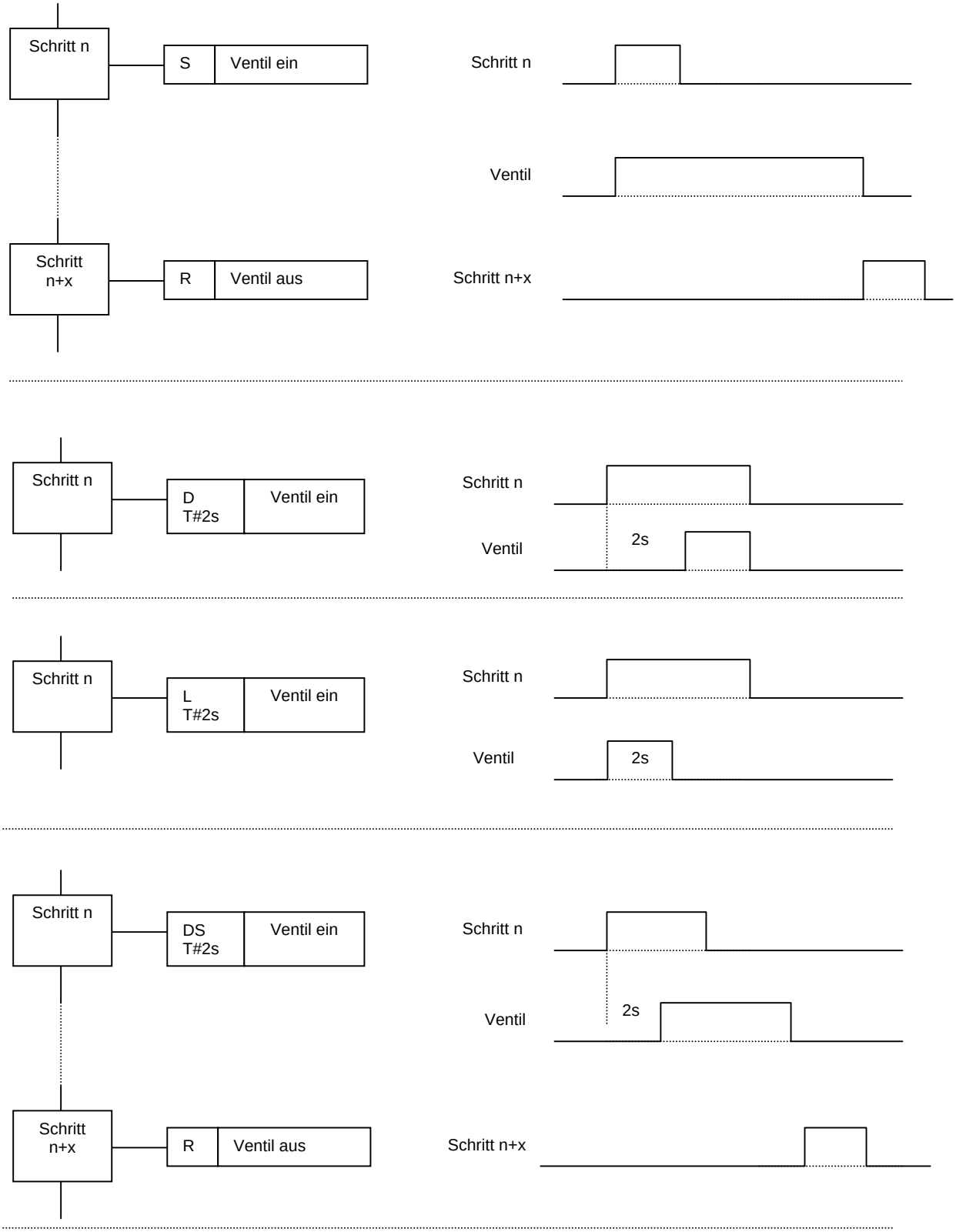


Bild 11-3: Beispiele für spezielle Aktionen

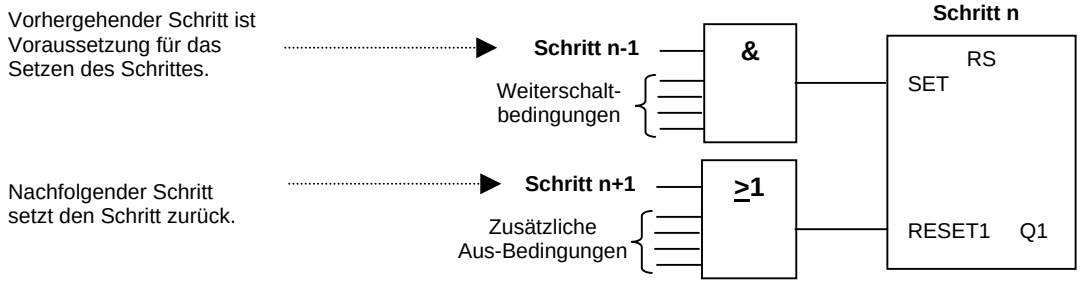
11.3 Grundregeln der linearen Schrittkette

- Setzen eines Schrittes bedeutet, dass dieser aktiv geschaltet wird, so dass seine Aktionen ausgeführt werden. Die hinterlegte Boolesche Variable des Schrittes hat dann den Wert TRUE.
- Rücksetzen eines Schrittes bedeutet, dass dieser inaktiv geschaltet wird und keine seiner Aktionen ausgeführt wird. Die hinterlegte Boolesche Variable des Schrittes hat dann den Wert FALSE.
- Bei der Aktivierung der Schrittkette wird der Initialschritt und nur dieser gesetzt.
- Alle anderen Schritte werden gesetzt, wenn der vorhergehende Schritt aktiv **und** die Weiterschaltbedingungen erfüllt sind.
- Ein Schritt wird zurückgesetzt, wenn der nachfolgende Schritt gesetzt wird.
- Die Einbindung der Schrittkette in das Gesamtprogramm erfordert in den meisten Fällen weitere Maßnahmen für das Rücksetzen von Schritten wie z.B. Gefahren-Aus u.ä.

11.4 Klassische Programmierung von Schrittketten

Schrittketten können durch Ausführung der Grundregeln auch ohne spezielle AS programmiert werden.

- Der Schritt wird durch ein RS-Flip-Flop realisiert, als dessen Instanz z.B. die Schritt-Nr. eingeführt wird.
- Die Weiterschaltbedingungen werden durch logische Verknüpfungen realisiert.




```
LD Schritt n-1
AND .....
AND .....        Weiterschaltbedingungen
AND .....
S Schritt n

LD       Schritt n+1
OR .....
OR .....         zusätzliche Aus-Bedingungen
OR .....
R  Schritt n
```

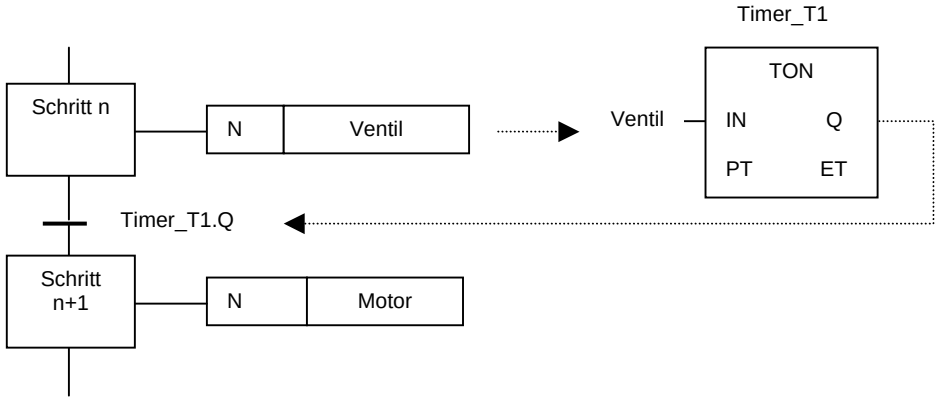
- Das RS-Flip-Flop des Initialschrittes muss beim Einschalten der Steuerung (z.B. Übergang STOP-> RUN) durch einen Richtimpuls (1-Impuls) gesetzt werden.

```
z.B.
LDN Variable_1
ST Richtimpuls
S Variable_1
```

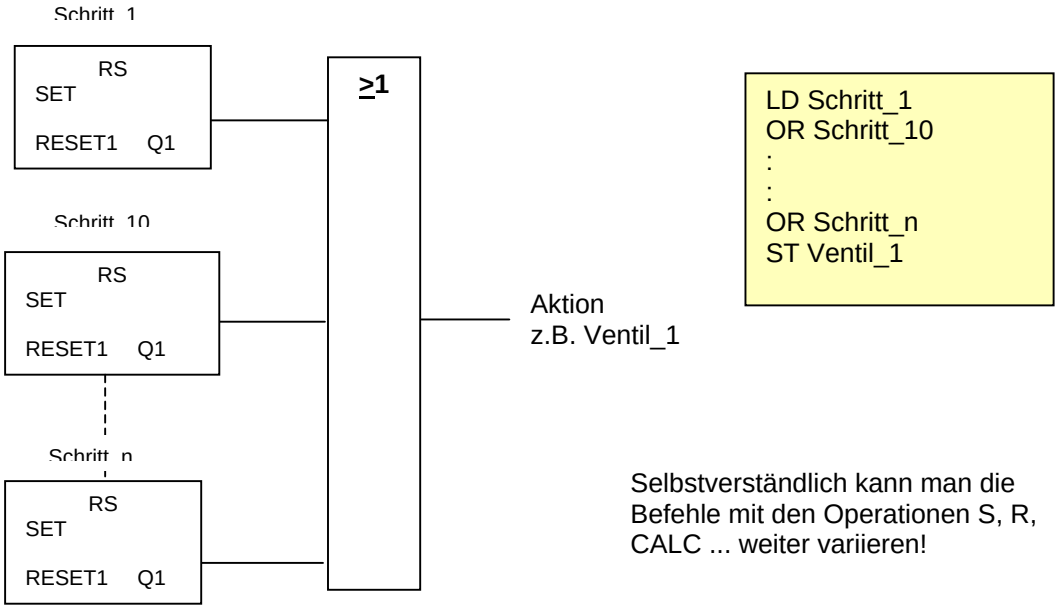
- Das zeitgesteuerte Weiterschalten von Schritten bzw. die limitierte Wirkung von Aktionen kann mit Einschaltverzögerungen TON realisiert werden:

Beispiel: Ein Ventil soll 2s lang eingeschaltet sein.

Indem Schritt n den Timer_T1 (TON) ansteuert und dessen Ausgangssignal Q nach 2s auf TRUE schaltet, wird die Weiterschaltbedingung zum nachfolgenden Schritt genau 2s nach Einschaltung von Schritt n erfüllt. Damit ist Schritt n genau 2s lang aktiv.



- Die Aktionen werden in einfachster Weise programmiert, indem man Zuweisungen mit allen für eine Aktion wirksamen Schrittmerkern in **ODER** (!)-Verknüpfung programmiert. (Es ist darauf zu achten, dass es sich hier nicht wie in der Umgangssprache üblicherweise ausgedrückt um eine UND-Verknüpfung handelt!!!)



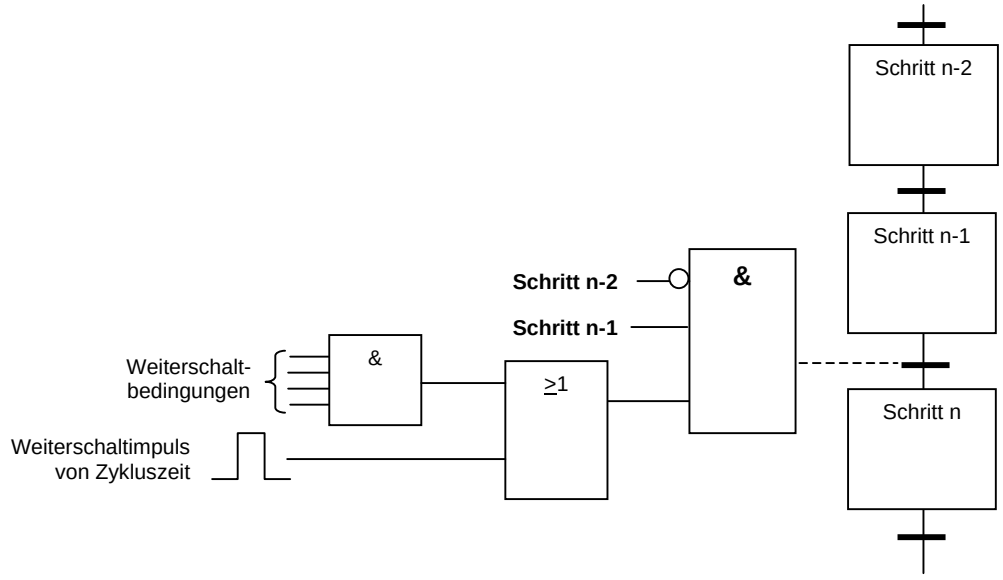
- Mit herkömmlichen Methoden programmierte Ablaufsteuerungen lassen sich auch mit **Betriebsartensignalen** ergänzen. Diese Signale nehmen Einfluss auf Initialisierung und Transitionen.

So kann man z.B. ein Steuersignal generieren, welches den **Initialschritt setzt und alle anderen Schritte zurücksetzt**.

Eine andere Ergänzung wäre eine Steuersignal, welches alle **Transitionen freigibt oder sperrt**. Nur bei Freigabe wäre dann ein Weiterschalten der Schritte gemäß erfüllter Transitionen möglich. Umgekehrt könnte man damit das Weiterschalten unter bestimmten Bedingungen auch sperren.

Mit Steuersignalen kann man auch bestimmte **Schritte priorisieren** und unabhängig von Transitionen einschalten.

Steuersignale können auch weiter **Einzelschrittbetrieb mit oder ohne Berücksichtigung der Transitionen** ermöglichen. Eine solche Betriebsweise ist besonders **beim Einrichten und beim Freifahren nach Havarien** interessant. Bleiben dabei die Transitionen unberücksichtigt, muss ein Weiterschaltsignal als 1-Impuls generiert werden. Zusätzlich sollte dann nicht nur der vorhergehende Schritt n-1 abgefragt werden, sondern auch der bereits zurückgesetzte Schritt n-2. Andernfalls würde die Schrittkette in nur einem Zyklus vollständig „durchlaufen“ werden (siehe nachfolgenden Auszug aus dem FUP einer solchen Lösung). Es muss Sicherheit geschaffen werden, dass der Weiterschaltimpuls das Weiterschalten um nur einen (!) Schritt bewirkt!



11.5 Anwendung der Ablaufsprache (AS)

Vorteilhafter als solche selbst erstellten Programme ist die Anwendung der speziellen Ablaufsprache. Diese ist eine grafisch orientierte Sprache mit Schrittelementen, Transitionen und zugeordneten (= assoziierten) Aktionen. In CoDeSys ist die Anwendung der Ablaufsprache in POE’s vom Typ PROGRAM oder FB möglich. Im System Simatic S7 werden Ablaufprobleme mit Graph 7 gelöst (Vorläufer: Graph 5). Dort kann die Ablaufsprache nur in Funktionsbausteinen angewendet werden. Nachfolgend werden zunächst einige Details der AS im Programmiersystem CoDeSys gezeigt.

11.5.1 Ablaufsprache mit einfachen Schritten

Im System CoDeSys ist die Programmierung von Schritten auf zwei Arten möglich:
Umschaltbar mit sogenannten **einfachen Schritten** oder mit Schritten in der Darstellung der Norm IEC 61131 (genannt **IEC-Schritte**).

Bei Start der Programmierung einer POE in der Sprache AS wird eine Grundstruktur mit Initialschritt und einer ersten Transition sowie dem Rücksprung zum Initialschritt angelegt (**Bild 11-4)**. Hier können nun weitere Schritt-Transition-Kombinationen eingefügt werden. Bei verzweigten Ketten werden parallele und alternative Zweige eingetragen.

Bei einfachen Schritte ist ein Flag in Format eines BOOL hinterlegt. Es trägt den gleichen Namen wie der Schritt selbst und zeigt an, ob der Schritt aktiv ist.
Ist z.B. wie im **Bild 11-5** der Name eines Schrittes „Warten“, so kann man mit „LD Warten“ dieses Flag für die Programmierung einer Aktion direkt abfragen. Ist zu einem Schritt eine Aktion implementiert, so erscheint ein kleines Dreieck in der rechten oberen Ecke des Schrittkästchens.

Genauso erscheint dieses Zeichen bei einer Transition, wenn diese nicht allein aus einer Variablen besteht, sondern ein Programm wie z.B. eine logische Verknüpfung hinterlegt ist. Dieser Fall liegt im Bild 11-5 für die Transition „Start“ vor. Transitionen können in beliebiger Sprache außer AS programmiert werden. Eine Transitionsbedingung muss den Wert TRUE oder FALSE haben. Deshalb sind alle Programmelemente denkbar, die zu einer Boolschen Variablen führen. Transitionen dürfen aber keine Programme, Funktionsbausteine und auch **keine Zuweisungen** (!) enthalten. Letzteres ist bei der Programmierung von Transitionen besonders zu beachten und bedeutet, dass allein die logischen Anweisungen **ohne Abschluss mit dem Befehl STORE** (ST) zu schreiben sind.

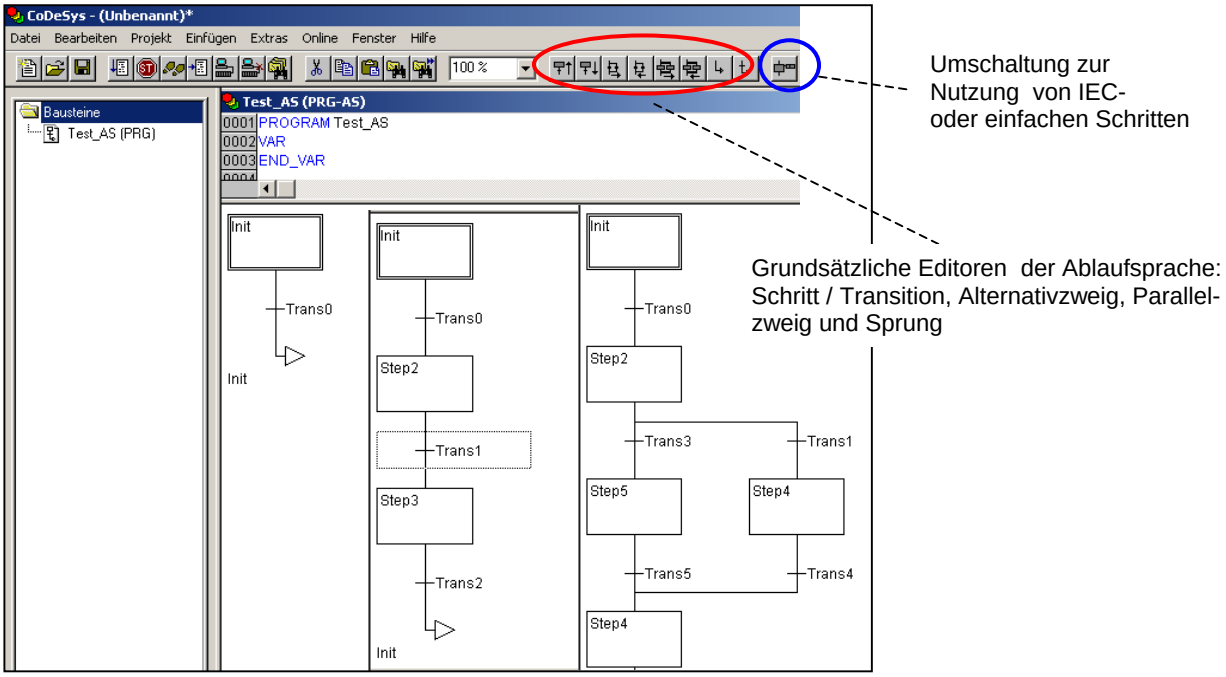


Bild 11-4: Anlegen einer Schrittkette in System CoDeSys
Links nach Anlegen der POE, in der Mitte nach Einfügen zweier weiterer Schritt-Transition-Kombinationen und rechts nach Anlegen einer alternativen Verzweigung.

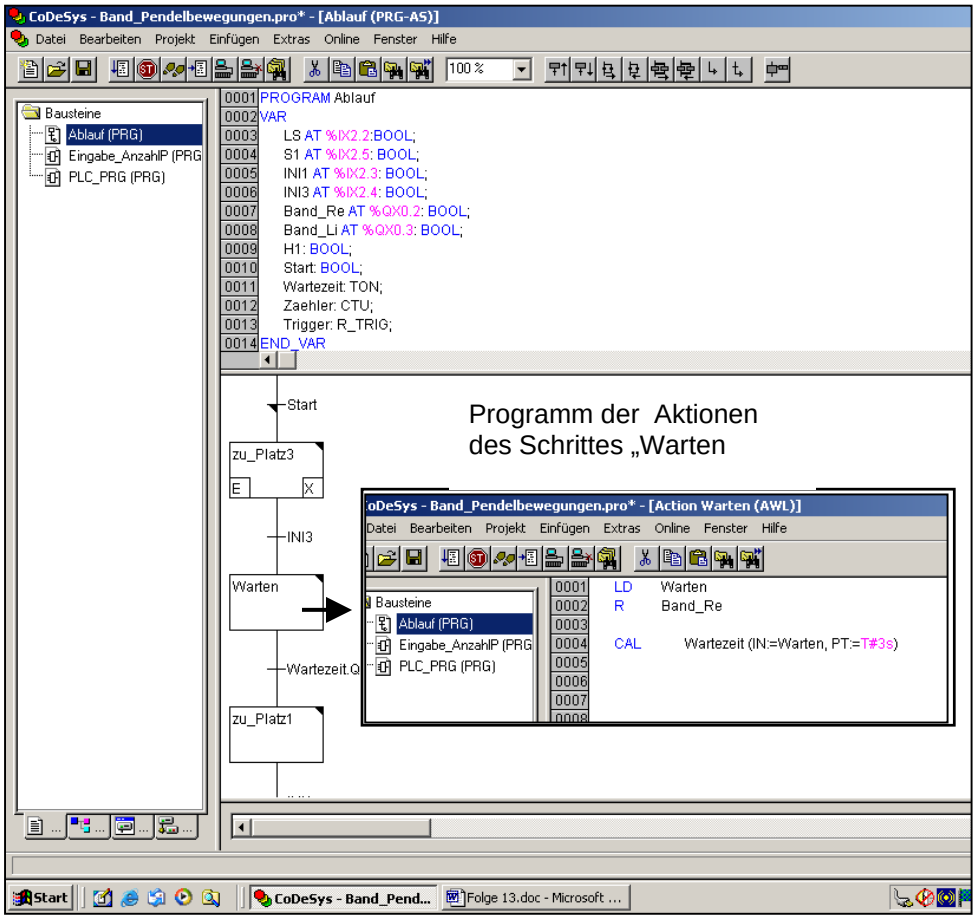


Bild 11-5: AS mit einfachen Schritten und hinterlegten Aktionen in AWL

Aktionen werden programmiert, indem man den Schrittnamen lädt und damit Operationen wie Setzen oder Rücksetzen oder den Aufruf von Standard FB, nicht aber Zuweisen durch STORE einleitet. Bild 11-5 zeigt beispielhaft das hinterlegte Programm des Schrittes „*Warten*“. Wird der Schritt aktiv, so wird die Variable „*Band_Re*“ zurückgesetzt und der Timer „*Wartezeit*“ vom Typ TON gestartet. Nach Ablauf der programmierten Zeit schaltet dessen Boolescher Ausgang in Form der Variablen „*Wartezeit.Q*“ auf TRUE. Diese wirkt als Transition und aktiviert den nachfolgenden Schritt ein. Im Bild 11-5 wird weiter ersichtlich, dass sowohl Transitionen und selbstverständlich alle Instanzen von Standard-FB als Variablen zu deklarieren sind, nicht aber die Namen der Schritte.

Zusätzlich zur eigentlichen Aktion kann jedem Schritt **optional eine Eingangsaktion und/oder eine Ausgangsaktion** hinzugefügt werden. Ein Schritt mit Eingangsaktion wird durch ein 'E' in der linken unteren Ecke gekennzeichnet, die Ausgangsaktion durch ein 'X' in der rechten unteren Ecke. Im Bild 11-5 trägt Schritt „*zu*_Platz3“ solche zusätzlichen Aktionen.
Eine Eingangsaktion wird nur einmal ausgeführt, gleich nachdem der Schritt aktiv geworden ist. Eine Ausgangsaktion wird nur einmal ausgeführt, bevor der Schritt deaktiviert wird. Ein- und Ausgangs-aktionen werden also nicht zyklisch, sondern nur in einem einzigen Zyklus ausgeführt. Das erleichtert vielfältige Sonderaktionen.
Ein treffendes Beispiel ist die Füllung eines Flüssigkeitsbehälters auf ein vorgegebenes Niveau mittels schaltbarem Ventil und Füllstandssensor: Die (nichtzyklische) Eingangsaktion des entsprechenden Schrittes „Füllen“ öffnet das Ventil. In der Aktion selbst wird zyklisch der Füllstand abgefragt. Die (nichtzyklische) Ausgangsaktion schließt das Ventil vor dem Weiterschalten zum nächsten Schritt.

Ein- und Ausgangsaktionen können wiederum in einer beliebigen Sprache implementiert werden.

- **Beispiel Ablaufsteuerung Verteilerband**

Nachfolgend soll eine Ablaufsteuerung mit einfachen Schrittketten programmiert werden. Die beispielhafte Aufgabenstellung für ein spezielles Transportband zeigt **Bild 11-6**:

> Am Platz 1 aufgelegte Teile sollen nach Betätigung des Tasters S1 zum Platz 3 transportiert werden. Sie verharren dort 3s und laufen dann zurück zum Platz 1. Von dieser Position aus sind eine wählbare Anzahl Pendelbewegungen zum Platz 3 und zurück auszuführen. Ist die Zahl erreicht, werden die Teile zur Lichtschranke abtransportiert.
> Die LED Bereitschaft soll während des Initialschrittes leuchten. Die Anzahl der Pendelbewegungen soll mit einem virtuellen Taster zwischen den Werten 1..10 eingegeben werden.

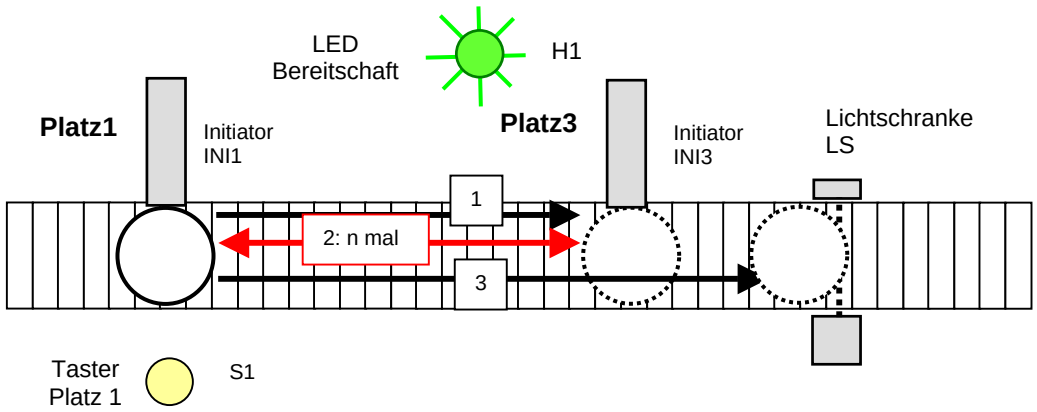


Bild 11-6: Technologieschema zur Aufgabenstellung „Ablaufsteuerung Bandanlage“

Die Lösung beginnt mit dem Festlegen der erforderlichen Schritte und Transitionen nach **Bild 11-7**. Schrittnamen können so vergeben werden, dass die Aktionen der Schritte leicht erkennbar sind. Die Pendelschritte sind zur Verdeutlichung rot gekennzeichnet.

Neben der Schrittkette sind im Bild 11-7 die hinterlegten Anweisungen für Transitionen und Aktionen aufgeführt. Die Weiterschaltbedingungen sind in den meisten Fällen Sensorsignale oder Ergebnis einfachster logischer Verknüpfungen.

Die Zahl der Pendelungen wird durch eine Variable „*Anzahl:INT*“ bestimmt. Um ohne zusätzliche Hardware eine Zahl zwischen 0 und 10 vorgeben zu können, kann man eine Visualisierung nach **Bild 11-8** erstellen. Im Hintergrund läuft das Programm „*Eingabe_AnzahlP (PRG)*“ mit einem zyklischen Ringzähler von 0..10. Dessen aktueller Wert zeigt das Feld „Vorgabe“. Mit einem Sprungbefehl JMPC zum Programmende wird bewirkt, dass nur bei Mausklick auf den Taster „*Eingabe*“ dessen Variable auf TRUE geschaltet und der aktuelle Zaehlerwert in die Variable „*Anzahl*“ geschrieben wird.

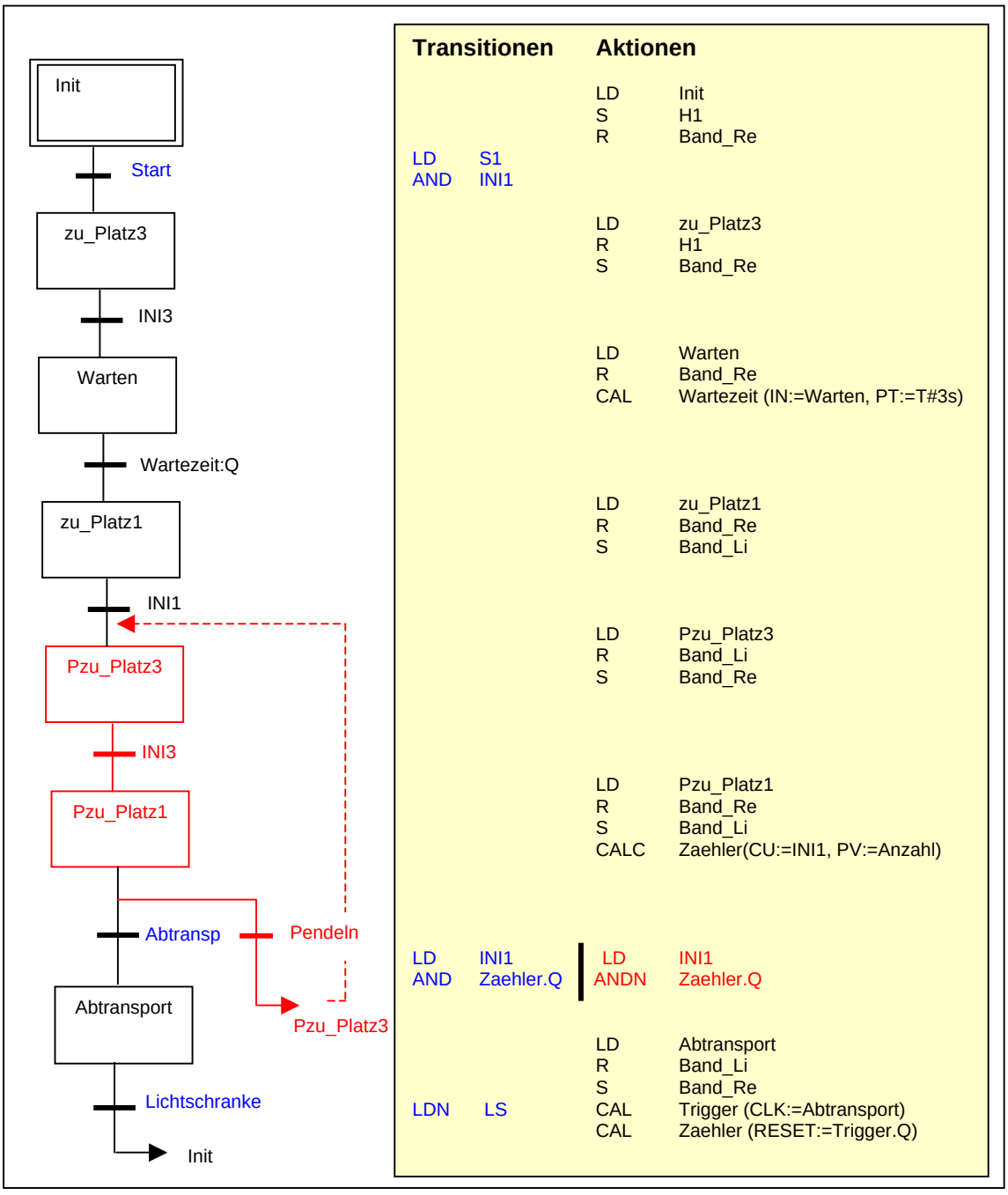
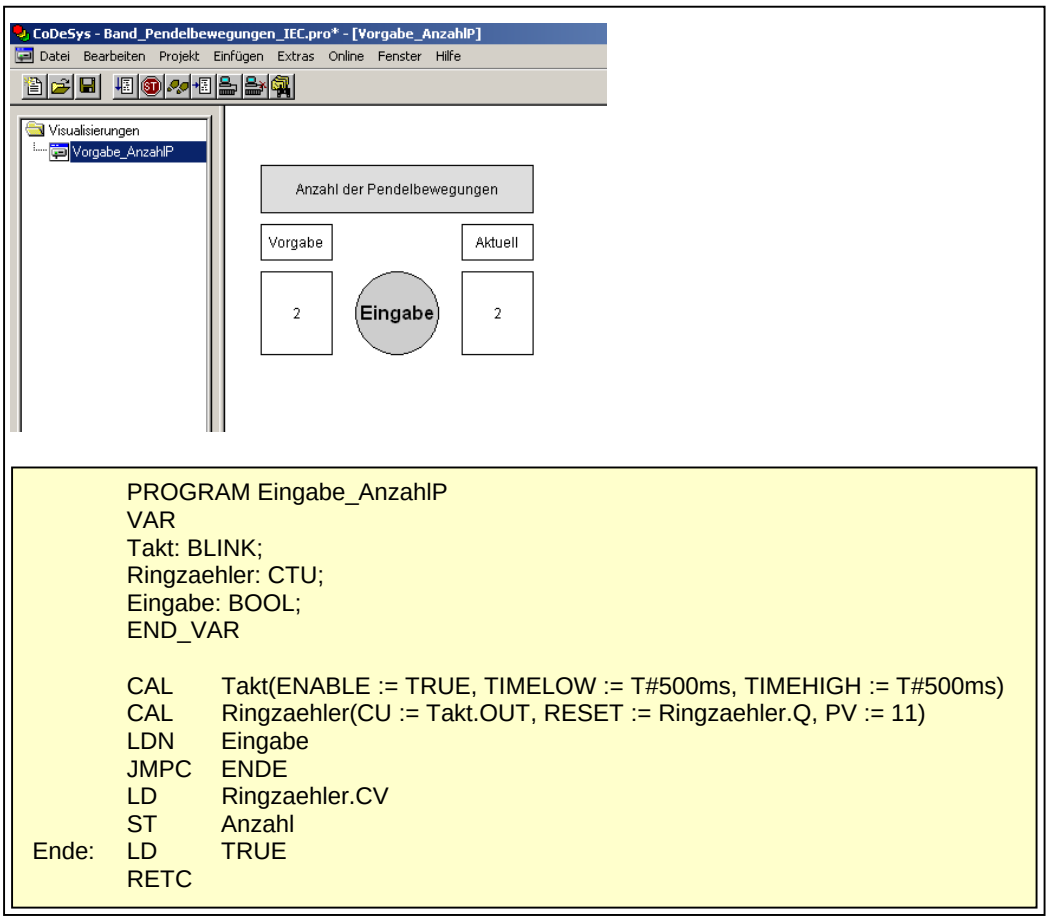


Bild 11-7: Ablaufsteuerung des Verteilerbandes mit zugehörigen Transitionen und Aktionen




```
        PROGRAM Eingabe_AnzahlP
        VAR
        Takt: BLINK;
        Ringzaehler: CTU;
        Eingabe: BOOL;
        END_VAR

        CAL     Takt(ENABLE := TRUE, TIMELOW := T#500ms, TIMEHIGH := T#500ms)
        CAL     Ringzaehler(CU := Takt.OUT, RESET := Ringzaehler.Q, PV := 11)
        LDN     Eingabe
        JMPC    ENDE
        LD      Ringzaehler.CV
        ST      Anzahl
Ende:   LD      TRUE
        RETC
```

Bild 11-8: Hilfsprogramm und integrierte Visualisierung zur Vorgabe einer Anzahl Pendelungen

11.5.2 Ablaufsprache mit IEC-Schritten

Noch effektiver wird die Ablaufsprache bei Nutzung sogenannter IEC-Schritte. Um diese im System CoDeSys verwenden zu können, muss die spezielle **SFC-Bibliothek "*lecsfc.lib*"** in das Projekt eingebunden werden. Weiter muss diese Darstellung im Editor unter ->Extras -> IEC Schritte benutzen aktiviert werden.

Bild 11-9 zeigt einen Abschnitt einer solchen Schrittkette im Online-Modus. Normkonforme IEC-Schritte bestehen aus einem Flag und bis zu neun assoziierten Aktionen. Diese können Boolsche Variablen oder aber auch Programme sein, deren Ergebnis eine Boolsche Variable ist. Mit den Bestimmungszeichen (Qualifier) nach Abschnitt 11.2 wird die Art der Aktivierung und Deaktivierung der Aktionen gesteuert. Ein Großteil der Aktionen sind allein durch Festlegung der Variablen und des Qualifier wirksam.

IEC-Aktionen sind nicht wie bei den einfachen Schritten fest einem Schritt zugeordnet. Sie liegen vielmehr getrennt von den Schritten vor und werden in der Baustein-Übersicht direkt unter dem AS-Baustein angeordnet. Innerhalb des Bausteins können sie dann mehrfach bei unterschiedlichen Schritten verwendet werden. Dazu müssen sie mit dem Befehl *->Extras -> Aktion assoziieren* an die gewünschten Schritte angeschaltet (assoziiert) werden. Die assoziierten Aktionen an einem IEC-Schritt werden rechts vom Schritt in einem zweigeteilten Kästchen dargestellt. Das linke Feld enthält den Qualifier, evtl. mit Zeitkonstanten, das rechte Feld den Aktionsnamen bzw. Boolschen Variablen-namen. Ein- und Ausgangsaktionen sind wie bei einfachen Schritten zusätzlich möglich.

Im **Bild 11-9** wurden zwei solche Aktionen mit den Namen „P_Zählen“ und „RESET_Zaehler“an den Baustein „Ablauf“ gehängt. Die Aktion „P_Zählen“ wurde beispielsweise mit dem Schritt „Pzu_Platz1“ verbunden. Die Aktion „Warten“ trägt als Beispiel einen Qualifier Typ D (verzögert).
Das Bild zeigt weiter auch die komfortable Art der Programmbeobachtung im Online-Modus: Die aktiven Schritte und Aktionen werden blau dargestellt.

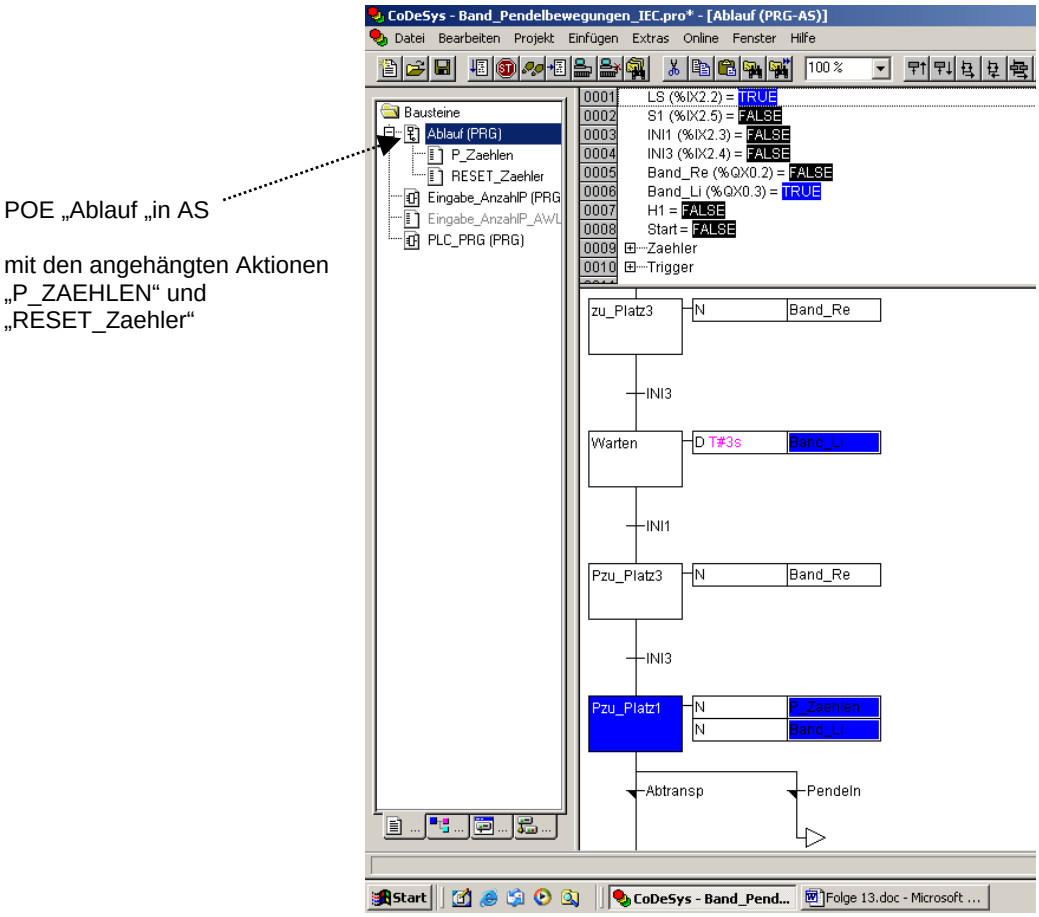


Bild 11-9: Teil einer Schrittkette mit IEC Schritten im Online-Modus

- **Ablaufsteuerung des Verteilerbandes mit IEC - Schritten**

Das Beispiel „Ablaufsteuerung Verteilerband“ nach Abschnitt 11.5.1 wird nachfolgend gleichlautend mit IEC_Schritten gelöst (**Bild 11-10**). Die Transitionen sind die gleichen wie die der Lösung mit einfachen Schritten. Dagegen wird das Programmieren der Aktionen durch Einsatz der Qualifier einfacher. Zu erkennen ist beispielsweise, dass kein Timer für die Wartezeit zu programmieren ist. Diese Aufgabe übernimmt allein der Qualifier „D“.

Es sei hier auf die Vorteile der Ablaufsprache für die Organisation der Einsprünge in bestimmte Schritte hingewiesen. Herkömmliche Programmierung erfordert weit mehr Überlegungen, um bei solchen Problemen die **Gefahr von Endlosschleifen** zu umgehen!

Im Beispiel sind bei zwei Aktionen Programme zu hinterlegen: bei den Aktionen „P_Zaehlen“ und „RESET_Zaehler“. Die zugehörigen Programme sind im Kasten Bild 11-10 rechts angegeben.

Zum Detail des zusätzlichen Schrittes „Bereitschaft“:
Da ein Rücksprung zum Initialschritt unumgänglich ist und dieser Schritt in CoDeSys mit keinen Aktionen assoziiert werden kann, wurde dem Initialschritt ein mit der Transition TRUE direkt aktivierter Schritt „Bereitschaft“ nachgeschaltet. Dieser schaltet aufgabengemäß die LED „H1“ ein.

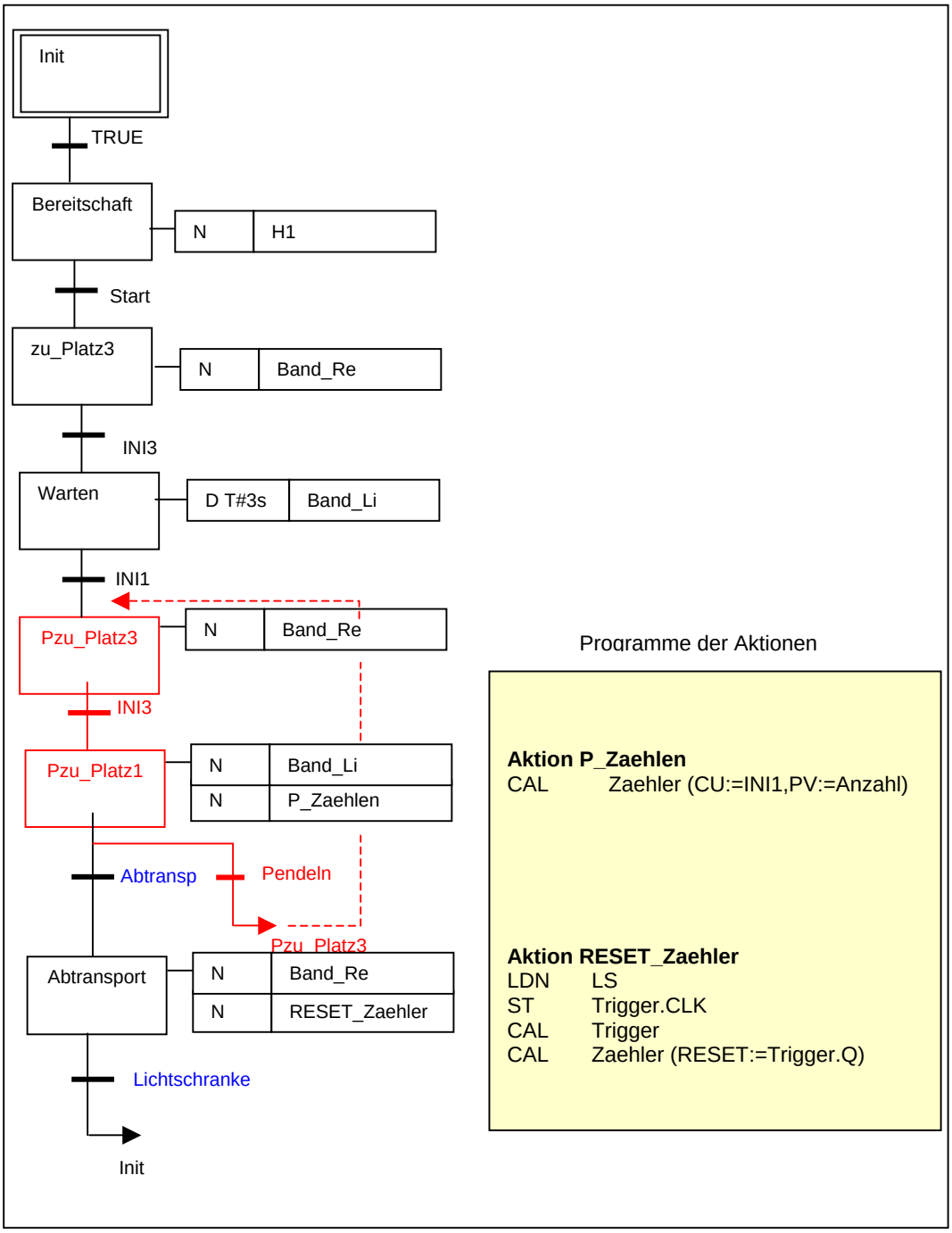


Bild 11-10: Ablaufsteuerung des Verteilerbandes mit IEC-Schritten und zugehörigen Transitionen

11.5.3 Benutzung von Flags in der Ablaufsprache

Mit der Bibliothek "*lecsfc.lib*" werden eine Reihe von Flags für die Steuerung und Überwachung von Schrittketten bereitgestellt (siehe Online-Hilfe „AS-Flags"). Diese sind Boolsche Variablen, die vom System **während der Programmabarbeitung automatisch erzeugt** werden. Beispiel: Wenn ein Schritt länger aktiv ist, als in seinen Attributen angegeben, wird das Flag „SFCError" gesetzt: Die Boolsche Variable „SFCError" wird dann TRUE.

Die **Flags müssen global oder lokal oder auch als Aus- oder Eingangsparameter deklariert** werden.
Folgende Flags sind möglich (Details siehe Online-Hilfe):
SFCEnableLimit ; SFCInit ; SFCReset ; SFCPause ; SFCErrorStep ; SFCErrorPOU ; SFCCurrentStep SFCErrorAnalyzationTable ; SFCTip ; SFCTipMode. Alle Flags werden jeweils mit dem Wert TRUE wirksam.

Bild 11-11 zeigt beispielhaft die Verwendung nachfolgender ausgewählter AS_Flags:

- SFCInit: Initialisierung der Schrittkette: Die Kette wird auf den Initialschritt zurückgesetzt, dieser jedoch nicht abgearbeitet
- SFCReset: wie SFCInit, jedoch wird der Initialschritt bearbeitet
- SFCPause: Die Abarbeitung der Kette wird unterbrochen.

Diese drei SFC-Flags wurden im Beispiel als Parameter vom Typ VAR_INPUT im Ablaufbaustein deklariert. Beim Aufruf des Ablaufbausteins in der POE PLC_PRG wurden diesen Parametern konkrete Signale des Prozesses übergeben. Im Beispiel sind dies Taster von Trainingsrack und Technologiemodell Verteilerband.

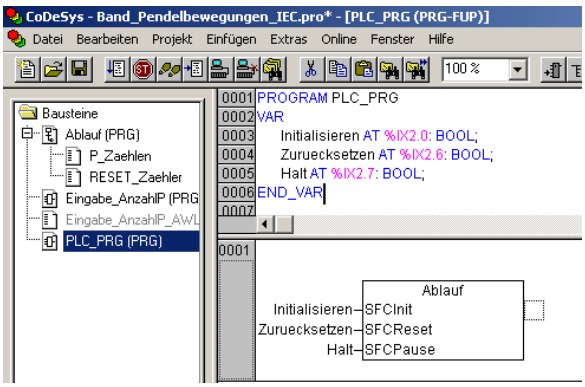


Bild 11-11: Prinzip der Nutzung von SFC-Flags zur Steuerung der Schrittkette

Die Nutzung solcher Funktionen erfordert einige Übung, weil man sich stets Rechenschaft ablegen muss, wie nach Unterbrechung oder Abbruch der weitere Ablauf der Kette erfolgen soll. So muss im Beispiel geprüft werden, wie das Rücksetzen des Zaehlers der Pendelbewegungen bei Abbruch des regulären Verlaufs erfolgen soll. Ohne Änderung der AS würde beim Start nach Abbruch der Zaehler von falschen Werten ausgehen. An dieser Stelle bewahrheitet sich erneut, dass der ungestörte Ablauf einer Kette der einfache Fall ist, während die Gestaltung von Randbedingungen für Einstieg in und Ausstieg aus einer Schrittkette durchaus mehr Überlegungen erfordern!

11.6 Zum Vergleich: Ablaufsprache Graph7

Graph7_Handbuch.pdf und Gaph7_Erste Schritte.pdf auf CD 2 und Online-Hilfe von Graph 7

Das in Step7 Professional integrierte Tool Graph7 ermöglicht die Anwendung der Ablaufsprache im Automatisierungssystem Simatic S7. Viele besondere Details machen das Tool sehr komfortabel. Die Ablaufsprache Graph7 kann **nur in Step7 – Funktionsbausteinen** angewendet werden, und die Daten werden im zugehörigen **Instanzdatenbaustein** verwaltet. Zusätzlich werden einige **System-funktionen** in das Programm eingebunden. Beim Aufruf des Funktionsbausteins stehen dann eine Vielzahl nützlicher Parameter zur Verfügung. Diese sind mit o.a. SFC-Flags vergleichbar. Beispielsweise kann mit einem dieser Parameter die Schrittkette auf den Initialschritt zurückgesetzt werden.

Graph7 verwendet IEC-Schritte. Die Qualifier sind eher begrenzt, erfüllen aber typische Anforderungen. Mit dem speziellen Qualifier CAL können aktive Schritte ohne zusätzliche Anweisungen andere Bausteine aufrufen.

- **Die Standardaktionen in Graph 7 sind:**

Kurzeichen	Bedeutung	Erläuterung
N	Nicht gesetzt	Aktion wird ausgeführt, solange der Schritt aktiv ist
S	Setzen (Gespeichert)	Wird der Schritt aktiv, so wird diese Aktion gesetzt und bleibt aktiv bis zu ihrem Rücksetzen
R	Rücksetzen	Wird der Schritt aktiv, so wird diese Aktion zurück-gesetzt
D	Einschaltverzögert (Delay)	Die Aktion wird eine programmierte Zeit nach Aktivierung des Schrittes ausgeführt. ◄
L	Zeitbegrenzt (Limited)	Ist der Schritt aktiv, so wird die Aktion für eine programmierte Zeit ausgeführt
CAL	**Aufruf eines FC oder FB**	**Solange der Schritt aktiv ist, wird eine POE ausgeführt.**

Von besonderem Wert sind darüber hinaus die Möglichkeiten, Aktionen **ereignisabhängig** zu gestalten. Hierzu werden die Qualifier ergänzt mit den Kennzeichen

„*S*“ (Schritt),
„*V*“ (Supervision) und
„*L*“ sowie „*C*“ (Interlock bzw. Condition).

Mit Interlock werden Schritte verriegelt. Ein programmierter Interlock kennzeichnet das Schrittsymbol mit „C“. Die Operationen der Aktionen werden ebenfalls mit „C“ ergänzt.
Die Programmierung einer Supervision bewirkt eine Schrittüberwachung, beispielsweise seiner Zeitdauer. Überwachte Schritte werden mit „*V*“ gekennzeichnet.

- **Ergänzende Bestimmungszeichen in Graph 7**

S1: Schritt wird aktiviert
S0: Schritt wird deaktiviert
V1: Überwachungsfehler tritt auf (Störung)
V0: Überwachungsfehler ist behoben (keine Störung)
L0: Verriegelungsbedingung kommt
L1: Verriegelungsbedingung geht (z.B. Störung)
C: Verriegelungsbedingung ist erfüllt

Die zweckmäßige Anwendung der ergänzenden Bestimmungs-zeichen erfordert einige Übung, vereinfacht dann aber die Programmierung erheblich!

Bild 11-12 zeigt einen Abschnitt einer Graph7 – Schrittkette, bei der solche ereignisabhängige Aktionen eingesetzt wurden. Schritt S3 wurde mit Interlock und Supervision ausgestattet. Weiter erscheint an Aktionen das Bestimmungszeichen „*S*“. Wird Schritt S3 aktiv, so bewirkt das ergänzende Bestimmungszeichen S1, dass ein Teilezähler um eine Einheit hochgezählt wird. Wird Schritt 2 aktiv, so wird dieser Zähler zurückgesetzt. Um die Programmierung des Zaehlers selbst braucht man sich ebensowenig zu kümmern wie etwa um die Programmierung eines Timers für die zeitliche Schrittüberwachung. Alle diese Aufgaben übernehmen die im Bild gezeigten Aktionen der IEC-Schritte.

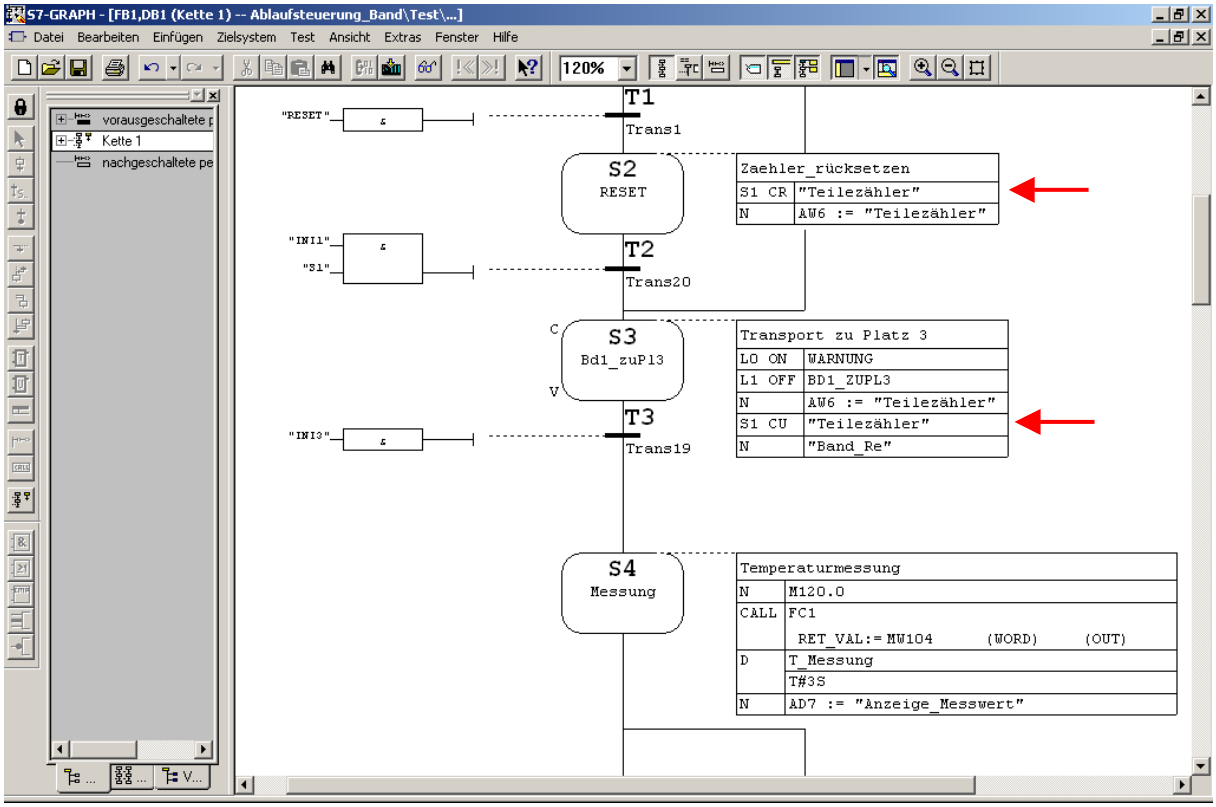


Bild 11-12: Abschnitt einer Schrittkette in der Ablaufsprache Graph7

Supervision und Interlock werden in der Darstellung „Einzelschritt“ programmiert (**Bild 11-13**). Graph7 ist ein rein graphisches Tool, so dass die wenigen Programmanweisungen für Interlock und Supervision ebenfalls **ausschließlich graphisch** geschrieben werden.

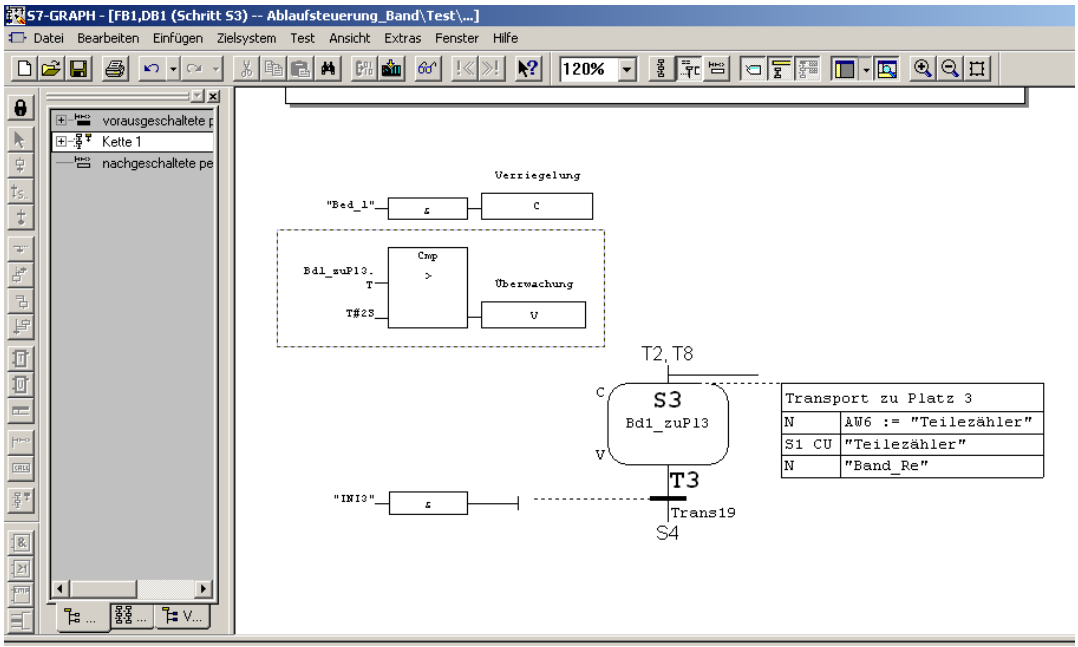


Bild 11-13: Die Programme Interlock und Supervision von Schritt S3 in der Darstellung „Einzelschritt“

Hinweis:

Ebenso wie die Ablaufsprache des Systems CoDeSys enthält das Tool Graph7 viele weiterführende Details, die das Erstellen von Programmen für Maschinen- und Anlagensteuerungen bedeutend erleichtern. Die konsequente Einarbeitung in die Ablaufsprache von CoDeSys und Graph7 wird an dieser Stelle dringend empfohlen!

Im System Graph7 kann das Selbststudium vorteilhaft mit der integrierte Hilfe von Graph7 erfolgen. Diese enthält neben dem „Handbuch Graph7“ in der Rubrik „Erste Schritte“ eine vollständig ausgeführte Ablaufsteuerung am Beispiel einer Bohrmaschine von der Aufgabenstellung bis hin zum Lösungsprogramm.