Vorlesungsinhalte

1. Integrierte Betriebsführung VL1
2. Vertikale und horizontale Integration
3. **Aufbau und Strukturen industrieller Steuerungssysteme** **VL2**
4. SPS-Programmierung nach IEC 61134
5. Übungen VL3

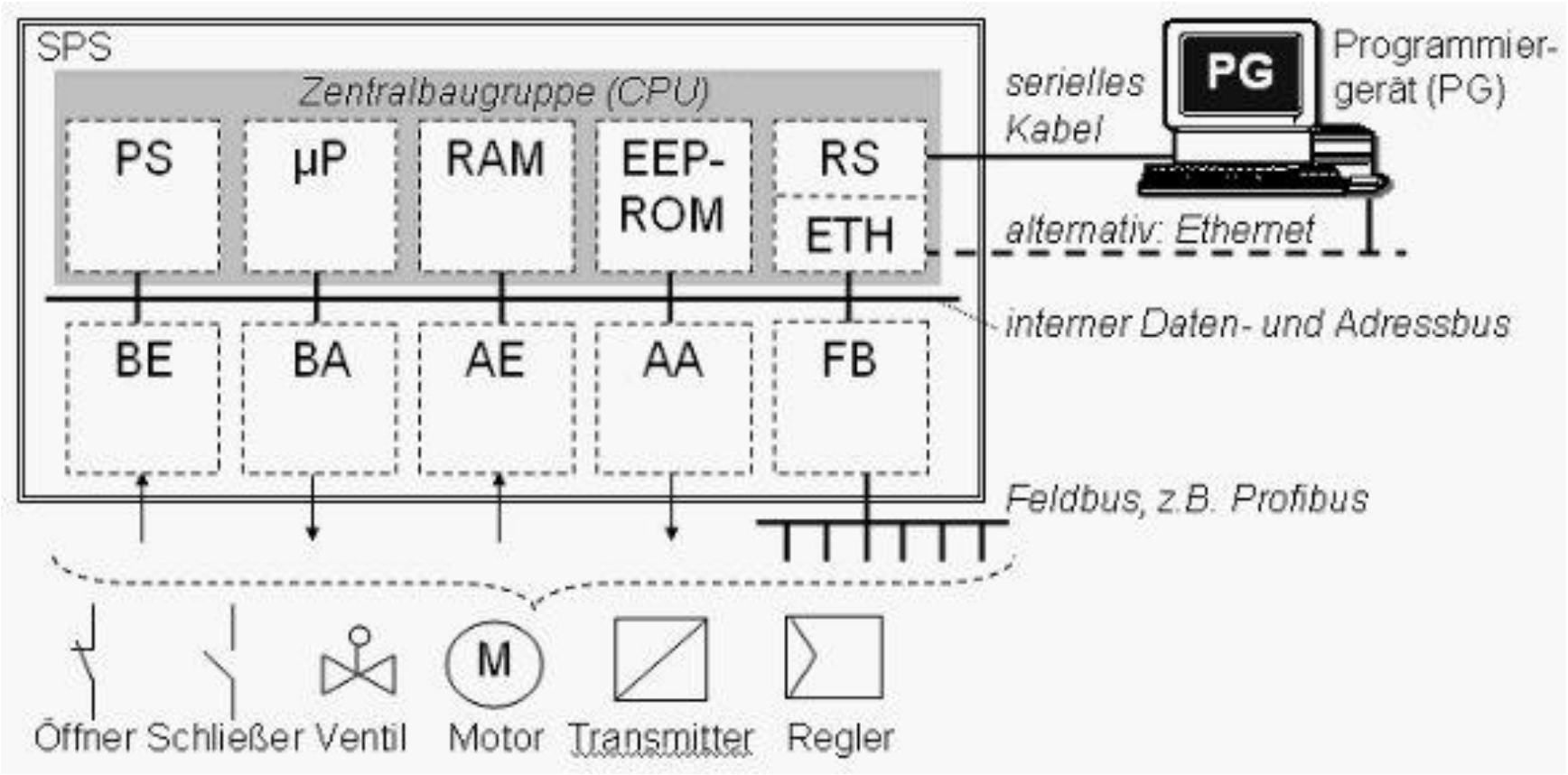
3 Aufbau und Strukturen industrieller Steuerungssysteme (01 von 99)

Den heutigen Einsatz industrieller Steuerungssysteme prägen drei wesentliche Entwicklungen:

- **Industrie-PCs** dienen als Standardplattform nicht nur für Programmiersysteme, sondern immer häufiger auch in Form von *Soft-SPS*en als Steuerungen,
- der **Windows-Standard** bietet *offene Schnittstellen* zu anderen PC-basierten Systemen und ermöglicht die Nutzung von Internet und Mobilfunk mit typischen Anwendungen, wie Fernwartung, Fernüberwachung, E-Business u.a.,
- Teile der Steuerungssoftware laufen in **intelligenten Sensoren und Aktoren** ab, was zu einer Dezentralisierung der Automatisierungseinheiten führt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (02 von 99)

3.1 Aufbau einer SPS



3 Aufbau und Strukturen industrieller Steuerungssysteme (03 von 99)

3.1 Aufbau einer SPS

Eine klassische SPS besteht aus den in Abb. 03.01 dargestellten Hardwaremodulen. Die Stromversorgungsbaugruppe PS (**P**ower **S**upply) wandelt die Netzspannung in eine 24V-Gleichspannung (24 VDC) um, mit der die gesamte Elektronik der SPS versorgt wird. Werden innerhalb der SPS-Baugruppen noch andere Spannungen benötigt (z.B. +5VDC, +/-12VDC), werden diese aus der 24V-Gleichspannung erzeugt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (04 von 99)

3.1 Aufbau einer SPS

Die Zentralbaugruppe oder CPU (**C**entral **P**rocessing **U**nit) mit einem Mikroprozessor (μP) zum Ausführen des Programms bildet das Kernstück der SPS. Die aktuell abgearbeiteten Programme stehen online im RAM (**R**andom **A**ccess **M**emory). Außer den laufenden Anwenderprogrammen und den Systemprogrammen des Betriebssystems werden im RAM auch die von den Programmen benötigten Variablen und Parameter (Zeiten, Zählerstände, Soll-/Istwerte) gespeichert. Achtung! Der Speicherinhalt des RAMs geht bei Spannungsausfall verloren. Ein RAM ist ein flüchtiger Speicher.

3 Aufbau und Strukturen industrieller Steuerungssysteme (05 von 99)

3.1 Aufbau einer SPS

Die SPS besitzt ein EEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory), in dem alle Anwender- und Systemprogramme offline gespeichert werden können. Der EEPROM ist häufig als gesteckte Memory-Card realisiert. In modernen SPSen werden inzwischen Flash-Speicherkarten (CF, SD) verwendet.

3 Aufbau und Strukturen industrieller Steuerungssysteme (06 von 99)

3.1 Aufbau einer SPS

Eine weitere Besonderheit einer SPS sind die *Ein-/Ausgabebaugruppen* zum Einlesen von Sensorinformationen und zum Ausgeben von Befehlen an die Aktoren. Hierbei werden die binären bzw. analogen Datenpunkte konventionell über je eine Anschlussleitung an die Ein-/Ausgabebaugruppen angeschlossen.

Busfähige Sensoren und Aktoren werden über Feldbusschnittstellen-Baugruppen an die SPS angeschlossen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (07 von 99)

3.1 Aufbau einer SPS

Neben *internen Daten- und Adressbussen* zum Datenaustausch zwischen den Hardwaremodulen kann eine SPS über *serielle* oder *Ethernet-Schnittstellen* an ein Programmiergerät (PG) angekoppelt werden.

Heute wird Ethernet oder besser Industriell Ethernet häufig zur Kopplung von SPSen untereinander und an ein Visualisierungssystem zum Bedienen und Beobachten verwendet.

3 Aufbau und Strukturen industrieller Steuerungssysteme (08 von 99)

3.1.1 Programmiergerät (PG)

Das Programmiergerät war früher ein spezieller, sehr robuster und tragbarer PC, der oft mit spezieller Betriebssoftware ausgestattet, allein dem Zweck der Programmierung und Inbetriebnahme/Fehlersuche an einer SPS diente. Heute wird ein Standard-PC oder ein Laptop/Notebook verwendet. Auf diesen Systemen muss dann die für die jeweilige SPS erforderliche Software installiert und eingerichtet werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (09 von 99)

3.1.1 Programmiergerät (PG)

Mitunter werden bei einigen SPSen noch serielle Schnittstellen (COM) für die Verbindung zwischen PC/Notebook und SPS benötigt. Die meisten neueren Systeme können aber auch über USB bzw. Ethernet programmiert werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (10 von 99)

3.1.1 Programmiergerät (PG)

Das *Laden der Programme* in die SPS verursacht, dass alle laufenden Programme abgebrochen und die Variablen neu initialisiert werden. Diese Arbeiten sollten also nur von befugtem Personal ausgeführt werden. Manche Steuerungen werden durch spezielle Hardware-Schlüssel oder Passwort vor einem versehentlichen Überschreiben des Programms geschützt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (11 von 99)

3.1.1 Programmiergerät (PG)

Bei Wartungsarbeiten an der Anlage, die durch eine SPS gesteuert wird kann durch die Betriebsart STOP der SPS verhindert werden, dass Signale über die Ausgänge der SPS an die Anlage (z.B. Motor –M01 „EIN“) ausgegeben werden. Für die Einhaltung der Maschinenrichtlinie reicht diese Maßnahme nicht aus. Entsprechende Anlagenteile, die Gefährdungen verursachen können, müssen über Reparaturschalter verfügen. Diese unterbrechen die Ansteuerung durch die SPS im Wartungsfall.

3 Aufbau und Strukturen industrieller Steuerungssysteme (12 von 99)

3.1.1 Programmiergerät (PG)

Programme sollten vor Inbetriebnahme an einer realen Anlage in wesentlichen Teilen simuliert werden. Viele Programmiersysteme unterstützen diese Funktion entweder als integraler Bestandteil (z.B. Mitsubishi IEC-Developer) oder als zusätzlich zu erwerbenden Software-Modul (z.B. Siemens SIMATIC Step 7).

3 Aufbau und Strukturen industrieller Steuerungssysteme (13 von 99)

3.1.2 Anzeige- und Bedienkomponente (ABK)

Auf entsprechenden PCs (Leitrechner) laufen bestimmte Programme (sog. OPC- oder DDE-Server), die die SPS-Daten für andere Anwendungen (z.B. Visualisierungsprogramme) zur Verfügung stellen. Insbesondere für *Anzeige- und Bedienkomponenten (ABKen)* sind die Daten aus der SPS wichtig, um Anlagenzustände anzuzeigen und Bedieneingriffe wirksam zu machen.

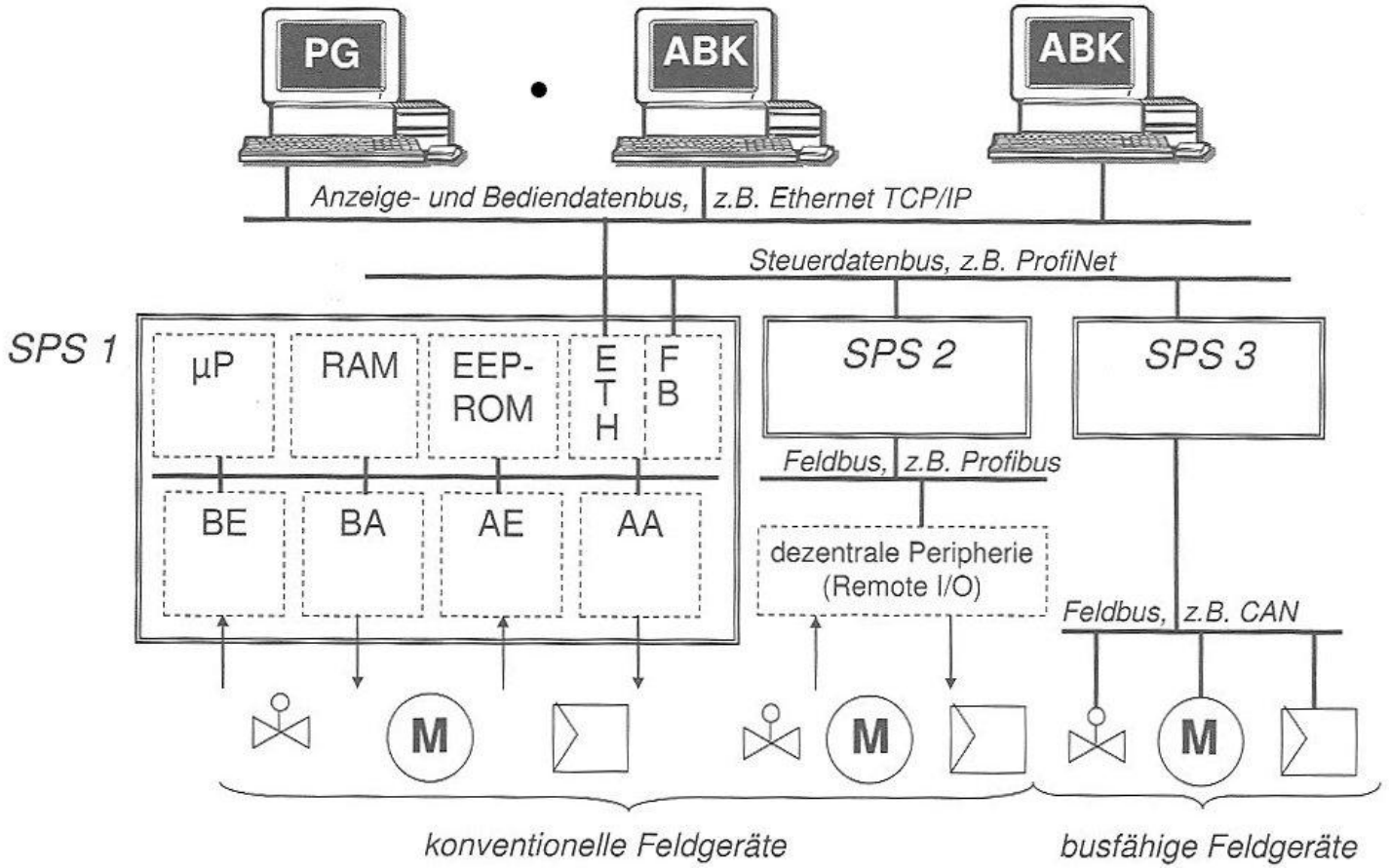
3 Aufbau und Strukturen industrieller Steuerungssysteme (14 von 99)

3.1.2 Anzeige- und Bedienkomponente (ABK)

Der Aufbau einer SPS ist demnach von den Feldgeräten ebenso wenig zu trennen wie von Programmiergerät und Anzeige- und Bedienkomponenten. Im Zuge immer komplexerer Anwendungen entstehen dabei Systemstrukturen wie in Abb. 03.02 dargestellt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (15 von 99)

3.1.2 Anzeige- und Bedienkomponente (ABK)



3 Aufbau und Strukturen industrieller Steuerungssysteme (16 von 99)

3.1.2 Anzeige- und Bedienkomponente (ABK)

Diese bestehen beispielsweise aus mehreren SPSen, die über einen Steuerdatenbus kommunizieren können. Das Programmiergerät kann anstatt über die serielle Schnittstelle auch über eine Busschnittstelle mit an den Anzeige- und Bediendatenbus angekoppelt werden und dadurch ebenso wie die Anzeige- und Bedienkomponenten mit den SPSen Daten austauschen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (17 von 99)

3.1.2 Anzeige- und Bedienkomponente (ABK)

Die Feldgeräte könne auf unterschiedliche Weise an die SPS angekoppelt werden:

- konventionell durch *Kupferdrahtleitungen* (siehe Abb. 03.02 links)
- über *Feldbus*, was natürlich busfähige Feldgeräte erfordert (siehe Abb. 03.02 rechts)
- über eine dezentrale Peripherie (*Remote-I/O*), die einerseits über Feldbus mit der SPS verbunden ist und andererseits nicht busfähige Feldgeräte über dezentrale E/A-Baugruppen anbindet (siehe Abb. 03.02 Mitte)

3 Aufbau und Strukturen industrieller Steuerungssysteme (18 von 99)

3.2 SPS - Arten

Man unterscheidet heute drei verschiedene Bauarten von SPSen:

- *Hardware* – SPS,
- *Slot* – SPS und
- *Soft* – SPS .

3 Aufbau und Strukturen industrieller Steuerungssysteme (19 von 99)

3.2.1 Hardware - SPS

Der im vorigen Abschnitt beschriebene Aufbau einer SPS bezieht sich auf den klassischen Aufbau einer *Hardware*-SPS. Eine Hardware-SPS ist kompakt (im wesentlichen aus einer Einheit bestehend) oder modular (aus unterschiedlichen Baugruppen bestehend, die entsprechend den Anforderungen zusammengestellt werden) aufgebaut.

Eine Hardware-SPS bedarf eines externen PCs als Programmiergerät.

3 Aufbau und Strukturen industrieller Steuerungssysteme (20 von 99)

3.2.2 Slot - SPS

Eine *Slot*-SPS ist eine Einsteckkarte für den PC, die alle Module einer SPS enthält. Anstatt einer CPU besitzt sie einen Co-Prozessor, auf dem ein eigenes multitaskingfähiges Betriebssystem mit einem multi-ported RAM (von PC und SPS geteilter Speicher, auf den beide zugreifen können) läuft.

3 Aufbau und Strukturen industrieller Steuerungssysteme (21 von 99)

3.2.2 Slot - SPS

Desweiteren befindet sich auf der Slot-SPS mindestens eine Feldbusankopplung zur Anbindung der Sensoren und Aktoren.

Im Grunde genommen benutzt die Slot-SPS lediglich die Stromversorgung des PCs. Über den multi-ported RAM können die CPU des PCs und der Co-Prozessor der Slot-SPS Daten untereinander austauschen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (22 von 99)

3.2.3 Soft - SPS

Eine *Soft*-SPS ist reine Software. Sie läuft komplett auf der CPU des Host-PCs. Die Soft-SPS nutzt die Hardware des Host-PCs. Die Anbindung der Sensoren und Aktoren erfolgt wie bei der Slot-SPS über eine Feldbusankopplung, heute häufig über Ethernet.

3 Aufbau und Strukturen industrieller Steuerungssysteme (23 von 99)

3.2.4 Vor- und Nachteile PC-basierter SPSen

Die Vorteile der SPS im PC ergeben sich hauptsächlich dadurch, dass die rasante Entwicklung der PC-Leistung für SPSen genutzt werden kann:

- PC-basierte SPSen erreichen *höhere Verarbeitungsgeschwindigkeiten* als Hardware-SPSen.
- Ein PC kann zur *Steuerung, Programmierung und Visualisierung* verwendet werden. Somit ergeben sich preisgünstigere, einfachere und durchgängige Systemstrukturen, mit denen der Anwender gewohnt ist umzugehen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (24 von 99)

3.2.4 Vor- und Nachteile PC-basierter SPSen

Die Vorteile …

- Es entstehen *offenere Systeme*, weil der Datenaustausch auf einer einheitlichen Plattform am Häufigsten unter Windows standardisiert wird. Somit wird die Ankopplung von Bedien- und Beobachtungssystemen sowie von übergeordneten Planungssystemen an die SPS vereinfacht.

3 Aufbau und Strukturen industrieller Steuerungssysteme (25 von 99)

3.2.4 Vor- und Nachteile PC-basierter SPSen

Neben diesen Vorteilen muss der PC aber auch die *harten, industriellen Anforderungen* erfüllen, wie z.B.:

- hohe Verfügbarkeit der Hardware,
- Robustheit und Echtzeitfähigkeit des Betriebssystems,
- Erfüllung der Kommunikationsstandards für die E/A-Anbindungen,
- Funktionssicherheit, EMV (elektromagnetische Verträglichkeit),

3 Aufbau und Strukturen industrieller Steuerungssysteme (26 von 99)

3.2.4 Vor- und Nachteile PC-basierter SPSen

Neben diesen Vorteilen muss der PC aber auch die *harten, industriellen Anforderungen* erfüllen, wie z.B.:

…

- Unempfindlichkeit gegenüber rauen Umwelteinflüssen (Erschütterungen, Staub, Temperaturbereich) und
- Absicherung gegenüber Stromausfällen (USV).

3 Aufbau und Strukturen industrieller Steuerungssysteme (27 von 99)

3.2.4 Vor- und Nachteile PC-basierter SPSen

Insbesondere bei der Soft-SPS besteht die Gefahr, dass die SPS-Programme von anderen im PC ablaufenden Programmen gestört werden könnten, was im industriellen Einsatz nicht toleriert werden kann.

Durch den Einsatz von Industrie-PCs mit Echtzeit-Kernel-Betriebssystemen, die die Abarbeitung der Programme in Echtzeit, also mit determinierten Wartezeiten (Zykluszeiten), gewährleisten, werden die hohen Anforderungen an PC-basierte SPSen zunehmend erfüllt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (28 von 99)

3.2.4 Vor- und Nachteile PC-basierter SPSen

Dennoch ist die konventionelle SPS-Hardware für relativ kleine und kleinste Steuerungsaufgaben, insbesondere in der Gebäudeautomatisierung, häufig kostengünstiger, zu mindestens solange keine Visualisierung benötigt wird.

3 Aufbau und Strukturen industrieller Steuerungssysteme (29 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS

Die Informationsverarbeitung in einer SPS verläuft zyklisch, also immer wiederkehrend. Das SPS-Programm wird zyklisch abgearbeitet. Auch hier findet sich das aus der Informatik bekannte EVA-Prinzip wieder (**E**ingabe, **V**erarbeitung, **A**usgabe).

3 Aufbau und Strukturen industrieller Steuerungssysteme (30 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS

In Abb. 03.03 wird das Schema der Signalverarbeitung in einer SPS veranschaulicht.

Dabei beschreibt das EVA-Prinzip:

- das Einlesen der Sensordaten bzw. Eingabedaten (Datenpunkte aus der Anlage),
- das Verarbeiten der Eingabedaten und weitere Informationen im SPS-Programm und
- das Ausgeben der Stellsignale an die Aktoren bzw. das Ausgeben der Ausgabedaten an die Anlage.

3 Aufbau und Strukturen industrieller Steuerungssysteme (31 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS

Die Eingabedaten (z.B. Messsignale von Sensoren oder Binärsignale von Endschaltern) werden in den Eingangs-Baugruppen elektronisch angepasst. Die CPU fragt nacheinander alle Eingabe-Baugruppen ab und legt die Eingangsdaten im Arbeitsspeicher (RAM) ab. Dieser Speicherbereich befindet sich im PEA-Abbildspeicher (**P**rozess-**E**in-/**A**usgabeabbildspeicher) und wird auch *Eingangsabbild* genannt, weil die hier abgelegten Daten nicht die aktuellen, sondern die zum Abtastzeitpunkt anliegenden Daten sind.

3 Aufbau und Strukturen industrieller Steuerungssysteme (32 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS

Das SPS-Programm wird dann von der CPU jeweils Schritt für Schritt (Step by Step) abgearbeitet. Dabei werden die im Arbeitsspeicher abgelegten Operationen, wie z.B. LD (LOAD), AND und ST (STORE) einzeln adressiert, interpretiert und mit den angegebenen Operanden (Variablen und Konstanten) ausgeführt.

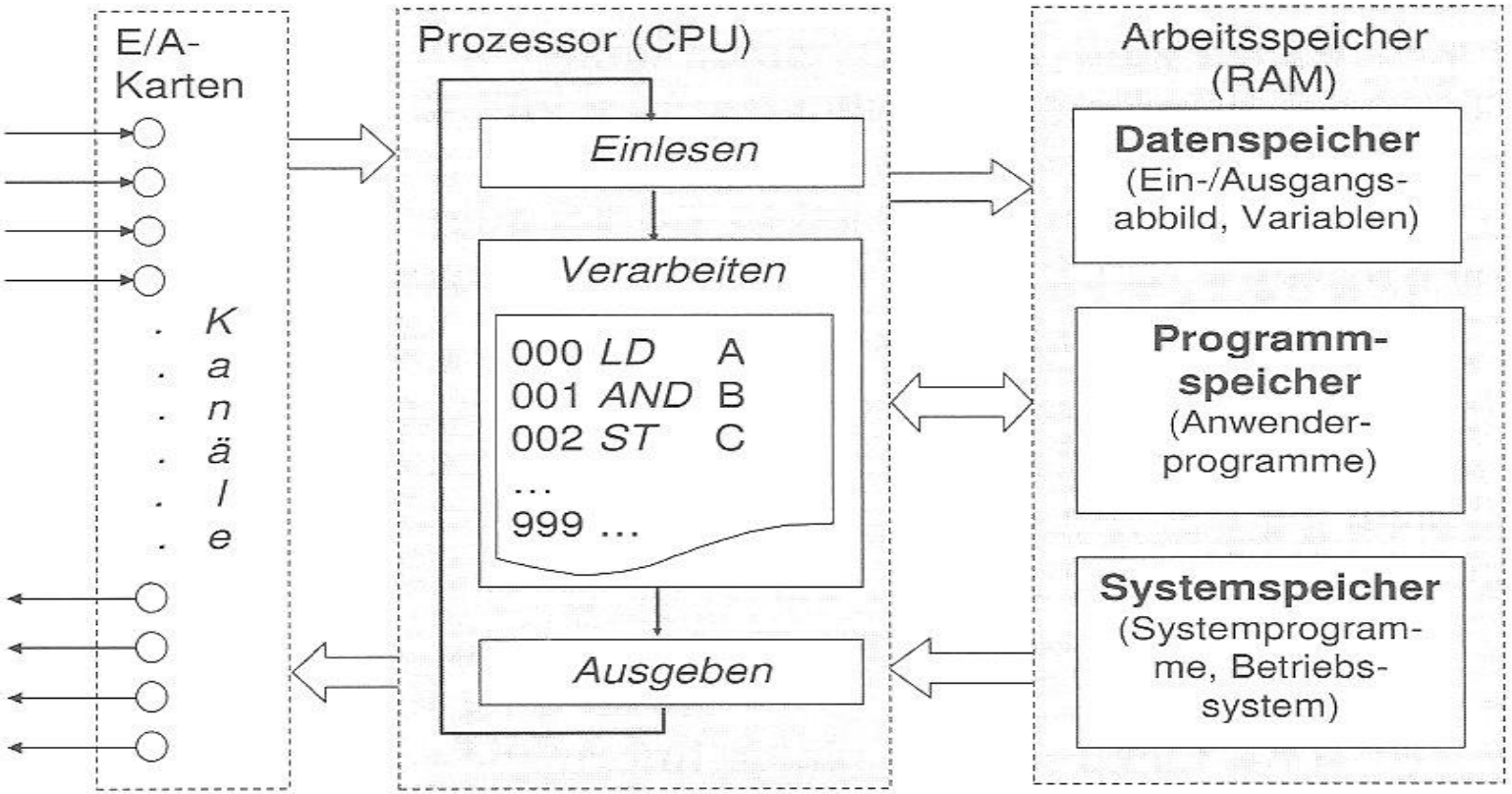
3 Aufbau und Strukturen industrieller Steuerungssysteme (33 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS

Die Operanden können direkte Adressen des Ein- oder Ausgangsabbildes sein, sollten aber im Allgemeinen als Variablen (z.B. A, B, C) deklariert werden. Prinzipiell werden in einem Programm nicht nur Daten des Ein- und Ausgangsabbildes, sondern auch Parameter und Zwischenwerte, wie Zählerstände, Zeitwerte, Sollwerte, Zustandswerte u.a. verarbeitet und im RAM gespeichert.

3 Aufbau und Strukturen industrieller Steuerungssysteme (34 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS



3 Aufbau und Strukturen industrieller Steuerungssysteme (35 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS

Wenn ein Programm Stellwerte für die Aktoren der Anlage berechnet, werden diese im Ausgangsabbild des PEA abgelegt.

Erst nach der Abarbeitung des gesamten SPS-Programmes werden die im *Ausgangsabbild* abgelegten Ausgabedaten (Stellwerte) an die Ausgangs-Baugruppen übertragen und stehen erst dann in der Anlage zur Verfügung.

3 Aufbau und Strukturen industrieller Steuerungssysteme (36 von 99)

3.3 Informationsverarbeitung in der SPS – Funktion der SPS

Der Arbeitsspeicher (RAM) lässt sich also in drei Teile gliedern:

- den *Datenspeicher* (PEA) mit Ein- und Ausgangsabbild und en verwendeten Variablen,
- den *Programmspeicher* mit den zyklisch abzuarbeitenden Anwenderprogrammen und
- den *Systemspeicher* mit den SPS-interne Systemprogrammen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (37 von 99)

3.4 Konventionelle Anbindung der Feldgeräte

Eine modular aufgebaute Hardware-SPS besitzt aus konventioneller Sicht Baugruppen für *binäre* Ein- und Ausgabesignale (BE bzw. BA) sowie für *analoge* Ein- und Ausgabesignale (AE bzw. AA).

Der Anschluss der Feldgeräte (Sensoren und Aktoren) erfolgt über Kupferleitungen, die mittels Steck- oder Schraubverbindungen an die Kanäle der Ein- oder Ausgangs-Baugruppen angeschlossen werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (38 von 99)

3.4.1 Binäre Eingänge der SPS

Eine binäre Eingangs-Baugruppe unterscheidet zwischen hohen und niedrigen Signalpegel der anliegenden Eingangsspannung. Wie Abb. 03.04 zeigt, wird die Eingangs-Baugruppe von einer 24V-Gleichspannung (DC) versorgt. Hat der binäre Sensor (Niveauschalter) ausgelöst, so ist der Kontakt geschlossen. Es liegt also ein Signalpegel von ca. 24 VDC am Eingang der binären Eingangs-Baugruppe an. Hat der binäre Sensor nicht ausgelöst, so liegt ein Signalpegel von ca. 0 VDC am Eingang der binären Eingangs-Baugruppe an.

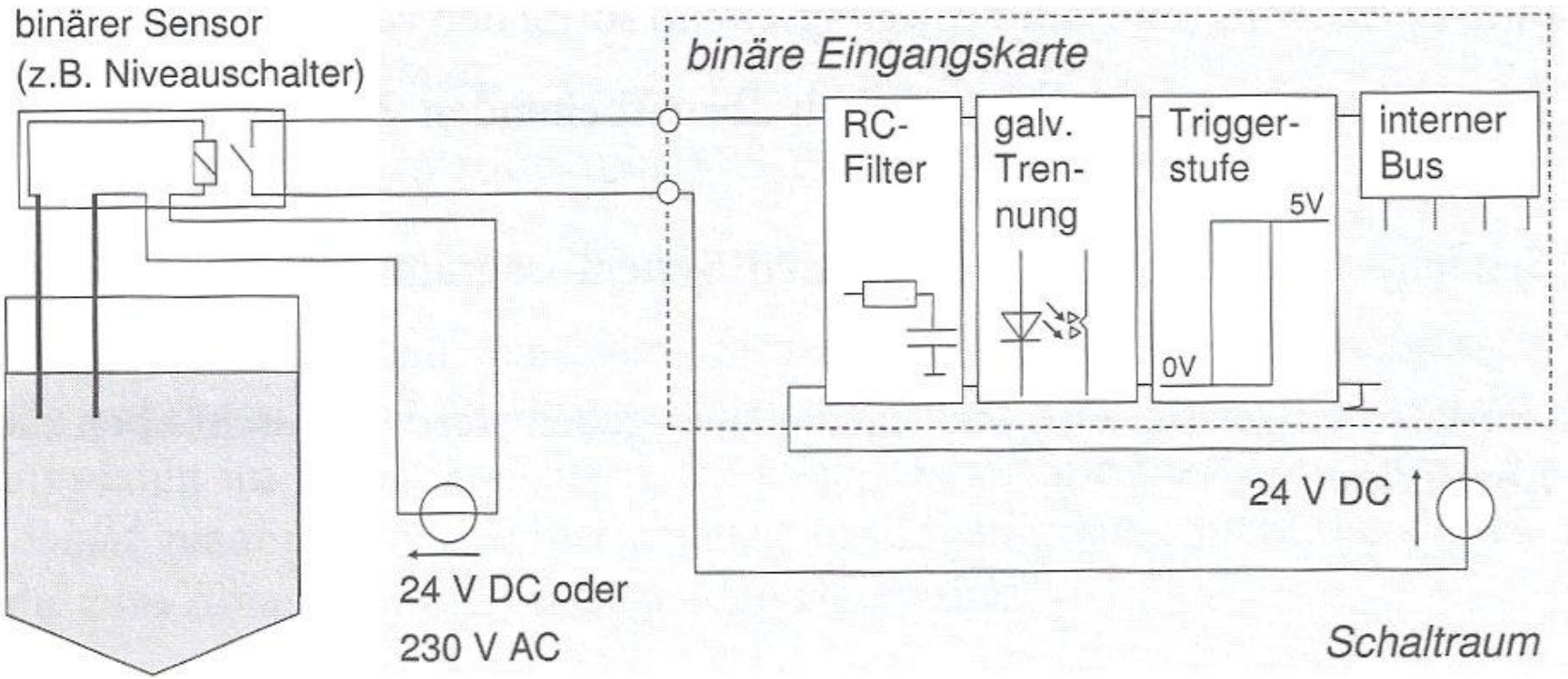
3 Aufbau und Strukturen industrieller Steuerungssysteme (39 von 99)

3.4.1 Binäre Eingänge der SPS

Diese beiden Signalpegel werden gleich hinter dem Eingang der binären Eingangs-Baugruppe über ein RC-Filter geleitet, das das binäre Eingangssignal von überlagerten Störsignalen (Rauschen) und Kontaktprellen befreit. Anschließend wird das aus der Anlage kommende Signal über einen Optokoppler geleitet, wodurch eine galvanische Trennung der anlagennahen Signale von den in der SPS verarbeiteten Signale durchgeführt wird, um Störungen bei der Signalverarbeitung innerhalb der SPS zu verhindern.

3 Aufbau und Strukturen industrieller Steuerungssysteme (40 von 99)

3.4.1 Binäre Eingänge der SPS



3 Aufbau und Strukturen industrieller Steuerungssysteme (41 von 99)

3.4.2 Binäre Ausgänge der SPS

Binäre Ausgangs-Baugruppen erzeugen aus den Boole’schen Ausgangsvariablen TRUE oder FALSE ein Signals, das über einen Optokoppler an einen binären Schaltverstärker geleitet wird. In vielen Fällen besitzen die binären Ausgangs-Baugruppen Ausgangsrelais, deren Kontakte das Ausgangssignal an die Anlage übergeben. Häufig werden aber auch Schaltverstärker mit Transistoren verwendet, die ein elektronisches Signal an die Anlage ausgeben (siehe Abb. 03.05).

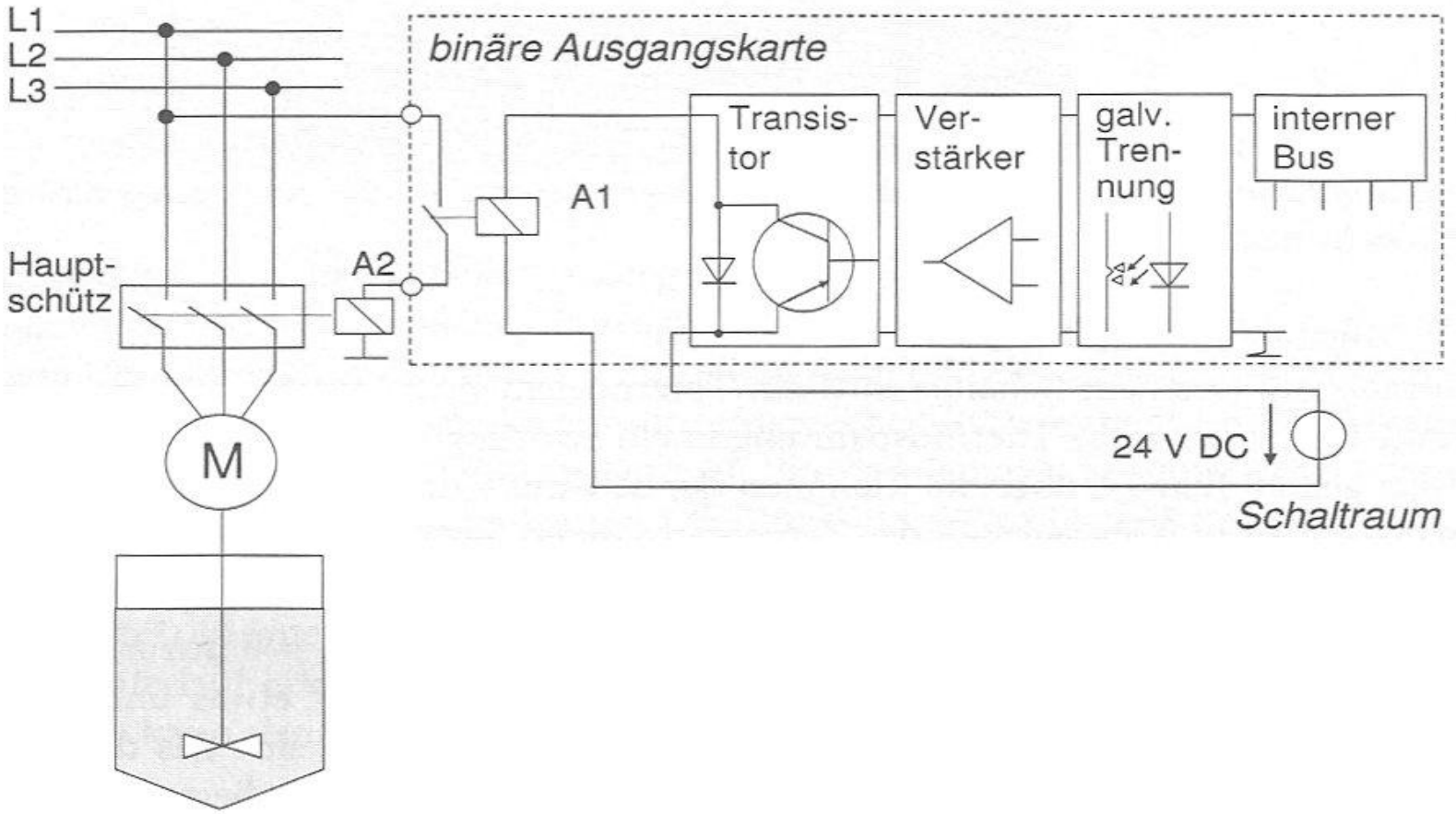
3 Aufbau und Strukturen industrieller Steuerungssysteme (42 von 99)

3.4.2 Binäre Ausgänge der SPS

Diese Transistorausgänge schalten 24 VDC bei einem Strom von i.d.R. maximal 0,5 A und sind über eine integrierte Freilaufdiode vor Überspannungen beim Anschalten von induktiven Verbrauchern (Relais, Magnetventil) geschützt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (43 von 99)

3.4.2 Binäre Ausgänge der SPS



3 Aufbau und Strukturen industrieller Steuerungssysteme (44 von 99)

3.4.3 Analoge Eingänge der SPS

Ein Sensor wandelt die physikalische Messgröße (z.B. Füllstand, Druck, Durchfluss, Temperatur, Drehzahl) entweder in einen elektrischen Strom-, Spannungs- oder Widerstandswert um. Strom- und Spannungsgeber werden im Allgemeinen über zwei Leitungen auf den entsprechenden Kanal einer analogen Eingangs-Baugruppe geführt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (45 von 99)

3.4.3 Analoge Eingänge der SPS

Man unterscheidet aktive und passive Sensoren. Passive Sensoren, sogenannte Zweidraht-Messumformer, werden über die Analogeingangsklemmen von der SPS (24 VDC) gespeist. Abb. 03.06 zeigt das. Aktive Sensoren besitzen dagegen oft eine separate Spannungsversorgung und werden deshalb über eine Vierdraht-Leitung (2 Leitungen für die Stromversorgung, 2 Leitungen für das eigentliche Messsignal) angeschlossen.

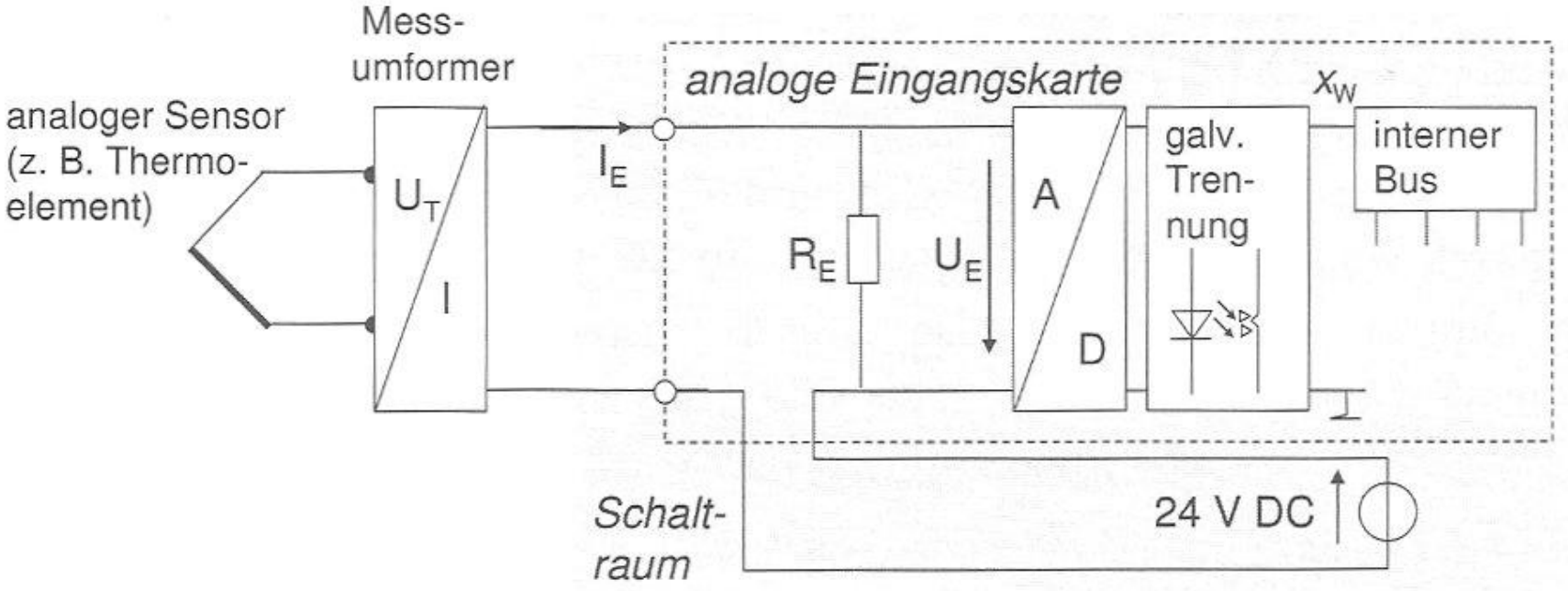
3 Aufbau und Strukturen industrieller Steuerungssysteme (46 von 99)

3.4.3 Analoge Eingänge der SPS

Moderne aktive Sensoren lassen sich aber z.T. auch über eine Zweidraht-Leitung betreiben, wenn das Messsignal als 4 … 20 mA-Signal übertragen wird. So kann die Elektronik des Sensors aus dem Grundstrom von etwas weniger als 4 mA versorgt werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (47 von 99)

3.4.3 Analoge Eingänge der SPS



3 Aufbau und Strukturen industrieller Steuerungssysteme (48 von 99)

3.4.3 Analoge Eingänge der SPS

Die analoge Eingangs-Baugruppe führt für jeden Kanal eine Analog-/Digital-Umwandlung durch. Der digitalisierte Messwert x_W wird nach der galvanische Trennung gemäß Tabelle 03.01 als 16-Bit-Datenwort im Eingangsabbild des Arbeitsspeichers (PEA) abgelegt. Je nach Bedarf gibt es Baugruppen mit 10-, 12- oder 16-Bit-Auflösung.

3 Aufbau und Strukturen industrieller Steuerungssysteme (49 von 99)

3.4.3 Analoge Eingänge der SPS

Tabelle 03.01: Umwandlung eines analogen Stromsignals I_E in die digitale Größe x_W, die sowohl dezimal als auch in Form eines 16-Bit-Datenwortes angegeben ist.

I_E	x_W	*Eingangsdatenwort* x_w															
[mA]	*dezimal*	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
> 22,810	32767	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22,810	32511	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
20,0005	27649	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	1
20,0	27648	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
16,0	20736	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
4,0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,9995	–1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1,1852	–4864	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0
< 1,1852	–32768	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3 Aufbau und Strukturen industrieller Steuerungssysteme (50 von 99)

3.4.3 Analoge Eingänge der SPS

Im vorliegenden Beispiel wird für die Übertragung der analogen Messsignale ein Einheitsstromsignal der Größe 4 … 20 mA verwendet.

Ein Bit entspricht einem Strom von $\mathbf{I_E = (20\text{-}4)\ mA\ /\ 27648 = 0{,}58\ \mu A}$.

Werte die unterhalb von $\mathbf{I_E = 4\ mA}$ liegen ($\mathbf{x_W < 0}$, also **negativ**), lassen auf einen Drahtbruch schließen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (51 von 99)

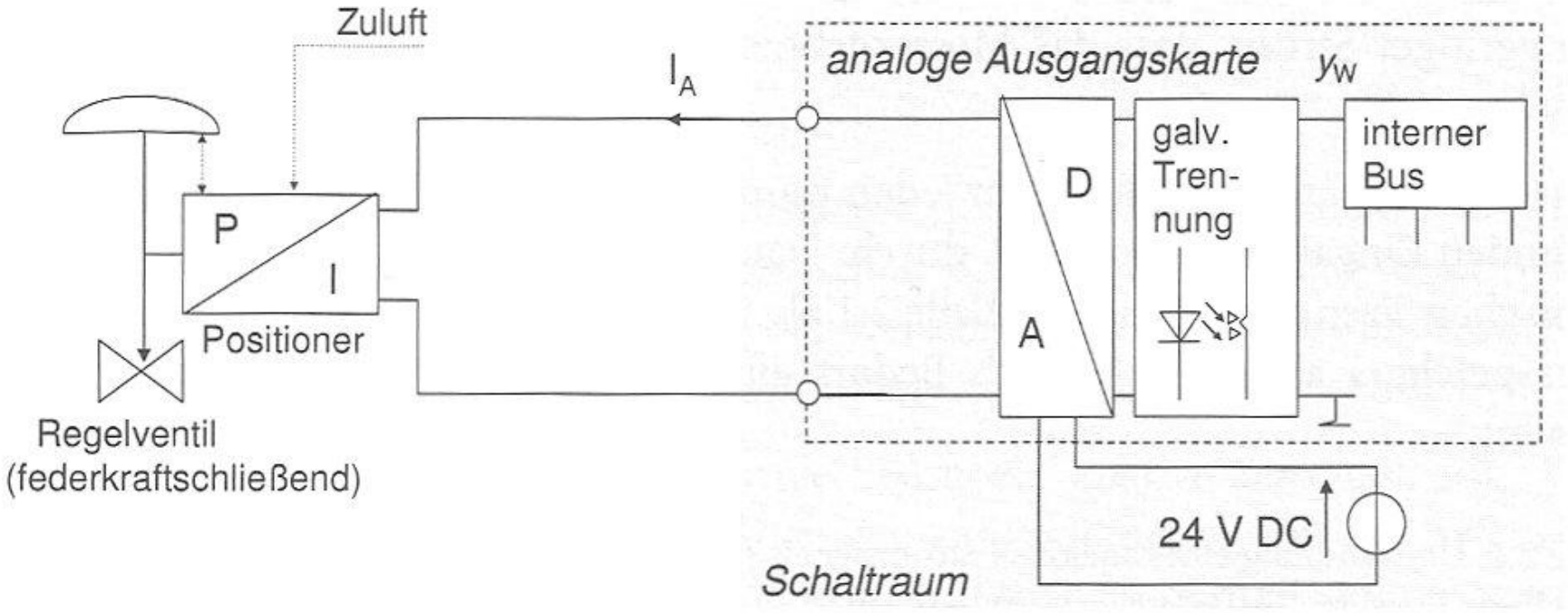
3.4.4 Analoge Ausgänge der SPS

Zur Ansteuerung eines analogen Stellgliedes, wie z.B. eines kontinuierlich verstellbaren Regelventils in Abb. 03.07, gibt die SPS den berechneten analogen Stellwert über eine analoge Ausgangs-Baugruppe aus. Auch hier wird wieder eine galvanische Trennung über Optokoppler und dann eine Digital-/Analog-Umwandlung durchgeführt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (52 von 99)

3.4.4 Analoge Ausgänge der SPS

Es entsteht ein analoges Strom- oder Spannungssignal, das an den Aktor geführt wird.



3 Aufbau und Strukturen industrieller Steuerungssysteme (53 von 99)

3.4.4 Analoge Ausgänge der SPS

Die in diesem Abschnitt beschriebenen Möglichkeiten, binäre und analoge Signale in die SPS einzulesen oder von der SPS auszugeben, sind mit einem großen Verkabelungsaufwand verbunden. Für jedes Signal sind mindestens zwei, wenn nicht sogar vier Leitungen, also jeweils ein Kabel zu verlegen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (54 von 99)

3.4.4 Analoge Ausgänge der SPS

Die in diesem Abschnitt beschriebenen Möglichkeiten, binäre und analoge Signale in die SPS einzulesen oder von der SPS auszugeben, sind mit einem großen Verkabelungsaufwand verbunden. Für jedes Signal sind mindestens zwei, wenn nicht sogar vier Leitungen, also jeweils ein Kabel zu verlegen.

Damit entsteht ein erheblicher Material- und Verlegungsaufwand, der durch moderne Feldbustechnik wesentlich verringert werden kann. Das soll im folgenden Abschnitt gezeigt werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (55 von 99)

3.4.4 Analoge Ausgänge der SPS

Dennoch wird die klassische Zwei- und Vierleitertechnik nicht verschwinden, da sie für kleine und kompakte Anlagen, aber auch beim Anschluss konventioneller Sensoren und Aktoren an dezentrale Feldbuseinheiten (Remote-I/O) , weiterhin eine einfache und zuverlässige Anschlussart darstellt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (56 von 99)

3.5 Busankopplung der Feldgeräte

Der Grundgedanke von Feldbussystemen ist es, in erster Linie elektrische Kabel und Leitungen einzusparen.

Mit Feldbussystemen ist es möglich mehrere Signale gleichzeitig über eine einzige Busleitung zu übertragen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (57 von 99)

3.5.1 Feldbussysteme und -strukturen

In der klassischen Verdrahtungstechnik muss eine Sortierung der Leitungen von den räumlich benachbarten Sensoren und Aktoren in der Anlage zu den funktional zusammenhängenden Ein- und Ausgabekanälen der Ein- und Ausgangs-Baugruppen der SPS erfolgen.

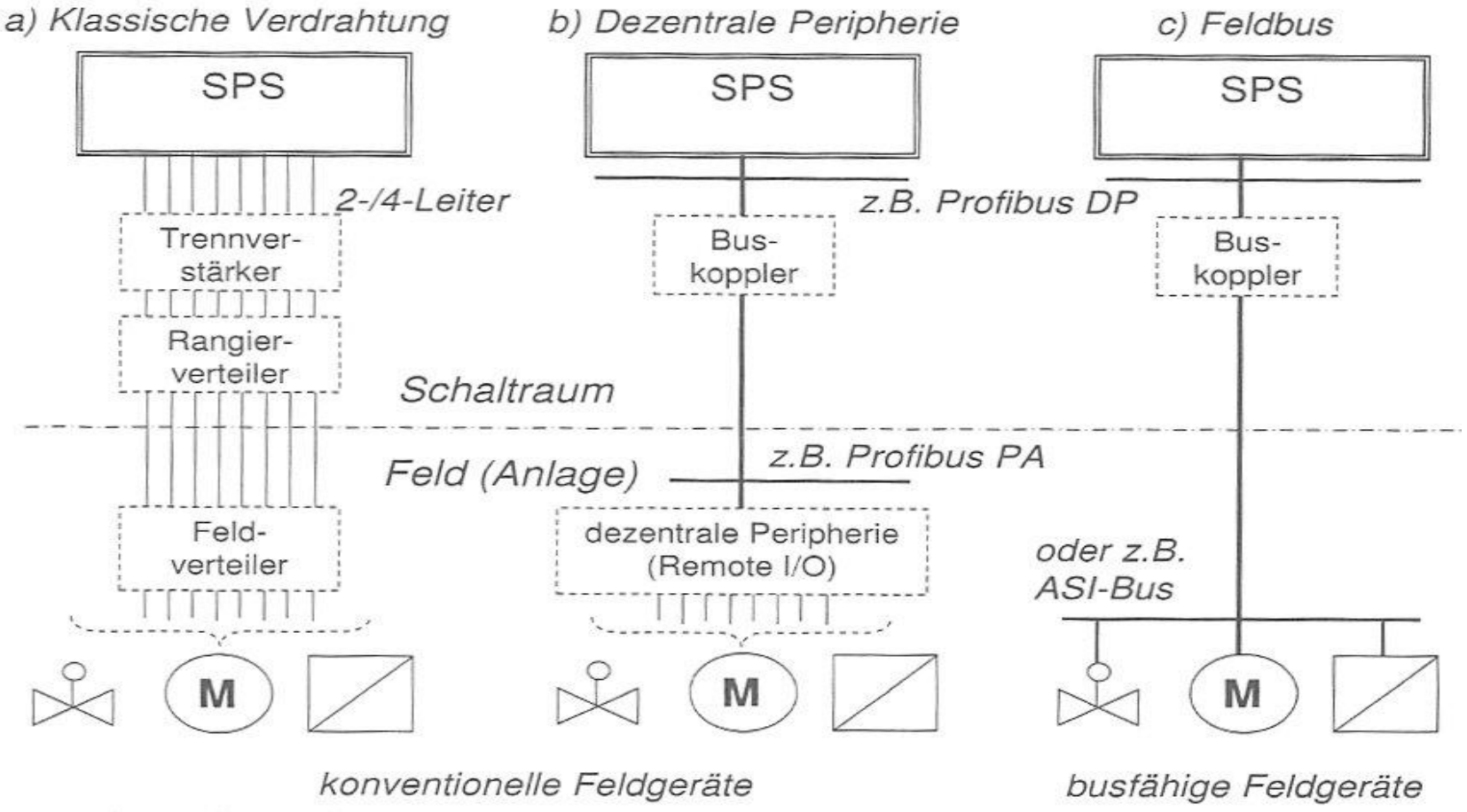
3 Aufbau und Strukturen industrieller Steuerungssysteme (58 von 99)

3.5.1 Feldbussysteme und -strukturen

Wie Abb. 03.08a zeigt, werden räumlich benachbarte Einzelleitungen von den Feldgeräten in einem Feldverteiler zu einem Stammkabel zusammengefasst und in den Schaltraum verlegt. Dort werden die Einzelleitungen in einem Rangierverteiler erneut aufgelegt und geordnet. Diese Umverdrahtung ist notwendig, weil die im Feld räumlich benachbarten Leitungen nun funktional in der Reihenfolge zusammengefasst werden müssen, in der sie mit den E/A-Kanälen der Ein- und Ausgangs-Baugruppen der SPS verbunden werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (59 von 99)

3.5.1 Feldbussysteme und –strukturen



3 Aufbau und Strukturen industrieller Steuerungssysteme (60 von 99)

3.5.1 Feldbussysteme und -strukturen

Mit der in Abb. 03.08c dargestellten Feldbustechnik werden nun Feldgeräte und SPS ausschließlich an eine Busleitung angeschlossen. Auch hier muss ggf. eine Umsetzung zwischen zwei Bussystemen erfolgen.

Voraussetzung für eine Feldbusankopplung ist, dass alle Feldgeräte über einen Mikroprozessor mit busfähiger Schnittstelle verfügen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (61 von 99)

3.5.1 Feldbussysteme und -strukturen

Das ist bei den meisten älteren und klassischen Feldgeräten nicht der Fall. Deshalb setzte sich zunächst die in Abb. 03.08b gezeigte „Zwischenlösung“ mit einer Feldgeräteankopplung über einen dezentralen Feldmultiplexer durch. Diese dezentrale Peripherie (DP) ermöglicht es, auch nicht busfähige Feldgeräte anzuschließen, was auch als Remote-I/O-System (RIO) bezeichnet wird.

3 Aufbau und Strukturen industrieller Steuerungssysteme (62 von 99)

3.5.1 Feldbussysteme und -strukturen

Das Remote-I/O-System (RIO) verfügt einerseits über die E/A-Baugruppen, die früher zentral im SPS-Schrank steckten und andererseits über eine Feldbusschnittstelle, die eine Busverbindung zur SPS ermöglicht.

3 Aufbau und Strukturen industrieller Steuerungssysteme (61 von 99)

3.5.1 Feldbussysteme und -strukturen

Die wichtigsten Vorteile der Feldbustechnik sind:

- Einsparung von Kabeln zwischen Anlage und Schaltraum,
- höhere Auflösung der Messwerte,
- Übertragung zusätzlicher Geräteinformationen (Grenzwerte, Zustände, Alarme, Betriebszeiten u.a.),
- zentrale Diagnosemöglichkeit (d.h. Fehlersuche von der Leitwarte aus),

3 Aufbau und Strukturen industrieller Steuerungssysteme (62 von 99)

3.5.1 Feldbussysteme und -strukturen

Die wichtigsten Vorteile der Feldbustechnik sind:

…

- dezentrale Intelligenz (Rechenoperationen, wie z.B. Dosieren im Feldgerät),
- Platzeinsparung im Schaltraum,
- geringerer Montage- und Planungsaufwand,
- geringerer Nachrüst- und Änderungsaufwand und
- zentrale Konfigurierung der Bussysteme am PC.

3 Aufbau und Strukturen industrieller Steuerungssysteme (63 von 99)

3.5.1 Feldbussysteme und -strukturen

Derzeit gibt es am Markt verschiedene Feldbussysteme, die sich für verschiedene Industrieanwendungen durchgesetzt haben. Während für die Fertigungsautomatisierung Zykluszeiten von kleiner 10 ms in relativ kleinen Bussegmenten (Kabellängen kleiner 100 m) gefordert werden, ist die Ausdehnung von Chemieanlagen i.d.R. größer, und die Zykluszeitanforderungen liegen dabei im Bereich von 100 ms und sogar darüber.

In Tabelle 03.02 werden die zurzeit gängigsten Feldbussysteme gegenüber gestellt.

3 Aufbau und Strukturen (64 von 99)

Bussystem	Einsatzfelder	Eigenschaften
ASI-Bus (**A**ktor-**S**ensor-**I**nterface)	Ankopplung einfacher Sensoren und Aktoren, wie Lichtschranken, Ventile, Lampen etc., zumeist in der Fertigungstechnik.	• Kostengünstig. • Schnelle Übertragung weniger binärer Daten. • Übertragung von Energie und Daten über dasselbe Kabel.
Profibus (**Pro**cess **Fi**eld **Bus**)	Profibus-DP für Datenaustausch mit **d**ezentraler **P**eripherie wie Antrieben, Ventilen und Messumformern in Fertigungs- und Verfahrenstechnik.	• Schneller, zyklischer Datenaustausch der SPS mit dezentralen Feldgeräten (Zykluszeit < 10 ms). • Auch Datenaustausch zwischen SPSen im Multi-Masterbetrieb möglich.
	Profibus-PA für Anwendungen in explosionsgefährdeten Umgebungen in der **Prozessa**utomatisierung.	• Vergleichsweise langsame Datenübertragung. • Gewährleistet Eigensicherheit der angekoppelten Sensoren und Aktoren, indem die Versorgungsspannung unterhalb der Zündenergie von explosiven Gasen gehalten wird.
Profinet (**Pro**cess **Fi**eld Ether**Net**)	Profinet-IO (Input Output) zur Ansteuerung von Sensoren und Aktoren durch eine zentrale Steuerung in der Fertigungstechnik.	• Zykluszeiten unter 1 ms im isochronen Modus möglich. • Kompatible Kabel- und Netzwerktechnik mit dem Ethernet der PC-Welt. • Relativ einfache Erweiterbarkeit.
	Profinet-CBA (Component Based Automation) zur Vernetzung von SPSen.	• Wie bei Profinet-IO, Zykluszeiten jedoch im Bereich von 50 ms.
EIB (**E**uropäischer **I**nstallations**b**us)	Kommunikation zwischen Sensoren und Aktoren zur Gebäudeautomatisierung.	• Steuerung erfolgt dabei über den Benutzer selbst oder über einen mit entsprechender Software ausgerüsteten Netzwerkcomputer.
SERCOS (**Se**rial **R**ealtime **Co**mmunication **S**ystem)	Kommunikation zwischen Steuerungen und Antrieben in der Fertigungstechnik.	• Synchronisierung digitaler Servoantriebe hochgenau und in Echtzeit. • Regelung von beispielsweise 40 Achsen mit einer Zykluszeit von 1 ms.
CAN-Bus (**C**ontroller **A**rea **N**etwork)	Vernetzung von Steuergeräten in Automobilen.	• Ausdehnung abhängig von Übertragungsrate 40 m bei 1 Mbit/s, hohe Datensicherheit.

3 Aufbau und Strukturen industrieller Steuerungssysteme (65 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Wie erfolgt nun die Datenübertragung in Feldbussystemen?

Ähnlich wie beim PC, bei dem man zwischen einer parallelen und seriellen Datenübertragung zu Drucker bzw. Maus und Tastatur unterscheidet, kann man die klassische 2-/4-Leitertechnik als parallele Schnittstelle ansehen, von der aus die Daten für jedes Signal zeitgleich ein- und ausgegeben wird.

3 Aufbau und Strukturen industrieller Steuerungssysteme (66 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Dementsprechend erfolgt die Feldbusankopplung über eine serielle Schnittstelle, denn hier werden die Daten hintereinander über zwei Leitungen (eine zum Senden TxD, eine zum Empfangen RxD) übertragen.

Den Vorteile der erheblich kleineren Anzahl von Leitungen erkauft man sich aber mit höheren Übertragungszeiten, denn die Daten der einzelnen Signale müssen nacheinander übertragen und durch eine Start- und Ende-Kennzeichnung voneinander getrennt werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (67 von 99)

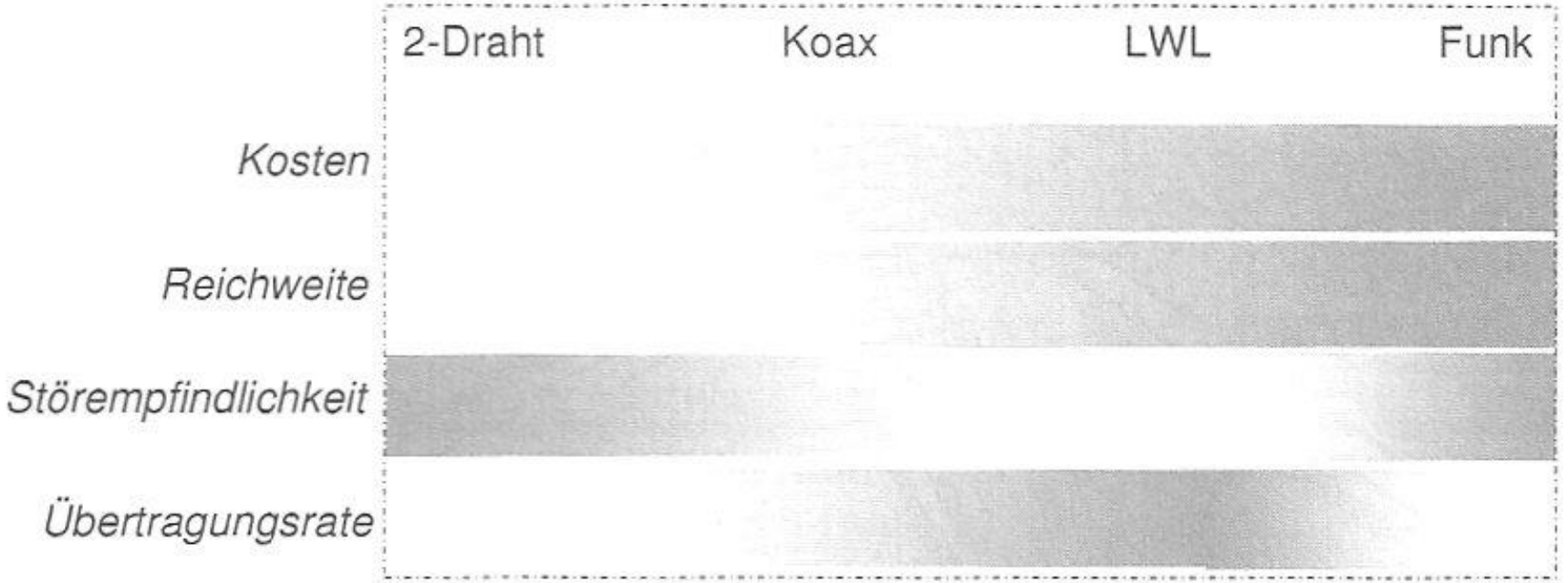
3.5.2 Datenübertragung in Feldbussystemen

Wie in Abb. 03.09 dargestellt, ist die Übertragung mit verschiedenen physikalischen Übertragungsmedien möglich, angefangen von der verdrillten Zweidrahtleitung (Doppelader DA) aus Kupfer, über ein Koaxialkabel, Lichtwellenleiter (LWL), bis hin zu Funkwellen (Bluetooth, WLAN).

3 Aufbau und Strukturen industrieller Steuerungssysteme (68 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Die Wahl des Mediums hängt von den in Abb. 03.09 gegenübergestellten Faktoren ab.



3 Aufbau und Strukturen industrieller Steuerungssysteme (69 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Die elektrischen Eigenschaften der Busleitung sind durch den Schnittstellenstandard RS485 festgelegt. Hiernach kann ein Bussegment bis zu 1200 m für 1 Master und 31 Slaves mit einer Übertragungsrate von 1,5 Mbit/s ausgelegt werden.

Im Gegensatz dazu leistet eine serielle PC-Schnittstelle (COM) nach dem Schnittstellenstandard RS232 (V.24) nur eine maximale Leitungslänge von 15 m bei 1 Master und 1 Slave und einer Übertragungsrate von 19,2 kBit/s.

3 Aufbau und Strukturen industrieller Steuerungssysteme (70 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Die RS485-Schnittstelle ist also busfähig, schneller und von größerer Reichweite.

Damit nun die SPS mit den Feldgeräten Daten austauschen kann, müssen alle Busteilnehmer determiniert auf den Bus zugreifen können. Das für diese Kommunikation erforderliche Buszugriffsverfahren ist das Master-/Slave-Verfahren. Dabei stellt i.d.R. die SPS den Master und die Feldgeräte die Slaves.

3 Aufbau und Strukturen industrieller Steuerungssysteme (71 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Master-/Slave-Verfahren:

Nur der Master darf die Kommunikation anstoßen. Die Slaves antworten nur auf die Anforderungen des Masters.

Damit ergibt sich folgender Kommunikationsablauf:

- Will der Master Daten von einem Slave empfangen, so fordert er die Daten durch ein Request-Telegramm an.

3 Aufbau und Strukturen industrieller Steuerungssysteme (72 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Damit ergibt sich folgender Kommunikationsablauf:

…

- Dieses Telegramm wird nun auf dem Bus von jedem Slave dahin gehend geprüft, ob die Anfrage an ihn gerichtet ist. Ist das der Fall, handelt es sich bei dem Slave also um den adressierten Slave, so antwortet der Slave mit einem Response-Telegramm, mit dem er die angeforderten Daten an den Master zurücksendet.

3 Aufbau und Strukturen industrieller Steuerungssysteme (73 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Damit ergibt sich folgender Kommunikationsablauf:

…

- Will der Master Daten an den Slave senden, so schreibt er neben der Zieladresse des anzusprechenden Slaves auch die Sendedaten in das Send-Telegramm.
- Der angesprochene Slave liest die Daten ein und sendet ein Acknowledge-Telegramm an den Master zurück.

3 Aufbau und Strukturen industrieller Steuerungssysteme (74 von 99)

3.5.2 Datenübertragung in Feldbussystemen

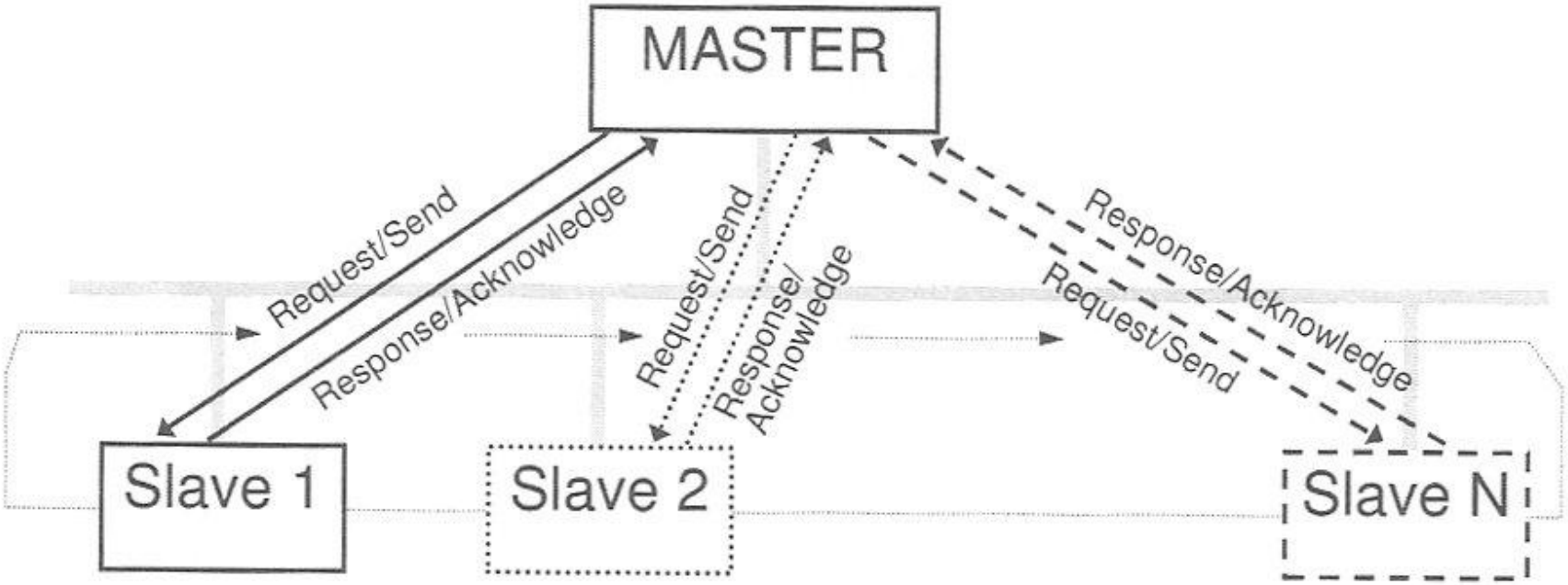
Der Aufbau der vier Telegrammtypen umfasst folgende Informationen:

<Startbyte, Quelladresse, Zieladresse, Steuerbyte, Daten, Prüfbyte, Endbyte>

Im Steuerbyte wird unterschieden, ob es sich um ein Send-, Request-, Response- oder Acknowledge-Telegramm handelt.

3 Aufbau und Strukturen industrieller Steuerungssysteme (75 von 99)

3.5.2 Datenübertragung in Feldbussystemen



3 Aufbau und Strukturen industrieller Steuerungssysteme (76 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Da stets nur der angesprochene Slave Daten senden darf, kann erst dann eine neue Abfrage erfolgen, wenn die aktuelle Kommunikation abgeschlossen ist, der Master also die angeforderten Daten erhalten hat.

Die SPS muss also die Feldgeräte nacheinander ansprechen. Wir sprechen daher auch von Polling (to poll = wählen).

3 Aufbau und Strukturen industrieller Steuerungssysteme (77 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Voraussetzung für die Buskommunikation ist, dass alle Geräte und deren Eigenschaften bekannt sind. Vor der Inbetriebnahme muss die Busstruktur konfiguriert werden. Dabei ist festzulegen, welcher Master, welches Bussystem und welche Slaves an der Kommunikation teilnehmen.

Da die Sensoren und Aktoren von verschiedenen Herstellern stammen, muss jeder Hersteller eine Gerätestammdatei (GSD) für sein Feldgerät mitliefern. Diese Datei wird bei der Buskonfiguration eingelesen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (78 von 99)

3.5.2 Datenübertragung in Feldbussystemen

Die GSD beinhaltet charakteristische Gerätemerkmale wie

- Hersteller, Gerätetyp,
- Baudrate (Übertragungsgeschwindigkeit),
- Diagnose-Codes (Meldungstypen),
- Art und Anzahl der Messwerte bzw. Stellwerte (Datenformate u.a.)

3 Aufbau und Strukturen industrieller Steuerungssysteme (79 von 99)

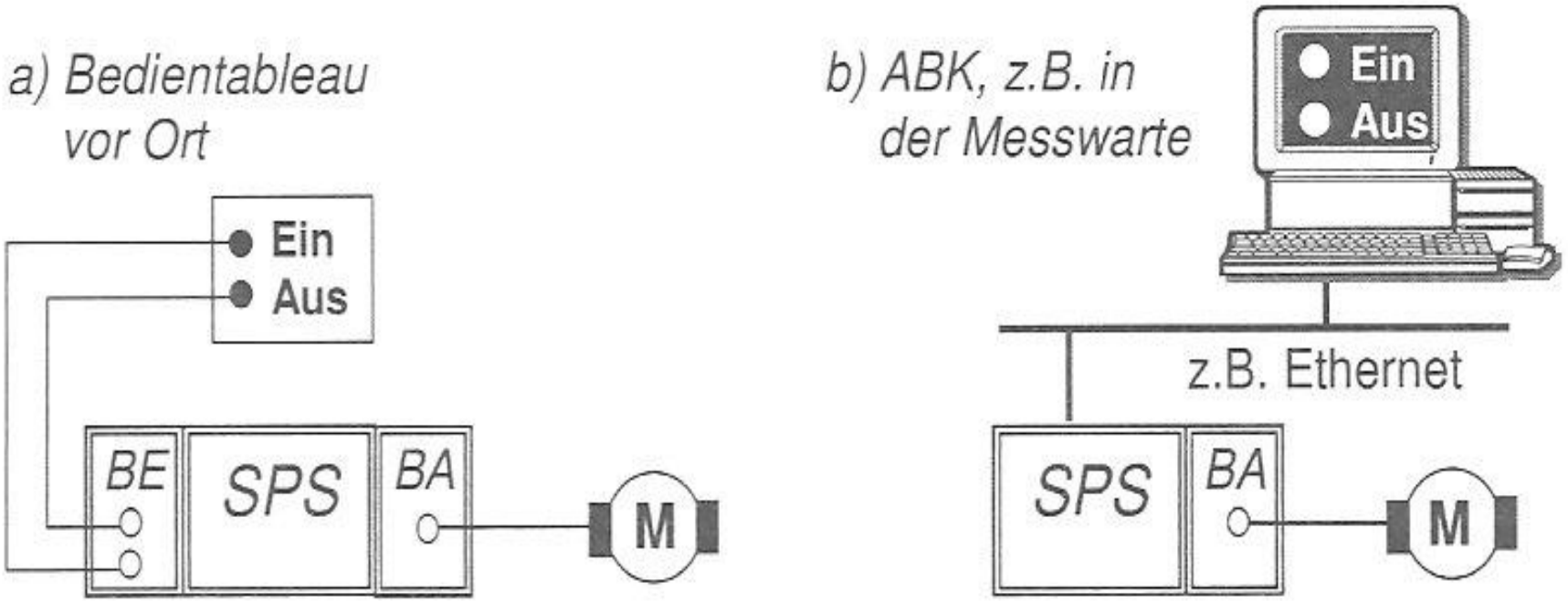
3.6 Bedienen und Beobachten

Die Bedienung und Beobachtung des von der SPS automatisierten Prozesses erfolgt häufig noch vor Ort über ein Bedientableau in unmittelbarer Nähe des Prozesses (siehe Abb. 03.11a).

Auf dem Bedientableau befinden sich zum einen Schalter zur Bedienung des Prozesses und zum anderen Leuchten zur Signalisierung bestimmter Prozesszustände. Die Schalter und Leuchten sind dabei als physikalische Ein- und Ausgänge der SPS verdrahtet.

3 Aufbau und Strukturen industrieller Steuerungssysteme (80 von 99)

3.6 Bedienen und Beobachten



3 Aufbau und Strukturen industrieller Steuerungssysteme (81 von 99)

3.6 Bedienen und Beobachten

Heutzutage werden die meisten Prozesse, Maschinen und Anlagen auch vor Ort durch Anzeige- und Bedienkomponenten (ABKen) bedient und beobachtet. Eine ABK ist ein Rechner mit einer Software zur Prozessvisualisierung.

Damit wird es dem Bediener ermöglicht, Prozesszustände auch aus der Ferne zu beobachten und per Mausklick Geräte zu Bedienen (siehe Abb. 03.11b).

3 Aufbau und Strukturen industrieller Steuerungssysteme (82 von 99)

3.6.1 Prozessvisualisierung

Ein Prozessvisualisierungssystem besitzt eine spezielle Software, mit der Prozesse schematisch dargestellt und Prozesszustände farblich gekennzeichnet werden. In dem in Abb. 03.12 dargestellten Beispiel ist der Behälter leer, so dass der Niveauschalter LS- farblich als TRUE markiert ist.

Das wird dadurch erreicht, dass im Dialog zur Dynamisierung unter Farbwechsel der Name der Variablen in der SPS eingetragen wird, die den Zustand des Sensorsignals angibt.

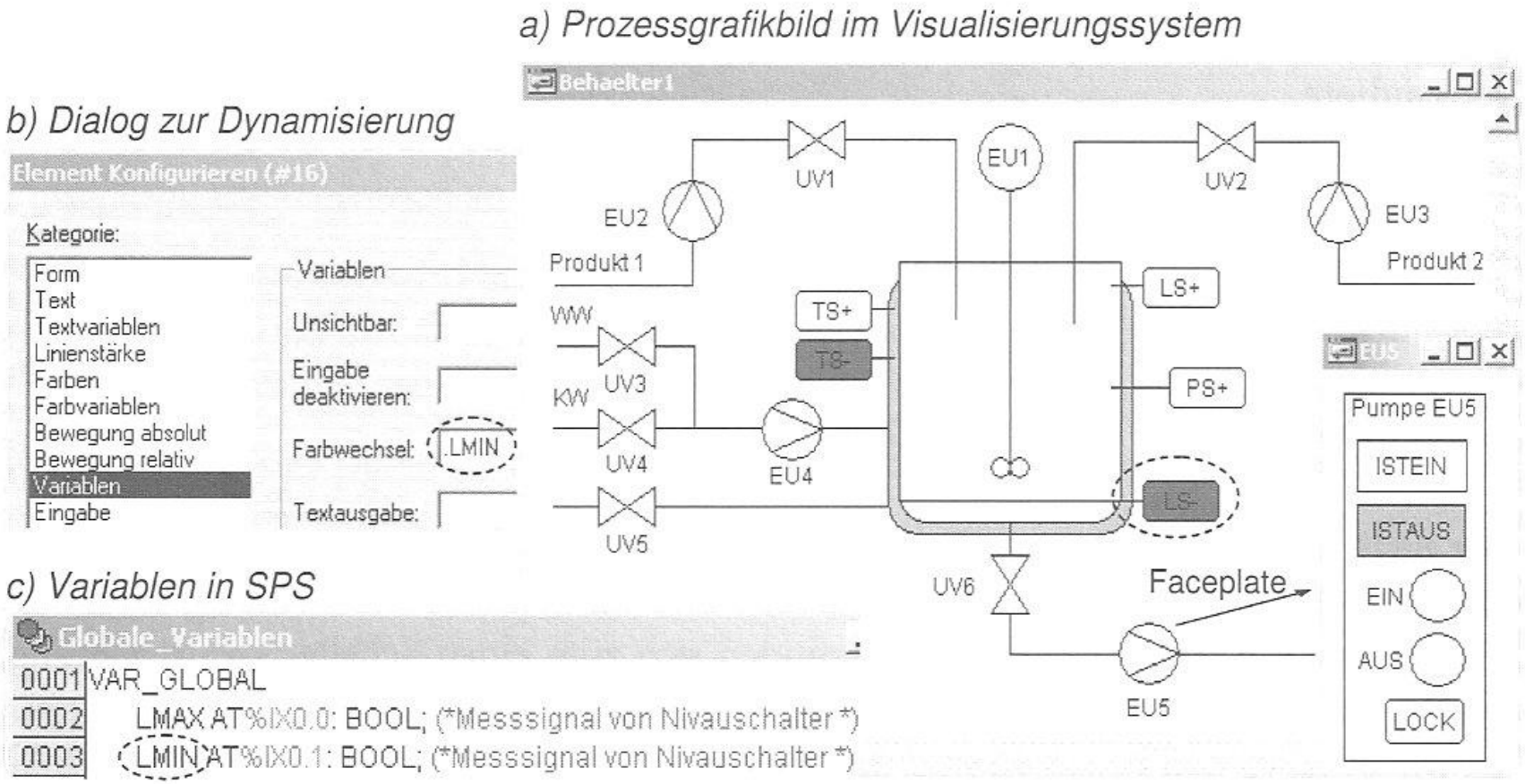
3 Aufbau und Strukturen industrieller Steuerungssysteme (83 von 99)

3.6.1 Prozessvisualisierung

Darüber hinaus kann der Bediener Schalter oder Taster als Soft-Keys betätigen, wodurch eine Aktivierung der SPS-Variablen erfolgt. Per Mausklick auf das Symbol (oder direkt mit dem Finger bei einem Touch-Screen) der Pumpe EU5 lässt sich hierfür das zugehörige Bedienfenster, das sogenannte Faceplate, öffnen, über das der Bediener die Pumpe ein- oder ausschalten kann.

3 Aufbau und Strukturen industrieller Steuerungssysteme (84 von 99)

3.6.1 Prozessvisualisierung



3 Aufbau und Strukturen industrieller Steuerungssysteme (85 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Für die Kopplung zwischen Visualisierungssystemen und SPSen gibt es wie in Abb. 03.13 dargestellt, verschiedene Konzepte.

Prinzipiell unterscheidet man Visualisierungssysteme, die wie z.B. InTouch (Fa. Wonderware) getrennt von der SPS laufen, und solche, die in das SPS-Programm integriert sind, wie z.B. CoDeSys (Fa. 3S). Wenn diese auf einer PC-basierten SPS implementiert sind, kann die ABK wie in Abb. 03.13a direkt an die Monitorschnittstelle des PCs angeschlossen werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (86 von 99)

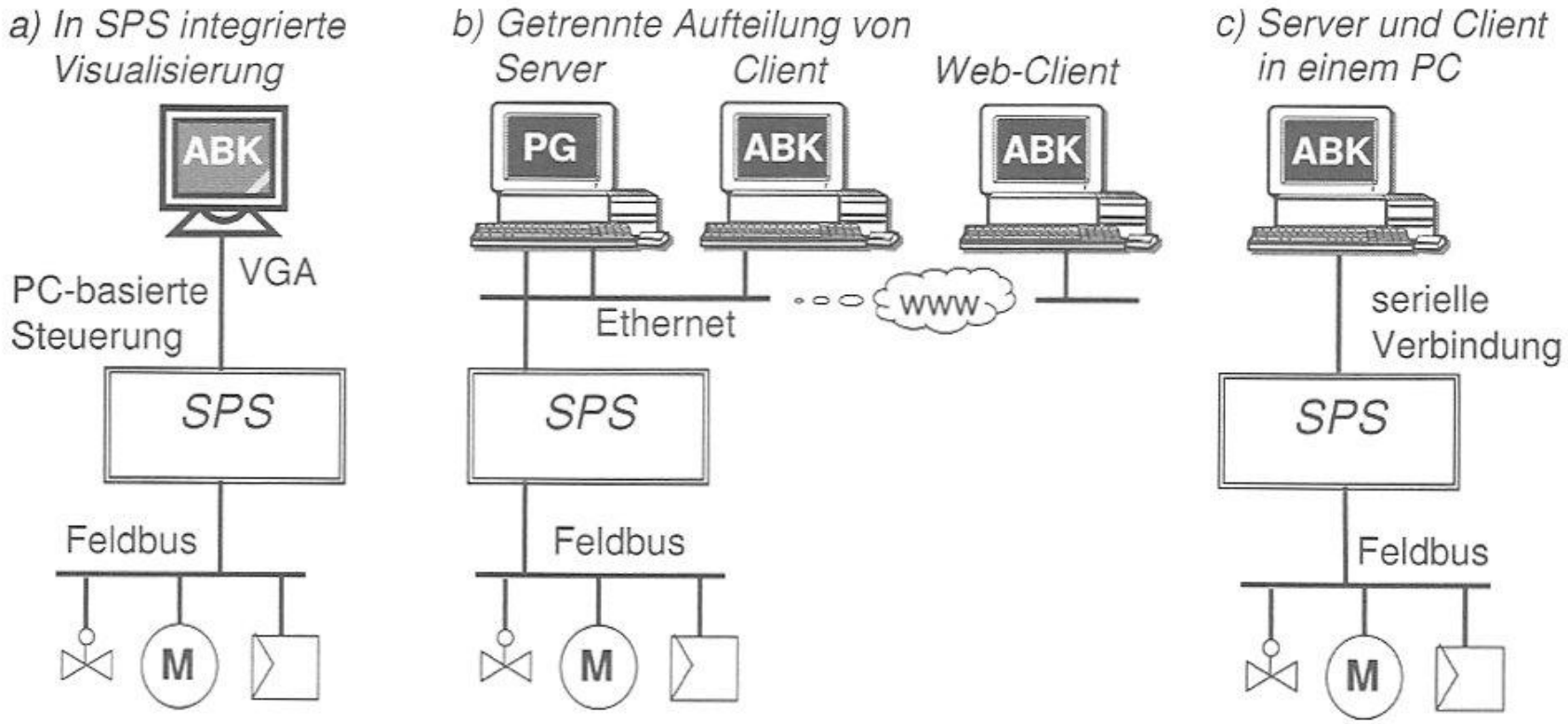
3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Der Datenaustausch zwischen diesen integrierten Visualisierungssystemen und der SPS erfolgt über denselben Mechanismus, der auch für das Programmiersystem verwendet wird.

Denn auch dort können die Zustände der Variablen in der SPS online beobachtet werden. Für die Prozessvisualisierung muss der Anwender lediglich, wie in Abb. 03.12 dargestellt, die Grafikelemente dynamisieren.

3 Aufbau und Strukturen industrieller Steuerungssysteme (87 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen



3 Aufbau und Strukturen industrieller Steuerungssysteme (88 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Für nichtintegrierte Visualisierungssysteme muss zusätzlich der Datenaustausch zur SPS konfiguriert werden. Dieser erfolgt über eine Client-Server-Verbindung. Der Server ist ein Programm. Das auf einem PC läuft, der mit der SPS verbunden ist. Er stellt dabei die Daten der SPS auf dem Netz zur Verfügung, ohne sich darum zu kümmern, wer diese Daten verwendet.

Ein beliebiger Client, also z.B. eine ABK, kann die Dienste des Servers in Anspruch nehmen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (89 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Um zu kennzeichnen, welche Signale zwischen Client und Server ausgetauscht werden sollen, sind mindestens folgende Festlegungen zu treffen (siehe Abb. 03.14):

- Name des Servers (SPS-Programm, z.B. STEP 7 oder CoDeSys),
- Pfad und Dateiname des Projekts, in dem die Anwenderprogramme laufen,
- Variablennamen der auszutauschenden Signale (z.B. Sensor- oder Stellsignale, interne Zustandsdaten der SPS).

3 Aufbau und Strukturen industrieller Steuerungssysteme (90 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Die gängigste Kopplung zwischen Prozessvisualisierung und SPS ist, dass der Server auf einem PC des Programmiersystems läuft und über Ethernet Daten mit einem anderen PC austauscht, auf dem die Prozessvisualisierung als Client läuft (siehe Abb. 03.13b).

3 Aufbau und Strukturen industrieller Steuerungssysteme (91 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Da die meisten Visualisierungssysteme auch web fähig sind, kann der Client auch über das Internet mit dem PC des Servers in Verbindung treten.

Dabei ist zusätzlich zu den oben genannten Festlegungen auch die IP-Adresse des Servers im Client zu spezifizieren.

Natürlich können Server und Client auch auf ein- und demselben PC laufen (siehe Abb. 03.13c).

3 Aufbau und Strukturen industrieller Steuerungssysteme (92 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Unter Windows gibt es verschiedene Server-Typen. Die bekanntesten und in der Automatisierungstechnik genutzten sind die DDE- (**D**ynamic **D**ata **E**xchange) und der OPC-Server (**O**bject Linking and Embedding for **P**rocess **C**ontrol). Diese Server nutzen eine Windows-interne Schnittstelle, über die z.B. Daten aus einer Excel-Tabelle automatisch nach Word geschrieben und ständig aktualisiert werden.

3 Aufbau und Strukturen industrieller Steuerungssysteme (93 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

DDE- und OPC-Server sprechen diese Schnittstelle mit bestimmten Windows-Befehlen an. Insofern sind DDE- und OPC-Server nichts anderes als spezielle Datenaustauschprogramme, die parallel zu den Anwenderprogrammen der SPS laufen.

3 Aufbau und Strukturen industrieller Steuerungssysteme (94 von 99)

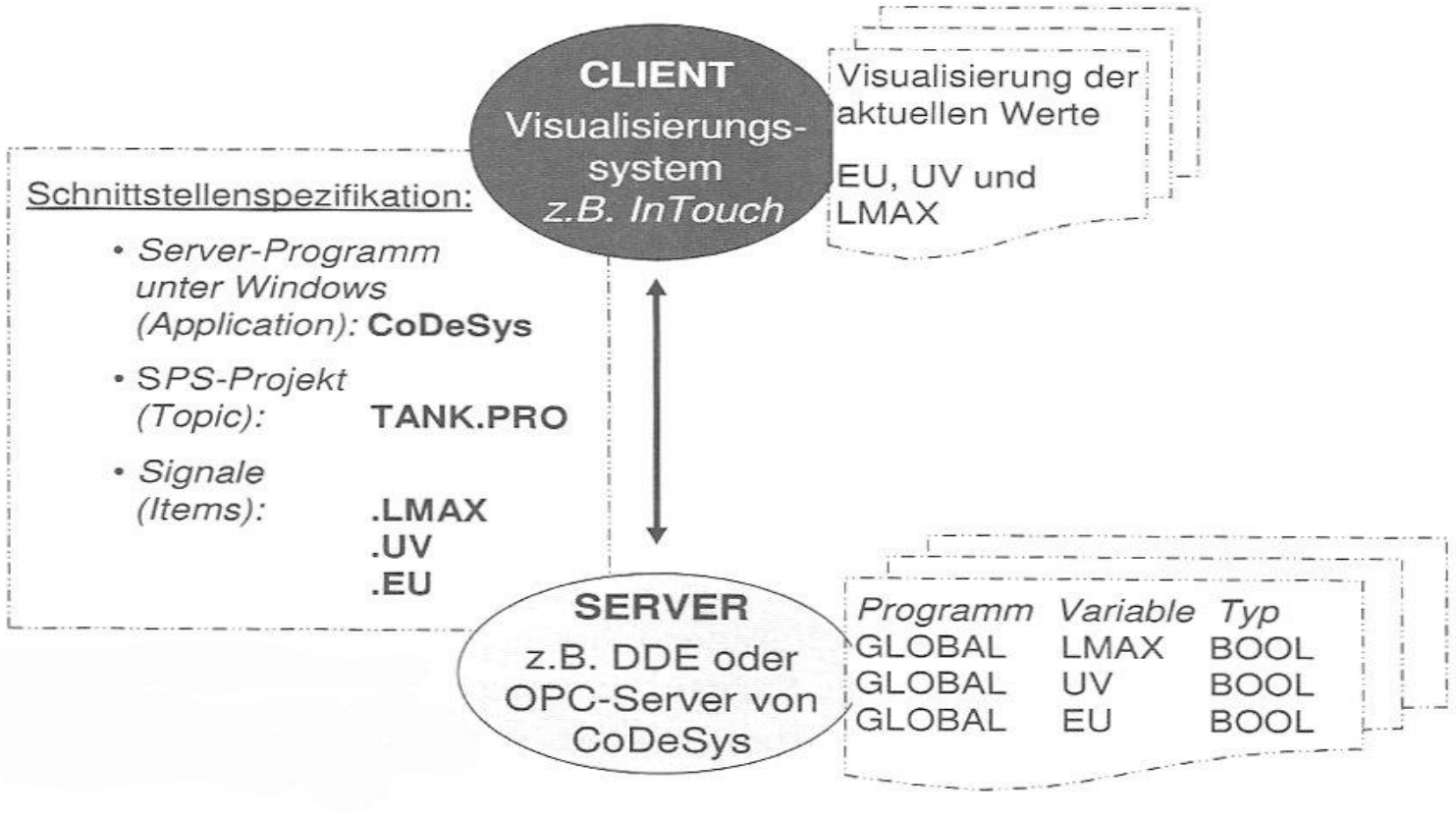
3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

Während der DDE-Server nur Anwendungen koppeln kann. Die wie in Abb. 03.13c auf demselben PC laufen, kann ein OPC-Server auch Daten für OPC-Clients auf anderen PCs bereitstellen (siehe Abb. 03.13b).

Deshalb hat sich OPC zu dem Standard-Kopplungsmechanismus für die vertikale Integration entwickelt. Trotzdem wird eine DDE-Kopplung noch häufig eingesetzt, weil sie einfach und kostengünstig ist.

3 Aufbau und Strukturen industrieller Steuerungssysteme (95 von 99)

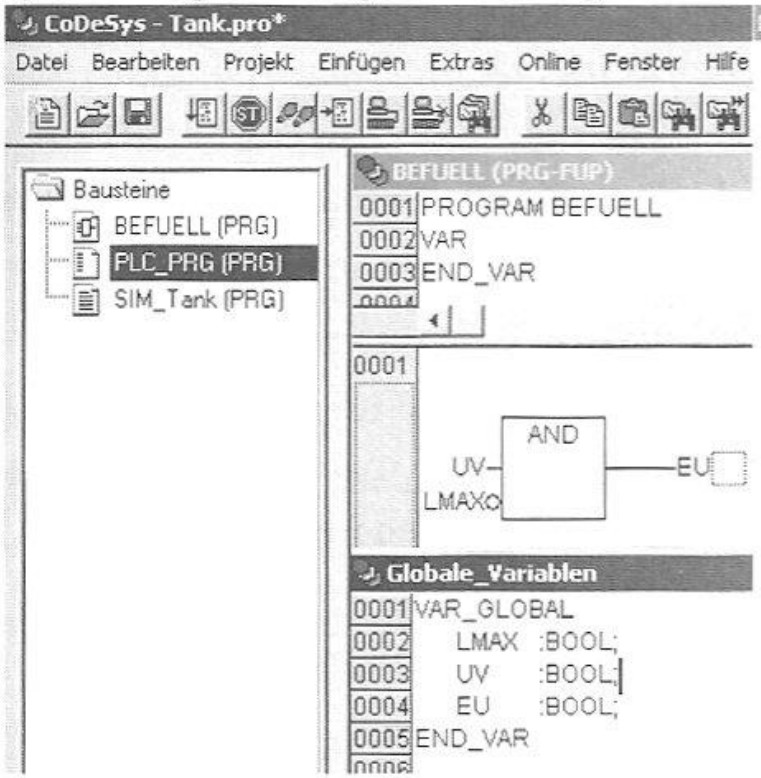
3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen



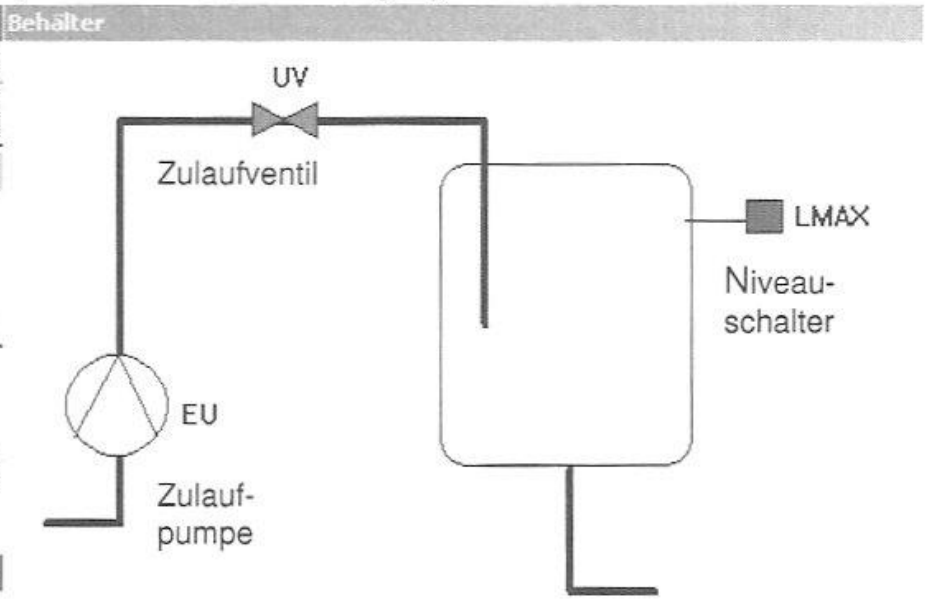
3 Aufbau und Strukturen industrieller Steuerungssysteme (96 von 99)

3.6.2 Kopplung zwischen Visualisierungssystemen und SPSen

a) SPS-Programmiersystem CoDeSys



b) Prozessvisualisierungssystem InTouch



3 Aufbau und Strukturen industrieller Steuerungssysteme (97 von 99)

3.6.3 Prozessleitsysteme

Der Begriff Leiten bezieht sich auf das Weiterleiten der Informationen von den Sensoren und Aktoren über die SPS zum Leitsystem und umgekehrt.

Nach DIN 19222 umfasst Leiten „die Gesamtheit aller Maßnahmen, ggf. auch unter Mitwirkung des Menschen, um anhand der aus dem Prozess oder der Umgebung erhaltenen Daten einen gewünschten Prozessablauf nach bestimmten, festgelegten Zielen zu bewirken“.

3 Aufbau und Strukturen industrieller Steuerungssysteme (98 von 99)

3.6.3 Prozessleitsysteme

Beispiele für solche Maßnahmen sind:

- Bedienung und Beobachtung des Prozesses und seiner Geräte,
- Meldung und Alarmierung bei Störungen,
- Protokollierung und Archivierung von Messwerten,
- Auswertung und Optimierung von Prozesszuständen.

Systeme, die alle diese Tätigkeiten ausführen, werden auch **S**upervisory-**C**ontrol-**a**nd-**D**ata-**A**nalysis-Systeme (SCADA) bezeichnet.

3 Aufbau und Strukturen industrieller Steuerungssysteme (99 von 99)

3.6.3 Prozessleitsysteme

Für reine Bedien- und Beobachtungssysteme gibt es branchenabhängig verschiedene Namen und Abkürzungen, wie Visualisierungssystem (Visualisierung) oder Anzeige- und Bedienkomponente (ABK) sowie Man Machine Interface (MMI) oder Human Machine Interface (HMI).

Immer häufiger realisieren auch diese Systeme weitere der oben aufgeführten Tätigkeiten und nähern sich damit den Prozessleitsystemen an.

- Ende Kap. 3 -

Vorlesungsinhalte

1. Integrierte Betriebsführung VL1
2. Vertikale und horizontale Integration
3. Aufbau und Strukturen industrieller Steuerungssysteme VL2
4. **SPS-Programmierung nach IEC 61134**
5. Übungen VL3

4 SPS-Programmierung nach IEC 61134 (01 von 89)

Zielsetzung der IEC 61131

- Trotz der stetig wachsenden Leistungsfähigkeit moderner SPSen sahen sich die Hersteller zu Beginn der 90er Jahre immer größeren Anforderungen gegenübergestellt.
- SPSen verfügten zwar über einen großen Funktionsumfang, doch die zu automatisierenden Anlagen wurden immer komplexer und der Druck zur Kostensenkung stieg.

4 SPS-Programmierung nach IEC 61134 (02 von 89)

Zielsetzung der IEC 61131

…

- Eine Folge dieser Entwicklung war, dass SPSen heute nicht mehr singulär betrachtet werden können, sondern wie im vorigen Kapitel dargestellt nur im Verbund mit Leitsystemen, Feldbussystemen und weiteren SPSen, mit denen die Automatisierungsaufgabe verteilt gelöst werden kann.

4 SPS-Programmierung nach IEC 61134 (03 von 89)

Zielsetzung der IEC 61131

…

Anforderungen an die Norm IEC 61131:

- *Standardisierung* der Vorgehensweise bei der Programmierung,
- *objektorientierte* Strukturierung der Programme,
- *herstellerunabhängige* Programmiersprachen, Einbindung v. Hochsprachen, wie z.B. PASCAL oder C,
- *offene*, standardisierte Schnittstellen,
- *Simulations*- und Testmöglichkeiten.

4 SPS-Programmierung nach IEC 61134 (04 von 89)

Zielsetzung der IEC 61131

Die Verwirklichung dieser Ziele ermöglichte es, Software für Planungs- und Wartungspersonal verständlicher und transparenter zu gestalten.

Außerdem ergaben sich erhebliche Synergieeffekte durch wieder verwendbare Softwaremodule und die Portabilität der Software zwischen Automatisierungssystemen unterschiedlicher Hersteller.

Auch verringerten sich der Lernaufwand und damit die Qualifizierungskosten zur Programmierung solcher Systeme.

4 SPS-Programmierung nach IEC 61134 (05 von 89)

Die Norm IEC 61131 gliedert sich in drei Teile:

1. Das ***Softwaremodell*** beschreibt, welche Softwaremodule es gibt und wie sie mit der SPS-Hardware verknüpft sind,
2. das ***Kommunikationsmodell*** beschreibt, wie der Datenaustausch zwischen den Softwaremodulen erfolgt,
3. das ***Programmiermodell*** beschreibt, wie Anwenderprogramme programmiert werden.

4 SPS-Programmierung nach IEC 61134 (06 von 89)

4.1 Das Softwaremodell

stellt die *Architektur* des Programmiersystems (siehe Abb. 04.01) dar.

Die Steuerungs-Konfiguration beschreibt die Hardware des Steuerungssystems. Diese umfasst i.d.R. verschiedene Ressourcen, wie z.B. CPUn, E/A-Baugruppen, Kommunikations-prozessoren.

Auf jeder CPU können verschiedene Tasks gleichzeitig ablaufen.

4 SPS-Programmierung nach IEC 61134 (07 von 89)

4.1 Das Softwaremodell

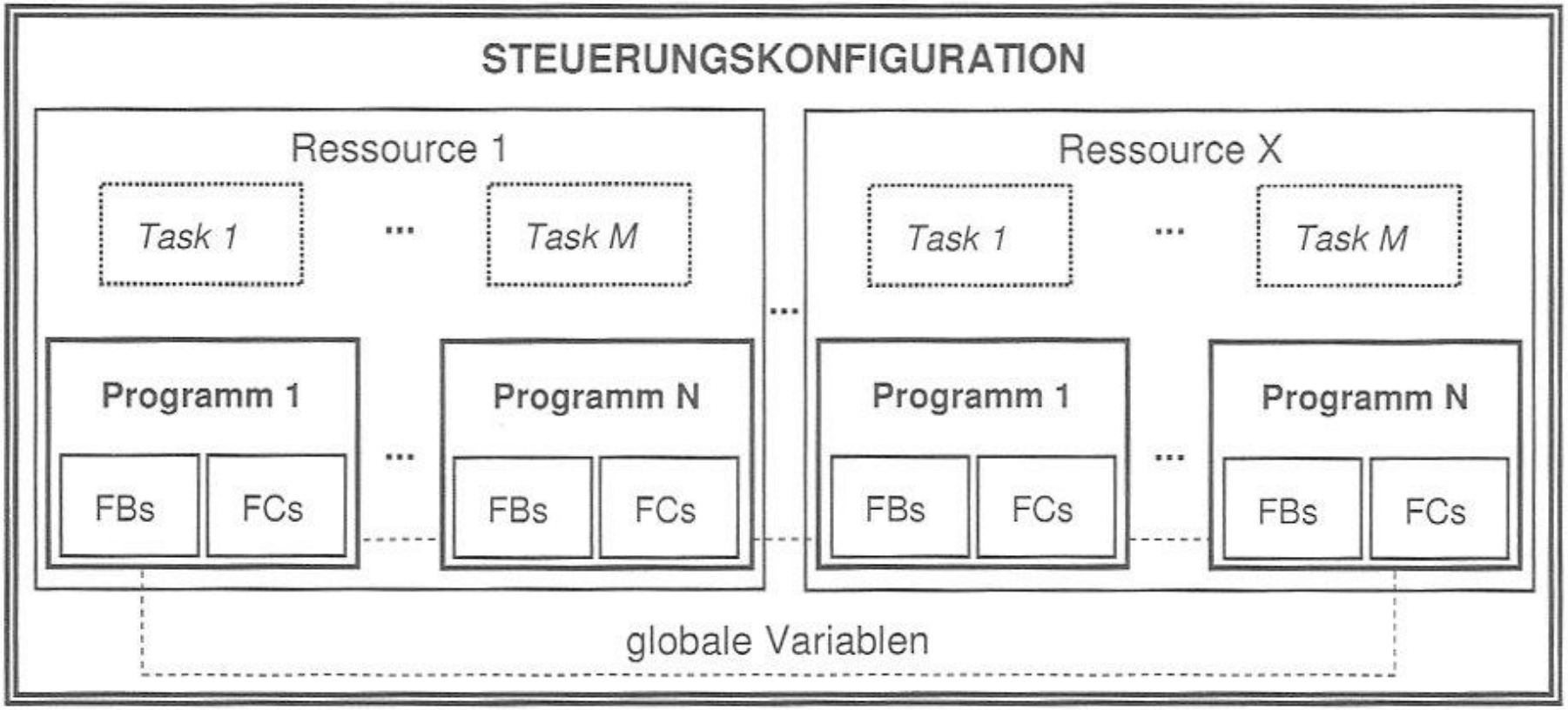


Abb. 04.01

4 SPS-Programmierung nach IEC 61134 (08 von 89)

4.1 Das Softwaremodell

Einer Task werden ein oder mehrere Programme zugeordnet, deren Ablauf durch die Task organisiert wird.

Damit laufen auf einer CPU mehrere Programme gleichzeitig. Jedes Programm kann aus mehreren Funktionsbausteinen (Function Blocks, FBs) und Funktionen (Function Codes, FCs) bestehen.

Globale Variablen ermöglichen den Datenaustausch zwischen den Programmen.

4 SPS-Programmierung nach IEC 61134 (09 von 89)

4.1.1 Steuerungskonfiguration und Ressourcen

In der Steuerungskonfiguration wird die bestehende *SPS-Hardwarestruktur* in der Programmierumgebung bekannt gemacht, damit beispielsweise Variablen mit den zugehörigen E/A-Kanälen der E/A-Baugruppen verbunden oder die Software auf verschiedene CPUn verteilt werden kann.

4 SPS-Programmierung nach IEC 61134 (10 von 89)

4.1.1 Steuerungskonfiguration und Ressourcen

Abb. 04.02 zeigt ein Beispiel für eine Steuerungskonfiguration, bestehend aus mehreren Ressourcen.

Unter *Ressourcen* versteht man die Baugruppen/Einsteckkarten des Steuerungssystems, auf die die Software zugreift. Im Fall von Hardware-SPSen bestehen die Ressourcen aus CPUn, E/A-Baugruppen, Kommunikationsbaugruppen und weiteren z.B. Spezial- bzw. Sonderbaugruppen (schnelle Zähler, Motion Contol u.a.).

4 SPS-Programmierung nach IEC 61134 (11 von 89)

4.1.1 Steuerungskonfiguration und Ressourcen

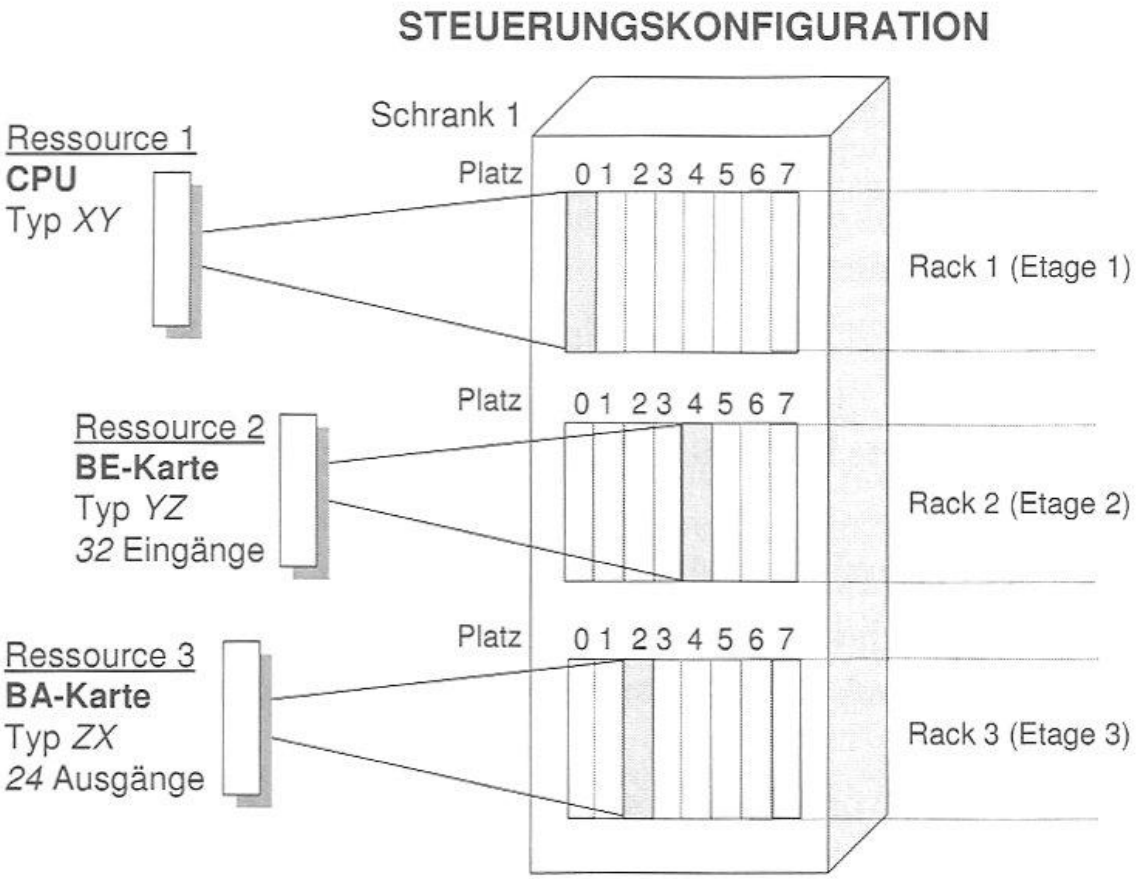


Abb. 04.02

4 SPS-Programmierung nach IEC 61134 (12 von 89)

4.1.1 Steuerungskonfiguration und Ressourcen

In den meisten Programmiersystemen erfolgt die Steuerungskonfiguration menügesteuert, d.h. man kann die gewünschten Baugruppen auswählen und dann per Drag and Drop an die entsprechende Stelle in der Konfiguration ziehen.

4 SPS-Programmierung nach IEC 61134 (13 von 89)

4.1.1 Steuerungskonfiguration und Ressourcen

Die Adressierung der E/A-Kanäle von E/A-Baugruppen erfolgt im Rahmen der Variablen-deklaration mit dem Schlüsselwort AT.

Wie Tabelle 04.01 zeigt, werden Eingangskanäle mit einem „%I“ für Input und Ausgangskanäle mit einem „%Q“ für Output gekennzeichnet.

Der danach folgende Bauchstabe beschreibt, ob es sich bei dem E/A-Signal um ein Bit, Byte, Word oder Doubleword handelt.

4 SPS-Programmierung nach IEC 61134 (14 von 89)

4.1.1 Steuerungskonfiguration und Ressourcen

Demnach wird beispielsweise der 2. Kanal einer binären Eingangskarte am Steckplatz 1 durch die Bezeichnung „%IX1.2“ angesprochen.

Kennzeichnung der E/A-Adressierung	1. Folgebuchstabe	2. Folgebuchstabe	Beispiel
AT %	I Eingang Q Ausgang	X Bit B Byte (8 Bit) W Word (16 Bit) D Doubleword (32 Bit)	AT %IX1.2 AT %IB0 AT %QW7 AT %QD5

Tabelle 04.01

4 SPS-Programmierung nach IEC 61134 (15 von 89)

4.1.1 Steuerungskonfiguration und Ressourcen

Das folgende Beispiel erläutert die Vorgehensweise zur Festlegung der Steuerungskonfiguration für eine Soft-SPS mit dem SPS-Programmiersystem CoDeSys der Fa. 3S.

(➔ Demonstration „Progamm-Bsp. 04.01“)
http://htw-berlin.applicad-atit.de/gf8wiw

4 SPS-Programmierung nach IEC 61134 (16 von 89)

4.1.2 Tasks

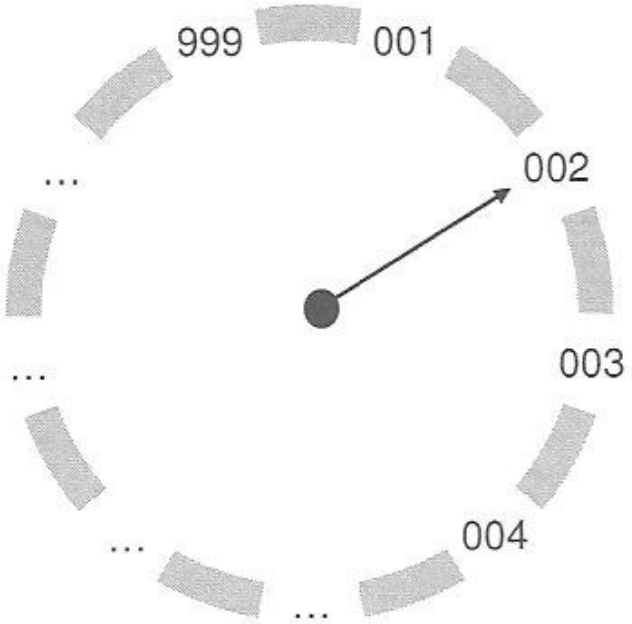
In einer CPU können eine oder mehrere Tasks ablaufen.

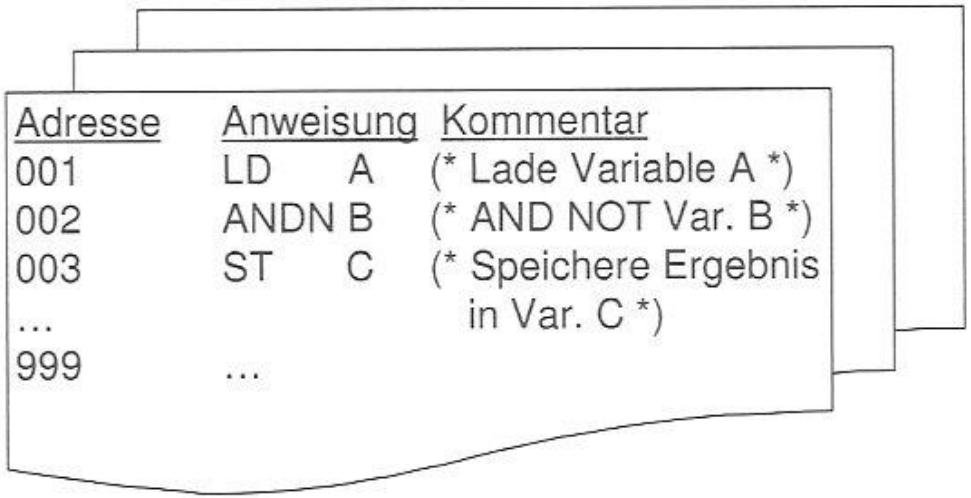
Eine Task organisiert den zeitlichen Ablauf der Programme.

Die Programme innerhalb einer Task werden in der Reihenfolge ihres Aufrufes nacheinander abgearbeitet (siehe Abb. 04.05).

4 SPS-Programmierung nach IEC 61134

4.1.2 Tasks





Adresse	Anweisung		Kommentar
001	LD	A	(* Lade Variable A *)
002	ANDN	B	(* AND NOT Var. B *)
003	ST	C	(* Speichere Ergebnis in Var. C *)
...			
999	...		

Abb. 04.05

4 SPS-Programmierung nach IEC 61134 (18 von 89)

4.1.2 Tasks

Einer Task wird eine Zykluszeit zugeordnet. Alle Programme einer Task laufen also innerhalb der gleichen Zykluszeit ab.

Bei der Standardtask, wie *Main* oder *PLC_PRG* bei CoDeSys kann es sein, dass die Zykluszeit nicht eingestellt werden muss, sondern sich aus der Programmverarbeitungszeit der Task plus einen gewissen zeitlichen Overhead ergibt.

4 SPS-Programmierung nach IEC 61134 (19 von 89)

4.1.2 Tasks

Innerhalb der Task werden die Daten streng nach dem **EVA-Prinzip** verarbeitet:

- Zuerst werden die Eingangsdaten aller Programme eingelesen,
- dann werden die Programme nacheinander abgearbeitet und
- schließlich werden die Ausgangsdaten aller Programme ausgegeben.

4 SPS-Programmierung nach IEC 61134 (20 von 89)

4.1.2 Tasks

Die Zykluszeit einer Task wird vom Prozess aus, also von außerhalb der Steuerung gesehen.

Sie wird gemessen zwischen Einlesen und Ausgeben der Prozessdaten und beinhaltet damit auch die Reaktionszeit der Steuerung.

4 SPS-Programmierung nach IEC 61134 (21 von 89)

4.1.2 Tasks

Die Zykluszeit einer Task wird vom Prozess aus, also von außerhalb der Steuerung gesehen.

Sie wird gemessen zwischen Einlesen und Ausgeben der Prozessdaten und beinhaltet damit auch die Reaktionszeit der Steuerung.

Da diese Zeit endlich ist (z.B. 150 ms), ergibt sich daraus auch, dass ein Prozess, der von einer Steuerung gesteuert werden soll, die Zykluszeit praktisch nicht wahrnehmen soll.

4 SPS-Programmierung nach IEC 61134 (22 von 89)

4.1.2 Tasks

Die Zykluszeit einer Steuerung muss also an die zeitlichen Anforderungen eines zu steuernden Prozesses angepasst sein.

Relativ langsame, träge Prozesse z.B. in der Verfahrenstechnik benötigen Reaktionszeit von mehreren hundert Millisekunden bis zu einigen Sekunden.

Hierfür sind Zykluszeiten von z.B. 150 … 300 ms in den meisten Fällen völlig ausreichend.

4 SPS-Programmierung nach IEC 61134 (23 von 89)

4.1.2 Tasks

Anlagen der Fertigungstechnik mit Montage- und Antriebssystemen sind zeitlich wesentlich anspruchsvoller und erwarten Reaktionen im Bereich von Millisekunden bis einigen zehn Millisekunden.

Hier müssen also Steuerungen eingesetzt werden, die sehr kleine Zykluszeiten realisieren können.

4 SPS-Programmierung nach IEC 61134 (24 von 89)

4.1.2 Tasks

Multitasking

In einer Ressource bzw. CPU können auch *mehrere* Task mit unterschiedlichen Zykluszeiten praktisch parallel ablaufen.

Wir sprechen dann von *Multitasking*.

Damit ist es möglich, die Rechenleistung auf die verschiedenen Programme zu verteilen.

Voraussetzung dabei ist jedoch, dass die CPU über eine ausreichende Rechenleistung verfügt.

4 SPS-Programmierung nach IEC 61134 (25 von 89)

4.1.2 Tasks

Multitasking

Den Ablauf in einer CPU mit einer Task zeigt Abb. 04.06a und mit mehreren Tasks zeigt Abb. 04.06b.

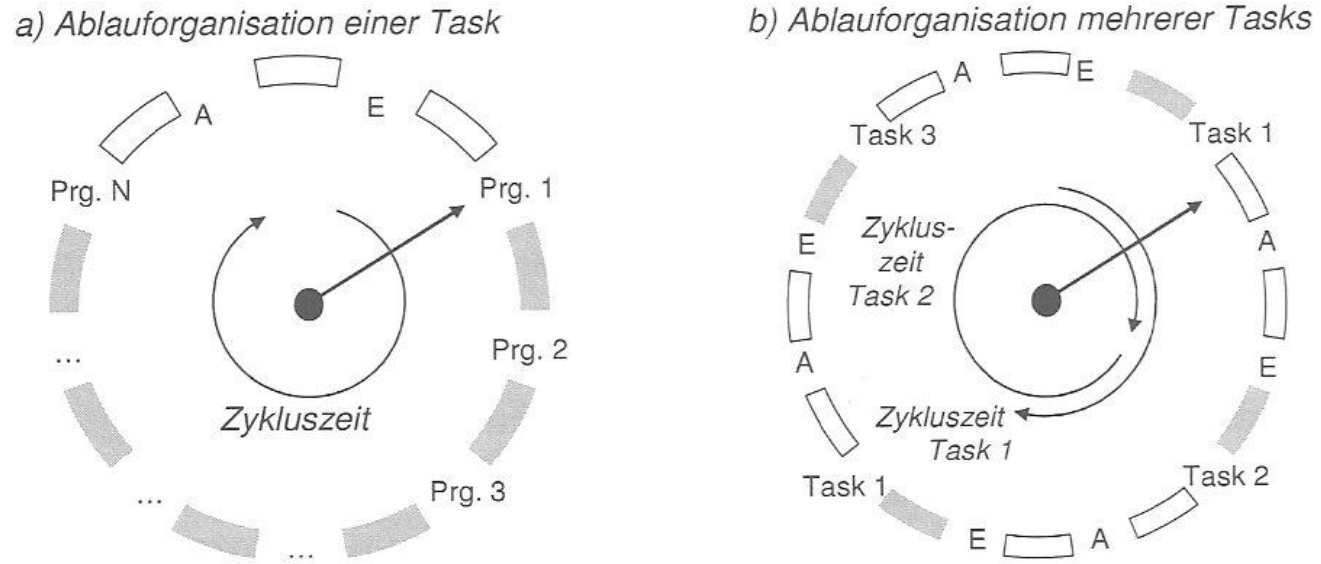


Abb. 04.06

4 SPS-Programmierung nach IEC 61134 (26 von 89)

4.1.2 Tasks

Das Multitasking wird durch eine Zeitgebereinheit in der CPU realisiert, die mit einem Interrupt-Signal die eine laufende Task unterbricht, wenn eine andere Task abgearbeitet werden soll.

Abhängig vom Rechenaufwand einer Task wird ihr ein bestimmter Prozentteil der verfügbaren Rechenzeit der CPU zugeteilt.

Bei Vorhandensein nur einer CPU nennt die Fähigkeit, mehrere Tasks quasi parallel ablaufen zu lassen auch *Pseudo-Multitasking*.

4 SPS-Programmierung nach IEC 61134 (27 von 89)

4.1.2 Tasks

Ein wirkliches paralleles Ablaufen von verschiedenen Tasks erreicht man genau genommen nur bei Einsatz von je einer CPU pro Task.

In der Praxis ist das jedoch fast nie der Fall.

Es kommen zwar bei SPSen auch mehrere CPUn zum Einsatz, doch laufen auch bei diesen SPSen i.d.R. mehrere Tasks auf einer CPU.

4 SPS-Programmierung nach IEC 61134 (28 von 89)

4.1.2 Tasks

Das folgende Beispiel geht auf die Arbeit mit Task für eine Soft-SPS mit dem SPS-Programmiersystem CoDeSys der Fa. 3S ein.

(➔ Demonstration „Progamm-Bsp. 04.02“)
http://htw-berlin.applicad-atit.de/gf8wiw

4 SPS-Programmierung nach IEC 61134 (29 von 89)

4.1.3 Programmorganisationseinheiten

Ein *Anwenderprogramm*, wie SPS-Programme üblicherweise genannte werden, das zur Automatisierung einer Anlage dient, besteht aus ***P****rogramm****o****rganisations****e****inheiten* **POEe**n, im englischen sind das **P**rogram **O**rganisation **U**nits **POU**s.

4 SPS-Programmierung nach IEC 61134 (30 von 89)

4.1.3 Programmorganisationseinheiten

Programmorganisationseinheiten POEen können sein:

- Programme,
- Funktionen (siehe Abb. 04.08a) und
- Funktionsbausteine (siehe Abb. 04.08b).

4 SPS-Programmierung nach IEC 61134 (31 von 89)

4.1.3 Programmorganisationseinheiten

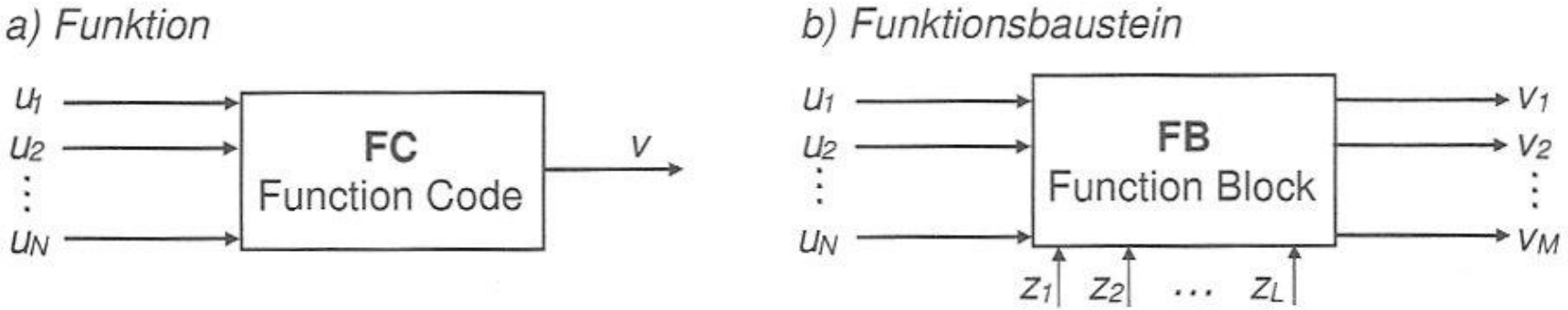


Abb. 04.08

Jede dieser POEen besteht aus einem

Deklarationsteil, in dem die Variablen deklariert werden, und einem Anweisungsteil, der den Programmcode enthält.

4 SPS-Programmierung nach IEC 61134 (32 von 89)

4.1.3 Programmorganisationseinheiten

In Abb. 04.09a wird das veranschaulicht.

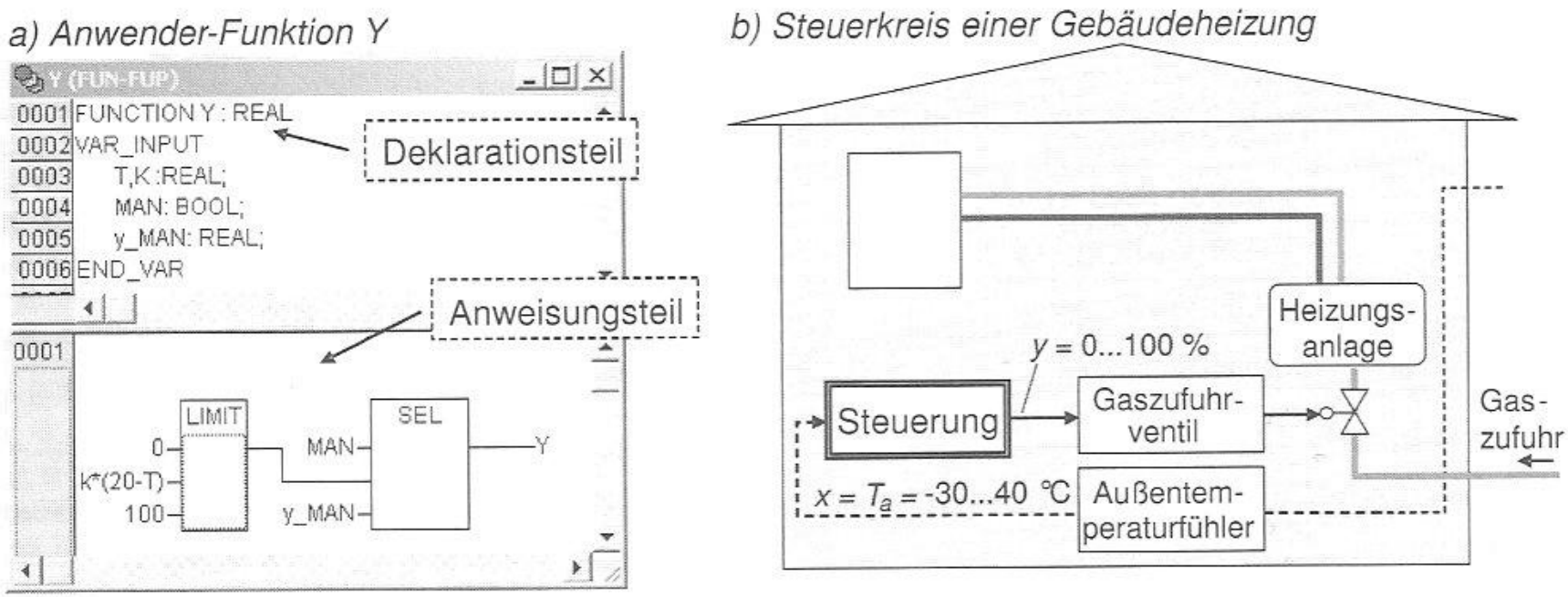


Abb. 04.09

4 SPS-Programmierung nach IEC 61134 (33 von 89)

4.1.3 Programmorganisationseinheiten

4.1.3.1 Funktionen

Funktionen (*Function Codes, FCs*) können mehrere Eingangsvariablen ***u***, aber nur eine Ausgangsvariable **v** haben.

Die Ausgangsvariable ist dabei der Rückgabewert der Funktion. Rekursive Aufrufe einer Funktion sind nach IEC 61131 nicht möglich.

Eine Funktion hat im Gegensatz zu einem Funktionsbaustein kein Gedächtnis.

Das bedeutet, dass es in Funktionen nicht erlaubt ist, Zwischenwerte zu speichern.

4 SPS-Programmierung nach IEC 61134 (34 von 89)

4.1.3 Programmorganisationseinheiten

4.1.3.1 Funktionen

Man unterscheidet Standard- und Anwenderfunktionen.

Die meisten SPS-Programmiersysteme stellen fertige Standard-Funktionen nach IEC 61131 zur Verfügung.

In Tabelle 04.02 sehen sie eine Übersicht.

Diese Standard-Funktionen befinden sich in einer Programmbibliothek und müssen vom Anwender nicht mehr programmiert werden, sondern kann er direkt in seinen Programmen verwenden.

4 SPS-Programmierung nach IEC 61134 (35 von 89)

4.1.3 Programmorganisationseinheiten

4.1.3.1 Funktionen

Man unterscheidet Standard- und Anwenderfunktionen.

Die meisten SPS-Programmiersysteme stellen fertige Standard-Funktionen nach IEC 61131 zur Verfügung.

In Tabelle 04.02 sehen sie eine Übersicht.

Diese Standard-Funktionen befinden sich in einer Programmbibliothek und müssen vom Anwender nicht mehr programmiert werden, sondern kann er direkt in seinen Programmen verwenden.

4 SPS-Programmierung nach IEC 61134

Darüber hinaus hat der Anwender die Möglichkeit, eigene Anwender-Funktionen zu schreiben.

Boole'sche Funktionen:		**Bitfolge-Funktionen:**	
AND	UND-Verbindung	ROL	Links rotieren
NOT	Negation (NICHT)	ROR	Rechts rotieren
OR	ODER-Verbindung	SHL	Links schieben
XOR	EXOR-Verbindung	SHR	Rechts schieben
Arithmetische Funktionen:		**Numerische Funktionen:**	
ADD	Addierer	ABS	Absolutwert
DIV	Divisor	ACOS	Arccos
EXPT	Potenzierung	ASIN	Arcsin
MOD	Modulo-Division	ATAN	Arctan
MUL	Multiplizierer	COS	Cosinus
NEG	Negierer	EXP	Exponent
SUB	Subtrahierer	LN	Natürlicher Logarithmus
Funktionen zur Typumwandlung:		LOG	Logarithmus zur Basis 10
BYTE_TO_WORD		SIN	Sinus
BYTE_TO_INT		SQRT	Quadratwurzel
BYTE_TO_REAL		TAN	Tangens
UINT_TO_INT		**Funktionen für Auswahl:**	
INT_TO_UINT		LIMIT	Begrenzung
DINT_TO_INT		MAX	Maximum
INT_TO_DINT		MIN	Minimum
INT_TO_BYTE		SEL	Binäre Auswahl
INT_TO_WORD		**Funktionen für Vergleich:**	
INT_TO_REAL		EQ	gleich (equal)
REAL_TO_BYTE		LT	kleiner als (less than)
REAL_TO_WORD		GT	größer als (greater than)
REAL_TO_INT		LE	kleiner gleich (less or equal)
TIME_TO_DINT		GE	größer gleich (greater or eq.)
DINT_TO_TIME		NE	ungleich (not equal)

Tabelle 04.02

4 SPS-Programmierung nach IEC 61134 (37 von 89)

4.1.3 Programmorganisationseinheiten

4.1.3.2 Funktionsbausteine

In Funktionsbausteinen (*Function Blocks, FBs*) können auch interne Zwischenwerte, auch Zustände genannt, gespeichert werden.

Im Unterschied zu einer Funktion kann ein Funktionsbaustein mehrere Ausgänge bzw. Ausgangsvariablen ***v*** haben.

Die werden durch Verarbeitung der Eingänge bzw. Eingangsvariablen ***x*** und der internen Zustände ***z*** erzeugt (siehe Abb. 04.08b).

4 SPS-Programmierung nach IEC 61134 (38 von 89)

4.1.3 Programmorganisationseinheiten

4.1.3.2 Funktionsbausteine

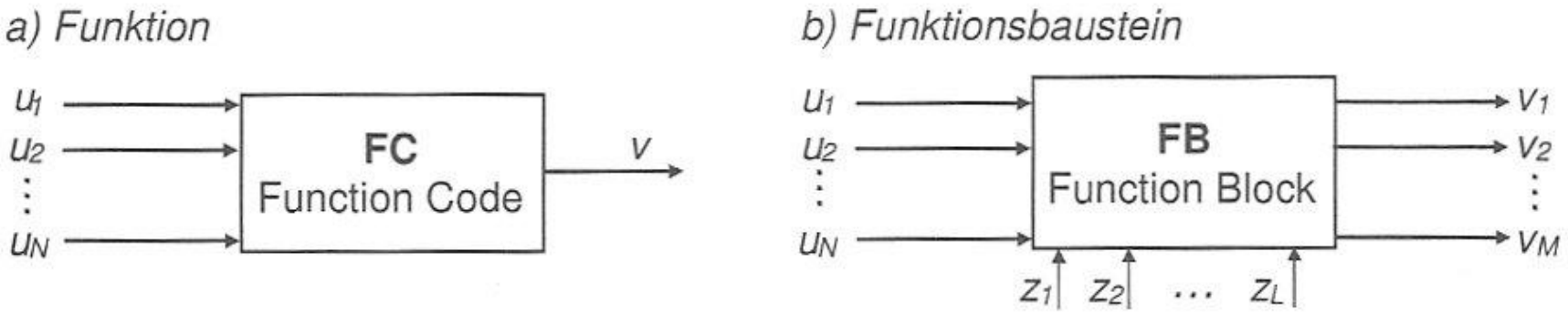


Abb. 04.08

4 SPS-Programmierung nach IEC 61134 (39 von 89)

4.1.3 Programmorganisationseinheiten

4.1.3.2 Funktionsbausteine

In Tabelle 04.03 sind die einige wichtige IEC-Standard-Funktionsbausteine aufgeführt.

Wie diese Bausteine für verschiedene automatisierungstechnische Aufgabenstellungen verwendet werden und wie sie aufgebaut sind, findet man in den Handbüchern zu den SPS-Programmiersystemen, so auch im Handbuch von CoDeSys.

4 SPS-Programmierung nach IEC 61131 (40 von 89)

4.1.3.2 Funktionsbausteine

Speicher *(s. Abschnitt 4.2.1)*	SR	Flip-Flop mit vorrangig Setzen
	RS	Flip-Flop mit vorrangig Rücksetzen
Trigger	F_TRIG	Erkennung einer fallenden Flanke
	R_TRIG	Erkennung einer steigenden Flanke
Zähler *(s. Abschnitt 4.2.2)*	CTU	Aufwärts-Zähler
	CTD	Abwärts-Zähler
	CTUD	Auf- und Abwärts-Zähler
Timer *(s. Abschnitt 4.2.3)*	TP	Pulstimer (Mono-Flop)
	TON	Timer mit Einschaltverzögerung
	TOF	Timer mit Ausschaltverzögerung

Tabelle 04.03

4 SPS-Programmierung nach IEC 61134 (41 von 89)

4.1.4 Variablen

Das letzte Element des Softwaremodells sind die *Variablen*, mit denen die Kommunikation zwischen den Programmorganisationseinheiten POEen realisiert wird.

Man unterscheidet prinzipiell folgende Variablenarten, gekennzeichnet durch die Schlüsselwörter:

- **VAR**:
  Lokale Variablen sind nur innerhalb der Programmorganisationseinheit POE gültig, in der sie deklariert wurden.

4 SPS-Programmierung nach IEC 61134 (42 von 89)

4.1.4 Variablen

Man unterscheidet prinzipiell folgende Variablenarten, gekennzeichnet durch die Schlüsselwörter:

- **VAR_GLOBAL**:
  Globale Variablen gelten in allen Programmen, Funktionen und Funktionsbausteinen.

- **VAR_INPUT**:
  Durch *Eingangsvariablen* werden Werte in Funktionen oder Funktionsbausteinen hineingeschrieben.

4 SPS-Programmierung nach IEC 61134 (43 von 89)

4.1.4 Variablen

Man unterscheidet prinzipiell folgende Variablenarten, gekennzeichnet durch die Schlüsselwörter:

- **VAR_OUTPUT**:
  Durch *Ausgangsvariablen* werden Werte von Funktionsbausteinen ausgegeben .

- **VAR_IN_OUT**:
  Im Gegensatz zu reinen Eingangsvariablen können Ein- und Ausgangsvariablen innerhalb des Funktionsbausteins verändert und dann ausgegeben werden .

4 SPS-Programmierung nach IEC 61134 (44 von 89)

4.1.4 Variablen

Man unterscheidet prinzipiell folgende Variablenarten, gekennzeichnet durch die Schlüsselwörter:

- **VAR_RETAIN**:
  Diese Variablen behalten ihren Wert, wenn die SPS aus- und wieder eingeschaltet wird.

- **VAR_PERSISTENT**:
  Persistente Variablen behalten ihren Wert, wenn die Software erneut in die SPS geladen wird.

4 SPS-Programmierung nach IEC 61134 (45 von 89)

4.1.4 Variablen

Die beiden letzten Variablentypen werden auch als ***permanente*** *Variablen* bezeichnet und können miteinander kombiniert werden. Sie lassen sich lokal oder auch global deklarieren.

Adressiert eine Variable einen E/A-Kanal, z.B. durch „*AT %IX0.0*“, spricht man auch von einer direkten Variablen.

4 SPS-Programmierung nach IEC 61134 (46 von 89)

4.1.4 Variablen

Eine Variablendeklaration besteht aus folgenden 5 Bestandteilen (optionale Bestandteile werden *kursiv* dargestellt):

<Variablenname> ***AT<Adresse>*** **:<Datentyp>** **:=<*Initialwert*>;** ***(* Kommentar *)***

z.B.

```
    LMAX        AT %IX0.0      :BOOL        := FALSE;        (* Niveau *)
```

oder

```
    LMIN        AT %IX0.1      :BOOL
```

4 SPS-Programmierung nach IEC 61134 (47 von 89)

4.1.4 Variablen – Wahl der Variablennamen

Über die Wahl der *Variablennamen* lohnt es sich ein wenig nachzudenken.

Der Variablenname sollte in großem Maße selbsterklärend (selbstdokumentierend) sein, um die Lesbarkeit eines Programms zu erleichtern.

Weit verbreitet ist die so genannte *ungarische Notation*, in der Datentypen im Variablennamen mit einkodiert werden, um in großen Programmen den Programmierer zu unterstützen.

Oft, z.B. in der Prozessautomatisierung, ist das Ziel aber ein anderes.

4 SPS-Programmierung nach IEC 61134 (48 von 89)

4.1.4 Variablen – Wahl der Variablennamen

Es geht in erster Linie darum, die Software für den Anwender verständlich zu gestalten, damit er leicht Änderungen und Ergänzungen vornehmen kann.

Aus diesem Grund sollten Variablen stets so bezeichnet werden, dass ihr Zweck durch den Namen verdeutlicht wird. Dann lässt sich ein Programm oft auch ohne bzw. mit kurzen Kommentaren verstehen.

Jede Variable ist durch einen Datentyp charakterisiert.

Die *Standard-Datentypen* nach IEC 61131, wie BOOL, INT, REAL und TIME, sind in Tabelle 04.04 aufgeführt.

4 SPS-Programmierung nach IEC 61131

4.1.4 Variablen – Standard-Datentypen

	Datentyp	Größe	Wertebereich	Beispiel
Bitfolgen	BOOL	1 Bit	FALSE/TRUE	FALSE
	BYTE	8 Bit	16#00...16#FF *(Hex-Darstellung)*	16#00
	WORD	16 Bit	16#0000...16#FFFF	16#0000
	DWORD	32 Bit	16#00000000...16#FFFFFFFF	16#00000000
Ganze Zahlen	SINT	8 Bit	–128...127	0
	INT	16 Bit	–32768...32767	0
	DINT	32 Bit	2147483648... 2147483647	0
Ganze Zahlen ohne Vorzeichen	USINT	8 Bit	0...255	0
	UINT	16 Bit	0...65535	0
	UDINT	32 Bit	0...4294967295	0
Fließkommazahlen	REAL	32 Bit	$-3{,}4 \cdot 10^{38}$...$3{,}4 \cdot 10^{38}$ (Auflösung $1{,}18 \cdot 10^{38}$)	0.0
Zeit	TIME			t#2h:1m:0s:0ms
Uhrzeit	TIME_OF_DAY			tod#15:23:17.456
Datum	DATE			d#2003-18-01
Zeichenfolge	STRING			STRING(4)='Text'

Tabelle 04.04

4 SPS-Programmierung nach IEC 61134 (50 von 89)

4.2 Das Kommunikationsmodell

Das Kommunikationsmodell der IEC 61131 beschreibt, wie der Datenaustausch *innerhalb* eines Programms und *zwischen* Programmen erfolgt.

4 SPS-Programmierung nach IEC 61134 (51 von 89)

4.2.1 Datenaustausch innerhalb eines Programms

Die Kommunikation zwischen den Programmen, die in den Programmorganisationseinheiten POEen formuliert wurden erfolgt i.d.R. über globale Variablen (VAR_GLOBAL).

Das gilt im Übrigen auch bei der Kommunikation zwischen mehreren SPSen bzw. SPSen und Visualisierung etc.

4 SPS-Programmierung nach IEC 61134 (52 von 89)

4.2.1 Datenaustausch innerhalb eines Programms

Innerhalb eines Programms, also einer POE, erfolgt die Kommunikation zwischen den Funktionen FCs untereinander, den Funktionsbausteinen FBs untereinander sowie zwischen Funktionen und Funktionsbausteinen über die Ein- und Ausgangsvariablen (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT) der Bausteine.

4 SPS-Programmierung nach IEC 61134 (53 von 89)

4.2.1 Datenaustausch innerhalb eines Programms

Einer Funktion müssen dabei genau genommen nur die Eingangsvariablen (VAR_INPUT) übergeben werden, da der Rückgabewert der Funktion ja direkt im aufrufenden Funktionsbaustein oder Programm verarbeitet wird.

Eine Verknüpfung über globale Variablen (VAR_GLOBAL) ist natürlich immer möglich, zeugt jedoch an dieser Stelle von einem schlechten Programmierstil.

4 SPS-Programmierung nach IEC 61134 (54 von 89)

4.2.1 Datenaustausch innerhalb eines Programms

Eine Besonderheit beim Aufruf eines Funktionsbausteins soll an dieser Stelle nicht unerwähnt bleiben.

Wird ein Funktionsbaustein in einem Programm aufgerufen, so wird genau genommen eine Instanz dieses Funktionsbausteins, also eine Art Kopie, aufgerufen.

Die Instanz ist als Variable vom Typ des Funktionsbausteins zu deklarieren.

Die deklarierten Variablen eines Funktionsbausteins gelten für die aufgerufene Instanz und sind über die gesamte Laufzeit gültig.

4 SPS-Programmierung nach IEC 61134 (55 von 89)

4.2.1 Datenaustausch innerhalb eines Programms

Wird der gleiche Funktionsbaustein an verschiedenen Stellen eines SPS-Programms aufgerufen, so werden mehrere unabhängige Instanzen dieses Bausteins ausgeführt, die alle unabhängig voneinander ablaufen und sich in keiner Weise gegenseitig beeinflussen.

4 SPS-Programmierung nach IEC 61134 (56 von 89)

4.2.2 Datenaustausch zwischen Programmen

Die Kommunikation zwischen verschiedenen Programmen kann nach IEC 61131 nur mittels *globaler Variablen* (VAR_GLOBAL) realisiert werden.

Das unterstützt nicht gerade einen guten Programmierstil.

Globale Variablen schränken die Übersichtlichkeit ein und führen zu unerwünschten Effekten, wenn Variablen mehrfach, d.h. lokal und global, deklariert werden.

4 SPS-Programmierung nach IEC 61134 (57 von 89)

4.2.2 Datenaustausch zwischen Programmen

Jeder Programmierer sollte sich bemühen, die Anzahl der globalen Variablen innerhalb eines Projekts so gering wie möglich zu halten, also auf das notwendige Minimum zu beschränken.

4 SPS-Programmierung nach IEC 61134 (58 von 89)

4.2.2 Datenaustausch zwischen Programmen

Werden in einem großen Projekt sehr viele Variablen zwischen einzelnen Programmen (POEen) einer SPS ausgetauscht oder enthält das Projekt mehrere SPSen zwischen denen sehr viele Variablen übertragen werden müssen, so lässt sich eine gewisse Unübersichtlichkeit bei den globalen Variablen nicht vermeiden.

Umso wichtiger ist hierbei eine überlegte und systematische Planung der Variablennamen.

4 SPS-Programmierung nach IEC 61134 (59 von 89)

4.2.2 Datenaustausch zwischen Programmen

Signalmodell einer SPS

Prinzipiell empfiehlt es sich, zwischen folgenden *Signaltypen* von Programmen zu unterscheiden:

- Der programmübergreifende Datenaustausch erfolgt durch die *Steuersignale* ***u*** und die *Statussignale* ***v***, die jeweils global zu deklarieren sind.
- Dagegen werden die *Messsignale* $\boldsymbol{x}_e$ von den Sensoren über die SPS-Eingangskanäle eingelesen und die *Stellsignale* $\boldsymbol{y}_a$ von den Programmen über die SPS-Ausgangskanäle an die Aktoren übertragen.

4 SPS-Programmierung nach IEC 61134 (60 von 89)

4.2.2 Datenaustausch zwischen Programmen

Die Statussignale ***v*** eines Programms (z.B. Variable Y des Programms P2 in Abb. 04.11) können als Steuersignale ***u*** eines anderen Programms (z.B. in P1) verwendet werden und verknüpfen somit die beiden Programme.

Abb. 04.12 zeigt, dass jedes Programm über seine Steuer- und Statussignale mit anderen Programmen und durch Mess- und Stellsignale mit den Sensoren und Aktoren kommunizieren kann.

4 SPS-Programmierung nach IEC 61131

4.2.2 Datenaustausch zwischen Programmen

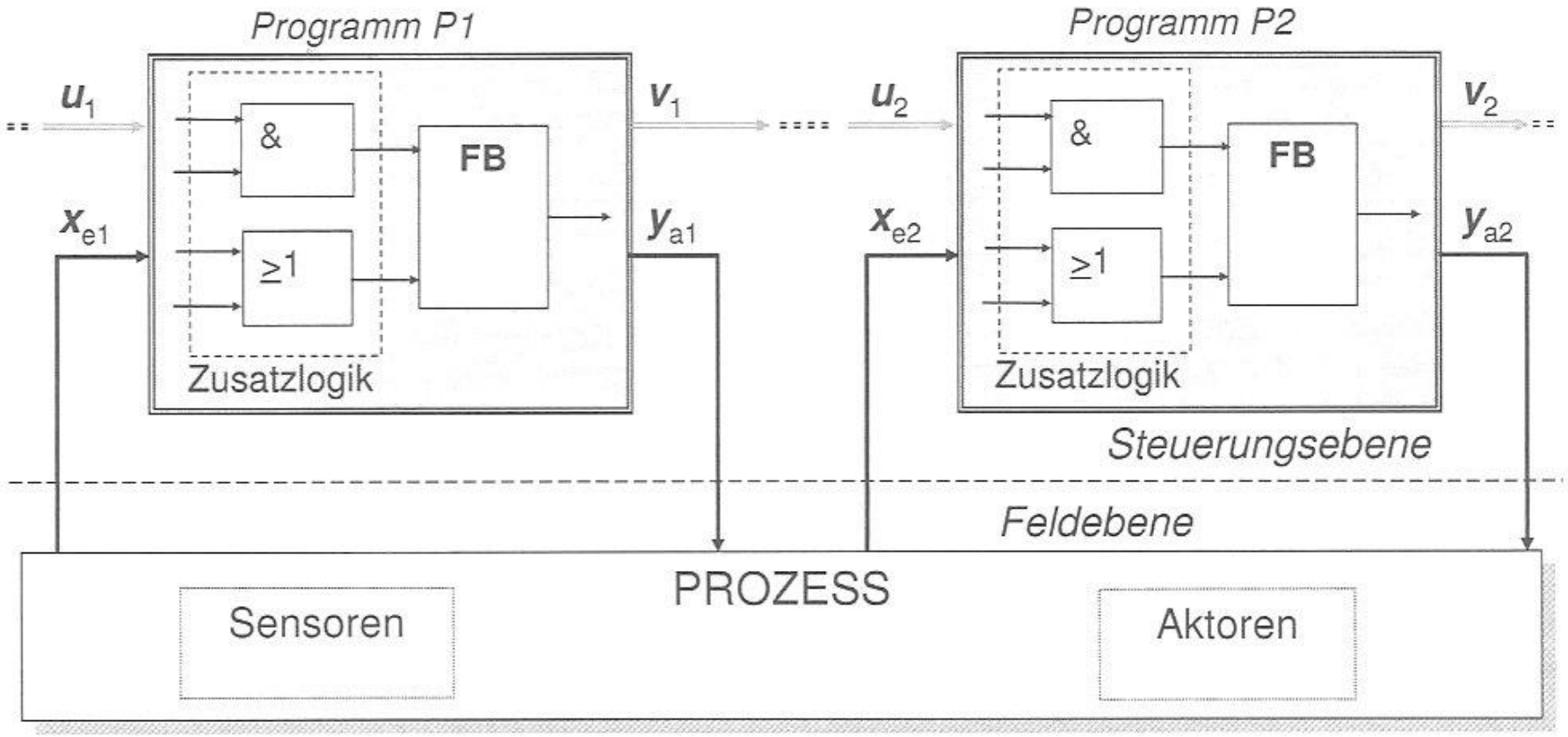


Abb. 04.12

4 SPS-Programmierung nach IEC 61134 (62 von 89)

4.2.2 Datenaustausch zwischen Programmen

Für den programmübergreifenden Datenaustausch sind die Steuer- und Statussignale ***u*** bzw. ***v*** als *globale Variablen* zu deklarieren.

Dagegen können die Mess- und Stellsignale $\boldsymbol{x}_e$ bzw. $\boldsymbol{y}_a$ lokal deklariert werden.

Sie müssen jedoch die vorgegeben E/A-Kanäle adressieren, um mit dem Prozess zu kommunizieren (z.B. AT %IX0.0).

4 SPS-Programmierung nach IEC 61134 (63 von 89)

4.3 Das Programmiermodell

Das Programmiermodell beschreibt, wie eine *Anwendersoftware* erstellt werden soll.

Bei der Erstellung der Anwendersoftware besteht die Möglichkeit, neben den in der IEC-Norm definierten Standard-Funktionen und –Funktionsbausteinen eigene, sogenannte Anwender-Funktionen und –Funktionsbausteine, zu entwickeln.

4 SPS-Programmierung nach IEC 61134 (64 von 89)

4.3 Das Programmiermodell

Davon wird in Projekten der verschiedenen Branchen auch rege Gebrauch gemacht.

Die Syntax zur Erstellung dieser Anwender-Funktionen und –Funktionsbausteine wurde im Beispiel 04.05 veranschaulicht.

Verwendet wurde in diesem Beispiel die Programmiersprache *Strukturierter Text (ST)*.

4 SPS-Programmierung nach IEC 61134 (65 von 89)

4.3 Das Programmiermodell

Außer dem Strukturierten Text (ST) sind in der IEC-Norm noch weitere Programmiersprachen definiert:

- *Anweisungsliste (AWL)*,
- *Funktionsbausteinsprache oder Funktionsplan (FBS oder FUP)*,
- *Kontaktplan (KOP)* und
- *Ablaufsprache (AS)*.

Diese werden in Abb. 04.13 dargestellt.

4 SPS-Programmierung nach IEC 61131

4.3.1 Programmiersprachen

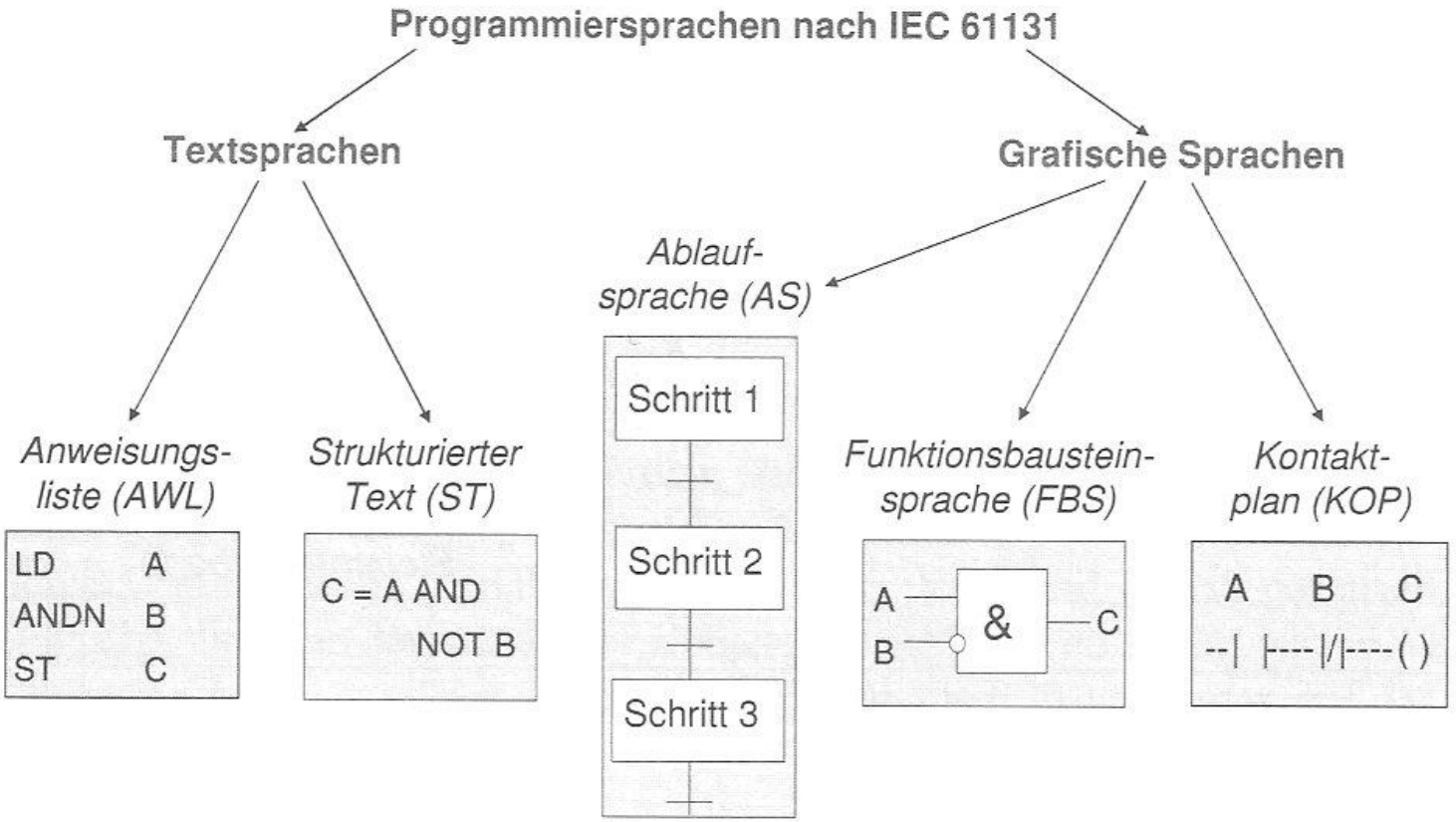


Abb. 04.13

4 SPS-Programmierung nach IEC 61134 (67 von 89)

4.3.1 Programmiersprachen

Die Einigung auf diese Sprachen ist historisch begründet.

So bildet der *Kontaktplan (KOP)*, im Englischen *Ladder Diagramm (LD)*, die Relais-Schaltung früherer verbindungsprogrammierter Steuerungen nach.

Er wird im amerikanischen Raum auch heute noch gern und oft eingesetzt.

4 SPS-Programmierung nach IEC 61134 (68 von 89)

4.3.1 Programmiersprachen

In Europa wird als grafische Sprache hauptsächlich die *Funktionsbausteinsprache (FBS)*, im Englischen *Function Block Diagram (FBD)*, verwendet, die an Blockschaltbilder und Logikpläne angelehnt ist.

4 SPS-Programmierung nach IEC 61134 (69 von 89)

4.3.1 Programmiersprachen - Textsprachen

Von den Textsprachen verwenden die Steuerungstechniker in Europa bis heute gerne die *Anweisungsliste (AWL)*, im Englischen *Instruction List (IL)*.

Sie ist der Assemblerprogrammierung sehr ähnlich und wurde in den Anfangszeiten der SPS eingesetzt um die Rechenzeit gering zu halten.

4 SPS-Programmierung nach IEC 61134 (70 von 89)

4.3.1 Programmiersprachen - Textsprachen

Wie in Abb. 04.13 zu sehen ist, enthält jede Zeile genau eine Anweisung.

Jede Anweisung besteht aus einem Operator gefolgt von einem Operanden.

Bei jeder Operation wird das aktuelle, im Arbeitsspeicher (auch Akku genannt) vorhandene Ergebnis über den Operator mit dem Operanden verknüpft.

4 SPS-Programmierung nach IEC 61134 (71 von 89)

4.3.1 Programmiersprachen - Textsprachen

In Tabelle 04.05 sind spezielle AWL-Operatoren (Anweisungen) aufgelistet.

Darüber hinaus können auch die IEC-Funktionen, z.B. AND aus Abb. 04.13, als Operatoren verwendet werden, wobei durch den Modifikator N die Operanden negiert werden können.

4 SPS-Programmierung nach IEC 61131 (72 von 89)

4.3.1 Programmiersprachen - Textsprachen

Operator	Beschreibung
LD	lädt den Wert einer Variablen in den Arbeitsspeicher.
LDN	lädt den Wert einer Variablen negiert in den Arbeitsspeicher.
ST	speichert aktuelles Ergebnis in einer Variablen.
STN	speichert aktuelles Ergebnis negiert in einer Variablen.
S	setzt Boole'sche Variable auf TRUE, wenn aktuelles Ergebnis TRUE ist.
R	setzt Boole'sche Variable auf FALSE, wenn aktuelles Ergebnis TRUE ist.
JMP	Sprung zur Marke (≡ Text mit nachfolgendem Doppelpunkt zu Beginn einer Zeile).
JMPC	Sprung zur Marke, wenn das Ergebnis TRUE ist.
JMPN	Sprung zur Marke, wenn das Ergebnis FALSE ist.
CAL	Aufruf eines Programms, eines Funktionsbausteins oder einer Funktion.
CALC	Aufruf, wenn das Ergebnis TRUE ist.
CALN	Aufruf, wenn das Ergebnis FALSE ist.
RET	Rücksprung von Funktion oder Funktionsbaustein.
RETC	Rücksprung, wenn das Ergebnis TRUE ist.
RETN	Rücksprung, wenn das Ergebnis FALSE ist.

Tabelle 04.05

4 SPS-Programmierung nach IEC 61134 (73 von 89)

4.3.1 Programmiersprachen - Textsprachen

Heute spielt die Rechenzeit eine immer kleinere Rolle, so dass die Hochsprachen wie Pascal, C oder *Strukturierter Text (ST)* sowohl zur Programmierung von Mikrocomputern als auch von SPSen immer mehr durchsetzen.

Im Strukturierten Text werden Befehle wie in Hochsprachen üblich bedingt (z.B. durch IF..THEN..ELSE) und /oder Schleifen (z.B. mit FOR..TO..DO) ausgeführt. Jeder Befehl endet mit einem Semikolon. Diese speziellen ST-Befehle sind in Tabelle 04.06 zusammengestellt.

4 SPS-Programmierung nach IEC 61131

4.3.1 Programmiersprachen - Textsprachen

	Befehl	Beschreibung (Beispiel)
Auswahlbefehle	IF	IF a < b THEN c:= 1; ELSIF a = b THEN c:= 2; ELSE c:= 3; END_IF
	CASE	CASE f OF 1: a:=3; 2: a:=4; ELSE a:=0; END_CASE;

Schleifen	FOR	FOR a:= 1 TO 10 BY 2 DO
		f[a] := b;
		END_FOR;

	WHILE	WHILE b > 1 DO
		b:=b/2;
		END_WHILE

	REPEAT	REPEAT a:= a * b; UNTIL a > 10000 END_REPEAT;
Abbruch	EXIT	Beendigung einer Schleife unabhängig von der Abbruchbedingung
Rücksprung	RETURN	Rücksprung aus einer Funktion oder einem Funktionsbaustein
Programmaufruf		PRG1();
Funktionsbau-steinaufruf		FB1(IN:=a, PT:=t#5s); c:=FB1.Q; (s. auch Bild 3.10)
Funktionsaufruf		F1(X,Y,Z); (s. auch Bild 3.10)

Tabelle 04.06

4 SPS-Programmierung nach IEC 61134 (75 von 89)

4.3.1 Programmiersprachen - Grafische Sprachen

Die Funktionsbausteinsprache kann Funktionen und Funktionsbausteinen wie in einem Blockschaltbild anordnen.

Sie eignet sich sehr gut zur Darstellung von Regelungen und binäre Logikschaltungen. Diese Darstellung wird weitläufig auch als *Funktionsplan (FUP)* bezeichnet.

4 SPS-Programmierung nach IEC 61134 (76 von 89)

4.3.1 Programmiersprachen - Grafische Sprachen

Eine Sonderstellung nimmt die *Ablaufsprache (AS)*, im Englischen *Sequential Function Chart (SFC)* ein, denn mit ihr können nur sequentielle Steuerungsaufgaben in Form von Schrittketten programmiert werden.

4 SPS-Programmierung nach IEC 61134 (77 von 89)

4.3.1 Programmiersprachen

4.3.1.3 Empfehlungen für den Einsatz der SPS-Programmiersprachen

Generell kann man sagen, dass die Textsprachen *unübersichtlicher* sind als die grafischen Sprachen.

Zur Entwicklung umfangreicher numerischer Algorithmen eignen sich Textsprachen aber häufig besser, da mit ihnen *effizienter* programmiert werden kann.

4 SPS-Programmierung nach IEC 61134 (78 von 89)

4.3.1 Programmiersprachen

4.3.1.3 Empfehlungen für den Einsatz der SPS-Programmiersprachen

Da SPSen jedoch stets Anlagen steuern, die gewartet und ab und zu umkonfiguriert werden müssen, ist die *Verständlichkeit* und *Übersichtlichkeit* der Software oberstes Gebot. Deshalb sollten übergeordnete Programme übersichtlich und änderungsfreundlich gestaltet und mit *grafischen Sprachen*, wie z.B. FBS oder AS, programmiert werden.

4 SPS-Programmierung nach IEC 61134 (79 von 89)

4.3.1 Programmiersprachen

4.3.1.3 Empfehlungen für den Einsatz der SPS-Programmiersprachen

Es wird folgender Einsatz der Programmiersprachen empfohlen:

- *Prozessabläufe* lassen sich sehr gut als Schrittketten beschreiben und werden deshalb in der *Ablaufsprache AS* programmiert,
- *Ansteuerprogramme* für Aktoren und Sensoren bestehen zumeist aus Verknüpfungslogik und werden deshalb in der *Funktionsbausteinsprache FBS* programmiert,

4 SPS-Programmierung nach IEC 61134 (80 von 89)

4.3.1 Programmiersprachen

4.3.1.3 Empfehlungen für den Einsatz der SPS-Programmiersprachen

Es wird folgender Einsatz der Programmiersprachen empfohlen:

- *Anwender-Funktionen und –Funktionsbausteine, die das Wartungspersonal in der Regel ohnehin als Black-Box ansieht, können effizient in Textsprachen wie AWL oder ST programmiert werden.*

4 SPS-Programmierung nach IEC 61134 (81 von 89)

4.3.2 Anwender-Datentypen

Für eine übersichtliche Programmstrukturierung hat der Anwender die Möglichkeit, eigene Anwender-Datentypen mit dem Schlüsselwort TYPE zu definieren.

Anwender-Datentypen können für verschiedene Zwecke verwendet werden.

Tabelle 04.07 zeigt eine Übersicht und im folgenden Beispiel 04.07 wird das verdeutlicht.

4 SPS-Programmierung nach IEC 61131

4.3.2 Anwender-Datentypen

Datentyp	Erläuterung	Beispiel
Referenz	Alternativer Name für Standard-Datentyp.	TYPE *Helligkeit* : USINT; END_TYPE
Aufzählung	Aufzählung von String-Konstanten.	TYPE *Ampeldaten* : (Rot,Gelb,Gruen); END_TYPE
Unterbereiche	Datentyp, dessen Wertebereich nur eine Untermenge eines Basistypen umfasst.	TYPE *SubInt* : INT (-4095..4095); END_TYPE
Datenfeld	Vektoren oder Matrizen elementarer Datentypen.	VAR *Bild* : ARRAY[0..8] OF *Helligkeit*; END_VAR
Struktur	Zusammenfassung ungleicher Datentypen.	TYPE *Merkmale* : STRUCT X_POS : REAL; Y_POS : REAL; GROESSE : INT; END_STRUCT END_TYPE

Tabelle 04.07

4 SPS-Programmierung nach IEC 61134 (83 von 89)

4.3.3 Anwender-Funktionsbausteine

Neben der Verwendung von IEC-Standard-Funktionsbausteinen, wie Counter, Timer, Speicher, hat der Anwender auch die Möglichkeit, eigene sogenannte *Anwender-Funktionsbausteine* zu erzeugen.

Wie jeder Funktionsbaustein können auch die Anwender-Funktionsbausteine mehrfach in Programmen eingesetzt werden.

4 SPS-Programmierung nach IEC 61134 (84 von 89)

4.3.3 Anwender-Funktionsbausteine

Diese Wiederverwendbarkeit von Funktionsbausteinen erlaubt es, die Software großer Projekte übersichtlich und *modular* zu strukturieren, wodurch sich der Programmcode reduziert und der Aufwand für Engineering und Test verringert.

4 SPS-Programmierung nach IEC 61134 (85 von 89)

4.3.3 Anwender-Funktionsbausteine

Die Kunst der *modularen Programmstrukturierung* besteht darin, Programmteile zu finden, die sich mehrfach wiederholen, und diese als Anwender-Funktionsbausteine zu programmieren.

4 SPS-Programmierung nach IEC 61134 (86 von 89)

4.3.3 Anwender-Funktionsbausteine

Einen Anwender-Funktionsbaustein würde man in der Informatik als *Klasse* mit bestimmten Eigenschaften bezeichnen, die in der Logik des Bausteins enthalten sind.

Durch Einfügen des Anwender-Funktionsbausteins in ein Programm entsteht ein *Objekt* oder eine sogenannte *Instanz* des Bausteins.

Instanzen müssen in den Programmen, in denen sie aufgerufen werden, wie Variablen mit Name und Typ deklariert werden (z.B. INSTANZ1:KLASSE_A in Abb. 04.16).

4 SPS-Programmierung nach IEC 61131

4.3.2 Anwender-Funktionsbausteine

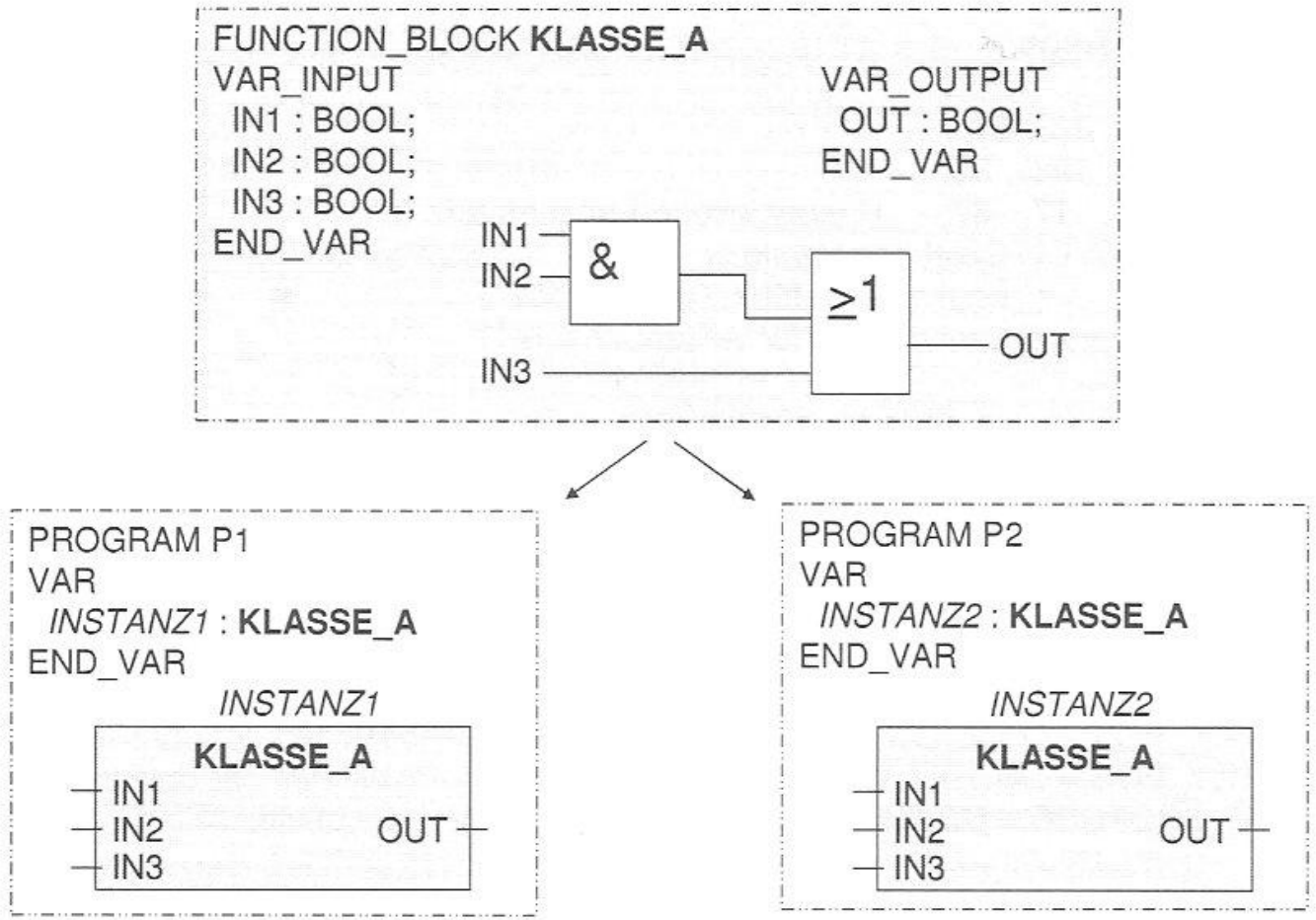


Abb. 04.16

4 SPS-Programmierung nach IEC 61134 (88 von 89)

4.3.3 Anwender-Funktionsbausteine

Man kann eine Instanz als Kopie oder spezielles Exemplar (Objekt) des Funktionsbausteins ansehen.

In einer Software kann es mehrere Instanzen einer Klasse geben.

Deshalb ist eine wichtige Eigenschaft der *Instanzierung*, dass die eingefügten Instanzen unabhängig voneinander ablaufen können und sich nicht gegenseitig beeinflussen.

4 SPS-Programmierung nach IEC 61134 (89 von 89)

4.3.3 Anwender-Funktionsbausteine

Beispielsweise bestehen INSTANZ1 und INSTANZ2 in Abb. 04.16 beide aus dem Baustein KLASSE_A.

Im Programm P1 wird INSTANZ1 aber im Allgemeinen mit anderen Werten abgearbeitet als INSTANZ2 im Programm P2.

- Ende Kap. 4 -