Einführung in die IEC 1131-3 Programmiersprachen und Elemente (Praktische Beispiele und Vorführungen) (version 2001)

Die IEC 1131 ist ein internationaler Standard für speicherprogrammierbare Steuerungen. Der Teil 3 der IEC 1131 wurde bereits 1992 veröffentlicht und definiert im wesentlichen 5 Programmiersprachen:

- Anweisungsliste
- Kontaktplan
- Funktionsbausteinsprache
- Strukturierter Text
- Ablaufsprache

Seit 1995 bieten die ersten SPS-Hersteller bereits Programmiersysteme an, die gemäß der IEC 1131-3 definiert sind. In diesem Tutorial werden verschiedene Hersteller anhand praktischer Vorführungen demonstrieren, wie sie die IEC 1131-3 in ihre Programmiersysteme implementiert haben.

Die Vorträge gliedern sich in:

- Gemeinsame Elemente der IEC 1131-3 (von R. Wohlschläger, EURO-Matsushita)
- Ablaufsprache (von G. Bonjean, CJ International)
- Anweisungsliste (von H. Einwag, Klöckner Moeller)
- Strukturierter Text (von A. Oksas, Softing)
- Funktionsbausteinsprache (von W. Brendel, infoteam software)
- Kontaktplan (von G. Süss, ObjectAutomation)

Ziel der IEC 1131-3 ist es, komplexe Programme zu strukturieren und dadurch übersichtlicher zu gestalten und einfacher zu programmieren sowie die Wiederverwendbarkeit bereits programmierter Programmteile, um damit Entwicklungszeit und Fehler einzusparen.

Gemeinsame Elemente der IEC 1131-3

(Ralf Wohlschläger, EURO-Matsushita Electric Works AG)

Neben den 5 Programmiersprachen der IEC 1131-3 ist es auch notwendig, eine gemeinsame Basis der Zeichen, der Zusammenarbeit und der Strukturierung zu definieren. Aus diesem Grund werden unter Punkt 2 der Norm die gemeinsamen Elemente der IEC 1131-3 definiert.

Darunter fallen:

- Zeichensatz
- Datentypen
- Variablen
- POEs, Programm-Organisations-Einheiten
- AS Elemente
- Konfigurationselemente: Tasks

Der Zeichensatz

Die Definition des Zeichensatzes in der Norm dient dazu, daß alle Hersteller nur bestimmte Buchstaben und Zeichen in ihren Systemen verwenden. Bereits hierdurch muß sichergestellt sein, daß eine Kompatibilität der Systeme herrscht.

Wie bekannt ist, werden in den unterschiedlichen Ländern Buchstaben oder auch Zeichen verwendet, die es in anderen Ländern nicht gibt, so wie in Deutschland z. B. die Umlaute ö, ä, ü und andere.

Da die Definition des Zeichensatzes nur für die Hersteller von Programmiersystemen interessant ist und nicht für den Anwender, wird in diesem Fachbeitrag nicht näher darauf eingegangen. Wer sich dafür interessiert, kann in der IEC 1131-3 unter dem Punkt 2.1 weiteres nachlesen.

Die Datentypen

Ein wesentlicher Vorteil der IEC 1131-3 ist die Wiederverwendbarkeit und Austauschbarkeit von Programmen und Programmteilen. Dies ist nicht möglich, wenn man mit festen Adressen und Speicherbereichen im Programm arbeitet.

Die Wiederverwendbarkeit wird nur dadurch gewährleistet, daß auch Daten und Variablen unabhängig vom Programmteil ausgetauscht werden können. Dazu definiert die IEC Datentypen und Variablen, die als Platzhalter für die tatsächlichen Adressen dienen. Die Zuweisung der tatsächlichen Adressen zu den Datentypen und den Variablen erfolgt für das Gesamtprogramm in der globalen Variablenliste und in den Teilprogrammen bzw. in den Funktionsbausteinen und den Funktionen in den dazugehörigen „Headern“, den lokalen Variablenlisten.

Um der Vielzahl der möglichen Anwendungen gerecht zu werden, die von Einzelkontakten bis hin zu „REAL-Zahlen“ geht, gibt es Datentypen der Länge 1 Bit bis zu 64 Bit.

IEC 1131-3 **Elementare Datentypen**

Nr.	Schlüsselwort	Datentyp	Bits
1	BOOL	Boolesche	1
2	SINT	kurze ganze Zahl (short integer)	8
3	INT	ganze Zahl (integer)	16
4	DINT	doppelte ganze Zahl (double integer)	32
5	LINT	lange ganze Zahl (long integer)	64
6	USINT	vorzeichenlose kurze ganze Zahl (unsigned short integer)	8
7	UINT	vorzeichenlose ganze Zahl	16
8	UDINT	vorzeichenlose doppelte ganze Zahl	32
9	ULINT	vorzeichenlose lange ganze Zahl	64
10	REAL	reelle Zahl	32
11	LREAL	lange reelle Zahl	64
12	TIME	Zeitdauer	systemabhängig
13	DATE	Datum (nur)	systemabhängig
14	TIME_OF_DAY oder TOD	Uhrzeit (nur)	systemabhängig
15	DATE_AND_TIME oder DT	Datum und Uhrzeit	systemabhängig
16	STRING	variabel-lange Zeichenfolge	systemabhängig
17	BYTE	Bit-Folge der Länge 8	8
18	WORD	Bit-Folge der Länge 16	16
19	DWORD	Bit-Folge der Länge 32	32
20	LWORD	Bit-Folge der Länge 64	64

Die Definition dieser Datentypen dient auch dazu, Fehler zu vermeiden, da gewisse Funktionen im Programm nur mit gewissen Datentypen zulässig sind. Da die

Programmiersysteme dies Nachprüfen müssen, ist sichergestellt, daß Datenmischung und daraus resultierende Fehler nicht zustande kommen können.

Die Variablen

Um nun die Zuweisung der tatsächlichen Speicherorte in der Steuerung mit den Datentypen im Programm auszuführen, gibt es die sogenannten Variablen.

Die Variable weist den SPS-Ort einem Datentypen zu und gibt zugleich an, welche Art der Variablen sie ist und was mit dieser Variablen erlaubt ist. Zur Art der Variablen gibt es die Variablen-Adressierung. Anhand dieser kann man feststellen, ob es sich z. B. um eine Eingangs-, eine Ausgangs- oder eine Merkervariable handelt, die in der SPS nur als Platzhalter dient. Genauso wird in der Adressierung bereits ersichtlich, ob es sich um einen Bit, Wort, Doppelwort oder anderen Datentyp handelt.

IEC 1131-3

Variablen Adressierung

Nr.	Präfix	Bedeutung
1	I	Speicherort Eingang
2	Q	Speicherort Ausgang
3	M	Speicherort Merker
4	X	(Einzel-)Bit-Größe
5	kein	(Einzel-)Bit-Größe
6	B	Byte-(8 bit) Größe
7	W	Wort-(16 bit) Größe
8	D	Doppelwort-(32 bit) Größe
9	L	Langwort-(64 bit) Größe

Falls nicht anders deklariert, muß der Datentyp einer direkt adressierten Variablen eine „(Einzel-)Bit“-Größe vom Typ BOOL sein.

Ralf Wohlschläger, EURO-Matsushita Electric Works

Anhand des obigen Bildes kann man z. B. eine Eingangsvariable definieren, die am Eingang 1 der Steuerung sitzt, indem man das Prozentzeichen hinzunimmt, d. h. eine externe tatsächliche Zuweisung. Somit erhält man die Variable „%IX0.1“, d. h. Eingangs-Bit auf Platz 1 der SPS.

Zusätzlich wird bei der Deklaration der Variablen aber auch angegeben in welchen Teilen des Programms die Variable verwendet werden darf und wie sie sich verhalten muß:

Variablen Deklaration

Schlüsselwort	Gebrauch der Variablen
VAR	Innerhalb der Organisationseinheit (POE)
VAR_INPUT	Von außerhalb geliefert, nicht innerhalb der Organisationseinheit veränderbar
VAR_OUTPUT	Von der Organisationseinheit nach außen geliefert
VAR_IN_OUT	Von außerhalb geliefert, kann innerhalb der Organisationseinheit geändert werden
VAR_EXTERNAL	Von der Konfiguration geliefert via VAR_GLOBAL, kann innerhalb der Organisationseinheit geändert werden
VAR_GLOBAL	Deklaration von globalen Variablen
VAR_ACCESS	Deklaration von einem Zugriffspfad
RETAIN	Gepufferte Variablen
CONSTANT	Konstante (Variable kann nicht geändert werden)
AT	Zuweisung des Speicherorts

Im obigen Bild sieht man z. B, daß die VAR_INPUT eine Variable ist, die von außen in eine POE geliefert wird und in der POE weiterverarbeitet werden kann.

Eine andere Variable, die VAR_OUTPUT, ist ein Ergebnis einer Funktion oder eines Funktionsbausteines, das wieder zurück an das Programm gegeben wird. Wieder eine andere Variable, z. B. die RETAIN, sagt aus, daß der Wert dieser Variablen bei Spannungsausfall erhalten bleibt und, wenn die Spannung wieder eingeschalten wird, weiterverwendet werden kann.

Anhand dieser Definitionen ist nun sichergestellt, daß Fehler vermieden werden und daß Programmteile wiederverwendet werden können. Der Datentyp der Variablen, der Adreßplatz der Variablen und die Verwendungsart der Variablen sind nun festgeschrieben.

POEs, Programm-Organisations-Einheiten

POEs sind abgeschlossene Programmeinheiten, die Werte von außen erhalten, im Programmteil diese Werte bearbeiten und Ergebnisse nach außen abliefern. Als POEs gibt es die Funktionen, Funktionsbausteine und das Programm.

Funktionen sind Unterprogramme ohne eigenen Speicher, d. h. es können mehr Eingangsvariablen in die Funktion gegeben werden, es ist allerdings nur 1 Ausgang der Funktion möglich, jeweils abhängig von den Eingangswerten. In der Funktion können keine Daten gespeichert werden.

Die Funktionsbausteine dagegen sind Unterprogramme mit eigenem Speicher. Sie können mehrere Eingänge aber auch mehrere Ausgänge haben und somit verschiedene Ergebnisse nach außen liefern. Zusätzlich können Daten im Funktionsblock gespeichert werden und somit ist auch ein verändertes Ergebnis bei gleichen Eingängen möglich, da interner Speicher addiert, subtrahiert o.ä. werden kann.

Die Funktionen und die Funktionsbausteine können vom Hersteller der SPS geliefert werden. Ein großer Vorteil ist allerdings, daß Anwender sich auch eigene Funktionen und Funktionsbausteine programmieren können und diese in allen anderen Projekten wiederverwenden können, d. h. einmal programmiert, einmal getestet, immer wieder verwendbar.

Die oberste Ebene der POEs ist das Programm. Allerdings können auch hier in einem Projekt verschiedene Programme in unterschiedlichen IEC 1131-3 Sprachen geschrieben werden, die dann beim Übersetzen in ein einziges Gesamtprogramm zusammengelinkt werden.

Eine POE besteht aus einem „Header“. In ihm werden die Variablen deklariert: Die Eingangsvariablen, die Ausgangsvariablen und die intern verwendeten Variablen. Mit diesem Datensatz kann jetzt der Programmbaustein arbeiten. Weiterhin bestehen die POEs aus einem „Body“, dem eigentlichen Programmteil mit Befehlen.

Die POEs können in jeder der 5 IEC 1131-3 Sprachen geschrieben werden und sie ermöglichen die Wiederverwendbarkeit der Software von den Programmen bis hin zu den Funktionsbausteinen und Funktionen.

In Bibliotheken abgespeichert bilden die POEs die Basis für projektunabhängige und neutrale Wiederverwendung der Software. Man kann sich das ähnlich einem Hardware-IC vorstellen, mit fest definierten Schnittstellen, die beliebig ein- und ausgesteckt werden können. Die Schnittstellen werden hier im „Header“ der POE, in der Variablen Deklaration, festgelegt.

IEC 1131-3

IEC 1131-3 Standard Funktionen

- Bit-String Funktionen (AND, OR, XOR, NOT, SHL, SHR, ROL, ROR)
- Numerische Funktionen (ADD, SUB, MUL, DIV, MOD, EXPT, ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN)
- Typenumwandlung (e.g. USINT_TO_DINT, BOOL_TO_BYTE)
- Auswahl-Funktionen (SEL, MIN, MAX, LIMIT, MUX)
- Vergleichs-Funktionen (GT, GE, EQ, LT, LE, NE)
- String Funktionen (LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND)

Hersteller- und anwenderprogrammierte Funktionen möglich

IEC 1131-3 Standard Funktionsbausteine

- Bistable (SR, RS, SEMA)
- Flankenerkennung (R_TRIG, F_TRIG)
- Zähler (CTU, CTD, CTUD)
- Zeitrelais (TP, TON, TOF, RTC)

Hersteller- und anwenderprogrammierte Funktionsbausteine möglich

Ralf Wohlschläger, EURO-Matsushita Electric Works

Die Elemente der Ablaufsprache

Die Ablaufsprache wird meistens als eine der 5 Programmiersprachen der IEC 1131-3 gesehen. Tatsächlich ist sie allerdings ein Element der Strukturierung, eine Unterteilung des Programms in eine Art Flußdiagramm, wobei in den Diagrammblöcken eine der anderen 4 Sprachen verwendet wird. Zu dieser Einteilung in eine Struktur kennt man im wesentlichen 3 Elemente der Ablaufsprache:

1. Die Schritte:

Sie beinhalten das eigentliche Programm, die sogenannten Aktionen. Ziel ist es, daß ein gewisses Ergebnis in diesem Schritt erreicht wird, das notwendig ist, um in den nächsten Schritt überzuwechseln und im Programm fortzufahren. Ist dieses Ergebnis erreicht, so wird eine Weiterschaltbedingung erfüllt und somit eine Transition geschaltet.

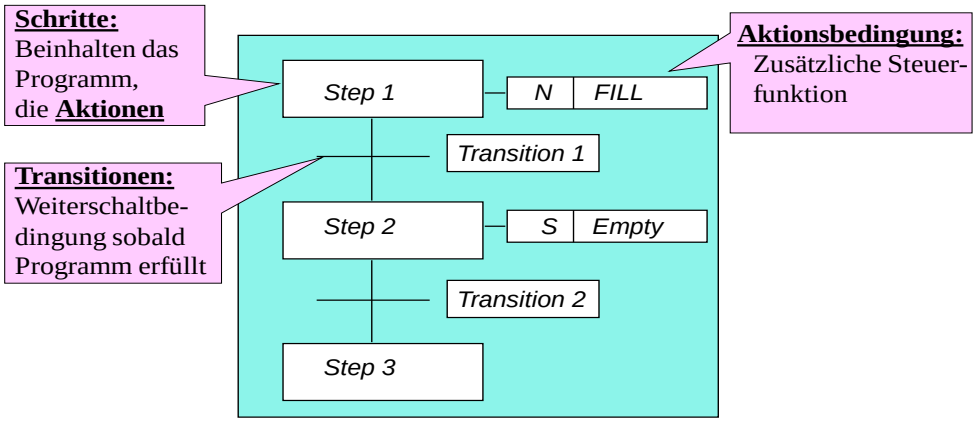
2. Transitionen

Die Transitionen sind Weiterschaltbedingungen, die ein gewisses Ereignis aus dem vorangegangenen Schritt abwarten und, sobald dieses erfüllt ist, in den nächsten Schritt weiterschalten.

3. Aktionsbedingungen

Zusätzlich können den einzelnen Schritten sogenannte Aktionsbedingungen zugeordnet werden. Neben der ganz normalen Abarbeitung eines Schrittes, können z. B. auch Bedingungen gesetzt werden, die Schritte zeitverzögert starten, zeitbegrenzt abarbeiten oder speichern lassen. Diese Aktionsbedingungen dienen dazu, weitere Programmteile zu sparen und verkürzen damit das Programmieren.

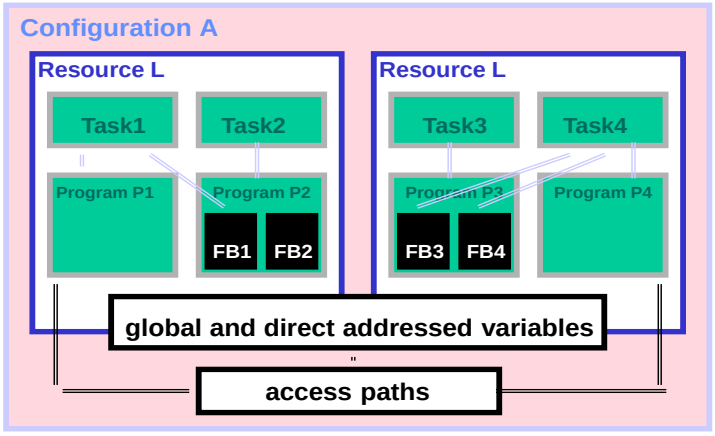
AS Elemente



Konfigurationselemente (Tasks):

Um das Zusammenspiel der einzelnen Variablen, POEs und Tasks zu definieren, gibt es das sogenannte Programmmodell, das im Punkt “Konfigurationselemente der Norm“ beschrieben ist.

Software Modell



Neben der Übergabe der Variablen und dem Aufruf bzw. der Abarbeitung von Funktionsbausteinen und Programmen, ist es auch notwendig, im sogenannten „Taskmanager“ einzutragen, welche Programme aktiv sein sollen. Laut Norm wäre hier auch ein „Multitask“- und ein „Multiresource“-Betrieb möglich, was heißt, daß sich verschiedene

SPSen in einem Softwaremodell befinden. Die üblichen Programmiersysteme beschäftigen sich allerdings noch mit einzelnen Tasks und für den Anwender ist es derzeit ausreichend, wenn er die programmierten POEs im „Taskmanager“ einträgt bevor er das Programm auf die SPS lädt.

Vorteil dieses zusätzlichen Eintragens in den „Taskmanager“ ist, daß einzelne POEs auch unabhängig voneinander getestet werden können, indem andere POEs vom „Taskmanager“ ausblendbar sind. Somit ist ein einfaches Testen und schrittweises Zusammenfügen des Hauptprogrammes möglich.

Zusammenfassung:

Zusammenfassend kann man also sagen, daß die gemeinsamen Elemente der IEC 1131-3 dazu dienen, die Programmiersprachen zusammenzufügen, die Strukturierung zu gewährleisten und auch eine Wiederverwendbarkeit der Programme zu ermöglichen. Das wird erreicht durch die Variablen mit den fixen Datentypen, die dann in die wirklichen Speicherplätze der SPS übergeben werden, sowie durch die Strukturdefinition der Programmorganisationseinheiten und der Ablaufsprache, die ein übersichtlicheres Programmieren ermöglichen.