SPS-Programmierung in Anweisungsliste nach IEC 61131-3

Eine systematische und handlungsorientierte Einführung in die strukturierte Programmierung

Bearbeitet von
Hans-Joachim Adam, Mathias Adam

5. Auflage 2015. Buch. XVII, 246 S. Kartoniert
ISBN 978 3 662 46715 2
Format (B x L): 16,8 x 24 cm
Gewicht: 448 g

Schaltungen mit Signalspeichern

6

Zusammenfassung

In diesem Kapitel erfahren Sie, wie man die Speicher in einer SPS verwendet. Einen Speicher haben Sie bereits kennengelernt: die Merker. Ein Merker ist in der SPS ein Speicherplatz für einen logischen Wert. In der SPS können aber auch Ausgänge als Speicherplätze angesprochen werden. Sie können den Merkern wie auch den Ausgängen direkt einen Wert zuweisen, um ihn anschließend wiederzuverwenden. Sie können Merker aber auch durch einen Setz- oder Rücksetzimpuls auf ‚1' bzw. ‚0' setzen, ähnlich wie bei den Flip-Flops. In diesem Zusammenhang werden wir Besonderheiten im Programmablauf der SPS beleuchten müssen, was uns dann zu den Funktionsbausteinen für die RS- und die SR-Flip-Flops führen wird.

Neben dem Zweck, Zwischenergebnisse festzuhalten, müssen Signale gespeichert werden, um einmalige Vorgänge oder kurzzeitige Impulse auch noch nach deren Beendigung „festzuhalten". Ein Anwendungsbeispiel sind Schaltungen, in denen Taster zum Einsatz kommen.

6.1 Ausgang mit Selbsthaltung

Oft werden in Anlagen Vorgänge durch Taster und nicht durch Schalter ausgelöst. Die Taster geben nur so lange Kontakt, wie sie betätigt werden; beim Loslassen ist das Signal nicht mehr vorhanden. Meistens werden zwei Taster eingebaut: einer zum Ein- und ein zweiter Taster zum Ausschalten. Nach Loslassen des EIN-Tasters muss der Ausgang weiterhin auf ‚1' bleiben, bis am AUS-Taster der Ausgang wieder auf ‚0' zurückgesetzt wird. Man benötigt dazu eine Art Speicherung des Tastendrucks, weil ja die Wirkung über die Dauer des Tastendrucks hinaus aktiv bleiben muss. Wir hatten gesehen, dass dieses Speicherverhalten mittels Flip-Flops realisiert werden kann.

In der Elektrotechnik kann dieses Verhalten mittels eines Hilfskontaktes an einem Relais ebenfalls erreicht werden. Im Schaltbild für den Stromlaufplan A (Abb. 6.1) finden

H.-J. Adam, M. Adam, *SPS-Programmierung in Anweisungsliste nach IEC 61131-3*,
DOI 10.1007/978-3-662-46716-9_6

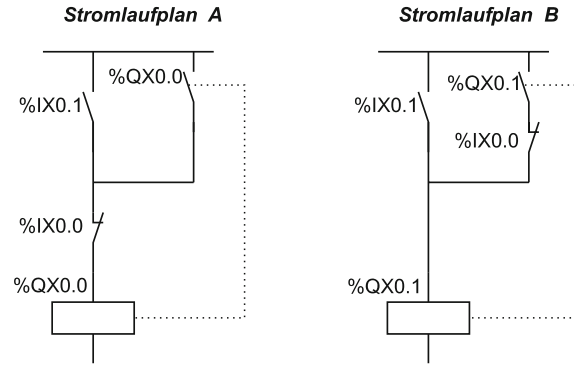


Abb. 6.1 Selbsthaltung: Schaltung zu Übung 6.1

Sie einen Schaltkontakt mit der Bezeichnung `%QX0.0`. Die gleiche Bezeichnung hat auch das Relais. Das bedeutet, dass dieser Schaltkontakt von dem Relais „betätigt“ wird: bei angezogenem Relais ist dieser Kontakt geschlossen.

Das bewirkt dann aber einen geschlossenen Stromweg für das Relais, auch wenn der parallel zum Hilfskontakt liegende Taster `%IX0.1` wieder losgelassen, also geöffnet ist. Sie erkennen die Speicherwirkung: das Relais „hält sich selbst“, bis durch (kurzzeitiges) Öffnen des in Reihe liegenden Tasters `%IX0.0` der Stromkreis unterbrochen wird; dann fällt das Relais nämlich ab und der Hilfskontakt `%QX0.0` wird wieder geöffnet.

Übung 6.1 (POWER61)

Zeichnen Sie für die Stromlaufpläne A und B (Abb. 6.1) jeweils die Funktionspläne, erstellen Sie die Anweisungsliste und testen Sie die Schaltungen mit der SPS.

Überprüfen Sie besonders das Verhalten, wenn die beiden Schalter `%IX0.0` und `%IX0.1` gleichzeitig gedrückt sind.

Hinweis:

Beachten Sie die Darstellung der beiden Schalterarten: Schließer und Öffner. Beim Schließer ist im Ruhezustand der Kontakt geöffnet; die Betätigung schließt den Kontakt. Beim Öffner wird der im Ruhezustand geschlossene Kontakt durch die Betätigung geöffnet. Diese Betätigung ist aber für die SPS nicht von Bedeutung, vielmehr ist das von dem Schalter abgegebene Signal entscheidend für die Programmierung, und das ist stets eine ‚1‘ bei geschlossenem und eine ‚0‘ bei geöffnetem Kontakt, unabhängig ob betätigt oder nicht betätigt!

6.2 Ausgänge setzen und rücksetzen

Beispiel 6.1 (Selbsthaltung: Rührwerk starten und stoppen mit Taster)

Ein Rührwerk soll mit einem Starttaster ein- und mit einem Stopp*taster* ausgeschaltet werden.

Wichtig an diesem Beispiel ist, dass die Aktionen mit Tastern ausgeführt werden sollen, die nach dem Loslassen wieder in die Ausgangslage zurückspringen. Diese Aufgabe könnten Sie mit der Schaltung aus der Übung 6.1 realisieren. In der Lösung zu diesem Beispiel wird das Rührwerk an Ausgang `x` durch den Starttaster an `b` eingeschaltet (gesetzt) und durch den Stopptaster an `a` ausgeschaltet (rückgesetzt):

Beispiel 6.2 (Selbsthaltung)

```
Program POWER61
var
  a AT %IX0.0: bool
  b AT %IX0.1: bool
  x AT %QX0.0: bool
end_var
  LD  b
  OR  x
  AND a
  ST  x
end_program
```

Die Schaltung funktioniert zwar, aber zugegeben, die ganze Sache ist wohl doch zu umständlich und vor allem zu unübersichtlich. Weil solch eine Funktion häufiger benötigt wird, ist es klar, dass es in der SPS spezielle Anweisungen gibt zum *Setzen* von Ausgängen auf ‚1' und zum *Rücksetzen* auf ‚0'. In der Anweisungsliste verwenden Sie den Operator ‚`S`' zum Setzen und den Operator ‚`R`' zum Rücksetzen eines Ausgangs oder eines Merkers.

Übung 6.2 (RS61)

Ergänzen Sie die folgenden Anweisungen (Beispiel 6.3) zu einem vollständigen Programm und testen Sie es mit der SPS. Die Variable `a` soll mit dem Eingang `%IX0.7`, `b` mit `%IX0.6` und `x` mit dem Ausgang `%QX0.0` verbunden werden.

Beispiel 6.3 (zu Übung 6.2)

```
LD  a
S   x
LD  b
R   x
```

Dieses Programm funktioniert als Selbsthalteschaltung! Das Drücken des Tasters `a` an `%IX0.7` setzt den Ausgang `x` an `%QX0.0` auf ‚1'. Dieser Zustand bleibt erhalten, auch wenn der Taster losgelassen wird. Erst Drücken des Tasters `b` an `%IX0.6` löscht den Ausgang wieder und setzt ihn auf ‚0' zurück. Beachten Sie, dass in jedem Fall die

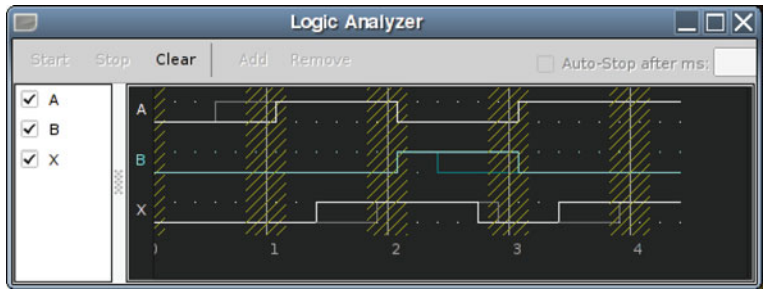


Abb. 6.2 PLC-lite: Logic Analyzer mit Zeitdiagramm zu Übung 6.2

Aktion nur dann ausgeführt wird, wenn die jeweilige Variable `a` bzw. `b` den Wert ‚1' hat. Hat die Variable den Wert ‚0' wird keine Aktion ausgeführt und der Ausgang behält den vorherigen Wert bei.

Hinweis
Mit Hilfe des Logic Analyzers von PLC-lite können Sie die zeitliche Abfolge der Signale beobachten. Öffnen Sie den Logic Analyzer. Klicken Sie dann auf „Start" und bedienen Sie die Taster. Sie erhalten jetzt eine Aufzeichnung der logischen Pegel. Zusätzlich können Sie durch Verwenden des Einzelschrittmodus die Vorgänge eingehend verfolgen. In Abb. 6.2 erkennen Sie dünne und dickere Linien im Zeitdiagramm. Die Unterschiede können Sie etwas später, mit den Erkenntnissen aus Abschn. 6.5 verstehen.

Übung 6.3

Beobachten Sie nun das Verhalten des Ausgangs beim Programm aus Übung 6.2, wenn beide Taster gleichzeitig gedrückt sind! Bei der Simulation können Sie die Taster einzeln „festklemmen", indem Sie mit gedrückt gehaltener Maustaste von dem Taster herunterfahren.

6.3 Ausführungsreihenfolge und Vorrang

Aus Übung 6.3 können Sie erkennen, dass der Ausgang `x` an `%QX0.0` stets rückgesetzt bleibt, solange der Taster `b` gedrückt ist, auch wenn zusätzlich noch Taster `a` gedrückt ist. Tauschen Sie nun die Reihenfolge der Anweisungen in Beispiel 6.3, sodass der `S`-Befehl die letzte Anweisung ist. Sie erhalten das Beispiel 6.4:

Beispiel 6.4

```
LD   b
R    x
LD   a
S    x
```

Nun werden Sie erkennen, dass beim Beispiel 6.4 der Setzbefehl den Vorrang hat und der Ausgang bei dauerhaftem Drücken von beiden Tastern stets gesetzt bleibt. Solch ein Verhalten hatten wir bei den digitalen Schaltungen auch schon erreicht, indem die Signale mit Schaltgliedern entsprechend verschaltet wurden. Blättern Sie hierzu zurück zum Abschn. 3.5.

Bei der SPS entsteht dies durch die Reihenfolge der Anweisungen in der Anweisungsliste. Bei mehrfachen Zuweisungen auf einen Ausgang innerhalb einer Anweisungsliste „gewinnt" die letzte Zuweisung. Es ist daher von größter Bedeutung, in welcher Reihenfolge die Anweisungen in der Anweisungsliste aufgeführt sind.

Die Anweisungen werden der Reihe nach (von oben nach unten) ausgeführt.
Bei mehrfachen Zuweisungen hat die letzte den *Vorrang*.

Doch halt! Bitte schauen Sie nochmal den Abschn. 5.10 (Abb. 5.8) an, in dem die Arbeitsweise der SPS beschrieben ist: Die Anweisungen aus der Anweisungsliste werden nicht nur ein einziges Mal durchgearbeitet, sondern immer wieder „im Kreis herum" vom ersten zum letzten Befehl und dann wieder beginnend mit dem ersten. Und damit wird nach der letzten immer wieder die erste Anweisung ausgeführt. Wie kann nun also die Reihenfolge der Setz- und Rücksetz-Anweisungen überhaupt eine Rolle spielen, schließlich werden sie ja letztlich einfach abwechselnd ausgeführt? – Das werden wir im folgenden Abschnitt betrachten:

6.4 Speicherung der Ein- und Ausgänge

Dieses Verhalten: „Die letzte Anweisung gewinnt" ist in den beiden Beispielen 6.3 und 6.4 sehr nützlich. Es wird dadurch erreicht, dass die SPS die neuen Werte immer nur erst am Ende des Zyklus an den Ausgang übergibt und dadurch der Ausgang einen eindeutigen, konstanten Wert erhält. Die zwischenzeitlichen Änderungen während des Zyklus bleiben nur intern und gehen nicht nach außen.[1]

[1] Genauso werden die Eingangswerte nur zu Beginn jedes Zyklus einmal eingelesen und bleiben dann bis zum Ende der Anweisungsliste unverändert verfügbar.

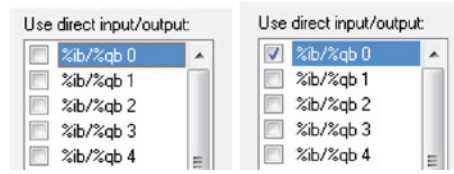


Abb. 6.3 Umschaltung auf direkten und indirekten Zugriff bei PLC-lite

Dies wollen wir nun ausführlicher betrachten: Überlegen Sie, was geschehen würde, wenn die Setz- bzw. Rücksetzanweisungen sofort und unmittelbar am Ausgang wirken würden und nicht erst beim Zyklusende?

Genau! Der Ausgang würde in schneller Folge ein- und ausgeschaltet werden. Es ergäbe sich eine „Blinkschaltung“ mit der Zyklusfrequenz. Man könnte auch sagen: der Ausgang „flattert“.

Durch die erste Anweisung wird der Ausgang zurückgesetzt, wenn der Taster an `%IX0.7` gedrückt ist. Gleich darauf wird der Ausgang wieder eingeschaltet, wenn der Taster an `%IX0.6` ebenfalls betätigt ist. Am besten können Sie dieses Verhalten studieren, wenn Sie das Programmbeispiel 6.2 mit dem Logikanalysator und mit dem Simulator im Einzelschrittmodus (Step) durchtesten.

Sie haben in PLC-lite über das Menü View - PLC-Setup die Möglichkeit, die Zwischenspeicherung der Anschlussklemmen zu deaktivieren, sodass die jeweiligen Werte direkt am Ausgang erscheinen und nicht erst, wenn das Programm den Zyklus bis zu Ende durchlaufen hat. Die Abb. 6.3 zeigt links die Standardeinstellung für die Ein-/Ausgänge: indirekter Zugriff, mit Zwischenspeicherung. Umschalten: Haken setzen um das Byte auf direkten Adresszugriff einzustellen. Im folgenden Abschn. 6.5 werden diese Zwischenspeicher mit den Namen „Prozessabbilder“ ausführlich erläutert.

In Abb. 6.4 wird das Programm aus Übung 6.2 im Einzelschrittmodus ausgeführt. PLC-lite markiert die Zeile, die als nächstes durch Klick auf den Button „Step“ ausgeführt wird. Im Beispiel sehen Sie, dass die Ausführung im „Step“-Modus bis zur Zeile 12 durchgeführt ist. Beide Eingangstaster sind auf den Wert ‚1‘ eingerastet. Die Ausgangsvariable `x` hat den den Wert (Value) ‚1‘, weil der vorhergehende Setzbefehl gerade durchgeführt worden ist. Der `x`-Wert am Ausgang der SPS (Direct Value) ist noch ‚0‘, weil wir den direkten input/output *nicht* gewählt haben (das ist die normale Einstellung). Führen wir nun die Schritte weiter aus, dann wird mit Ausführen der Zeile 13 der Ausgang zurückgesetzt. Nach Ausführen der Zeile 14 kommt dieser Wert an den Ausgang. Trotz gleichzeitigem Setzen und Rücksetzen „gewinnt“ Rücksetzen.

Schalten Sie nun in PLC-Setup durch Setzen des Hakens auf direkten Zugriff der Ein-/Ausgänge. Wenn Sie nun bis zur Zeile 12 durchsteppen, dann erkennen Sie, dass der Setzbefehl sich direkt auf den Ausgang auswirkt und die LED leuchtet. Mit dem Ausführen des Rücksetzbefehls in Zeile 13 wird der Ausgang unmittelbar wieder zurückgesetzt. Die LED blitzt im laufenden Betrieb also ganz kurz auf. In Abb. 6.5 sehen Sie im Logic Analyzer zwei vollständig durchgesteppte Zyklen. Im dritten Zyklus ist gerade der Setzbefehl aus Zeile 11 ausgeführt.

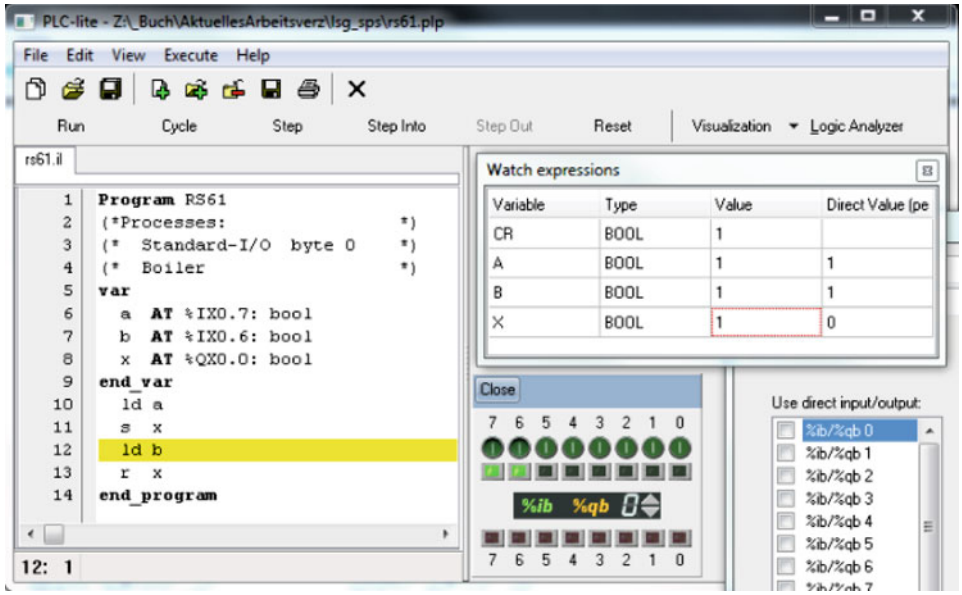


Abb. 6.4 Überwachung der Variablen bei PLC-lite im Step-Betrieb und Ein- und Ausgänge im „normalen“ Modus

Übung 6.4

Führen Sie nun das Programm aus Übung 6.2 erneut aus, jetzt im Einzelschrittmodus und mit Logic Analyzer. Beobachten Sie das Verhalten des Ausgangs `x` abhängig von der Einstellung der Eingänge „direct input/output“ (eingeschaltet oder ausgeschaltet). Testen Sie auch ausgiebig den Fall, dass beide Eingänge gleichzeitig den Wert ‚1‘ haben. Nun können Sie auch das Zeitdiagramm des Logic Analyzers in Abb. 6.2 interpretieren!

Bei einer SPS sind die Werte der Variablen innerhalb eines Programmzyklus von den jeweils zugehörigen peripheren Werten zu unterscheiden, die mittels der Sensoren vom Prozess gelesen werden bzw. die mittels der Aktoren an den Prozess übergeben werden. Je nach Anwendungsfall kann es sinnvoll sein, die peripheren Werte direkt und unmittelbar im Programm auszuwerten oder mittels der Zwischenspeicher indirekt mit den Variablen im Programm zu verbinden. Manche SPS-Systeme bieten hierfür neben den normierten `%i` und `%q` zusätzliche Adressbereiche oder besondere Anweisungen, über die direkt auf die Ein-/Ausgangsklemmen zugegriffen werden kann.

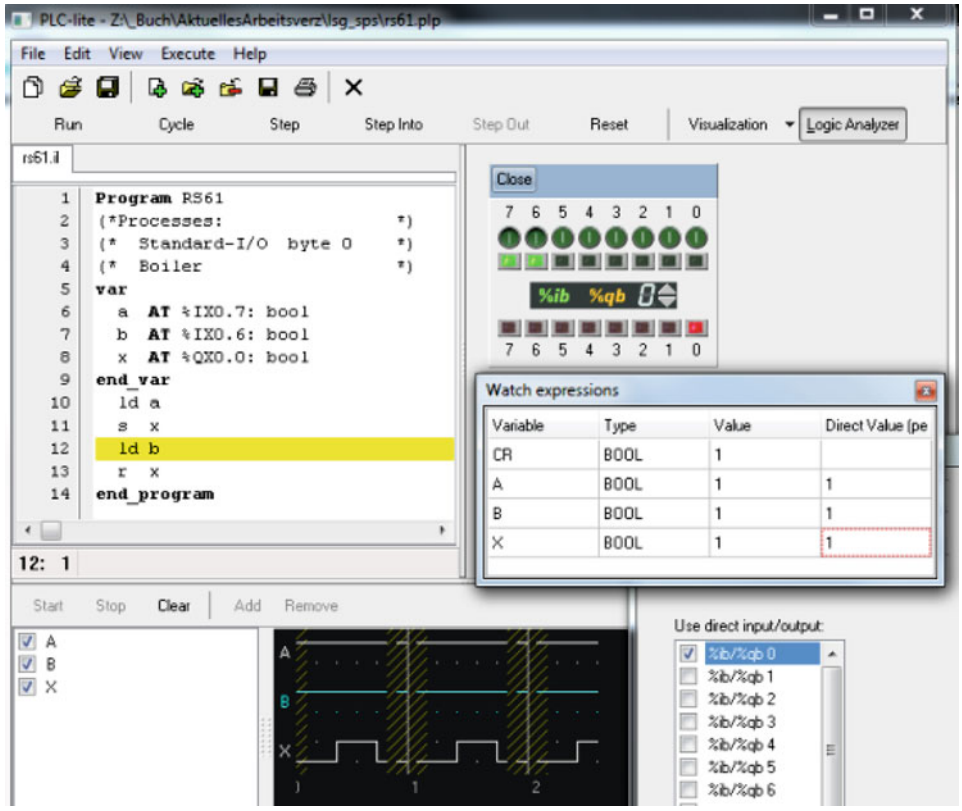


Abb. 6.5 PLC-lite im Step-Betrieb mit Logic Analyzer und Ein- und Ausgängen im direkten Modus

6.5 Prozess-Abbilder der Ein- und Ausgänge

Die Verknüpfungsergebnisse, die den Ausgängen zugewiesen werden, erscheinen nicht sofort an den Ausgangsklemmen, sondern werden erst im „*Prozess-Abbild der Ausgänge*“ (PAA-Register) zwischengespeichert. Erst nach dem letzten Befehl der Anweisungsliste wird das PAA auf die Ausgangsklemmen übertragen. Man erreicht dadurch ein gleichmäßiges, synchrones Arbeiten aller Ausgänge.

Wenn beispielsweise derselbe Ausgang im Laufe des Zyklus mehrmals mit unterschiedlichem Wert angesprochen wird, wird ein „Flattern“ des Ausgangs vermieden durch die Verwendung des PAA als Zwischenspeicher. Dann kann die Zuweisung auf die zum Ausgang gehörige Variable innerhalb des Zyklus ruhig mehrmals erfolgen; es wechselt zwar jedes Mal der Speicherplatz im PAA seinen Wert, aber nur die letzte Zuweisung im Zyklus wird tatsächlich auf den Ausgang übertragen.

Im obigen Beispiel 6.3 „gewinnt“ daher das Rücksetzen. Man sagt, das Programm verhält sich R-dominant. Beispiel 6.4 ist vorrangig setzend, S-dominant.

Aber auch die Eingänge werden nicht direkt im Programm eingelesen. Hier werden zum Zyklusanfang die Werte in das „*Prozessabbild der Eingänge*“ (PAE-Register) eingelesen. Um während eines Zyklus klare Verhältnisse zu haben, auch wenn ein Eingang mehrmals während eines Programmzyklus abgefragt wird, werden die Eingangswerte aller Eingangsklemmen nur immer unmittelbar vor Beginn eines jeden Programmzyklus in das „Prozess-Abbild der Eingänge“ (PAE) übernommen. Während des Programmlaufs werden die Eingangswerte stets vom PAE gelesen. Zwischenzeitliche Änderungen an den Eingängen werden nicht sofort wirksam, sondern erst im nächsten Programmzyklus, nach Übernahme ins PAE. Dadurch kann in einem Zyklus der gleiche Eingang mehrmals abgefragt werden, und er hat immer den gleichen Wert.

Als Nachteil wäre anzumerken, dass damit der kürzeste, sicher erkennbare Eingangsimpuls die Länge der Zyklusdauer haben muss. Kürzere Impulse können einfach „verschluckt“ werden, wenn sie nicht bis zum nächsten Übernahmezeitpunkt andauern.

Oft ist es üblich, die Ein- und Ausgänge als „Peripherie“ zu bezeichnen, im Gegensatz zu den Variablen in der Anweisungsliste und in den Prozessabbildern.

Die Arbeitsweise der SPS ist in Abb. 6.6 dargestellt. Gegenüber Abb. 5.8 sind nun die Prozessabbildregister mit eingezeichnet. Diese Speicherung der Ein- und Ausgänge im PAE bzw. PAA wird von den meisten SPS durchgeführt. Je nach Automatisierungsgerät sind unterschiedliche Verfahren implementiert, die Ein- oder Ausgänge wahlweise über Zwischenspeicher oder direkt als Peripherie-Anschlüsse anzusprechen. Letzteres kann durch Zuweisung bestimmter Speicher- oder Adressbereiche, durch entsprechende Konfiguration, durch Verwendung besonderer Befehle usw. erfolgen.

Wenn die SPS keine Prozessabbilder zur Verfügung stellt oder wenn diese im konkreten Fall nicht verwendbar sind, können wir auch selbst für die Speicherung sorgen, indem wir das Setzen/Rücksetzen zunächst auf Merker anwenden und erst am Schluss des Zyklus den (letzten) Merkerzustand auf den Ausgang übertragen. Mit Hilfe der Merker konstruieren wir uns gewissermaßen ein eigenes PAA! Entsprechend können wir mit Merkern auch einen Ersatz für ein PAE herstellen. Das untersuchen Sie in der nächsten Aufgabe:

Übung 6.5 (RS62)

Testen Sie die beiden verschiedenen Schaltungen aus Abb. 6.7 mit der SPS. Vergleichen Sie auch besonders das Verhalten der beiden Lösungen, wenn beide Taster gleichzeitig gedrückt sind! Beobachten Sie mittels Logikanalysator und im Schrittmodus die Zuweisungen auf die Merker und vergleichen Sie diese mit den Zuweisungen auf die Ausgänge! Begründen Sie die Bezeichnungen: *vorrangig rücksetzend* und *vorrangig setzend*. Woran erkennt man die jeweilige Eigenschaft im Funktionsplan bzw. in der Anweisungsliste?

An diesem Beispiel erkennen Sie, wie die Probleme des sequentiellen, zyklischen Verhaltens einer SPS bei einer direkten Zuweisung auch ohne PAE/PAA vermieden werden:

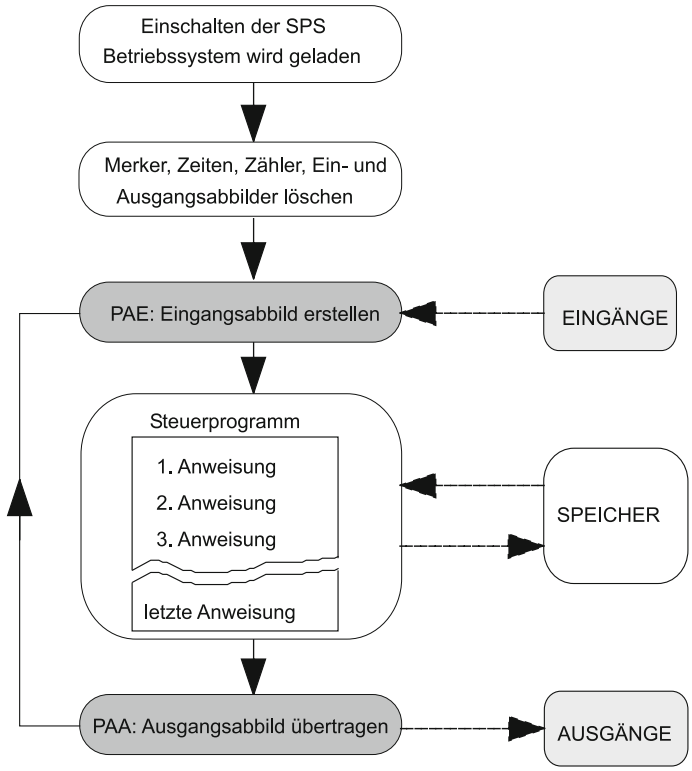
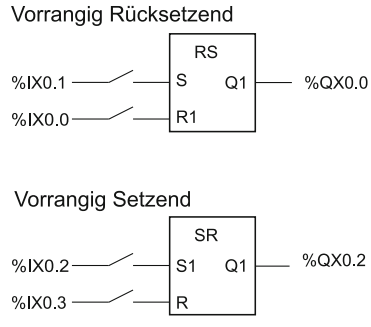


Abb. 6.6 Eingangsabbild PAE und Ausgangsabbild PAA (zu Übung 6.5)

durch die Zuweisung der „Zwischenergebnisse“ auf den Merker `Memo1` bzw. `Memo2`. Durch diese indirekte Zuweisung „gewinnt“ die zuletzt ausgeführte Anweisung; im einen Fall das Rücksetzen, im anderen das Setzen.

Abb. 6.7 Schaltungen zu Übung 6.5



R-Dominant

```
LD   a
S    Memo1
LD   b
R    Memo1
LD   Memo1
ST   x
```

S-Dominant

```
LD   c
R    Memo2
LD   d
S    Memo2
LD   Memo2
ST   y
```

Übung 6.6

Welche der beiden Schaltungen in der erwähnten Übung 6.1 ist vorrangig rücksetzend, welche ist vorrangig setzend?

6.6 Füllstandsteuerung eines Behälters

Der Füllstand des Behälters in Abb. 6.8 soll mit Hilfe einer SPS zwischen einem Minimal- und einem Maximalwert gehalten werden. Ein Taster öffnet das Ventil `V3`. Der Behälter wird solange über das Ventil `V3` entleert, wie der Taster gedrückt bleibt.

Die Steuerung sorgt für den nötigen Füllstand, indem die Sensoren (Grenzwertgeber) `LIS1` und `LIS2` abgefragt werden. Das Einlassventil `V1` wird geöffnet, sobald der Füllstand unter den Stand von `LIS1` absinkt. Das Ventil wird erst wieder geschlossen, wenn der Füllstand den Grenzwertgeber `LIS2` erreicht hat. Die Signale der Grenzwertgeber `LIS` sind ‚1', wenn sie in Flüssigkeit eintauchen.

```
Zuordnungsliste:
LIS1   Unterer Grenzwertgeber   %IX0.1
LIS2   Oberer Grenzwertgeber    %IX0.2
V1     Einlassventil            %QX0.1
V3     Auslassventil            %QX0.3
Sw1    Entleerschalter          %IX0.7
```

Beispiel 6.5 (Füllstand)

Die Füllstandsteuerung kann mit dem folgenden Programmteil verwirklicht werden:

```
LDN   LIS1   (* Behälter leergelaufen      *)
S     V1     (* dann füllen (speichernd!) *)
LD    LIS2   (* Behälter vollgefüllt       *)
R     V1     (* dann füllen beenden        *)
```

Übung 6.7 (TANK61)

Erstellen Sie den Logikplan nach der im Beispiel 6.5 gegebenen Anweisungsliste und programmieren Sie die SPS für die Aufgabe „Füllstandsteuerung". Testen Sie die

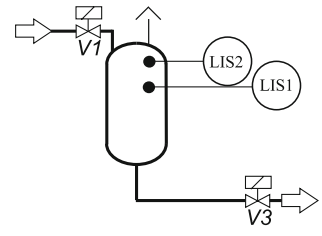


Abb. 6.8 Gefäß mit Sensoren zur Pegelüberwachung

Steuerung am verfahrenstechnischen Modell (Prozess „Niveau“ bzw. „level“). Zum Leeren können Sie den Handtaster bei `V3` betätigen.
Prüfen Sie nach, ob Betriebszustände aufstreten können, bei denen die Bedingungen zum Setzen und Rücksetzen gleichzeitig eintreten. In diesem Fall könnte das Ventils `V1` „Flattern“. Geben Sie an, welche Fehlfunktionen bei den Gebern zu unzulässigen Betriebszuständen führen können!

Erweiterung:

Mit zwei getrennten Tastern soll das Entleeren über `V3` eingeschaltet (gesetzt) und ausgeschaltet (rückgesetzt) werden.

6.7 Alarmschaltung 4

Durch eine Alarmschaltung soll das Bedienpersonal einer Anlage auf gefährliche oder besondere Betriebszustände hingewiesen werden. Meistens ist dann auch ein manueller Eingriff in den Ablauf des Prozesses erforderlich. Durch ein Signal, das der Prozess abgibt, wird die Alarmierung ausgelöst, meist durch eine Hupe oder/und ein Lichtzeichen. Das „Quittungssignal“, das der Anlagenbediener durch einen Tastendruck abgibt, soll die Alarmierung bestätigen. Durch die Quittierung wird meist die Hupe abgeschaltet und das Lichtsignal ein- oder umgeschaltet.

Bei dieser Anwendung müssen einmalige oder kurze Impulse erkannt werden. Daher müssen sie gespeichert werden. Ob diese Speicherung als vorrangig setzend bzw. vorrangig rücksetzend programmiert wird, hängt vom jeweiligen Anwendungsfall ab. Wenn eine Störung gemeldet wird, so soll das Störungssignal erhalten bleiben, solange die Störung nicht beendet ist, auch wenn der Quittierknopf gedrückt wird oder gar in gedrückter Position blockiert wird. Das Störungssignal muss also vorrangig setzend, und damit erst nach der Rücksetzung programmiert werden! Entsprechend ist das Quittungssignal als vorrangig rücksetzend zu realisieren.

Übung 6.8 (ALARM61[2])

In Abb. 6.9 ist eine sehr einfache Alarmschaltung abgebildet. Das Signal S „Stoerung“ soll eine Hupe H (Horn) einschalten, die erst wieder durch eine Quittungstaste „Quitt“ ausgeschaltet werden kann. Erstellen Sie die Anweisungsliste so, dass bei bestehender Störung die Hupe trotz gedrückter Quittungstaste in Funktion bleibt. Das Störungssignal wird am Bit `%ix1.1` wirksam, also im Byte 1. Im Simulationsprogramm können Sie den Prozess „Standard-I/O“ ein zweites Mal öffnen und durch Klicken mit der Maus auf die kleine Ziffer unten die Byte-Nummer umschalten.

```
Zuordnungsliste:
Quitt     %IX0.6
Stoerung  %IX1.0
```

[2] Die zu dieser Übung gehörige Zeichnung (Abb. 6.9) ist nicht vollständig. Es ist bei dieser und bei vielen weiteren Übungen Teil Ihrer „Hausaufgabe“, die Zeichnung zu ergänzen.

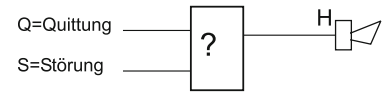


Abb. 6.9 Schaltung zu Übung 6.8

Übung 6.9 (ALARM62)

Eine Hupe soll von drei verschiedenen Schaltstellen aus durch Taster eingeschaltet werden. An jeder dieser Schaltstellen soll je ein Quittierungstaster die Hupe wieder abschalten. Erstellen Sie die Schaltung mit SPS.

Übung 6.10 (ALARM63)

Erweiterung der Übung 6.9: Es muss an drei Lämpchen erkennbar sein, von welcher Meldestelle aus der Alarm ausgelöst wurde.

6.8 Signalspeicher als Funktionsbausteine

Diese im vorigen Abschnitt beschriebene Handarbeit zur Verwendung von Flip-Flop-Schaltungen ist bei einer SPS nach IEC 61131-3 nicht nötig. Es sind nämlich die Flip-Flop-Typen als sogenannte *Funktionsbausteine* enthalten. Solch ein Funktionsbaustein verwaltet die notwendigen Maßnahmen zur Erreichung des vorrangigen Setzens bzw. Rücksetzens selbständig.

Das vorrangig rücksetzende Speicherglied heißt *RS-Flip-Flop* und das vorrangig setzende heißt *SR-Flip-Flop*. In der graphischen Darstellung (Abb. 6.10) stehen die Namen innerhalb des Rechtecks am oberen Rand. Die Bezeichnungen an den Eingängen und dem Ausgang innerhalb des Rechtecks sind die Namen der „*Formalparameter*“. Diese Namen verwendet der Funktionsbaustein intern. Die Eingänge stehen links und die Ausgänge rechts. Die angehängte Ziffer „1“ deutet auf den Vorrang hin. Sie können außerhalb des Rechtecks die Namen der Signalvariablen anschreiben, die Sie in Ihrem Programm selbst verwenden. Diese nennt man die „*Aktualparameter*“.

Abb. 6.10 Funktionsbausteine: vorrangig rücksetzendes und vorrangig setzendes Flip-Flop

6.9 Verwendung von Funktionsbausteinen

Die Anwendung von Funktionsbausteinen geschieht in vier Schritten:

1. Baustein definieren	*instanziieren*
	Baustein verfügbar machen
2. Parameter übergeben	*parametrieren*
	Baustein mit Werten versorgenBaustein liest Werte ein
3. Baustein ausführen lassen	*aufrufen*
	Baustein arbeitet
4. Ausgangswerte auslesen	*Ergebnisse*
	Baustein gibt Werte aus
	Werte können verwendet werden

Sie können hier das in der Computertechnik allgemein bekannte „EVA“-Prinzip erkennen: im Schritt 2 liest der Baustein Werte ein (E), im Schritt 3 werden diese verarbeitet (V) und im Schritt 4 gibt der Baustein die Ergebnisse aus (A).

Schritt 1: Instanziierung
Die Funktionsbausteine kann man nicht direkt aufrufen, sondern man muss für die Anwendung eine sogenannte „*Instanz*“ erzeugen. Das ist viel einfacher, als sich das hier liest, und geht wie die Zuweisung der symbolischen Namen zu den Variablentypen im Deklarationsteil des Programms.

Während Sie bei den Variablen den Datentyp (z. B. BOOL) angeben, müssen Sie zur Instanziierung eines Funktionsbausteines den Funktionsbaustein-Typ angeben. Dieser Typ ist der abgekürzte Name oben in der graphischen Darstellung, also z. B. `RS` oder `SR`.

Die Wirkung der Instanziierung ist bei Variablen und Funktionsbausteinen prinzipiell gleich: bei Variablen wird eine Instanz erzeugt, die ein Speicherplatz vom Typ „BOOL“ ist. Die Funktionsbausteine belegen nach der Instanziierung sowohl Speicherplatz für Variable als auch Verweise auf Funktionen für die durchführbaren Aktionen.

Im Beispiel wird eine Instanz eines vorrangig setzenden Speicherbausteins erzeugt. Den Instanzennamen „`FlipFlop`“ können Sie beliebig wählen:

```
PROGRAM Pulser
VAR
  FlipFlop: SR
END_VAR
```

Schritt 2: Parametrierung
Vor dem Aufruf des Funktionsbausteins im Programm (s. Schritt 3) müssen Sie die Werte für die Eingänge an den Funktionsbaustein übermitteln! Dies nennt man „*parametrieren*“. In der Anweisungsliste können Sie auf zwei verschiedene Weisen parametrieren:

a) Aufruf mit Laden und Speichern der Eingangsparameter Sie können die Aktualparameter aber auch bereits vor dem Aufruf in den Formalparametern speichern. Das hat den Vorteil, dass die Aktualparameter *sofort beim Entstehen* weiterverarbeitet sind.

In den Speicherbefehlen wird der Instanzenname getrennt mit einem Punkt vor dem Formalparameternamen angegeben. Beim späteren Aufruf des Bausteins werden keine Parameter mehr übergeben:

```
LD  a
ST  FlipFlop.R   (* Rücksetzen vorbereiten *)
LD  b
ST  FlipFlop.S1  (* Setzen vorbereiten *)
```

b) Aufruf mit der Liste der Eingangsparameter Bei dieser Methode werden die Schritte 2 und 3 zusammengefasst. Beim Aufruf (vgl. Schritt 3) übergeben Sie eine Liste mit den Aktualparametern. Das Betriebssystem versorgt dann die Formalparameter mit den aktuellen Werten (vgl. Schritt 2).

In der Klammer hinter dem Instanzennamen des Funktionsbausteins geben Sie den Namen des Formalparameters an gefolgt von der Zeichenfolge „`:=`“. Danach wird der Aktualwert, in diesem Fall der Bezeichner von direkten SPS-Eingängen angegeben. Durch diese Schreibweise sehen Sie die Parameterversorgung „auf einen Blick“. *Diese Schreibweise ist in PLC-lite nicht möglich*.

```
CAL FlipFlop (R:=a, S1:=b)
```

Schritt 3: Aufruf im Programm
Den Funktionsbaustein rufen Sie im Programm durch den Operator `CAL` auf. Durch diesen Funktionsbausteinaufruf wird er ausgeführt, d. h. aus den aktuell anliegenden Eingangssignalen und den im Funktionsbaustein gespeicherten Zwischenwerten werden die Ausgangssignale ermittelt.

```
CAL FlipFlop     (* Funktionsbaustein ausführen *)
```

Schritt 4: Verwenden der Ausgangswerte
Mit dem LD-Operator können Sie die Ausgangswerte aus dem Funktionsbaustein in das aktuelle Ergebnis auslesen und anschließend beliebig weiterverwenden, zum Beispiel auf einen SPS-Ausgang geben.

```
LD  FlipFlop.Q1
ST  x
```

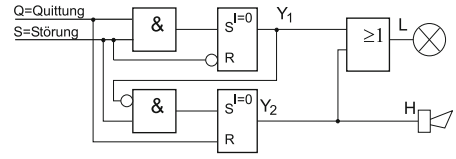


Abb. 6.11 Schaltung zu Übung 6.11

In der Anweisungsliste müssen stets *beide* Anweisungen (Setzen *und* Rücksetzen) vorkommen, weil jedes Flip-Flop, das gesetzt wurde, auch irgendwann einmal zurückgesetzt werden muss.

Übung 6.11 (ALARM64)

Untersuchen Sie die Alarmschaltung 4 (Abb. 6.11). Wann leuchtet die Lampe und wann ertönt die Sirene? Vergewissern Sie sich auch, ob die Flip-Flops vorrangig setzend oder rücksetzend programmiert werden sollten. Die Initialisierung mit dem Wert „0“ („I=0“) wird ohne weiteres Zutun erreicht.

Ergänzen Sie das Zeitdiagramm aus Abb. 6.12!

Verwenden Sie in PLC-lite den Prozess „Boiler“. Mittels des Handtasters für die Heizung können Sie eine Störung erzeugen.

```
Zuordnungsliste:
Quitt   Quittungstaste   %IX0.6
Fault   Störungssignal   %IX1.0
Lamp    Meldelampe       %QX1.0
Horn    Warnsignal       %QX0.7
```

6.10 Steuerung zum Füllen und Entleeren eines Messgefäßes

In der chemischen Verfahrenstechnik müssen die einzelnen Bestandteile genau dosiert werden. Flüssigkeiten werden in einem Gefäß gemessen. Eine Steuerung sorgt dafür, dass das Gefäß genau gefüllt wird. Das Messgefäß wird immer ganz gefüllt und ganz entleert; dadurch ist eine exakte Flüssigkeitsmessung gewährleistet. In den folgenden Aufgaben entwickeln Sie eine Steuerung für ein Messgefäß.

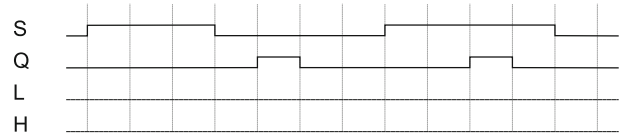


Abb. 6.12 Zeitdiagramm zu Übung 6.11

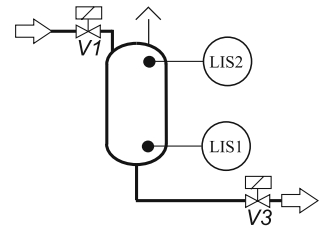


Abb. 6.13 Messgefäß zu Übung 6.12

Übung 6.12 (TANK62)

Das Messgefäß (Abb. 6.13) soll so gesteuert werden, dass nach Impulsgabe über einen Taster „Füllen“ Wasser über das Magnetventil `V1` einläuft, bis der Zustand „Voll“ vom Grenzsignalgeber `LIS2` gemeldet wird. („Leeren“ programmieren wir erst in der nächsten Übung!)
Die `LIS` geben Signal „1“, wenn sie in Flüssigkeit eintauchen.
Erstellen Sie den Logikplan und die Anweisungsliste. Verwenden Sie für die Speicherung die Flip-Flop-Funktionsbausteine `SR` bzw. `RS`! (Prozess: „Tanks (small)“)

```
Zuordnungsliste:
Start   Behälter füllen (Handtaster)    %IX0.7
LIS2    Oberer Grenzwertgeber           %IX0.2
V1      Einlassventil                   %QX0.1
```

Übung 6.13 (TANK63)

Erweitern Sie das Programm aus Übung 6.12 so, dass bei Impulsgabe über den Taster „Leeren“ das Gefäß über das Ventil `V3` entleert wird, bis `LIS1` „Leer“ signalisiert, `LIS1` also „0“-Signal liefert.

```
Zuordnungsliste (zusätzlich zu vorigen Übung):
S1      Behälter ablassen (Handtaster)   %IX0.6
LIS1    Unterer Grenzwertgeber           %IX0.1
V3      Auslassventil                    %QX0.3
```

Auch diese Lösung zeigt noch Schwächen: Wenn Sie den Leeren-Taster drücken, bevor der Kessel ganz voll ist, läuft das Wasser durch. Nun kann es sein, dass weder der obere Grenzwert `LIS2` noch der untere Grenzwert `LIS1` erreicht wird: die Anlage ist blockiert!

Übung 6.14 (TANK64)

Das Leeren bei der Übung TANK63 soll nur möglich sein, wenn der Behälter ganz voll ist (`LIS2` in Flüssigkeit). Umgekehrt darf auch erst bei ganz leerem Behälter das Füllen möglich sein (`LIS1` nicht in Flüssigkeit). Testen Sie Ihr Programm gründlich, und kontrollieren Sie auch, ob während des Füllvorgangs nicht entleert und während des Entleerens nicht gefüllt werden kann! Korrigieren Sie es gegebenenfalls.

An dieser Stelle möchten wir Sie an den Abschn. 2.7 „UND-Verknüpfung als Datenschalter“ erinnern. Mit Hilfe der UND-Verknüpfung können Sie Datenwege abschalten und damit die Wirksamkeit von Eingängen unterbinden. Mit Hilfe dieser Torschaltungen können Sie bei dieser Aufgabe Signale gegenseitig verriegeln und damit beispielsweise verhindern, dass die Kesselentleerung beginnt, bevor der Kessel ganz gefüllt ist.

Untersuchen Sie, ob die beiden Speicher-Flip-Flops vorrangig rücksetzend oder vorrangig setzend sein müssen. Zeichnen Sie auch für diese Aufgabe den Logikplan!

Übung 6.15 (TANK65)

Bei der gleichen Anordnung wie in der vorigen Übung 6.14 soll nun der Handtaster zum Entleeren entfallen. Das Gefäß soll sofort automatisch geleert werden, wenn es vollgelaufen ist.

Erstellen Sie den zugehörigen Logikplan, die Anweisungsliste und testen Sie Ihr Programm mit der SPS und dem verfahrenstechnischen Prozessmodell.

Prüfen Sie, wie sich die Anlage verhält, wenn der Taster „Füllen“ während der ganzen Zeit festgehalten (in gedrückter Stellung blockiert) wird!

Übung 6.16 (SWITCH61)

Versuchen Sie, einen „Druckschalter“ zu programmieren: abwechselnd sollen durch das Drücken *eines* Tasters (nicht zwei Taster!) eine Lampe ein- und wieder ausgeschaltet werden. Also: der gleiche Taster dient sowohl zum Ein- als auch zum Ausschalten der Lampe (wie es z. B. bei Nachttischlampen üblich ist.)

Hinweis:

Diese Aufgabe ist mit dem jetzigen Wissen nur schwer zu verwirklichen; im Abschn. 12.1 wird eine systematische Lösungsmöglichkeit durch Programmierung als „Ablaufsteuerung“ gezeigt. Es lohnt sich aber, wenn Sie sich an diesem Problem jetzt schon einmal versuchen (ohne hinten zu „spicken“) um später die Vorteile des systematischen Vorgehens umso besser zu erkennen.

http://www.springer.com/978-3-662-46715-2

SPS-Programmierung in Anweisungsliste nach IEC 61131-3
Eine systematische und handlungsorientierte Einführung in die strukturierte Programmierung
Adam, H.-J.; Adam, M.
2015, XVII, 246 S. 170 Abb. Mit Online-Extras., Softcover
ISBN: 978-3-662-46715-2