



Software für Industrie 4.0 (Vorlesung & Übung)

Speicherprogrammierbare Steuerungen (SPS)



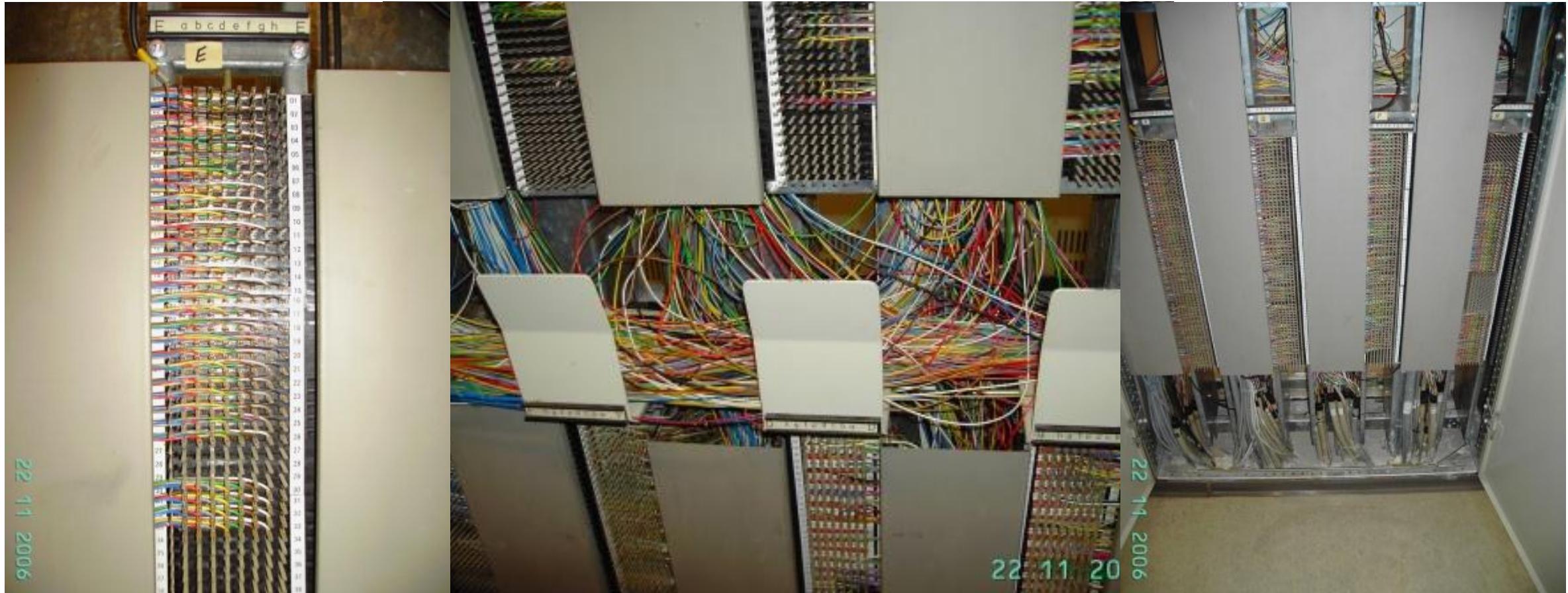
Rückblick: Verbindungsprogrammierte Steuerung

- Früher wurden Automatisierungsaufgaben mit **Verbindungsprogrammierten Steuerungen (VPS)** realisiert
- Diese Technik wurde auf allen Steuerungsebenen eingesetzt, also von der Leit- bis zur Zellebene
- In einfachen Maschinen ist diese Technik auch heute noch im Einsatz
- Hierbei werden Sensoren und Aktoren festverdrahtet, eine Auswertung von Signalen erfolgt hierbei meist über Relais, Schütze, Timer, etc.
- Bei komplexeren Auswertungsanforderungen kommen teilweise auch programmierte Logikbausteine zum Einsatz, welche aber wiederum nur eine begrenzte Anzahl an semantisch festgelegten Ein- und Ausgängen haben



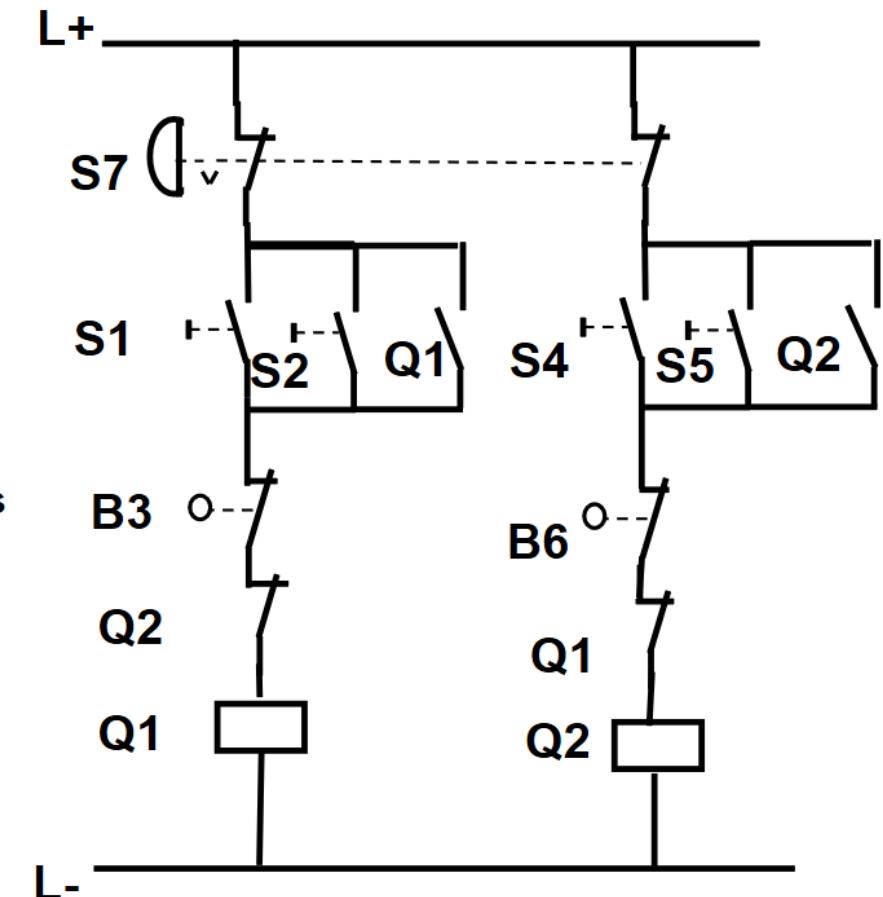
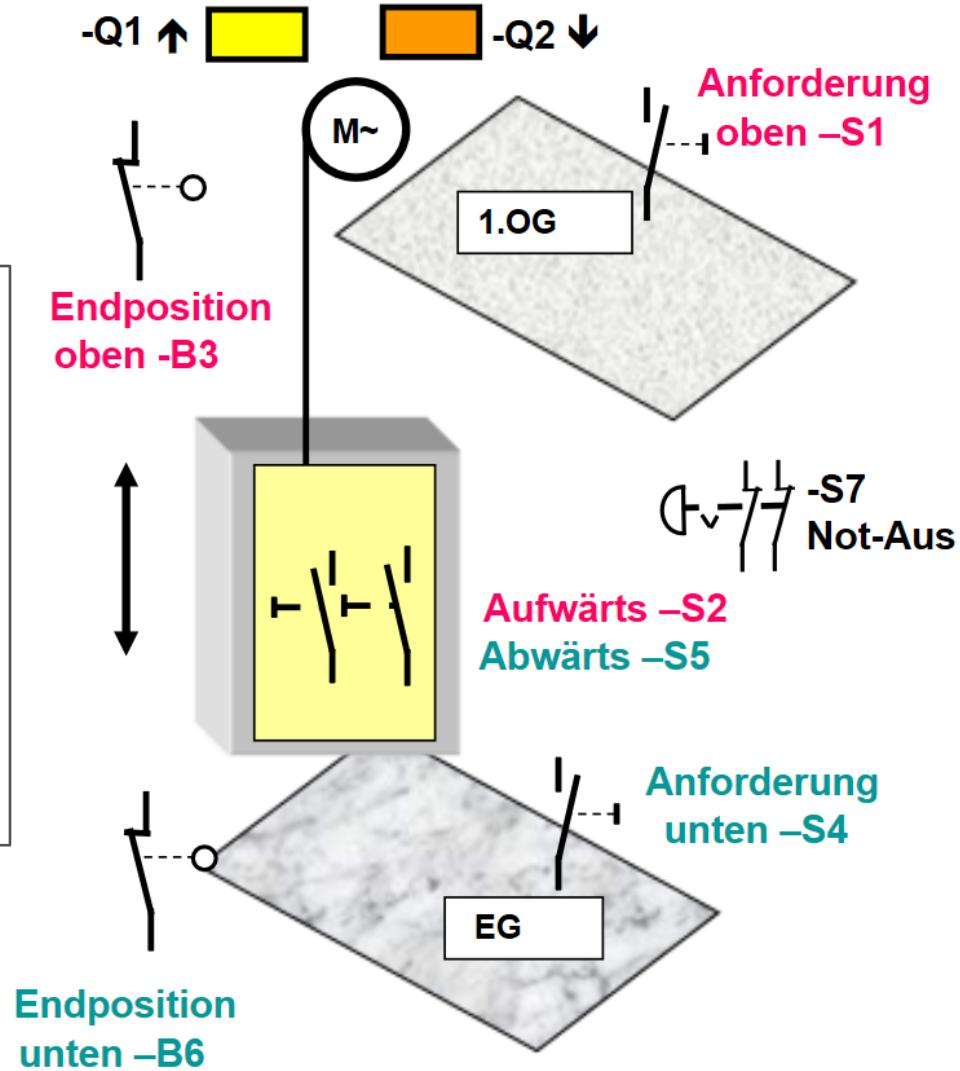
Rückblick: Verbindungsprogrammierte Steuerung

Da der Schaltschrank jedoch zentral gelegen sein soll wird ein hoher Verkabelungsaufwand erforderlich

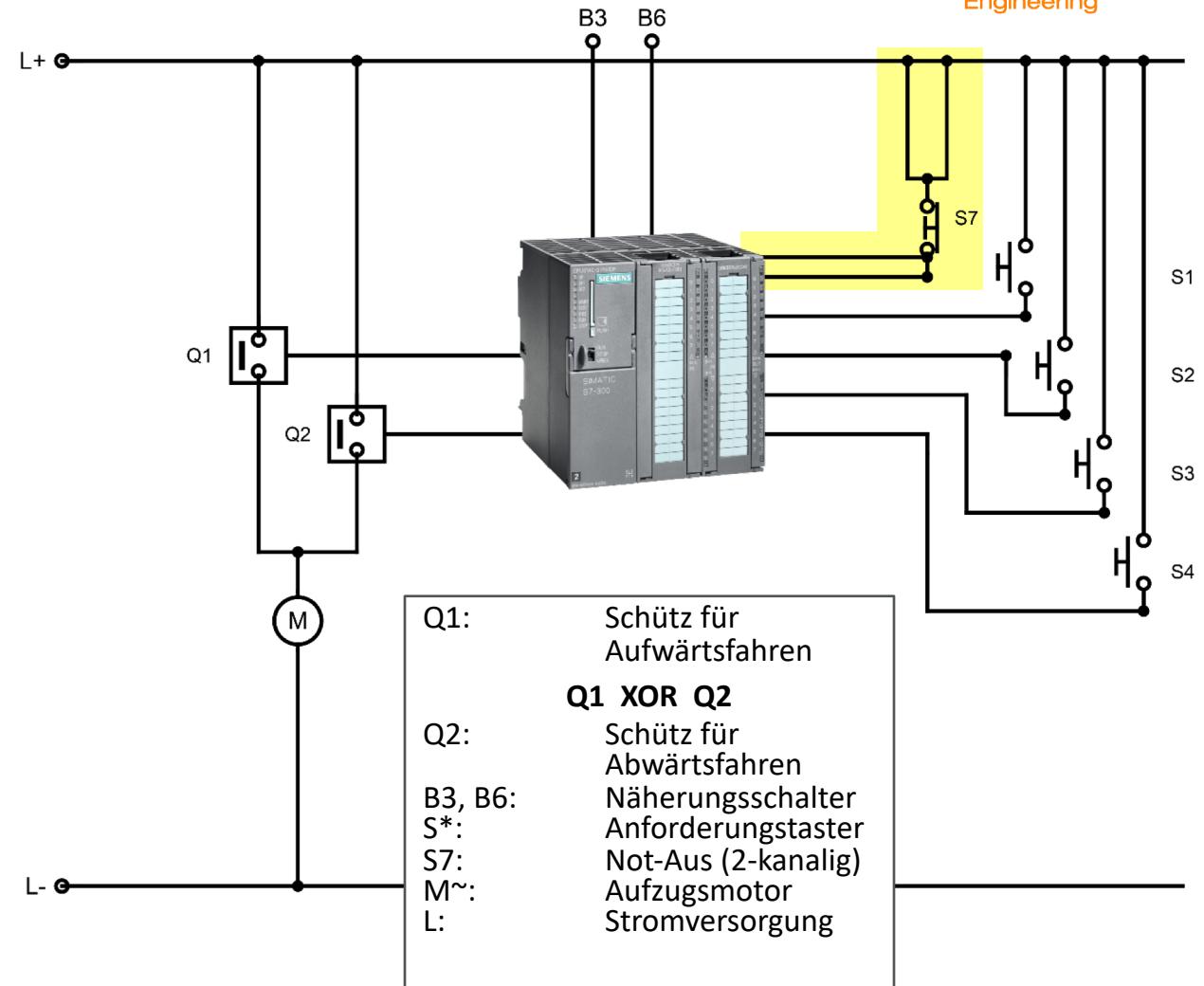
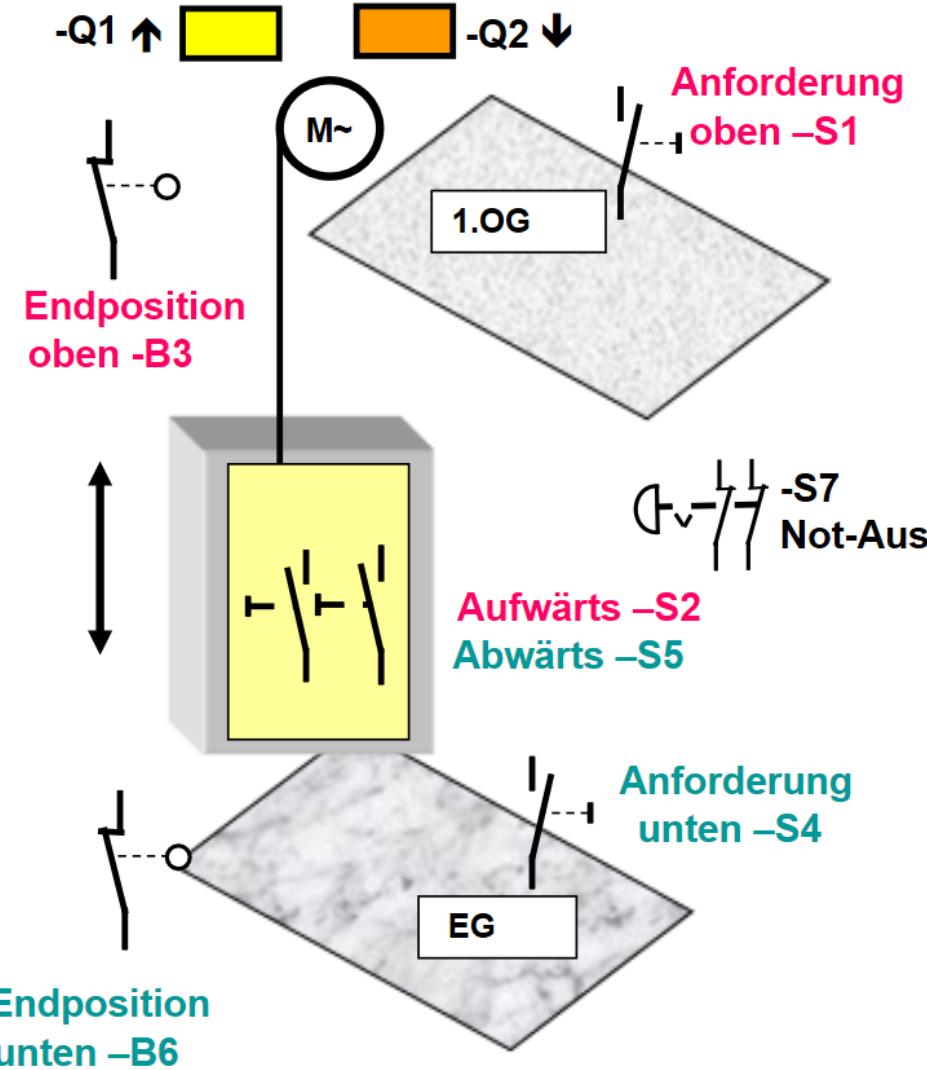


Beispiel Aufzugsteuerung - VPS

Q1:	Schütz für Aufwärtsfahren
Q1 XOR Q2	
Q2:	Schütz für Abwärtsfahren
B3, B6:	Näherungsschalter
S*:	Anforderungstaster
S7:	Not-Aus (2-kanalig)
M~:	Aufzugsmotor
L:	Stromversorgung



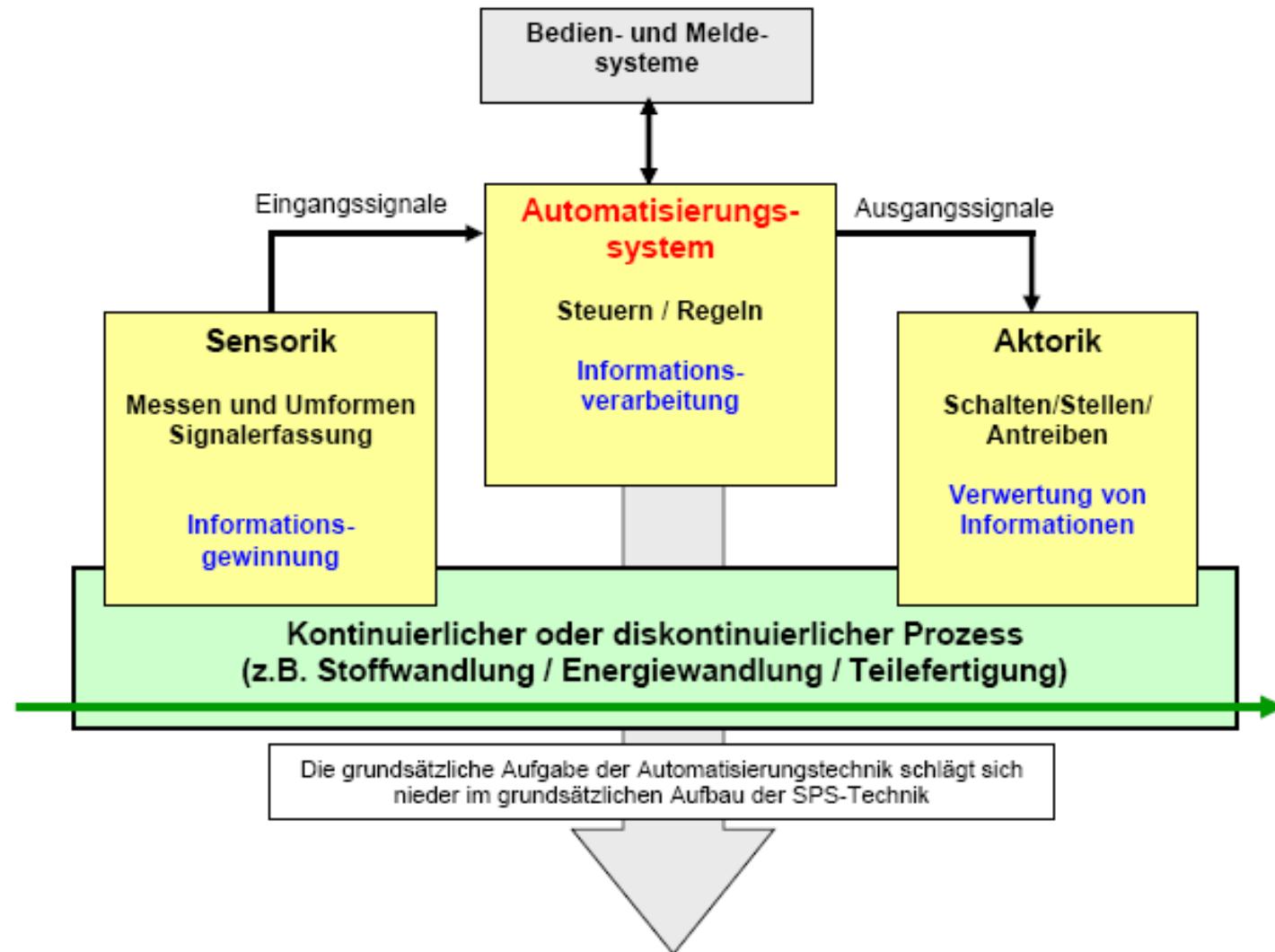
Beispiel Aufzugsteuerung - Lösung mit einer SPS



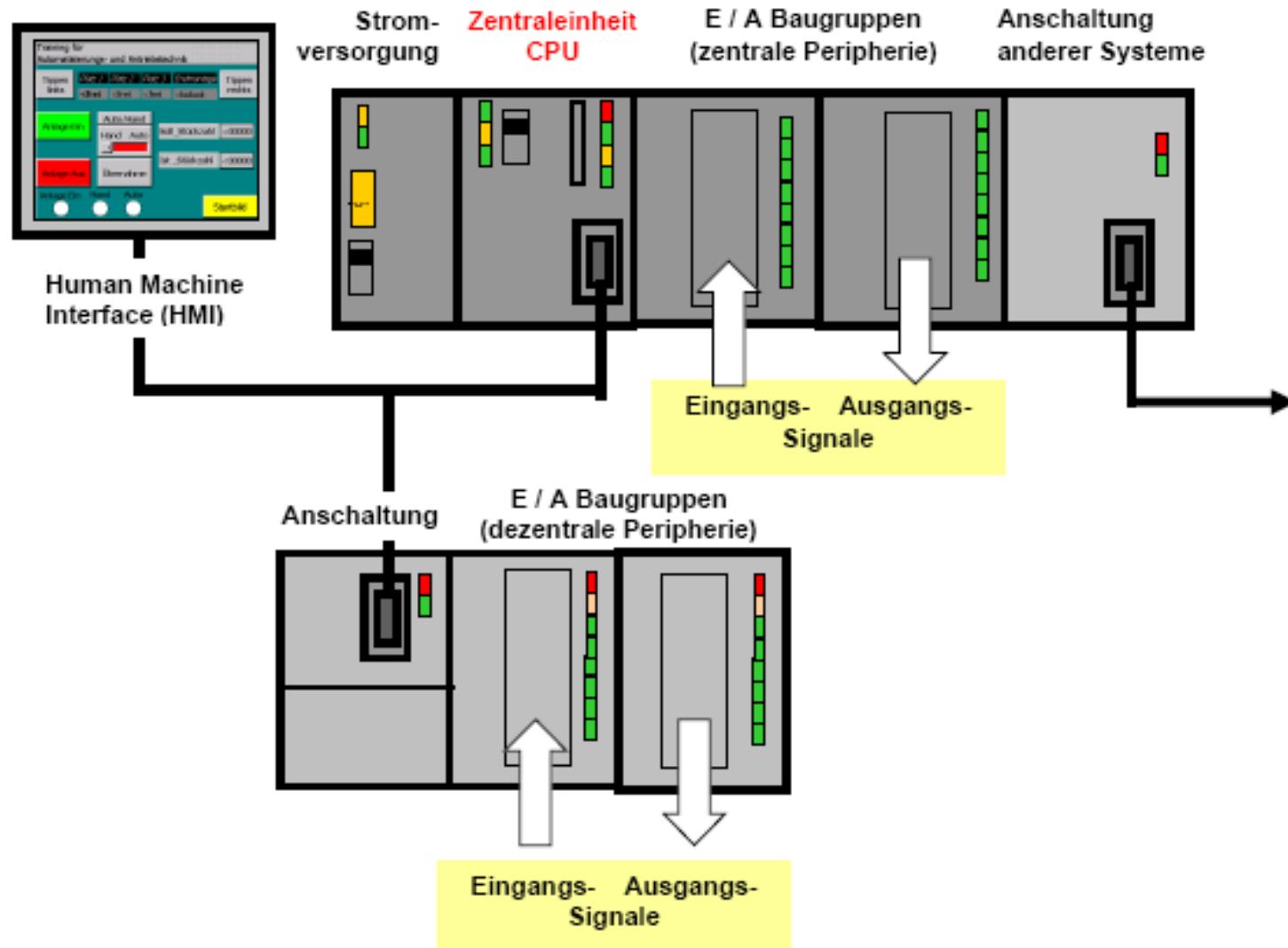
- Eine **SpeicherProgrammierbare Steuerung** (engl. PLC: Programmable Logic Controller) unterscheidet sich von der VPS in ihrer Flexibilität durch die Definition von „Schaltungen“ in Software
- Auch die Möglichkeit von Fehlerdiagnose & Fernwartung, geringerer Kostenaufwand und höhere Zuverlässigkeit gegenüber VPS waren Gründe für die Durchsetzung der SPS-Technologie
- Zudem bieten sie die Möglichkeit der Vernetzung mit anderen Systemen über Feldbusse



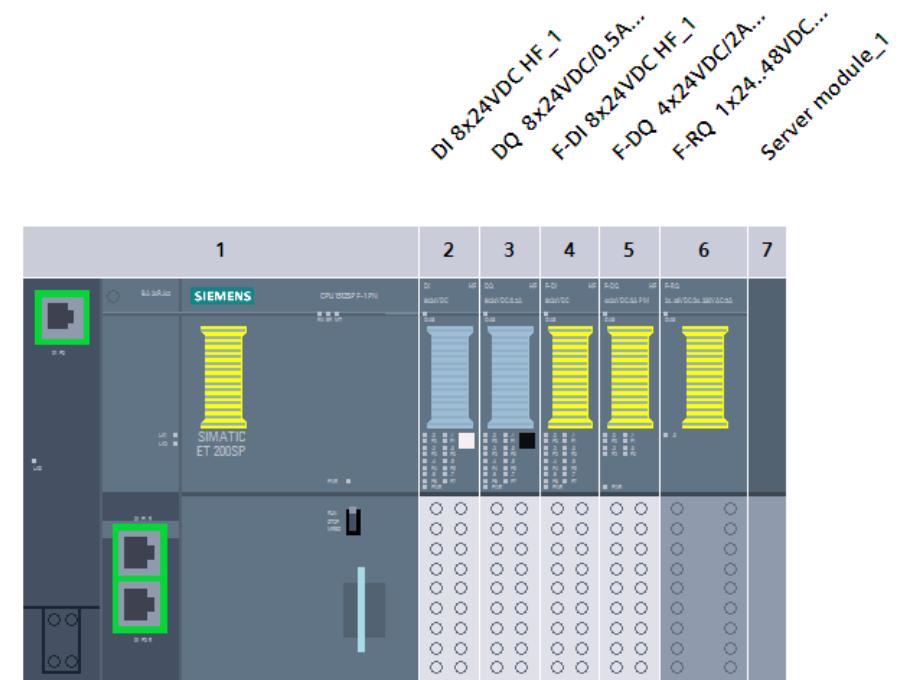
Logischer Aufbau einer SPS...



...und die Umsetzung in Hardware



- SPSen sind zumeist in Modulbauweise ausgeführt und daher einfach an die jeweiligen Bedürfnisse anpassbar.
- Solche Module sind z.B.:
 - Hauptmodul mit Prozessor (**sicher** / nicht sicher)
 - Digitale Eingänge / Ausgänge (**sicher** / nicht sicher)
 - Analoge Eingänge / Ausgänge
 - Relaisausgänge (direkte Schaltung von Lasten)
 - Busmodule (Feldbussysteme)



ET200SP mit EA-Modulen

- Des Weiteren gibt es **Kompaktausführungen**, welche als dezentrale Peripherie per Bus an eine Haupt-SPS angeschlossen werden können

- Außerdem existieren spezielle Modelle für den Einsatz in **extremen Umgebungen**, welche in abgedichteten Gehäusen und mit den benötigten Anschlüssen ausgeliefert werden

- **Eigensicherheit** einer SPS bedeutet, dass sie grundsätzlich nur mit begrenzten Strömen arbeitet (4-20 mA), sodass keine Zündfunken entstehen können. Diese sind beispielsweise für den Einsatz in der Umgebung leicht entzündlicher Chemikalien vorgesehen.

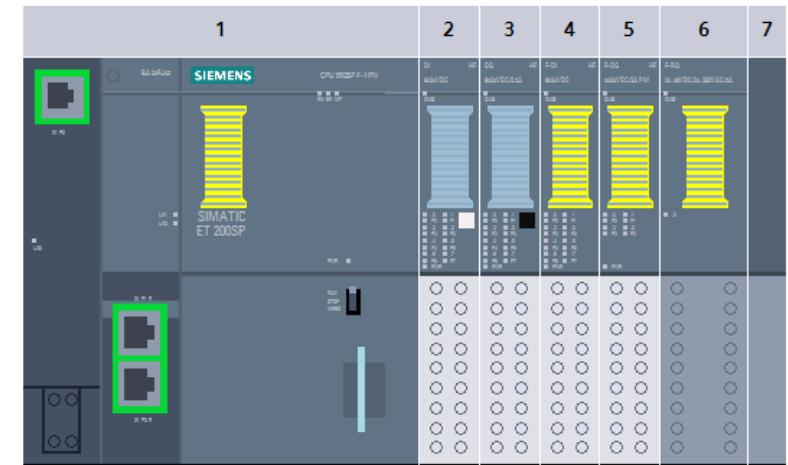


S7-300 CPU 312C mit 10E und 6A integriert

SPS: Hardware-Module – Beispiel Siemens ET200

- 1) Hauptmodul mit sicherem Prozessor
- 2) Nicht-sicheres, digitales Eingangsmodul
- 3) Nicht-sicheres, digitales Ausgangsmodul
- 4) Sicherer, digitales Eingangsmodul
- 5) Sicherer, digitales Ausgangsmodul
- 6) Sicherer, Relais-Ausgangsmodul

DI 8x24VDC HF-1
DO 8x24VDC 0.5A...
F-DI 8x24VDC HF-1
F-DO 4x24VDC/2A...
F-RQ 1x24.48VDC...
Server module_1



- **Zyklische Ausführung**

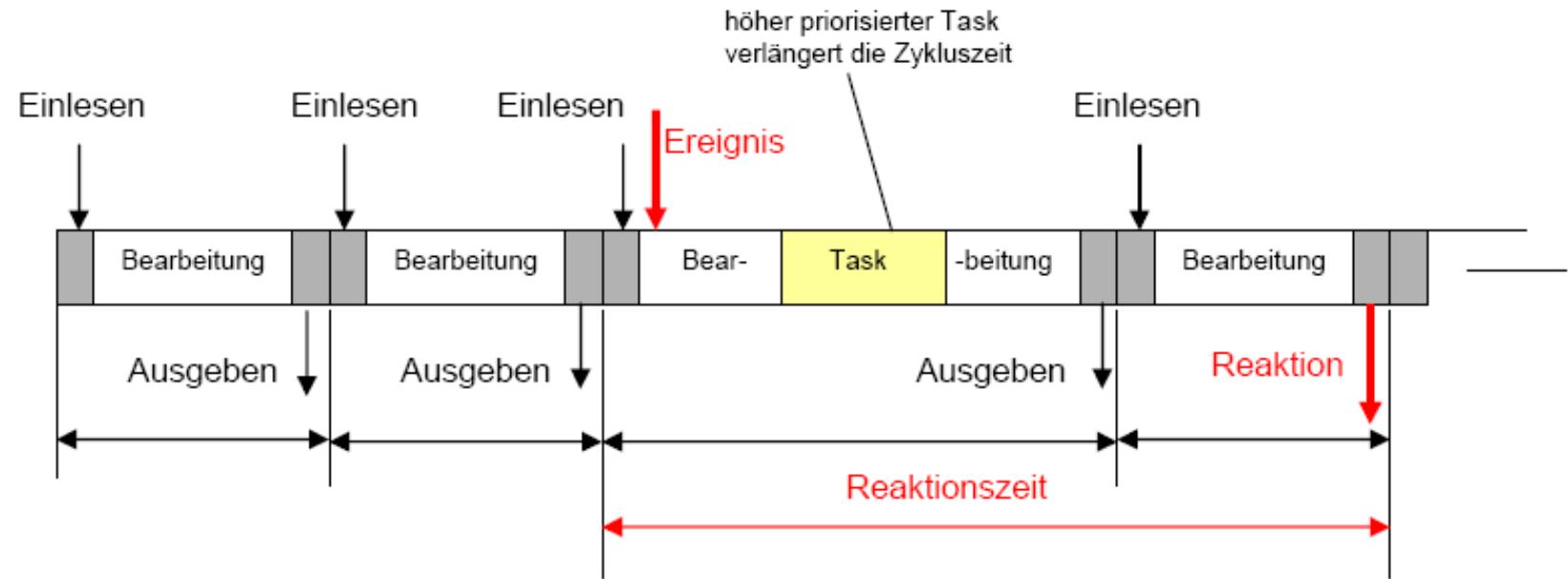
Dies ist die häufigste Variante. Bei der zyklischen Ausführung werden vom Betriebssystem pro Taktzyklus zunächst alle Eingänge eingelesen, dann das Anwenderprogramm ausgeführt und zum Schluss alle Ausgänge geschrieben (Im Schreiben und Lesen von EA-Daten sind Prozessabbilder von Bussystemen mit eingeschlossen). Dieser Prozess wird in einer vorgegebenen Taktzeit wiederholt.

Dies wird auch als **EVA-Prinzip** bezeichnet:

Eingänge lesen - Daten Verarbeiten - Ausgänge schreiben

- **Zyklische Ausführung mit Unterbrechungsverwaltung**

Hierbei kommt noch die spezielle Eigenschaft hinzu, dass das laufende, zyklische Programm durch eine Zustandsänderung eines Einganges unterbrochen werden kann. Damit kann auf höherpriore Ereignisse sofort reagiert werden. Bei Eintritt eines solchen Ereignisses wird eine zuvor definierte *Interrupt Service Routine (ISR)* aufgerufen.



- **Ereignisgesteuerte Ausführung**

Bei dieser Ausführung wird ausschließlich auf externe Ereignisse reagiert. Dies eignet sich vor allem für Anwendungen in denen nicht in einem festen, kurzen Taktzyklus Daten verarbeitet werden müssen. Durch eine rein anforderungsbasierte Ausführung der jeweiligen ISR-Methode können Ressourcen gespart werden.

- In den Standards ISO 13849, IEC 61508, etc. werden **funktionale Sicherheitsanforderungen** definiert.
- Diese spezifizieren, wie im laufenden Betrieb Fehler erkannt werden müssen und in welchen Zustand das System übergehen muss.
- Dabei obliegt es dem Sicherheitsinbetriebnehmer bzw. Programmierer, die Sicherheitsanforderungen und den **sicheren Zustand** korrekt zu spezifizieren und zu implementieren
- Dieser sichere Zustand muss z.B. bei Vorhandensein von bewegten Teilen den Stillstand dieser in der kürzest möglichen Zeit garantieren

- Vollständige Sicherheit kann niemals gewährleistet werden, insbesondere da der Mensch als Entwickler und Bediener immer als Risikofaktor einwirkt
- Über **Gefahrenabschätzungen** wird allerdings versucht die Risiken auf ein absolutes Minimum zu reduzieren
- Sollten bei dieser bisher nicht berücksichtigte Risiken aufgedeckt werden, oder durch einen Umbau der Anlage neue hinzukommen, muss die Spezifikation der Sicherheitssteuerung überarbeitet werden

Funktionale Sicherheit

- Nach den Standards werden verschiedene Stufen der garantierten Sicherheit in **Performance Level (PL a-d)** eingeteilt
- Durch einen Risikobaum kann das benötigte Performance Level für die jeweilige Anwendung bestimmt werden:

Schwere der Verletzungen

S1: leichte Verletzungen

S2: schwere Verletzungen

Häufigkeit und Aufenthaltsdauer

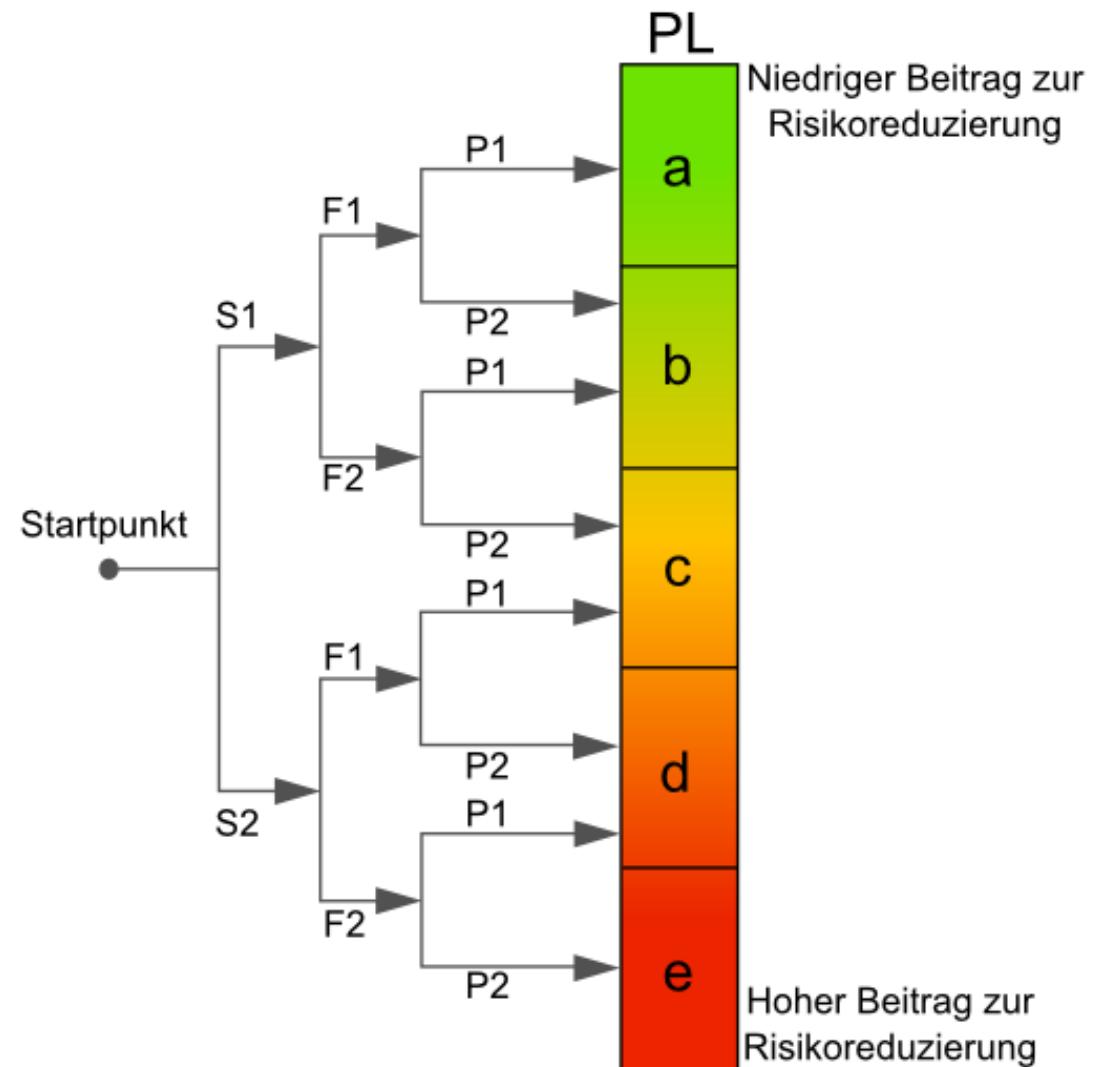
F1: selten bis öfter

F2: häufig bis dauernd

Möglichkeit zur Vermeidung der Gefährdung

P1: möglich unter bestimmten Bedingungen

P2: kaum möglich



Bei SPSen werden funktionale Sicherheitsanforderungen nach den Standards durch **zweikanalige Auswertung** realisiert:

- Im Hauptmodul wird diese durch eine doppelte Berechnung auf zwei voneinander getrennten Schwestерprozessoren und den anschließenden, gegenseitigen Vergleich der Ergebnisse realisiert.
- Erkennt einer der Prozessoren eine Abweichung, wird das System in den sicheren Zustand versetzt.

Außerdem gibt es die Möglichkeit sichere Ein- und Ausgabemodule zu verwenden:

- Hierbei werden digitale Signale auf zwei separaten Drähten übertragen und die jeweiligen Zustände permanent miteinander verglichen.
- Diese Module können außerdem die elektrischen Verbindungen auf Kabelbrüche und Kurzschlüsse hin überwachen, sofern die eigene Spannungsversorgung eingesetzt wird.
- Für einige Spezialfälle existieren auch fehlersichere analoge Module, diese werden allerdings nicht häufig eingesetzt.

- Im Standard **IEC 61131-3** wurden SPS-Programmiersprachen definiert.
- Die Hersteller halten sich auch an diese Definitionen, weshalb man jede beliebige SPS in einer Variante dieser Sprachen programmieren kann.
- Eine Übertragung der Programme von einem zum anderen Hersteller ist allerdings nicht ohne weiteres möglich, da spezifische Erweiterungen vorgenommen und Bibliotheken zur Vereinfachung der Programmierung zur Verfügung gestellt werden.

- **Eingang**
(engl. input) mit dem physikalischen Eingang verbunden
- **Ausgang**
(engl. output) mit dem physikalischen Ausgang verbunden
- **Merker**
(engl. flag) Speicher für beliebige Zwischenergebnisse (vom Programmierer festgelegt)
- **Zähler**
(engl. counter) Zählt die erfolgten Taktzyklen, kann mit einem Startwert initialisiert werden und vor- oder rückwärts laufen
- **Zeitglied**
(engl. timer) kann z.B. zur Nachbildung der Funktionalität von Zeitrelais verwendet werden

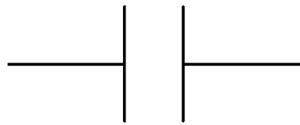
Merker und Zähler können auch *remanent* definiert werden, d.h. ihre Werte bleiben auch über Neustarts hinweg erhalten

Ladder-Diagram (LAD) / Kontaktplan (KOP)

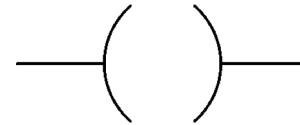
Diese graphische Programmiersprache wurde entwickelt um die Transition von klassischen VPS-Systemen zu SPS-Programmierung zu erleichtern.

Die Programme in dieser Sprache gleichen einem elektrischen Schaltplan, in welchem die Eingänge mit den Ausgängen logisch verschaltet werden. Über diverse Gatter können auch logische Berechnungen auf den Signalen ausgeführt werden.

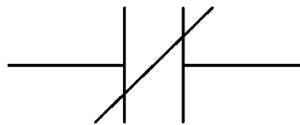
Ladder-Diagram (LAD) / Kontaktplan (KOP)



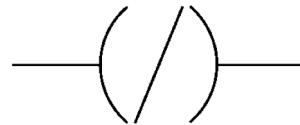
Eingang (normal geöffnet)



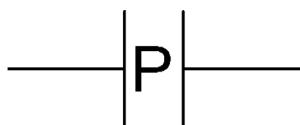
Ausgang



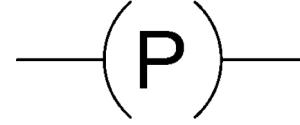
Eingang (normal geschlossen)



Ausgang (negiert)



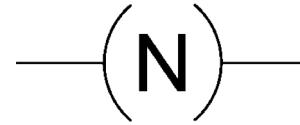
Eingang (positive Flanke)



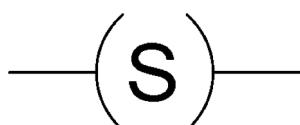
Ausgang (positive Flanke)



Eingang (negative Flanke)



Ausgang (negative Flanke)



Ausgang (eingeschaltet halten)

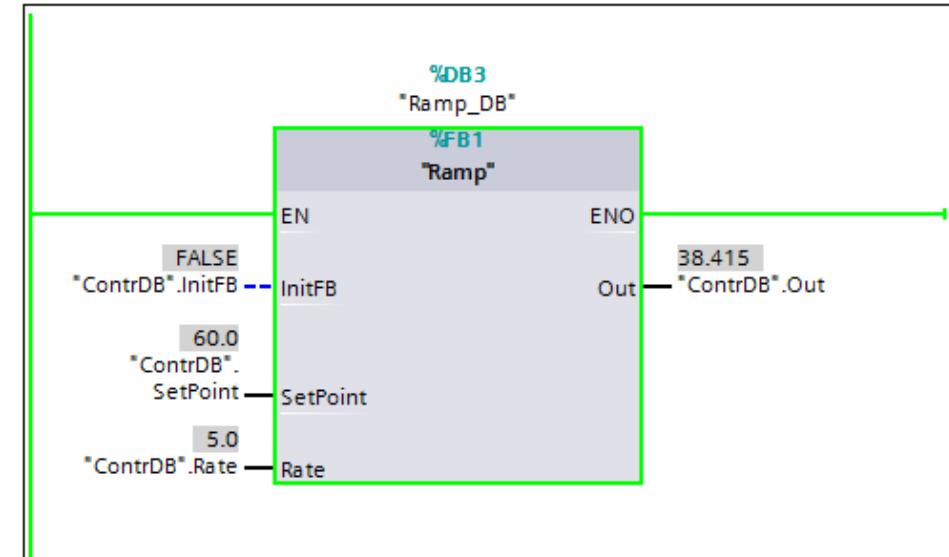


Ausgang (ausgeschaltet halten)

Function Block Diagram (FBD)

ist eher als Erweiterung zu LAD zu sehen. Mit diesem Diagrammtyp kann das gesamte Programm in einzelne Funktionsblöcke aufgeteilt werden. Die tatsächliche Ausformulierung des Programms geschieht daher auch in LAD. Jeder Funktionsblock muss seine Schnittstelle nach außen bzw. sein Speicher-Mapping definieren.

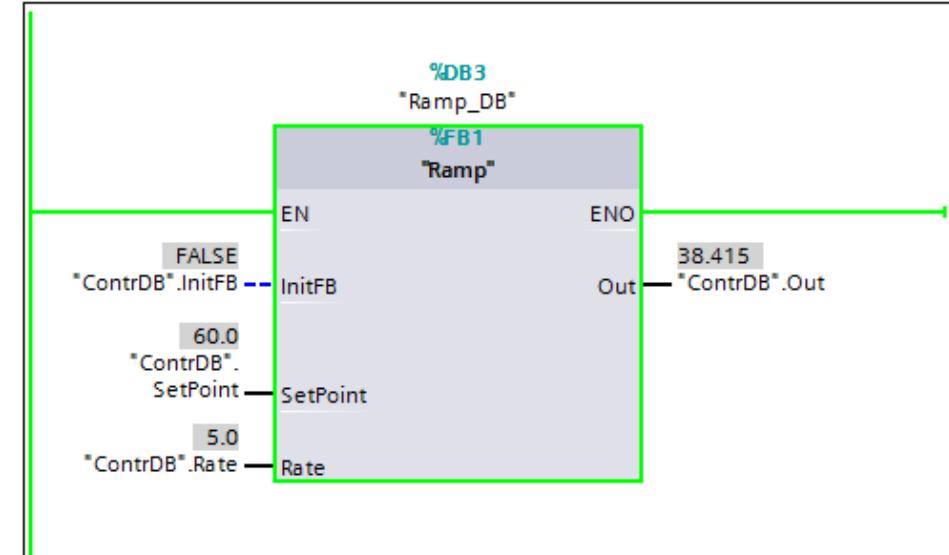
Auch Funktionsblöcke werden vom Betriebssystem nach dem EVA-Prinzip bearbeitet. Um mehrere Instanzen eines Funktionsblocks unterstützen zu können, muss jedem Aufruf eines Funktionsblocks eine konkrete Instanz der benötigten Memory-Map übergeben werden.



Function Block Diagram (FBD)

Folgende Typen von Variablen werden in Funktionsblöcken unterstützt:

- **Input:** Kann innerhalb des Funktionsblocks nur gelesen werden
- **Output:** Kann innerhalb des Funktionsblocks nur geschrieben werden
- **InOut:** Kann in einem Funktionsblock gelesen und geschrieben werden
- **Static:** Interne Variable, die über Zyklen hinweg bestehen bleibt
- **Temp:** Interne Variable, die jedoch nach jedem Zyklus gelöscht wird
- **Constant:** Interne Konstante, unveränderlich



Herstellerbibliotheken (Beispiel: Siemens)

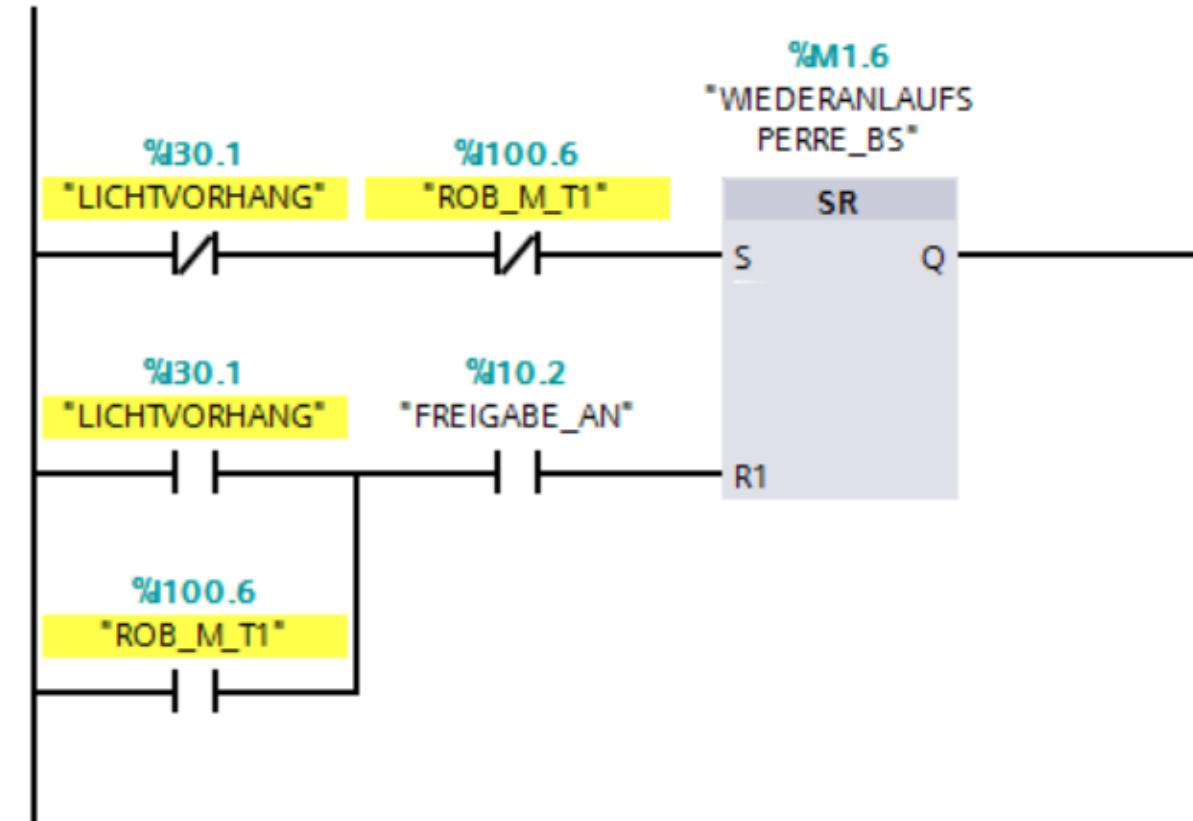
Die Hersteller von SPSen liefern meist ausgedehnte Bibliotheken aus, die dem Programmierer das Nachimplementieren häufig benötigter Funktionsblöcke abnehmen.

Name	Description
- Safety functions	
ESTOP1	Emergency STOP up to stop category 1
TWO_H_EN	Two-hand monitoring with enable
MUT_P	Parallel muting
EV1oo2DI	1oo2 evaluation with discrepancy analysis
FDBACK	Feedback monitoring
SFDOOR	Safety door monitoring
ACK_GL	Global acknowledgment of all F-I/Os in an ...
- Timer operations	
IEC timers	
TP	Generate pulse
TON	Generate on-delay
TOF	Generate off-delay
- Counter operations	
IEC Counters	
CTU	Count up
CTD	Count down
CTUD	Count up and down

Herstellerbibliotheken

Beispiel: Set-Reset-Flip-Flops

Ein aus der Schaltungstechnik entlehnter Baustein, der es erlaubt ein Ausgangssignal mit einem vorübergehend anliegenden Eingangssignal zu Setzen und diese nur mit einem anderen Signal wieder zurückzusetzen.



Structured Text Language (STL)

ist eine sehr einfache prozedurale Programmiersprache. Es gelten die selben Paradigmen und Konventionen wie für LAD/FBD, allerdings wird das Programm textuell ausformuliert.

```
IF newval <> prevval THEN
    index := index MOD COUNT;
    measdata[index] := measval_in;
    index := index + 1;
END_IF;
```

Graph (SFC)

ist vergleichbar mit einem Sequenz- bzw. Aktivitätsdiagramm. In diesem Diagramm können verschiedene Verzweigungen und Einstiegs- / Ausgangspunkte definiert werden. Diese können selber wiederum andere Unterprogramme aufrufen, beispielsweise einen FBD-Block.

