3. Auswahl eines geeigneten Programmiersystems

3.1 Aktuelle Situation: Automatisierungssysteme und Programmiersprachen

Quelle: Einzelne Passagen wurden unter Nutzung von Wikipedia: -> http://de.wikipedia.org erarbeitet.

Speicherprogrammierbare Steuerungssysteme sind nach wie vor die wichtigsten Elemente der Automatisierungstechnik. Der Begriff SPS suggeriert viel weniger Leistungsfähigkeit als seit Jahren praktisch angeboten wird. SPS-Systeme beinhalten heute z.B. auch Panels zum Bedienen und Beobachten (HMI), automatische Kommunikation mit übergeordneten „Prozessrechnern“, Datenkommunikation in Netzen, schnelle geregelte Antriebe und komplexe Regelfunktionen sowie Applikationen von IndustriePC.

Die Programmierung solcher Systeme ist kostenintensiver als die Hardware selbst. Ursache dafür war bisher vor allem, dass genormte Programmiersprachen nur auf dem Papier existierten. Bis zu 70 Steuerungshersteller waren zeitweise am Markt aktiv und benutzten genauso viele **„Dialekte“** der Programmiersprachen. Dazu kam der „Feldbuskrieg“ der achtziger und neunziger Jahre. Die Vielzahl der Sprachelemente – quasi ein „babylonisches Sprachgewirr“ - verhinderten Automatisierungslösungen mit Komponenten unterschiedlicher Hersteller, die hinsichtlich der Automatisierungs- und Kommunikationsstruktur hätten optimiert werden können.

Aktuelle Aufgabe ist die **Überwindung der Zersplitterung der Programmiersysteme für Automatisierungstechnik**. Der Aufwand an Ingenieurstunden für die Projektierung und Programmierung von Systemen ist gemessen an den Kosten der Hardware eindeutig zu hoch! Dazu kommen immer wieder zu hohe Einarbeitungsaufwände, wenn unterschiedliche Systeme eingesetzt werden sollen. Weiter wünschen viele Anwender Unabhängigkeit von einzelnen Herstellern.

Das Automatisierungssystem Siemens **Simatic S7** mit dem Programmiersystem Step7 darf sich hinsichtlich Verbreitung und Bekanntheitsgrad durchaus als Weltmarktführer bezeichnen. Oftmals wird vom **„Industriestandard S7“** gesprochen. Viele andere SPS-Systeme wurden am Markt mehr oder weniger verdrängt.

Die Vorzüge des Systems S7 sind neben dem Bekanntheitsgrad u.a.
- das Prinzip „Alles aus einer Hand“ über vielfältige Teilgebiete der Automatisierungstechnik
- damit zusammenhängend die durchgehende Datenhaltung
- die großen „Mengengerüste“ bei Eingängen, Ausgängen und Speichern
- die integrierten Diagnose- und Fernwartungs-Systeme

Es gibt aber auch **nachteilig wirkende Faktoren**: Das System Simatic S7 ist aus dem ebenfalls weit verbreiteten Vorläufer Simatic S5 hervorgegangen. S5 wurde auf dem Betriebssystem DOS aufgesetzt, wodurch aus heutiger Sicht eine Vielzahl von Beschränkungen bestanden. Obwohl die Umstellung bereits ab 1995 erfolgte, arbeiten auch heute noch Steuerungen auf dem Stand S5!

Bei der Umstellung sollten möglichst viele von S5 her gewohnten Methoden und Verfahrensweisen übernommen werden wie z.B. die S5-Timer und S5-Zähler, die Verwendung von Merkern und Datenbausteinen u.a. Diese Details und einige damit zusammenhängende Datenformate sind aber nicht IEC-konform!
Jedoch: Wenn der Wille besteht, kann auch mit Step7 deutlich „IEC-konformer“ programmiert werden (siehe Abschnitt 3.4). Dies ist aber in der Praxis derzeit wenig üblich.

Eine annähernde Lösung der Probleme bei der Programmierung von Automatisierungstechnik kann **nur durch eine internationale Norm** für Programmiersysteme erfolgen. Die Normung konnte lange nicht mit der aktuellen Entwicklung Schritt halten, aber seit 1993 liegt mit IEC 61131 eine solche Norm vor. Allerdings ist auch die **Normung nicht ohne Probleme**! Die schnellebige Entwicklung verlangt immer wieder neue Lösungen, und es gibt vielfältige Unternehmensinteressen!! Auch „genormte“ Programmiertools sind oftmals (leider) nur theoretisch portabel.

Am Markt haben sich in den letzten Jahren eine Vielzahl kleinerer Unternehmen etabliert, die hochwertige Automatisierungskomponenten herstellen wie z.B. die Unternehmen Beckhoff oder WAGO.

Ihre Controller werden selbstverständlich nicht mit Step7 programmiert, sondern mit Programmiersprachen, die sich strikt an der Norm orientieren. Am europäischen und insbesondere deutschen Markt hat sich de facto folgende **Polarisation** entwickelt: Auf der einen Seite stehen Applikationen des Systems Siemens Simatic / Step7, auf der anderen Seite eine Vielzahl unterschiedlicher Automatisierungskomponenten, die nach IEC 61131-3 programmiert wurden.

3.2 Die Norm IEC 61131 und die Organisation PLCopen

Die Norm IEC 61131 wurde im März 1993 von der International Electronic Commission (IEC) veröffentlicht. An dieser Norm mit Teilen 1- 5 arbeitete eine Untergruppe der IEC auf Antrag des amerikanischen Unternehmens Allen Bradley seit 1983. Der Entwurf erschien 1987 als IEC 65A.

Die Norm versucht, möglichst viele Aspekte der Hersteller und viele Einsatzfälle der SPS-Technik zu berücksichtigen. Andererseits müssen sich die Hersteller der Norm unterwerfen und eventuelle Abweichungen offenlegen und vollständig dokumentieren.

Wegen der enormen Menge an Festlegungen ist die Norm IEC 61131-3 eine Richtlinie zur Programmierung und weniger eine starr bindende Vorschrift. Es ist davon auszugehen, dass die verschiedenen Programmiersysteme nur einen mehr oder weniger großen Teil der Norm realisieren.

Die Norm umfasst 5 Teile:

IEC 61131-3 Teil 1: Allgemeines ... legt die in der Norm verwendeten Begriffe fest.

IEC 61131-3 Teil 2: Anforderungen an Geräte ... legt die SPS-Struktur und physikalische Anforderungen fest wie Temperaturbereiche, Störeinflüsse, Umgebungsbedingungen, Impedanzwerte für Analogeingänge u.a.

IEC 61131-3 Teil 3: Programmiersprachen ... legt die fünf Programmiersprachen
- Anweisungsliste (AWL)
- Strukturierter Text (ST)
- Kontaktplan (KOP)
- Funktionsbausteinsprache (FUP)

und zusätzlich die Freigrafische Funktionsbausteinsprache fest. Weiter werden die Variablendeklaration, Datenformate, Schlüsselworte und die Programmorganisation beschrieben.

Dieser Teil der Norm ist „ein Internationaler Standard für moderne Systeme mit SPS-Funktionalität. Aufbauend auf einem strukturierten Softwaremodell definiert er eine Reihe leistungsfähiger Programmiersprachen, die für unterschiedliche Automatisierungsaufgaben eingesetzt werden können“.
Quelle: Glossar des Handbuches Ethernet TCP / IP 750-841Vers. 1.0.1 WAGO GmbH

IEC 61131-3 Teil 4: Benutzer-Richtlinien ... beschreibt Anforderungen an die Dokumentation von Hard- und Software.

IEC 61131-3 Teil 5: Kommunikation ... beschreibt einheitliche Bausteine für den Aufbau der Datenkommunikation zu beliebigen internen und externen Kommunikationsobjekten.

Zur Durchsetzung der Norm wurde 1992 die **Organisation PLCopen** gegründet. Sie ist eine internationale, hersteller- und produktunabhängige Vereinigung von SPS-Herstellern, Softwarehäusern und Instituten. **Ziel ist die Förderung von Entwicklung und Einsatz kompatibler Software** für PLC. PLC heißt Programmable Logic Control und ist die weltweite Bezeichnung des im Deutschen gebräuchlichen Begriffes SPS.

Ziele der PLCopen sind im einzelnen
- Bestrebungen, um IEC 61131-3 international durchsetzen
- gemeinsame Marketingstrategie
- Hilfestellung für das IEC-Normungsgremium
- Festlegung von Konformitätsklassen.
- Produktprüfung zur Einhaltung von IEC 61131 und Zertifizierung
- Schaffung von Datenaustauschformaten für Anwenderprogramme

- Die Zertifizierung von Programmiersystemen erfolgt von unabhängigen Institutionen nach einer Anforderungsliste. Für jede Programmiersprache der Norm IEC 61131-3 erfolgt eine Abstufung in drei Klassen
- **Base Level**. Mit dem Programmiersystem entwickelte Programme müssen in ihrem Grundaufbau IEC 61131-3 verträglich sein. Die wesentlichen Sprachelemente einer Programmiersprache müssen vorhanden sein.
- **Portability Level**. Die Auswahl der Anforderungen wird soweit ausgedehnt, dass es möglich ist, reale Software-Bausteine zwischen zertifizierten Programmiersystemen auszutauschen.
- **Full Level**. Weiterer Ausbau des Portability Level durch Einbeziehung von Konfigurations-Informationen

3.3 Das Programmiersystem CoDeSys und die CoDeSys Automation Alliance

CoDeSys ist eine Entwicklungsumgebung für die Software von Automatisierungstechnik nach IEC 61131-3. Die Produktbezeichnung CoDeSys steht für **Co**ntroller **De**velopment **Sys**tem. Hersteller des Softwaretoolds CoDeSys ist der 1994 gegründete mittelständische Softwarehersteller **3S-Smart Software Solutions** aus Kempten (-> www.3s-software.com).

Über 150 namhafte Firmen der unterschiedlichsten Branchen unterstützen das Software Tool CoDeSys in Ihren Steuerungen. Damit ergeben sich mehrere tausend Anwender, die CoDeSys bei Ihrer täglichen Arbeit nutzen. Im Vergleich sind das mehr als bei jedem vergleichbaren IEC 61131-3 Programmiersystem in Europa. Damit kann CoDeSys als Marktstandard bezeichnet werden.

Einige bekannte Firmen und ihre Produkte auf der Basis CoDeSys sind u.a.

- ABB: System AC1131
- Beckhoff Automation GmbH: SystemTwinCAT
- Bosch Rexroth AG: System IndraLogic
- Micro Innovation AG: System MXpro
- Moeller GmbH: System XSoft
- Lenord+Bauer: System Motionline
- WAGO Kontakttechnik GmbH : I/O System 750

Einheitliche Programmiertools für eine Gruppe von Herstellern sind eindeutig von Vorteil! Mehrere Hersteller können gemeinsam in die (teure) Software investieren. Um diese Vorteile am Markt zu nutzen, schließen sich Hersteller von Automatisierungstechnik zu **Alliancen** zusammen. Beispiele sind

- Profibus-Nutzer-Organisation (-> www.profibus.org)
- EtherCat Technology Group (-> www.ethercat.org)
- CoDeSys Automation Alliance (-> www.automation-alliance.com)

Die **CoDeSys Automation Alliance (CAA)** ist die Vereinigung von Herstellern, die sich dem System CoDeSys mit dem Ziel zugewandt haben, die Schwierigkeiten bei der Verwendung von Steuerungen unterschiedlicher Hersteller zu vermeiden.
Die Alianz wurde im Jahr 2000 auf Initiative von 3S-Smart Software Solutions gegründet. Bereits im Juli 2005 waren über 60 Automatisierungsanbieter Mitglied der CAA.

Die Verwendung einer einheitlichen Benutzeroberfläche für unterschiedliche Hardware Plattformen wird durch die Integration eines so genannten **Target Support Package** erreicht. Der Steuerungshersteller beschreibt darin den Prozessortyp, die Speicherausstattung und die Verfügbarkeit bestimmter durch die Steuerung bereitgestellter Möglichkeiten und Bibliotheken. Der Hersteller von CoDeSys sorgt für die Integration der entsprechenden Codegeneratoren.

Die Auswahl einer bestimmten Steuerung erfolgt innerhalb der Programmierumgebung und ist so einfach wie z.B. die Selektion eines bestimmten Druckertreibers innerhalb eines Windows Programms. So können Applikationen, die für die Steuerung des Herstellers A erstellt wurden, sehr einfach auf die Hardware des Herstellers B übertragen werden. Für Anlagen, in denen mehrere Steuerungen unterschiedlicher Hersteller verbaut sind, besteht die Möglichkeit des Austauschs von Daten über sogenannte Netzwerkvariablen.
Interessante Neuerungen werden alljährlich auf den Anwenderkonferenzen der CAA vorgestellt (-> www.users-cenference.de).

Ein mit CoDeSys vergleichbares Produkt ist die **KW-Software** der KW-Software GmbH Lemgo (-> www.kw-software.com). Die Hauptprodukte sind

- **Multiprog ®** : IEC 61131 - Programmiersystem
- **ProConOS ® (**Programmable Controller Operating System) : echtzeit- und multitaskfähiges IEC-konformes Laufzeitsystem für Automatisierungsssysteme

Ein Kristallisationspunkt für die Entwicklung dieses IEC-Tools war die Zusammenarbeit mit dem Unternehmen Phoenix Contact GmbH & Co.KG (-> www.phoenixcontact.de): 2001 wurde die KW-Software GmbH ein unabhängiges Tochterunternehmen dieses Unternehmens. Inzwischen rankt sich um KW-Software eine mit der CoDeSys Alliance vergleichbare Firmengruppierung und es gibt Vertretungen in China, Japan und de USA.

3.4 Welches Programmiersystem für Studierende?

Bildungseinrichtungen und Studierende der Automatisierungstechnik stehen in Deutschland (insbesondere bei der Gestaltung von Trainingsmodulen und Praktika) vor der Frage, ob sie sich dem System Simatic S7 oder einem „IEC-Programmiersystem“ zuwenden sollen.

Vorteile einer Entscheidung für Simatic S7 sind hierbei:

- weite Verbreitung und hoher Bekanntheitsgrad
- „de facto – Standard“ in der Industrie: Kenntnisse sind bei Industriepraktika und späterem Einsatz fast immer unverzichtbar!
- häufig Vorkenntnisse aus vorbereitenden Bildungseinrichtungen wie Berufs- und Meisterschulen

Nachteile einer Entscheidung für Simatic S7 sind hierbei:

- relativ hohe Kosten für Hardware-Komponenten, insbesondere dann, wenn die Leistungsmerkmale hochwertiger Hardware nicht genutzt werden
- Hohe Kosten für Lizenzierung der Software Step7
- sehr viele – zunächst(!) eher unwichtige – Details vor dem ersten Erfolg
- Simulation der Programme auf dem PC wenig komfortabel
- Visualisierung nur über zusätzliche Tools, da keine integrierte Visualisierung
- Zwar ist die Ablaufsprache in der Software Step7Professional integriert, doch ist für die Programmiersprache Strukturierter Text ein weiteres kostenpflichtiges Softwaretool erforderlich.
- (bisher) spezielle Kommunikationsprozessoren wie CP5611 für die Anschaltung eines PC an die CPU erforderlich, da der Zugang nicht über Standard-Netzkarte möglich ist.

Vorteile einer Entscheidung für ein „IEC-Programmiersystem“ sind hierbei:

- Auswahl an unterschiedlicher und preisgünstiger Hardware, die für das Labor auch extrem feinmodular gestaltet werden kann
- Die Software ist kostenfrei! Kosten entstehen lediglich durch Target-Files (Treiber), die dann erforderlich werden, wenn die Programme auf einer Hardware-Plattform laufen sollen. Durch die komfortable integrierte Simulation und Visualisierung braucht der Studierende aber zunächst nicht unbedingt eine Hardwareplattform!
- **Wer sich der Norm IEC 61131-3 zuwendet, arbeitet mit Sprachelementen, die eine standardisierte und auch den Ansprüchen der aktuellen Informatik genügende Programmierung erlaubt. Insbesondere ist die Sprache Strukturierter Text kostenfreier Bestandteil des Programmiersystems.**

Nach Abwägung aller Vor- und Nachteile entscheidet sich die Fachhochschule Schmalkalden für das Programmiersystem CoDeSys und im Praktikum für die Hardware-Plattform WAGO-I/O-System 750. Zwar ist danach ein Einstieg in Step7 gewöhnungsbedürftig, aber gut machbar. Umgekehrt erfordert der Umstieg von der (klassischen) Anwendung von Step7 auf ein „IEC-Programmiersystem“ deutlich mehr Kraft!

Bei der kostenlosen CoDeSysy-Software ist derzeit die Version 2.3.7.3 aktuell. Ein Umstieg auf eine neue Software 3.0 ist angekündigt und derzeit bereits möglich.

Hinweis: Die Vorteile eines Programmiersystems wie CoDeSys hinsichtlich Kosten, Simulation und integrierter Visualisierung kann ein Studierender aber nur dann vorteilhaft nutzen, wenn er zu seinem Ingenieurstudium mit eigenem mobilen PC antritt. In technischen Studienrichtungen sollte dies selbstverständlich geworden sein!

3.5 Step7 konsequent nach den Regeln von IEC 61131-3 einsetzen! Ist das möglich?

Zur Beantwortung der Frage müssen zunächst die **wesentlichen Unterschiede** zwischen Step7 und Systemen nach IEC 61131-3 genannt werden.

1. Unterschiede zeigen sich zuerst in der **Datenhaltung**:
2.

- In der Norm IEC 61131-3 existieren keine globalen Datenbausteine und keine Merkerbereiche (obwohl einige Systeme das Wort Merker als spezielle RETAIN-Variablen zulassen). Es ist in der Norm nicht üblich, solche Speicherbereiche vom Anwender adressieren zu lassen.

- IEC 61131-3 kennt auch keine Instanzdatenbausteine, über die bei Step7 Parameter sowie statische Variablen abgespeichert werden und bei Bedarf auch mit ihren Adressen bearbeitet werden können. Der spezielle Begriff „statische Variable“ wird in der Norm nicht geführt.

2. Weitere Unterschiede zeigen sich in **Eigenschaften von POE**:

- Funktionen können in Step7 mehrere Ausgänge und innere Speicher besitzen. Abgesehen von der Verfügbarkeit statischer Variablen bei FB gibt es bei Step7 beim Programmieren keine entscheidenden Unterschiede zwischen Funktionen und Funktionsbausteinen.

- Die Abarbeitung des Programms wird über Organisationsbausteine mit unterschiedlichen Prioritäten gesteuert. Solche OB’s sind in der Norm IEC 61131-3 nicht vorgesehen.

3. Weiter gibt es bei Step7 eine Reihe spezieller **Eigenheiten**, wie sie auch bereits von Step5 her bekannt sind:

- Es existieren in Step7 weiterhin die Operanden E, A, M, P, T und Z mit absoluter Adresse. Der Zugriff auf absolute Adressen wird nicht mit % gekennzeichnet.

- In Step7 können Zeitfunktionen und Zähler noch Step5-konform und mit speziellen Zeit- bzw. Zählerformaten (S5TIME:S5T#..... bzw. C#....) realisiert werden, aber auch IEC-gerecht durch Anwendungen von Systemfunktionsbausteinen (SFB’s) aus der Standardbibliothek!

- Eine Symboltabelle für die Zuordnung von Symbolen zu absoluten Hardwareadressen kennt IEC 61131-3 so nicht!

- In AWL existiert weiterhin die CPU-spezifischen Anzahl von Akkumulatoren und das Verknüpfungsergebnis (VKE).

- Step7 unterscheidet wie Step5 beim Lesen (Laden) und Schreiben (Transferieren) zwischen Bit- und digitalen Daten. Für das Laden eines Bitwertes in AWL werden wie in Step5 die Befehle U bzw. O anstatt LD verwendet.

Dennoch lautet die Antwort auf obige Frage mit Einschränkungen ja!
Obwohl man die genannten speziellen Eigenheiten von Step7 nicht immer und vollständig umgehen kann, insbesondere nicht die direkt adressierten Operanden E, A, M, PE und PA sowie selten die globale Datenelemente, sollte man sich bemühen, nahe an der Norm zu programmieren.

Dazu muss man bestimmte Gewohnheiten, die zuerst von Step5 her bestehen, überwinden und folgende Bedingungen einhalten:

- Das Programm ist strikt als **strukturiertes** Programm aufzubauen. Das bedeutet, dass das Programm in universelle und mehrfach nutzbare Einheiten zu unterteilen ist.

- Alle Bausteine sind als bibliotheksfähige, der Norm entsprechende Programm-Organisationseinheiten (POE) **hardwareunabhängig** zu schreiben. Deshalb werden Programmbausteine grundsätzlich nur mit lokalen Variablen geschrieben, welche in der Deklarationstabelle deklariert werden.

- Hardwareadressen bzw. deren symbolischen Namen werden (am besten) erst im Organisationsbaustein OB 1 verwendet und über die Schnittstellen in Form der Parameter IN, OUT und IN_OUT den jeweiligen Bausteinen übergeben. Symbolische Operanden sollten Vorrang vor absoluten Operanden genießen.

- S5-Timer und S5-Zähler sind durch **IEC-Timer und IEC-Zähler** zu ersetzen. Diese finden sich als Bausteine vom Typ Systemdatenbaustein (SFB) in der dem System S7 zugeordneten Standardbibliothek.
  Die IEC-Timer und Zähler können wie andere FB eigenständig im Programm benutzt werden, sie können aber – sofern sie in einem FB aufgerufen werden – in dessen Instanzdatenbaustein integriert werden. Damit entstehen Multiinstanzen. Diese entstehen immer dann, wenn FB's als statische Variablen eines übergeordneten FB verwaltet werden. Vom Kern her ist dies zuerst eine Frage der Speicherverwaltung.

  > Achtung! Beim Einsatz von Instanzdatenbausteinen ist zu bedenken:
  > Beim mehrfachen Aufruf eines FB mit gleichen Instanzdatenbausteinen werden gemeinsame Speicherräume genutzt! Dies ist nur in bestimmten Fällen sinnvoll.
  > Dagegen stehen beim mehrfachen Aufruf mit unterschiedlichen Instanzdatenbausteinen ´ jeweils getrennte Speicher zur Verfügung!

- Grundsätzlich ist bei den POE's **auf Merker und Datenbausteine zu verzichten** und stattdessen mit lokalen Variablen zu arbeiten.
  Für Variable, deren Werte über den Zyklus hinaus nicht erhalten werden müssen, sind **temporäre** Variablen zu benutzen.
  Für Variable, deren Werte über den Zyklus hinaus zu erhalten sind, müssen **statische** Variablen benutzt werden.

  > Diese Forderung stellt solange kein Problem dar, wie ausschließlich mit lokalen Variablen gearbeitet werden kann, denn sowohl temporäre als auch statische Variablen sind lokale Variablen! Bausteinübergreifende Variablen stehen dann zunächst nicht zur Verfügung!
  >
  > Wenn das Programm so aufgebaut wird, dass bausteinübergreifende (globale) Variablen erforderlich werden, dann werden diese bei Step 7 traditionell überwiegend mit Merkern und Datenbausteinen ausgeführt.
  > Man kann dennoch auf Merker und Datenbausteine verzichten, wenn man bausteinübergreifende Variablen als Parameter vom Typ IN, OUT oder IN_OUT deklariert und über die Schnittstellen aller verschachtelten Bausteine zum Beispiel bis zum OB1 „durchschiebt".
  > Dort kann dann ein „Verschalten" mit Hilfe temporärer Variablen erfolgen. Das Verschalten mit temporären Variablen kann selbstverständlich auch in anderen lauffähigen Bausteinen erfolgen.
  > Achtung! Dieses Prinzip könnte an Grenzen der verfügbaren lokalen Speicher stoßen (derzeit 256 Byte für OB1 / Priorität 1)!
  >
  > Eine andere Methode ist die konsequente Anwendung von **Multiinstanzen**:
  > Das Programm wird in FB's strukturiert und mit statischen Variablen geschrieben. Diese FB's selbst werden wiederum als statische Variablen vom Typ FB im Instanz-Datenbaustein eines übergeordneten FB verwaltet. Dann stehen sämtliche (statischen) Variablen aller FB's in einem Instanz-Datenbaustein und damit in einem einzigen Speicher. Bausteinübergreifende Variablen werden dann nicht mehr benötigt. Achtung! Auch hier müssen die Grenzen der verfügbaren lokalen Datenspeicher beachtet werden!