Norm EN 61131

Die Europäische Norm EN 61131, die auf der internationalen Norm IEC 61131 basiert, befasst sich mit den Grundlagen Speicherprogrammierbarer Steuerungen.
Eine objektorientierte Weiterentwicklung für verteilte Steuerungen ist die EN 61499.

Teil 1: Allgemeine Informationen; aktuelle Ausgabe 3.2004
Teil 2: Betriebsmittelanforderungen und Prüfungen; aktuelle Ausgabe 04.2008
Teil 2 Berichtigung 1: Betriebsmittelanforderungen und Prüfungen; aktuelle Ausgabe 01.2009
Teil 3: Programmiersprachen; aktuelle Ausgabe 06.2014
Teil 3 Beiblatt 1: Leitlinien für die Anwendung und Implementierung von Programmiersprachen für Speicherprogrammierbare Steuerungen; aktuelle Ausgabe 4.2005
Teil 5: Kommunikation; aktuelle Ausgabe 11.2001
Teil 6: Funktionale Sicherheit; aktuelle Ausgabe 10.2013
Teil 7: Fuzzy-Control-Programmierung; aktuelle Ausgabe 11.2001

Des Weiteren wurden zwei Technical Reports (TR) von der International Electrotechnical Commission herausgegeben:

IEC TR 61131-4: User Guidelines
IEC TR 61131-8: Guidelines for the application and implementation of programming languages
IEC TR 61131-9: Single-drop digital communication interface for small sensors and actuators (SDCI) - allgemein bekannt als IO-Link

Teil 2: Betriebsmittelanforderungen und Prüfungen (EN 61131-2:2008)

Teil 2 der Norm beschreibt Anforderungen an die Steuerungshardware und gibt Testanleitungen.
Eine Steuerungshardware, die alle Anforderungen der IEC 61131-2 erfüllen kann, gilt auch als sicher im Sinne der Konformitätsbewertung für die CE-Kennzeichnung.

Teil 3: Programmiersprachen (EN 61131-3:2014-06)

Die EN 61131-3 (auch IEC 1131 bzw. 61131) ist die einzig weltweit gültige Norm für Programmiersprachen von speicherprogrammierbaren Steuerungen. Sie löste am 1. August 1994 die DIN 19239 ab und definiert die folgenden fünf Sprachen:

Englisch		Deutsch		
Abk.	**Bezeichnung**	**Abk.**	**Bezeichnung**	
IL	Instruction List	AWL	Anweisungsliste	Text
Vergleichbar mit Assembler; (Ausgabe 2014-06 der Norm als veraltet gekennzeichnet, soll künftig entfallen.)				
LD	Ladder Diagram	KOP	Kontaktplan	Grafik
Vergleichbar mit einem Elektro-Schaltplan, der um 90° gedreht ist				
FBD	Function Block Diagram	FBS	Funktionsbaustein-Sprache	Grafik
Auch als FUP (Funktionsplan) bekannt, insbesondere bei Siemens STEP 7. Ähnelt Logik-Schaltplänen				
SFC	Sequential Function Chart	AS	Ablaufsprache	Grafik
Eine Art Zustandsdiagramm, bei STEP 7 als S7 GRAPH bekannt. Basiert auf Grafcet nach EN 60848.				
ST	Structured Text	ST	Strukturierter Text	Text
Angelehnt an Hochsprachen, bei STEP 7 als SCL (Structured Control Language) bezeichnet.				

IL	Instruction List	AWL	Anweisungsliste	Text

AWL dient hauptsächlich zur logischen Verknüpfung von Steuerungseingängen und -ausgängen. Typischerweise wird ein (digitaler) Eingang in das Arbeitsregister (auch Akkumulator genannt) geladen (load digital input 0, „LD %IX0.0“), mit anderen Eingängen, Konstanten oder Speicherwerten verknüpft (exklusiv-oder speicher bit 3, „XOR %MX0.3“) und auf einen Ausgang geschrieben (store digital output 1, „ST %QX0.1“).

Die Hauptmerkmale von AWL sind, dass Operatoren nur einen Operanden besitzen und die Syntax der Sprache an die Assemblersprache angelehnt ist. Somit bietet sie nur sehr umständliche Strukturierungsmöglichkeiten durch Sprungbefehle. Vorteile ergeben sich aber, wenn aufgrund einer Speicherknappheit der eingesetzten CPU der Programmcode kleingehalten werden soll. Auf älteren Steuerungen sind AWL-Programme noch relativ häufig anzutreffen. AWL-Programme sind aber im Vergleich zu Programmen in höheren Sprachen insbesondere bei größeren Projekten sehr unübersichtlich und schlecht wartbar.

Viele SPS-Hersteller bezeichnen die von ihnen verwendete Sprache auch dann als AWL, wenn sie sich nicht exakt an die IEC 61131-3 hält, so dass sich bestehende AWL-Programme kaum auf Steuerungen anderer Hersteller übertragen lassen.

Beispiel 1: Und-Verknüpfung zweier binärer Eingänge auf einen Ausgang

```
L  INPUT1
AND INPUT2
ST  OUTPUT
```

Beispiel 2: Addition zweier Integer-Werte

```
L  WERT1
ADD WERT2
ST  OUTPUT
```

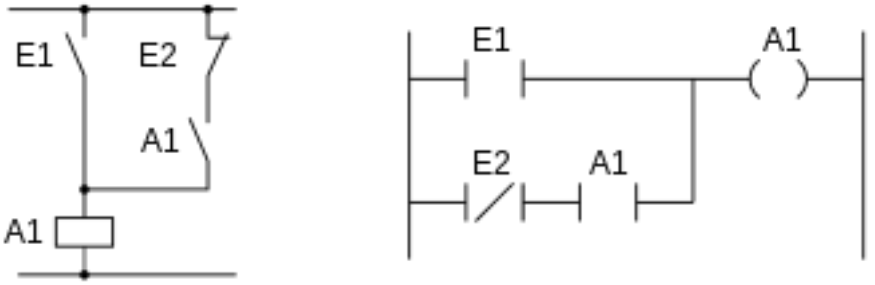
Beispiel 3: RS-Flipflop (rücksetzdominant)

```
L  S-INPUT
S   AUSGANG
L  R-INPUT
R   AUSGANG
```

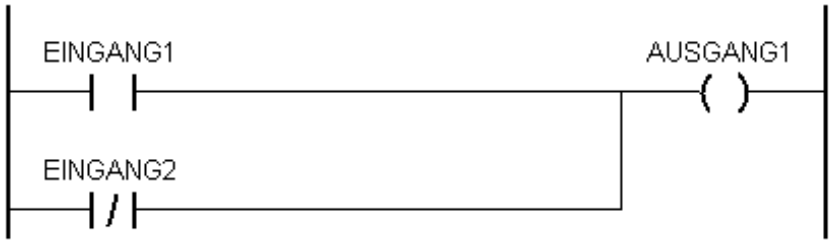
LD	Ladder Diagram	KOP	Kontaktplan	Grafik

Die Programmiersprache Kontaktplan ist insbesondere für Verknüpfungssteuerungen geeignet ist.
In ihrer Darstellung ist sie an Stromlaufpläne angelehnt.
Werden die Elemente in Reihe geschaltet, so bedeutet dies eine UND-Verknüpfung. Werden sie parallel geschaltet, so ist dies eine ODER-Verknüpfung. Ein Schrägstich im Element bedeutet eine Negierung. Eingänge werden dabei als zwei vertikale parallele Linien dargestellt, Ausgänge dagegen als gegenüberliegende gebogene Linien.
In fast allen modernen KOP-Sprachen sind aber auch Funktionsblöcke verfügbar, die weit über die eigentliche Verknüpfungssteuerung hinausgehen.

Beispiel 1: Vergleich Stromlaufplan und Kontaktplan



Beispiel 2: AUSGANG1 = EINGANG1 OR NOT EINGANG2



FBD	Function Block Diagram	FBS	Funktionsbaustein-Sprache	Grafik

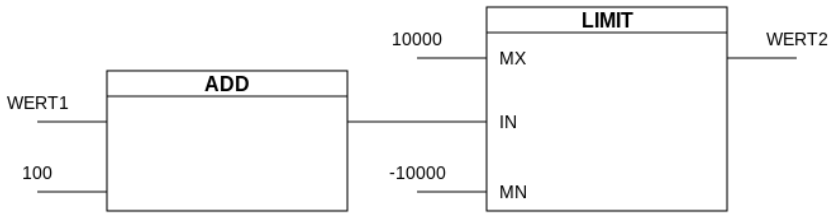
Die Funktionsbausteinsprache (Abkürzung FBS) wird oft auch Funktionsplan (Abkürzung FUP) genannt (insbesondere bei Siemens Step7).
Die grafisch orientierte Programmiersprache verwendet in ihrer Darstellung die Logiksymbole der Booleschen Algebra. Sie ist insbesondere für Verknüpfungssteuerungen geeignet und vor allem bei Anfängern und wenig fortgeschrittenen Programmierern beliebt, da die Programmlogik durch die Visualisierung relativ leicht nachvollziehbar ist.

Ein FBD-Programm besteht aus folgenden grafischen Elementen:

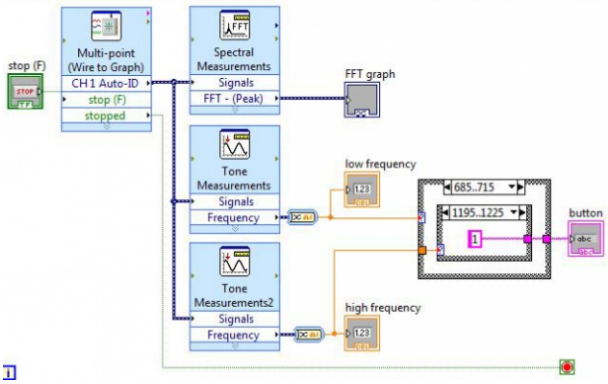
- Verbindungen und Linien
- Variablen und Konstanten
- Funktionen und Funktionsbausteinen

Der Signalfluss eines FBD-Programms verläuft von links nach rechts.
Die Abarbeitungsreihenfolge der Bausteine lässt sich individuell festlegen.

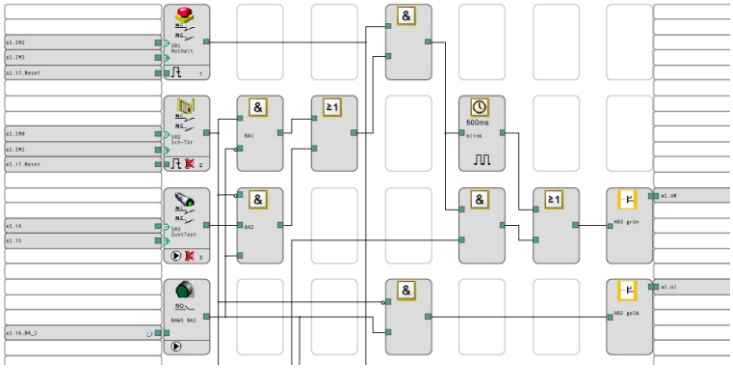
Beispiel 1: 100 zu WERT1 Addieren, Ergebnis auf +/-10000 begrenzen und in WERT2 schreiben



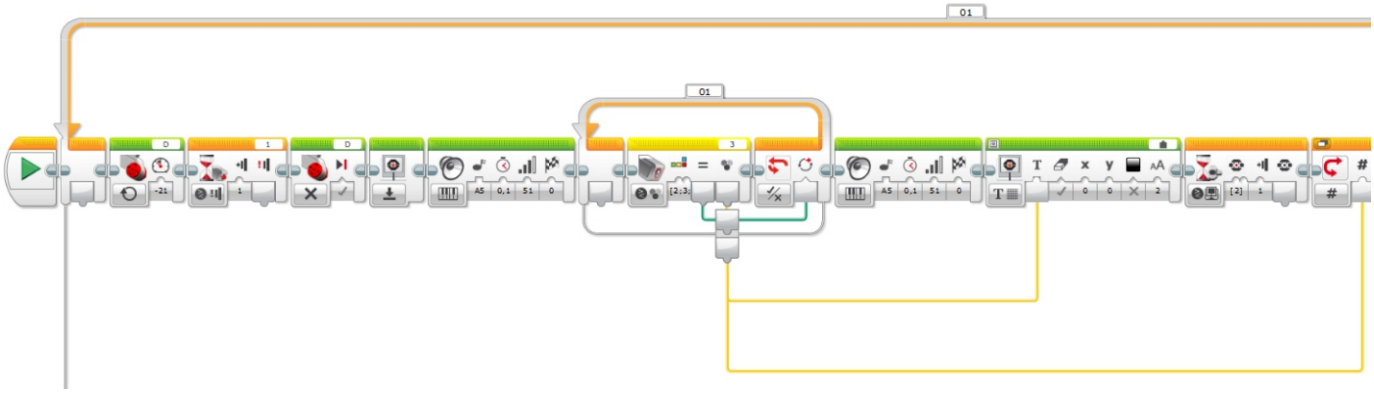
Beispiel 2: LabView



Beispiel 3: Pilz - PNOZmulti



Beispiel 4: LEGO MINDSTORM Eduction EV3



SFC	Sequential Function Chart	AS	Ablaufsprache	Grafik

Die Ablaufsprache dient zur Programmierung einer speicherprogrammierbaren Steuerung in Form eines Petri-Netzes. Unter Siemens STEP 7 ist die Ablaufsprache als S7-GRAPH bekannt.

Eine Ablaufsteuerung ist eine Kette von Steuerungsschritten, welche durch Weiterschaltbedingungen (Transitionen) miteinander verbunden sind.

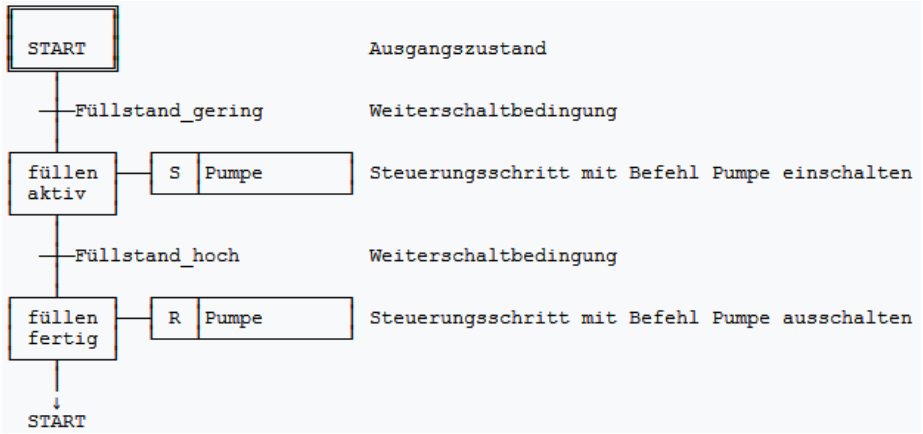
Direkt an den Schrittsymbolen werden Befehle, auch Aktionen genannt, eingebunden. Beim Erreichen eines Steuerungsschrittes mit angehängter Aktion wird ein Ausgang der Steuerung gesetzt und damit ein Aktor betätigt (z. B. Pumpe ein, Magnetventil öffnen) oder es wird ein interner Programmsprung durchgeführt.

Die Transitionen, welche sich zwischen den einzelnen Steuerungsschritten befinden, werden jeweils mit einem Eingangsbit verknüpft (Weiterschaltbedingung), das von einem Sensor auf einen Eingang der Steuerung gelangt ist (wie z. B. Grenztaster für Füllstand, Endschalter, Lichtschranke). Ein Signalwechsel des Eingangsbits löst das Schalten der Transition aus, sodass der nachfolgende Steuerungsschritt aktiviert wird.

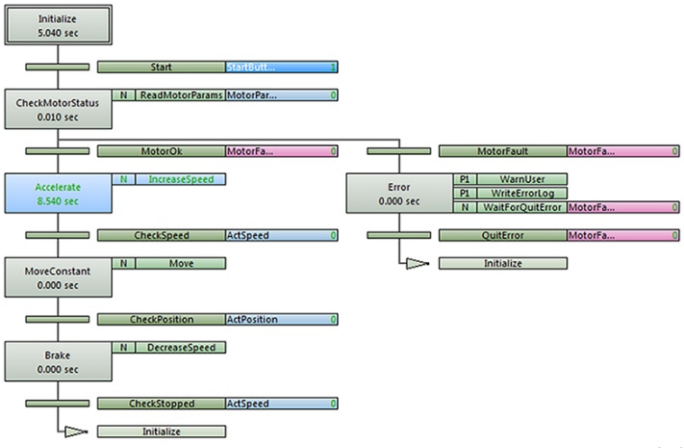
Der erste Steuerungsschritt besitzt normalerweise keine eigene Aktion, da dieser auch als Initialisierungsschritt gilt. Das bedeutet, dass das Programm beim ersten Betriebszyklus mit dem initialisierten Schritt startet. Jeder Steuerungsschritt ist durch eine Transition mit dem nächsten Steuerungsschritt verbunden.

Die Ablaufsteuerung ist vor allem bei Großanlagen weit verbreitet. Trotz Normung bietet nicht jeder SPS-Hersteller die Ablaufsprache als Programmiersoftware an.

Beispiel 1:



Beispiel 2: Sigmatek LASAL CLASS



ST	Structured Text	ST	Strukturierter Text	Text

Die Syntax der Sprachelemente ähnlich denen der Hochsprache Pascal und es wird wie bei allen Sprachen der EN 61131-3 bei Schlüsselwörtern keine Unterscheidung zwischen Groß- und Kleinschreibung gemacht (Case Insensitive).

ST bietet mehr Strukturierungsmöglichkeiten als AWL und löst diese daher immer mehr ab. Komplexe Algorithmen und mathematische Funktionen lassen in ST übersichtlicher und schneller programmieren.

ST-Programme bringen nach der Kompilierung meist einen erhöhten Speicherbedarf mit sich. Dies kann auf kleineren SPS durch schnelleres Erreichen der Speichergrenzen durchaus zu Problemen führen.

Beispiel 1: IF-Statement

```
IF (MASCHINE_EINGESCHALTET = TRUE) THEN
  AUSGANG1 := EINGANG1 AND EINGANG2;
ELSE
  AUSGANG1 := FALSE;
END_IF;
```

Beispiel 2: WHILE-Statement

```
If RST_Tisch then
  M[15] := false;
  k := 1;
  while Daten[X[2]+k] <> 60 and Daten[X[2]+k] <> 99 and k < 49 do
    k := k+1;
  end_while;
  Daten[X[2]] := int_to_uint(k); (* neu setzen für 2. Vorgang *)
  Wait := false;
end_if;
```

Beispiel 3: Pilz PAS4000

```
PROGRAM POU_1
VAR

OUT1_0 AT %q*:BOOL;      // Motor Rechtslauf
OUT5_0 AT %q*:BOOL;      // Motor Schleichfahrt
OUT5_1 AT %q*:BOOL;      // Stopper senken
OUT5_2 AT %q*:BOOL;      // Stopper heben
OUT5_3 AT %q*:BOOL;      // Magnetventil vereinzeln
OUT6_0 AT %q*:BOOL;      // Stopper hinten
OUT7_0 AT %q*:BOOL;      // FUS - langes Gurtband
OUT7_1 AT %q*:BOOL;      // LED gelb

(* Kommentar Beginn    -    Kommentar ENDE*)

Timer1();   //Timer aufrufen

IF NOT IN0_2 & NOT IN4_3  THEN;     // Freigabe und Taster TEST  //IN0_2 & IN4_3

    OUT1_0:=FALSE;       // Motor Rechtslauf
    OUT5_0:=FALSE;       // Motor Schleichfahrt
    OUT5_1:=FALSE;       // Stopper senken
    OUT5_2:=FALSE;       // Stopper heben
    OUT5_3:=FALSE;       // Magnetventil vereinzeln
    OUT6_0:=TRUE;        // Stopper hinten
    OUT7_0:=FALSE;       // FUS - langes Gurtband
    OUT7_1:=TRUE;        // LED gelb
    OUT7_2:=FALSE;       // Leuchte gelb
    OUT7_3:=FALSE;       // Signal fertig

    schritt:=0;
    timer1.in:=FALSE;   //Timer zurücksetzen

ELSE;

OUT7_1:=FALSE;       // LED gelb
CASE SCHRITT OF
```