1. Steuerungen

1.1 Allgemein

- SPS = kostengünstig, effizient
- Anforderungen:
  - E/A von Prozesssignalen
  - Echtzeitfähigkeit
  - Sicherheit und Zuverlässigkeit
  - Widerstandsfähigkeit gegen Umwelteinflüsse
- Programmierung gemäß IEC 61131-3

1.1.1 Aufbau

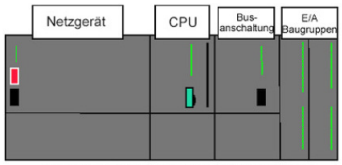
- Kompakt SPS = Kleinsteuerung alles ein Block
- Modulare SPS = Größere Steuerung mit Aufsteckmodulen mit einzelnen Aufgaben
- Integierte SPS = Einbau SPS in andere Steuerung (zB Slot SPS für PC Einbau)
- Soft SPS = Programmbearbeitung durch Standardhardware (PC) auf einem Echtzeit-Kernel, Kommunikation über Karten idR Feldbus

Minimalkonfig einer SPS:

- Zentraleinheit mit Speicher und Prozessor (CPU)
- Eine E/A Ausgabeeinheit (binär)
- Eine Stromversorgungseinheit

Erweiterung:

- Zusätzliche E/A (auch Bus oder Analog)
- Spezialbaugruppen (zB Regler)



Beispielkonfiguration einer modularen SPS

Slot SPS

- Vorteile bei eigener CPU auf Karte: Läuft auch wenn PC ausgeschaltet ist

1.1.2 Ein-/Ausgabe von Prozesssignalen

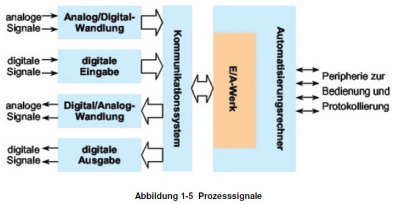


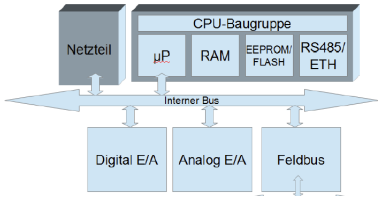
Abbildung 1-5 Prozesssignale

- Aufbau einer Steuerung:

Eingabe- + Vearbeitungs- + Ausgabeeinheit

- Aufbau Eingabeeinheit
  1.) RC Glied = Endstörung mit Zeitkonstante länger als erwartete Störung => zusätzliche Verzögerung (1ms)
  2.) Optokoppler = Galvanische Trennung
  3.) Schwellwertschalter (Schmitt Trigger) = Verhindert mehrfache Signalwechsel beim Einlesen (typ 8-10V mit Hysterese von 2 V)

- Aufbau Verarbeitungseinheit



- Serielle Programmbearbeitung durch µProzessor
- Merker = statische, globale Variablen
- Mittlere typische Zykluszeit = Leistungindikator, gemessen bei Programm mit 1024 Befehlen
- Sonderfall: FPGA als Rechner (=> echte Parallelisierung mit sehr geringen Programmlaufzeiten)

- Aufbau Ausgabeeinheit
  - Kleine Leistungen schalten: Transistoren
  - Große Leistungen schalten: Leistungsrelais
  - Schutzbeschaltung / Löschglied gegen Spannungsspitzen beim Schalten nötig
    => Parallelschalten eines Schutzelements zur ind. Last
    a)Varistor Löschglied:Mittlere Entstörung & Abfallzeit
    b)RC Löschglied:
    c)Dioden Löschglied: Beste Entstörung, langsam
    => Verzögerungen von 20-30ms durch Löschglied

1.1.3 Prinzipieller Ablauf innerhalb einer Steuerung

Zykluszeit = Programmbearbeitungszeit + Systemzeit (= Eing. lesen, Ausg. setzen, Datentransport)

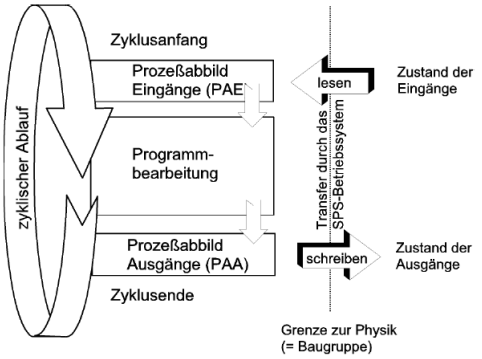


Abbildung 1-8 Ablauf der Steuerung

- Eingänge einlesen nötig um schnelleren Zugriff und einen konsistenten Zustand für das gesamte Programm zu haben
- Reaktionszeit (= Änderung am Eingang führt bis Reaktion am Ausgang) liegt ca bei
  „Zykluszeit+Eingangsverzögerung < Reaktionszeit < 2* Zykluszeit+Eingangsverzögerung“
  - evtl zusätzliche Verzögerung durch Ausgänge
  - Schwankungen der Reaktionszeit besonders bei schnellen Bewegungen mit Stopp durch externes Signal bemerkbar
- Endlosschleife in SPS => keine Ausgänge werden mehr geschalten(Ablauf hängt in Programmbearbeitung)
- Reduktion von Reaktionszeiten durch
  - Zeitgesteuerter Alarm
  - Hardware-Interrupts (Alarm)
  - Programm in HDW

2. Programmierung nach IEC61131-3

2.1 Entwicklung der Programmiersprachen

- DIN EN 61131 Teil 3 seit 1994
- Programmiersprachen:
  - Anweisungsliste AWL
  - Strukturierter Text ST
  - Kontaktplan KOP
  - Funktionsplan FUP (Function Block Diagram FBD)
  - Ablaufsprache AS (Sequential Function Chart SFC)
- Sprachenumsetzung in die Steuerung mittels
  - Interpreter (für alle HW Konfigs verwendbar)
  - Compiler (Vorteil: schneller)

2.2 Klassischer Programmentwurf

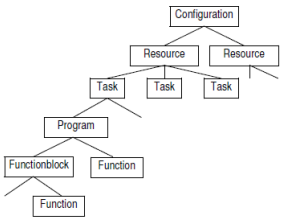
- Nur sinnvoll für wenige Eingänge (4-5)
- Vorgehen
  1.) Funktions- / Wahrheitstabelle aufstellen
  2.) Minimieren
  3.) Umsetzung
- Heute: Minimierung selten, da Aufwand für Minimierung > Einsparung
- Umsetzung ins Programm durch Prioritäten, Zykluszeit und Alarme
- Beispiel:

Lüfter1 muss laufen wenn einer der Rauchmelder zuviel Rauch (Ri =1) meldet. Die Lüfter 1 und 2 müssen laufen wenn zwei Melder Alarm melden.
Alle drei Lüfter laufen wenn alle drei Rauchmelder auf Gefahr im Tunnel hinweisen.

R1	R2	R3	L1	L2	L3	!L1
0	0	0	0	0	0	1
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	1	0	0
1	0	0	1	0	0	0
1	0	1	1	1	0	0
1	1	0	1	1	0	0
1	1	1	1	1	1	0

2.3 Programmstruktur nach IEC 61131

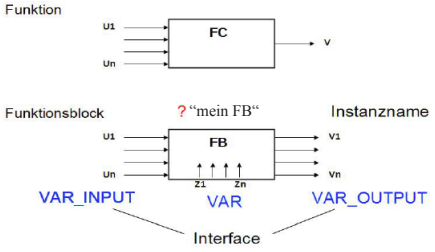
- Projektebene (Configuration)
- Steuerungen (Resource)
- Programmebene (POE, Program Objekt Element) mit Tasks (= mehrere Programme) mit Prioritäten und Zykluszeiten



2.3.1 Programm Objekt Elemente (POE)

Arten von POEs

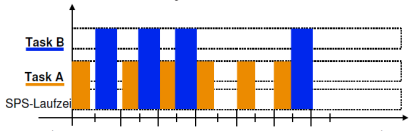
- Program (PROG) = Hauptmodul zum Aufruf der Funktionen und Blöcke
- Function Block (FB) = Unterprogramm mit Übergabe- und Rückgabeparametern (Interface)
  - Können Informationen speichern (static)
  - Instanzierung der FBs mit Namen
  => Mehrfachverwendung möglich
- Function (FC) = Funktion ohne statischen, lokalen Daten (=> kein Wertespeicher), Rückgabewert = Funktionsname



2.3.2 Taskkonfiguration

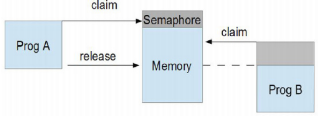
- Aufruf: Tasks => Programme => FBs & FCs
- Multitasking OS in SPS => Zykluszeitüberwachung für unterschiedliche Tasks, bei Überschreitung Stop und alle Ausgänge auf Null
- Aktivierungsarten für Tasks
  - Zyklisch (in festem Zeitraster)
  - Freilaufend (so schnell wie möglich, abhängig vom Programm)
  - Ereignisgesteuert (durch Alarm = Interrupt)
- Beispiel Gantt-Diagramm

Task A, Priorität 0 (höchste), Zyklus 1ms, Rechenzeit 0,5ms
Task B, Priorität 10, Zyklus 5ms, Rechenzeit 1,5ms

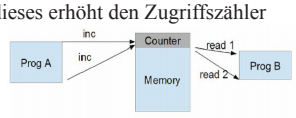


2.3.3 Datenaustausch zwischen Programmen über gemeinsamen Speicher

- Schutz von exklusiv verfügbaren Betriebsmitteln (zB Speicherteil)
- Möglichkeiten
  - Semaphore = Beide können lesen und schreiben



  - Zählvariable = Nur ein Programm darf schreiben, dieses erhöht den Zugriffszähler



Prog B liest zu Beginn und Ende die Zählvariable bei Unterschied muss nochmal probiert werden.

2.4 Programmstruktur nach DIN 19239 (veraltet)

- Lineare Programmierung, für eine Programmierung mit Organisationsbaustein OB1 (Regiebaustein)
- Strukturierte Programmierung, Gesamtprogramm unterteilt in sich abgeschlossene Programmbausteine
- Funktionsbausteine = häufig verwendete Standardfunktionen
- Datenbaustein = Datenspeicher mit Laufnummer
- Vergleich:

DIN 19239	IEC 61131
OB	Tasks
PB	PROG
FB	FB
FC	FC

2.5 Variablen und Datentypen im IEC 61131 Standard

2.5.1 Geltungsbereich und Eigenschaften von Variablen

- Notwendig durch Strukturierung siehe 2.3
- Keine dynamischen Variablen, weil diese zu langsam für Echtzeitfähigkeit sind
- Bestimmung im Deklarationsteil

VAR	Lokale Variable (PROG, FB)
VAR_INPUT	Übergabevariable (Eingang in einen Baustein) modifizierbar ist
VAR_OUTPUT	Ausgangs-Variable (Ausgang aus einem Baustein)
VAR_IN_OUT	Aufruf-Variable, die auch modifizierbar ist (*per Zeiger*)
VAR_EXTERNAL*	Verweis auf Globale Variable, die lokal verwendet werden soll **=nicht in allen Programmiersystemen notwendig*
VAR_GLOBAL	Deklaration von globalen Variablen
VAR_ACCESS	Deklaration des Zugriffspfades für Remote-Variablen (für Netzwerkvariablen = auf einer anderen Steuerung)

- VAR_GLOBAL = Nur innerhalb einer Ressource
- VAR_ACCESS = Mehrere Ressourcen / CPUs
- Unterschied VAR_INPUT <=> VAR_IN_OUT
  VAR_INPUT = Kopie der Variablen
  VAR_IN_OUT = Zugriff über Zeiger (für große Var)
- Qualifaktoren = Spezielle Eigenschaften

PERSISTANT	Variable bleibt bei Programm Download unverändert	VAR, VAR_OUTPUT, VAR_GLOBAL
RETAIN	puffert Werte, bei Spannungsunterbrechung und normalem Aus- und Einschalten, jedoch nicht bei 'Reset Kalt' , 'Reset Ursprung' oder 'Laden'	VAR, VAR_OUTPUT, VAR_GLOBAL
CONSTANT	nicht veränderbare Variable	VAR, VAR_GLOBAL
READ_ONLY	schreibgeschützt	VAR_ACCESS
READ_WRITE	schreib- und lesbar	VAR_ACCESS

Online Befehl	VAR	RETAIN	PERSISTENT	RETAIN PERSISTENT PERSISTENT RETAIN
Reset	-	✓	-	✓
Reset Kalt	-	-	-	-
Reset Ursprung	-	-	-	-
Laden(=Download)	-	-	✓	✓

✓ = Variable unverändert - Variable wird initialisiert

- Beispiele für Retain: Betriebsstundenzähler, Gesamtstückzahl

2.5.2 Variablen

- Programmaufbau: Deklarationsteil, Programteil
- Zuweisung von Variablen im Deklarationsteil
- Hardwareadresszuweisung: AT %
- Art

Kennung	Bedeutung
I	Eingang
Q	Ausgang
M	Merker

- Typkennung

Kennung	Variablenlänge
X	Bit
ohne	Bit
B	Byte (8 Bit)
W	Wort (16 Bit)
D	Doppelwort (32 Bit)
L	Langwort (64 Bit)

- Beispiele:

a) Boolsche Variable ohne feste Adresszuordnung

```
VAR
    Name:BOOL:=optionale_Vorbelegung;
END_VAR
```

b) Boolsche Variable mit fester Adresszuordnung als Eingang bei 0.3

```
VAR
    Name AT %IX0.3:BOOL
END_VAR
```

Namenskonvention:
Die Bezeichnung der Variablen sollte eine eindeutige Beschreibung des Zustands (Aktion) für das „1“-Signal des Eingangs (Ausgangs) wiedergeben.
Bsp: Endschalter Jalousie (Öffner, E=0 wenn Offen)
=> Jalousie_nicht_offen

zu 2.5.2 Variablen

- Bei Objektnamen Präfix GROSS
- Bei Variablen- und Instanznamen Präfix KLEIN
  Präfix_Objektname

Objekt- /Typbezeichner

Objekt	Präfix
FUNCTION_BLOCK	FB_
STRUCT	ST_
ENUM	E_
TYPE	T_
PROGRAM	P_
FUNCTION	F_

Instanznamen

Objekt	Präfix
function block	fb
struct	st
enum	e
alias type	kein

Variablennamen

Typ	Präfix
SINT, USINT, ..., DINT, UDINT, BYTE, WORD, DWORD, LWORD....	n, i
BOOL	b
REAL, LREAL	f
STRING	s
TIME	t
DATE	d
DATE_AND_TIME	dt
ARRAY[...] OF ...	arr
p	p

2.5.3 Standarddatentypen

BOOL	Boolescher Wert H oder L	TRUE, FALSE	ANY BOOL
BYTE	Bitkette der Länge 8	0 ... 255	
WORD	Bitkette der Länge 16	0 ... 65535	
DWORD	Bitkette der Länge 32	0 ... 4294967295	
LWORD	Bitkette der Länge 64		
SINT	Kurze ganze Zahl 8 Bit	-128 ... +127	ANY INT
INT	Festkommazahl 16 Bit	-32768 ... +32767	
DINT	Doppelte Festkommazahl 32 Bit	-2147483648 ... +2147483647	
LINT	Lange Festkommazahl 64 Bit	-2^{63} ... $+2^{63}-1$	
USINT	Kurze Festkommazahl ohne Vorzeichen 8 Bit	0 ...255	
UINT	Festkommazahl ohne Vorzeichen 16 Bit	0 ... 65535	
UDINT	Doppelte Festkommazahl ohne Vorzeichen 32 Bit	0 ... 4294987295	
ULINT	Lange Festkommazahl ohne Vorzeichen 64 Bit	0 ... $2^{64}-1$	
REAL	Gleitkommazahl 32 Bit	1.18×10^{-38} ... 3.40×10^{38}	ANY REAL
LREAL	Lange Gleitkommazahl 64 Bit		
TIME oder T	Zeitdauer		TIME
DATE oder D	Datum		ANY DATE
TIME_OF_DAY oder TOD	Tageszeit		
DATE_AND_TIME oder DT	Datum und Tageszeit		
STRING	Zeichenkette definierter Länge		STRING

2.5.4 Abgeleitete Datentypen

- Deklaration mit TYPE ... TYPE_END
- Aufzählungstypen ENUM
  = Typ INT, fortlaufende Zählung bei 0 außer etwas anderes ist angegeben

```
TYPE
  Ich_bin_der_neue_Datentyp: (Nr1, Nr2:=10 , Nr3:=15, ...);
END_TYPE
```

- Bereichtyp (Range)

```
TYPE
  Ich_bin_der_Bereich: INT (-20..20);
END_TYPE                (*INT steht hier für den Datentyp *)
```

- Felder (Array)
  - Deklaration kann auch im Programmteil erfolgen

```
TYPE
  Ich_bin_ein_Array: ARRAY [Nr_1 .. Nr_n] OF INT;
END_TYPE                (*INT steht hier für den Datentyp *)
```

- Strukturen (Struct)
  Defintion

```
TYPE
  Ich_bin_das_Strukt: STRUCT
  Datentypname_1: INT;
  ...
  Datentypname_n: INT;
  END_TYPE;             (*INT steht hier für den Datentyp *)
```

  Deklaration

```
Strukt_Name : Ich_bin_das_Strukt := ( Datentypname_1 := 1, .. ,
Datentypname_n =: 66);          (*Die Zahl steht hier als Beispiel *)
oder
Strukt_Name. Datentypname_1:=1;
```

  Zugriff

```
Strukt_Name. Datentypname_1
```

- Zeiger (Pointer)

```
Deklaration:
    Name_Zeiger : POINTER TO Datentyp_oder_FB;

Zuweisung in strukturierten Text:
    Name_Zeiger := ADR ( Variablenname);

Zuweisung in Anweisungsliste:
    LD    Variablenname
    ADR
    ST    Name_Zeiger

Dereferenzierung in strukturierten Text:
    Name_Zeiger^;
```

2.6.2 Strukturierter Text ST

Anweisung	Beschreibung
:=	Zuweisung
IF	... wenn ...
CASE	Fall ...
FOR	Zählschleife
WHILE	Abbruchbedingung am Anfang
REPEAT	Abbruchbedingung am Ende
EXIT	verläßt den FB

- Operatoren Prioritäten

Priorität	Operation	Bemerkungen
1	( )	Klammerung
2		Funktionsaufruf
3	EXPT	Potenzieren
4	-	Negieren (unär)
5	NOT	Komplementbildung
6	*	Multiplikation
7	/	Division
8	MOD	Modulo
9	+	Addition
10	-	Subtrahieren (binär)
11	<, >, <=, >=	Vergleiche
12	=	Gleichheit
13	<>	Ungleichheit
14	&, AND	Bool Und
15	XOR	Bool Exklusivoder
16	OR	Bool Oder

Anweisungen

- Syntax: RETURN;
  Beschreibung: Verlassen des Bausteins
- Syntax: EXIT;
  Beschreibung: Verlassen einer Schleife
- Syntax: ;
  Beschreibung: Für das Setzen von Breakpoints

Bedingte Anweisungen:

- IF Anweisung

Beispiel:

```
IF a < b THEN
    c := 0;
END_IF

IF a < b THEN
    c := 0;
ELSIF a = b THEN
    c := 100;
ELSE
    c := 1;
END_IF
```

Syntax:

```
IF <Bedingung1> THEN
    <Anweisung1>
ELSIF <Bedingung2> THEN
    <Anweisung2>
    ...
ELSE
    <Anweisung3>
END_IF;
```

Beschreibung: „Wenn *<Bedingung1>* erfüllt, dann *<Anweisung1>*, wenn *<Bedingung2>*, dann *<Anweisung2>*, usw., ansonsten *<Anweisung3>*.„

- CASE Anweisung

Beispiel:

```
CASE a OF
    1:      c := 10;
    2,5:    c := 20;
    7..9:   c := 30;
ELSE
            c := 0;
END_CASE
```

Syntax:

```
CASE <Variabel> OF
    <Wert_1>:              <Anweisung1>;
    <Wert_2>:              <Anweisung2>;
    <Wert_3>, <Wert_4>:    <Anweisung3>;
    <Wert_5>.. <Wert_10>:  <Anweisung4>;
    ...
    ELSE                   <Anweisung5>;
END_CASE;
```

Beschreibung: „Wenn *<Variabel>* einen *<Wert_n>* hat, dann *<Anweisung_n>*, ansonsten *<Anweisung5>*. Wenn der *<Wert_n>* in den Wertebereich von *<Wert5>* bis *<Wert10>* fällt, dann *<Anweisung4>*. „

- FOR Schleife

Beispiel:

```
FOR i:=1 TO 10 BY 2 DO
    a[i] := TRUE;
END_FOR
```

Syntax:

```
FOR <Variabel> := <Wert> TO <Endwert> BY <Schrittgröße> DO
    <Anweisung>
END_FOR;
```

Beschreibung: „Für *<Variabel>* wird ein *<Wert>* zugewiesen. Die Schrittweite ist optional und kann jeden Integerwert besitzen. Fehlt diese, wird sie auf den Wert 1 gesetzt. Die *<Anweisung>* wird solange ausgeführt, solange der *<Wert>* den *<Endwert>* nicht übersteigt.“

Achtung: Der <Endwert> darf nicht der Grenzwert des Datentyps sein, sonst erfolgt eine Endlosschleife.

- WHILE Schleife

Beispiel:

```
WHILE value1 < 100 DO
    Counter:=Counter+1;
END_WHILE
```

Syntax:

```
WHILE <Bedingung> DO
    <Anweisung>
END_WHILE;
```

Beschreibung: „Solange *<Bedingung>* wahr ist, wird *<Anweisung>* ausgeführt.“

- REPEAT Schleife

Beispiel:

```
REPEAT
    Counter:=Counter+1;
UNTIL Counter = 200
END_REPEAT
```

Syntax:

```
REPEAT
    <Anweisung>
UNTIL <Bedingung>
END_WHILE;
```

Beschreibung: „Solange *<Bedingung>* wahr ist, wird *<Anweisung>* ausgeführt. Die Schleife wird mindestens einmal durchlaufen.“

Aufruf von Funktionen:

Syntax: Variablenname := Funktionsname (Variable_an_Par_1, Variable_an_Par_2);

Aufruf von Funktionsblöcken:

Im Deklarationsteil:

```
VAR
    ...
    Instanzname : Funktionsblockname ;
    ...
END__VAR
```

Im Programmteil:

```
Instanzname (Wert1 := Variable1, Wert2 := Varaible2)
...
WHILE Instanzname . Rückgabe1 DO
...
END_WHILE
```

2.5.6 Standardfunktionsblöcke

A.1.1 RS-Speicher (Rücksetzen vorrangig)

Parameter	Datentyp	Bedeutung
SET	BOOL	Eingang setzen
RESET1	BOOL	Eingang rücksetzen
Q1	BOOL	Ausgang

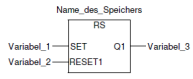
Deklaration:

Name_des_Speichers : RS;

Beispiel in AWL:

```
CAL     Name_des_Speichers (SET:=Variabel_1, RESET1:=Variabel_2)
LD      Name_des_Speichers.Q1
ST      Variabel_3
```

Beispiel in FUP:



Beispiel in ST:

```
Name_des_Speichers (SET:=Variabel_1, RESET1:=Variabel_2);
Variabel_3 := Name_des_Speichers.Q1;
```

A.1.2 SR-Speicher (Setzen vorrangig)

Analog zum RS-Speicher, siehe oben.

A.1.3 Semaphore SEMA

Parameter	Datentyp	Bedeutung
CLAIM	BOOL	Anspruch
RELEASE	BOOL	Freigabe
BUSY	BOOL	Ausgang

Erklärung: Mit einem TRUE Signal an CLAIM wird die Semaphorvariable belegt. War die Semaphore schon vorher belegt, so liefert BUSY *TRUE* zurück. RELAESE setzt BUSY und interne Variabel immer zurück. Sollte (darf) nur von dem Programm gemacht werden, bei dem CLAIM erfolgreich war.

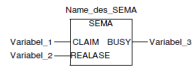
Deklaration:

Name_des_SEMA : SEMA;

Beispiel in AWL:

```
CAL     Name_des_SEMA (CLAIM:=Variabel_1, REALASE:=Variabel_2)
LD      Name_des_SEMA.BUSY
ST      Variabel_3
```

Beispiel in FUP:



Beispiel in ST:

```
Name_des_SEMA (CLAIM:=Variabel_1, REALASE:=Variabel_2);
Variabel_3 := Name_des_Speichers.BUSY;
```

A.1.4 Einschaltverzögerung TON

Parameter	Datentyp	Bedeutung
IN	BOOL	Verzögerung starten
PT	TIME	Verzögerungszeit
Q	BOOL	Ausgang
ET	TIME	verstrichene Zeit

Erklärung: TON, reagiert auf eine steigende Flanke des Eingangs und liefert nach der vorgegebenen Zeit (PT) am Ausgang Q = *TRUE*, wenn der Eingang noch *TRUE* ist . Die verstrichene Zeit kann am Ausgang ET ausgelesen werden. Sobald der Eingang wieder auf *FALSE* geht, wird der Ausgang Q ebenfalls sofort *FALSE*.

Hinweis! Ein TON-FB kann Änderungen natürlich nur dann feststellen, wenn er abgearbeitet (aufgerufen) wird. Es gibt keine versteckten Parallelaktionen zum Programm.

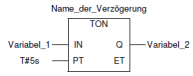
Deklaration:

Name_der_Verzögerung: TON;

Beispiel in AWL:

```
CAL     Name_der_Verzögerung (IN:=Variabel_1, PT:=T#5s)
LD      Name_der_Verzögerung.Q
ST      Variabel_2
```

Beispiel in FUP:



Beispiel in ST:

```
Name_der_Verzögerung (IN:=Variabel_1, PT:=T#5s);
Variabel_2 := Name_der_Verzögerung.Q;
```

A.1.5 Ausschaltverzögerung TOF

Parameter	Datentyp	Bedeutung
IN	BOOL	Verzögerung starten
PT	TIME	Verzögerungszeit
Q	BOOL	Ausgang
ET	TIME	verstrichene Zeit

Erklärung: Sobald an IN eine *fallende Flanke* erscheint (*TRUE -> FALSE*) beginnt die voreingestellte Zeit (PT) zu laufen. Der Ausgang Q bleibt noch bis zum Ablauf der Zeit *TRUE* und wechselt dann erst auf *FALSE*, falls der Eingang noch *FALSE* ist. Die aktuell verstrichene Zeit kann am Ausgang ET ausgelesen werden.

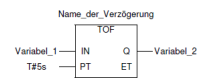
Deklaration:

Name_der_Verzögerung: TOF;

Beispiel in AWL:

```
CAL     Name_der_Verzögerung (IN:=Variabel_1, PT:=T#5s)
LD      Name_der_Verzögerung.Q
ST      Variabel_2
```

Beispiel in FUP:



Beispiel in ST:

```
Name_der_Verzögerung (IN:=Variabel_1, PT:=T#5s);
Variabel_2 := Name_der_Verzögerung.Q;
```

A.1.6 Puls TP

Parameter	Datentyp	Bedeutung
IN	BOOL	Verzögerung starten
PT	TIME	Verzögerungszeit
Q	BOOL	Ausgang
ET	TIME	verstrichene Zeit

Erklärung: Eine *steigende Flanke* an IN setzt den Ausgang Q für die Zeit PT auf *TRUE*. Auch falls das Eingangssignal kürzer ist als die eingestellte Zeit an PT, dann ist Q solange *TRUE* bis die an PT eingestellte Zeit abgelaufen ist.

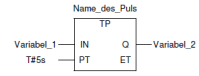
Deklaration:

Name_des_Puls: TP;

Beispiel in AWL:

```
CAL     Name_des_Puls (IN:=Variabel_1, PT:=T#5s)
LD      Name_des_Puls.Q
ST      Variabel_2
```

Beispiel in FUP:



Beispiel in ST:

```
Name_des_Puls (IN:=Variabel_1, PT:=T#5s);
Variabel_2 := Name_des_Puls.Q;
```

A.1.12 Auswahloperator SEL

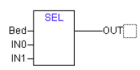
Parameter	Datentyp	Bedeutung
Bed	BOOL	Auswahl-Eingang (Bedingung)
IN0	any	Wert für Bed=FALSE
IN1	any	Wert für Bed=TRUE
OUT	any	Ausgang

Erklärung:

OUT := SEL(Bed, IN0, IN1) bedeutet:

OUT := IN0 if Bed=FALSE;

OUT := IN1 if Bed=TRUE.



IN0, IN1 und *OUT* können jeden Typ haben, *Bed* muss vom Typ BOOL sein. Das Ergebnis der Selektion ist *IN0*, wenn *Bed* FALSE ist, bzw. *IN1*, wenn *Bed* TRUE ist.

zu 2.5.6 Standardfunktionsblöcke

A.1.7 Abwärtszähler CTD

Parameter	Datentyp	Bedeutung
CD	BOOL	Trigger Eingang
LD, LOAD	BOOL	Lade Anfangswert
PV	INT, DINT	Anfangswert
Q	BOOL	Ausgang
CV	INT, DINT	Zählwert

Erklärung: Wenn der Eingang *LOAD TRUE* ist, wird der Startwert vom Eingang PV eingelesen. Sobald an CD eine *steigende Flanke* anliegt, wird der Zähler (CV) um 1 erniedrigt, wenn er noch größer Null ist. (Zähler kann nicht negativ werden!) Sobald CV (Zählwert) gleich 0 ist, wird der Ausgang Q *TRUE*.

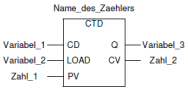
Deklaration:

Name_des_Zaehlers: CTD;

Beispiel in AWL:

```
CAL     Name_des_Zaehlers (CD:=Variabel_1, LOAD:=Variabel_2, PV:=Zahl_1)
LD      Name_des_Zaehlers.Q
ST      Variabel_3
```

Beispiel in FUP:



Beispiel in ST:

```
Name_des_Zaehlers (CD:=Variabel_1, LOAD:=Variabel_2, PV:=Zahl_1)
Variabel_3 := Name_des_Zaehlers.Q;
```

A.1.8 Aufwärtszähler CTU

Parameter	Datentyp	Bedeutung
CU	BOOL	Trigger Eingang
RESET	BOOL	Reset
PV	INT, DINT	Vergleichswert
Q	BOOL	Ausgang
CV	INT, DINT	Zählwert

Erklärung: Wenn RESET *TRUE* ist, wird der Zähler mit 0 initialisiert. Sobald an CU eine *steigende Flanke* anliegt, wird der Zähler (CV) um 1 erhöht. Sobald CV größer gleich PV ist, wird Q *TRUE*.

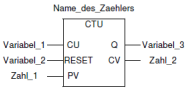
Deklaration:

Name_des_Zaehlers: CTU;

Beispiel in AWL:

```
CAL     Name_des_Zaehlers (CU:=Variabel_1, RESET:=Variabel_2, PV:=Zahl_1)
LD      Name_des_Zaehlers.Q
ST      Variabel_3
```

Beispiel in FUP:



Beispiel in ST:

```
Name_des_Zaehlers (CU:=Variabel_1, RESET:=Variabel_2, PV:=Zahl_1)
Variabel_3 := Name_des_Zaehlers.Q;
```

A.1.9 Auf- Abwärtszähler CTUD

Parameter	Datentyp	Bedeutung
CU	BOOL	Trigger Eingang
CD	BOOL	Trigger Eingang
RESET	BOOL	Reset
LD, LOAD	BOOL	Lade Anfangswert
PV	INT, DINT	Anfangswert
QU	BOOL	Ausgang
QD	BOOL	Ausgang
CV	INT, DINT	Zählwert

Erklärung: Wenn RESET *TRUE* ist, dann wird der Zähler mit 0 initialisiert, falls LOAD *TRUE* ist, wird der Wert an PV eingelesen. Falls beide Eingänge *TRUE* sind, hat der RESET Eingang Vorrang. Wenn eine *steigende Flanke* an CU anliegt, dann wird Wert an CV um 1 erhöht. Bei einer *steigenden Flanke* an CD erniedrigt den Wert an CV um 1, so lang CV nicht 0 ist. Der Ausgang QU ist *TRUE*, wenn CV größer gleich PV ist. Der Ausgang QD ist *TRUE*, so bald der Zählwert CV gleich 0 ist.

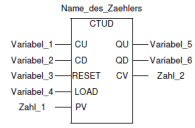
Deklaration:

Name_des_Zaehlers: CTUD;

Beispiel in AWL:

```
CAL     Name_des_Zaehlers (CU:=Variabel_1, CD:=Variabel_2, RESET:=Variabel_3,
   LOAD:=Variabel_4, PV:=Zahl_1)
LD      Name_des_Zaehlers.QU
ST      Variabel_5
LD      Name_des_Zaehlers.QD
ST      Variabel_6
LD      Name_des_Zaehlers.CV
ST      Zahl_2
```

Beispiel in FUP:



Beispiel in ST:

```
Name_des_Zaehlers (CU:=Variabel_1, CD:=Variabel_2, RESET:=Variabel_3,
LOAD:=Variabel_4, PV:=Zahl_1);
Variabel_5:=Name_des_Zaehlers.QU;
Variabel_6:=Name_des_Zaehlers.QD;
Zahl_2:=Name_des_Zaehlers.CV;
```

A.1.10 Steigende Flanke R_TRIG

Parameter	Datentyp	Bedeutung
CLK	BOOL	Trigger Eingang
Q	BOOL	Ausgang

Erklärung: Bei einer *steigenden Flanke* an CLK wird der Ausgang Q auf *TRUE* gesetzt. Beim nächsten Aufruf wird der Ausgang Q zurückgesetzt.

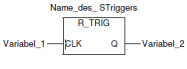
Deklaration:

Name_des_STriggers: R_TRIG;

Beispiel in AWL:

```
CAL     Name_des_STriggers (CLK:=Variabel_1)
LD      Name_des_STriggers.Q
ST      Variabel_2
```

Beispiel in FUP:



Beispiel in ST:

```
Name_des_STriggers (CLK:=Variabel_1)
Variabel_2 := Name_des_STriggers.Q;
```

A.1.11 Fallende Flanke F_TRIG

Parameter	Datentyp	Bedeutung
CLK	BOOL	Trigger Eingang
Q	BOOL	Ausgang

Erklärung: Bei einer *fallenden Flanke* an CLK wird der Ausgang Q auf *TRUE* gesetzt. Beim nächsten Aufruf wird der Ausgang Q zurückgesetzt.

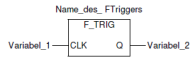
Deklaration:

Name_des_FTriggers: F_TRIG;

Beispiel in AWL:

```
CAL     Name_des_FTriggers (CLK:=Variabel_1)
LD      Name_des_FTriggers.Q
ST      Variabel_2
```

Beispiel in FUP:



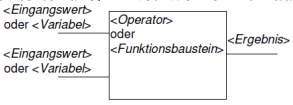
Beispiel in ST:

```
Name_des_FTriggers (CLK:=Variabel_1)
Variabel_2 := Name_des_FTriggers.Q;
```

2.6 Programmiersprachen im IEC 61131 Standard

2.6.1 Funktionsplan FUP

- Unterteilung in Netzwerke mit Bausteinen



- Negierung der Eingänge möglich mit o

2.5.5 Standardoperatoren

- Kommentare: (*....*)

Operatoren	Gültig für	Ergebnis	Beschreibung
Arithmetische Operatoren			
ADD	ANY BOOL; ANY INT; ANY REAL; TIME		Addition
MUL	ANY BOOL; ANY INT; ANY REAL		Multiplikation
SUB	ANY BOOL; ANY INT; ANY REAL; TIME		Subtraktion
DIV	ANY BOOL; ANY INT; ANY REAL		Division
MOD	ANY BOOL; ANY INT; ANY REAL		Modulo
INDEXOF	Baustein	Index	interner Index des Bausteins
SIZEOF	ANY	Bytes	Anzahl Bytes, die der angegebene Datentyp benötigt

Operatoren	Gültig für	Ergebnis	Beschreibung
Bitstring Operatoren			
AND	ANY BOOL		Bitweises AND
OR	ANY BOOL		Bitweises OR
XOR	ANY BOOL		Bitweises XOR
NOT	ANY BOOL		Bitweises NOT

Operatoren	Gültig für	Ergebnis	Beschreibung
Bit-Shift Operationen			
SHL	ANY BOOL		Bitweises Linksverschieben eines Operanden
SHR	ANY BOOL		Bitweises Rechtsverschieben eines Operanden
ROL	ANY BOOL		Bitweise Linksrotation eines Operanden
ROR	ANY BOOL		Bitweise Rechtsrotation eines Operanden
Auswahl-operatoren			
SEL	ANY		Binäre Selektion
MAX	ANY		Sucht den größten Wert
MIN	ANY		Sucht den kleinsten Wert
LIMIT	ANY		Limitierung
MUX	ANY		Multiplexer
Vergleichs-operatoren			
GT >	ANY	TRUE / FALSE	Größer als
LT <	ANY	TRUE / FALSE	Kleiner als
LE <=	ANY	TRUE / FALSE	Kleiner oder gleich
GE>=	ANY	TRUE / FALSE	Größer oder gleich
EQ =	ANY	TRUE / FALSE	Gleichheit
NE <>	ANY	TRUE / FALSE	Ungleichheit

Operatoren	Gültig für	Ergebnis	Beschreibung
Adreß-operatoren			
ADR	DWORD	Adresse	liefert die Adresse eines Argumentes und kann als Pointer behandelt werden
Inhaltsoperator "^"			Die Dereferenzierung eines Pointers erfolgt über den Inhaltsoperator "^" nach dem Pointerbezeichner

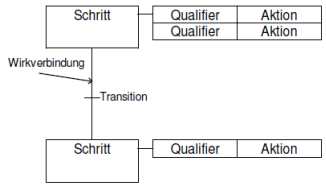
Operatoren	Gültig für	Ergebnis	Beschreibung
Aufrufoperator			
CAL			Aufruf einer Instanz eines Funktionsblockes

Operatoren	Gültig für	Ergebnis	Beschreibung
Typkonvertier-ungen			
BOOL_TO	ANY BOOL; ANY INT; ANY REAL; TIME; STRING	1 / 0 (bei String TRUE / FALSE)	
TO_BOOL	ANY BOOL; ANY INT; ANY REAL; TIME; STRING	TRUE / FALSE	Das Ergebnis ist FALSE, wenn es gleich 0 ist.
Konvertierungen zwischen ganzzahligen Zahlentypen	ANY BOOL; ANY INT; ANY REAL; TIME; STRING		Von größeren auf kleineren Datentyp gehen Daten verloren
REAL_TO- / LREAL_TO	ANY BOOL; ANY INT; STRING		
DATE_TO / DT_TO	ANY BOOL; ANY INT; ANY REAL; TIME; STRING		Von größeren auf kleineren Datentyp gehen Daten verloren
STRING_TO	ANY		Wenn der Typ die selbe Information wie der String hat, klappt es, ansonsten ist der Wert 0
TRUNC	REAL	INT	Von REAL zu INT. Es können Daten verloren gehen.

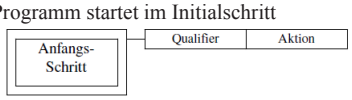
Operatoren	Gültig für	Ergebnis	Beschreibung
Numerische Operatoren			
ABS	ANY INT; ANY REAL;		Absolutwert (Betrag)
SQRT	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Wurzel
LN	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	natürlicher Logatithmus
LOG	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Logatithmus zur Basis 10
EXP	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Exponentialfunktion
SIN	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Sinus
COS	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Cosinus
TAN	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Tangens
ASIN	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Arcussinus
ACOS	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Arcuscosinus
ATAN	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Arcustangens
EXPT	ANY BOOL; ANY INT; ANY REAL	REAL; LREAL	Potenzierung

2.6.4 Ablaufsprache AS (Sequential Function Chart SFC)

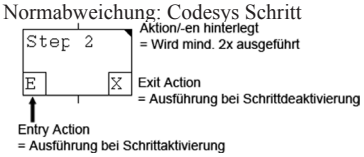
- Geeignet für Abläufe in denen schrittweise, verschiedene Zustände durchlaufen werden



- Elemente:
  - **Schritt (= Zustand)**
    - Immer nur ein Schritt aktiv
    - Ablauf je Zyklus:
      Aktionen durchführen => Transistionen prüfen
    - Aktionen = Boolsche Ausdrücke oder Programme
    - Programm startet im Initialschritt



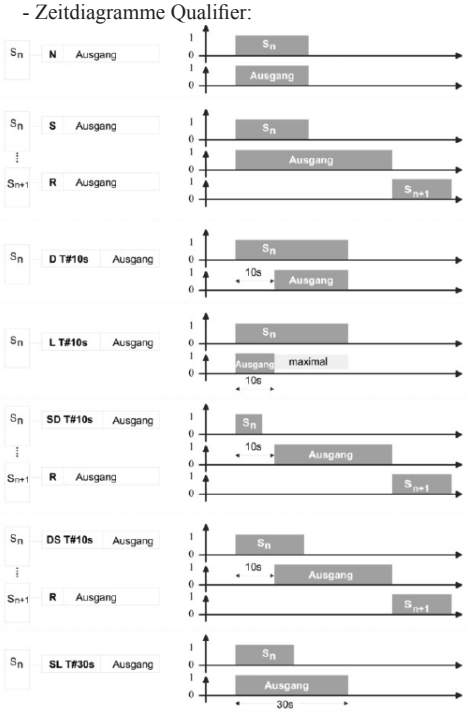
- Normabweichung: Codesys Schritt



- Qualifier = Modifikation der Aktionsausführung

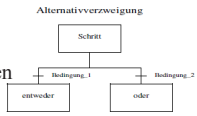
Qualifier	Kurzbeschreibung	Erklärung
leer oder **N**	Nicht gespeichert	Aktion solange wie Schrittmerker aktiv
R	Rücksetzen	Aktion deaktiviert (Bool: Wert auf FALSE)
S	Setzen	Aktion aktiv (Bool: TRUE) bis Reset
L*	Zeitbegrenzt	Aktiv (TRUE) bis Ende der angegebenen Zeit oder maximal bis Schritt Deaktivierung
D*	Zeitverzögert	Aktiv (TRUE) nach Ablauf der angegebenen Zeit bis zur Schritt Deaktivierung
P	Impuls	Aktion wird einmal ausgeführt (Bool: True für einen Zyklus – *Beachte Hinweis !*)
SD*	Gespeichert (unbedingt), Zeitverzögert	Aktiv (TRUE) nach Ablauf der angegebenen Zeit, auch wenn der Schritt nicht mehr aktiv ist. Bleibt Aktiv (TRUE) bis zum Rücksetzen.
DS*	Zeitverzögert, Gespeichert	Aktiv (TRUE) nach der angegebenen Zeit, nur wenn der Schritt noch aktiv ist. Bleibt Aktiv (TRUE) bis zum Rücksetzen.
SL*	Gespeichert, Zeitbegrenzt	Aktiv (TRUE) bis Ende der angegebenen Zeit.

- Hinweise Codesys: Wird eine Aktion deaktiviert, so wird sie noch einmal ausgeführt. Das heißt, daß jede Aktion mindestens zweimal ausgeführt wird (auch eine Aktion mit Qualifier P).
- Zeitdiagramme Qualifier:



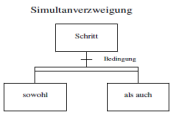
zu 2.6.4 Ablaufsprache AS

- **Transition (= Weiterschaltbedingung)**
  - Boolscher Ausdruck oder Transitionsprogramm
  - Verzweigungsarten:
    - Alternative Verzweigung
      -Nur ein Folgeschritt aktiviert
      -Verriegelung der Bedingungen

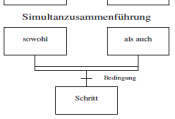


    - Alternative Zusammenführung
      - Folgeschritt wird aktiv wenn irgendein Vorgängerschritt + dessen Transition erfüllt

    - Simultanverzweigung
      - Gleichzeitige Aktivierung aber unabhängige Bearbeitung



    - Simultanzusammenführung
      - Folgeschritt erst wenn alle Vorgängerschritte aktiv und deren Trans. erfüllt



Sprünge in der Ablaufsprache

- Sprungmarke mit Ziel-Schrittname nach Transisiton
- Programmverlauf wird dort im nächsten Durchlauf fortgesetzt

Textuelle Beschreibung der Ablaufsprache

- Schritt:
  INITIAL_STEP oder STEP ... END_STEP
- Transition
  TRANSITION FROM ... TO ... ... END_TRANSITION

```
INITIAL_STEP <Name_Step_Initial>:
      Init_Aktion (N);
END_STEP

TRANSITION FROM <Name_Step_Initial> TO <Name_Step_1> :=
      Transitionsprogramm;
END_TRANSITION

STEP <Name_Step_1>:
      Step_1_Aktion (L);
END_STEP
```

- Verzweigungen (... , ...)

```
TRANSITION FROM <Name_Step_1> TO
(<Name_Step_2a>, <Name_Step_2b>) := Transitionsprogramm;
END_TRANSITION                    (*textuelle Simultanverzweigung*)
```

2.6.5 Mehrdeutigkeiten und herstellerspezifische Interpretationen des SFC

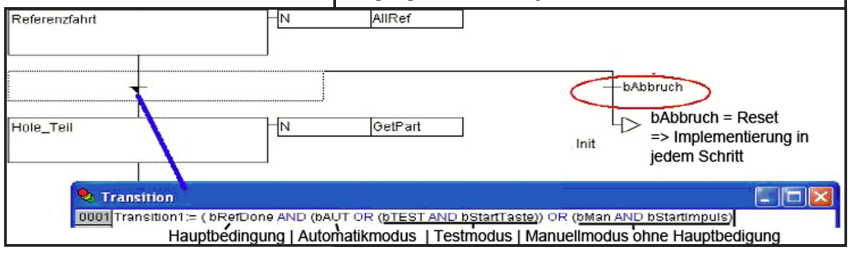
- Programmfortschritt pro Programmzyklus
  - Pro Durchlauf maximal ein Schrittweiterschaltung
  - Ausnahme S7 Graph: Einstellbar, dass über alle gültigen Transitionen weitergeschaltet werden soll

- Reihenfolge der Abarbeitung
  - Nicht festgelegt ob erst Aktionen oder Transitionen bearbeitet werden
  - Parallelzweige = Erst alle Aktionen, dann alle Transitionen (nicht zweigabhängig)
  - Alternativzweige = Meistens von Links nach Rechts (Ausnahme: S7)
  - Hierarchischer Aufbau nicht sinnvoll

2.7.2 Ablaufsteuerung mit Betriebsarten

- Schrittnummer, Name
  - Array mit Namen
- Zeitüberwachung für Schritte (min,max)
- Betriebsarten
  - Einricht-/Handbetrieb
  - Einzelschrittbetrieb
    - mit Bedingung (Testbetrieb)
    - ohne Bedingung (Manueller Betrieb)
  - Automatik
    - freigegeben
    - gestoppt (z.B. bei Fehlern)

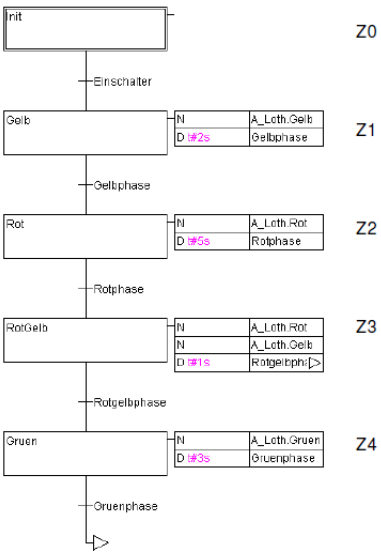
- Zusätzlich: Abbruch der Schrittkette vorsehen (zB bei Fehlern)
  - Parallelüberwachung mit zusätzlicher Abruchbedingung bAbbruch zu jeder Transistion



2.7 Programmbeispiele AS

Aufgabe: Ampelsteuerung (Rot, Gelb, Grün)

In Ablaufsprache:

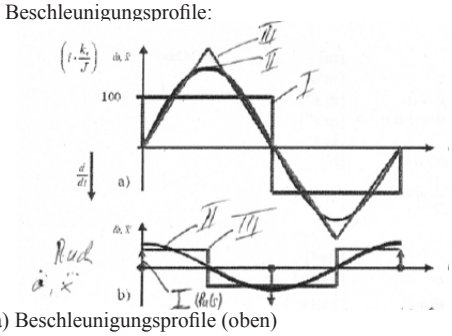


Realisierung in ST

```
PROGRAM ST_PROG
VAR
    Zustand:INT:=0;
    Gelbphase: TON;
    Rotphase: TON;
    Rotgelbphase: TON;
    Gruenphase: TON;
END_VAR

CASE Zustand OF
 0:     A_Loth.Rot:=FALSE;
        A_Loth.Gelb:=FALSE;
        A_Loth.Gruen:=FALSE;
        IF Einschalter THEN
                Zustand:=1;
        END_IF

 1:     A_Loth.Gelb:=TRUE;
        Gelbphase(IN:=TRUE,PT:=t#2s);
        IF Gelbphase.Q THEN
                A_Loth.Gelb:=FALSE;
                Gelbphase(IN:=FALSE,PT:=t#2s);
                Zustand:=2;
        END_IF

 2:     A_Loth.Rot:=TRUE;
        Rotphase(IN:=TRUE,PT:=t#5s);
        IF Rotphase.Q THEN
                A_Loth.Rot:=FALSE;
                Rotphase(IN:=FALSE,PT:=t#5s);
                Zustand:=3;
        END_IF

 3:     A_Loth.Rot:=TRUE;
        A_Loth.Gelb:=TRUE;
        Rotgelbphase(IN:=TRUE,PT:=t#1s);
        IF Rotgelbphase.Q THEN
                A_Loth.Rot:=FALSE;
                A_Loth.Gelb:=FALSE;
                Rotgelbphase(IN:=FALSE, PT:=t#1s);
                Zustand:=4;
        END_IF

 4:     A_Loth.Gruen:=TRUE;
        Gruenphase(IN:=TRUE,PT:=t#3s);
        IF Gruenphase.Q THEN
                A_Loth.Gruen:=FALSE;
                Gruenphase(IN:=FALSE,PT:=t#3s);
                Zustand:=0;
        END_IF

END_CASE
```

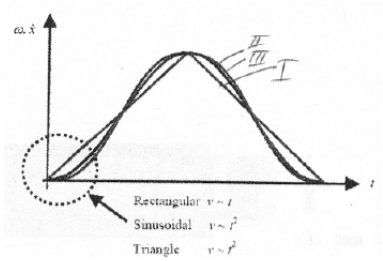
2.8 Bewegungssteuerung mit SPS

- Beschleunigungsprofile:



a) Beschleunigungsprofile (oben)
I) rechteckförmig i_{max} = 100%
II) sinusförmig i_{max} = 157%
III) dreieckförmig i_{max} = 200%
Strom == Beschleunigung
b) Ruckverlauf (unten)

- Geschwindigkeitsprofile:



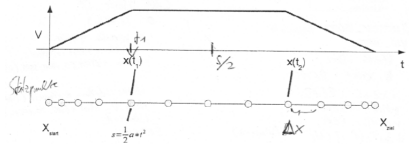
I) rechteckförmig v_{max} = 100%
II) sinusförmig v_{max} = 100%
III) dreieckförmig v_{max} = 100%

zu 2.8

- Bewegungsgleichungen:

Linear:			
Lage / Weg:	x	[m]	$s=1/2a*t^2$
Geschwindigkeit:	$\dot{x} = v$	[m/s]	
Beschleunigung:	$\ddot{x} = \dot{v} = a$	[m s^{-2}]	
Ruck:	$\dddot{x} = \ddot{v} = \dot{a} = r$	[m s^{-3}]	
Bewegte Masse:	m	[kg]	
Beschleunigende Kraft:	$F = m \cdot a$	[N]	
Energie (mech.):	$E = \frac{1}{2} m \cdot v^2$	[W s]	
Rotatorisch:			
Lage / Winkel:	φ	[Rad]	
Winkelgeschwindigkeit:	$\omega = \dot{\varphi}$	[Rad/s]	
Winkelbeschleunigung:	$\alpha = \dot{\omega} = \ddot{\varphi}$	[Rad s^{-2}]	
Ruck:	$\dot{\alpha} = \dot{\omega} = \dddot{\varphi}$	[Rad s^{-3}]	
Trägheitsmoment	J	[kg m^2 bzw. kg cm^2]	
Beschl. Drehmoment	$M = J \cdot \alpha$	[Nm]	
Energie (mech.):	$E = \frac{1}{2} J \cdot \omega^2$	[W s]	

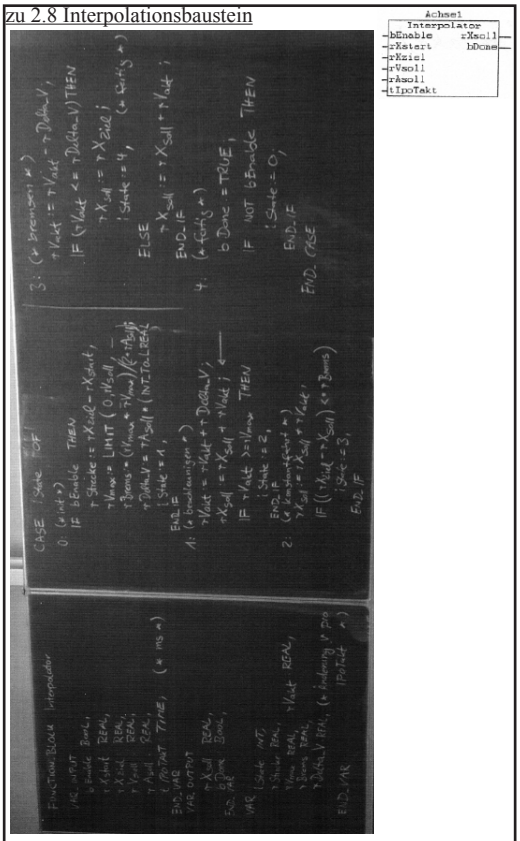
- Interpolation mit festem Takt



Strecke: $s = X_{ziel} - X_{start}$
Maximal mögliche Geschwindigkeit: $v = \sqrt{(a*s)}$
Beschleunigungszeit: $t_1 = v / a$
Bremsweg: $s_{brems} = v^2 / 2a$

Geschwindigkeit: $v = \frac{x(t_{n+1}) - x(t_n)}{IpoTakt}$

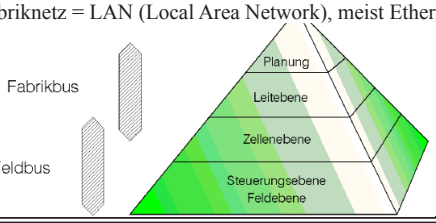
zu 2.8 Interpolationsbaustein



3. Feldbusse

3.1 Überblick Lokale Netze

- Fabriknetz = LAN (Local Area Network), meist Ethernet



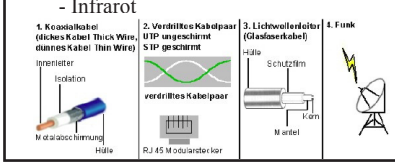
3.2 Einführung Feldbusse

Merkmal	Konventionell	Feldbus
Anschluss	Jeder Aktor / Sensor einzelnd mit 2 Ader Kabel	Alle Geräte an Feldbuskabel (2, 4 oder 5 Adrig)
Signal	Analog, 4-20mA oder 0-10V	Siehe 3.3.2
Signalfluss	Eine Richtung	Bidirektional
SPS Anbindung	1 I/O Anschluss je Aktor/Sensor	1 Feldbuskarte
Sonstiges		- Flexibler - idR günstiger - Inbetriebnahme & Wartung einfacher

Anforderungen an Feldbusse:
- Ausdehnung 100m bis wenige km
- kurze Verzögerungszeiten (Echtzeit ?)
- Datenraten von 0,1 bis 100 Mbit/s
- Geringe Fehlerraten (10^{-9})
- Kostengünstige Netzanschlußkomponenten
- Einfache Anpassung an die Topologie der Anlage

3.3 Technische Grundlagen

Medien für Feldbusse:
- Serielle Busse
  - Mediengebunden
    - Drahtleitung (Wechselstromnetz)
    - Verdrillte Zweidrahtleitung
    - Koaxkabel
    - Lichtwellenleiter
  - Medienungebunden
    - Funk
    - Infrarot

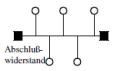


3.3.1 Netzwerktopologien

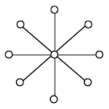
= physikalische Anordnung des Bussystems

- Ring / Doppelring
  - Punkt-zu-Punkt Verbindung
  - Daten laufen in einer Richtung
  - Für beide Richtungen: Doppelring
  - Vorteildoppelring: Redundanz bei Kabelbruch

- Linie / Busstruktur
  - Multipoint Verbindung
  - Passiver Anschluss ans Kabel
  - Leicht erweiterbar
  - Bei Unterbrechung: Restteile bleiben verfügbar
  - Abschlusswiderstände gegen Reflexionen



- Stern
  - Punkt-zu-Punkt Verbindung über zentralen Knotenpunkt (Hub)
  - Hubausfall = ganzer Bus fällt aus
  - Sternkoppler zu weiteren Hubs
  - Repeater = Aktiver Hub (Signal wird verstärkt)
  - Passive Hub = Nur Weiterleitung
  - Switch = Durchschaltung zum jeweiligen Empfänger, Duplexleitung (Senden und Empfang)

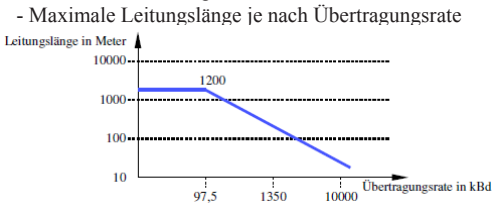


3.3.2 Übertragungsstandard RS485

- Erdsymmetrische Schnittstelle für Mehrpunktverbindungen (Linie, Bus)
- Für Punkt-zu-Punkt: RS 422
- Signalübertragung durch Differenzspannungsverfahren
  - Messung der Spannungsdifferenz U_{ab} zwischen zwei Leitungen
  - Unterschiedliche Schwellwerte

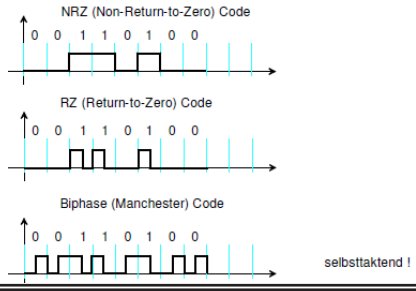
	Sender	Empfänger
„0“	1,5 V < U_{ab} < 5 V	U_{ab} > 0,2 V
„1“	-5 V < U_{ab} < -1,5 V	U_{ab} < -0,2 V

  - Spannungsfestigkeit gegen Masse -7V und 12V
    - Je nach Treiberbaustein bis zu 25V
  - Hohe Störsicherheit bei Auswertung durch verdrillte Leitung (gute Gleichtakt-Unterdrückung) und Differenzspannungsmessung zwischen Zwei Leitern
- Laut Norm: max 32 parallele Teilnehmer
- Maximale Leitungslänge je nach Übertragungsrate

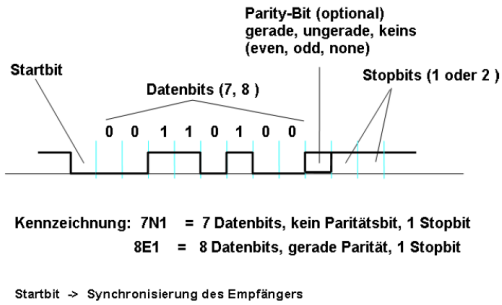


3.3.3 Übertragungscodes

- Beispiele:



3.3.4 Zeichenkodierung



3.3.5 Adressierung

- Knotenadresse (Subadresse: Port)
  - Zieladressierung
  - Nachrichtenadressierung
- Sonderformen: Broadcast bzw. Multicast

3.3.6 Buszugriffsverfahren / Arbitrierung

- Master / Slave
  - Master entscheidet welcher Slave sendet
  - Zyklische Erfassung aller Sendeanfragen (Polling)
  - Kommunikation exklusiv nur über Master
- Token Passing
  - Spezielle Nachricht (Token) erlaubt das Senden
  - Weitergabe des Tokens zum Nachbarknoten
  - Probleme: Blockierung ,Tokenverlust, Knotenausfall
- Carrier Sense Multiple Access
  - CSMA / CD (Collision Detection)
    - Zufälliges Verfahren, Senden bei freier Leitung
    - Mithören bei Sendung zur Kollisionsdetektion, falls ja Störsignal (jam) danach neuer Sendeversuch nach zufälliger Wartezeit
    - Rahmenformat: IEEE 802.3
    - Reaktionszeit steigt stark mit steigender Busbelastung
    - Vorteil: Hohe Effizienz, geringer Overhead

zu 3.3.6 Buszugriffsverfahren / Arbitrierung

- CSMA / CA (Collision Avoidance)
  - Zusätzlich zum Abhören: Teilnehmerpriorität
  - Zusätzlicher Identifier im Signal und ein elektrisch dominanter Zustand auf Bus
  - Signallaufzeit t_s << Bitzeit t_b
  - Maximale Übertragungsrate von Ausdehnung des Netzwerks abhängig
  - Echtzeitfähig für hochpriore Teilnehmer
- Time Distributed Multiple Access (TDMA)
  - Zeitgesteuertes Verfahren: jeder Knoten kann nacheinander für ein bestimmtes Zeitintervall auf Bus senden
  - Busmaster als Taktgeber nötig
  - Nachteil: Schlechte Auslastung, da Leerintervalle nicht genutzt
  - Vorteil: Vorhersagbarkeit der Datenzyklen

3.4 Beispiele von Bussystemen

3.4.1 ASI

Offener Bus ?	ja
Nutzerorganisation	Verein zur Förderung busfähiger Interfaces für binäre Aktuatoren und Sensoren e.V. (über 50 Mitgliedsfirmen) www.as-interface.com
Normen	EN 50295
Topologie	offene Baumstruktur
Teilnehmer	max. 31 Slaves *Version V2.1: max. 62 Slaves*
Buszugriffsverfahren	Master/Slave mit zyklischem Polling
Übertragungsrate und Leitungslängen	167 kBd, max 100 m
Telegrammformat	fest, 4 Bit Daten
Datensicherung	Paritätsbit, Mehrfachabastung in physikalischer Schicht
Buspegel	4 V_{SS} Datenpegel und 24 V/ 2A Hilfsenergie gemeinsam auf einer Zweidrahtleitung
Leitung	Zweidrahtleitung, nicht verdrillt, ungeschirmt
Sonstiges	Zykluszeit deterministisch ≤ 5 ms

- Zykluszeit ASI

  T = (s + 2)× 25× t

  s = Anzahl der Slaves und t = Bitzeit
- Zweidrahtleitung mit automatischer Teilnehmerkontaktierung mittels Schneidklemmtechnik
- Energieversorgung über Feldbusleitung
- Signalerzeugung mittels Spannungsspitzen

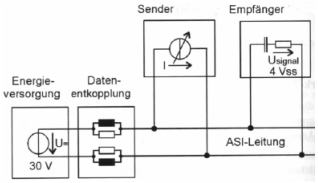
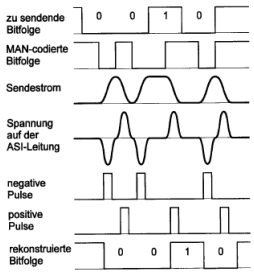


Abbildung 3-27 Signalerzeugung und Energieversorgung

- Modulationsverfahren mit Manchestercode als Stromsignal im Sender => Umwandlung durch Spule in überlagerte Signalspannung auf dem ASI Kabel (Alternierende Puls Modulation APM)
- Spannungspulse ca. $\sin^2$ Form = wenig Oberwellen



- Master / Slave Verfahren
  - Kurze Reaktionszeiten (Zykluszeit Anfrage/Antwort = 25 Bitzeiten / 150µs)
  - Antwortzeit Slave 3 Bitzeiten
- Datensicherung
  - 16 Bitabtastungen => nur korrekte Signalform ok
  - Gleichstromfreie Modulation (Auf Pos Puls folgt immer negativer Puls)
  - Länge der Telegramme bekannt, kurze Leitungspausen zwischen den sendungen
- Telegrammstruktur

Masteraufruf:

ST	SB	A4	A3	A2	A1	A0	I4	I3	I2	I1	I0	PB	EB

ST=Startbit SB=Steuerbit A4...A0=Slaveadresse
I4...I0= Informationen PB= Paritybit EB=Endebit
Informationen abhängig nach Aufruftyp in Steuerbit.

Slaveaufruf:

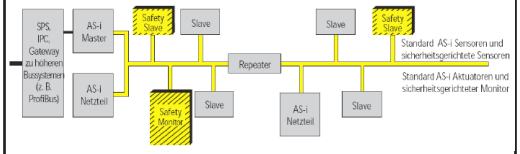
ST	I3	I2	I1	I0	PB	EB

ST=Startbit I3...I0= Informationen PB= ParitybitEB=Endebit

zu 3.4

ASI Version 2.1
- Erweiterter Adressmodus => 2 Gruppen (Slaves A & Slaves B) mit je 31 Slaves
- Programmierung der Slaves im eingebauten Zustand
- Unabhängiger Watchdog der Reset auslöst und Ports deaktiviert

ASI-Safety Anwendung
- Zusätzlicher Safety Monitor im Netzwerk
  - Erkennung Slave Ausfall
  - Erkennung Kommunikationsstörung
  - Erkennung Master Ausfall



- Sicherheitsslaves
  - Wiederholung einer festen Nachrichtensequenz zur Fehlerprüfung

3.4.X Profibus

Offener Bus ?	ja	
Nutzerorganisation	Profibus-Nutzerorganisation www.profibus.com	
Normen	DIN 19245, Teil 1-3	
	EN50170 Vol 2	
	Profibus PA	DIN19245, Teil 4
		IEC 1158-2
Topologie	Linie	
Teilnehmer	32 ohne, 127 mit Repeater	
Buszugriffsverfahren	Master/Slave	
	Token Passing	
Übertragungsrate und Leitungslängen	12 MBd, 100 m	
	9,6 kBd, 1200 m	
	LWL bis einige km	
Telegrammformat	0...246 Byte	
Datensicherung	Längs- und Querparität, HD=4	
Buspegel	RS 485	
	(LWL)	
Leitung	Zweidrahtleitung, verdrillt, geschirmt	
	Lichtwellenleiter	

- Zykluszeit Profibus DP

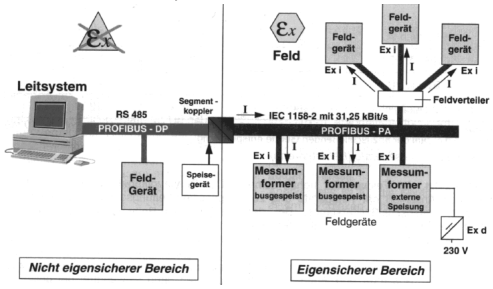
  T = (467 + s × (158 + d ×11))× t

  s =Anzahl Slaves, d = Anzahl Bytes/Slave,

  t = Bitzeit (=1/Übertragungsrate)
- Profibus Varianten:
  - Profibus DP (Dezentrale Peripherie)

    = Für Anschluss dezentraler Peripherien
  - Profibus PA (Process Automation)

    = Für Explosionsgefährdete Bereiche
    - Signalübertragung im Ex: 31,25kBit/s
    - Außerhalb 93,74 kBit/s



  - Profibus FMS (Fieldbus Message Specification)

    = für Kommunikation zwischen intelligenten Geräten
    - hybrides Zugriffsverfahren: Master / Slave, wobei mehrere Master mit Token Passing arbeiten
    - Datenübertragung mit UARTZeichen
    - Telegrammformat mit Prüfsumme (Frame Check Sequence FCS)

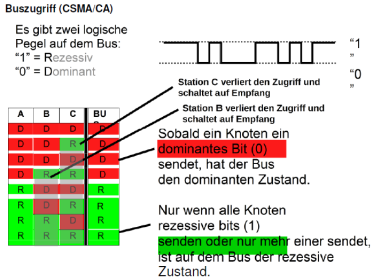
Startzeichen	Zieladresse	Quelladresse	Kontrollbyte	Daten	FCS	Endzeichen

    - Definition von Virtual Field Devices (VFD) = Teil des Geräts der für Kommunikationssystem sichtbar ist

3.4.2 CAN

Offener Bus ?	ja	
Nutzerorganisation	CiA e.V. (CAN in Automation) www.can-cia.com	
Normen	ISO/DIS 11519-1 (Schicht 2) ISO/DIS 11898, 9141 (Schicht 1)	
Topologie	Linie mit kurzen Stichleitungen	
Teilnehmer	ohne Repeater	32
	mit Repeater	unbegrenzt
Buszugriffsverfahren	CSMA/CA mit Prioritäten	
Übertragungsrate und Leitungslängen	1 MBd bis 40 m	
	500 kBd bis 130 m	
	250 kBd bis 270 m	
	125 kBd bis 530 m	
	100 kBd bis 620 m	
	50 kBd bis 1300 m	
	20 kBd bis 3300 m	
	10 kBd bis 6700 m	
	5 kBd bis 10000 m	
Telegrammformat	0...8 Byte	
Datensicherung	16 Bit CRC, HD=6	
Buspegel	RS485 modifiziert (ISO 11898)	
Leitung	Zweidrahtleitung, verdrillt, geschirmt	
Sonstiges	verschiedene Schicht 7-Protokolle definiert	
	CANOpen (auf Basis CAL)	
	SDS (Honeywell)	
	DeviceNet (Allen Bradley)	

- Unterschiedliche Treiberbausteine je nach Anforderung (Erhöhte Störsicherheit, +- 80 V Offset Stabilität)
- Unterschiedliche Client Prioritäten möglich
- Quellenbasierte Adressierung (ID des Telegramms gibt Inhalt der Nachricht an, nicht Zieladresse)
- Buszugriff per CSMA/CD



- Anwendungsschicht CANopen
  - Plug and Play Fähigkeit durch genomierte IDs, Geräte Profile, Datentypen, Stecker und Stromversorgung
  - Netzwerk Management
  - Konfigurations- und Analyse-Tools
  - Syncronisierte Datenübertragung
  - Objekt Verzeichnis mit 16bit index und 8bit subindex (Ähnlich Nachschlage Tabelle)
    - Bestimmte Bereiche für verschiedene Zwecke (Datentypen, Kommunikationsprofile, Geräteprofile)
    - Zugriff per Konfigwerkzeug

Minimal erlaubtes Objektverzeichnis

Index	SubIdx	Type	Description
1000h	0	UNSIGNED32	Device Type Information
1001h	0	UNSIGNED8	Error Register
1018h			Identity Object
	0	UNSIGNED8	= 4 (Number of sub-index entries)
	1	UNSIGNED32	Vendor ID
	2	UNSIGNED32	Product Code
	3	UNSIGNED32	Revision Number
	4	UNSIGNED32	Serial Number

Abbildung der Objekte auf CAN Nachrichten

- Nachrichten IDs die für ProzessdatenObjekte (PDO) und ServiceDatenObjekte (SDO) erhält man aus der Knotennummer plus einer Startadresse
- Die Knotennummer findet sich also in den letzten 7 Bits der CAN Nachricht -> max. 127 Knoten in einem CANopen System

CAN ID From	To	Communication Objects
0h		NMT Service
80h		SYNC Message
81h	FFh	Emergency Messages
100h		Time Stamp Message
181h	1FFh	1st Transmit PDO
201h	27Fh	1st Receive PDO
281h	2FFh	2nd Transmit PDO
301h	37Fh	2nd Receive PDO
381h	3FFh	3rd Transmit PDO
401h	47Fh	3rd Receive PDO
481h	4FFh	4th Transmit PDO
501h	57Fh	4th Receive PDO
581h	5FFh	Transmit SDO
601h	67Fh	Receive SDO
701h	77Fh	NMT Error Control

Prof. Dr. Sommer

- SDO Transfer (azyklisch, niederprior)
  - Direkter Transfer (Daten ≤ 4 Byte)
    - 2 CAN Nachrichten notwendig (Anfrage vom Client (Master), Antwort vom Server)
  - Segementierte Übertragung (4 Byte < Daten < 127*7 Byte)
    - 2 CAN Nachrichten zur Initialisierung
    - Jedes weitere Nachrichtenpaar (Anfrage <=> Antwort) übertragt weitere 7 Byte Daten

zu 3.4.3 Ethernet

Offener Bus ?	ja	
Nutzerorganisation	-	
Normen	Basis IEEE 802.3 IEEE 802.3u (Fast Ethernet, 1995)	
Topologie	802.3	Bus Stern Baum
	802.3u	Stern Baum
Teilnehmer	100 je Segment ohne Repeater	
Buszugriffsverfahren	CSMA/CD	
Übertragungsrate und Leitungslängen	10 Mbd max. 4520m	
	100 MBd max. 412m	
	(1000 MBd)	
Telegrammformat	72...1526 Byte (Nutzdaten 0 ... 1500 Byte) mit Manchesterkodierung (10M),bzw 4B5B Kodierung (100M)	
Datensicherung	4 Byte CRC, HD=6 $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$	
Buspegel	10MBd	high 0V, low -2,05V idle -2,05V
	100 MBd	andere Festlegungen
Leitung	10base5	Triaxkabel (Doppelter Schirm)
	10base2	Koaxkabel
	10baseT 100baseT4 100baseTX	Zweidrahtleitung verdrillt, geschirmt
	10baseF 100baseFX	Lichtwellenleiter

- Kabelbezeichnung:
  [Datenrate][Übertragungsband][Länge/100m]
  - 10base2 = 10 Mbd im Basisband und 200m pro Segment
- Vorteile von Ethernet: Günstig, Standard, hoher Verbreitungsgrad
- CSMA / CA
  - Mindespaketlänge für CA: 5,1µs bzw. 64 Byte
  - Kein garantiertes Zeitverhalten
  - Switch kann Kollisionsbereiche entkoppeln und Full Duplex möglich (2 Adernpaare nötig, Zeitverzögerung > 5µs)
- Echtzeitfähigkeit bie Ethernet mittels:
  - Master / Slave (=Polling) => Ethernet Powerlink
  - Zeitgesteuert (TDMA) => PROFINET
  - Datentrigger => EtherCAT
- Telegrammformat

7	1	6	6	2	Daten 0..1500	0..46	4
PRE	SFD	DA	SA	LEN	Data	Pad	FCS

PRE = Preamble (Vorspann) mit dem Bitmuster 10101010.....(-> Synchronisation)
SFD = Starting Frame Delimiter (Startzeichen = 10101011)
DA = Destination Address
SA = Source Address
LEN = Length, Data = Information
Pad = Padding (Füllzeichen)
FCS = Frame Check Sequence
- 0 bis 1500Byte Nutzdaten mit 26 Protokollbytes (Overhead), Kurze Pakete werden aufgefüllt (mindest Telegrammlänge = 72Bytes)

- Ethernet Adressen
  - Jede Station hat eindeutige Adresse mit 6 Byte Codierung (Ethernet / Physikalische / Stations / MAC Adresse)

D47	D46	D45 - D24	D23 - D0
Typ1	Typ1	Herstellerkennung	Adapter Seriennummer

D47 = 0 -> individuelle Adresse (für Source-Adresse immer 0)
= 1 -> Gruppenadresse
D46 = 0 -> Adresse global verwaltet (IEEE)
= 1 -> Adresse lokal verwaltet, nicht koordiniert
0xFFFFFFFFFFFF = Broadcast Adresse

- Fast Ethernet
  - Hochwertigere Kabel, geringere Entfernungen, 4B5B Codierung und Einschränkung bei Reapeteranzahl erlauben 100MBd
  - Protokollergänzungen für Autosensing
  = Autom. Aushandeln der Übertragungsrate und -art
- Switches in der Automatisierung
  - Effekte: Volle Datenrate in jedem Segment durch Lasttrennung, Mehre Datenpakete gleichzeitig bearbeitbar durch Switch, Keine Kollisionen bei Vollduplex
  - Spannung Tree Algorithmus verhindert das Nachrichten in einem vermaschten Netz kreisen
  - Adressierung über MAC Adresse

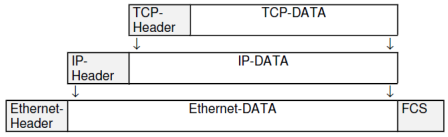
zu 3.4.3 Kommunikationsprotokoll TCP/IP

- TCP = Transmission Control Protocol
- IP = Internet Protokoll

Application Programs	Application-protocols - Telnet - NFS - FTP, etc	Application	7: Application 6: Presentation 5: Session
TCP	UDP	Transport	4: Transport
IP / ICMP / ARP / RARP		Internet	3: Network
Packet-Driver		Network-Interface	2: Data Link
Ethernet-Adapter		Hardware	1: Physical Link

ARP = Address resolution protocol
ICMP = Internet control message protocol
RARP = Reverse address resolution protocol

- Telegramm Aufbau für TCP/IP



- Internet Paket IP

	32 Bit			
Byte 0	Version	Length	Serv. Type	Length (total)
Byte 4	Identification		Flags	Fragment Offset
Byte 8	Life-Time	Protocol	Header-Checksum	
Byte 12	Internet Source Address			
Byte 16	Internet Destination Address			
Byte 20	Options (Padding auf 4 Byte Grenze)			
	DATA			

- 20 bis 60 Protokoll (je nach Optionen)
- 0 bis 65516 Datenbytes
- Erläuterungen zum IP-Datagramm

Feld	Anzahl Bit	Bedeutung
Version	4	IP-Version
Header Länge	4	Header-Länge in Bytes (20..60)
Service Typ	8	nicht verwendet
Länge	16	Länge Header + Datenfeld (Bytes)
Identifikation	16	Angaben für Segmentierung
Flags	4	Angaben für Segmentierung
Fragment Offset	12	Angaben für Segmentierung
Lebensdauer	8	max. Verweilzeit im Netz
Protokoll	8	Identifikation des Protokolls
Prüfsumme Header	16	einfache Prüfsumme für Header
Internet Source Adresse	32	Adresse des Senders
Internet Destination Adresse	32	Adresse des Empfängers
Optionen	0..320	Angaben über Optionen
Padding	0...31	Ergänzung der Optionen auf Vielfaches von 4 Byte
Daten		Daten der TCP-Schicht

- Lebensdauer: Jeder Rechner den das Paket durchläuft reduziert Wert um eins => max. 255 Zwischenstationen bis Lösung des Pakets (= Wert 0)
- Protokoll: Kennung für Übergeordnetes Protokoll (TCP =0x0006)
- Prüfsumme Header: für Kopfdaten, eigentliche Nutzdaten werden durch Ethernet Checksumme geprüft
- Internet Adressen (IP V4)
  - 32 Bit, aufgeteilt in Netzwerk- und Teilnehmer Teil
  - Klassen: A, B, C

Class A	0	Netzwerk	Rechner (Host)
Class B	10	Netzwerk	Rechner (Host)
Class C	110	Netzwerk	Rechner

  - Logische Adresse = IP Adresse
  - ARP (Adress Resolution Protokoll): Zusammenhang physische (Ethernet) <=> logisch (IP)
  - Internet Controll Message Protokoll (ICMP) = Austausch von Informationen zwischen Netzwerkknoten und Routern

- Transmission Control Protocol TCP
  - Aufgabe: Zuverlässige Datenverbindung zwischen 2 Stationen realisieren
  - Gezielter Verbindungs Auf- und Abbau nötig
  - TCP Paket

	32 Bit		
Byte 0	Source Port		Destination Port
Byte 4	Sequence Number		
Byte 8	Acknowledgement Number		
Byte 12	Offset	Flags	Window
Byte 16	Checksum		Urgent Pointer
Byte 20	Options (Padding auf 4 Byte Grenze)		
	DATA		

  - Zerlegung in kleinere Pakete (Segemente mit Sequence Number)

3.4.4 PROFINET

= Ethernet mit Echtzeitfähigkeit

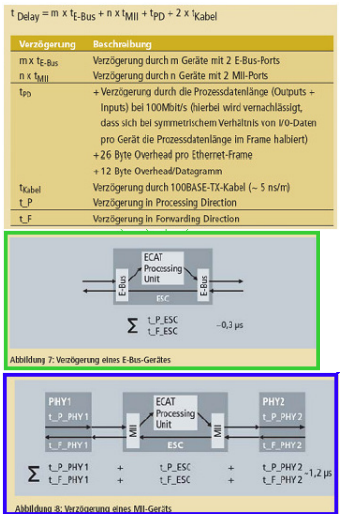
- Vergleich PROFINET <=> PROFIBUS

Funktionalität	PROFINET IO	PROFIBUS
Adressierungsmöglichkeit	Slot, Subslot, Index	Slot, Index
Datenaustausch	IO-Device wird einmal parametriert und arbeitet dann autark (Provider/Consumer-Modell)	Nur nach Aufforderung
Datenkanäle	Es können mehrere Datenkanäle zwischen Controller/Supervisor und Device aufgebaut werden.	Nur ein genau definierter Datenkanal zwischen Master und Slave
Daten-Priorisierung	Möglich, durch flexible Einstellung der Aktualisierungsrate.	Gleich prioritärer Datenverkehr
Anzahl der Teilnehmer	Nur durch die Netzwerk-ID bestimmt	Maximal 126 Teilnehmer
IT-Services	Können uneingeschränkt integriert werden	Nicht möglich
Gerätebeschreibung	XML-basiert mit Schema-Definition	Schlüsselwort-basiert
Zugriff auf Daten eines Feldgerätes	Von mehreren Teilnehmern lesend und schreibend möglich	Von mehreren Teilnehmern nur lesend möglich
Alarme und Diagnosen	Können unterschiedlich priorisiert werden	Nur eine Priorität möglich
Gerätemodellierung	Mehrere Feldgeräte einer Gerätefamilie können in einer GSD-Datei mehrsprachig beschrieben werden	Ein Feldgeräte einer Gerätefamilie kann in einer GSD-Datei beschrieben werden
Adresseinstellung	Automatische Adressvergabe im Konzept enthalten	Über DIP-Switch oder per Telegramm
Übertragungsrate	100 Mbit/s vollduplex	Max. 12 Mbit/s

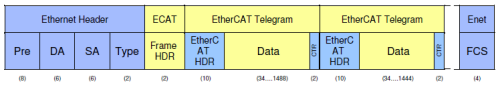
- Konformitätsklassen
  A - 100 ms -Component based Automation(SPS, Steuerungsrechner)
  B - 10ms - Soft Real Time (Fertigungskontrolle)
  C - 1ms - Isochronous Real Time (Bewegungssteuerung)
- Zwei Stufen von Echtzeit
  - Software (B)
  - Hardware (C) : Quasiparalleler Datenverkehr

3.4.5 EtherCat

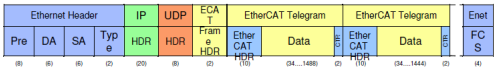
- Echtzeitfähig, Full Duplex
- Doppelring Struktur mit Hin- und Rückleitung, Abzweige als Baum mittels Kopplern
- Pakete durchlaufen immer alle Stationen
  - Ethernet Frame durchläuft alle Stationen mit mehreren Sub-Telegramen mit variabler Länge und Infos
- Verzögerungszeit



- Nachrichtenformat



EtherCAT/UDP: with Routing and for Socket Interface Implementation
UDP Port: 34980 – 0x88A4, registered with IANA



- FMMU: Fieldbus Memory Management Unit
= Prozess Daten von Ethernet-I/Os können beliebigen Telegrammen und logischen Adressen zugeordnet werden

3.4.6 Echtzeit Ethernet Vergleich

- EtherCAT 276µs
- Sercos III 479µs
- PROFINET IRT 763µs
- Powerlink 2347µs
- Profinet I/O 6355µs

4. Sicherheit/Zuverlässigkeit/Normen

4.1 Abgrenzung: Sicherheit - Zuverlässigkeit

Zuverlässigkeit	keine Ausfälle = keine Fehlerzustände
Sicherheit	kein Auftreten von Gefahren für Mensch und Maschine

	Sicherheit	Zuverlässigkeit
Maßnahmen zur	Verminderung von Fehlerauswirkungen	Verringerung der Häufigkeit von Fehlern und Ausfällen
Grund	Gesetze und Vorschriften	Wirtschaftlichkeit (Image)
Nachweis	Prüfbericht Gutachten	Theoretische Berechnungen Praxiserfahrung (Lange Garantiezeit)

4.2 Sicherheit von Maschinen

- Jeder Hersteller ist verpflichtet alle konstruktiven Möglichkeiten auszuschöpfen um Unfälle zu vermeiden. Lassen sich Gefahren nicht beseitigen, so sind geeignete Schutzeinrichtungen vorzusehen.

4.2.1Aufbau des sicherheitstechnischen Regelwerkes

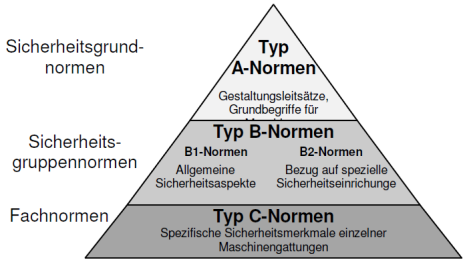
- Zutreffend können sein
  - Europäische Richtlinien (RL)
  - Europäische Normen (EN)
  - Internationale Normen (IEC)
  - VDE Vorschriftenwerk
  - zusätzlich: Unfallverhütungsvorschriften (UVV) der jeweiligen Berufsgenossenschaften, maßgeblich:
    - BGV A1 (Allgemeine Vorschriften, bisher VBG 1)
    - BGV A2 (Elektrische Anlagen und Betriebsmittel, bisher VBG 4)
    - VBG 5 (Kraftbetreibende Arbeitsmittel)

4.2.2 Europäische Richtlinien (RL)

- Sind nur Rahmengesetzte mit allgemein gehaltenen Schutzzielen, mit Umsetzung in nationales Recht
- Wichtig:
  - Niederspannungsrichtlinie 72/23/EWG:
    = Schutz vor Gefahren durch elektrischen Strom bei Niederspannungsgeräten im Spannungsbereich 50...1000 VAC, 75...1500 VDC, CE-Kennzeichnungspflicht seit 1997
  - Maschinenrichtlinie 89/392/EG und ISO13849:
    = Grundlegende Anforderungen an die Sicherheit der Maschinen zum Schutz der Gesundheit des Betreibers. inzwischen gilt diese Richtlinie auch für Sicherheitsbauteile, CE-Kennzeichnungspflicht seit 1995
    - Informationspflicht über Abweichungen von Sicherheitsanforderungen
    - Beseitigung oder Minimierung der Gefahren
    - Schutzmaßnahmen bei Risiken
    - Benutzerinfo über verbleibende Risiken
    - Hinweis auf persönliche Schutzausrüstung
    - Berührungssicherung vorhersehbarer Fehlbedingungen oder Fehler
    - Vollständige Lieferung mit Sicherheitszubehör
    - Betriebsanleitung
    - Art und Intervalle von Wartungs- und Inspektionsarbeiten
    - Kennzeichnung und EG-Konformitätserklärung
    - EG-Baumusterprüfungspflicht (nur bei bestimmten Maschinen)
  - EMV-Richtlinie 89/336/EWG (Elektromagnetische Verträglichkeit):
    = Zwei Grundlegende Anforderungen an die Geräte sind die allgemein gehaltenen Grenzen für Störaussendung und Störfestigkeit bei Einstrahlung, CE-Kennzeichnungspflicht seit 1996

4.2.3 Europäisches Normenwerk zur Sicherheit von Maschinen

- Hierarchische Struktur des Normenwerks:
  - Typ A-Normen:
    = Gestaltungsleitsätze, die für alle Maschinen gültig sind
  - Typ B-Normen:
    = Sicherheits-Gruppennormen, anwendbar auf unterschiedliche Maschinengruppen
    - B1-Normen = konstruktive und funktionale Aspekte für eine Reihe von maschinen
    - B2-Normen = Spezielle Sicherheitseinrichtungen
  - Typ C-Normen
    = Sicherheits Fachnormen, mit konkreten Anforderungen für einzelne Maschinenarten



- Normbeispiele

Normentyp	Bezeichnung	Typische Beispiele	
Typ A-Normen	Sicherheits-Grundnormen	EN292	Allgemeine Gestaltungssätze
		EN1050	Risikobewertung
Typ B1-Normen	Sicherheits-Gruppennormen	EN294	Sicherheitsabstände
		EN394	Mindestabstände
Typ B2-Normen		EN418	NOT-AUS-Einrichtungen
		EN574	Zweihandschaltungen
		EN953	Trennende Schutzeinrichtungen (TSE)
		EN954-1	Sicherheitsrelevante Teile von Steuerungen
		EN1037	Energiezufuhr /-abbau
		EN1088	Verriegelungseinrichtungen an TSE
		EN60204	Elektrische Ausrüstung von Maschinen
		EN61496	Berührungslos wirkende Schutzeinrichtungen (BWS)
Typ C-Normen	Sicherheits-Produktnormen	EN201	Spritzgießmaschinen
		EN422	Blasformmaschinen
		EN692	Mechanische Pressen
		EN693	Hydraulische Pressen
		EN775	Industrieroboter
		EN12415	Kleine NC-Drehmaschinen
		EN2417	Bearbeitungszentren

4.3 Grundsätze der Maschinensicherheit

4.3.1 Maschinenbegriff

Maschine (gemäß EG Maschinenrichtlinie)
= „...ist eine Maschine eine Gesamtheit von miteinander verbundenen Teilen oder Vorrichtungen, von denen mindestens eines beweglich ist,...“

-elektr Ausrüstung: DIN EN 60204-1 (VDE 0111 Teil 1)
- Abgrenzung Maschine <=> Anlage gemäß DIN EN 60204-1 (VDE 0113 Teil 1)
  - Hauptschalter
  - Steuertransformatoren
  - Risikobewertung
  - Definierte Leiterfarben
  - Schutz gegen automatischen Anlauf
  - Spezifische Anforderungen an „Handlungen im Notfall“

Sicherheit einer Maschine
= ergibt sich aus Gesamtheit aller Betriebsmittel an und außerhalb der Maschine

4.3.2 Sicherheitsbegriff

- Sicherheit eines Steuerungssystem bezieht sich auf Folgen für Personen und Sachen wenn einer Fehler auftritt
- Arten der Gefährdun: DIN EN 292-1, DIN EN 1050
- Fehlerhafte Steuerungen: DIN EN 954-1
- Elektrische Sicherheit = Schutz vor der Elektritizät
- Funktionale Sicherheit = Schutz vor einer nicht korrekten Funktion an der Maschine
- Sicherheitsrelevante Teile von Steuerungen:
  DIN EN 954-1, setzt korrekte Funktion von Schutz- und Steuereinrichtungen vorraus und Fehlerfälle müssen Maschine in sicheren Zustand belassen oder in diesen überführen
- Erreichte funktionale Sicherheit gemäß:
  - Safety Integrity Level in IEC 61508
  - Kategorien in der DIN EN 954-1
  - Anforderungsklassen in DIN V 19250 und DIN V VDE 0801

4.3.3 Wege zur sicheren Steuerung bei Neukonstruktion

- Fall A: Fachnorm liegt vor

Für die entsprechende Maschinenart ist die C-Norm anzuwenden.
Mit einer ausreichend konkret gefassten Norm kann überprüft werden, ob das angemessene Niveau der Sicherheit erreicht wird.
Die im Anhang I der Maschinenrichtlinie eingeforderte Gefahrenanalyse ist auch bei Anwendung von C-Normen durchzuführen, es vereinfacht sich lediglich die Maßnahmenfindung.

- Fall B: Fachnorm fehlt

Es ist eine Gefahrenanalyse für die Maschine vorzunehmen, um in der Abhängigkeit von der Technologie für die sicherheitsrelevanten Teile der Maschine eine Kategorie festzulegen.
Eine Hilfestellung dabei gibt die Typ B-Norm DIN EN 954-1.
Es gibt weitere Typ B-Normen zu beachten, z. B.:
DIN EN 418 (NOT-AUS Einrichtungen)
DIN EN 574 (Zweihandschaltungen)
DIN EN 1037 (Vermeiden von unerwarteten Anlauf)
DIN EN 60204-1 (VDE 0113 Teil 1) (Elektrische Ausrüstung von Maschinen)
Weitere Normen im Zusammenhang mit der DIN EN 60204-1 (diese Norm weist auf alternative technische Lösungen mit einer tabellarischen Übersicht im europäischen Vorwort hin)

4.4 Risikoanalyse

- Sicherere Steuerungen, insbesondere Software gemäß Norm IEC 61508

Teilnorm	Beschreibung
IEC 61508-1	Allgemeine Anforderungen
IEC 61508-2	Anforderungen an ein programmierbares elektronisches Steuerungssystem
IEC 61508-3	Anforderungen an die Software
IEC 61508-4	Definitionen und Abkürzungen
IEC 61508-5	Beispiele und Methoden zur Bestimmung des SIL
IEC 61508-6	Richtlinien zur Anwendungen der Teile -2 und -3
IEC 61508-7	Übersicht der Messwerte und Methoden

- Safety Integrity Level (SIL)
= Fehlerwahrscheinlichkeit eines gefährlichen Fehlers pro Jahr, unter Randbedingung eines häufigen oder kontinuierlichen Betriebs

Ebene	Fehlerwahrscheinlichkeit
SIL 1	10^{-1} bis 10^{-2}
SIL 2	10^{-2} bis 10^{-3}
SIL 3	10^{-3} bis 10^{-4}
SIL 4	10^{-4} bis 10^{-5}

4.4.1 Berührungslos wirkende Schutzeinrichtungen

- Nur einsetzbar falls Vorschriften gemäß Vorschriften EN 61496-1 und -2 eingehalten werden
- Betriebsarten
  - Mutingbetrieb = Hineinbringen von Gütern in die Maschine, ohne dass diese angehalten wird
    => Vorraussetzung: Sensoren die unzulässiges Eindringen erkennen und abschalten
  - Blanking Betrieb = Ausblenden von Lichtstrahlen, wenn feste Bereiche des Schutzfeldes unterbrochen werden, wird abgeschaltet

4.4.2 Programmierbare Sicherheitssteuerungen

- Mehrkanalig mit sicherheitsgeprüften Softwarebausteinen
- Nur fehlersichere Teil darf sicherheitsgerichtete Aufgaben bearbeiten

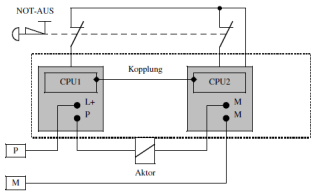


Abbildung 4-37 2-kanalige fehlersichere Speicherprogrammierbare Steuerung