Grundwissen SPS-Technik

1.1 Historie

SPS steht für Speicherprogrammierbare Steuerung (engl. Programmable Logic Controller PLC).Mit der Betonung der Programmierbarkeit und der Ablage eines Programms in einem (Programm-)Speicher werden die Hauptmerkmale der Technik bereits im Namen festgeschrieben: Die Steuerungen selbst sind flexibel und universell für die verschiedensten Aufgaben einzusetzen, die aktuelle Funktion selbst wird mit einem Programm festgelegt. Der Part „Steuerung“ im Namen besagt nicht, dass diese Technik keine Regelungsaufgaben übernehmen kann.
SPS ist die dominierende Automatisierungstechnik für breiteste Aufgaben außerhalb der speziellen CNC- und Roboterwelt, wobei die Grenzen zunehmend fließend sind. Die erste Speicherprogrammierbare Steuerung wurde 1970 in Chicago für eine Werkzeugmaschinevorgestellt. 1973 erschienen erste Applikationen in Deutschland. Im Zeitraum 1975 bis 1985, spätestens bis 1990 vollzog sich der grundsätzliche Übergang von den **verbindungsprogrammierten Steuerungen (VPS)** mit Relais und Schützen und danach auch von **fest verdrahteter Logik** hin zu den speicherprogrammierten Steuerungen. 1983 erschien mit DIN 19239 „SPS-Programmierung“ die erste Norm zu speziellen Problemen der SPS, 1993 wurde diese Norm durch IEC 61131-3 ersetzt. Die Vorteile der SPS-Technik wie Flexibilität, geringerer Platzbedarf, höhere Zuverlässigkeit, geringere Kosten, Möglichkeit der Vernetzung mit anderen Systemen, Möglichkeiten für Fehlerdiagnose und Fernwartung der Programme und hauptsächlich der schnelle Funktionswechsel durch Programmänderung setzten sich durch. Verbindungsprogrammierte Steuerungen müssen bei jeder Änderung im Steuerungsablauf hardwareseitig umgebaut werden, was heute nicht mehr konkurrenzfähig ist. Auf der Aktorenseite sind bei der Anschaltung von Schützen Restfelder verdrahteter Steuerungen zumeist im Rahmen von Hilfskontakt-Schützen erhalten geblieben. Weiter gibt es sicherheitsrelevante Fälle, welche Festverdrahtung verlangen und bei denen es sogar verboten ist, sie per Standard- SPS zu steuern. Für solche Zwecke können heute allerdings **spezielle fehlersichere SPS** verwendet werden.
Das bekannteste und weitverbreiteste SPS-System ist Simatic S7 der Siemens AG mit den Baureihen S7-200 mit der Software Step7MicroWin und S7-300 sowie S7-400 mit Software Step7. Daneben gibt es eine Vielzahl anderer Hersteller. Die Systeme sind zueinander nicht kompatibel! Kompatibilität entsteht, wenn sich Hersteller bei der Entwicklung von Programmiersystemen hardwareunabhängig an die Norm IEC 61131-3 halten. Mit derartigen Systemen entstehen Programme, die auf verschiedensten Hardware-Plattformen laufen (z.B. CoDeSys Automation Alliance).

1.2 Grundsätzliche Struktur und Baugruppen von SPS

Die Struktur der SPS-Technik kann am besten mit der grundlegenden Lösung von Automatisierungsaufgaben demonstriert werden. Diese besteht im Einlesen von Signalen aus dem zu automatisierenden Prozess, der rechentechnischen Signalverarbeitung in einer CPU und dem Ausgeben von Signalen an die Stellorgane des Prozesses.

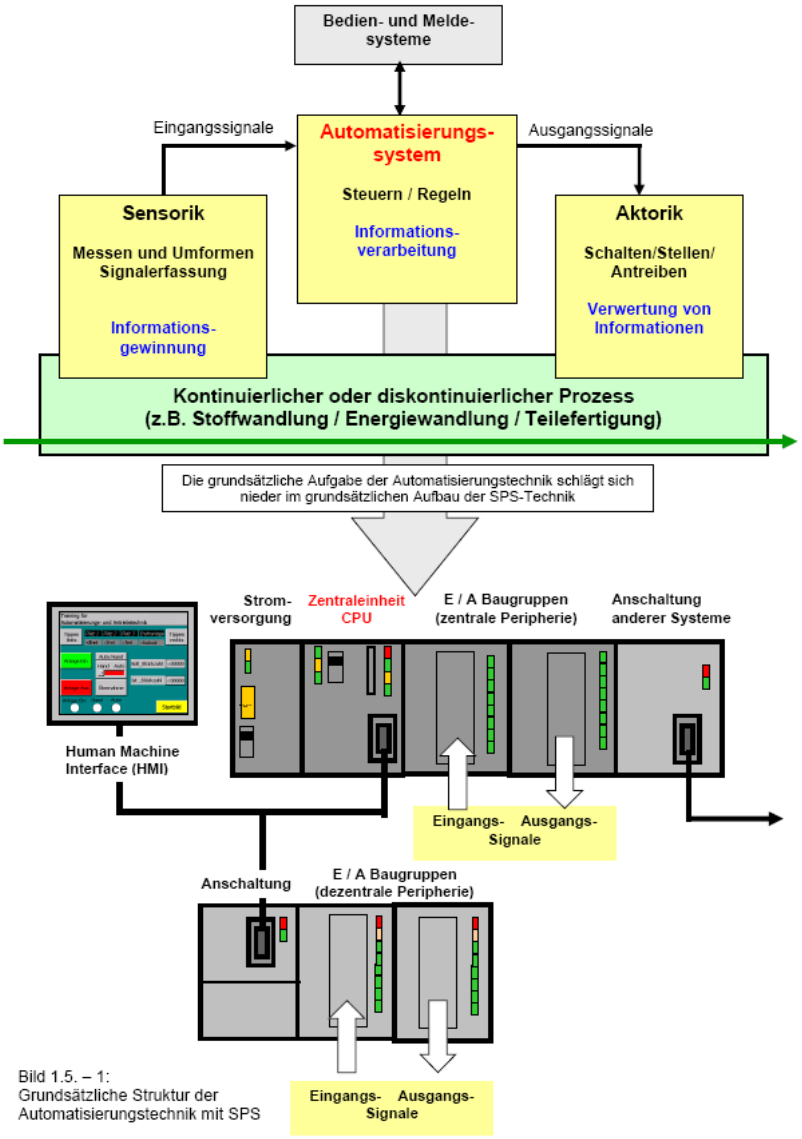


Bild 1.5. – 1:
Grundsätzliche Struktur der Automatisierungstechnik mit SPS

Bild 1.2 zeigt, dass eine SPS in minimaler Ausführung immer aus Eingabeeinheit, **Verarbeitungseinheit und Ausgabeeinheit** besteht (**EVA -Prinzip**). Vielfältige Zusatzbaugruppen sind möglich.

1.3 Typische Eigenschaften der Hardware:

Die Versorgungsspannung von SPS-Systemen ist heute überwiegend DC 24 V. Daraus werden für die CPU intern DC 5V bereitgestellt. Für analoge Signale sind +/-10V gebräuchlich. Die CPU beinhaltet den Prozessor und die erforderlichen internen und / oder externe Speicher für Firmware, Anwenderprogramm sowie remanente und nichtremanente Daten. Prozessor und Firmware sind auf ein „Echtzeitbetriebssystem“ für Logik und Arithmetik optimiert, d.h. auf schnelle Bit-, Byte- und Wortverarbeitung, wobei die vorzugsweise Adressierung von Byte oder Wort bei unterschiedlichen SPS auch unterschiedlich sein kann.
Für Standard-Applikationen steht eine Vielzahl von Software-Bibliotheken zu Verfügung. Aktuelle CPU verfügen über Schnittstellen für gängige Bussysteme.

Eingänge werden überwiegend über Optokoppler galvanisch vom Prozess entkoppelt und mittels Tiefpässen entstört. Beim Anschalten von Signalgebern ist zu beachten, dass mehrere Eingänge häufig ein gemeinsames Massepotential besitzen.

Digitale Ausgangsbaugruppen haben meist kontaktlose Ausgänge mit Transistoren mit 24V Ausgangsspannung und typisch 500mA Strombelastbarkeit, oder Baugruppen mit Relaisausgang. Grundsätzlich ist die Versorgungsspannung an die Baugruppen heranzuführen, teilweise auch über Baugruppensicherungen. Bei kontaktlosen Ausgängen ist die Notwendigkeit einer galvanischer Trennung besonders zu prüfen!

Analoge Eingangsbaugruppen enthalten Analog-Digital-Wandler (ADU) mit unterschiedlicher Auflösung (8, 10, 12, oder 15 Bit). Jedem Analogsignal wird ein Eingangswort zugeordnet, in welches der digitale Wert geschrieben wird. Bit 15 ist dem Vorzeichen vorbehalten. Die Wandlung erfolgt zyklisch. Typische Wandlungszeit <100 ms. Die Kanäle werden per Software, teilweise auch noch mit Hardwareeinstellungen parametriert. Typische Signale: 4 .. 20 mA, -10 ..+10V, 0 .. 10V.

Analoge Ausgangsbaugruppen enthalten Digital-Analog-Umsetzer (DAU). Diese wandeln digitale Ausgangsworte in genormte analoge Signale wie 4 .. 20mA, 0..10V, -10..+10V Die Wandlung erfolgt zyklisch, Wandlungszeit typisch <100ms.

1.3 Programmbearbeitung, Prozess-Abbilder und Mehrfachzuweisungen

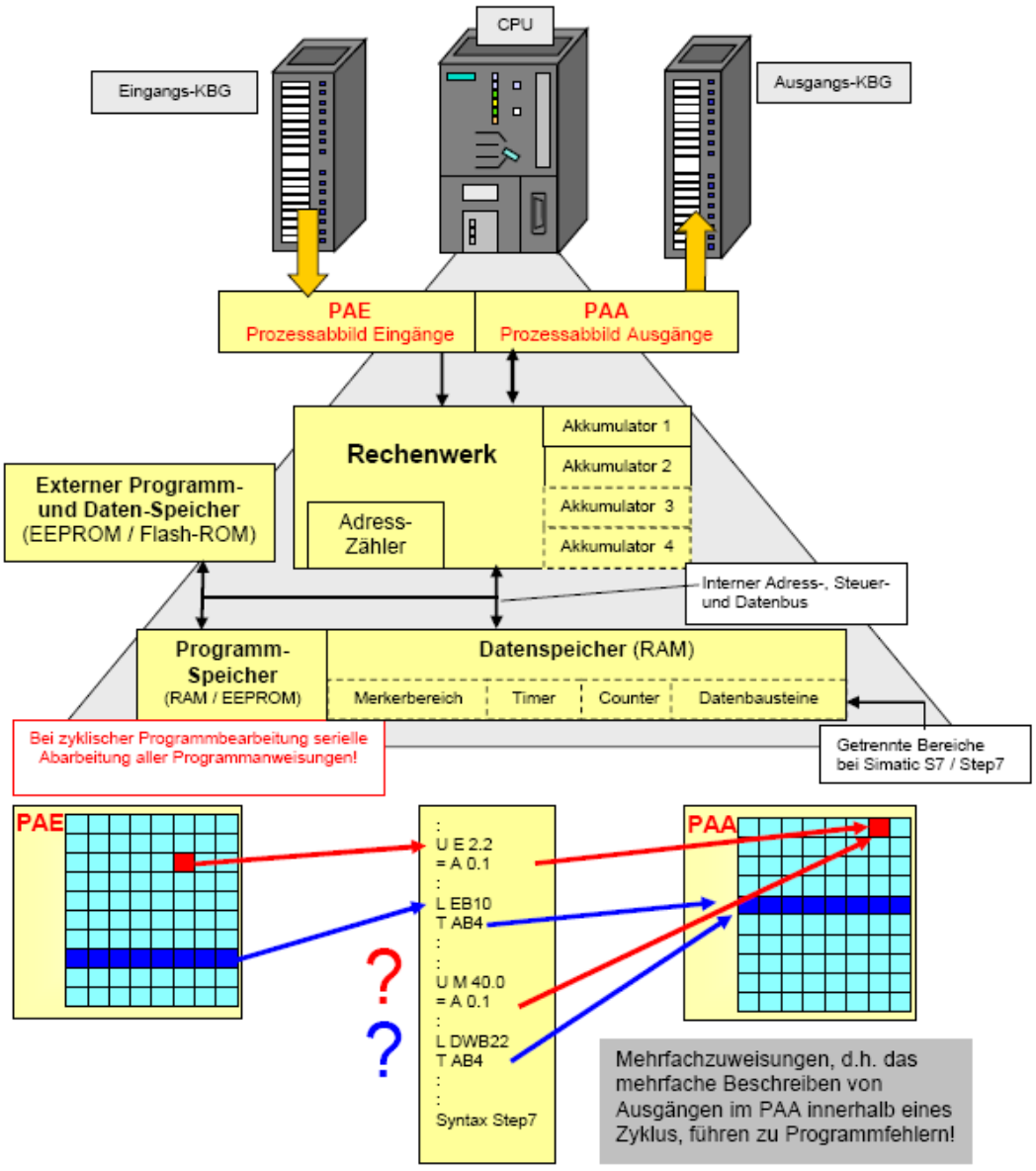


Bild 1.3: Schematische Darstellung wesentlicher Hardwarekomponenten einer SPS und Probleme der Mehrfachzuweisungen bei zyklischer Programmbearbeitung
Hinsichtlich des Datenspeichers und der Programmsyntax orientiert sich die Darstellung an Simatic S7.

Aktuelle SPS-Technik unterscheidet **drei Formen der Bearbeitung des Anwenderprogramms**.

- zyklische Programmbearbeitung (Bild 1.4):

Standard, niedrigste Priorität, angewendet z.B. für Ablaufsteuerungen, Verriegelungen, Grenzwertüberwachungen, langsame Zähler, allgemeine Aufgaben
Die zyklische Programmbearbeitung wird bei Step7 durch den Organisationsbaustein OB1 bewirkt.

Der Standard-Zyklus einer SPS:

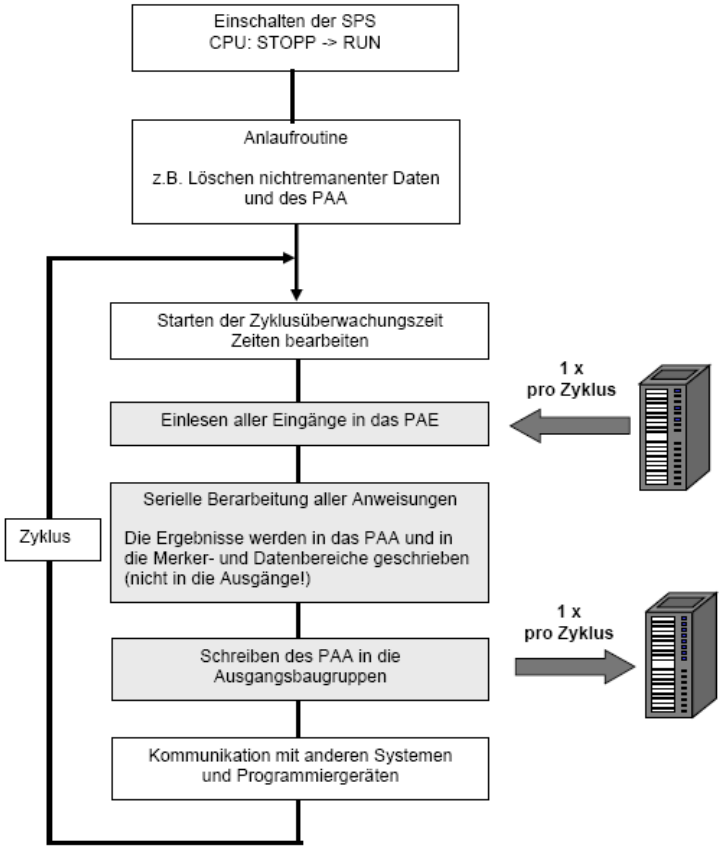
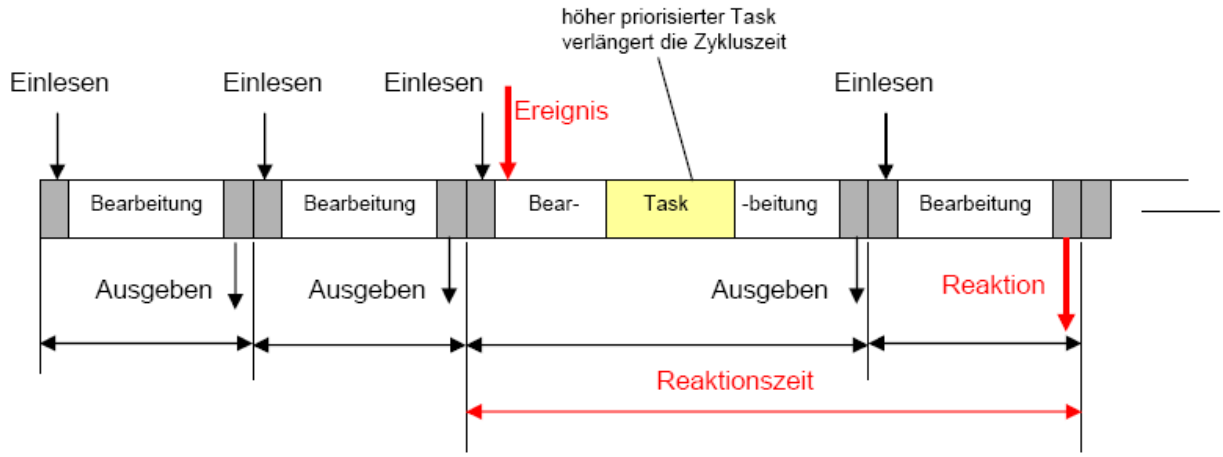


Bild 1.4: Schema des Zyklus einer SPS

Die **Zykluszeit** der SPS ist die Zeit für die Abarbeitung eines Zyklus einschließlich aller Kommunikationsaufgaben. Sie ist abhängig von der Rechenzeit für das Programm, also abhängig von der Zahl der Anweisungen. Höherpriorisierte Tasks unterbrechen den Zyklus und **verlängern** damit die
Zykluszeit. Für eine Ermittlung der **Rechenzeit** stellen die Hersteller Werte zur Verfügung: z.B. CPU 214: für 1000 AWL- Befehle 0,8 ms
Die Zykluszeit wird überwacht. Überschreitet sie eine einstellbare **Zyklusüberwachungszeit** (typisch 150 ms), so geht die CPU in Stopp.

Die Reaktionszeit ist die Zeitdauer zwischen der Änderung eines Eingangssignals und der Reaktion
durch Änderung des Ausgangssignals.



Die Reaktionszeit auf ein binäres Ereignis ist im ungünstigsten Fall die Summe aus der doppelten maximal zu erwartenden Zykluszeit (worst case), der Verzögerungszeit des Binäreingangs (typisch 2-5 ms) und der Verzögerung des Aktors (z.B. Magnetventil typisch 20-30 ms).

Vorteile der zyklischen Programmbearbeitung:
Die Rechenleistungen der CPU werden immer genutzt. Es sind keine Berechnungen der Zykluszeiten erforderlich.

Nachteile der zyklischen Programmbearbeitung:
Weil die Rechenzeit schwanken kann, ist das System streng genommen nicht deterministisch!
Zeitkritische Programmteile werden durch Programmänderungen beeinflusst, auch wenn dort selbst
keine Veränderungen vorgenommen werden.
Die serielle Bearbeitung aller Anweisungen hat Konsequenzen: Das Programm wird grundsätzlich nur
mit Werten aus dem Prozessabbild der Eingänge berechnet. Andernfalls könnten die Eingangssignale
während des Zyklus mit unterschiedlichen Werten in die Programmbearbeitung eingehen!
Ausgangssignale dürfen im Programm nicht mehrmals „geschrieben“ werden, weil zuvor berechnete
Werte im PAA durch nachfolgende Werte überschrieben werden. Die gefürchteten „**Mehrfachzuweisungen**“ führen immer zu **Programmfehlern**!

- zeitgesteuerte (zeitzyklische) Programmbearbeitung
mittlere Priorität, angewendet z.B. für regelungstechnische Funktionen, zeitkritische Binärverarbeitungen Die zeitgesteuerte Programmbearbeitung wird bei Step7 durch Organisationsbausteine für Uhrzeit- und Weckalarme (z.B. OB10, OB35) bewirkt. Bei der zeitgesteuerten Programmabarbeitung muss die Zykluszeit bestimmt werden, damit zeitgesteuerte Programmaufrufe und Rechenzeiten nicht kollidieren. Die Rechenzeit sollte maximal 70-80 % der zugewiesenen Zykluszeit betragen. Werden mehrere höher priorisierten Task's aufgerufen, sollte die Rechenzeitreserve der zeitzyklischen Task entsprechend erhöht werden. Bei regelungstechnischen Anwendungen sollte die Zykluszeit z.B. mindestens um den Faktor 5 kleiner sein als die Summenzeitkonstante des geschlossenen Regelkreises.

- ereignisgesteuerte Programmbearbeitung
höchste Priorität, vergleichbar mit Interrupt, angewendet z.B. für ausgewählte, besonders zeitkritische Prozesse. Für die Auswertung von Ereignissen des Prozesses benötigt man spezielle interruptfähige Binäreingangskarten (höherer Preis!).
Die (interne) ereignisgesteuerte Programmbearbeitung wird bei Step7 durch Fehlerbausteine (z.B. OB86: Ausfall eines DP-Slave) bewirkt.

- Multitasking
Im Zusammenhang mit der Bearbeitung umfangreicher Programme und bei zeit- und ereignis- gesteuerter Programmbearbeitung werden Programmteile als **Task** bezeichnet (Task engl. für Aufgabe, Prozess, in Anspruch nehmen).
Ein Task ist der aktuell wirksame Teil des Anwenderprogramms zur Laufzeit, organisiert z.B. durch Zeitschlitze oder Prioritäten. Programmsteuerung durch Task stellt zumeist eine Alternative zur zyklischen Programmbearbeitung dar. Einige Task's vom Betriebssystem können von der Anwendersoftware nicht beeinflusst werden (Funktionen des Betriebssystems – z.B. Kommunikation mit der Programmierschnittstelle oder die Task des Windows-Betriebssystems. Es ist zu beachten, dass bei Zugriff auf die Daten einer anderen Task u. U. nicht alle Daten aus dem gleichen Zyklus stammen. Wird das Anwenderprogramm in mehrere Task unterteilt, dann muss sichergestellt werden, dass die zur Verfügung stehende Rechenzeit nicht überschritten wird. Von der Gesamtrechenzeit muss die Systemzeit des Betriebssystems abgezogen werden.

Beispiel: Task 1 Zykluszeit 20 ms, Rechenzeit 10 ms
Task 2 Zykluszeit 50 ms, Rechenzeit 30 ms
Systemzeit ca. *5 %*
Die eingestellten Zykluszeiten können nicht eingehalten werden (Echtzeitfehler!).
Task2 kann nur alle 100 ms aufgerufen werden!

1.4 Signale, Daten und Speicher

Automatisierung bedeutet immer auch Signalverarbeitung. **Signale sind Träger von Informationen** aus dem zu automatisierenden Prozess. Nach der Verarbeitung wird mit Signalen über Stellglieder in den Prozess eingegriffen. Aus der Sicht der Datenverarbeitung übertragen Signale **Daten** des Prozesses, und die Automatisierungseinrichtungen vollziehen eine Datenverarbeitung. Daten stehen für Informationen und Werte. Dies erfordert die exakte **Festlegung von Datenformaten**. neben **binären** Signalen von Schaltelementen und **digitalen** Signalen (z.B. von Encodern) liegen **analoge** Signale in Form von Spannung, Strom, Druck, Lichtintensität, Temperatur u. andere vor. Für die Automatisierungseinrichtungen werden sie zu genormten Signalen (überwiegend Strom oder Spannung) gewandelt und digitalisiert. Primär wird die Information von Signalen durch Amplitude, Frequenz oder Form des Signals dargestellt, in der Digitaltechnik ausschließlich durch eine Anzahl von Bits. Die CPU von Automatisierungsgeräten verarbeiten ausschließlich binäre und digitale Signale (Daten). Dazu müssen diese auf bestimmten Speicherplätzen abgelegt werden und über deren **Adressen** verfügbar sein.

Für diese Aufgaben haben sich zwei Methoden entwickelt:

1. Die Speicherplätze müssen vom Anwender selbst mit Byte- und Bitadresse adressiert werden (**absolute oder direkte Adressierung**). Dies gilt z.B. für das System Simatic S5 / S7 für den Merker- und Datenbaustein-Bereich.
Anstelle der absoluten Adressen kann der Anwender für jedes Datum ein **Symbol** definieren. Die symbolische Adressierung ist der absoluten Adressierung vorzuziehen!

2. Das System adressiert seinen Datenspeicher selbst. Hier muss. der Anwender für alle Signale (Daten) **Variablen deklarieren**. Durch das Anlegen von Variablen mit zugeordnetem Datentyp werden die erforderlichen Speicherplätze reserviert. Variablen wirken wie **Platzhalter** für die Daten im Datenspeicher. Diese Adressierung erfolgt beispielsweise im System CoDeSys und wird nach IEC 61131-3 vorgeschrieben. Einzelheiten sind nachfolgenden Abschnitten zu entnehmen.

1.4.1Datenhaltung

Eine Besonderheit des Systems Simatic S7 gegenüber der Norm IEC 61131 ist die Pflicht des Programmierers, globale Daten selbst zu adressieren. Dazu stehen unterschiedliche Speicherbereiche zur Verfügung. Weiter erfolgt bei Simatic S7 die Einbindung von Ein- und Ausgangssignalen in ein Programm nicht durch Legen von Variablen auf Adressen, sondern durch Ansprechen der Werte in den Prozessabbildern PAE und PAA mit den Operanden E und A und deren Adressen.
Der Begriff der Variablen und die Variablendeklaration spielen deshalb in Step7 nicht die zentrale Rolle wie in IEC- Programmiersystemen.

Systemspeicher (CPU)			
Operanden			Beispiel
Prozessabbild der Eingänge PAE	Binärer Eingang (Eingangsbit) Eingangsbyte Eingangswort Eingangsdoppelwort	**E**	E 4.0 EB 4 EW 4 ED 4
Prozessabbild der Ausgänge PAA	Binärer Ausgang (Ausgangsbit) Ausgangsbyte Ausgangswort Ausgangsdoppelwort	**A**	A 12.0 AB 12 AW 12 AD12
Merker	Merkerbit Merkerbyte Merkerwort Merkerdoppelwort	**M**	M 1.0 MB 1 MW 2 MD 2
Zeitglieder	Zeitworte der Timer	**T**	T0 ... T63 ...
Zähler	Zählerworte der Counter	**Z**	Z0 ... Z63 ...
Datenbausteine	Datenbit Datenbyte Datenwort Datendoppelwort	**DB**	DB2.DBX 4.0 DB2.DBB 4 DB2.DBW 4 DB2.DBD 4

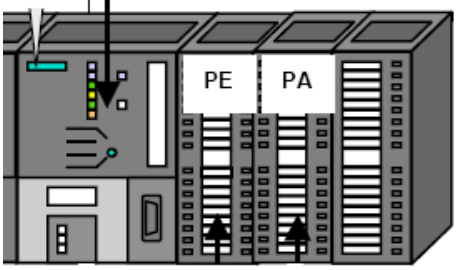


Bild 1.5 zeigt die globalen Daten in der Übersicht mit ihren Kennungen sowie Beispiele der Adressierung

Der **Merkerbereich** dient zur Ablage von Zwischenergebnissen jeder Art, wenn diese global verfügbar sein müssen. Seine Größe ist abhängig von der CPU und kann z.B. 2048 Byte betragen. Auf Merker kann im Format Bit, Byte, Wort oder Doppelwort zugegriffen werden. Binäre Merker werden mit Kennzeichen M und Bytenummer **.** Bitnummer adressiert. Die Adresse von Wort und Doppelwort richtet sich stets nach dem nieder- wertigsten enthaltenen Byte (**Bild 1.6**).

Adressieren eines BOOL / Bit:	Byteadresse . Bitadresse
Adressieren eines BYTE:	Byteadresse
Adressieren eines WORD:	Byteadresse des niederwertigen BYTE
Adressieren eines DWORD:	Byteadresse des niederwertigsten der vier BYTE.

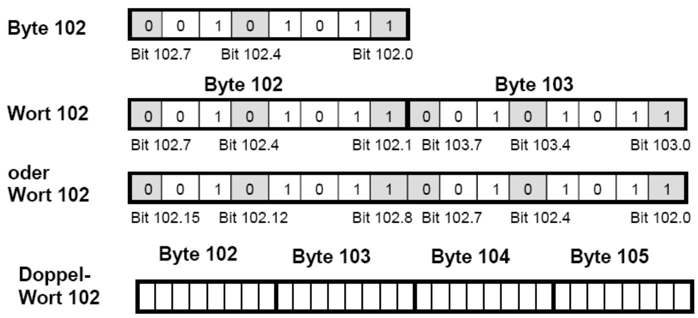


Bild 1.6: Regeln der Adressierung im System Simatic

In gleicher Weise werden Eingänge und Ausgänge als Bit, Bit, Wort und Doppelwort angesprochen.

Beispiele: E 22.0; EB 22; EW 22; ED 22
A 40.2; AB 40; AW 40; AD 40

Gültige Bereiche für die S7-200 CPUs

Tabelle 1: Speicherbereiche und Funktionen der S7-200 CPUs

Beschreibung	CPU 221	CPU 222	CPU 224
GrößeAnwenderprogramm	2 K Wörter	2 K Wörter	4 K Wörter
GrößeAnwenderdaten	1 K Wörter	1 K Wörter	2,5 K Wörter
Prozeßabbild der Eingänge	E0.0 bis E15.7	E0.0 bis E15.7	E0.0 bis E15.7
Prozeßabbild der Ausgänge	A0.0 bis A15.7	A0.0 bis A15.7	A0.0 bis A15.7
Analogeingänge (nur Lesen)	--	AEW0 bis AEW30	AEW0 bis AEW30
Analogausgänge (nur Schreiben)	--	AAW0 bis AAW30	AAW0 bis AAW30
Variablenspeicher (V)[1]	VB0.0 bis VB2047.7	VB0.0 bis VB2047.7	VB0.0 bis VB5119.7
Lokaldatenspeicher (L)[2]	LB0.0 bis LB63.7	LB0.0 bis LB63.7	LB0.0 bis LB63.7
Merker (M)	M0.0 bis M31.7	M0.0 bis M31.7	M0.0 bis M31.7
Sondermerker (SM) Schreibgeschützt	SM0.0 bis SM179.7 SM0.0 bis SM29.7	SM0.0 bis SM179.7 SM0.0 bis SM29.7	SM0.0 bis SM179.7 SM0.0 bis SM29.7
Zeiten	256 (T0 bis T255)	256 (T0 bis T255)	256 (T0 bis T255)
Speichernde Einschaltverzögerung 1 ms	T0, T64	T0, T64	T0, T64
Speichernde Einschaltverzögerung 10 ms	T1 bis T4, T65 bis T68	T1 bis T4, T65 bis T68	T1 bis T4, T65 bis T68
Speichernde Einschaltverzögerung100 ms	T5 bis T31, T69 bis T95	T5 bis T31, T69 bis T95	T5 bis T31, T69 bis T95
Ein-/ Ausschaltverzögerung 1 ms	T32, T96	T32, T96	T32, T96
Ein-/ Ausschaltverzögerung 10 ms	T33 bis T36, T97 bis T100	T33 bis T36, T97 bis T100	T33 bis T36, T97 bis T100
Ein-/ Ausschaltverzögerung 100 ms	T37 bis T63, T101 bis T255	T37 bis T63, T101 bis T255	T37 bis T63, T101 bis T255
Zähler	Z0 bis Z255	Z0 bis Z255	Z0 bis Z255
Schnelle Zähler	HC0, HC3, HC4, HC5	HC0, HC3, HC4, HC5	HC0 bis HC5
Ablaufsteuerungsrelais (S)	S0.0 bis S31.7	S0.0 bis S31.7	S0.0 bis S31.7
Akkumulatoren	AC0 bis AC3	AC0 bis AC3	AC0 bis AC3
Sprünge/Sprungmarken	0 bis 255	0 bis 255	0 bis 255
Aufrufe/Unterprogramme	0 bis 63	0 bis 63	0 bis 63
Interruptprogramme	0 bis 127	0 bis 127	0 bis 127
PID-Regler	0 bis 7	0 bis 7	0 bis 7
Schnittstelle	Schnittstelle 0	Schnittstelle 0	Schnittstelle 0

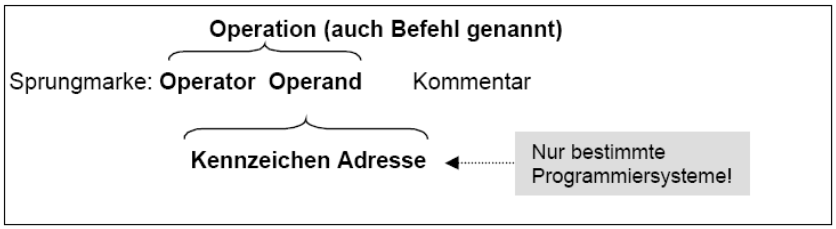
1 Der gesamte Variablenspeicher kann nullspannungsfest gespeichert werden.
2 LB60 bis LB63 sind für STEP 7-Micro/WIN 32 Version 3.0 oder höher reserviert.

1.5 Die klassischen Programmiersprachen

Mit der Entwicklung der SPS-Technik wurden zuerst **drei Programmiersprachen** eingeführt:
Anweisungsliste (AWL), Funktionsplan (FUP) und Kontaktplan (KOP)

- **Anweisungsliste**

Die AWL ist eine zeilen- und textorientierte Sprache. Der Anwender muss die Syntax der Befehle selbst kennen, d.h. er kann nicht aus einer Liste von Befehlen „per Mausklick“ auswählen. In AWL können Aufgaben effizient, elegant und individuell gelöst werden.
Eine (maximal ausgeführte) Anweisung besteht aus



Beispiel 1: MARK: UN E 2.0 //Abfrage Lichtschranke

oder mit Symbolvereinbarung: Lichtschranke ←→ E2.0

Beispiel 2: MARK: UN “Lichtschranke“

Beispiel 3: LD Lichtschranke (*Abfrage Lichtschranke, Lichtschranke ist eine Variable vom Typ BOOL*)

Die **Sprungmarke** erlaubt optional, diese Anweisung von anderer Stelle des Programms aus zu erreichen. In den Beispielen 1 und 2 ist der Name der Sprungmarke „MARK“.

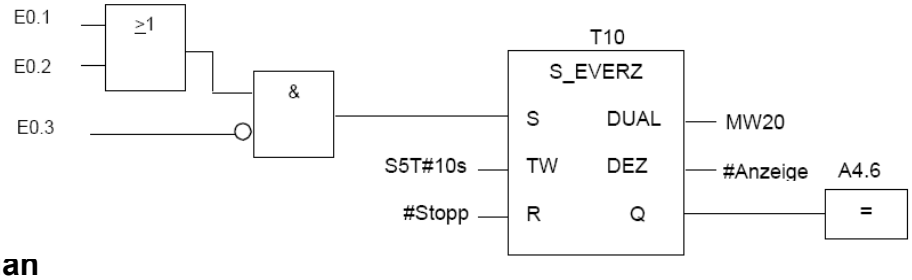
Der **Operator** sagt aus, **was** getan werden soll. In den Beispielen werden die Operatoren UN bzw. LD verwendet.
Der **Operand** sagt aus, **womit** dieses getan werden soll.
Der **Kommentar** hat kein Auswirkungen auf das Programm und wird auch nicht zur Hardware übertragen.

- **Funktionsplan**

Der Funktionsplan ist eine graphische Sprache und benutzt u. a. die bekannten Bausteine und Symbole von Logikplänen. Weitere Bausteine wurden ergänzt. Ein Vorzug von FUP ist, dass man die Operationen zumeist aus bereitgestellten Katalogen auswählen kann und dadurch nicht alle Einzelheiten der Operation selbst kennen muss.

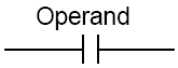
Beispiel eines FUP: Darstellungsart Step7

.

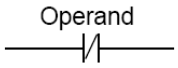


- **Kontaktplan**

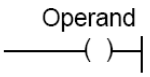
Die graphische Sprache Kontaktplan lehnt sich am Stromlaufplan kontaktbehafteter Steuerungstechnik an. Ein Argument: Durchgeschaltete Strompfade kann man Online am Programmiergerät auch von weitem erkennen. Auch im KOP wählt man die Operationen aus bereitgestellten Katalogen aus .Bei Verwendung komplexer Elemente wie Zeitgeber, Zähler, Datentransfer u. a. verlieren die Kontakte zunehmend an Bedeutung. Der KOP wird dann dem FUP sehr ähnlich. Als Grundbausteine kennt der KOP drei Elemente:



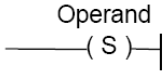
Signalabfrage auf „1“
Dieser Strompfad wird geschlossen, wenn der Wert des Operanden TRUE ist.



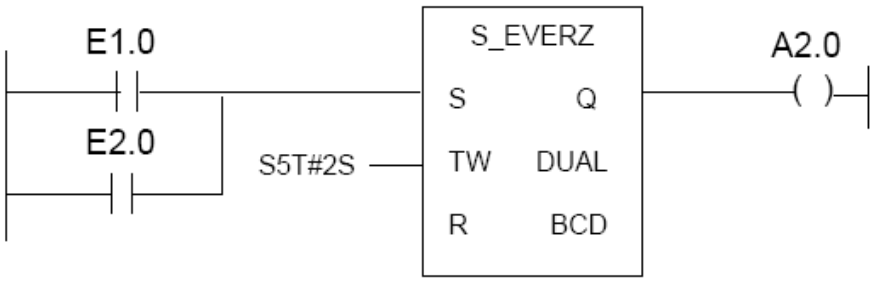
Achtung! Nicht formal mit Schließer-Kontakt gleichsetzen!
Signalabfrage auf „0“
Dieser Strompfad wird geschlossen, wenn der Wert des Operanden FALSE ist.



Spule
Die Spule führt Signal TRUE, wenn der Strompfad geschlossen ist.
oder Das Spulenzeichen kann modifiziert werden, z.B. mit Eigenschaft Setzen oder Rücksetzen



Beispiel eines KOP: Darstellungsart Step7



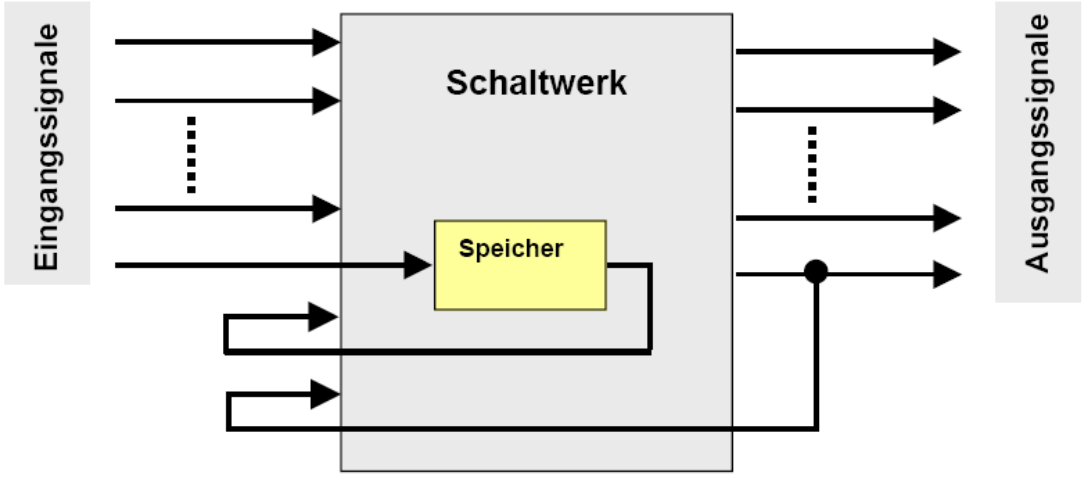
2. Binäre Steuerungen und Boolesche Algebra

2.1 Übersicht

Binäre Steuerungen zeichnen sich durch ihre Beschränkung auf **Bitoperationen** aus. Sie verarbeiten ausschließlich Bitsignale. Alle Variablen sind vom Typ BOOL. Systeme der binären Signalverarbeitung heißen auch **logische Netzwerke.** Diese enthalten **logische Funktionen**. Je nachdem ob die logischen Netzwerke innere Signalspeicher enthalten oder nicht, werden sie unterschieden:

• **Kombinatorische Netzwerke = Schaltnetze = Verknüpfungssteuerungen ohne Speicher**
Sie enthalten keine inneren Speicher und keine Rückführungen von Ausgangssignalen auf Eingänge. Als Folge sind die Ausgangssignale eindeutig eine logische Funktion (Kombination) allein der Eingangssignale.

• **Sequentielle Netzwerke = Schaltwerke = Verknüpfungssteuerungen mit Speicher**
Sie enthalten innere Speicher und / oder Rückführungen von Ausgangssignalen auf Eingänge. Als Folge sind die Ausgangssignale nicht mehr eindeutig nur eine logische Funktion (Kombination) der Eingangssignale, sondern hängen auch noch vom Zustand der Ausgangssignale bzw. der inneren Speicher ab.

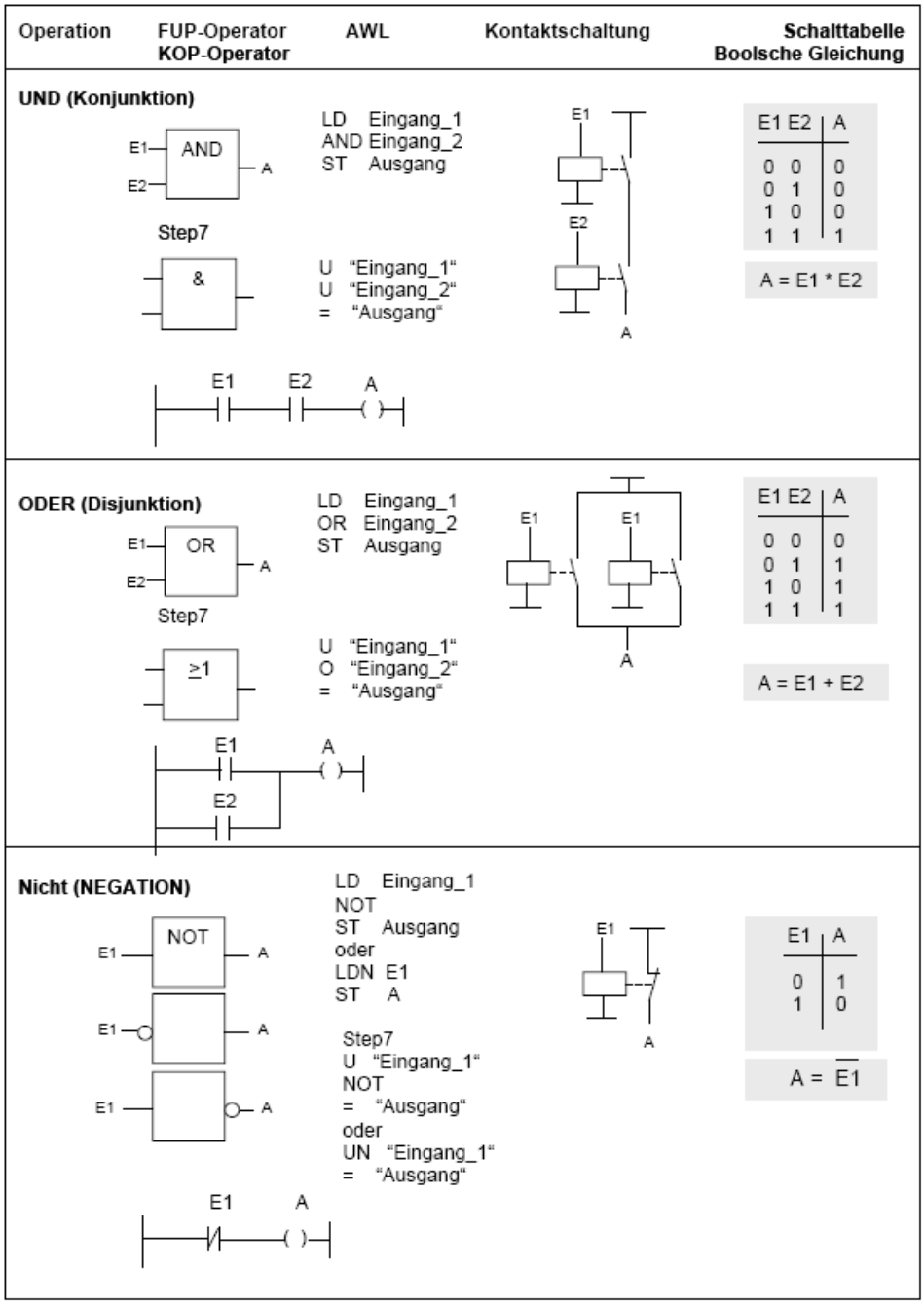


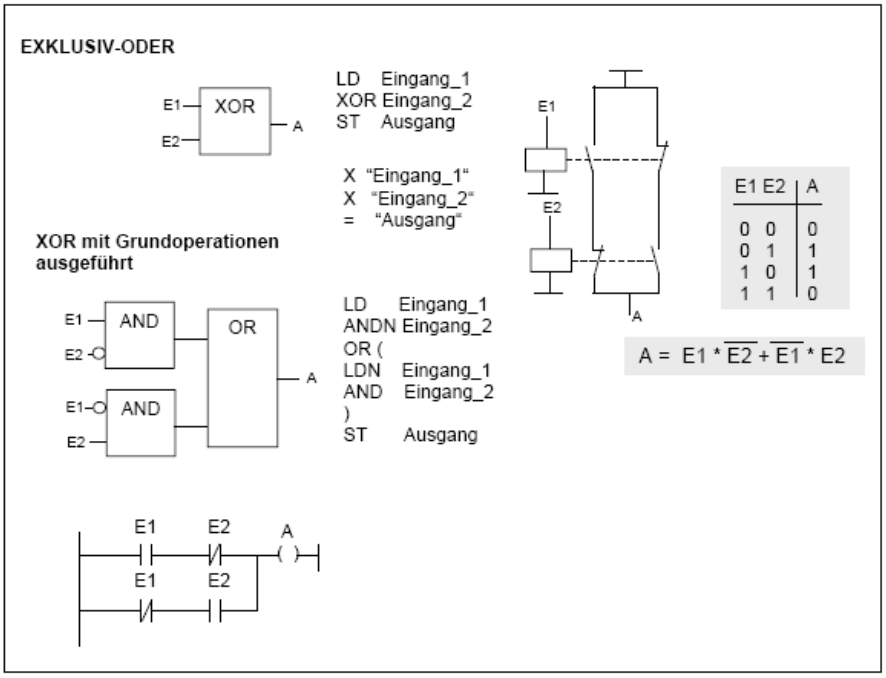
Binäre Steuerungen wurden mit unterschiedlichsten Techniken realisiert:
• mit pneumatischen und hydraulischen Schaltgliedern
• Elektrisch mit Kontakt-Schaltelementen, Logikbaugruppen und schließlich mit programmierbaren Baugruppen

Grundsätzlich haben pneumatische und hydraulische Steuerungen deutlich an Bedeutung verloren. Hingegen besitzen die pneumatischen und hydraulischen **Aktoren und Stellglieder** weiterhin eine große Bedeutung.
Durchgesetzt haben sich auch in der binären Steuerungstechnik **programmierbare Steuerungen**, da sie bereits für kleinere Aufgaben mit nur wenigen Verknüpfungen wirtschaftlich sind. Binäre Steuerungstechnik auf Basis von **Relais und Schützen** wird seit Beginn der Elektrotechnik praktiziert. Bereits hier wurden die logischen Grundfunktionen definiert. Da die Realisierung einer logische Funktion wie z.B. eines UND-Gliedes durch Relais nicht zu vernachlässigende Kosten verursacht, wurde durch Anwendung der **Boolschen Algebra und Minimierungsverfahren** versucht, die Logik von jedweder Redundanz zu befreien. Jedes eingesparte Relais bedeutete geringere Kosten. Die **Boolesche Algebra** (Schaltalgebra) ist eine mathematische Beschreibung für die Beziehungen zwischen logischen Variablen, die nur zwei Zustände annehmen können. Ähnlich wie in der Algebra existieren ein Kommutativ-, Assoziativ- und Distributiv-Gesetz für das Ausklammern und Tauschen von Termen mit Variablen. Bei programmierbarer Technik hat die Einsparung einer Anweisung nur in Sonderfällen Auswirkungen!
Der Speicherplatz für einige Anweisungen mehr oder weniger fällt im Allgemeinen nicht ins Gewicht. Deshalb haben heute die Minimierungsverfahren wie auch die Boolesche Algebra **an Bedeutung verloren**. Es gibt aber weiterhin Sonderprobleme, wo Logik entwickelt und minimiert werden muss.

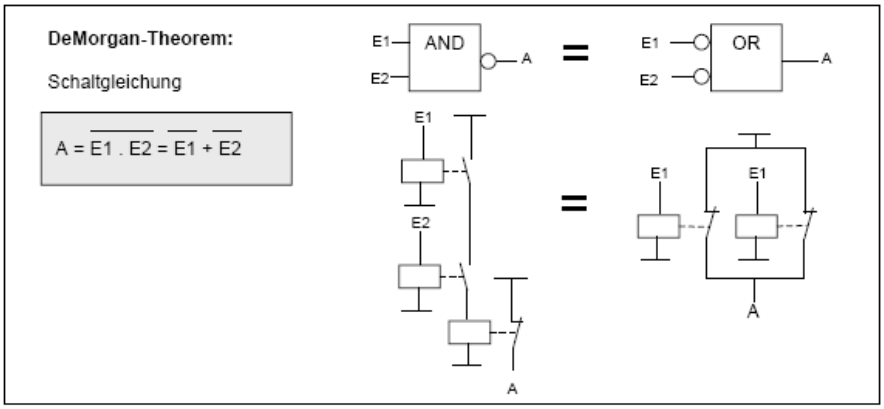
5.2 Logische Grundfunktionen

Für die Verknüpfung von Signalen in Schaltnetzen reichen wenige logische Grundfunktionen aus, so wie sie mit der Relaistechnik realisierbar waren. Nachfolgende Übersicht zeigt die möglichen Darstellungen der logischen Verknüpfungen. Die AWL oben wurde mit Variablen geschrieben gemäß IEC 1131-3, unten jeweils die Schreibweise von Step7 mit Symbolen.

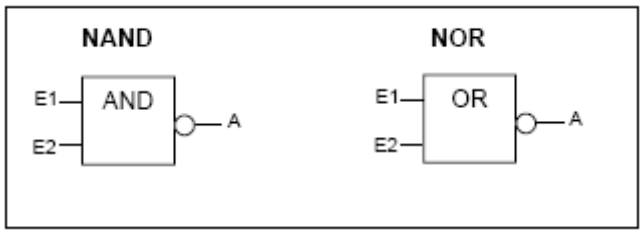




Ein Beispiel der **Booleschen Algebra** von besonderem Interesse ist das **DeMorgan-Theorem**:
Danach liefert die Negation eines Terms das gleiche Ergebnis wie die alternative Verknüpfung der einzeln negierten Signale. Mit dem Theorem kann man beispielsweise eine UND-Verknüpfung gegen eine ODER-Verknüpfung austauschen



Das Beispiel des DeMorgan Theorems macht deutlich, dass Boolesche Algebra durchaus zur Einsparung von Schaltgliedern führen kann und in bestimmten Fällen Schließerkontakte gegen Öffnerkontakte getauscht werden können.
Die hierbei benutzten Glieder **NAND und NOR** waren Basis verschiedener Transistor-Logik-Familien und galten deshalb historisch ebenfalls als logische Grundfunktionen



Nachfolgender Auszug der AWL Referenzliste von CoDeSys zeigt die IEC-Operatoren für Logische Verknüpfungen:

LD	Lädt den Operanden in den Akkumulator.
LDN	Lädt den negierten Wert des Operanden in den Akkumulator.
ST	Speichert den Inhalt des Akkumulators in die Variable, die als Operand verwendet wird.
STN	Speichert den negierten Inhalt des Akkumulators in die Variable, die als Operand verwendet wird.
AND	Bitweises AND von Akkumulator und Operand
ANDN	Bitweises AND von Akkumulator und negiertem Operand
OR	Bitweises OR von Akkumulator und Operand
ORN	Bitweises OR von Akkumulator und negiertem Operand
XOR	Bitweises exklusives OR von Akkumulator und Operand
XORN	Bitweises exklusives OR von Akkumulator und negiertem Operand
NOT	Bitweise Negation des Akkumulatorinhalts

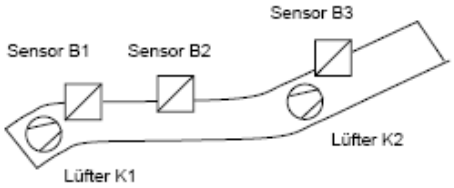
Das Programmieren binärer Logik erfolgt nach folgendem Grundschema:
1. Laden der ersten Variablen (des ersten Operanden) in den Akkumulator mit den Operationen LD oder LDN (bei Step7 mit U oder UN). Dieser Schritt wird auch als Erstabfrage bezeichnet.

2. Nach jeder logischen Verknüpfung steht das Ergebnis wieder im Akkumulator.

3. Ausgabe des Akkumulatorinhaltes in die gewünschte Variable (den gewünschten Operanden) mit der Zuweisung (=) bei Step7.

5.3 Lösungsverfahren für Schaltnetze
Ein bekanntes Verfahren zur Lösung von Aufgaben mit Schaltnetzen ist die empirische Aufstellung der Schaltgleichung aus einer Schalttabelle. Dies demonstriert nachfolgendes Standardbeispiel:

Ein Tunnel wird über zwei Lüfter versorgt, welche von drei Sensoren gesteuert werden. Diese messen die Luftqualität und sprechen bei Überschreitung von Grenzwerten mit Signal FALSE (!) an

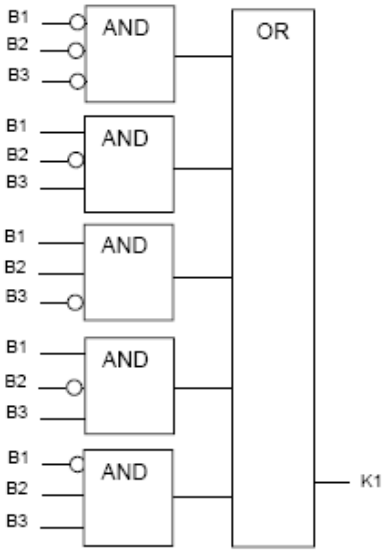


Theoretisch müssen bei Aufstellung der Schalttabelle die möglichen 8 Kombinationen der drei Sensoren betrachtet werden. Mitunter kann man aber empirisch die wenigen Kombinationen finden, bei denen überhaupt die Lüfter laufen müssen.

B3	B2	B1	K1	K2	Zeile
0	0	0	1	1	0
0	0	1	0	1	1
0	1	0	1	1	2
0	1	1	1	0	3
1	0	0	0	1	4
1	0	1	1	1	5
1	1	0	1	0	6
1	1	1	0	0	7

Schalttabelle für Lüfter K1 und K2

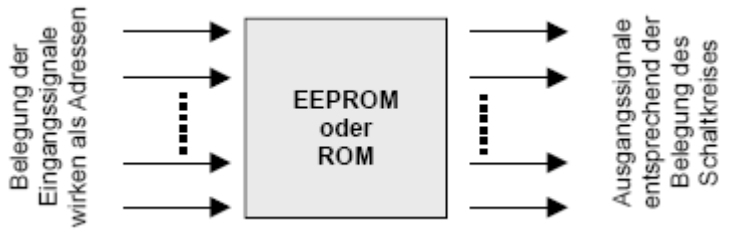
Liegt die Belegung der Ausgangssignale in der Schalttabelle fest, wird z.B. die **ODER-Normalform (Disjunktive Normalform)** der Ausgangssignale aufgeschrieben.



Dazu sind alle Zeilen, in denen das Ausgangssignal den Wert 1 (TRUE) aufweist, über ODER-Logik verbunden. An die einzelnen ODER-Glieder werden die Eingangssignale mit ihren Tatsächlichen Belegungen über UND-Verknüpfungen angeschaltet.

Daneben könnte man auch eine UNDNormalforn (Konjunktive Normalform)nutzen.
An dieser Stelle nun könnte ein Minimierungsverfahren angesetzt werden, was sich aber bei Realisierung mit einem SPS-Programm kaum mehr lohnt.

Größere **Schaltnetze können mit Speicherschaltkreisen realisiert** werden.
Die Kombination der möglichen Eingangssignale wird als Adresse interpretiert, und der Schaltkreis
gibt die gewünschte Kombination von Ausgangssignalen aus.



Beispiel: Die Belegung der Sensoren 2#1 0 0 von Zeile 4 der Schalttabelle müsste als Adresse
für eine Speicherzelle dienen, von der dann die Belegung 2# 0 1 für die Lüfter K1 und K2
ausgegeben wird. Es ist ersichtlich, dass sich dieses Verfahren nur bei Schaltnetzen mit einer
Vielzahl von Eingangs- und Ausgangssignalen loht.

5.4 Speicherfunktionen

Um Schaltwerke (sequentielle Netzwerke) zu realisieren, benötigt man zusätzlich zu den logischen Funktionen auch Speicherelemente. In der Technik werden dazu **R-S-Flip-Flop** verwendet. Diese werden auch als bi- stabile Funktionsblöcke bezeichnet.

Das Ein- und Ausschalten des Speicherelementes erfolgt durch Setzen und Rücksetzen. Unabhängig vom Setzsignal bleibt ein einmal gesetzter Ausgang solange auf Wert TRUE, bis er zurückgesetzt wird.

Auch Speicher wurden ursprünglich durch Kontaktschaltungen realisiert. Das entscheidende Element ist hierbei der **Selbsthaltekontakt**, welcher eine **Selbsthaltung** der Relaisspule ermöglicht. Wenn des Setz- und das Rücksetzsignal gleichzeitig anstehen, entscheidet die Art der Schaltung, ob **dominierend gesetzt** oder **dominierend zurückgesetzt** wird. Mehr als 95% aller Anwendungsfälle der Automatisierungstechnik verlangen ein dominierendes AUS (Rücksetzen).
In der aktuellen SPS-Technik gibt es für die Realisierung von Speicherfunktionen die Operationen Setzen (S) und Rücksetzen (R).

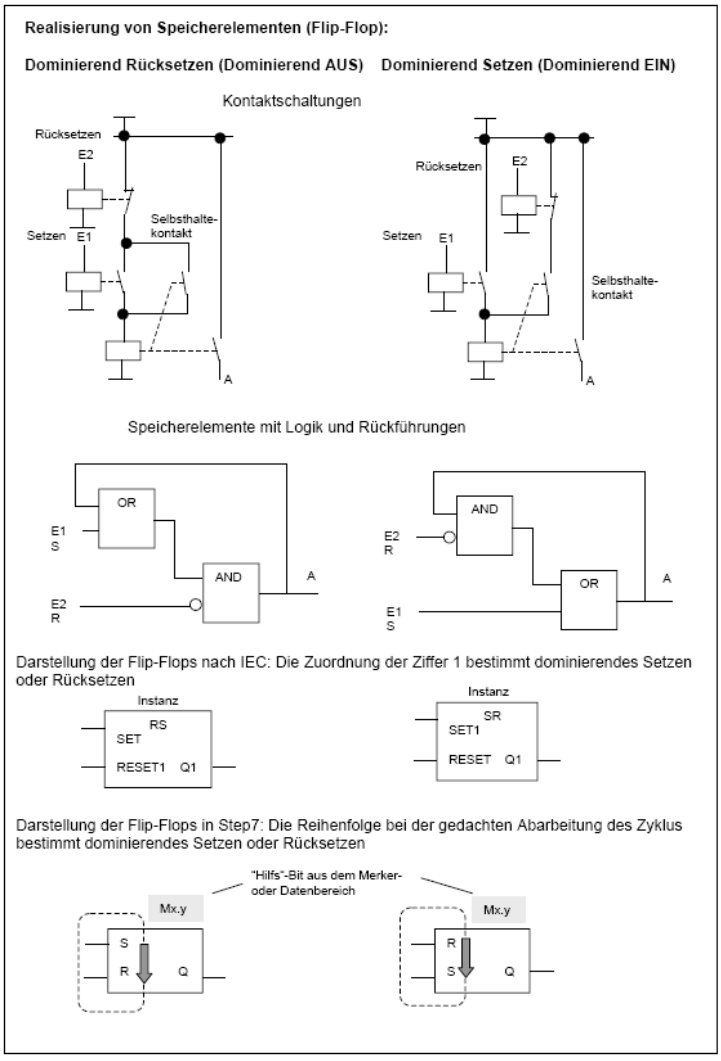
S Setzt den Operanden (Typ BOOL) auf TRUE, wenn der Akkumulatorinhalt TRUE ist.
R Setzt den Operanden (Typ BOOL) auf FALSE, wenn der Akkumulatorinhalt FALSE ist.

Je nachdem, ob Setzen vor oder nach dem Rücksetzen programmiert wird, entstehen durch Wirkung der zyklischen und seriellen Programmbearbeitung die dominierenden Eigenschaften!

Hinweis: **Drahtbruchsichere Programmierung:**
(hierzu Übersicht Realisierung von Speicherelementen auf nachfolgender Seite)
„Drahtbruchsichere Programmierung“ bedeutet Vorkehrungen zu schaffen, dass
• ein unterbrochener Leiter zwischen Aus-Befehlsgeber und SPS einen Aus-Befehl bewirkt
• ein unterbrochener Leiter zwischen Ein-Befehlsgeber und SPS einen Ein-Befehl verhindert.

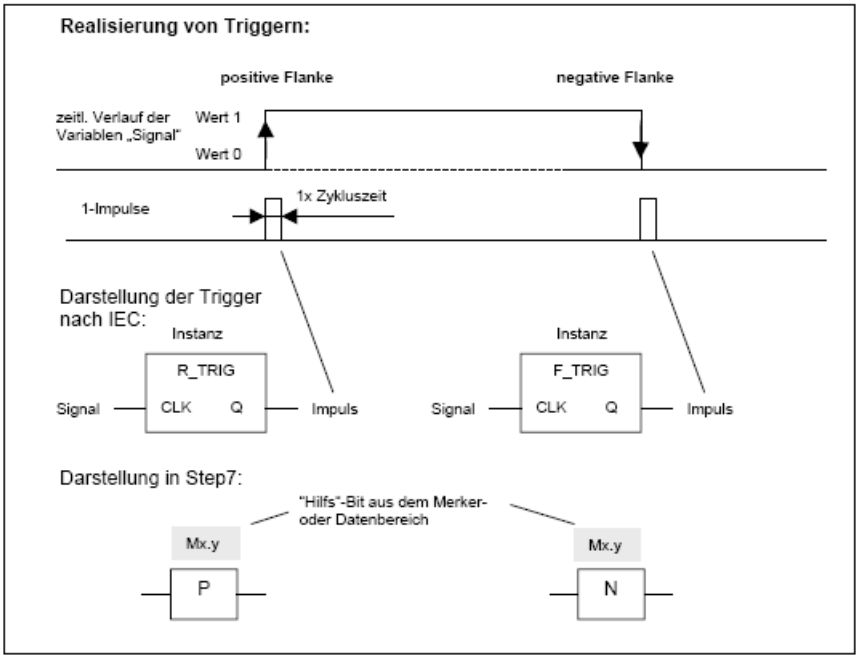
Nach VDE-Vorschriften muss deshalb
• ein Einschaltbefehl durch **Schalten eines Arbeitsstromes** gegeben werden

• ein **Ausschaltbefehl durch Unterbrechung eines Ruhestroms** gegeben werden, d.h. durch FALSE-Signal. In der Praxis erfolgt dies überwiegend durch **Betätigung eines Tasters mit Öffnerkontakt.**
Das Rücksetzen erfolgt nur bei Wert TRUE am Eingang RESET (bzw. bei Step7 mit Verknüpfungsergebnis VKE=1 am Eingang R. Deshalb muss in den allermeisten Fällen das **Rücksetzsignal über eine Negation angeschaltet** werden!



5.5 Triggerfunktionen

Zu den Bitoperationen gehören weiter die Triggerfunktionen, auch als **Flankenauswertung** bezeichnet. Sie werden dann benötigt, wenn nicht das Signal an sich, sondern nur die **Signaländerung** eine Operation auslösen soll. Sie werden weiter eingesetzt, wenn dauerhaft anstehende Signale die Funktion eines Programms stören.
Das Ausgangssignal der Trigger ist jeweils ein **Impuls von Zykluszeit**. Er kann genutzt werden, um z.B. Variablen zu setzen oder zurückzusetzen. Der Impuls wird mitunter auch als **1-Impuls** bezeichnet, weil er nur einen Programmzyklus lang zur Verfügung steht.



5.6 Hinweise

Während es für Schaltnetze brauchbare Entwurfsverfahren gibt, ist man bei Schaltwerken häufig auf Intuition und Erfahrung angewiesen. Einige Empfehlungen zum Entwurf sollen hier angeführt werden. Es kann erfolgreich sein, den Entwurf von den Ausgangssignalen hin zu den Eingangssignalen (de facto „rückwärts denkend") zu gestalten:

- Welche Ausgangssignale sind zu steuern?
- Müssen – da die Eingangssignale nur kurzzeitig anstehen - RS-Speicher eingesetzt werden oder stehen die Signale z.B. durch rastende Schalter dauerhaft zur Verfügung. Im letzteren Fall genügen logische Verknüpfungen (führt zu Schaltnetzen).
- Erforderliche Anzahl Speicher anlegen und Ausgangssignale anschalten
- Welche Signale setzen einen Speicher? Sind dies mehrere Signale, dann sind diese zumeist UND- verknüpft anzuschalten!
- Welche Signale setzen Speicher zurück? Sind dies mehrere Signale, dann sind diese zumeist ODER- verknüpft anzuschalten!