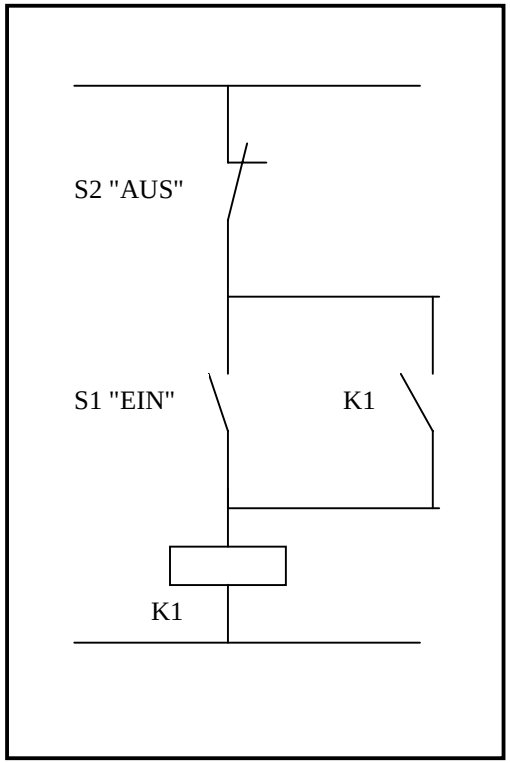
SPS
Einführung 1
(systemunabhängig)

Prinzip der SPS-Programmierung:

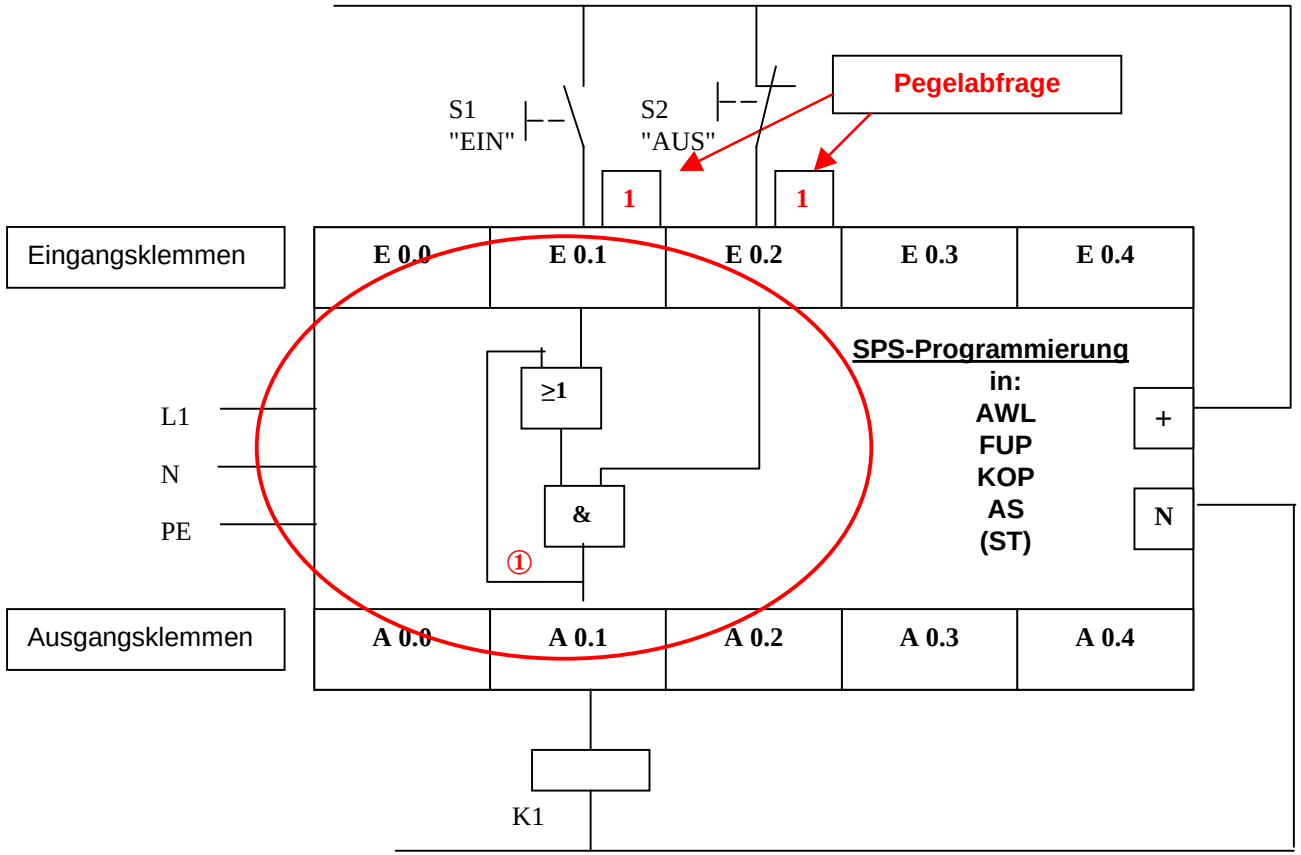


Alle Befehlsorgane (z.B. Taster, Sensoren, Relais, Lastschütze, Ventilspulen, etc.) werden jeweils **einzeln** an Ein- und Ausgangsklemmen angeklemmt.
Die Verknüpfungslogik wird also nicht mehr durch die Verdrahtung erzeugt, sondern dafür wird das Programm der SPS geschrieben. Das Programm fragt die Spannungspegel an den entsprechenden Ein- und Ausgangsklemmen ab.
Der Programmierer muss entscheiden,

1) Welches Befehlsorgan an welche Klemme gelegt werden soll.
2) Welche Befehlsorgane bei der Abfrage betätigt werden sollen.
3) Welche nicht betätigten Befehlsorgane auch noch abgefragt werden müssen.
4) Welche Spannungspegel zum Einschalten des gewünschten Ausganges vorhanden sein sollen.

Siehe nächste Seite

Die Verknüpfungslogik kann aus der Schützensteuerung übernommen werden. Dabei ist auf die **logisch richtige Reihenfolge** (siehe weiter unten) zu achten. Im obigen Stromlaufplan darf nur S1 betätigt werden, damit der Stromkreis für K1 geschlossen wird. Dem entsprechend muss an der SPS auch der Taster S1 gedrückt werden >> an der Klemme E 0.1 liegt dann eine logische 1 an. Würde oben S2 betätigt, wäre der Stromkreis unterbrochen. Sinngemäß darf S2 (Öffner) an der SPS auch nicht betätigt werden. Dann liegt im Ruhezustand an der Klemme E 0.2 ebenfalls eine logische 1 an. Auch Ausgangsklemmen können abgefragt werden. ①



vergleiche vorstehende Schaltung

Die älteren und bekannteren Programmiersprachen für SPS sind

	AWL	**A**N**W**EISUNGS**L**ISTE
	FUP	**FU**NKTIONS**P**LAN
und	KOP	**KO**NTAKT**P**LAN

AWL ist die "eigentliche" Programmiersprache, die der Rechner versteht, d.h., das Programm wird in Textform zeilenweise geschrieben und gelesen.
FUP und KOP sind grafische Makromasken, die im Hintergrund auch wieder in AWL-Befehle übersetzt werden. Sie erleichtern die Lesbarkeit des Programms und vereinfachen die Befehlseingabe, vor allem bei aufwendigeren Bausteinen wie z. B. Zähler und Timern.

Praktisch ist es, wenn es die software-Routine erlaubt, durch Mausklick von einer Sprache in die andere zu wechseln. Dabei ist schön zu sehen, wie z. B. ein Zählerbaustein im FUP von der software in eine AWL-Anweisung umgeschrieben wird. In der AWL wird der vollständige Baustein mit allen möglichen Ein- und Ausgängen dargestellt, auch wenn man nicht alle Anschlussmöglichkeiten im FUP genutzt, bzw. benötigt hat. Die nicht belegten Ein- u. Ausgänge tauchen dann in der entsprechenden Zeile als NOP-Befehl (no operation) auf. Man kann diese Zeilen in AWL auch löschen, das Programm arbeitet trotzdem einwandfrei. Allerdings ist dann dieses Rumpfprogramm nicht nach FUP oder KOP zu übersetzen. Warum also sollte man sich die Mühe machen, und die korrekte zeilenweise Struktur in AWL schreiben, wenn der ganze Baustein auch mit Mausklick entstehen kann?
Das Ganze ist mehr eine "Geschmacks-" oder Erfahrungssache, je nachdem, welche Darstellungsart man gewöhnt ist. Der Nachrichtentenchniker wird wohl auf FUP schwören und der "Steinzeitelektriker" aus der Welt der Schützentechnik könnte sich wahrscheinlich eher mit KOP anfreunden.
Es gibt aber noch ein anderes Problem, das je nach Programmiersprache unterschiedlich groß wird: Angenommen, eine etwas aufwendigere Schützkontaktverriegelung mit Reihen- u. Parallelschaltungen der Kontakte soll durch eine SPS ersetzt werden.

Weiter: nächste Seite

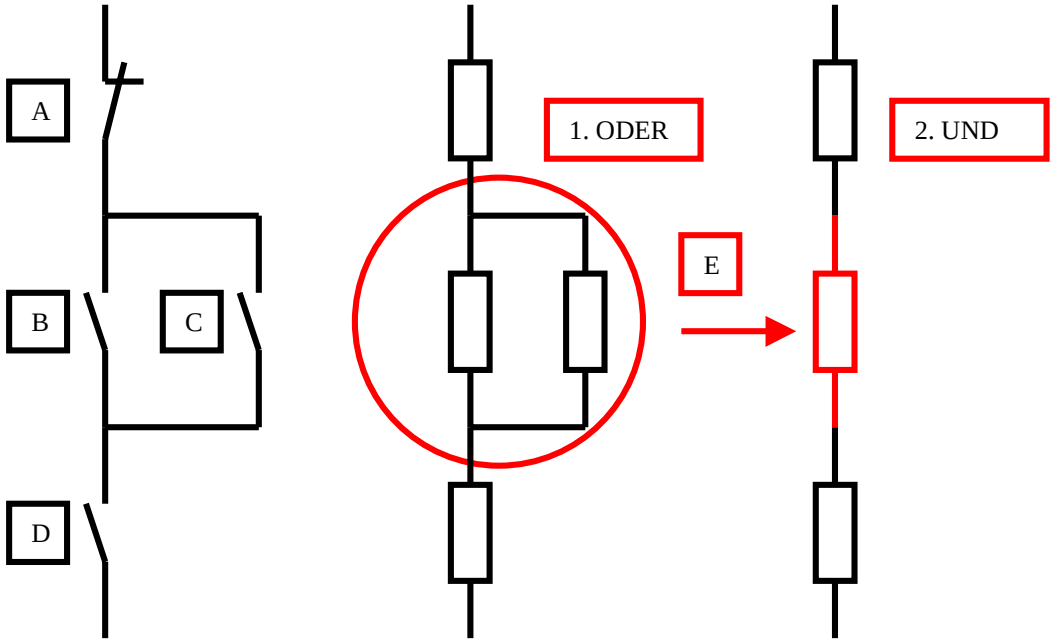
Die Kernfrage ist: **Bei welchen Kontakten beginne ich mit dem SPS-Netzwerk?**

Beginne ich wie bei der Schützensteuerung oben und arbeite mich Kontakt für Kontakt nach unten?

Bevor es in AWL die Möglichkeit der Klammerbildung gab, ging das nicht. Genausowenig wie beim FUP.
Der KOP heißt nicht von ungefähr so, sondern hier werden genau die Kontakte der Schützensteuerung in gleicher Anordnung nachgebildet, nur von links nach rechts und nicht von oben nach unten.
In AWL ist durch die heutige Klammerbildung auch die gleiche Reihenfolge nachzubilden.
Würde man das selbe aber im sonst so übersichtlichen FUP machen, führe dieser so beliebte Fehler zum Funktionschaos. Denn:

entscheidend ist die <u>logisch</u> richtige Reihenfolge.

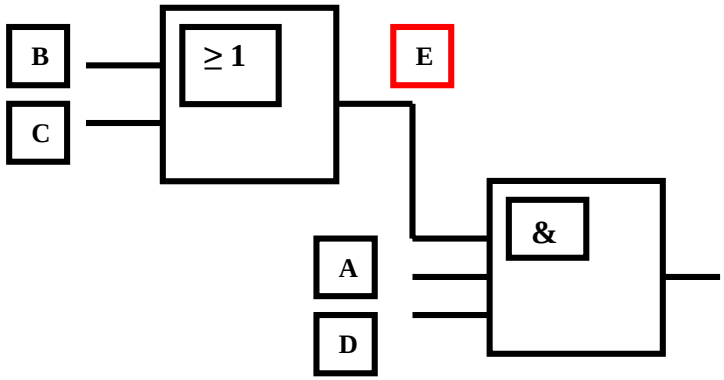
Reihenfolge der Verknüpfungen



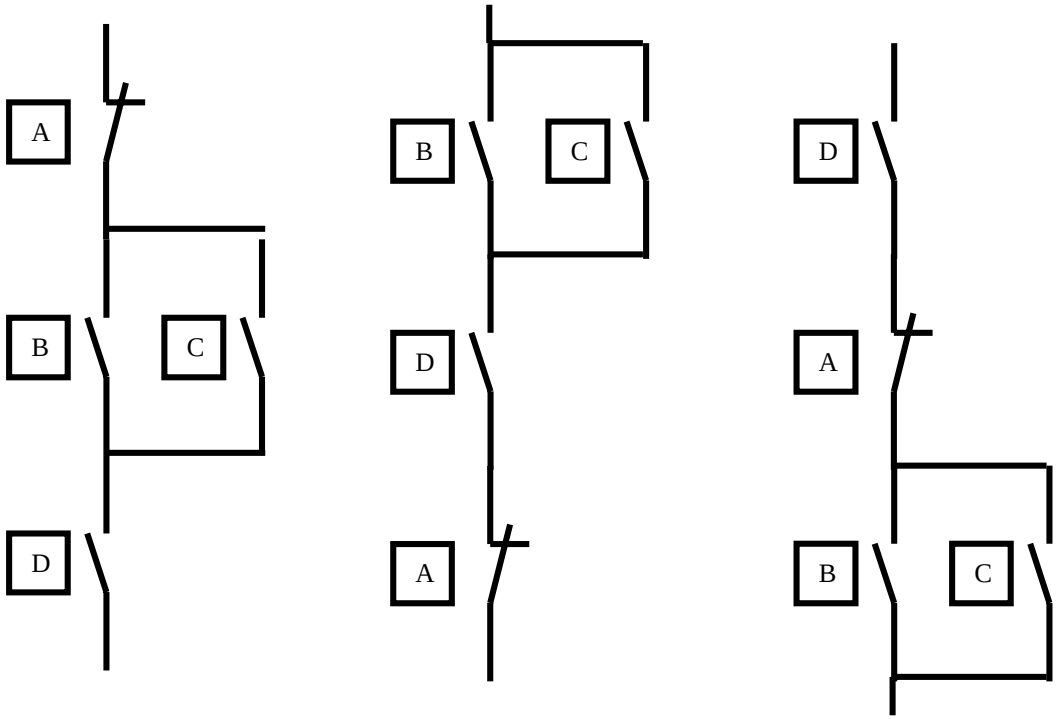
Zuerst wird die Parallelschaltung (logisch ein ODER) zu einem Ersatzwiderstand zusammengefasst. Dadurch entsteht eine einfache Reihenschaltung (logisch ein UND). Als Hauptaufgabe bleibt also die UND-Verknüpfung der Variablen A, D und E. Woraus sich das Verknüpfungsergebnis (VKE) E ergibt, ist untergeordnet.

Siehe nächste Seite

Umsetzung als digitale Verknüpfung - FUP



Logisch gleichwertige Schaltungen



Im FUP sieht das Programm für alle Schaltungen gleich aus:

"ODER vor UND"

Etwas umständlicher könnte man natürlich auch alle möglichen Einzelpfade ODER-verknüpfen.
Also: A UND B UND D **ODER** A UND C UND D.
Bei größeren Anlagen würde man dabei aber schnell den Überblick verlieren

Für die Übernahme der Logik gilt:

In der Schützensteuerung muss der (ein) Stromkreis geschlossen sein, damit das Schütz anziehen kann. Dabei ist es dem Strom egal, ob er über einen nicht betätigten Öffner fließt, oder über einen betätigten Schließer.

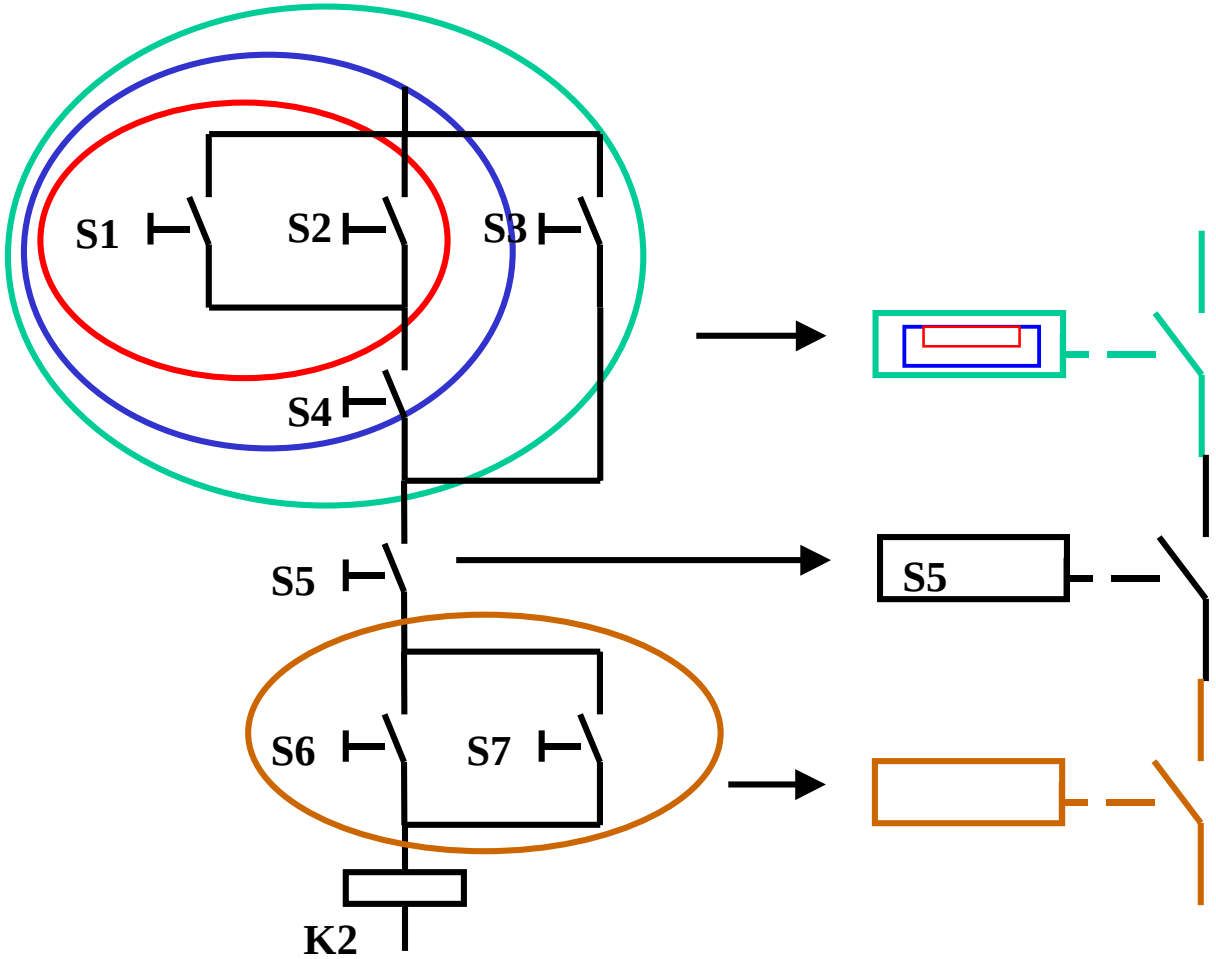
Das SPS-Programm wird für die Setzbedingung geschrieben, d.h., wenn die entsprechenden Eingangs- und evtl. auch Ausgangsbedingungen erfüllt sind, führt der Ausgang 1-Pegel.

Also ist zu überlegen,

> **welche Taster**, etc. sind an welchen SPS-Klemmen anzuschließen
> **welche Klemmen** sind abzufragen
> **welche Pegel** sollen die entsprechenden Klemmen als Setzbedingung führen, 1 oder 0?

Merke: Die "kurzsichtige" SPS kann nur 0 / 1-Pegel an den Klemmen feststellen und kann keinen Millimeter darüber hinaus sehen!. Deswegen kann sie niemals zwischen einem Signal von einem Öffner oder von einem Schließer unterscheiden!! Der Programmierer hat nach Sicherheitskriterien zu entscheiden, ob ein Befehl über einen Schließer oder einen Öffner eingegeben werden soll. Wenn er das getan hat, liegt eigentlich schon fest, auf welche Pegel die SPS warten soll, damit die Anlage wunschgemäß einschaltet.

Reihenfolge der Verknüpfungen



vergleiche nächste Seite

Reihenfolge der Verknüpfungen