verfügbar. Das Schlüsselwort „stat“ taucht in den Symboltabellen auf. Intern wird das Schlüsselwort „VAR“ verwendet.

Anfangswerte in S7 sind auch für Durchgangsparameter zulässig (Beispiel 3.2).

Adr.	Dekl.	Name	Typ	Anfangswert	Kommentar
0.0	in	starter	BOOL	TRUE	Boolescher Eingangsparameter Anfangswert = TRUE
0.1	in	ein	BOOL	FALSE	Boolescher Eingangsparameter Anfangswert = FALSE
2.0	in	stueck	INT	0	Integer Eingangsparameter Anfangswert = 0
0.1	in_out	pl	BOOL	FALSE	Boolescher Durchgangsparameter
4.0	stat	statisch	WORD	W#16#000F	Statische Wortvariable Anfangswert = 000F
6.0	out	ausgang	BOOL	FALSE	Boolescher Ausgangsparameter Anfangswert = FALSE

Beispiel 3.2 Variablendeklarationstabelle

3.2 Bausteinkonzepte nach IEC 61131 und STEP7

Beide Bausteinkonzepte weisen einige Parallelen auf, unterscheiden sich jedoch in wesentlichen Eigenschaften. Während nach IEC die Bindung an das Betriebssystem über zyklische bzw. ereignisgetriebene *Task* organisiert ist, existiert eine spezielle Bausteinart, die *Organisationsbausteine* (OB) unter S7. Beide decken jedoch weitgehend ähnliche Funktionalität ab. Der Task ist ausschließlich für den Aufruf der zugeordneten Bausteine verantwortlich. OB’s sind hingegen programmierbare, durch das Betriebssystem aufgerufene Bausteine. Bausteininstanzen sind im Code durch den Instanznamen identifiziert. S7 verwendet zum Fb einen Instanzdatenbaustein, der immer zusätzlich anzugeben ist. Auf die S7-Datenbausteine, die keinen Code beinhalten, verzichtet die IEC vollständig. Für die Handhabung größerer Datenmengen sind dafür Arrays und Strukturen vorgesehen. Eigenheit der S7-Bausteine ist der Name, der sich aus Bausteintyp und einer Nummer zusammensetzt (z. B. FC 1 für Funktion 1). Ferner kann ein symbolischer Name in der Symboltabelle vergeben sein. Nach IEC gibt es nur symbolische Namen.

Beide Systeme unterstützen die Sprachen AWL (Anweisungsliste), ST (Strukturierter Text), KOP (Kontaktplan), FBS (Funktionsbausteinsprache) und AS (Ablaufsprache).

3.2.1 IEC 61131

Als oberste Maxime gilt der *objektorientierte Ansatz der Datenkapselung*, der es ermöglicht, selbständige, unabhängige, seiteneffektfreie und damit wiederverwendbare oder mehrfachverwendbare Einheiten zu schaffen. Ebenso wie Datentypen nur eine typgerechte Nutzung der Daten zulassen, sorgt die Kapselung dafür, dass Daten (lokale) nur innerhalb der POE (Programmorganisationseinheit → Baustein der IEC) zur Verfügung stehen, in der sie deklariert sind. Somit ist ausgeschlossen, dass Daten versehentlich überschrieben werden, sie unterliegen ausschließlich der Kontrolle ihrer POE. Ausnahmen bilden die Schnittstellen zur Umwelt und globale Variablen. Alle POEs besitzen den gleichen prinzipiellen Aufbau. Als wichtigste Elemente sind der Deklarationsteil und der Bausteinrumpf, der den Quellcode zur Lösung der Steuerungsaufgabe aufnimmt, zu nennen.

Vielfach wird vom *objektorientierten Ansatz* gesprochen, dieser begründet sich mit der Tatsache, dass der Bausteintyp einer instanzierbaren POE (Programm und Funktionsbaustein) nur einmal zu deklarieren bzw. anzulegen ist und durch die Instanzbildung theoretisch beliebig viele *Instanzen* erzeugbar sind. Das heißt, die Datenstruktur einer Instanz wird ausnahmslos von ihr verwaltet und mit dem Algorithmus des Typs ausgewertet, den alle Instanzen dieses Typs verwenden. Daten und Prozeduren werden aus Anwendersicht als zusammengehörig angesehen und verwaltet.

Die folgende Auflistung zeigt die *IEC-Bausteintypen* und ihre wesentlichen Eigenschaften.

POE	**Aufruf durch**	**Sprachen**	**Parameter / Rückgabe**
Programm (Pg)	Task	AWL, ST, FBS, KOP, AS	optional / optional
Funktionsbaustein (Fb)	Task, Pg, Fb	AWL, ST, FBS, KOP, AS	optional / optional
Funktion (Fun)	Pg, Fb, Fun	AWL, ST, FBS, KOP	mind. VAR_INPUT / mind. FunName

Prinzip der Instanz

Durch das Erstellen eines Programms bzw. eines Funktionsbausteins wird im Grunde nur festgelegt, wie mit welchen Daten umzugehen ist. Um diese beiden POE-Typen mehrfach verwenden zu können, ist eine Arbeitskopie der Daten, die *Instanz* genannt wird, mit eigenem Namen (Bezeichner) versehen, unter dem ihre privaten Speicherplätze bereitstehen, zu erzeugen. Damit stehen je Instanz die vereinbarten lokalen Variablen bereit, die somit vor Überschreiben (lokale Variablen) durch erneuten Aufruf des gleichen Bausteintyps (mehrfach verwendeter Bausteintyp), jedoch unter anderem Instanznamen, geschützt sind. Die Verwaltung lokaler Variablen erfolgt nur unter dem Instanznamen.

Die Deklaration einer POE kann man sich als eine Arbeitsanweisung vorstellen, die besagt: Wird eine Instanz dieses POE-Typs erzeugt, sind die im Deklarationsteil angegebenen Variablen unter dem Instanznamen zu verwalten und entsprechender Speicherplatz ist zu reservieren.

Diese Daten sind bei Aufruf der Instanz mit den Anweisungen im Bausteinrumpf zu bearbeiten (vgl. Bild 3.2).

Die *Schnittstelle* zur Umgebung wird über die vereinbarten (deklarierten) Formalparameter (Ein- und Ausgänge der Instanz) bereitgestellt, die zur Unterscheidung durch Nennung des Instanznamens (auch Instanzvariable genannt) gemeinsam mit dem Parameter anzusprechen sind. Deklarationsmäßig unterliegen sie der gleichen Behandlung wie ihre nahen Verwandten, die lokalen Variablen.

Beispielsweise muss eine Einschaltverzögerung nur einmal als Funktionsbausteintyp deklariert werden. Durch Instanzbildung lassen sich beliebig viele Kopien der Daten erzeugen, die alle unabhängig voneinander mit unterschiedlichen Zeitwerten zu beliebigen Zeitpunkten ablaufen können. Der Rumpf existiert nur einmal. Das SPS-Betriebssystem kennt durch die Instanzbildung den Bausteintyp und weiß somit, dass die Daten mit entsprechendem Rumpf zu bearbeiten sind. Eine Instanz bzw. ihr frei wählbarer Name eines POE-Typs wird vielfach auch als *Instanzvariable* bezeichnet.

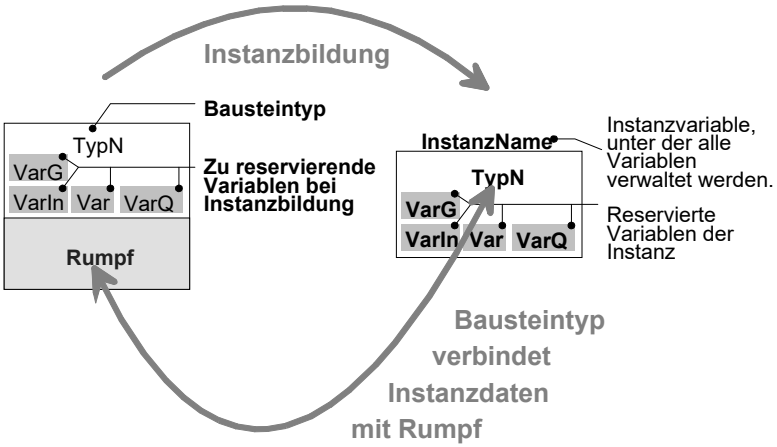

Bild 3.2 Schematischer Zusammenhang zwischen Instanz und Bausteintyp

Auf global deklarierte Variablen haben nur *Programme* und *Funktionsbausteine* Zugriff, nicht jedoch Funktionen.

Die wesentlichen Eigenschaften von Programmen und Funktionsbausteinen (instanzierbare POE) bestimmt ihr Typ genau so, wie Variableneigenschaften im Wesentlichen von ihrem Datentyp bestimmt sind.

Funktionen sind eher als mathematische Funktionen zu sehen, die, sofern immer mit den gleichen Parametern aufgerufen, auch gleiche Ergebnisse liefern, da sie nicht auf gespeicherte Vergangenheitswerte zurückgreifen.

In Tabelle 3.12 sind Beispiele zur Deklaration der POE, ihre Instanzbildung und die meist verwendeten Aufruf-Formen angegeben. In grafischen Sprachen stellt das grafische Symbol den Aufruf dar. Die IEC beschreibt einige Standard-Funktionsbausteine und Funktionen

Tabelle 3.12 POE Deklaration, Instanzierung und Aufruf

Programm	Funktionsbaustein	Funktion
Deklaration		

PROGRAM PgName
VAR
  V1, V2: INT;
  IFbTyp: FbTyp;
END_VAR
	FUNCTION_BLOCK FbTyp
	VAR_INPUT (*Formalparameter*)
	  IN1, IN2: INT;
	END_VAR
	VAR_OUTPUT
	  Q : INT;
	ENDVAR
		FUNCTION FunName : BOOL
		VAR_INPUT (*Formalparameter*)
		  IN1, IN2, IN3 : BOOL;
		END_VAR

Implementierung		
LD V1 ST IFbTyp.IN1 LD INT#7 ST IFbTyp.IN2 CAL IFbTyp (*Aufruf*) LD IFbTyp.Q ST V2 END_PROGRAM	(*Funktionsbausteinrumpf in ST*) Q := IN1 + IN2; END_FUNCTION_BLOCK	(*Funktionsbausteinrumpf in AWL*) LD IN1 AND IN2 AND IN3 ST FunName END_FUNCTION

Tabelle 3.12 *(Fortsetzung)*

Instanzbildung		
Erfolgt durch Zuordnung einer Task	VAR(*Instanzbildung*) Instanz : FbTyp; END_VAR	nicht möglich
Aufruf in AWL		
Aufruf beim Programm durch Task des Betriebssystems möglich.	LD Wert1 ST Instanz.IN1 LD INT#7 ST Instanz.IN2 CAL Instanz (*Aufruf*) LD Instanz.Q ST Ergebnis	LD Wert1 FunName Wert2, Wert3 ST Ergebnis
Aufruf in ST		
s. o.	Instanz(IN1 := INT#7, IN2 := Wert1); Ergebnis := Instanz.Q;	Ergebnis := FunName (Wert1, Wert2, Wert3);

3.2.2 STEP7

S7 kennt zwei Arten von Bausteinen: *Codebausteine* enthalten Programmanweisungen (Code). Alle Bausteine außer DB‘s sind Codebausteine. *Datenbausteine* (DB‘s) enthalten ausschließlich Daten, die Zugriffe durch Anweisungen in Codebausteinen gewähren. Codebausteine können hinsichtlich des Programmablaufs in zwei Gruppen unterschieden werden. *Organisationsbausteine* (OB’s), deren Aufruf das Betriebssystem bei bestimmten Ereignissen veranlasst. Aufrufe aller *anderen Codebausteine* (Funktionsbausteine und Funktionen) sind durch Anweisungen im Anwenderprogramm ausgelöst. Abweichend von der IEC, wo es unzulässig ist, innerhalb einer Funktion Funktionsbausteine aufzurufen, ist dies unter S7 gestattet.

Organisationsbausteine (OB): OB‘s können nicht durch Programmanweisungen gestartet werden (vgl. Tabelle 3.14). Ihr Aufruf wird durch das SPS-Betriebssystem beim Eintreten von festgelegten Ereignissen bzw. zyklisch veranlasst. Ein OB höherer Priorität (Pri., vgl. Tabelle 3.13) unterbricht die Ausführung eines OB mit niedrigerer Priorität. Die Fortsetzung des Unterbrochenen erfolgt am Ende des höherprioren OB‘s. Sind Bausteine gleicher Priorität gleichzeitig auszuführen, werden diese sequentiell abgearbeitet. Niedrigste Priorität hat OB1. Die Anzahl der verfügbare OB‘s hängt von der verwendeten CPU ab.

Tabelle 3.13 Organisationsbausteine

Pri.	OB	Name	Systemereignis
-	1	Zyklus	am Zyklusende und nach Anlauf (OB100ende)
	10	Uhrzeitalarm	Uhrzeit oder Datum
⇩	35	Weckalarm (Zeitinterrupt)	periodisch jeweils nach einer definierten Zeit
	40	Prozessalarm(I/O-Interrupt)	bei bestimmten Signalen der I/O-Peripherie
	80	Zykluszeitfehler	definierte OB1-Zykluszeit überschritten
	100	Anlauf	beim Anlauf der SPS, d. h. bei Stop=>Run
+	122	Baugruppenfehler	Fehler beim Baugruppenzugriff erkannt

Das S7-Betriebssystem-Konzept ersetzt das Multitasking-System der IEC (vgl. Abschnitt 1.5.5).

Funktion (Function, FC): Die S7-Funktion (vgl. Tabelle 3.14) entspricht weitgehend der IEC-Funktion. Beim Funktionsaufruf können Parameter übergeben werden, am Ende der Ausführung kann die Funktion dem aufrufenden Baustein Parameter zurückliefern. Anzahl und Datentype der Parameter werden bei der Erstellung der Funktion deklariert. Ferner ist die Deklaration temporärer Variablen möglich. Variablen der FC sind nicht statisch.

Datenbaustein (Data Block, DB): Auf Datenbausteine (strukturierter Speicher) kann durch Programmanweisungen in Codebausteinen zugegriffen werden. Die Datenstruktur wird bei der Erstellung des DBs festgelegt. Die IEC kennt keine Datenbausteine. DBs werden nicht im Sinne von Aufrufen und Ausführen gebraucht, da es keine Codebausteine sind. Für einen Datenzugriff ist der DB zunächst zu öffnen bzw. aufzuschlagen. Der Zugriff bleibt solange möglich, bis ein neuer aufgeschlagen wird.

Tabelle 3.14 POE Deklaration, Instanzierung und Aufruf

Organisationsbaustein	**Funktionsbaustein**	**Funktion**
Deklaration		
Variablendeklarationstabelle OB1: Dekl. Name Typ Anfangswert temp v1 BOOL FALSE temp v2 BOOL FALSE	Variablendeklarationstabelle für FB1: Dekl. Name Typ Anfangswert in IN1 BOOL FALSE stat v BOOL FALSE out Q BOOL FALSE	Variablendeklarationstabelle für FC1: Dekl. Name Typ Anfangswert in IN1 BOOL FALSE in IN2 BOOL FALSE out Q BOOL FALSE
Implementierung //Bausteinaufruf CALL FB 1, DB 1 IN1 := Wert1 Q := Ergebnis	//Funktionsbausteinrumpf in AWL U IN1 U v = Q = v	//Funktionsrumpf in AWL U IN1 U IN2 = Q
Instanzbildung		
nicht möglich	Symboltabelle: Name Typ myFb FB 1 instDb DB 1	nicht möglich
Aufruf in AWL		

Aufruf erfolgt durch das Betriebssystem	//Absoluter Aufruf
	CALL FB 1, DB 1
	IN1 := Wert1
	Q := Ergebnis
	//Symbolischer Aufruf
	CALL myFb, instDb
	IN1 := Wert1
	Q := Ergebnis
		CALL FC 1
		IN1 := Wert1
		IN2 := Wert2
		Q := Ergebnis

Funktionsbaustein (Function Block, FB): Beim Aufruf (vgl. Tabelle 3.14) des FBs ist stets ein Datenbaustein anzugeben, der den statischen Speicher der aufgerufenen Instanz des FBs zur Verfügung stellt. Dieser DB heißt *Instanz-Datenbaustein*. Instanz-Datenbausteine bzw. dessen Struktur legt der Compiler bei der Erzeugung des FB automatisch an. Für jede Instanz ist ein DB zu erzeugen. Der S7-FB ist funktionell ähnlich dem der IEC, der Aufruf ist jedoch anders.

Der IEC Fb-Type entspricht etwa dem S7-FB und der Instanz-Datenbaustein erfüllt die Funktion der IEC-Instanz. Durch Verwendung mehrerer Instanz-Datenbausteine mit einem FB existieren mehrere unabhängige Instanzen vom gleichen Typ im Sinne der IEC.

Systemfunktion (SFC) und *Systemfunktionsbaustein* (SFB): Das sind spezielle Bausteine des Betriebssystems. Ihr Aufruf gestaltet sich wie bei ganz normalen FCs oder FBs. Typische Funktionen sind Diagnosefunktionen, Alarmbearbeitung, Uhrzeit oder Betriebsstundenzähler.

3.3 IEC 61131 und STEP7 bei der Ergebnisbildung

Anweisungen sind in ihrem Aufbau genau spezifiziert und bestehen aus der Sprungmarke, dem AWL-Operator bzw. der Funktion, dem Operanden oder einer Operandenliste und einem Kommentar, wobei nicht sämtliche Elemente in einer Anweisung vorhanden sein müssen. Bild 3.3 zeigt den grundsätzlichen Aufbau einer AWL-Anweisung.

Die *S7-AWL-Anweisung* ist identisch aufgebaut, jedoch beginnt der Anwenderkommentar mit // und endet mit dem Zeilenende. Außerdem darf die Marke eine Länge von 4 Zeichen nicht überschreiten. Die Marke ist zudem Case-Sensitiv, dies bedeutet zwischen Groß- und Kleinschreibung wird unterschieden. Die Marken „Marke1:“ und „marke1:“ stellen zwei völlig unterschiedliche Sprungziele dar.

Marke :	Operator	Operand	(*Anwenderkommentar*)

```
SprungZiel :
             LD      Var1          (*Sprung vorbereiten*)
             AND     % IX1.0
             JMPC    SprungZiel    (*Sprung ausführen*)
```

IEC-AWL

		Op	AE	Kommentar	
				(*AE ist nach Einschalten undefiniert	*)
LD	INT#10	10	10	(*AE-Speicherformat INT, 10 laden	*)
NE	INT#12	12	1	(*Vergleich AE und Operand (12), Ergebnis	*)
				(*steht in AE (BOOL) zur Verfügung	*)
ST	Ergebnis	1	1	(*AE speichern in Variable »Ergebnis «	*)

Bild 3.3 IEC-AWL-Anweisung