



Technische Hochschule Georg Agricola

INDUSTRIEAUTOMATION

SPEICHERPROGRAMMIERBARE STEUERUNG (SPS)

Bochum

Prof. Dr.-Ing. Lars N. Josler

INHALT DER VORLESUNG

- **Definition und Grundlagen SPS**
- **Hardwareaufbau einer SPS**
- **Funktionalität**
- **Steuerungs-Betriebssystem**
- **Grundlagen SPS-Programmierung**

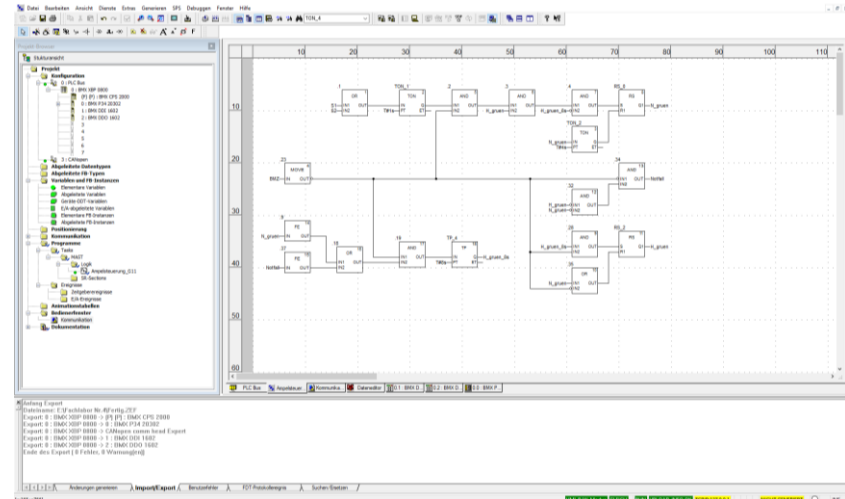


SPEICHERPROGRAMMIERBARE STEUERUNG



Eine speicherprogrammierbare Steuerung (SPS) ist

- ein Gerät, das programmiert und eingesetzt wird, um eine Maschine oder Anlage zu regeln bzw. zu steuern
- ein industrieller Rechner mit Schnittstellen zu Sensoren und Aktoren



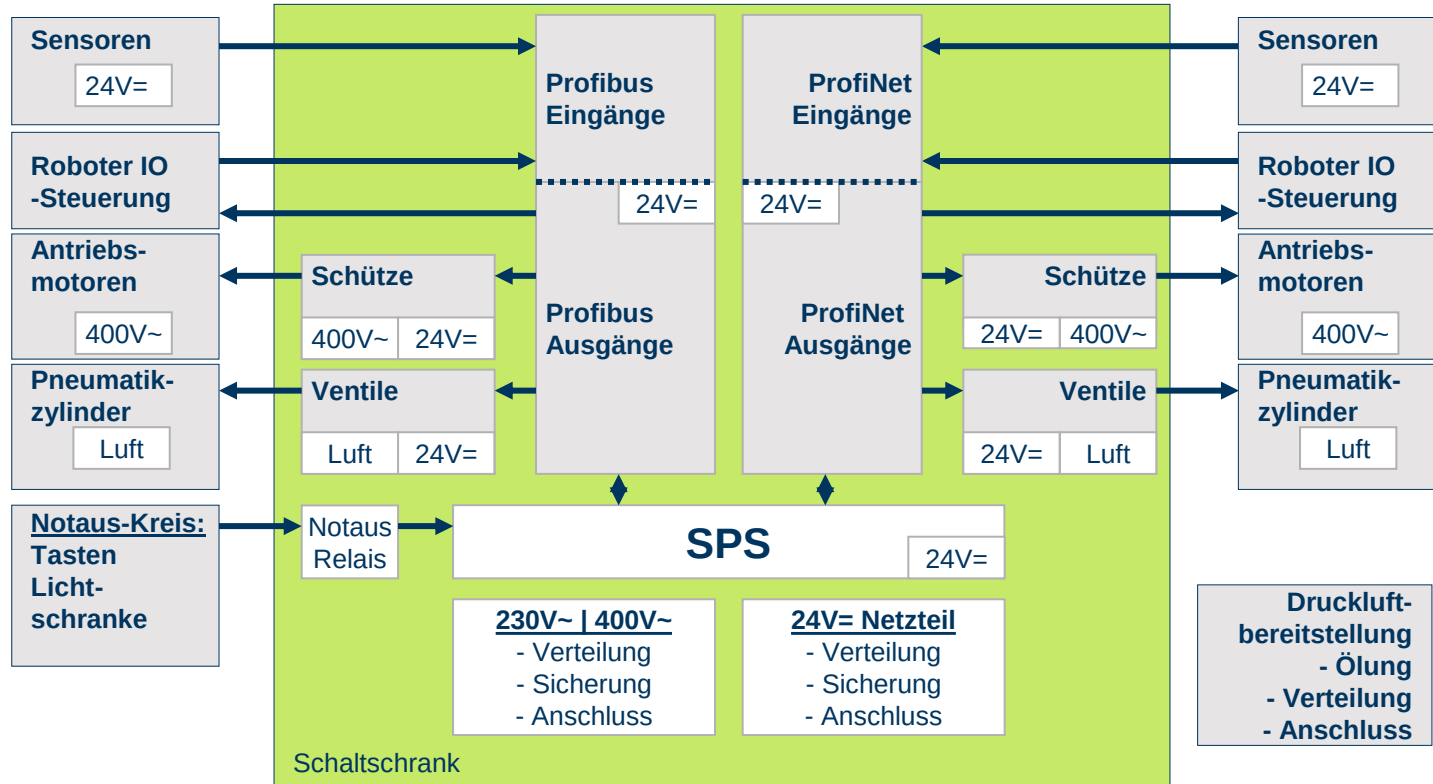
SPS: DEFINITION



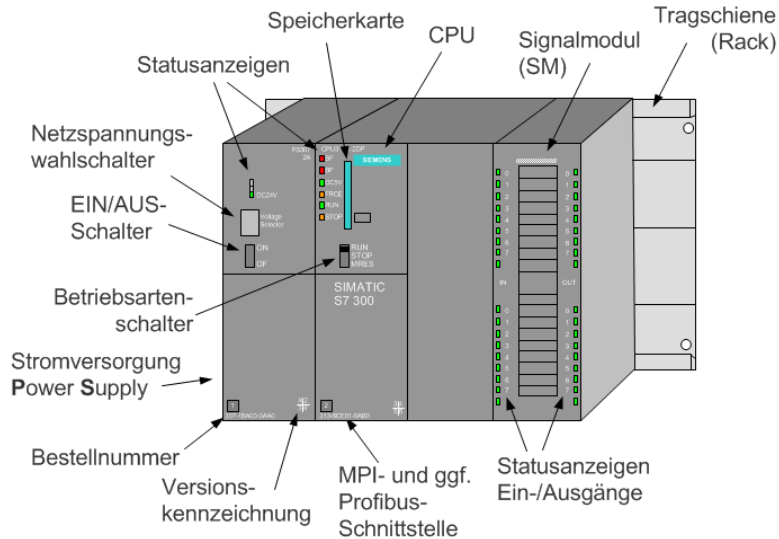
- **Definition nach IEC 61131-1:**
 - Eine SPS ist ein digital arbeitendes, elektronisches System, mit einem programmierbaren Speicher zur internen Speicherung von anwenderorientierten Anweisungen, um besondere Funktionen wie Logik, Ablauf, Zeit, Zählen und Arithmetik auszuführen und durch digitale oder analoge Eingänge und Ausgänge verschiedene Typen von Maschinen oder Prozessen zu steuern.
 - Eine SPS ist ein kleiner Computer der Maschinen steuert.
- **SPS = PLC (Programmable Logic Controller)**



SPS IM KONTEXT

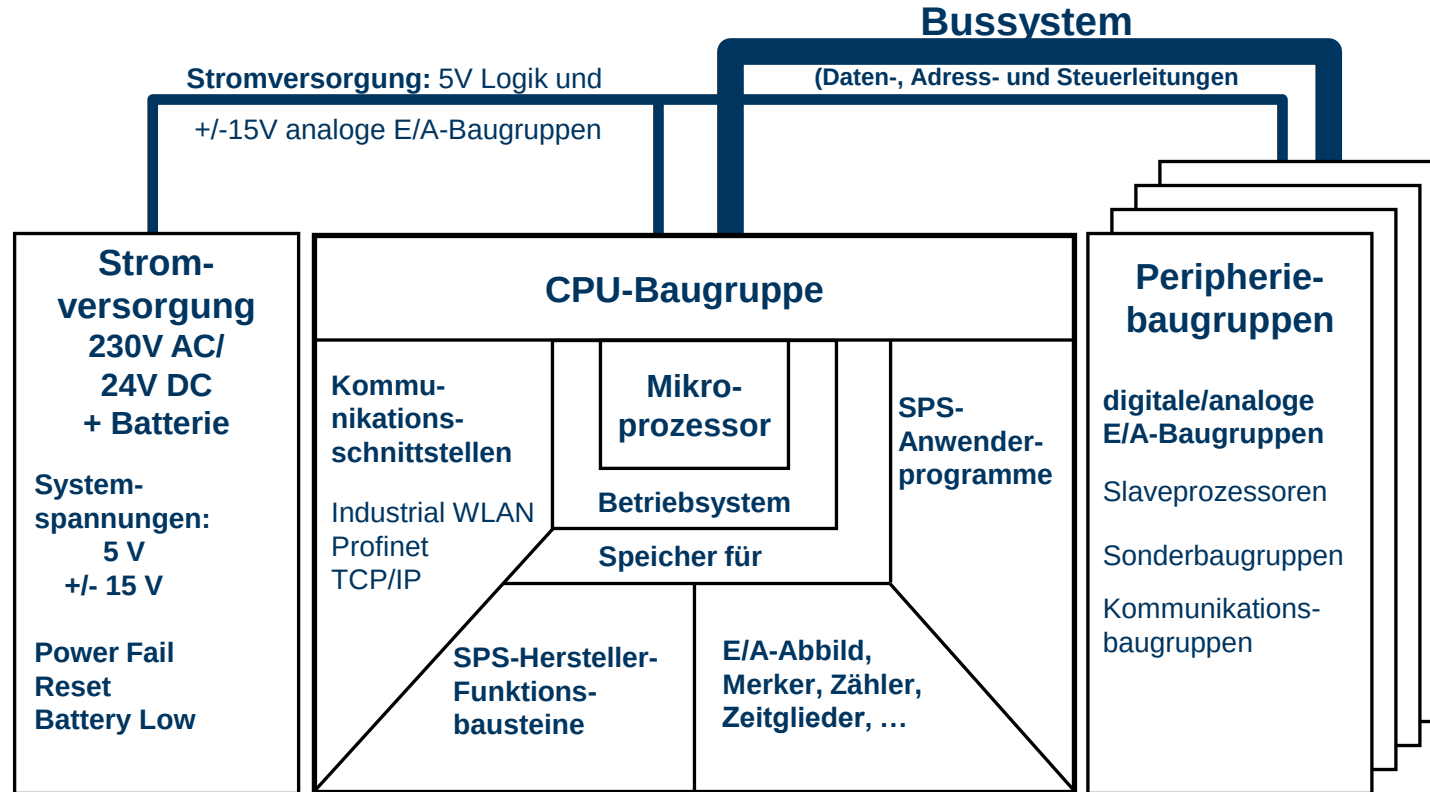


SPS-AUFBAU: MODULARER AUFBAU



Quellen: <https://www.xplore-dna.net/mod/page/view.php?id=294> (links); eigene Aufnahme (Labor der THGA) (rechts)

HARDWARE-STRUKTUR EINES MODULAREN SPS-SYSTEMS





Stromversorgungskarte oder PS (Power Supply) → 230 V AC in 24 V DC

CPU (Central Processing Unit) → Gehirn der SPS

Serie:

3 = 300er

4 = 400er

12 = 1200er

15 = 1500er

1516-3 PN/DP

Leistung

(Arbeitsspeicher):

1511, 1512, ..., 1516

Anschluss:

MPI = Multi-Point-Interface

DP = Profibus

PN = Profinet (LAN)

3 Profinet-Anschlüsse und 1 Profibus



1511C-1 PN → Compact

1513F-2 PN → Failsafe → für Sicherheitsfunktionen



Eingangsbaugruppe → E-Modul:

Digitale Eingangsbaugruppe (DI=Digital Input)=BE

Analoge Eingangsbaugruppe (AI=Analog Input)=AE

Ausgangsbaugruppe → A-Modul:

Digitale Ausgangsbaugruppe (DO=Digital Output)=BA

Analoge Ausgangsbaugruppe (AO=Analog Output)=AA

Mischbaugruppe: DI/DO oder AI/AO

UNTERSCHIEDE ZWISCHEN KONVENTIONELLER DATENVERARBEITUNG UND SPS



Konventionelle Datenverarbeitung	SPS
Datenorientierte Programmbearbeitung	E/A-orientierte Programmbearbeitung
Prinzipiell wird das Programm nach dem Start einmalig durchlaufen.	Die Programmbearbeitung erfolgt in Echtzeit , zyklisch oder ereignisorientiert.
Das Programm wartet üblicherweise bei Peripheriezugriffen auf die Peripheriereaktion. Einem anderen Programm kann ggf. Rechenzeit zugewiesen werden.	Im zyklischen Steuerungsprogramm werden blockierende Warteoperationen vermieden, um deterministische Zyklus- und Reaktionszeiten sicherzustellen
Zur Programmierung werden Hochsprachen (C#, C++, Java ...) eingesetzt.	Zur SPS-Programmierung werden spezielle SPS-Sprachen (KOP, AWL, FUB, ST, AS) eingesetzt.
Üblicherweise werden Standard-Betriebssysteme eingesetzt.	Es gelangen überwiegend herstellerspezifische Echtzeitbetriebssysteme zum Einsatz.

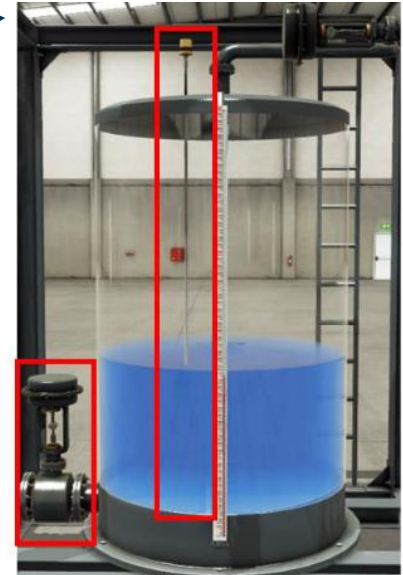


■ Digitale Signale

- Binäre (digitale) Sensoren → Schalter, Taster, Lichtschranken, Näherungsschalter, ...
- Digitale Aktoren → Schütze, Relais, Meldeleuchten

■ Analoge Signale

- Analoge Sensoren → Füllstandssensoren, Durchflusssensoren, Drucksensoren, Temperatursensoren
- Analoge Aktoren → Proportionalventil, Frequenzumrichter



VORTEILE DER SPS



- Flexibilität
- Geringerer Platzbedarf
- Höhere Zuverlässigkeit
- Geringere Kosten
- Möglichkeit der Vernetzung mit anderen Systemen
- Möglichkeit zur Fehlerdiagnose und Fernwartung
- Schneller Funktionswechsel durch Programmänderung



DIE IDEALE SPS



Die Anforderungen an eine ideale SPS lassen sich nicht in einem System vereinigen. Aus diesem Grund haben sich viele SPS-Anbieter mit verschiedenen Systemen am Markt etabliert.

- mechanisch robust durch voll gekapselte Module
- mechanisch kompakt für kleinste Einbauvolumina
- mechanisch robust durch hohe Schock- und Rüttelfestigkeit
- thermisch robust durch hohe zulässige Umgebungstemperatur
- thermisch robust ohne Zwangskonvektion
- elektrisch robust durch hohe EMV-Festigkeit
- elektrisch robust durch hohe Toleranzen für die Stromversorgung
- elektrisch robust durch zerstörungsfreie Peripherieschnittstellen
- beliebige Skalierbarkeit für Prozessperipherie und CPU
- einfachste Anschluss technik
- einfache Installation und Konfiguration
- einfachste Programmierung mit Standard-Programmierungswerkzeugen
- wartungsfreier Betrieb
- gute Diagnosemöglichkeiten
- hohe Zuverlässigkeit, hohe Sicherheit
- Unterstützung aller Kommunikationsschnittstellen
- geringster Preis
- langfristige Verfügbarkeit

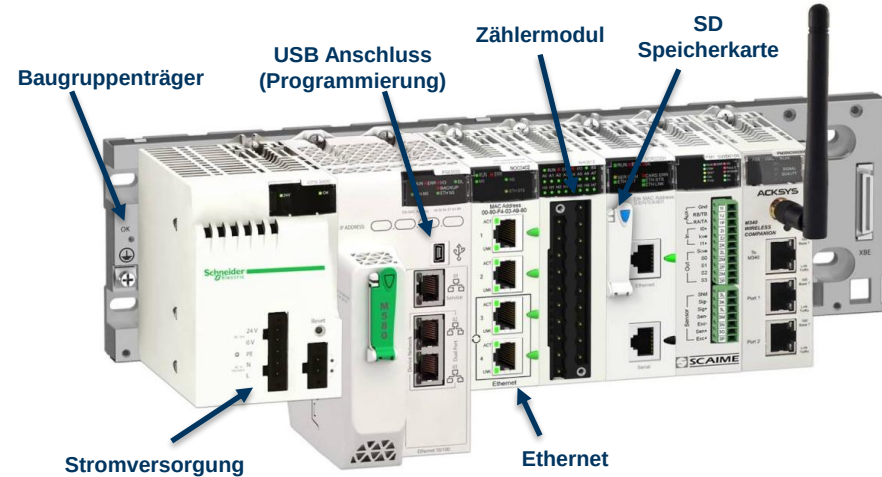
MODULARER SPS-GERÄTEAUFBAU



Speicherprogrammierbare Steuerungen werden modular aufgebaut, um sie an die unterschiedlichsten Aufgaben einfach anpassen zu können.

- Skalierbarkeit der CPU-Leistung
(Prozessor, Multi-Master, Master Slave)
- Skalierbarkeit des Speicherausbaus
(Speicherart und -kapazität)
- Art der Prozessinterfaces
(digital, binär, analog)
- Skalierbarkeit der Prozessinterfaces
(Gesamtanzahl, Kanäle je Modul)
- Kommunikationsinterfaces

(Punkt-zu-Punkt, Feldbusse, Sensor-/Aktorbusse, TCP/IP, ...)



BAUFORMEN VON STEUERUNGEN



- Industrie-PC (Standardrechner) mit dezentraler E/A
- Industrie-PC mit Slot-SPS und dezentraler E/A
- SPS-CPU mit Industrie-PC Funktionalität
- Modulare Steuerung mit dezentraler E/A
- Vernetzte, modulare Steuerung

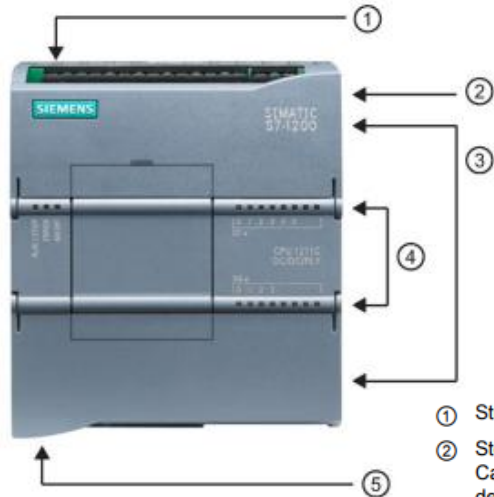


Zentrale, modulare Steuerung

- Kompaktsteuerung
- Vernetzte Kompaktsteuerung
- Integrierte Kompaktsteuerungen (z.B. Ventilinsel, HMI)
- Klemmensteuerung



KOMPAKTSTEUERUNG SIMATIC S7-1200 MIT ANSCHLUSSBELEGUNG

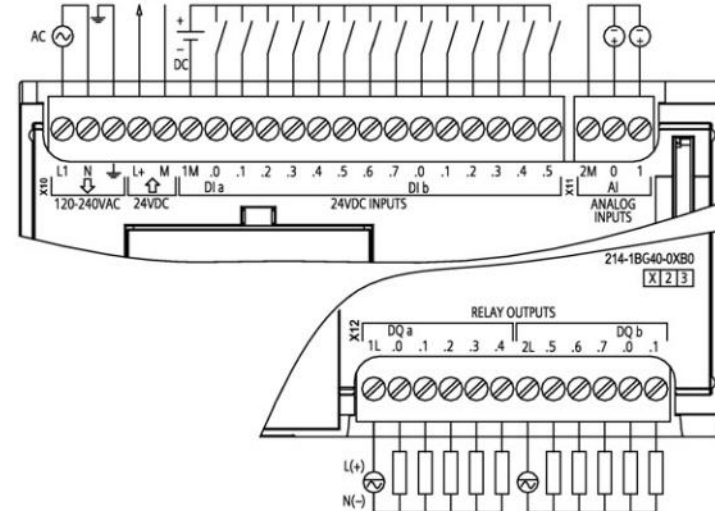


- ① Stromanschluss
- ② Steckplatz für eine Memory Card unter der oberen Abdeckklappe
- ③ Steckbarer Klemmenblock für die Anwenderverdrahtung (hinter den Abdeckklappen)
- ④ Status-LEDs für die integrierten E/A
- ⑤ PROFINET-Anschluss (auf der Unterseite der CPU)

Stromversorgung
(AC 85...264 V)

14 Eingänge (DC 15...30 V)

DC 24 V für
Eingangsgeber
oder Erweiterungs-
module (280 mA)

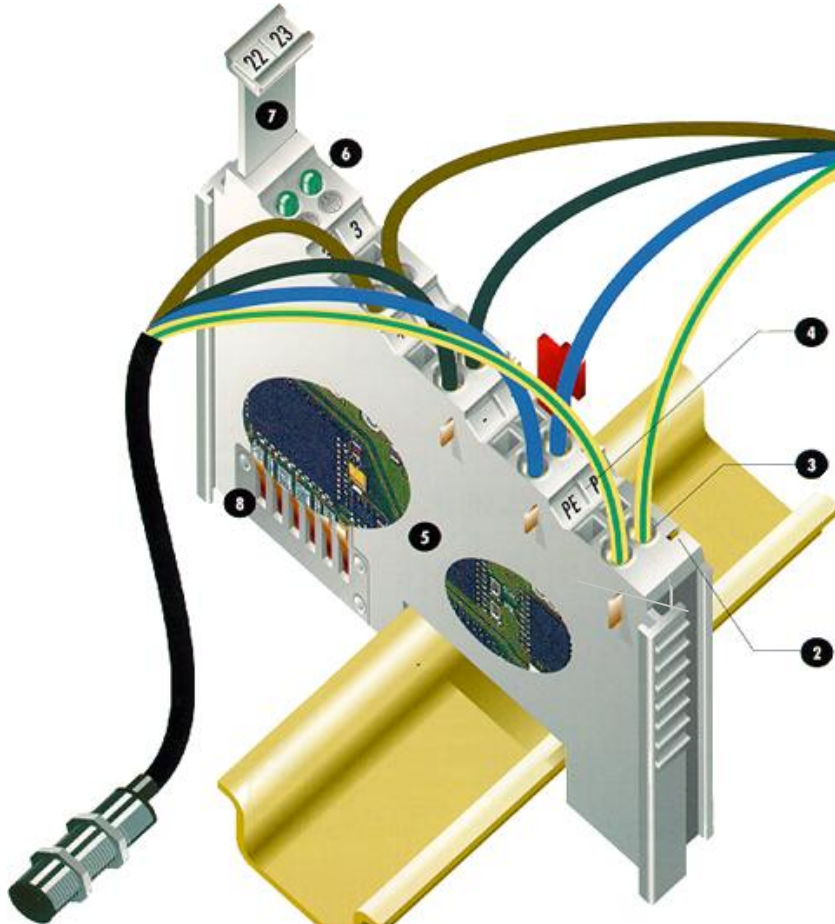


10 Relais-Ausgänge (DC 20,4...28,8 V)



Modular aufgebaute SPS

- 1 - 4-Leiter-Anschlußtechnik (Signal, +24V, 0 V, Schutzleiter)
- 2 - Brückung der Leistungs-stromversorgung durch Power-Kontakte zwischen den Klemmen
- 3 - CAGE CLAMP Anschlusstechnik
- 4 - Beschriftungsfelder für Kontaktbeschriftung
- 5 - Potentialtrennung zwischen Ein-/Ausgangselektronik und Bus
- 6 - Signalstatus wird durch LED angezeigt
- 7 - herausziehbare Lasche für Klartext- und Gruppenbeschriftung
- 8 - K-Bus erkennt, adressiert und aktiviert Klemme und transportiert Daten zum Buskoppler. Keine Adress- und Konfigurationseinstellung notwendig.



KLEMMEN-SPS-SYSTEM AUF HUT-SCHIENE



CPU

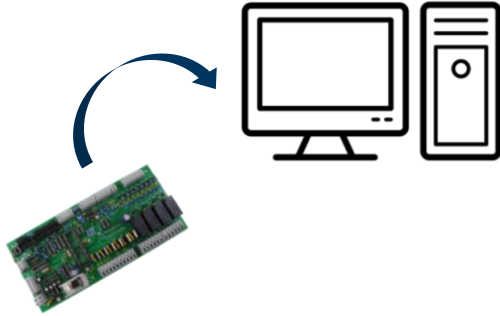
Signalzusammenstellung:

- 1 Stromversorgungsklemme
- 6 digitale Eingänge, 24V DC, 3ms
- 6 digitale Ausgänge, 24V DC, 0.5A
- 2 Zählereingänge
- 4 analoge Eingänge, +/-10V

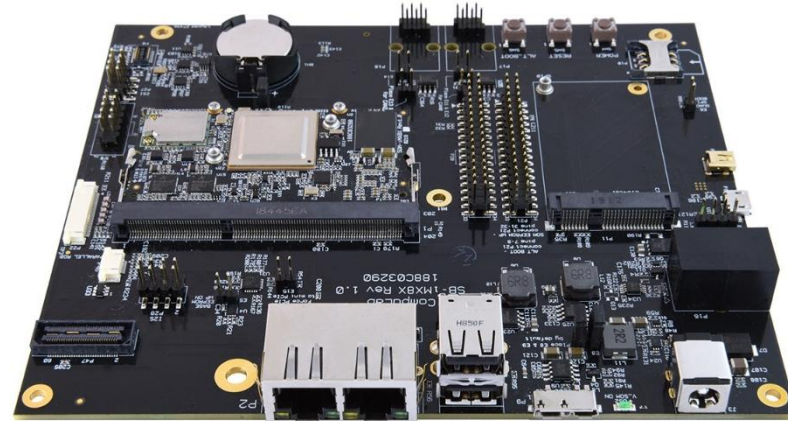
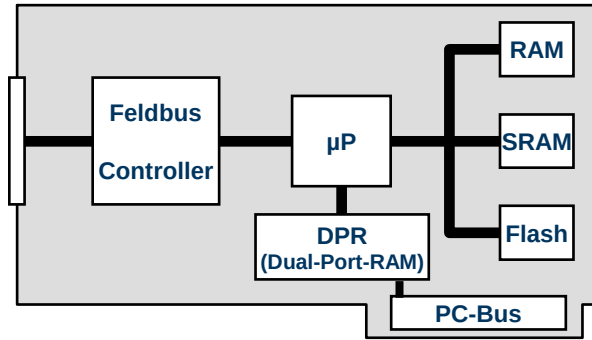
INTEGRIERTE SPS: BEISPIEL ANLAGE UND BEDIENGERÄT



INTEGRATION DER SPS IN EINEN PC



- Basis des Steuerungssystems ist ein Standard- oder Industrie-PC (IPC)
- durch Einsteckkarten kann die SPS die Ressourcen des PC nutzen (PC-Busanschluss, Netzversorgung, Gehäuse)
- PC-Sonderbauformen





■ Alte Generation

- S7-300 → kleine/mittlere Anlagen
- S7-400 → anspruchsvolle Applikationen (z.B. Atomkraftwerke)



- SIMATIC Manager = Step 7 Classic = Step 7 V5.x (Version V5.1 – V5.6)

■ Neue Generation

- S7-1200 → kleine/mittlere Anlagen
- S7-1500 → mittlere/große Anlagen



- TIA Portal (TIA = Totally Integrated Automation) Version V11 bis V16

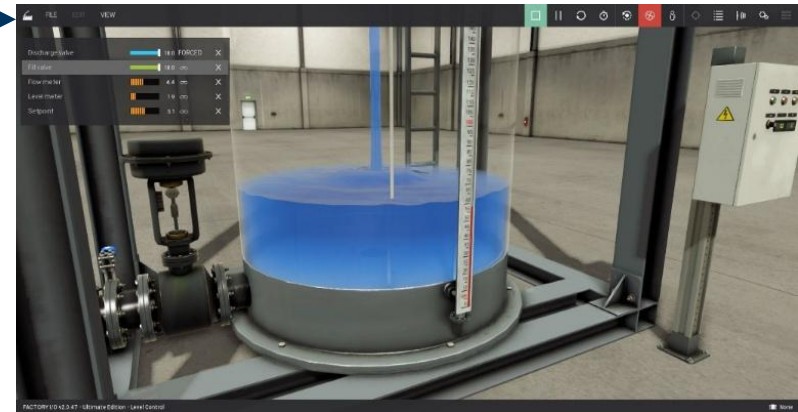
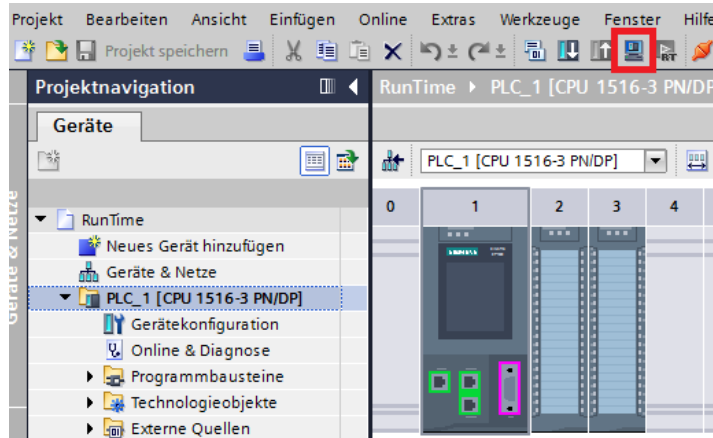


- TIA Portal gibt es in verschiedenen Versionen → Alle 1-2 Jahre kommt eine neue Version heraus.
- Windows Version → Nicht alle Versionen sind mit Windows 10 kompatibel
- Systemvoraussetzungen
- Das TIA Portal gibt es in der Version
 - Basic → Funktionsplan (FUP), Kontaktplan (KOP), Structured Control Language (SCL)
 - Professional → FUP, KOP, SCL, Anweisungsliste (AWL), S7-Graph
- Mit der „Basic-Version“ kann nur die Modellreihe S7-1200 programmiert werden.
- Mit der „Professional-Version“ können alle Modellreihen programmiert werden.
- Trial-Lizenz (21 Tage kostenlos) → TIA-Portal Free Trial





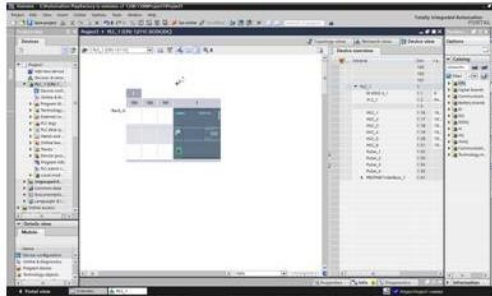
- Mit PLCSIM können wir unsere Programme auch ohne physische SPS simulieren.
- Mit dem Simulationsprogramm Factory I/O (Free Trial (30 Tage kostenlos) | Starter Edition 99 € - dauerhaft (36 € für 1 Jahr) Starter Edition) können virtuelle Industrieanlagen erstellt und programmiert werden.
- Die Kombination aus PLCSIM und Factory I/O ist perfekt zum üben geeignet.



TIA-PORTAL FREE TRIAL



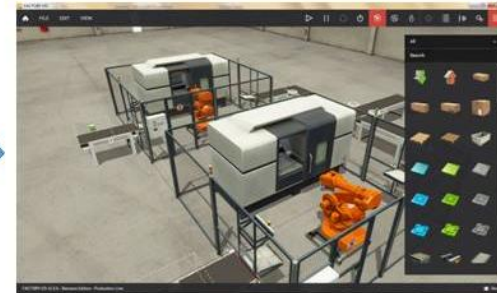
**TIA Portal
(v13-v16)**



**S7- PLC SIM
(v13-v16)**



Factory I/O



**“USE FACTORY I/O
WITHOUT PLC”**

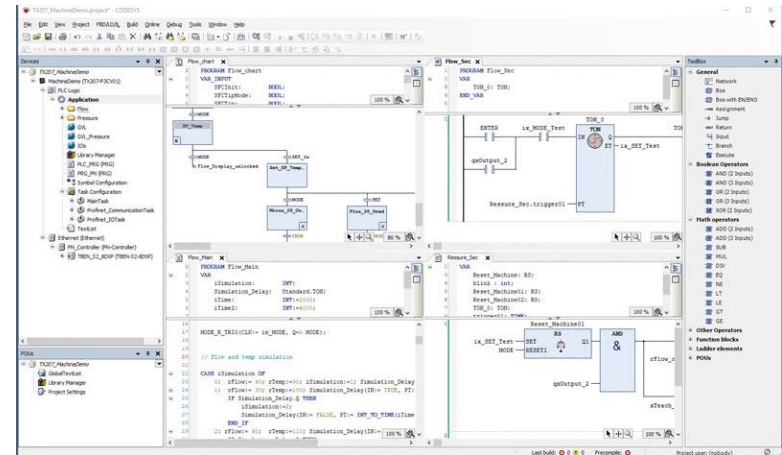


TIP: PG/PC Settings in
Control Panel must be set
to PLCSIM.TCP/IP

Quelle: <https://support.industry.siemens.com/cs/document/109772803/simatic-step-7-inkl-safety-und-wincc-v16-trial-download?dti=0&lc=de-WW>

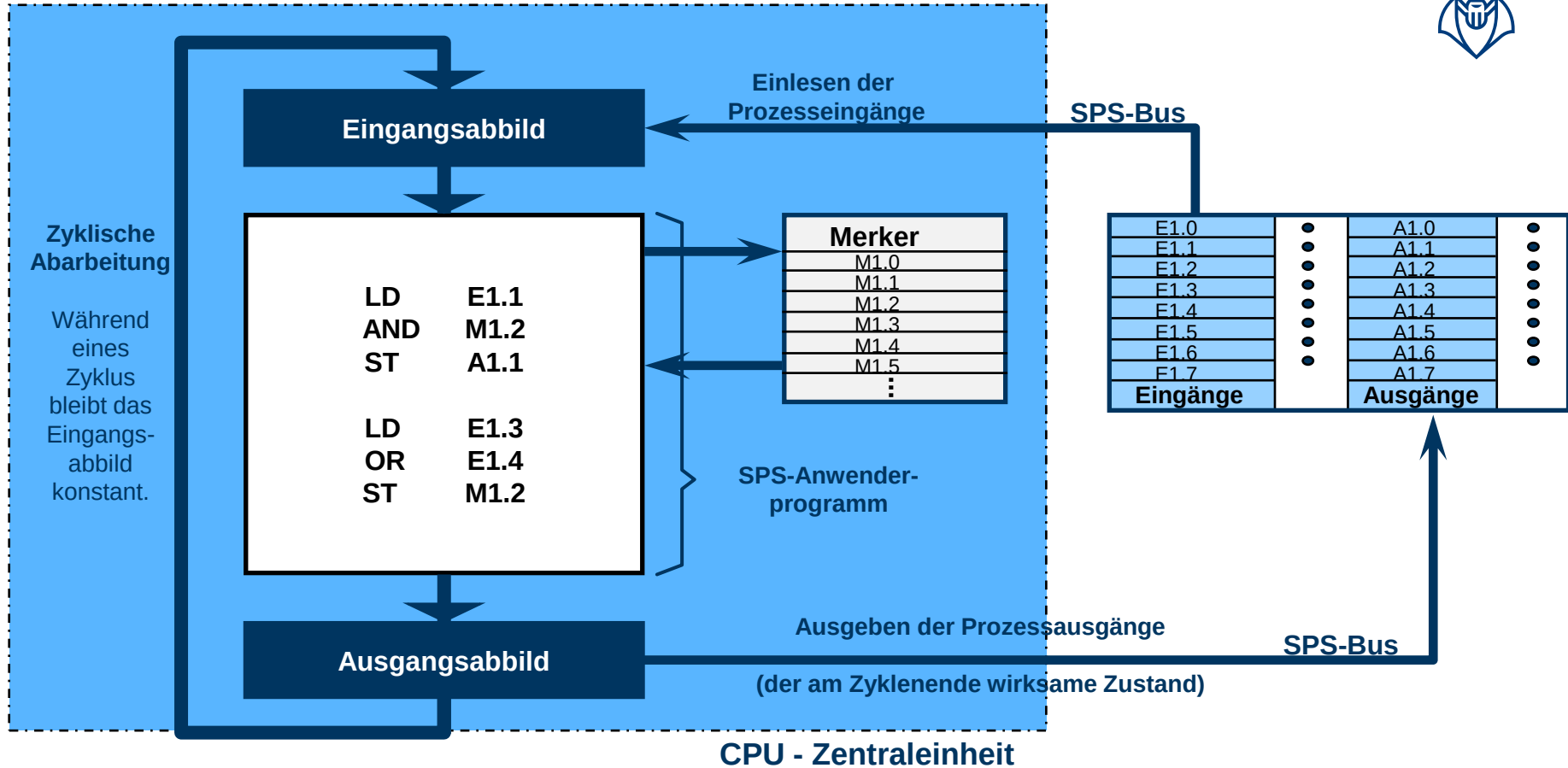


- Herstellerübergreifende Entwicklungsumgebung zur Programmierung von speicherprogrammierbaren Steuerungen (SPS)
- Kostenlose Entwicklungsumgebung (CODESYS Development System); Laufzeitsysteme und Zusatzfunktionen teilweise kostenpflichtig
- Programmierung nach IEC 61131-3 mit verschiedenen SPS-Programmiersprachen (z. B. KOP, FUP, ST und AS)
- Integrierte Visualisierung zur Erstellung von Bedienoberflächen (HMI) für Maschinen und Anlagen
- Simulation und Debugging zum Testen und Analysieren von SPS-Programmen
- CODESYS Control Win: Softwarebasierte SPS-Laufzeitumgebung, mit der ein Windows-PC als SPS eingesetzt werden kann
- Breite Hardwareunterstützung für Steuerungen unterschiedlicher Hersteller
- Einsatzgebiete: Maschinenbau, Anlagenautomatisierung und industrielle Steuerungstechnik



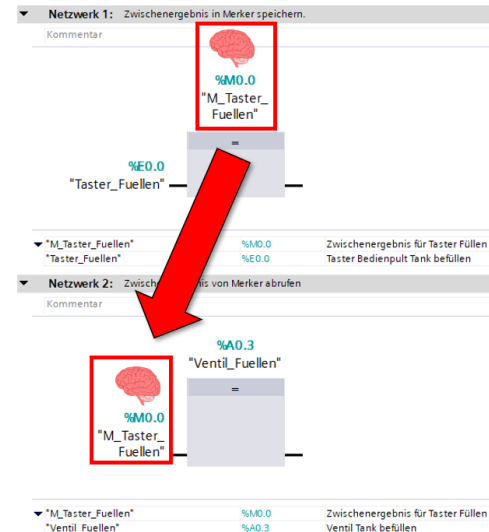
<https://de.codesys.com/>

ZYKLISCHE PROGRAMMABARBEITUNG IN EINER SPS





- Merker (oder Datenbausteine) werden genutzt um Zwischenergebnisse zu speichern.
- Merker können ebenfalls Bit-, Byte-, Wort- und Doppelwortweise adressiert werden.



Bitweise:

z.B. M10.0, M3.4, M8.7

Byteweise:

z.B. MB1, MB18, MB44

Wortweise:

z.B. MW5, MW22, MW111

Doppelwortweise:

z.B. MD0, MD10, MD18



Im Unterschied zu den Systemen der konventionellen Datenverarbeitung erfüllen SPS-Steuerungssysteme **erhöhte Anforderungen an die zeitgerechte Datenerfassung und Datenverarbeitung** (Echtzeitfähigkeit) sowie an die **Robustheit im industriellen Umfeld** und sind über eine geeignete Prozessperipherie bzw. Kommunikationsverbindung **direkt an die zu steuernden oder zu beobachtenden technischen Prozesse gekoppelt**. Die **Programmbearbeitung erfolgt zyklisch**.

Echtzeitfähigkeit (Rechtzeitigkeit)

- Eigenschaft, das SPS-System ständig ablaufbereit zu halten, damit es innerhalb einer vorgegebenen Zeitspanne auf Ereignisse eines technischen Prozesses reagieren kann.

Unterbrechungsfähigkeit (Priorität)

- Fähigkeit zum Anhalten/Unterbrechen eines aktiven SPS-Programms, welches durch ein Ereignis innerhalb oder außerhalb des SPS-Systems bewirkt wird und Starten eines höher-prioren SPS-Programms.

Wiederanlauffähigkeit

- Fähigkeit zur geordneten Wiederaufnahme der SPS-Funktion nach Ein-/Ausschalten, Ausfall oder Störung (Heißstart, Warmstart, Kaltstart).

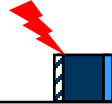
ZEITLICHE MÖGLICHKEITEN DER SPS-PROGRAMMAUSFÜHRUNG



Ereignisorientierte Bearbeitung

▨ Eingangsabbild einlesen
■ SPS-Programmbearbeitung
■ Ausgangsabbild ausgeben

■ SPS-Programmbearbeitung
□ freie Rechenkapazität



Zyklische Bearbeitung in festem Zeitraster

Ausgangsabbild am Zyklusende ausgeben

Ausgangsabbild am Zyklusanfang ausgeben

Dauerbearbeitung



ANFORDERUNGEN AN EIN ECHTZEIT-MULTITASKING-BETRIEBSSYSTEM



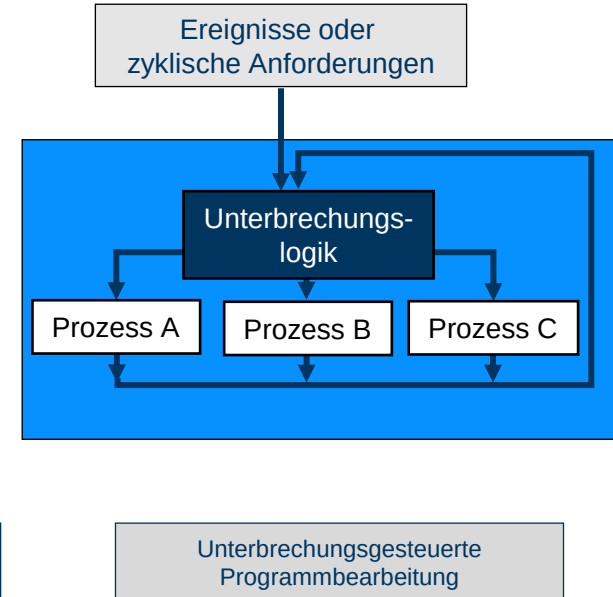
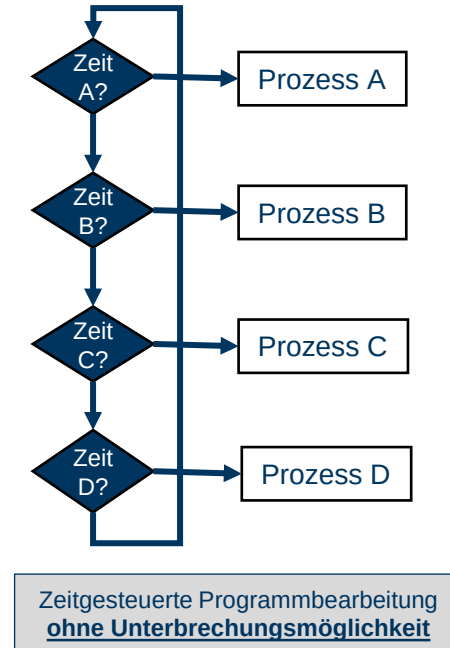
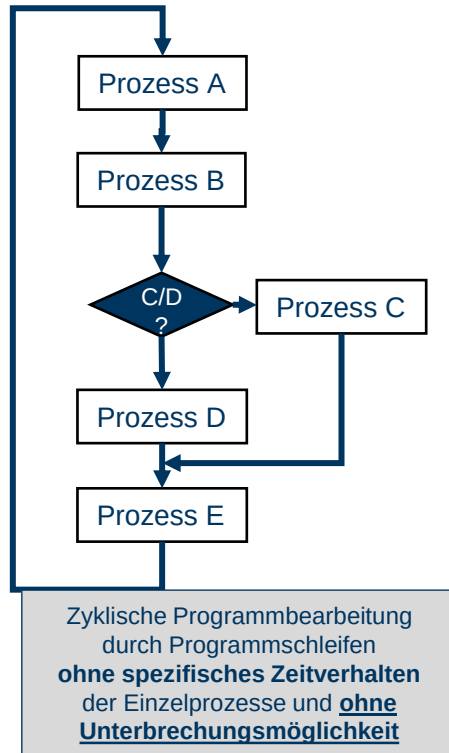
Generelle Betriebssystemfunktionen:

- Bindeglied zwischen Hardware und Anwendersoftware
- Den Anwendungen Ressourcen (Prozessor, Speicher und Geräte) zuweisen

Anforderungen an ein Echtzeit-Betriebssystem:

- Garantierte, maximale Reaktionszeit auf äußere Ereignisse (Rechtzeitigkeit)
- Beginn der Programmausführung nach einer maximal zulässigen Zeit (Latenzzeit)
- Beendigung von Aufgaben innerhalb einer bestimmten Zeit, Einhaltung der „Deadline“
- Verwaltung zyklisch wiederkehrender Aufgaben
- Verwaltung unterschiedlicher Prioritäten von Ereignissen und Aufgaben
- Hohe Zuverlässigkeit

ORGANISATIONSFORMEN ZUR ABARBEITUNG MEHRERER AUFGABEN

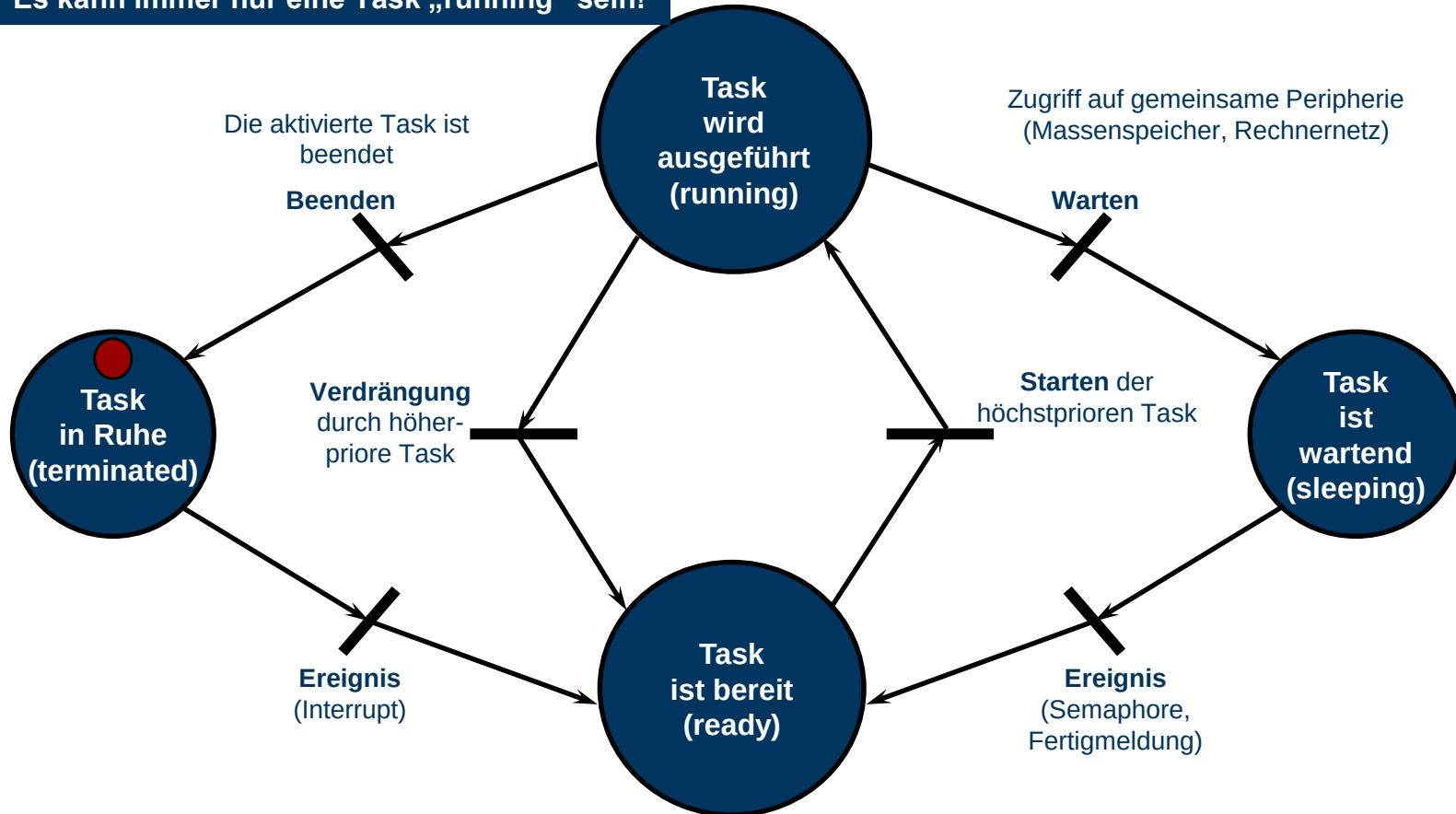




Begriff	Bedeutung
Echtzeit-bearbeitung	Bearbeitung, bei der das Programm synchron zum Prozess läuft. Die Reaktionszeit des Rechners muss deterministisch sein und innerhalb einer angemessenen Reaktionszeit des Prozesses liegen (garantiert rechtzeitige Reaktionen).
Echtzeit, harte/weiche	Bei harter Echtzeit müssen Zeitschranken (Deadlines) unbedingt eingehalten werden. Bei weicher Echtzeit ist deren Überschreitung in gewissen Grenzen erlaubt, ohne dass das System funktionsuntüchtig wird.
Task	In einem Echtzeitbetriebssystem definieren zyklisch aktivierte und ereignisaktivierte Tasks das Zeitverhalten des Systems. In einer Task werden Programme instanziiert und ausgeführt. Tasks können sich in verschiedenen Zuständen befinden (<i>bereit</i> , <i>ausgeführt</i> , <i>wartend</i> und <i>ruhend</i>). Tasks laufen unabhängig voneinander, sie können allerdings miteinander kommunizieren, sich auch gegenseitig starten und anhalten.
Multitasking	Verarbeitung einer Vielzahl konkurrierender, quasiparalleler Tasks
Multitasking (kooperativ)	Verfahren, bei dem Tasks nur unterbrochen werden können, wenn sie dazu bereit sind. Unkooperative Tasks können dabei das ganze System anhalten. Daher ist dieses Verfahren nicht für Echtzeit geeignet (Beispiel: Office-Betriebssysteme).
Multitasking (preemptiv)	Verfahren, bei dem Tasks jederzeit unterbrochen werden können. Es ist für die harte Echtzeitverarbeitung geeignet. (Preemption = Taskverdrängung)
Scheduler	Der Scheduler entscheidet, welche Task als nächstes laufen darf und schaltet zwischen den Tasks um.
Preemptive Scheduling	Prioritätsgesteuertes Zuteilen der CPU zu den Tasks mit Unterbrechung laufender Tasks

PETRI-NETZ EINER TASK

Es kann immer nur eine Task „running“ sein!

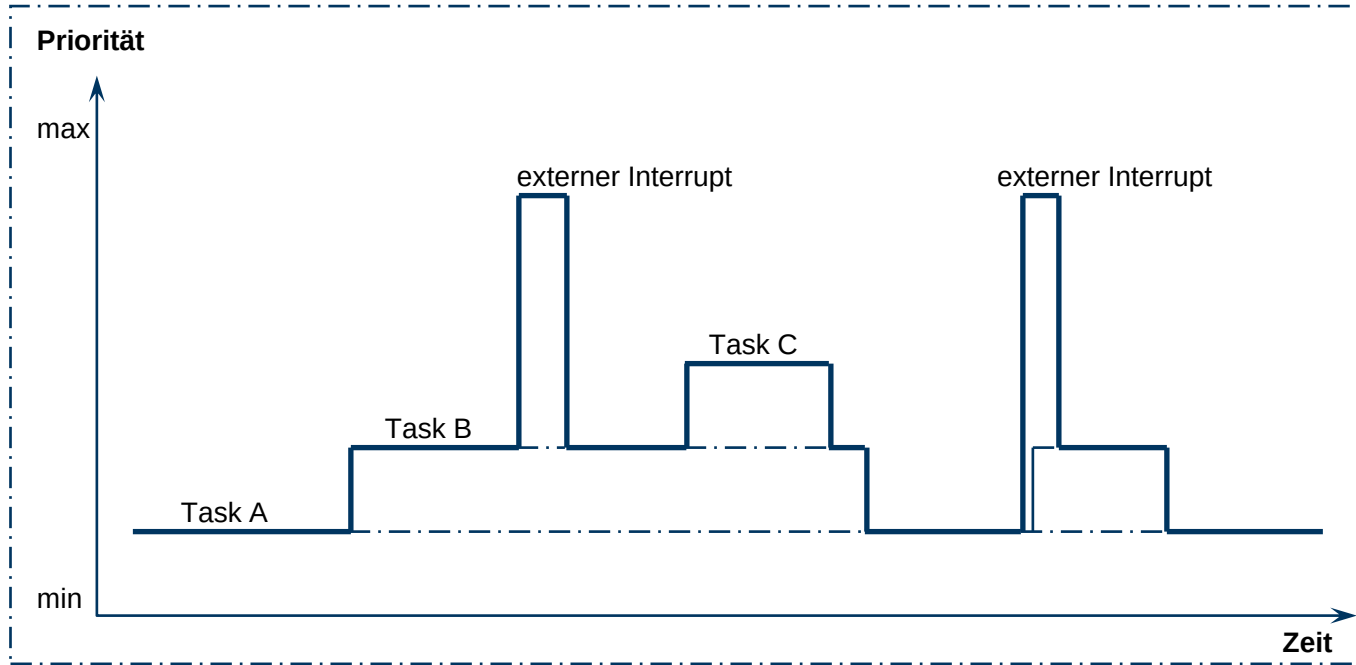


PETRI-NETZ EINER TASK



- Task bereit (ready): Alle Ressourcen bis auf den Prozessor zugeteilt
 - Taskausführung (running): Auch der Prozessor ist der Task zugeteilt
 - Task wartend (sleeping): Angeforderte Ressource ist belegt (Datei, Gerät, Speicher)
 - Task ruhend (terminated): Task hat Aufgabe erfüllt, keine weitere Prozessoranforderung (Ressourcenanforderung)
-
- UNIX: Priorität auch abhängig von der Wartezeit, Anwendung bei Steuerungsaufgaben unsinnig!
 - Beispiel: Zugriff auf Festplatte durch niedrig priorisierte Task, Unterbrechung durch hoch priorisierte Task mit ebenfalls Plattenzugriff!

PREEMPTIVE SCHEDULING

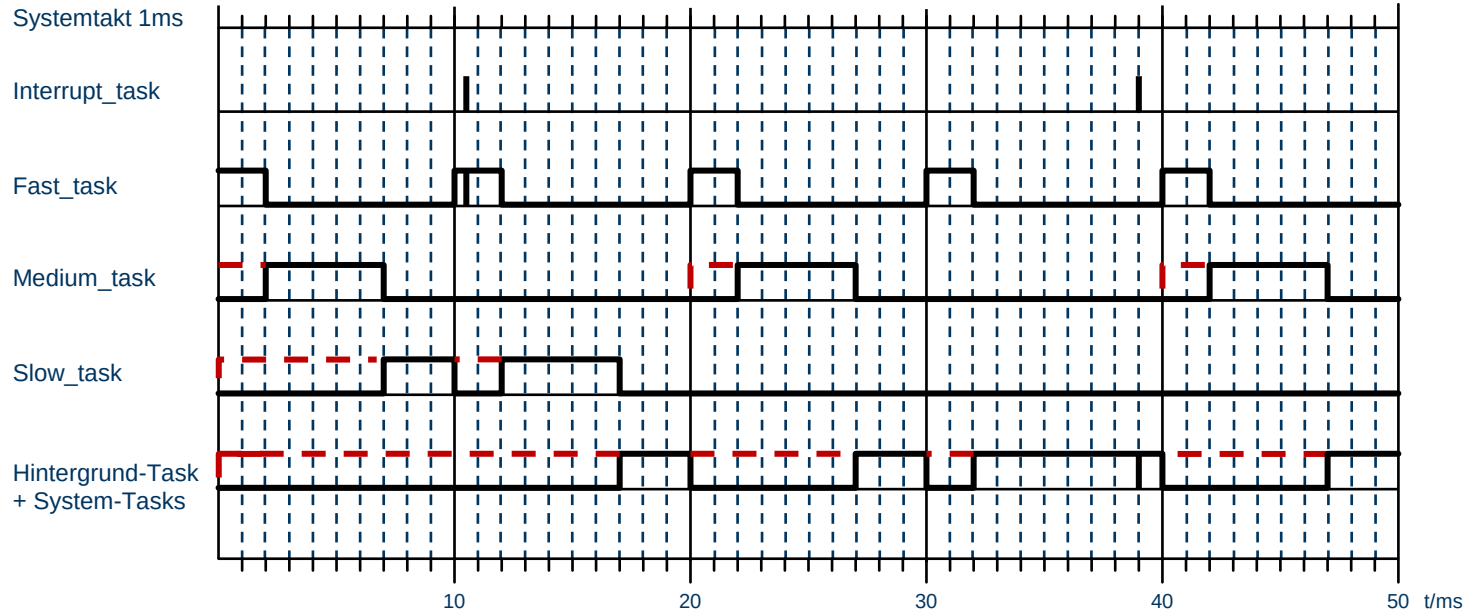
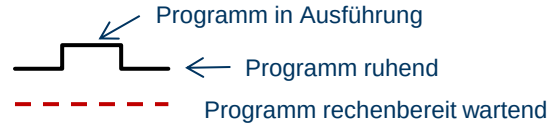


Preemptives Multitasking ist das wesentliche Kennzeichen von **Echtzeit-Betriebssystemen**. Tasks hoher Priorität können jederzeit laufende Tasks unterbrechen. Liegen mehrere Tasks gleicher Priorität vor, so entscheidet der Scheduler meist nach dem Round-Robin-Verfahren, welche Task die CPU zugeteilt bekommt.

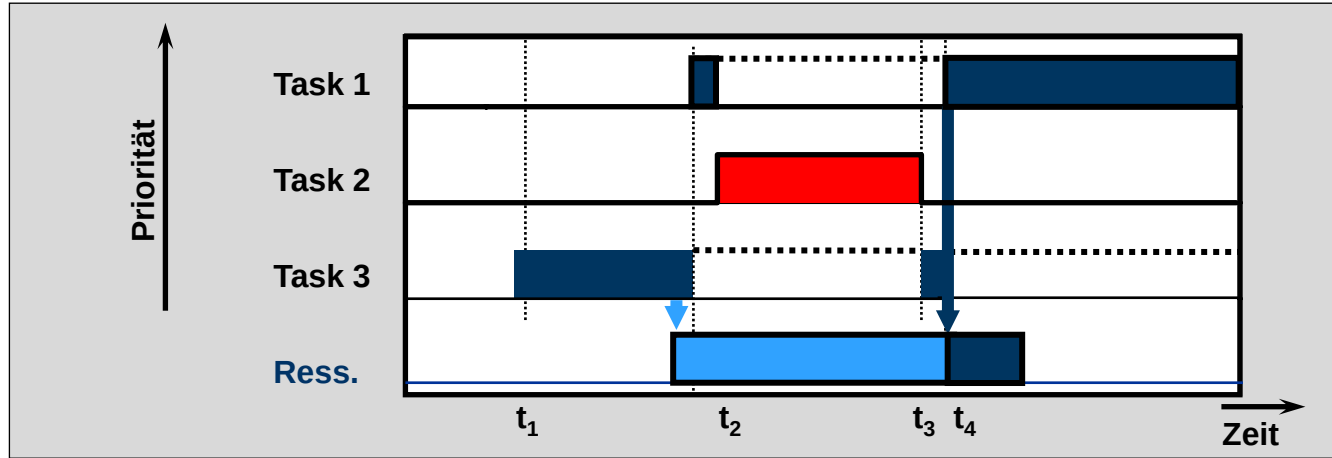
MULTITASK-SPS



TASK Interrupt_task	(PRIORITY:=1)	(* Peripherie-Interrupt *)
TASK Fast_task	(INTERVAL := t#10ms, PRIORITY := 2);	(* Ausführungszeit = 2ms *)
TASK Medium_task	(INTERVAL := t#20ms, PRIORITY := 3);	(* Ausführungszeit = 5ms *)
TASK Slow_task	(INTERVAL := t#50ms, PRIORITY := 4);	(* Ausführungszeit = 8ms *)



PROBLEM DER PRIORITÄTsumKEHR (PRIORITY INVERSION)



- t_1 : Task 1 wird blockiert, weil Task 3 Ressourcen belegt, die Task 1 benötigt → Task 3 läuft
- t_2 : Task 2 ist höher priorisiert als Task 3, also darf sie Task 3 unterbrechen → Task 2 läuft
- t_3 : Task 2 ist beendet, Task 3 läuft weiter → Task 3 läuft
- t_4 : Task 3 beendet Ressourcenzugriff, Task 1 kann auf benötigte Ressourcen zugreifen → Task 1 läuft

PROBLEM: Task 1 muss auf Task 2 und 3 warten, es erfolgt somit eine Prioritätsumkehr.

LÖSUNG: Vererbung der Priorität von Task 1 auf Task 3

Solange Task 1 wegen Task 3 blockiert ist, vererbt Task 1 ihre Priorität der Task 3. Dies schützt Task 3 und somit indirekt Task 1 vor einer Unterbrechung durch Task 2.

PROBLEM DER PRIORITÄTSSUMKEHR (PRIORITY INVERSION)

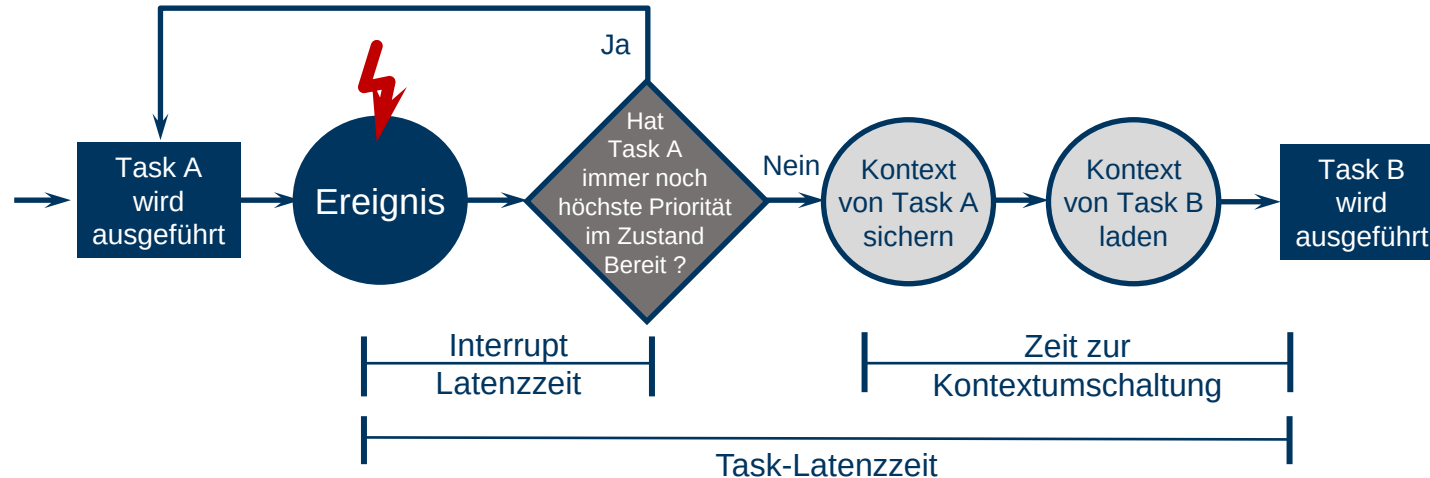


- Ein Prozess mit hoher Priorität will auf eine Ressource zugreifen, kann dies aber nicht, da die Ressource bereits von einem niedriger priorisierten Prozess belegt ist.
- Der hoch priorisierte Prozess muss warten, bis der andere Prozess die Ressource wieder freigibt. Dadurch sind die Prozess-Prioritäten in einer ersten Form der Prioritätsinversion vertauscht.
- Existiert nun ein Prozess mit mittlerer Priorität, der die fragliche Ressource nicht verwendet, kann dieser mittel priorisierte Prozess den niedrig priorisierten verdrängen, d.h. der mittel priorisierte Prozess wird anstelle des niedrig priorisierten Prozesses ausgeführt.
- Der niedrig priorisierte Prozess kann die Ressource nun nicht mehr freigeben, so dass der hoch priorisierte Prozess nicht zur Ausführung kommt.
- Damit hindert der mittel priorisierte Prozess indirekt auch den hoch priorisierten Prozess an der Ausführung, was er nach dem Prinzip des Prioritäts-Schedulings nicht darf.
- Die Priorität des hoch priorisierten Prozesses und des mittel priorisierten Prozesses sind somit in einer zweiten Form der Prioritätsinversion vertauscht.

TASK-LATENZZEIT UND CONTEXT SWITCHING



- Jede Task erhält ihren eigenen Kontext (Umgebung) vom Betriebssystemkern zugeteilt. Zum Kontext gehören die CPU-Umgebung (Rechenregister, Adressregister, Statusregister) und die Systemressourcen.
- Für eine schnelle Reaktion des Rechners auf ein Ereignis ist die Interrupt-Latenzzeit und die Zeit für die Kontextumschaltung von wesentlicher Bedeutung.





Begriff	Bedeutung
Interrupt	Unterbrechungsanforderung des Programmablaufes durch ein internes oder äußeres Ereignis
Task-Latenzzeit	Zeitspanne vom Eingang der Startanforderung für eine Task bis zum Übergang in den Zustand "Ausführung"
Interrupt-Latenzzeit	Zeitspanne vom Eintritt eines Interrupts bis zur kontrollierten Übernahme durch eine Interrupt-Behandlungsroutine (Scheduler)
Context Switching	Kontextumschaltung; Wechsel von einer Task zur nächsten
Ressourcen	Alle Hilfsmittel, die ein Programm zur Erfüllung seiner Aufgaben benötigt (Dateien, Bussystem, Hauptspeicher, Datenbereiche, Rechenzeit)
Semaphore	Flag zur Synchronisation von Zugriffen auf gemeinsame Ressourcen (aus dem griechischen Wort Signal).
Mutual Exclusion	Synchronisationsverfahren, bei dem Systemdienste der Taskverwaltung wechselseitig durch Semaphoren gesperrt werden.

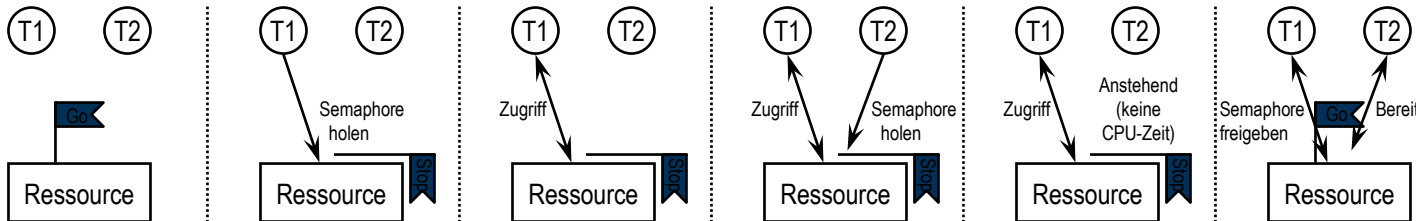


Task-Verdrängungsschutz:

- Problem: Task A verlangt exklusiven Zugriff auf Datenstruktur → kritischer Zustand
höherpriorisierte Task B verdrängt Task A
Datenstruktur nicht vollständig bearbeitet → inkonsistent, nicht mehr verfügbar
- Verdrängungsschutz durch Mutual Exclusion (gegenseitigen Ausschluss), d. h. Task A nimmt Semaphore, Task B kann Task A erst verdrängen, wenn kritischer Zustand der Task A vorüber und die Mutual-Exclusion-Semaphore wieder freigegeben ist.

Semaphoren:

- Kopplung von Semaphoren mit Ressourcen
- Sobald eine Task eine Ressource benötigt, muss sie zuerst auf die Semaphore zugreifen.
- Solange eine Task die Semaphore behält, wird allen anderen Tasks der Zugriff auf die Ressource verweigert.



[Quelle: FORCE COMPUTER]



Trend zur Nutzung standardisierter Betriebssysteme

- 😊 Bekannte grafische **Benutzeroberflächen** aus dem Bürobereich werden auch in der Fertigung gefordert (Anspruch, Akzeptanz, Vertrautheit, Schulungsaufwand).
- 😊 Leistungsstarke Büro-Rechner ermöglichen **preiswerte Hochleistungssteuerungen** mit hoher HW-Innovationsrate. (Leistungsfähige SPS-CPU's sind meist deutlich teurer als ein Büro-PC)
- 😊 Durch die Entwicklung der **Feldbussysteme** ist eine preiswerte industrielle Ankopplung der Sensorik und Aktorik an die PC-Hardware möglich geworden.
- 😊 Skalierbare Windows-Betriebssysteme (CE) mit der gleichen grafischen Bedienoberfläche. Durch den Preisverfall bei den FLASH-Speichern sind **keine mechanisch bewegten Massenspeicher** erforderlich.
- 😊 Microsoft entwickelt oberhalb der Feldbusebene mit dem **OPC-Standard** Aktivitäten zur Kommunikation im industriellen Bereich (OLE for Process-Control).
- 😊 **Durchgängige** einheitliche Programmierschnittstellen und damit Austauschbarkeit von SW-Lösungen (Lösungen auch von Drittanbietern erhältlich).
- 😊 Ausgereifte **preiswerte Entwicklungswerkzeuge** mit einem hohen Bekanntheitsgrad (Entwicklungskapazitäten ohne Schulungsaufwand weltweit verfügbar).
- 😞 **Problem beim Windows-Einsatz im industriellen Bereich:**
Die fehlende harte Echtzeitfähigkeit verhindert die Anwendung der Windows Betriebssysteme im Steuerungsbereich. Windows CE ist in Steuerungsanwendungen noch nicht sehr verbreitet.

SPS-PROGRAMMIERSPRACHEN NACH IEC 61131



- **Ziel der Norm IEC 61131 ist die Kostenreduktion durch**
 - Vereinheitlichung
 - Synergien
 - Wiederverwendung von Softwareprodukten
 - Nutzung von Bibliotheken

- **POE = Programm-Organisationseinheit**

- **POE bildet die kleinste unabhängige Software-Einheit**

- **3 Arten von POEs:**
 - **Programm (PROG)**
 - **Funktionsbaustein (FB)**
 - **Funktion (FUN)**

PROGRAMM-ORGANISATIONSEINHEITEN (POU)



- **Programm-Organisationseinheiten (POUs) sind grundlegend Bausteine des Anwenderprogramms**
 - Können aus Programmen, Funktionen oder Funktionsbausteinen bestehen.
 - Dienen der Modularisierung und Strukturierung des Programms
 - Jede POU folgt einem einheitlichen Aufbau:
 - Deklarationsteil: Hier werden Variablen, Schnittstellen und die Einheitsbezeichnung definiert.
 - Anweisungsteil: Dieser enthält den eigentlichen Code und greift auf die Ressourcen aus dem Deklarationsteil zu.
 - POUs müssen im Projektnamen eindeutig sein, um innerhalb des Namensraums eindeutig identifizierbar zu sein.
 - POUs sind die kleinste Einheit, die es ermöglicht, die Logik strukturiert und modular zu organisieren.

DEKLARATIONSTEIL BEISPIEL



Variablen werden zu Beginn jeder POE deklariert

(Lokale Variable *)*

VAR VarLokal : BOOL; END_VAR (* lokale boolesche Var. *)

(Aufrufschnittstelle: Eingangsparameter *)*

VAR_INPUT VarEin : REAL; END_VAR (* Eingangsvariable *)

VAR_IN_OUT VarEinAus : UINT; END_VAR (* Ein- und Ausgangsvariable *)

(Rückgabewerte: Ausgangsparameter *)*

VAR_OUTPUT VarAus : INT; END_VAR (* Ausgangsvariable *)

(Globale Schnittstelle: Globale/Externe Variablen und Zugriffspfade *)*

VAR_EXTERNAL VarGlob : WORD; END_VAR (* extern von anderer POE *)

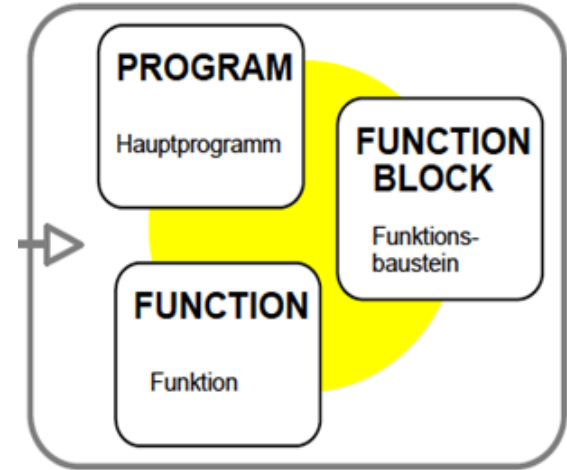
VAR_GLOBAL VarGlob : WORD; END_VAR (* global für andere POEs *)

VAR_ACCESS VarPfad : WORD; END_VAR (* Zugriffspfade der Konfiguration *)

PROGRAMM-ORGANISATIONSEINHEIT (POE)



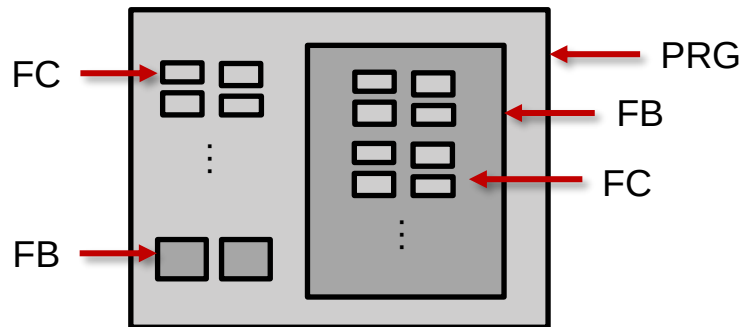
- **Programm (PROG)**
 - Hauptprogramm
 - Alle Variablen müssen in dieser POE deklariert werden
- **Funktionsbaustein (FB)**
 - Parametrierbare POE mit statischen Variablen
 - Liefert bei gleichem Eingangsparameter Ergebnisse die auch von internen oder externen Variablen abhängen können
 - Beispiel: Zeit- oder Zählerbaustein
- **Funktion (FUN)**
 - Parametrierbare POE ohne statische Variablen
 - Kein Gedächtnis
 - Liefert bei gleichem Eingangsparameter dasselbe Ergebnis als Funktionswert



PROGRAMM (PRG)



- **Ein Programm ist die übergeordnete POU.**
 - Programm kann andere Programme, Funktionsbausteine und Funktionen aufrufen
 - Programme besitzen interne Variablen sowie Ein-/Ausgangssignale
 - Programme werden über Tasks (oder ein Programm oder FB) vom Laufzeitsystem aufgerufen
 - Ein Programm realisiert den Funktionsablauf des Systems indem es Sensorwerte ausliest und Aktoren ansteuert
 - Können nicht von anderen Programmen instanziiert werden
 - Ein Programm existiert nur einmal und steuert beispielsweise den gesamten Prozessablauf



```
PROGRAM Main
VAR
    StartButton: BOOL;
    StopButton: BOOL;
    MotorStatus: BOOL;
END_VAR

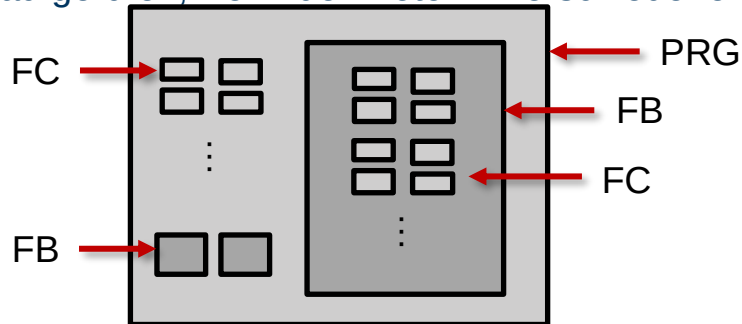
IF StartButton AND NOT StopButton THEN
    MotorStatus := TRUE;
ELSIF StopButton THEN
    MotorStatus := FALSE;
END_IF
```

[Quelle: Codesys.com]

FUNKTIONSBAUSTEIN (FB)



- Ein Funktionsbaustein (FB) ist ein wiederverwendbares Modul, das eine oder mehrere Funktionen oder Operationen umsetzt.
- FB kann Eingänge (Input-Parameter), Ausgänge (Output-Parameter, mehrere) und interne Variablen haben
- FB kann in verschiedenen Teilen des Steuerungsprogramms wiederverwendet werden
- Funktionsbausteine (FB) können interne Variablen besitzen, deren Werte zwischen den Aufrufen erhalten bleiben („Gedächtnis“).
- Jeder Aufruf erzeugt eine *Instanz* des FB mit den jeweiligen Datentypen des FB
- Beispiel: Motorsteuerung wurde entwickelt und wird in verschiedenen Teilen des Programms aufgerufen, wenn der Motor in verschiedenen Situationen verwendet wird.



```
FUNCTION_BLOCK Motor_Control
VAR_INPUT
    Start: BOOL;
    Stop: BOOL;
    Speed: INT;
END_VAR
VAR_OUTPUT
    Running: BOOL;
    ActualSpeed: INT;
END_VAR
VAR
    Acceleration: INT; // Interner Speicher
END_VAR

IF Start AND NOT Stop THEN
    Running := TRUE;
    ActualSpeed := Speed;
END_IF
```

[Quelle: Codesys.com]

FUNKTIONSBAUSTEIN (FB)



- Werden von Programmen und FBs aufgerufen oder können selbst Funktionen und FBs aufrufen
- Die Anwendung von Funktionsbausteinen geschieht in vier Schritten:

Baustein definieren

instanzieren

Baustein verfügbar machen

Parameter übergeben

parametrieren

Baustein mit Werten versorgen, Baustein liest ein

Baustein ausführen lassen

aufrufen

Baustein arbeitet

Ausgangswerte auslesen

Ergebnisse

Baustein gibt Werte aus, Werte können verwendet werden



Generell:

„Ein FB ist eine eigenständige, nach außen gekapselte Datenstruktur mit einer auf ihr definierten Berechnungsvorschrift“

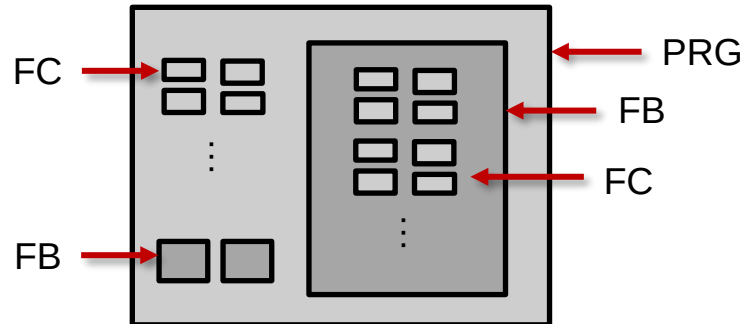
Einschränkungen bei Funktionsbausteinen zur Sicherstellung der Wiederverwendbarkeit

- Keine feste Zuordnung zu SPS-Adressen
- Keine Deklaration von Zugriffspfaden
- Dateninformation von außerhalb kann nur über die zulässigen Variablenarten als POE-Schnittstelle in den FB gebracht werden

FUNKTION (FC)



- **Funktionen sind die grundlegenden Bausteine der Steuerungslogik und werden in Funktionsbausteinen oder anderen POU-Typen implementiert.**
- Führen eine spezifische Aufgabe oder Operation aus, die innerhalb des Steuerungsprogramms durchgeführt werden soll
- Enthält kein „Gedächtnis“ – Gleiche Eingangssignale führen immer zu den gleichen Ausgangssignalen
- Immer nur ein Rückgabewert
- Beispiel: Eine Funktion kann eine einfache oder komplexe Aufgabe sein, wie beispielsweise das Prüfen eines Sensors, die Durchführung einer Berechnung oder das Steuern einer Aktorbewegung.



```
FUNCTION Calculate_Area : REAL
VAR_INPUT
    Length: REAL;
    Width: REAL;
END_VAR

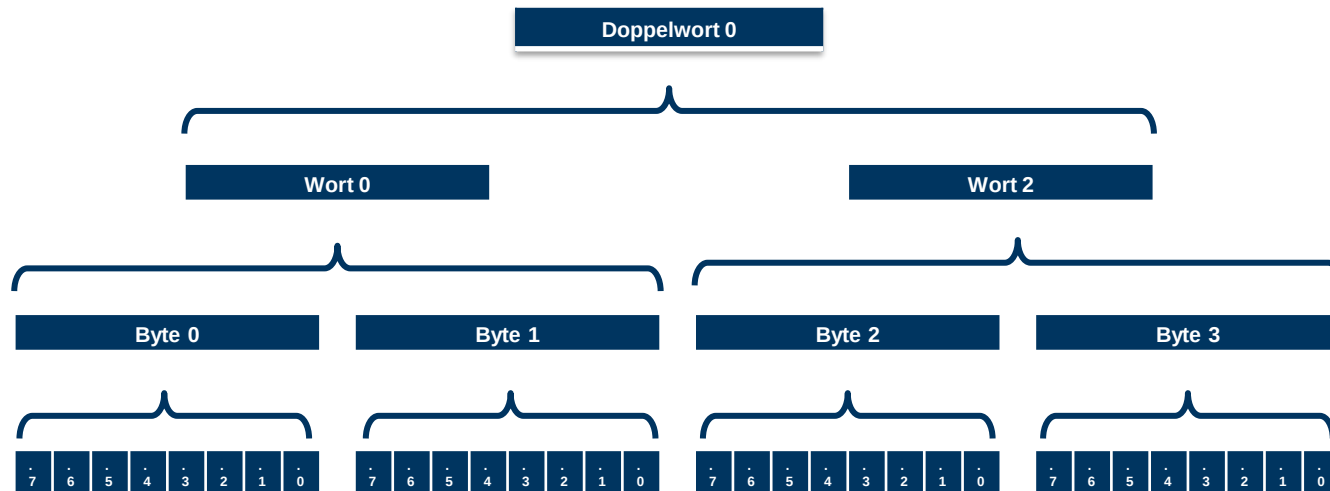
Calculate_Area := Length * Width;
```

[Quelle: Codesys.com]



- In einer SPS gibt es unterschiedlich große Speicherbereiche.
- Diese unterscheiden sich in ihrer Größe bzw. Länge.
- Folgende Datenlängen sind besonders relevant:

Datenlänge	Speicherbedarf	Ohne Vorzeichen	Mit Vorzeichen
Bit	1 Bit	0 oder 1	—
Byte	8 Bit	0 bis 255	–128 bis 127
Word	16 Bit	0 bis 65.535	–32.768 bis 32.767
DWord	32 Bit	0 bis 4.294.967.295	–2.147.483.648 bis 2.147.483.647





- **Dezimal**
 - Basis: 10
 - Grundelemente: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
 - Beispiel: 365
- **Dualsystem/ Binärsystem → SPS**
 - Basis: 2
 - Grundelemente: 0, 1
 - Beispiel: $00000001\ 01101101 = 256 + 64 + 32 + 8 + 4 + 1 = 365$
- **Hexadezimalsystem → SPS → große Werte mit möglichst wenig Ziffern darzustellen**
 - Basis: 16
 - Grundelemente: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
 - Beispiel: $16D = 1 \times 256 + 6 \times 16 + 13 \times 1 = 365$
- **BCD-Code**
 - Binär Codierte Dezimalzahl → jede Zahl wird einzeln in eine duale Zahlenfolge umgewandelt
 - Beispiel: $0011\ 0110\ 0101 = 365$

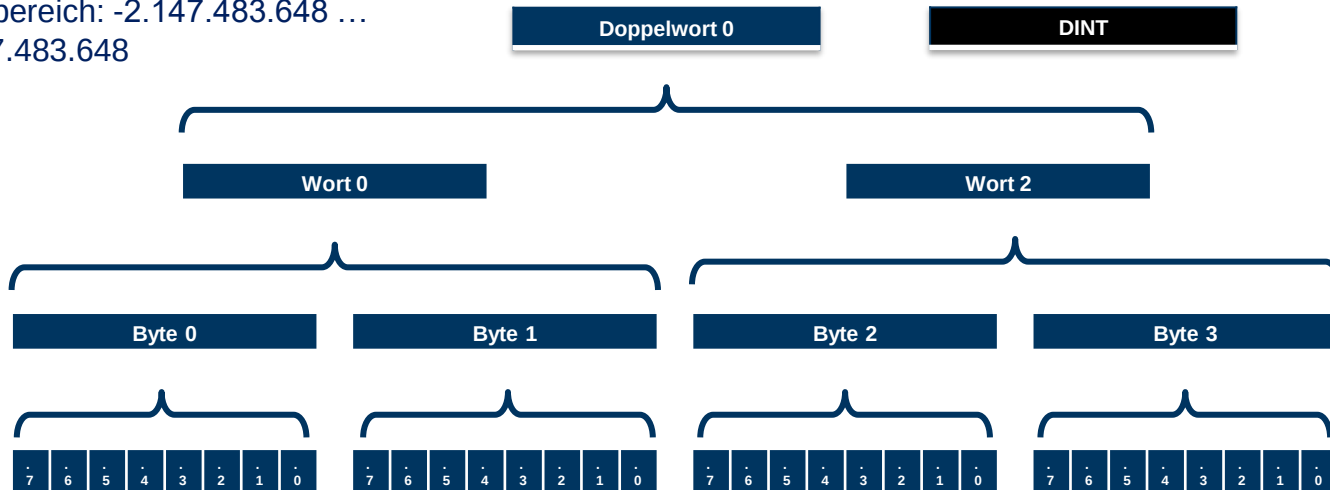


- Den unterschiedlichen Speicherbereichen müssen wir beim programmieren auch noch Datentypen (über 50 verschiedene) zuweisen.
- Datentypen definieren die Länge und den Inhalt eines Speicherbereiches.
- Die wichtigsten Datentypen

Datentyp	Beschreibung	Länge	Wertebereich
BOOL	Boolescher Wert	1 Bit	TRUE oder FALSE
INT	Ganzzahl mit Vorzeichen	16 Bit	-32.768 ... +32.767
DINT	Ganzzahl mit Vorzeichen	32 Bit	-2.147.483.648 ... +2.147.483.647
REAL	Gleitpunktzahl (Kommazahl)	32 Bit	-3,402823e+38 ... -1,175495e-38 ± 0 +1,175495e-38 ... 3,402823e+38
CHAR	Zeichen	8 Bit	ASCII-Zeichensatz
S5TIME	Zeitdauer S5#-Format	16 Bit	S5T#0H_0M_0S_0MS ... S5T#2H_46M_30S_0MS
TIME	Zeitdauer IEC-Format	32 Bit	T#-24d20h31m23s648ms ... T#+24d20h31m23s647ms



Datentype: DINT
Beschreibung: Ganzzahl mit Vorzeichen
Wertebereich: -2.147.483.648 ...
+2.147.483.648



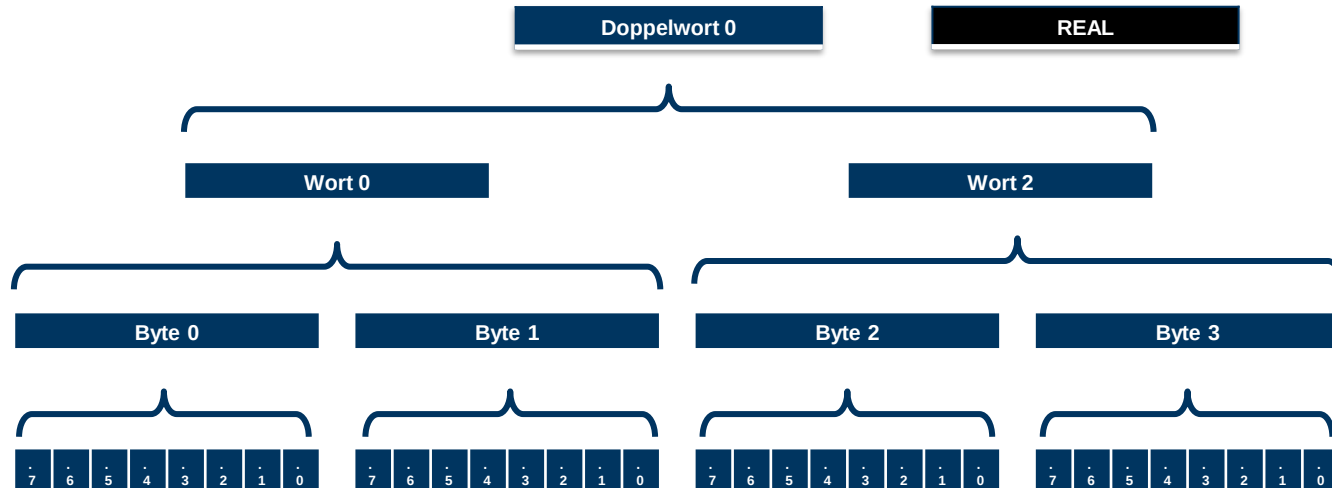
DATENTYPEN



Datentype: REAL

Beschreibung: Gleitpunktzahl

Wertebereich: $-3,402823e+38 \dots -1,175495e-38$
 ± 0
 $+1,175495e-38 \dots 3,402823e+38$



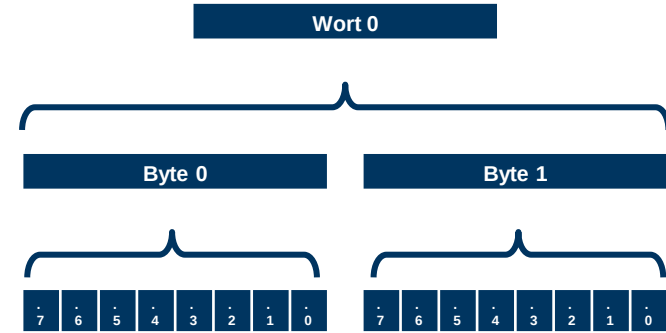


1. Wort 0

- Datentype: INT
- Beschreibung: Ganzzahl mit Vorzeichen
- Wertebereich: -32.768 ... +32.767

2. Wort 0

- Datentype: UINT
- Beschreibung: Ganzzahl ohne Vorzeichen
- Wertebereich: 0 ... 65.535



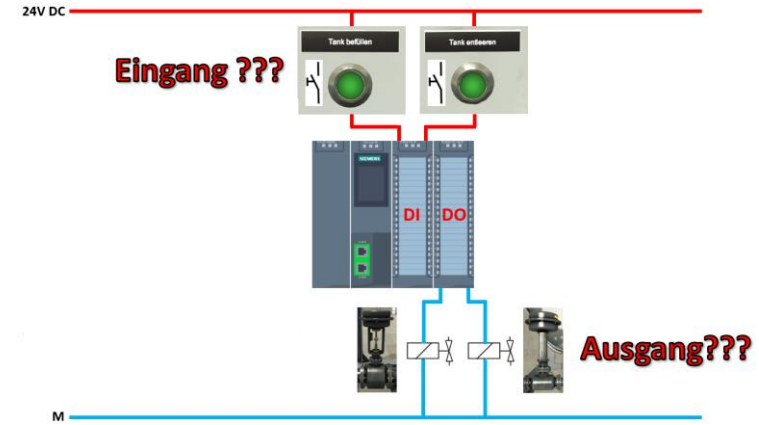


- Beim Programmieren von speicherprogrammierbaren Steuerungen benötigen wir die Eingangs- und Ausgangsadressen der angeschlossenen Sensoren und Aktoren.

Operanden			Beispiel	
	IEC 61131	TIA-Portal	IEC 61131	TIA-Portal
Binärer Eingang	I	E	% IX1.2	% E1.2
Eingangsbyte			% IB0	% EB0
Eingangswort			% IW7	% EW7
Eingangsdoppelwort			% ID4	% ED4
Binärer Ausgang	Q	A	% QX2.1	% A2.1
Ausgangsbyte			% QB12	% AB12
Ausgangswort			% QW6	% AW6
Ausgangsdoppelwort			% QD4	% AD4



- Beim Programmieren von speicherprogrammierbaren Steuerungen benötigen wir die Eingangs- und Ausgangsadressen der angeschlossenen Sensoren und Aktoren.
- Beispiel: Füllstand eines Tanks



<https://docs.factoryio.com/manual/scenes/filling-tank/>

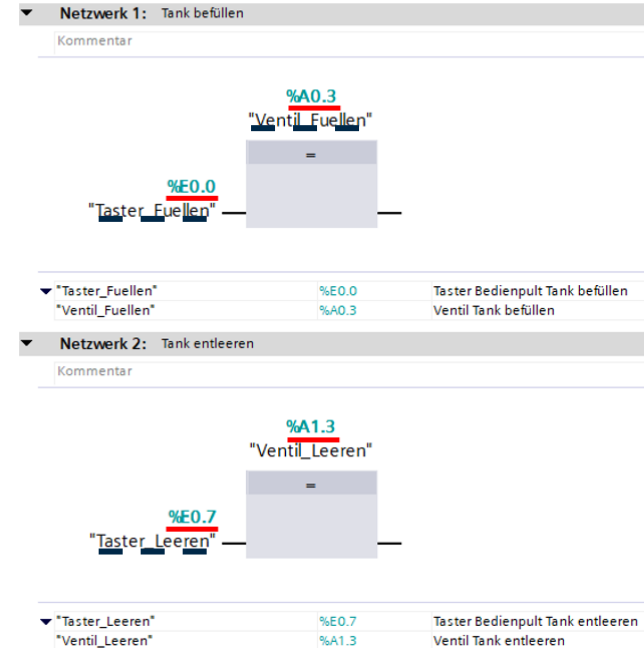


- Beim Programmieren von speicherprogrammierbaren Steuerungen benötigen wir die Eingangs- und Ausgangsadressen der angeschlossenen Sensoren und Aktoren.





■ Absolute/Symbolische Adressierung





- Jeder SPS-Eingang oder Ausgang hat im TIA-Portal einen symbolischen Namen.
- Diesen legen wir in den PLC-Variablen an.



PLC-Variablen						
		Name	...	Datentyp	Adresse	Kommentar
1		Taster_Fuellen	...	Bool	%E0.0	Taster Bedienpult Tank befüllen
2		Taster_Leeren	...	Bool	%E0.7	Taster Bedienpult Tank entleeren
3		Ventil_Fuellen	...	Bool	%A0.3	Ventil Tank befüllen
4		Ventil_Leeren	...	Bool	%A1.3	Ventil Tank entleeren



PLC-Variablen						
		Name	...	Datentyp	Adresse	Kommentar
1		Taster_Fuellen	...	Bool	%I0.0	Taster Bedienpult Tank befüllen
2		Taster_Leeren	...	Bool	%I0.7	Taster Bedienpult Tank entleeren
3		Ventil_Fuellen	...	Bool	%Q0.3	Ventil Tank befüllen
4		Ventil_Leeren	...	Bool	%Q1.3	Ventil Tank entleeren



Ein- und Ausgänge können auch Byte-, Wort- und Doppelwortweise angesprochen werden.

