Zusammenfassung SPS Grundkurs Kapitel 2

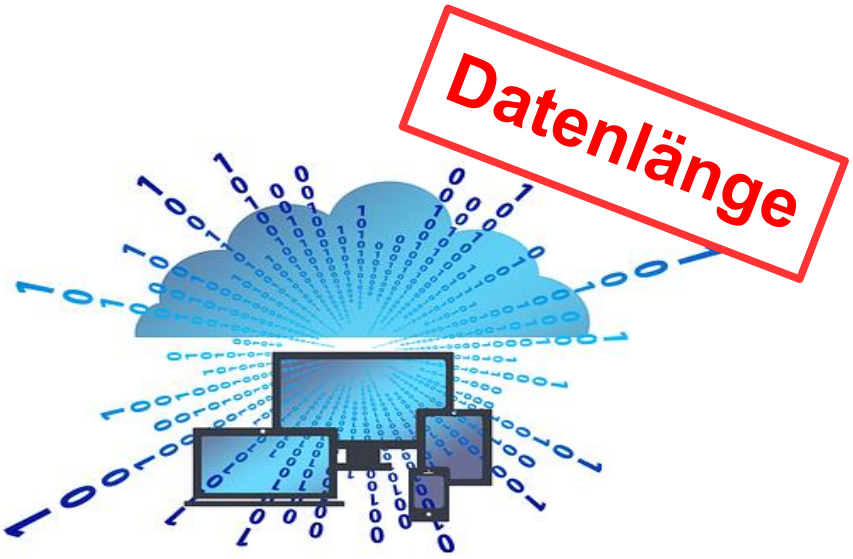
www.spskurs.de

sps4you

Datenlänge

In der SPS-Programmierung arbeiten wir mit unterschiedlich langen Daten.

- ✓ **Bit**
- ✓ **Byte**
- ✓ **Wort**
- ✓ **Doppelwort**



Was ist ein „Bit“?

Ein Bit ist die kleinste elektronische Speichereinheit in der Informationstechnik.

Ein Bit kann nur zwei Zustände einnehmen:

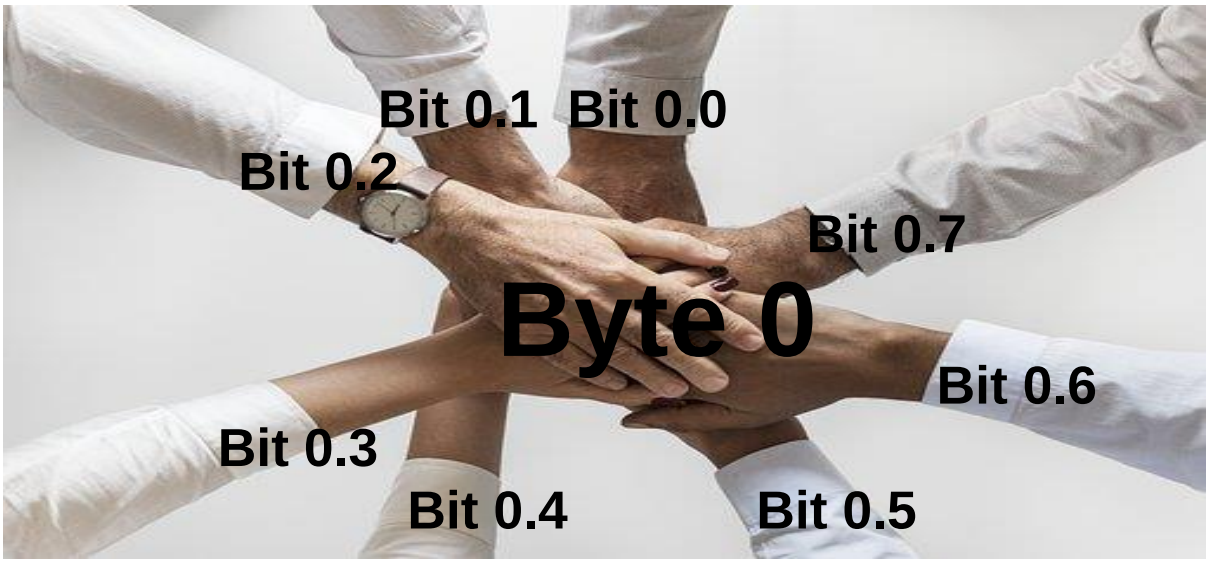
→ EIN/ON/wahr/true (Signalzustand 1)

→ AUS/OFF/falsch/false (Signalzustand 0)



Die CPU organisiert die einzelnen Bits in Gruppen, sogenannte „Bytes“.

Ein Byte besteht auf 8 Bits.



Eine Gruppe von 2 Bytes wir als „Wort“ bezeichnet. Ein Wort besteht also aus 16 Bit.

Eine Gruppe von 2 Wörtern wir als „Doppelwort“ bezeichnet. Ein Doppelwort besteht also aus 32 Bit.

Zahlensysteme

Folgende Zahlensysteme müssen wir als SPS-Anfänger zunächst kennen.

- ✓ Duales Zahlensystem
- ✓ Dezimales Zahlensystem
- ✓ Hexadezimales Zahlensystem
- ✓ BCD Zahlensystem

Wenn wir an Zahlen denken, denken wir für gewöhnlich im dezimalen Zahlensystem.

Das dezimale Zahlensystem besteht aus den Ziffern 0-9.

Beispiel:

Das Jahr hat 365 Tage. 365 ist eine Dezimalzahl.



Jeder Computer arbeitet nach dem dualen Zahlensystem oder auch Binärsystem.

Das duale Zahlensystem kennt nur die Ziffern 0 und 1.

Die Zahl 365 im dualen Zahlensystem sieht so aus:

101101101



Der große Nachteil des dualen Zahlensystems ist die große Bitlänge, die selbst für kleine Zahlen benötigt wird.

(365 = 101101101)

Deswegen hat sich das Hexadezimale Zahlensystem etabliert.

Zusätzlich zu den Ziffern 0-9 (siehe dezimales Zahlensystem), verfügt das hexadezimale Zahlensystem auch über die Buchstaben A – F.

Ziffern: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

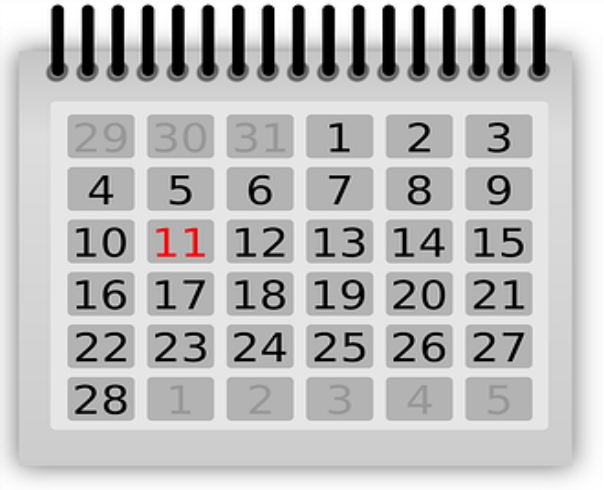
Beispiel:

Das Jahr hat 365 Tage.

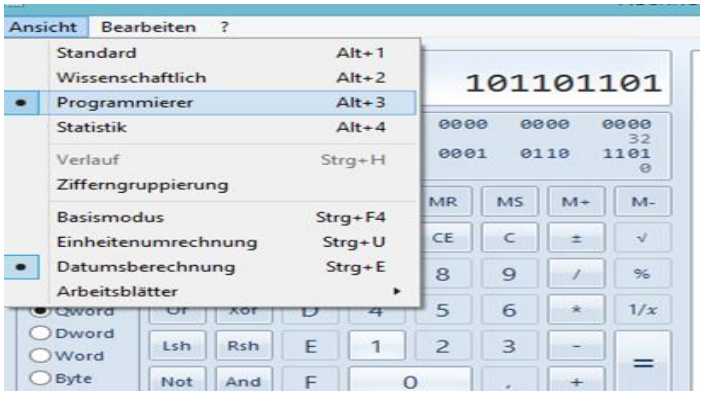
Dezimal: 365

Dual: 101101101

Hexadezimal: 16D

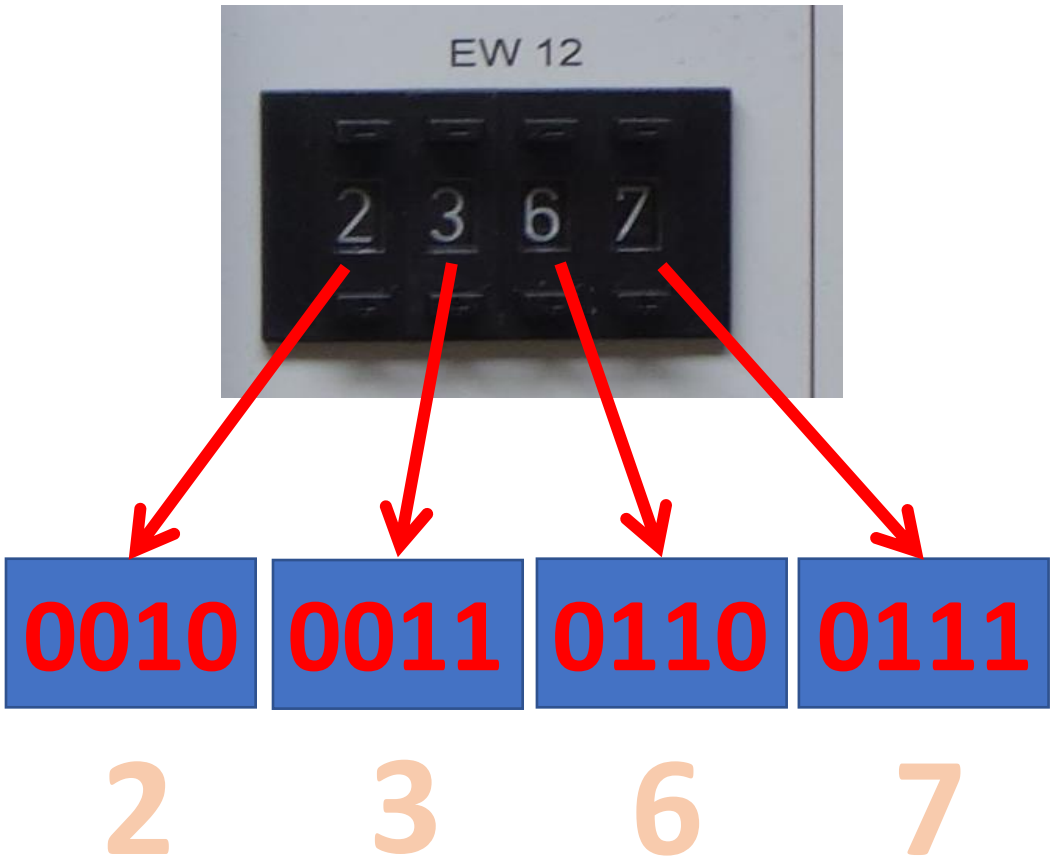


Das Umrechnen der verschiedenen Zahlensysteme erfolgt am besten mit dem Windows Taschenrechner in der Ansicht „Programmierer“



Beim BCD-Zahlensystem handelt es sich um eine binär codierte Dezimalzahl.

Dieses Zahlensystem wir meist an der Schnittstelle zwischen Mensch und Maschine eingesetzt.



Hier sehen wir zum Beispiel einen BCD-Ziffernschalter.

Der Bediener stellt eine Dezimalzahl ein, welche dann von der SPS im BCD-Code erfasst wird.

Zusammenfassung SPS Grundkurs Kapitel 2

www.spskurs.de

sps4you

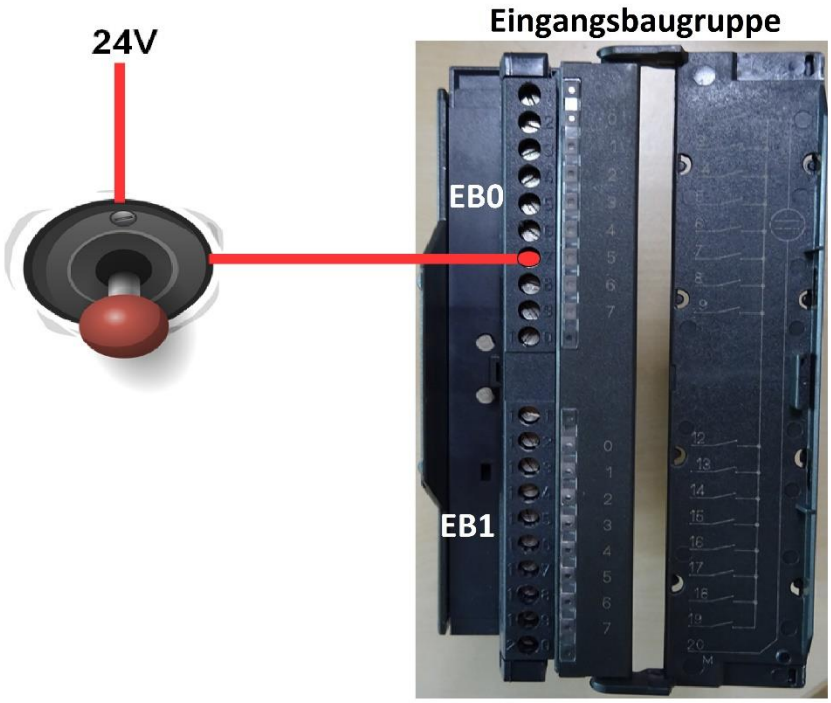
Bei der Programmierung einer SPS hat man es mit unterschiedlichen Datentypen zu tun.

Ein Datentyp gibt die Länge und die Struktur von Daten an.

Datentyp	Bezeichnung	Größe	Bereich
BOOL	Boolescher Wer	1 Bit	False (0), True (1)
BYTE	Dual- und Hexadezimalzahlen mit 8 Bits	8 Bit	B#16#00 bis B#16#FF
WORD	Dual- und Hexadezimalzahlen mit 16 Bits	16 Bit	W#16#0000 bis W#16#FFFF
DWORD	Dual- und Hexadezimalzahlen mit 32 Bits	32 Bit	DW#16#00000000 bis DW#16#FFFFFFFF
INT	Ganze Zahlen mit Vorzeichen (Festpunktzahlen)	16 Bit	-32768 bis +32767
DINT	Ganze Zahlen mit Vorzeichen (Festpunktzahlen)	32 Bit	-2147483648 bis +2147483647
REAL	Reelle Zahlen (Gleitpunktzahlen)	32 Bit	Dezimalzahl mit Punkt (156,235) Exponentialdarstellung (1.568 E+04)
S5TIME	Zeitdauer im S5T#-Format Step7 Zeit in Schritten von 10ms	16 Bit	S5T#0H_0M_0S_10MS bis S5T#2H_46M_30S_0MS
TIME	Zeitdauer im IEC-Format IEC-Zeit in Schritten von 1ms	32 Bit	-T#24D_20H_31M_23S_648MS T#24D_20H_31M_23S_647MS

Adressierung

Um die Sensoren und Aktoren im Programm ansprechen zu können, benötigen wir deren genaue Adresse.



Modul: SM321 16 x DI

→ 16 digitale Eingänge bzw 16 Bit

Zugriff auf einzelnen Eingang:

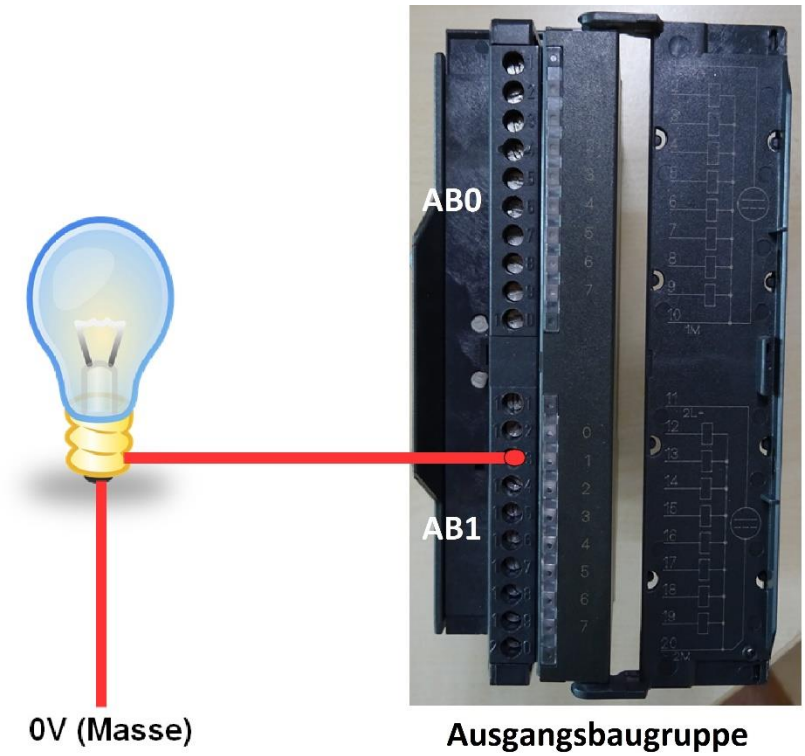
z.B. ***E0.5 (siehe Bild)***

Zugriff auf ganzes Byte

z.B. ***EB0***

Zugriff auf ganzes Wort

z.B. ***EW0***



Modul: SM322 16 x DO

→ 16 digitale Ausgänge bzw 16 Bit

Zugriff auf einzelnen Ausgang:

z.B. ***A1.1 (siehe Bild)***

Zugriff auf ganzes Byte

z.B. ***AB0***

Zugriff auf ganzes Wort

z.B. ***AW0***

Merker

Merker dienen im Programm um Zwischenergebnisse zu speichern.

Merker werden sehr häufig verwendet und genau wie Ein- und Ausgänge adressiert.

Merkerbit: M11.3

Merkerbyte: MB124

Merkerwort: MW10

Merkerdoppelwort: MD2



Zeiten

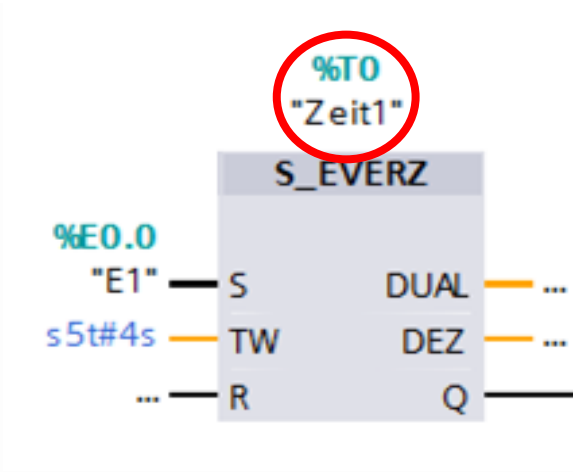
In der Automatisierungstechnik dreht sich viel um Zeiten. Es kommen verschiedene Zeitglieder zum Einsatz.

z.B. Einschaltverzögerung, Ausschaltverzögerung, Impulse, etc

Auch diese Zeitglieder müssen adressiert werden.

Für Simatic-Zeiten wird der Buchstabe „T“ verwendet.

z.B. T1, T12, T56



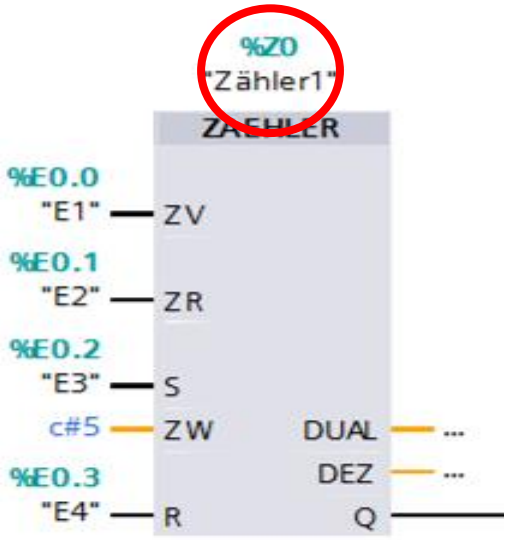
Zähler

Ebenso müssen oft Zählfunktionen genutzt werden.

z.B. Werkstückzähler, Hübe einer Presse, Ausschusszähler

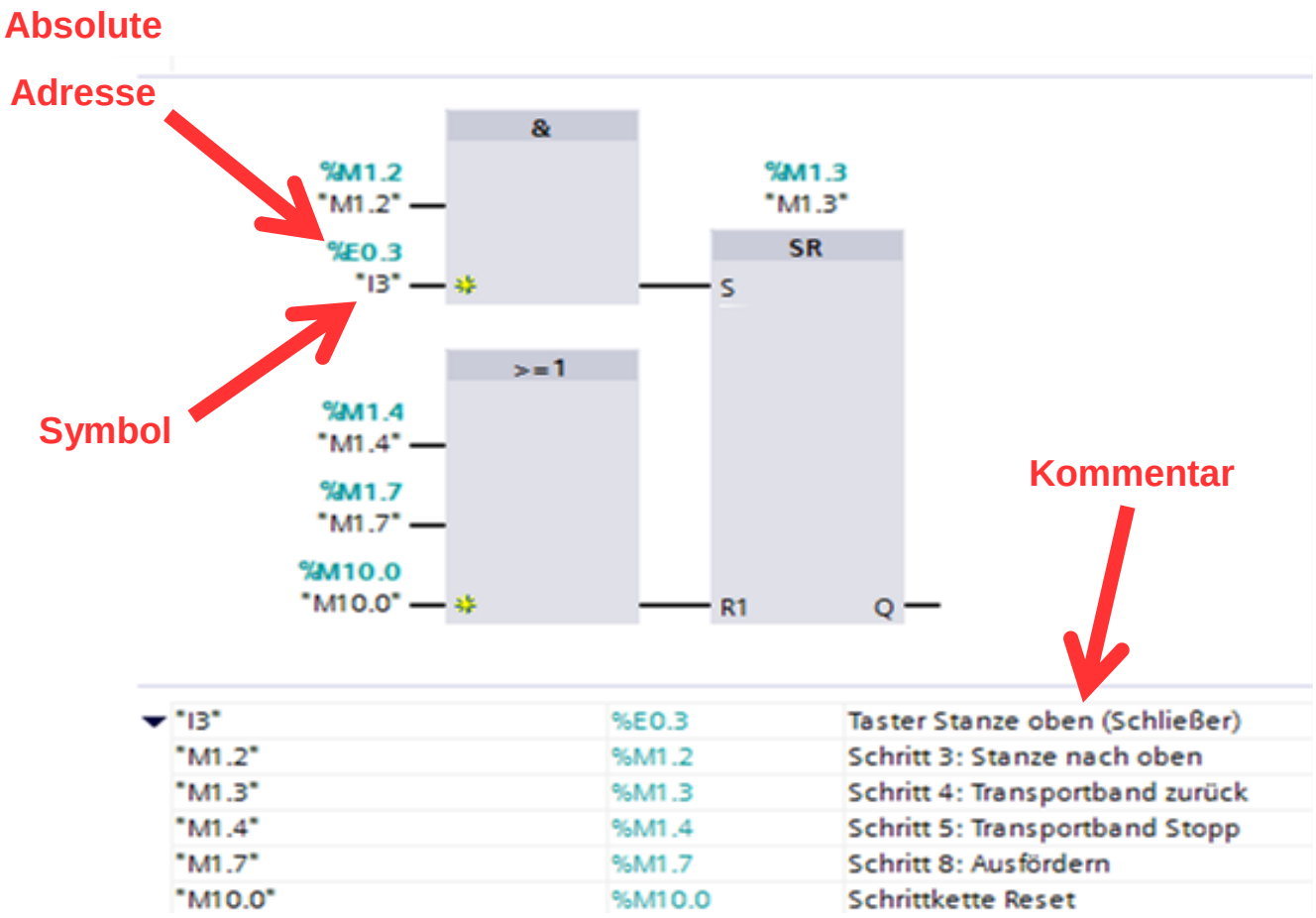
Simatic-Zähler werden mit dem Buchstaben „Z“ adressiert

z.B. Z2, Z12, Z34



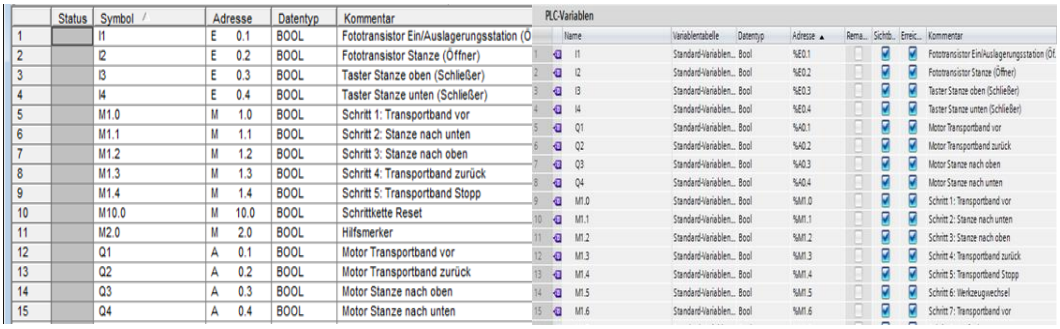
Symbolische / Absolute Adressierung

Wir können symbolisch (z.B. I3) und absolut (z.B. E0.3) programmieren.



Der Trend geht klar zur symbolischen Programmierung.

Um überhaupt symbolisch programmieren zu können müssen wir eine **Symboliktabelle** (Step 7 V5.x) bzw eine **PLC-Variablentabelle** (TIA-Portal) erstellen.



	Status	Symbol	Adresse	Datentyp	Kommentar
1		I1	E 0.1	BOOL	Fototransistor Ein/Auslagerungsstation (Ö
2		I2	E 0.2	BOOL	Fototransistor Stanze (Öffner)
3		I3	E 0.3	BOOL	Taster Stanze oben (Schließer)
4		I4	E 0.4	BOOL	Taster Stanze unten (Schließer)
5		M1.0	M 1.0	BOOL	Schritt 1: Transportband vor
6		M1.1	M 1.1	BOOL	Schritt 2: Stanze nach unten
7		M1.2	M 1.2	BOOL	Schritt 3: Stanze nach oben
8		M1.3	M 1.3	BOOL	Schritt 4: Transportband zurück
9		M1.4	M 1.4	BOOL	Schritt 5: Transportband Stopp
10		M10.0	M 10.0	BOOL	Schrittkette Reset
11		M2.0	M 2.0	BOOL	Hilfsmerker
12		Q1	A 0.1	BOOL	Motor Transportband vor
13		Q2	A 0.2	BOOL	Motor Transportband zurück
14		Q3	A 0.3	BOOL	Motor Stanze nach oben
15		Q4	A 0.4	BOOL	Motor Stanze nach unten

PLC-Variablen

	Name	Variablentabelle	Datentyp	Adresse	Rema...	Sichtb...	Erreic...	Kommentar
1	I1	Standard-Variablen...	Bool	%E0.1		☑	☑	Fototransistor Ein/Auslagerungsstation (Öf
2	I2	Standard-Variablen...	Bool	%E0.2		☑	☑	Fototransistor Stanze (Öffner)
3	I3	Standard-Variablen...	Bool	%E0.3		☑	☑	Taster Stanze oben (Schließer)
4	I4	Standard-Variablen...	Bool	%E0.4		☑	☑	Taster Stanze unten (Schließer)
5	Q1	Standard-Variablen...	Bool	%A0.1		☑	☑	Motor Transportband vor
6	Q2	Standard-Variablen...	Bool	%A0.2		☑	☑	Motor Transportband zurück
7	Q3	Standard-Variablen...	Bool	%A0.3		☑	☑	Motor Stanze nach oben
8	Q4	Standard-Variablen...	Bool	%A0.4		☑	☑	Motor Stanze nach unten
9	M1.0	Standard-Variablen...	Bool	%M1.0		☑	☑	Schritt 1: Transportband vor
10	M1.1	Standard-Variablen...	Bool	%M1.1		☑	☑	Schritt 2: Stanze nach unten
11	M1.2	Standard-Variablen...	Bool	%M1.2		☑	☑	Schritt 3: Stanze nach oben
12	M1.3	Standard-Variablen...	Bool	%M1.3		☑	☑	Schritt 4: Transportband zurück
13	M1.4	Standard-Variablen...	Bool	%M1.4		☑	☑	Schritt 5: Transportband Stopp
14	M1.5	Standard-Variablen...	Bool	%M1.5		☑	☑	Schritt 6: Werkzeugwechsel
15	M1.6	Standard-Variablen...	Bool	%M1.6		☑	☑	Schritt 7: Transportband vor

Step 7 V5.x **TIA-Portal**

Programmiersprachen

In Step 7 gibt es folgende Programmiersprachen:

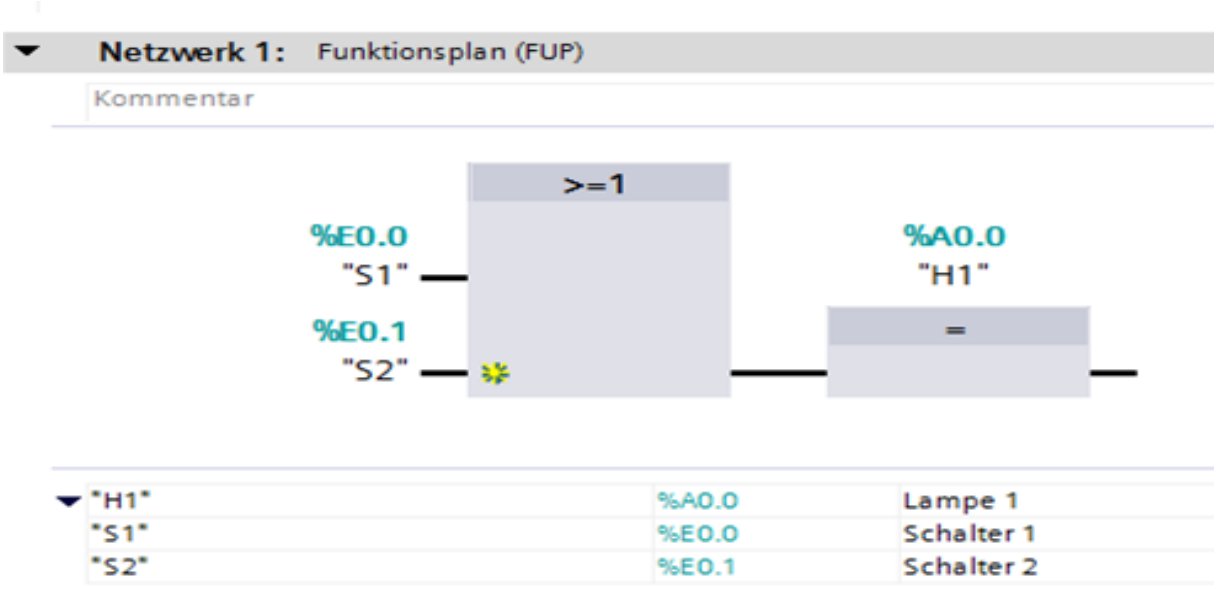
- ✓ Funktionsplan (FUP)
- ✓ Kontaktplan (KOP)
- ✓ Anweisungsliste (AWL)
- ✓ Structured Control Language (SCL)
- ✓ Graph

Funktionsplan (FUP)

Der Funktionsplan (FUP) oder auch Funktionsbausteinsprache (FBS) ist eine grafische Programmiersprache.

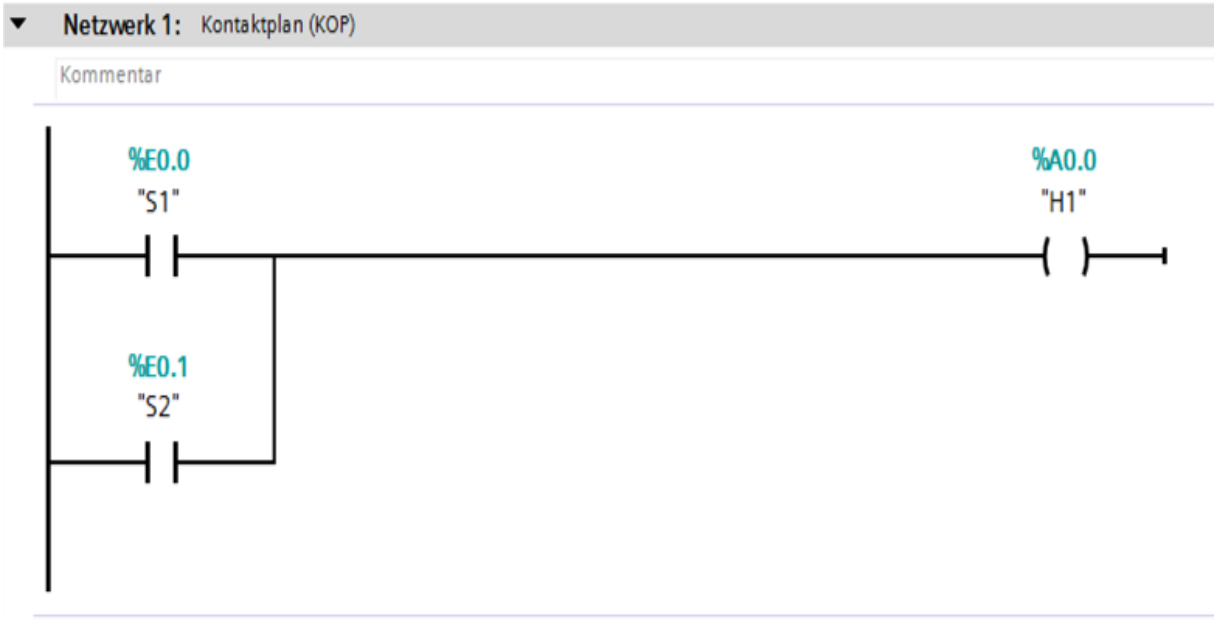
Durch die Programmierung mit Logikbausteinen ist FUP besonders für Anfänger geeignet.

FUP hat einige Vorteile gegenüber anderen Programmiersprachen. Deshalb würde ich euch raten, standardmäßig mit FUP zu arbeiten.



Kontaktplan (KOP)

Der Kontaktplan (KOP) ist eine grafische Programmiersprache und erinnert stark an einen elektrischen Stromlaufplan. Dies kann, vor allem für Elektriker, von Vorteil sein.

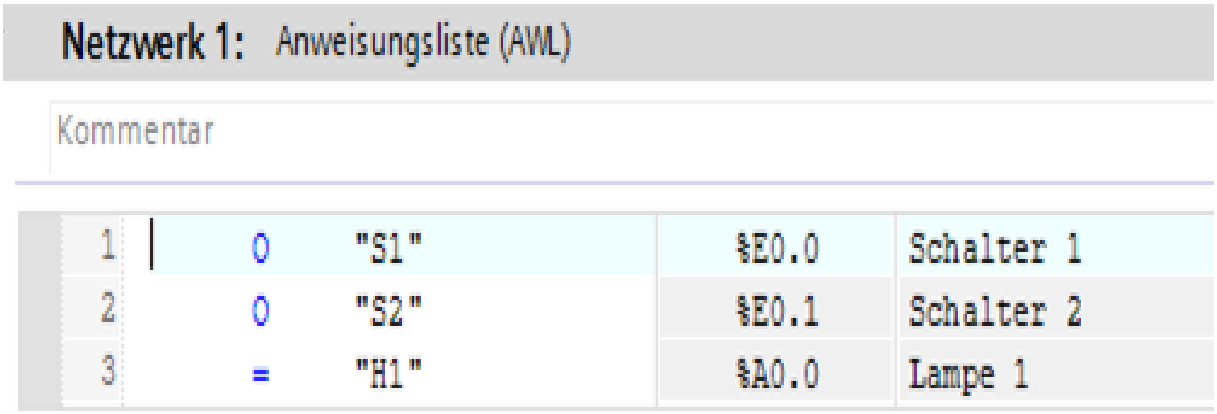


Anweisungsliste (AWL)

Im Gegensatz zu FUP wird bei AWL nicht grafisch programmiert. Das Programm wird in Textform erstellt.

Einige Anweisungen lassen sich ausschließlich mit AWL programmieren bzw sind mit AWL deutlich leichter zu realisieren als mit FUP. Mittlerweile gibt es aber für fast alle AWL-Anweisungen einen FUP-Baustein.

Siemens Simatic 1200-Steuerungen können nicht mit AWL programmiert werden.



Structured Control Language (SCL)

SCL bietet mehr Strukturiermöglichkeiten als AWL. Es handelt sich um eine textbasierte Programmiersprache. SCL wir z.B. für komplexere mathematische Funktionen und Datenverwaltung eingesetzt und wird hier deshalb nur am Rande erwähnt.

Als Anfänger könnt ihr SCL zunächst vernachlässigen.

Graph

Graph ist eine grafisch programmierbare Ablaufsprache. Das Programm besteht aus einer Kette von einzelnen Schritten und Weiterschaltbedingungen (Transitionen).

S7 Graph eignet sich sehr gut zur Programmierung von automatisierten, lineare Abläufen. Für Anfänger, zunächst wenig relevant.