Fachbeitrag zum Tutorial

Einführung in die IEC1131-3 Programmiersprache und Elemente

anläßlich der SPS/IPC/Drives 1999

Teil 3: Programmiersprache Anweisungsliste (AWL)

Dipl. Ing. Heribert Einwag
Moeller GmbH
53115 Bonn
Tel: 0228 602-0
Internet: http://www.moeller.net

Einleitung:

Die Anweisungsliste (AWL) hat im Bereich der Programmiersprachen für Speicherprogrammierbare Steuerungen (SPS) eine lange Tradition. Vor dem Kontaktplan und dem Funktionsplan ist die Anweisungsliste speziell im deutschen Sprachraum die am meisten eingesetzte Programmiersprache. Zunehmende Anforderungen aus den Bereichen Visualisierung, Kommunikation, Daten- und Rezeptverwaltung sowie Regelungstechnik haben aber in der Vergangenheit auch deutlich die Grenzen der bisherigen AWL Implementierungen gezeigt. Es fehlten einfach die programmsprachlichen Mittel, um diese Anforderungen adäquat in einem Anwenderprogramm umsetzen zu können. Als Reaktion hierauf entwickelten viele Anbieter von Programmiersystemen Spracherweiterungen oder Sonderhardware, um diese Probleme zu lösen.
Der 1993 erstmals verabschiedete internationale IEC 1131-3-Standard hat neben anderen Programmiersprachen auch die Anweisungsliste neu definiert. Mit den hier vorhandenen Möglichkeiten, die wohl bis heute von keinem Anbieter vollständig realisiert worden sind, stehen mächtige Sprachmittel zur Verfügung, um auch künftige Automatisierungsprobleme effektiv lösen zu können.

Bevor die Umsetzung der neuen Möglichkeiten der IEC1131-3 am Beispiel der Moller Programmiersoftware Sucosoft S40 gezeigt wird, erfolgt eine kurze Zusammenfassung der wesentlichen Unterschiede von bisherigen Programmiersystemen zu einem IEC-1131-3 Programmiersystem.

Datenmodell

Bisherige Programmiersysteme waren durch ein globales Datenmodell gekennzeichnet. Das heißt, daß von jeder Stelle des Anwenderprogramms auf alle Variablen (bisher vorzugsweise Merker oder Datenbausteine) zugegriffen werden konnte. Dies führt dazu, daß gerade bei größeren Anwendungen mit mehreren tausend Merkern eine sehr strenge Programmierdisziplin für die Verwaltung und Benutzung dieses Speicherbereich notwendig war; für die Verwaltung in der Hinsicht, daß der Anwender nachhalten mußte, welche Merker noch frei sind. Bezüglich der Benutzung entstand immer wieder das Problem, daß unberechtigte Zugriffe auf Merker eine Fehlerursache waren, die gerade in größeren Anwendungen langwierige Fehlersuche zur Folge hatte. Eine Wiederverwendung von bisher erstellten Programmteilen scheiterte meist daran, daß der benötigte Speicherbereich schon von anderen Programmteilen benutzt wurde. Selbst bei symbolischer Programmierung war es nicht unwahrscheinlich, daß der gleiche Symbolname schon an anderer Stelle verwendet wurde.
Ein zentrales Element der IEC1131-3 ist die Bereitstellung von geschützten Datenbereichen. Variablen werden nicht mehr auf absolute Adressen abgebildet sondern werden vom Programmiersystem frei in dem zur Verfügung stehenden SPS-Speicher verwaltet. Weiterhin ist der Zugriff auf diese Daten normalerweise nur dort möglich, wo diese Variable auch angelegt d.h. deklariert wurde. Dazu wurden in der IEC drei verschiedenen ProgrammOrganisationsBausteintypen (POE) definiert: das Programm, der Funktionsbaustein und die Funktion. Ein Zugriff von außerhalb auf Daten innerhalb einer POE ist nur über die in der Norm festgelegten Übergabe- oder Kopplungsverfahren möglich. Ein ”versehentliches” Überschreiben von Variablen ist also ausgeschlossen.

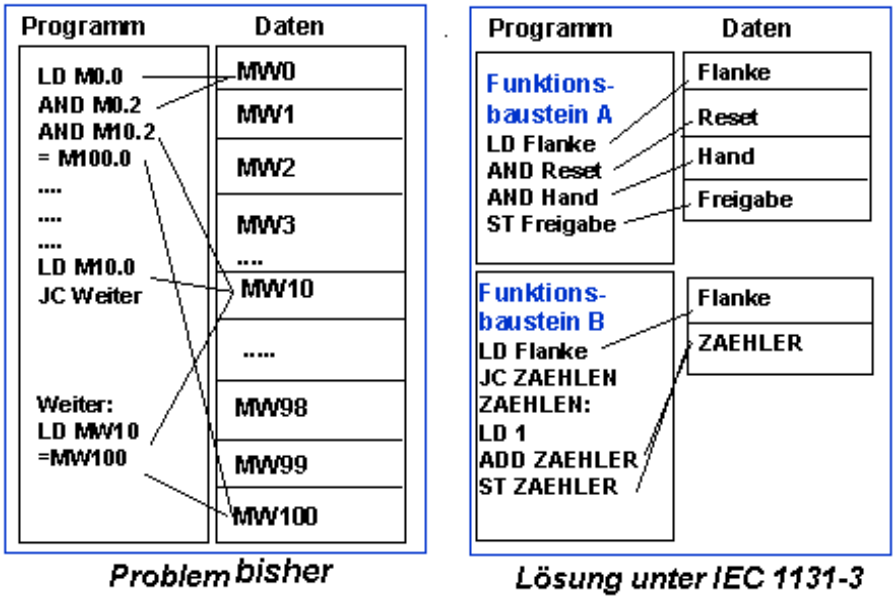


Abbildung 1: Unterschiede in der Speicherverwaltung

Durch die Instanzierung von Funktionsbausteinen wird quasi eine Kopie des dort benutzten Datenbereichs vom Programmiersystem angelegt. Bestehende Funktionsbausteine können damit völlig unabhängig in jedem anderen Programm verwendet werden. Damit ist es erstmals möglich, Standardfunktionsbausteine zu erstellen und ohne Rücksicht auf bereits benutzte Datenbereiche wiederzuverwenden.
Durch eine hierarchische Aufrufstruktur lassen sich sehr komplexe Funktionsbausteine erstellen. Auch hier ermöglicht es erst das automatische Speichermanagement des Programmiersystems, daß der Anwender sich ganz auf die Umsetzung seines Problems konzentrieren kann und sich nicht um das Rangieren von Merkern und Datenbausteinen kümmern muß. Dies ist vor allem dann wichtig, wenn irgendwo noch eine zusätzliche Variable benötigt wird, die problemlos eingefügt werden kann.

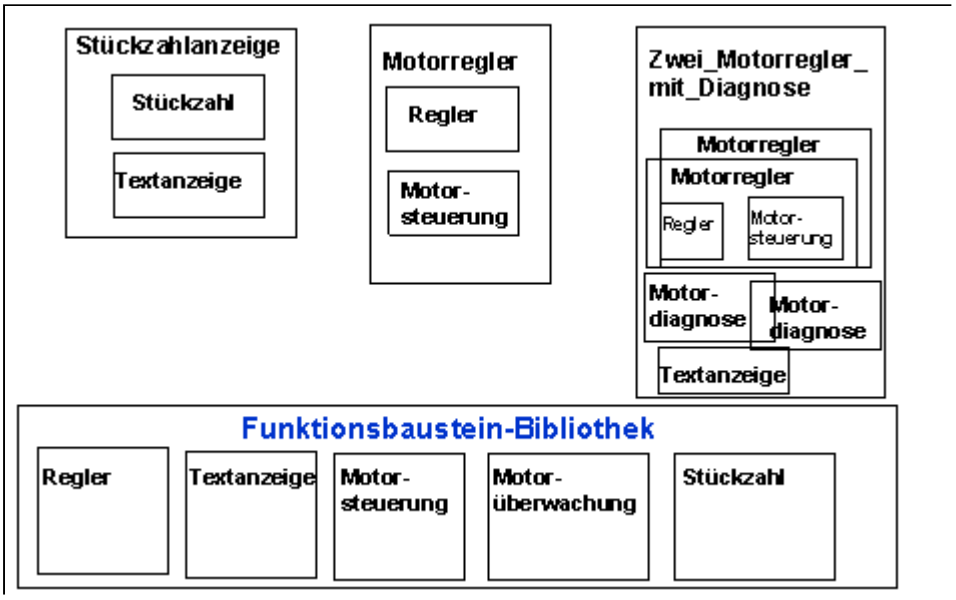


Abbildung 2: Aufbau komplexer Funktionsbausteine

Datentypen

Eine weitere wichtige Neuerung ist die Einführung von Datentypen. Diese sind notwendig, um die Größe und die Funktionsweise von Variablen festzulegen. In bisherigen Programmiersystemen war der Merkerbereich meist in Bit, Byte, Wort und evtl. Doppelworte strukturiert. Ob bei der Addition zweier Merkerworte aber mit oder ohne Vorzeichen gearbeitet wurde, war dem Anwenderprogramm nicht anzusehen. Hier half nur die Herstellerdokumentation. Weiterhin war auf diesem Datenstrukturen die Abbildung oder Verarbeitung von Texten schlecht möglich. Uhrzeiten wurden als mehrere Bytewerte gespeichert; die Subtraktion zweier Zeiten war schon eine Programmierarbeit, die nicht in zwei Zeilen zu erledigen war. Gleiches gilt für die Rezepturverarbeitung, für die es ebenfalls keine adäquaten Mittel gab.
In der IEC1131-3 stehen nun für diese Fälle die entsprechenden Datentypen zur Verfügung. Arithmetische für vorzeichenbehaftete und vorzeichenlose Operationen, Stringdatentypen für die Textverarbeitung, Strukturen und Felder (auch mehrdimensionale) für die Speicherung und Verwaltung von Daten.

Wert1:UINT; Wert2:UINT; Wert3:UINT; LD Wert1 ADD Wert2 ST Wert3	Wert1:INT; Wert2:INT; Wert3:INT; LD Wert1 MUL Wert2 ST Wert3	Zeit1:TIME; Zeit2:TIME; Zeit3:TIME; LD Zeit1 SUB Zeit2 ST Zeit3	Feld1,Feld2: ARRAY[1..100] of USINT; LD Feld1 EQ Feld2 JMPC WerteSindGleich ..... WerteSindGleich:
Vorzeichenlose Addition	Vorzeichenbehaftete Multiplikation	Subtraktion zweier Zeiten	Vergleich zweier Felder

Abbildung 3: Beispiel für die Anwendung verschiedener Operatoren

Befehlssatz

Der Befehlssatz der AWL nach IEC 1131-3 besteht aus Operatoren, Standard-Funktionen und Standard-Funktionsbausteinen, die auch in den anderen Programmiersprachen der Norm verwendet werden könne, aber hier eben in einer speziellen AWL-Notation aufgerufen werden.
Das Wichtige an der Normung der AWL ist zum einen die Tatsache, daß der Anwender endlich vom Problem zahlloser “AWL-Dialekte” mit geringen, in der Auswirkung dann aber zeitraubenden, Unterschieden in der Schreibweise befreit wird. Zum anderen ist festzuhalten, daß die Operatoren für die Verknüpfung unterschiedlichster Datentypen einheitlich sind; die Überprüfung auf Kompatibilität der verknüpften Operanden übernimmt das System. So gibt es z.B. nur einen Addier-Befehl ADD anstelle von zahlreichen Varianten für jeden Datentyp (ADD_INT, ADD_TIME, etc.).

Umsetzung der IEC 1131-3 in der Sucosoft S40

Die IEC definiert den Sprachumfang der in ihr beschriebenen Programmiersprachen sowie die Programmstrukturierung und die Daten-Kommunikationsmechanismen. Weitere Teile, die jedoch für die gesamte Programmerstellung notwendig sind, werden nicht beschrieben. Dies sind zum Beispiel die Projektverwaltung, die Gerätekonfiguration, die eigentliche Programmerstellung mit Test und Inbetriebnahme und die Dokumentation.

Projektverwaltung:

Die Projektverwaltung verwaltet verschiedene Projekte des Benutzers. In jedem Projekt können mehrere Programme für einen oder mehrere Steuerungstypen verwaltet werden. Unterverzeichnisse erlauben es dem Anwender,
individuelle Projektstrukturen anlegen, um seine Programmteile (POEs) nach seinen Wünschen abzulegen. Funktionen wie das Sichern

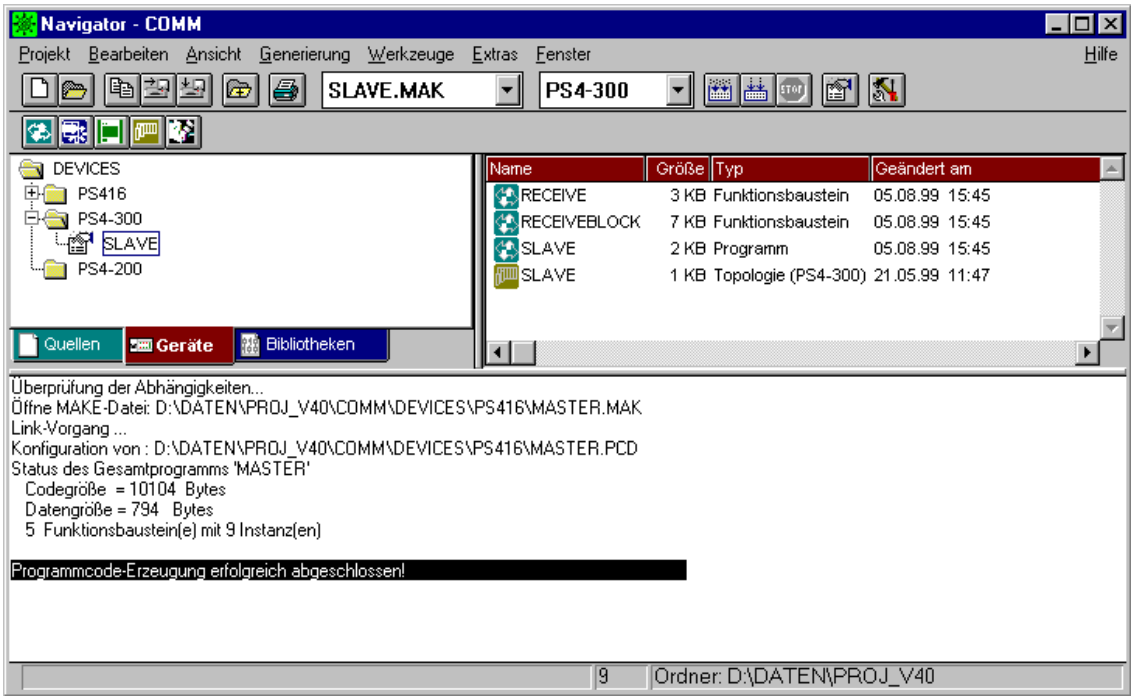


Abbildung 4: S40 Navigator

Topologiekonfigurator:

Im Topologiekonfigurator definiert der Anwender die Hardwarestruktur seines Automatisierungsgerätes. Die Sucosoft S40 stellt für die Netzwerke Suconet-K, AS-Interface (ASI) I und Profibus-DP einen integrierten Konfigurator mit einheitlicher E/A-Adressierung zur Verfügung. Aus Sicht des Anwenders gibt es also keine Unterschied in der Adressierung von lokalem oder dezentralen E/A, auch über Netzwerktypen hinweg.

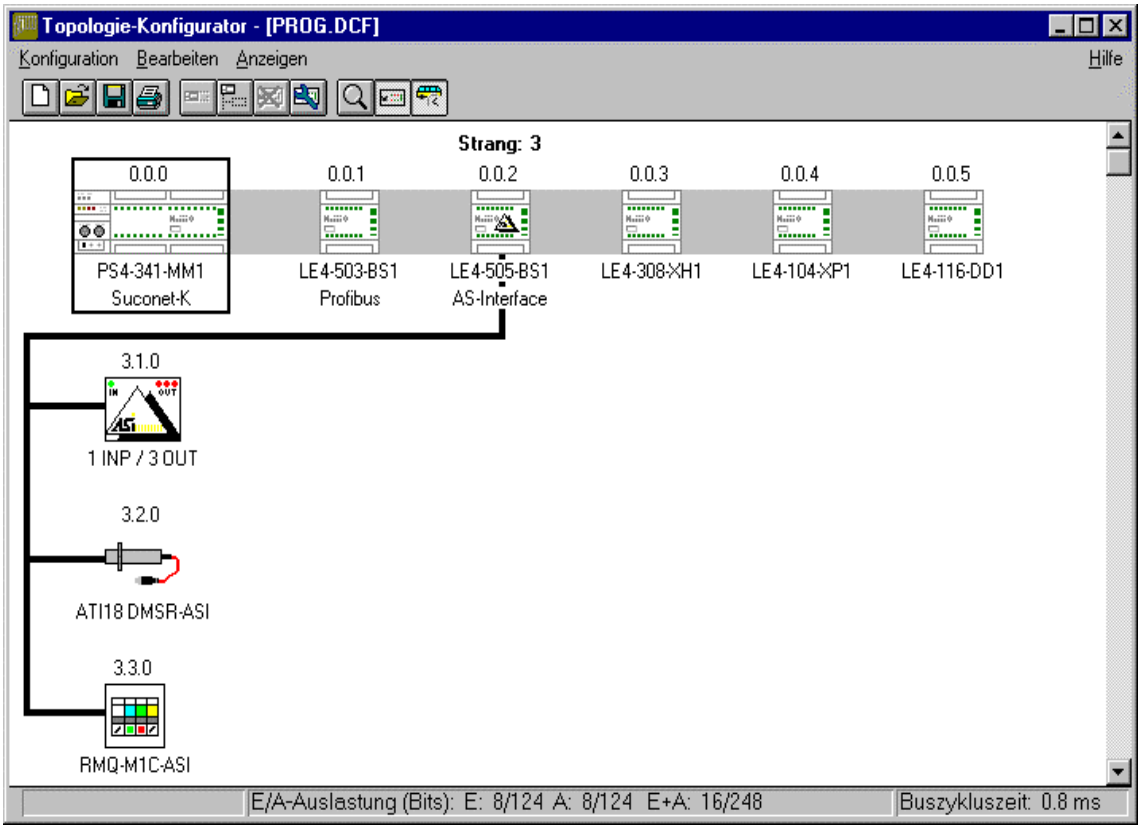
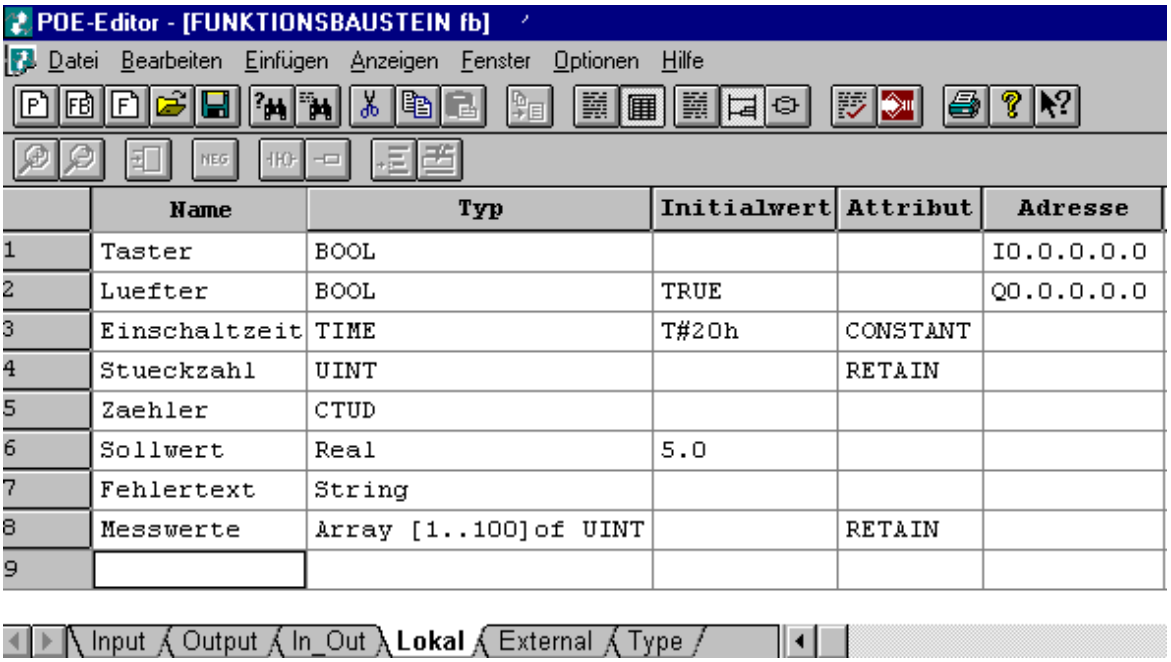


Abbildung 5: Topologiekonfigurator vereint unterschiedliche Bussysteme

POE-Editor

Das Schreiben einer POE gliedert sich in zwei Schritte, die Definition der Variablen, dies ist der Deklarationsteil und das Erstellen des Programms, dies ist der Anweisungsteil. Beim Anlegen einer neuen POE wird der Benutzer zunächst gefragt, ob er ein Programm, einen Funktionsbaustein oder eine Funktion erstellen möchte. Eine Festlegung der Programmiersprache ist nicht notwendig, da in der Sucosoft S40 jederzeit zwischen den Programmiersprachen AWL, KOP und FBS gewechselt werden kann.
Hat man den gewünschten POE-Typ ausgewählt, so folgt das Erstellen der Variablen. In der Sucosoft S40 stehen dazu zwei verschiedene Betriebsarten zur Verfügung. Der Syntaxgesteuerte Variableneditor stellt eine tabellenorientierte Eingabe zur Verfügung, während der freie Variableneditor Variablendeklarationen nach der IEC-

Syntax erlaubt. Beide Modi sind ineinander überführbar. Der Vorteil des Syntaxgesteuerten Variableneditors besteht darin, daß einerseits die Eingabe der Variablen ähnlich der der bekannten Zuordnungslisten ist, es gibt nur noch zusätzlich Möglichkeiten wie z. B. die Vergabe eines Initialwert oder die Festlegung des Remanenzverhaltens. Zum anderen werden hier je nach Variablentyp kontextsenitiv nur diejenigen Felder zum Ausfüllen angeboten, die für diesen Variablentyp auch sinnvoll sind. Eine lokale Variable in einer Programm-POE kann zum Beispiel eine physikalische Adresse zugeordnet werden, einer Input-Variablen nicht. Darüberhinaus können hier einfach die neuen Möglichkeiten der IEC genutzt werden wie z.B. die Initialisierung von Variablen , die Definition des Verhaltens wie CONSTANT oder RETAIN und anderes mehr. Die Auswahl von Datentypen und Herstellerbausteinen erfolgt ebenfalls kontextsensitiv bezogen auf den ausgewählten Steuerungstyp. In einer Kleinsteuerung stehen z. B, keine 32 Bit Datentypen zur Verfügung. Soweit möglich erfolgt die Auswahl der Daten über Auswahlfelder. Auch alle selbsterstellten Funktionsbausteine werden als Datentyp angeboten.



	Name	Typ	Initialwert	Attribut	Adresse
1	Taster	BOOL			I0.0.0.0.0
2	Luefter	BOOL	TRUE		Q0.0.0.0.0
3	Einschaltzeit	TIME	T#20h	CONSTANT	
4	Stueckzahl	UINT		RETAIN	
5	Zaehler	CTUD			
6	Sollwert	Real	5.0		
7	Fehlertext	String			
8	Messwerte	Array [1..100]of UINT		RETAIN	
9					

Abbildung 6: Deklaration von Variablen

Die Definition von eigenen Datentypen, die projektweit verwendet werden, erfolgt separat von den Projekt-POEs. Als Beispiel sei hier die Definition von Strukturen und Feldern gezeigt.

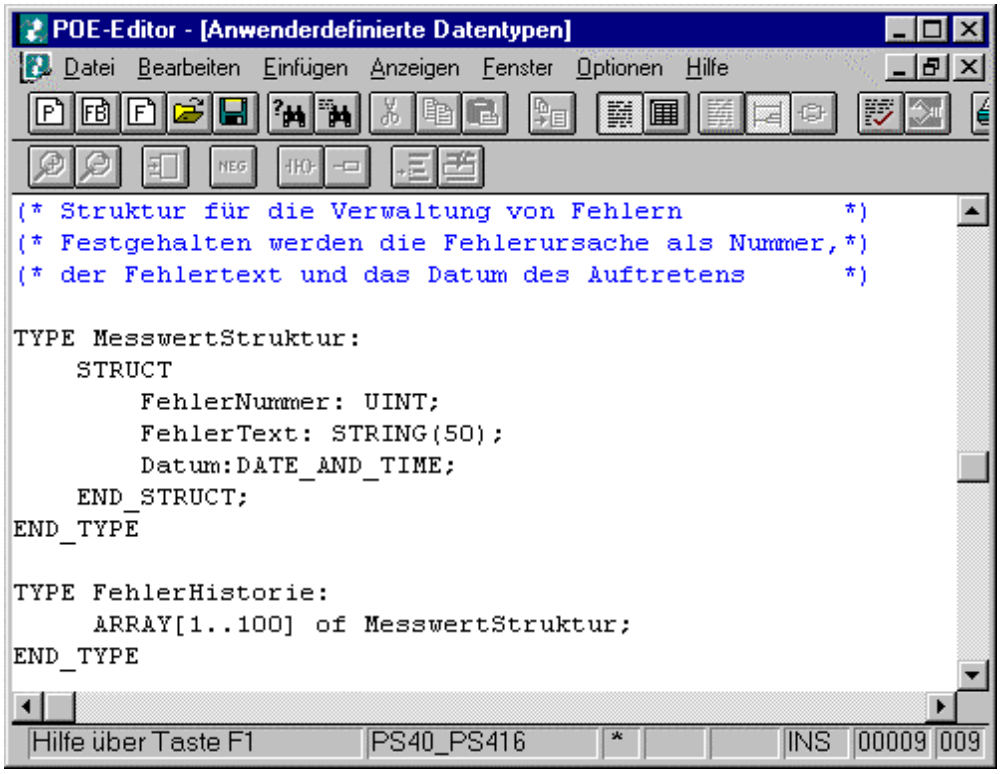


Abbildung 7: Definition von projektglobalen Datentypen

Im Anweisungsteil wird in einem zweiten Schritt das eigentliche Anwenderprogramm erstellt. Hier werden die zuvor angelegten Variablen benutzt, es können aber natürlich jederzeit neue Variablen hinzugefügt oder Variablen gelöscht werden.
Für die Auswahl von Operatoren, Variablen und Bausteinen stehen hier kontextsensitive Auswahlboxen zur Verfügung. Zur besseren Übersicht trägt auch die farblich unterschiedliche Darstellung von Kommentaren, IEC-Schlüsselwörtern etc. bei, die vom Anwender selbst festgelegt werden kann. Über das Kontextmenü läßt sich zu einer Variablen schnell die Deklarationszeile auffinden, die Hinweis auf z.B. Datentyp, Adresse und Beschreibung liefert. Auch eine schnelle Nachdeklaration neuer Variablen ist hier möglich. Mit der Syntaxprüfung lassen sich viele Fehler direkt finden, ein Doppelclick auf die Fehlerzeile setzt den Cursor direkt an die fehlerhafte Stelle. Über die Kontexthilfe lassen sich sofort weitere Informationen zu benutzten Bausteinen abrufen. Wird der Aufruf eines Funktionsbausteines markiert und die Hilfe aufgerufen (z.B. durch Betätigen der F1 Taste), so wird der zugehörige Funktionsbausteintyp ermittelt und die zugehörige Hilfe angezeigt. Neben der Beschreibung der Datentypen für die Ein- und Ausgänge ist hier auch eine funktionale Beschreibung des Bausteine vorhanden. Diese Funktion ist auch für selbsterstellte Funktionsbausteine verfügbar.

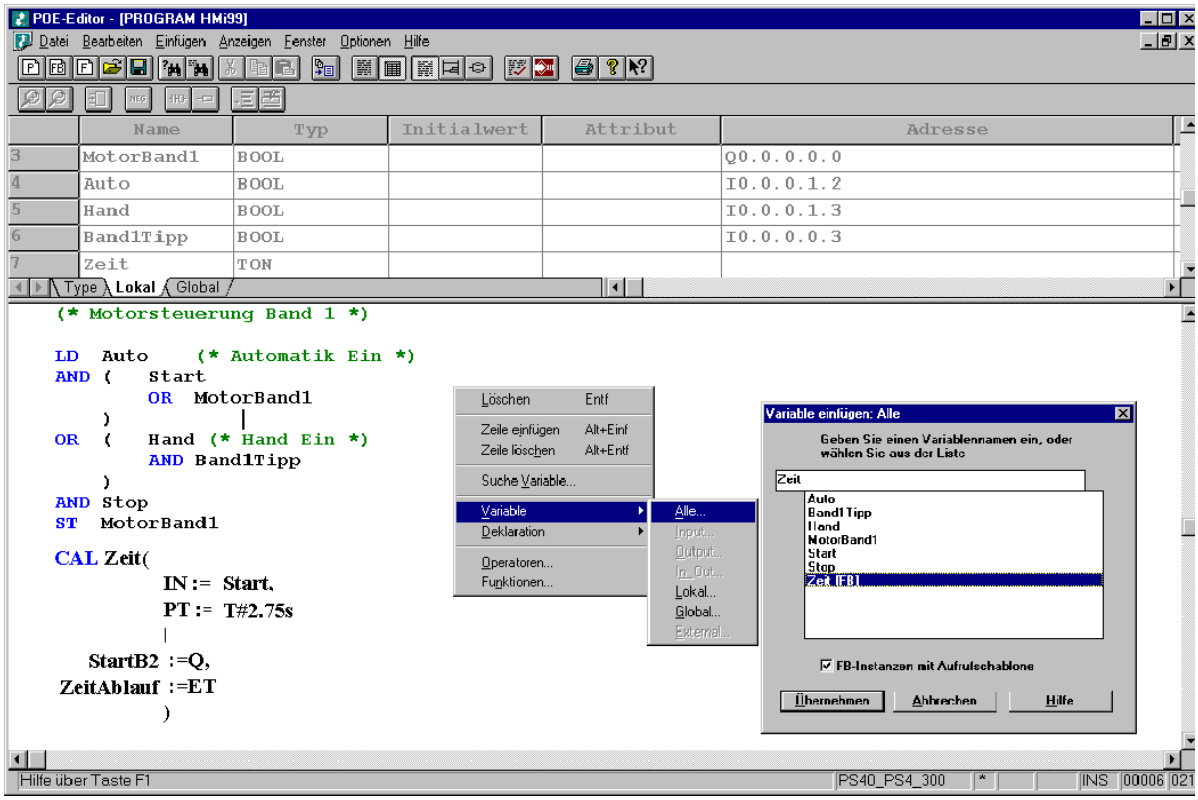


Abbildung 8: Kontextmenüs helfen bei der Programmerstellung

Codegenerierung:

Ist das Programm mit seinen Funktionsbausteinen fertig erstellt, wird es für den Transfer auf die Steuerung übersetzt. Diese Übersetzung ist steuerungsspezifisch, da zum Beispiel je nach Steuerung mehr oder weniger Funktionalität und Speicher zur Verfügung stehen. Die PS416 stellt bis zu 1 MB für ein Anwenderprogramm zur Verfügung, die Kompaktsteuerung PS4-201 dahingegen 64 KB.
Da in einem Projekt mehrere Programme verwaltet werden können, werden bei der Codegenerierung der Name des zu übersetzenden Programms sowie dessen Gerätekonfiguration angegeben. Ausgehend von der Programm-POE werden alle benutzten Funktionsbausteine und Funktionen übersetzt und zu einem ablauffähigen Programm zusammengebunden. Weitere Parameter wie zyklische oder periodische Abarbeitung oder die maximale Zykluszeit können ebenso eingegeben werden. Bei der Übersetzung werden auch alle im Programm verwendeten Ein- und Ausgänge gegen die in der Gerätekonfiguration konfigurierten Baugruppen geprüft. Dies ist vor allem in verteilten Anlagen wichtig, wo nahezu beliebige E/A Konstellationen entstehen können. Durch diesen Test ist ausgeschlossen, daß im Programm Peripherie verwendet wird, die physikalisch gar nicht vorhanden ist.

Test und Inbetriebnahme

Der Transfer des erzeugten Programms auf die Steuerung kann über eine Punkt-zu-Punkt-Verbindung zwischen dem PC und der SPS oder, in einem dezentralen Aufbau mit mehreren Steuerung, über das Suconet-K Netzwerk erfolgen. Dabei übernimmt die Kopfsteuerung die Funktion eines Gateway, es ist also keine gesonderte Netzwerkkarte für den PC notwendig. Selbstverständlich können alle Test- und Inbetriebnahmefunktionen auch über ein Modem bzw. bei Verwendung geeigneter Terminalserver auch über Ethernet erfolgen.
Die Übertragungsrate kann bis zu 57600 Baud betragen. Nach Start der Anwendung kann das Anwenderprogramm dann im Online-Editor getestet und zur Laufzeit ohne Anhalten des Programms geändert werden. Dazu wird zuvor im Instanzbaum der Funktionbaustein, und bei Mehrfachverwendung, die zugehörige Instanz ausgewählt.
Der Instanzbaum ist die hierarchische Darstellung der Aufrufstruktur der einzelnen Programmelemente, also der POEs.

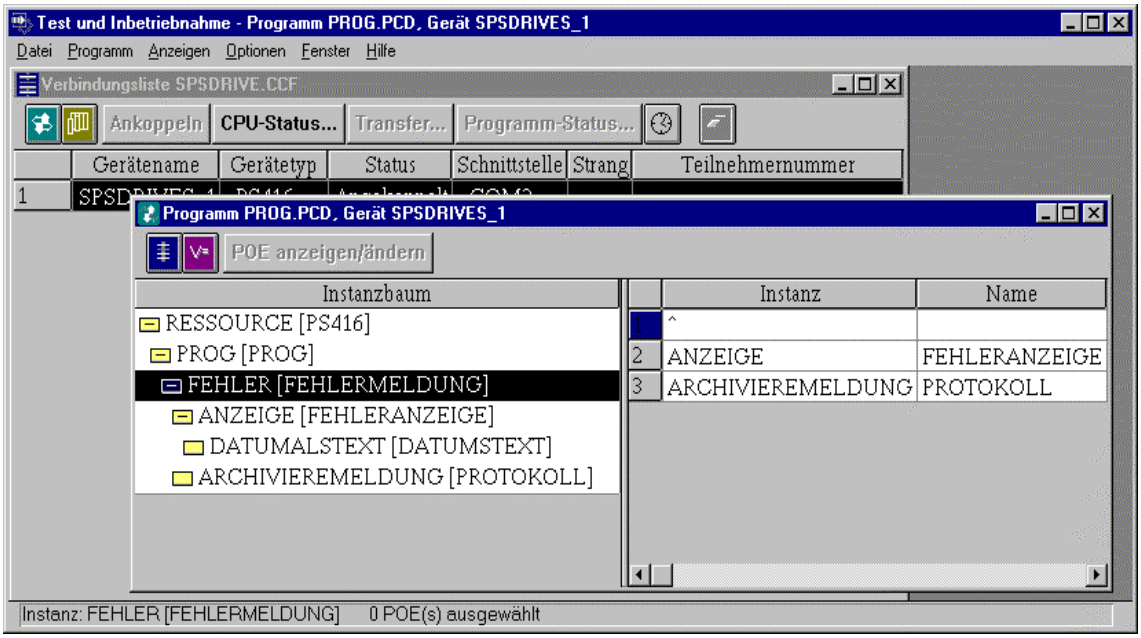


Abbildung 9: Auswahl der Instanz für die Online-Anzeige

Im Online Editor können die Zustände der Variablen im Anwenderprogramm betrachtet und Online-Änderungen durchgeführt werden. Das Löschen und Ändern ist hier ebenso möglich wie das Einfügen neuer Anweisungen. Die Visualisierung der Werte erfolgt gemäß dem verwendeten Datentyp. Somit werden vorzeichenbehaftete Variablen mit, und nicht vorzeichenbehaftete Variablen ohne Vorzeichen dargestellt. Für binäre Variablen besteht die Möglichkeit der Umschaltung von Bit- auf Hexadezimal-Anzeige. Einzelne Variablen lassen sich - auch aus mehreren Instanzen oder Funktionsbausteinen - im Variablenfenster zusammenfassen. Damit lassen sich die Zustände von Variablen beobachten, die an verschiedenen Stellen im Anwenderprogramm benutzt werden. Auch die Inhalte komplexer Datentypen wie Strukturen oder Felder werden dargestellt.

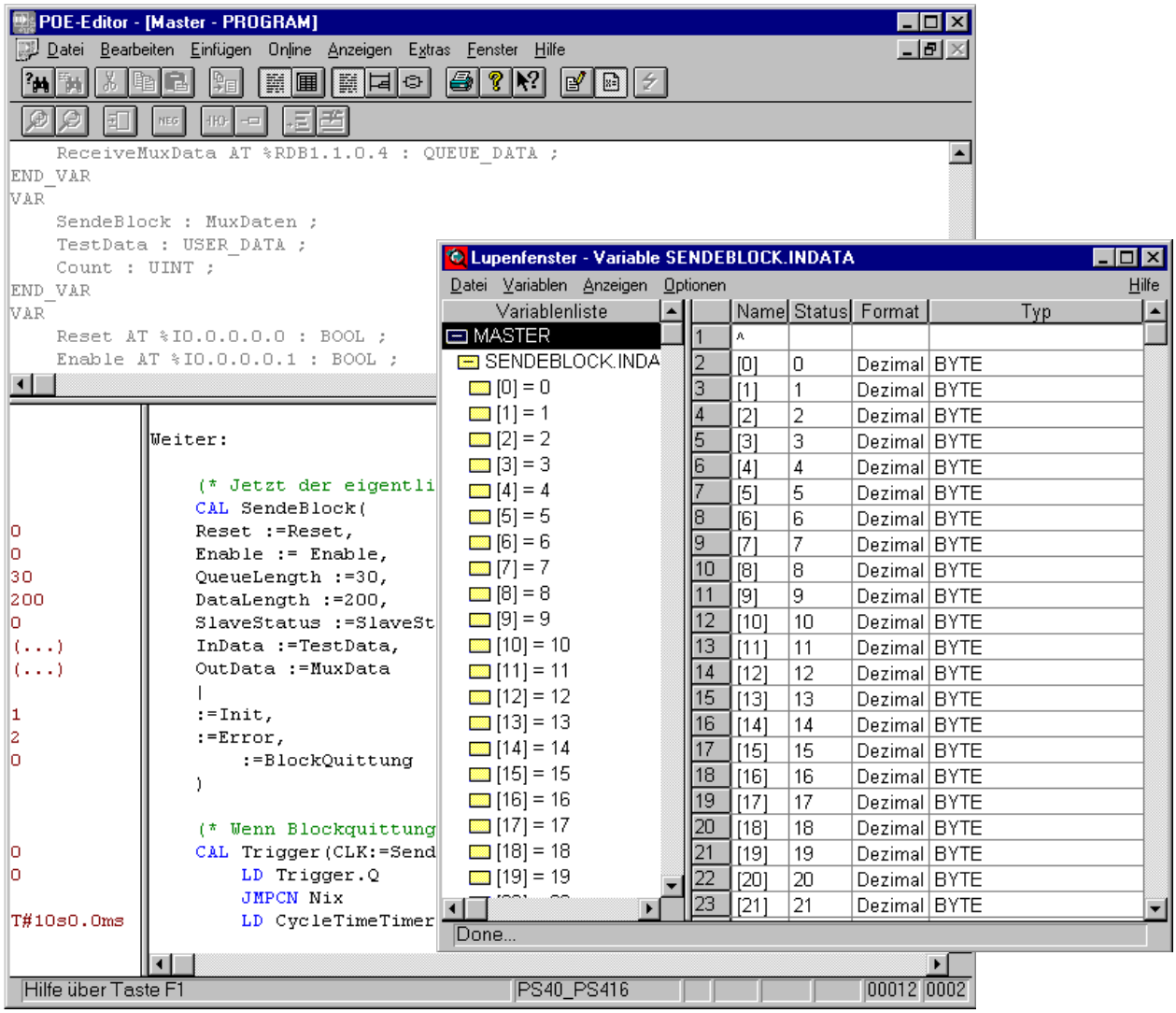


Abbildung 10: Darstellung komplexer Variablen in der Zustandsanzeige

Zusammenfassung:
Mit der Programmiersprache Anweisungsliste steht dem Anwender eine bekannte, leistungsfähige Programmiersprache für die Bearbeitung von Steuerungsaufgaben zur Verfügung. Voraussetzung hierfür ist natürlich, daß die in der IEC 1131-3 zur Verfügung gestellten Spracheigenschaften auch von einem Programmiersystem unterstützt werden. Moeller stellt mit der Sucosoft S40 dem Anwender alle IEC-Datentypen bis 32 Bit zur Verfügung, daneben den Datentyp STRING für die Textverarbeitung sowie alle Datums- und Zeitdatentypen. Strukturen und Felder sind auch verschachtelt möglich. Weitere Informationen erhalten Sie bei unseren Vertriebsniederlassungen oder im Internet unter **http://www.moeller.net.**