

Programmierung von speicherprogrammierbaren Steuerungen

T. Tyczynski, Thale

Mit der Entwicklung der Mikroprozessor-Technik wuchs gleichzeitig die Anzahl der Anwendungen fernab von Büroautomation und Kommunikation. Eine frühe Applikation war der Einsatz in Prozesssteuerungen. Hier ist in den letzten 20 Jahren ein regelrechter Durchbruch gelungen, der sich vor allem in einer hohen Zuverlässigkeit und Preiswürdigkeit von Automatisierungsprodukten niederschlägt. Zeitig erkannte man, dass Prozesssteuerungen von ganz anderen Motiven geleitet werden als z.B. die Datenverwaltung oder eine Berechnungsroutine. Diese Gesichtspunkte spiegeln sich z. T. in der Norm IEC 1131 wider. Der Beitrag befasst sich mit ihren Grundlagen.

Speicherprogrammierbare Steuerungen (SPS) bzw. Prozess-Steuerungen für die Automatisierung zeichnen sich durch einen hohen Grad an Parallelität verschiedener Aufgaben mit vergleichsweise niedriger Komplexität aus. Das wird deutlich,

wenn z. B. die Auswertung einer Vielzahl von Grenztastern mit logischer Verknüpfung (hohe Parallelität, geringe Komplexität) mit der Lösung einer partiellen Differentialgleichung (geringe Parallelität, hohe Komplexität) verglichen wird. Aus dieser Erkenntnis entstand 1983 die DIN-Norm 19239, an die sich viele SPS-Systeme unverbindlich anlehnten. Dennoch waren und sind die Produkte der verschiedenen Produzenten von SPS so unterschiedlich, dass sich die erstellten Lösungen nicht ohne Aufwand austauschen las-

sen. Die Nachteile uneinheitlicher Plattformen wurden letztlich selbst den Herstellern zu viel.

Daher übernahm eine Arbeitsgruppe innerhalb der IEC (International Electrotechnical Commission) die Aufgabe der Koordinierung und legte im Verbund mit namhaften SPS-Herstellern den weltweiten Standard IEC 1131 vor [1].

1 Inhalt der Norm

Zu ihren grundsätzlichen Inhalten zählen:

- Einsatz einheitlicher, genormter Programmiersprachen für SPS mit dem Ziel einer hohen Portabilität der Programme von System zu System
- Anwendung der strukturierten Programmierung
- Typisierung und einheitliche Erklärung von Variablen, Parametern und Anweisungen
- Einheitliche Konzepte für die Programmdarstellung auf der Basis textorientierter (textueller) und grafischer Methoden
- Hohe Zuverlässigkeit der SPS-Programme unter anspruchsvollen Betriebs- sowie Prüfbedingungen für Hard- und Software.

Insgesamt umfasst die Norm die fünf Teile gemäß Tafel 1, in denen die Anforderun-

Autor

Dipl.-Ing. Thomas Tyczynski ist freiberuflicher Entwickler und Dozent.

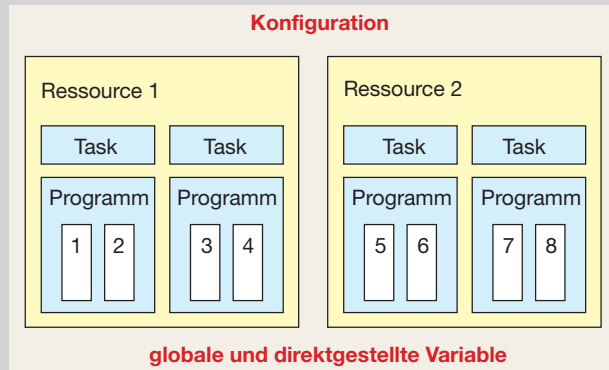
Tafel 1 Bestandteile der IEC 1131

Teil 1: Allgemeines
Begriffsbestimmungen, Beschreibung des technischen Umfeldes und aller SPS-typischen Funktionsmerkmale
Teil 2: Testbedingungen und Betriebsmitelanforderungen
Definition eines allgemeinen Hardwaremodells für SPS, Zusammenstellung aller elektrischen und mechanischen Forderungen bezüglich Funktion, Umgebungsbedingungen, Tests und Typprüfungen
Teil 3: Programmiersprachen (zentraler und für den Anwender wichtigster Teil)
Beschreibung des grundlegenden Softwarekonzepts sowie einer Reihe von leistungsfähigen SPS-Programmiersprachen für die unterschiedlichen Anwendungen
Teil 4: Anwenderrichtlinien
Behandlung der verschiedenen Projektphasen (Arbeitsablauf) von der Systemauswahl über die Analyse bis zu Inbetriebnahme und Wartung
Teil 5: Kommunikation
Kommunikationsmodelle und deren Modalitäten des Informationsaustausches von mehreren SPS untereinander und mit anderen Systemen

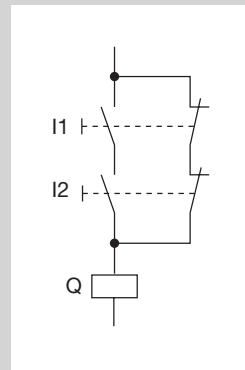
Tafel 2 Programmorganisationseinheiten (POE)

POE-Typ	Schlüsselwort	Beschreibung	Aufrufmöglichkeiten
Programm	PROGRAMM	Hauptprogramm mit Deklaration der globalen Variablen und Zugriffspfade	Funktionsbausteine, Funktionen
Funktionsbaustein	FUNKTION_BLOCK	Parametrierbares (Unter)Programm mit statischen Variablen, d.h. mit „Gedächtnis“	Funktionsbausteine, Funktionen
Funktion	FUNKTION	Parametrierbares (Unter)Programm ohne statische Variablen, d.h. ohne Gedächtnis	Funktionen

Konfiguration



1 Systemgliederung einer SPS



2 Schützsteuerung mit Äquivalenzfunktion

gen an Steuerungssysteme beschrieben werden.

Für den Programmierer und Anwender bedeutet der Teil 3 die wesentlichste Information mit dem größten Vorteil. Wird doch hier ein Satz einheitlicher, herstellerunabhängiger SPS-Sprachen beschrieben, ohne dass die Hersteller auf eigene Softwarekomponenten verzichten müssen. Die Sprachen besitzen trotz ihrer grundsätzlichen Universalität bevorzugte Anwendungsgebiete [2].

Die Norm unterteilt ein Steuerungsteuerungssystem grob in die Rubriken „Konfiguration“, „Ressource“ und „Task“. Eine Konfiguration gemäß Bild 1 ist eine komplette SPS, die aus einer oder mehreren Zentraleinheiten und der Peripherie zur Bereitstellung der Variablen besteht.

Der Vorteil der Anwendung dieser Norm liegt auch darin, dass die dazugehörigen, theoretischen Inhalte dem Anwender/Programmierer nicht bekannt sein müssen.

Die von der Norm vorgeschriebene Gliederung der SPS-Software in die Programmorganisationseinheit/en (POE) „Programm“, „Funktion“ und „Funktionsbaustein“ entspricht dem Ziel einer strukturierten Programmierung (Tafel 2).

2 Die Programmiersprachen

2.1 Übersicht

Die Festlegung eines einheitlichen Sprachkonzepts ist zweifellos die größte Errungenschaft des Vorschriftenwerkes. Bei der Aufzählung der möglichen Sprachvarianten wurden die Belange der bestehenden Sprachkonzepte, die Praxiserfahrung mit Steuerungen und künftige Entwicklungen der gesamten Programmiergemeinschaft weitgehend berücksichtigt. Die einheitlichen Werkzeuge bieten nicht nur für jeden Geschmack, sondern vor allem auch für jedes Problem eine angepasste Lösungsmöglichkeit. Prinzipiell existieren wie bereits vor Erarbeitung der IEC zwei grundsätzliche Sprachnormen.

Die textorientierten Sprachen „Anweisungsliste“ (AWL) und „Strukturierter Text“ (ST) lehnen sich in ihren Elementen an die Maschinensprache (AWL) bzw. an Hochsprachen wie Pascal, C oder Modula (ST) an.

Zu den grafischen Sprachen gehören der Kontaktplan (KOP) mit seiner starken Nähe zu Steuerstromkreisen und die Funktionsbausteinsprache (FBS) mit Schwerpunkt logische bzw. allgemeine Verknüp-

fungs- und Verarbeitungsglieder.

Für die übersichtliche Darstellung von Abläufen (Schrittketten) ist die Ablaufsprache (AS) gedacht.

Alle Beschreibungsmittel sind vollwertige Varianten, nur dass sich nicht jeder Typ optimal zur Lösung eines bestimmten Problems eignet. Außerdem lassen sich die Varianten in hohem Maße auch einfach umwandeln, d.h. man kann in AWL schreiben und dann z.B. die Ansicht KOP wählen.

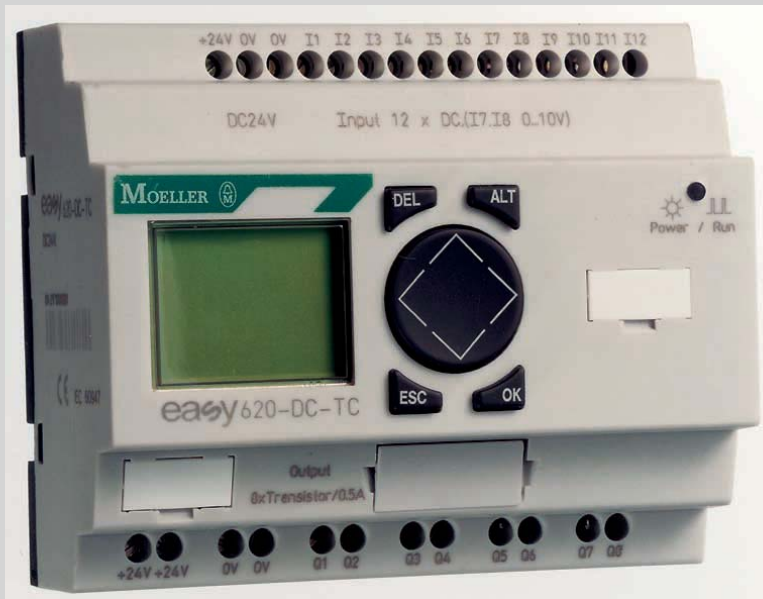
Um die unterschiedlichen Umsetzungen in den Sprachen besser zu verstehen, wird von der Schaltung Bild 2 ausgegangen. Sie besteht aus einer Reihenschaltung zweier Schließer I1 und I2 mit Parallelschaltung der Reihe aus den Öffnern I1 und I2. Logisch gesehen liegt hier eine Äquivalenz vor, beide Taster müssen entweder geöffnet oder geschlossen sein.

2.2 Anweisungsliste AWL

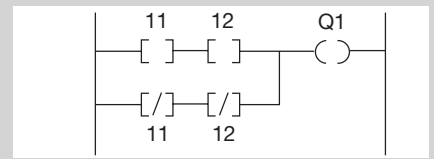
Eine Anweisungsliste besteht aus Zeilen mit folgender Gliederung:

(Sprungmarke) Operation Operand
(Kommentar)

Die in Klammern stehenden Begriffe sind optional.



3 Kleinststeuerung Easy 600
(Foto: Moeller)



4 Äquivalenz als Kontaktplan KOP

Tafel 3 Symbole der Sprache Kontaktplan (KOP)

Symbol	Beschreibung
{ / }	Symbol für Kontakt („/“ für Öffner)
(/)	Spule („/“ für negiert)
{ S }	Spule mit Setzfunktion wird gesetzt
{ R }	Spule mit Setzfunktion wird rückgesetzt
I-Variable->>Marke	Bedingter Sprung nach Marke, wenn Variable = wahr (true)
I->>Marke	Unbedingter Sprung nach Marke

Für die Schaltung Bild 2 ergibt sich das nachstehende Programm.

```
LD I1      (*Schließer von Taster I1 holen*)
AND I2     (*UND Schließer von Taster I2*)
OR (I1     (*Klammer, um zweite Reihe zu separieren*)
  NOT      (*Öffner von I1 bilden*)
  ANDN I2  (*Durch ANDN Öffner von I2 in Reihe*)
)         (*Klammer schließen*)
ST Q1     (*Ergebnis an Ausgang Q1 übergeben*)
```

Das Ergebnis einer jeden Programmzeile steht in einem Arbeitsregister und kann so von der nächsten Zeile genutzt werden. Die Umsetzung der Schaltung mutet auf den ersten Blick recht umständlich an. Für die gezeigte Aufgabe ist die AWL auch nicht optimal. Wenn jedoch verlangt werden würde, dass das Ergebnis, hier in der Form 0/1, als numerische Zahl weiterverarbeitet soll, dann zeigt die AWL echte Stärken.

```
LD Q1      (*Der Zustand von Q1 wird geladen, 0 oder 1*)
BOOL_TO_INT (*Diese Funktion wandelt BOOL in INTEGER um*)
MUL 5     (*Ergebnis ist entweder 5 oder 0*)
```

Auf diese Weise kann ein Wert gebildet werden, der sich z.B. mit anderen digitalisierten Analogwerten verrechnen lässt. Der Vorteil der AWL liegt bei Aufgaben mit hohem Anteil an arithmetischen Re-

chenoperationen. Ein weiterer Vorzug findet sich bei realisierbaren Verzweigungen und Programmsprüngen.

2.3 Strukturierter Text ST

Programme als ST ähneln ausgeprägt Programmen in einer Hochsprache wie Pascal oder C. Die Sprache ist nicht zeilenorientiert, abgeschlossene Anweisungen benötigen ein Semikolon am Ende. ST spielt bei SPS eine relativ untergeordnete Rolle.

2.4 Kontaktplan KOP

Der Kontaktplan besitzt die größte Ähnlichkeit mit der klassischen Schützschaltung. Daher empfiehlt sich die Verwendung immer dort, wo eine bewährte Schaltung elektronisch umgesetzt werden soll. Der Programmentwurf mit dem KOP ist für Steuerungspraktiker und Elektrotechniker recht gewohnt. Deshalb wurde diese Sprache zur Basis für die Kleinststeuerungen „Easy“, „Logo“ usw. (Bild 3).

Die gestellte Aufgabe Bild 2 wird gemäß Bild 4 umgesetzt.

Dabei stehen die vier linken Symbole für die (Kontakt-)Eingänge, das rechte für eine Spule, den Ausgang. Durch Reihen- und Parallelschaltung verschiedener Eingänge entsteht die logische Struktur.

Ein bildliche Darstellung verdeutlicht dem Fachmann die Funktion natürlich viel schneller als einige Textzeilen. Für rein logische Verknüpfungen ist der KOP daher eine ausgesprochen anwenderfreundliche Programmiersprache, die allerdings auch ihre Grenzen kennt. Bei mathematischen Operationen nämlich sind Textzeilen we-

gen der Nähe zu mathematischen Formeln übersichtlicher. Im Grunde genommen ist die KOP-Sprache fast nur für Operanden geeignet, die die Zustände 0 und 1 (Bit-Operanden) kennen.

Die in der KOP-Sprache für die Programmierung zur Verfügung stehenden Symbole zeigt Tafel 3.

2.4 Funktionsbausteine

Funktionsbausteine symbolisieren komplexere logische Verknüpfungen, vor allem aber auch standardmäßig häufig benötigte Funktionsabläufe. In der IEC werden neben den einfachen logischen Verknüpfungen UND, ODER usw. daher Funktionen zur Typkonvertierung von Variablen, aber auch Zähler, Schieberegister und Zeitbausteine beschrieben. Darüber hinaus stellen die Hersteller eigene Bausteine zur Verfügung, die dann natürlich nicht unbedingt der Norm entsprechen und demzufolge nicht auf die Steuerung eines anderen Herstellers übertragbar sind. Ist Portierbarkeit oberstes Ziel, muss der Programmierer streng auf die Kennzeichnung der Bausteine in seinen Unterlagen achten. Nur das Zeichen IEC garantiert den komplikationsfreien Übergang. Häufig sind die Eigenschaften der Steuerungen aber auch ähnlich. Dann bringt der Übergang auf eine andere SPS keine Schwierigkeiten, obwohl nichtgenormte Elemente verwendet wurden.

Die Funktionsbausteinsprache wird immer dort mit Vorteil eingesetzt, wo elektronische Schaltungen mit integrierten digitalen Bausteinen durch ein Programm ersetzt werden sollen. Die Funktionsnachbildung

eines Schaltkreises entspricht dann den Bausteinen der Sprache. Wer es gewohnt ist, einen Vierfach-NAND-Chip aus dem Regal zu ziehen und damit loszulöten, bekommt mit dem Funktionsplan keinerlei Probleme.

Bild 5 entspricht der Funktion Äquivalenz nach Bild 2 und 4.

Die Kreissymbole an Ein- oder Ausgängen symbolisieren eine Negation und sind für alle logischen Gatter nutzbar. Neben den Standardbausteinen AND, OR, XOR existieren bereits komplexere Bausteine, wodurch die Sprache zur „Schaltungssprache“ wird. Zwei Beispiele in Bild 6 verdeutlichen dies. In solchen Funktionsbausteinen FBS stecken natürlich auch nur wieder fertige Programme, die man selber per AWL schreiben könnte. Der Reiz besteht nun darin, dass der hardwareorientierte Anwender genau dies nicht tun muss. Außerdem ist die Darstellung weitaus übersichtlicher als in der AWL, besonders wenn eine Mischung mit hohem Anteil logischer Verknüpfungen stattfindet.

Nachteilig wirkt häufig die begrenzte Anzahl der Eingänge je Logikbaustein. Aus

dem deshalb notwendig werdenden Zerlegen von Verknüpfungen folgt u. U. ein unübersichtliches Bild der Programmstruktur..

2.5 Ablaufsprache AS

Die Ablaufsprache ermöglicht die Programmierung von sequenziellen Abläufen (Ablaufsteuerung im Gegensatz zur Verknüpfungssteuerung). Ein typisches Beispiel hierfür ist die Waschmaschine. Hier folgt ein Schritt dem anderen in Abhängigkeit vom gewählten Programm und der jeweiligen Weichschaltbedingung (z.B. Temperatur der Trommel).

Die Ablaufsprache besteht aus den Grundelementen „Schritt“ und „Transition“. Zu jedem Schritt gehören Aktionen (Befehle), die die dem Schritt zugeordneten Aufgaben lösen. In den Transitionen (Weichschaltbedingungen) werden die Übergangsbedingungen von einem Schritt auf einen (oder mehrere) Folgeschritte festgelegt. Schritte werden grafisch als Rechtecke, Transitionen als Querstriche in den Verbindungslinien dargestellt (Bild 7).

Schritte und Transitionen folgen einander stets im Wechsel. Nie können zwei Transitionen oder zwei Schritte hintereinander stehen. Die detaillierten Aktionen der Schritte und die Übergangsbedingungen werden in einer beliebigen Sprache erstellt. Eine Schrittkette lässt sich über einen Funktionsbaustein verwalten. Alle Schritte und Transitionen werden dann in eigenen Bausteinen programmiert und vom Schrittkettenbaustein aufgerufen (Bild 8).

Ein Schritt kann aktiv oder inaktiv sein. Ist er aktiv, wirkt er über die ihm zugeordneten Aktionen auf den Prozess ein. Zu jedem Schritt gehört ein Schrittmerker, der den Wert „1“ besitzt, wenn der Schritt aktiv ist. Aktiv wird ein Schritt dann, wenn die vor ihm liegende Transition erfüllt wurde. Ebenso kann eine Übergangsbedingung (Transition) nur erfüllt werden, sofern der zurückliegende Schritt aktiv ist. Die Bedingungen für den Übergang werden genau wie die Aktionen im Programm festgelegt. Existiert eine erfüllte Transition, wird nicht nur der folgende Schritt aktiv, sondern der zurückliegende sofort inaktiv.

Bei Bedarf gehören zu einer Schrittkette

auch Verzweigungen. Man unterscheidet UND- und ODER-Verzweigungen.

Die Schrittkettenprogrammierung in AS entlastet den Programmierer von der zeitraubenden Arbeit der detaillierten Analyse einzelner Prozesse. Sie aber bildet eine unbedingte Voraussetzung, den Sicherheitsforderungen zu genügen. Der Test der möglichen Schrittabfolgen unter allen Weichschaltbedingungen erfordert hohen Aufwand. Außerdem besteht das Risiko des Übersehens. Die Zuverlässigkeit der Schrittkettenbausteine hingegen ist nachgewiesen. Es kommt also lediglich auf die Kontrolle der Einzelaktionen an.

3 Zusammenfassung

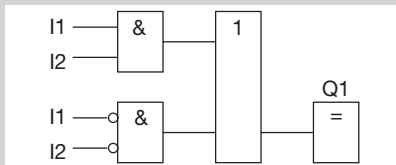
Mit der Norm IEC 1131 bzw. der DIN EN 61131 wurden Programmierverfahren geschaffen, die sich über das Vorbild der Schützsteuerung hinaus sehr stark an die Denkweisen der Datenverarbeitung anlehnen. Der Techniker muss sich damit in eine neue und komplexere Programmierung einarbeiten. Auf lange Sicht bringen solche weltweiten Vereinbarungen ausschließlich Vorteile. Gestatten sie doch dem Anwender, sich auf die Inhalte und Lösungen zu konzentrieren und befreien ihn vom lästigen Nachdenken über syntaktische Regeln und Schreibweisen. Auch das Einbeziehen von Computern als zusätzliche „Ressource“ in ein Steuerungssystem, z.B. zum Zwecke der Visualisierung, wird damit einfacher und schneller.

Für die verschiedenen Programmiersprachen stehen leistungsfähige Werkzeuge zur Verfügung. Als Beispiele für solche Hilfen zum Erstellen von Programmen nach IEC 1131 seien hier stellvertretend genannt:

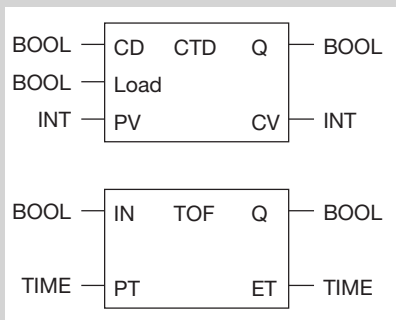
- SUCOSOFT S40 Fa. Moeller
- ACON-ProSys 1131 Fa. Deltalogic, Mutlangen
- PC WORX Fa. Phoenix Contact

Literatur:

- [1] Pusch, K.: Grundkurs IEC 1131. Würzburg: Vogel Buchverlag 1999.
- [2] Tyczynski, T. SPS-Einsatz in der Gebäudetechnik. Berlin: Verlag Technik 1999.

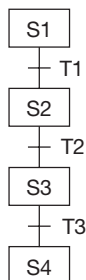


5 Äquivalenz in Funktionsbausteinen FBS

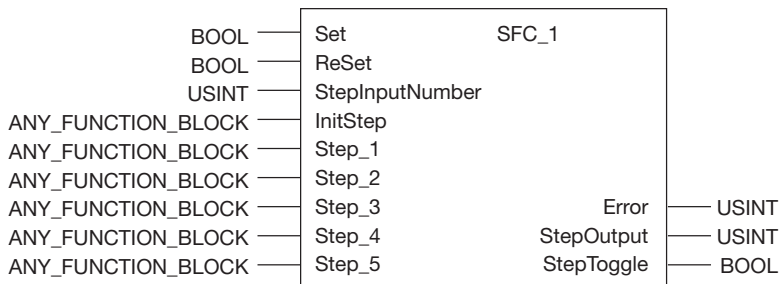


6 Komplexe Funktionsbausteine

- CTD** Rückwärtszähler
- TOF** Ausschaltverzögerung (Ersatz für einen Treppenhausautomaten)



7 Schrittkette in der Ablaufsprache AS



8 Funktionsbaustein für eine Schrittkette