Steuerungstechnik

2. Diskrete Steuerungen

OBERÖSTERREICH

V4.2

Prof. (FH) DI Dr. Franz Auinger

Übersicht

Schaltsysteme

Steuerungssysteme, die binäre Signale verarbeiten, werden als **binäre Systeme** oder meistens als **Schaltsysteme** bezeichnet.

Zur Lösung einer Steuerungsaufgabe ist eine vollständige, übersichtliche und widerspruchsfreie Angabe der notwendigen logischen Verknüpfungen erforderlich.

Als Grundlage für die *Darstellung* des Schaltsystems stehen meist

- *Wirkschema,*
- *Blockschaltbilder,*
- *verbale Beschreibungen* o.ä.

der zu steuernden Anlage zur Verfügung.

Die beiden möglichen Zustände in Schaltsystemen werden als (logisch) 1 und 0 bezeichnet (ja - nein, high - low, wahr - falsch, true - false).

Schaltsysteme

Bei ***reinen Schaltsystemen*** treten *sowohl an der Eingangs-* als auch an der *Ausgangsseite* ***nur binäre Zustände*** auf.

Schaltsysteme enthalten also Elemente, die lediglich zwei (logische) Zustände einnehmen können (Schaltelement, auch Logikelement oder Logikvariable bezeichnet), unabhängig vom tatsächlichen physikalischen Signalverlauf (!).

Schaltsysteme

Steuerungssysteme, die binäre Signale verarbeiten, werden als binäre Systeme oder meistens als **Schaltsysteme** bezeichnet.

Man unterscheidet

- Schaltsysteme *ohne Rückkopplung* der Ausgänge auf die Eingangsseite (**Schaltnetze**) und
- Schaltsysteme *mit einer Rückführung* der Ausgänge auf die Eingangsseite (**Schaltwerke**).

Schaltsysteme werden mathematisch durch **Logikfunktionen** (Logikgleichungen) beschrieben.

Der Zusammenhang zwischen Funktionswert (Ausgangssignal) und Variablen (Eingangssignale) kann mit der **Wertetabelle** (auch **Funktionstabelle** bezeichnet) vollständig und eindeutig beschrieben werden.

Durch mathematische oder graphische Verfahren kann daraus die **Logikfunktion** ermittelt werden.

Grundelemente von Logikfunktionen

Alle Logikfunktionen lassen sich aus vier Grundelementen aufbauen:

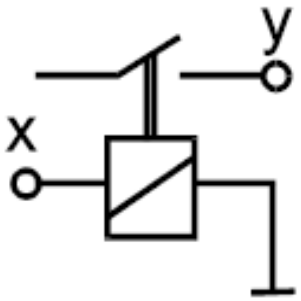
- · Identität (JA-Elemenet)
- · Negation (NICHT-Element)
- · Konjunktion (UND-Element)
- · Disjunktion (ODER-Element)

Anmerkung: Zeitelemente benötigen zusätzlich einen Taktgeber.

Identität (Ja-Element)

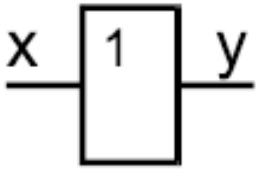
ist eine Funktion mit einem Eingang und folgender Darstellung

Liegt am Schützspuleneingang x eine Spannung an, so liefert auch der Ausgang y eine Spannung. Diese Funktion wird auch als Folgeschaltung bezeichnet.



Schließer (Arbeitskontakt)

Normally Open („no“-Contact)



JA-Element (Identität)

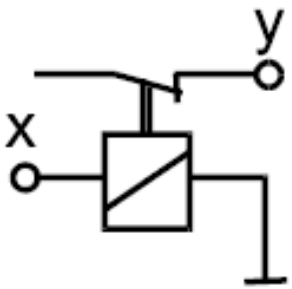
x	F(x)
0	0
1	1

F(x) =y = x

Negation (Nicht-Element)

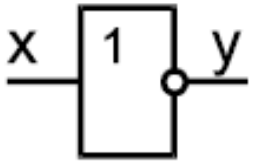
ist eine Funktion mit einem Eingang und folgender Darstellung (vereinbarungsgemäß wird das Zeichen "/" für die Negation verwendet):

Liegt am Schützspuleneingang x eine Spannung an, so liefert der Ausgang y keine Spannung.



Öffner (Ruhekontakt)

Normally Closed („nc“-Contact)



NICHT-Element (Negation)

x	F(x)
0	1
1	0

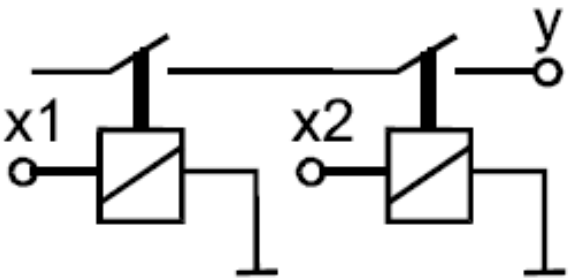
F(x) = y = /x

Alternative Schreibweise:

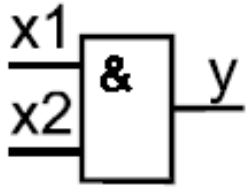
$y = \neg x$

Konjunktion (UND-Element)

ist eine Funktion mit mindestens zwei Eingängen und folgender Darstellung (vereinbarungsgemäß wird das Zeichen "*" für die Konjunktion verwendet):



$F(x1,x2) = y = x1 * x2$



UND-Element (Konjunktion)

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

Laut Definition nimmt die Funktion nur dann den Wert logisch 1 an, wenn beide Eingänge, also x1 UND x2 den Wert logisch 1 führen (UND-Verknüpfung, entspricht einer Serienschaltung).

Alternative Schreibweisen:

$y = x1 \wedge x2$

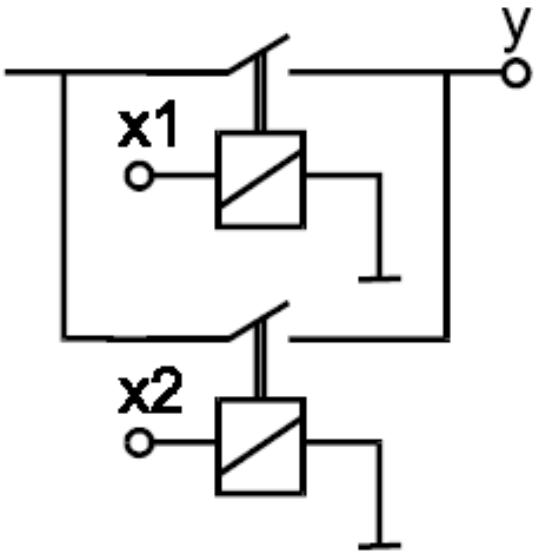
$y = x1 \,\&\, x2$

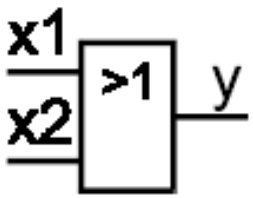
$y = x1\ x2$

Disjunktion (ODER-Element)

ist eine Funktion mit mindestens zwei Eingängen und folgender Darstellung (vereinbarungsgemäß wird das Zeichen "+" für die Disjunktion verwendet):

Definitionsgemäß nimmt die Funktion nur dann den Wert logisch 1 an, wenn wenigsten einer der beiden Eingänge, also x1 ODER x2 den Wert logisch 1 führt (ODER-Verknüpfung, entspricht einer Parallelschaltung).





ODER-Element
(Disjunktion)

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

$$F(x1,x2) = y = x1 + x2$$

Alternative Schreibweise:

$$y = x1 \vee x2$$

Darstellung der Logikfunktionen durch Wertetabellen

Zeile	a	b	c	F(a,b,c)
D0	0	0	0	F0
D1	0	0	1	F1
D2	0	1	0	F2
D3	0	1	1	F3
D4	1	0	0	F4
D5	1	0	1	F5
D6	1	1	0	F6
D7	1	1	1	F7

Zeile	a	b	c	F(a,b,c)
D0	0	0	0	1
D1	0	0	1	1
D2	0	1	0	x
D3	0	1	1	0
D4	1	0	0	0
D5	1	0	1	a
D6	1	1	0	b
D7	1	1	1	0

Don't Care

Mit n Eingangsvariablen ergeben sich 2^n Zeilen

Darstellung der Logikfunktionen: Minterm, Maxterm

Wertetabelle (Funktionstabelle) für 3 Variable und einen Funktionswert:

Zeile	a	b	c	F(a,b,c)
D0	0	0	0	F0
D1	0	0	1	F1
D2	0	1	0	F2
D3	0	1	1	F3
D4	1	0	0	F4
D5	1	0	1	F5
D6	1	1	0	F6
D7	1	1	1	F7

Zeile	a	b	c	F(a,b,c)
D0	0	0	0	1
D1	0	0	1	1
D2	0	1	0	x
D3	0	1	1	0
D4	1	0	0	0
D5	1	0	1	a
D6	1	1	0	b
D7	1	1	1	0

Mathematische Funktion - Begriffe:

Logikprodukt oder Vollkonjunktion als Konjunktion einer ganzen Zeile, auch als Minterm Dj bezeichnet

z.B. Minterm in der 4. Zeile: **D3 = /a * b * c** und diese Funktion nimmt den Wert F3 an.

Logiksumme oder Vollkdisjunktion als Disjunktion einer ganzen Zeile, auch als Maxterm Mj bezeichnet

z.B. Maxterm M3 = **a + /b + /c**

/ nicht b

Merke:
Minterm einer Zeile: Logikprodukt der Zeile
Maxterm einer Zeile: Logiksumme der Zeile

Darstellung der Logikfunktionen

Ein Maxterm ist der negierte Minterm: Mj = /Dj

Für die Funktionswerte Fj sind folgende Werts möglich:

Fj=0, wenn der Minterm Dj in der Funktion sich nicht auswirkt,

Fj=1, wenn der Minterm Dj in der Funktion Auswirkung hat,

Fj=x, wenn der Minterm Dj in der Funktion sich weder mit 0 noch mit 1 auswirkt (sog. Don't-care-Term,)

Fj=F(a,b,c,...), wenn beim Minterm Dj die Funktion den Wert einer anderen Funktion F(a,b,c,...) annimmt.

Die Bezeichnung Minterm kommt daher, daß diese Verknüpfung eine minimale Anzahl von Funktionswerten mit Zustand logisch 1 enthält, nämlich genau einen.

Die Bezeichnung Maxterm kommt daher, daß diese Verknüpfung eine maximale Anzahl von Funktionswerten mit Zustand logisch 1 enthält und eine minimale Anzahl von Funktionswerten mit logisch 0, nämlich wiederum genau einen.

Normalformen der Logikfunktionen: KDNF, KKNF

Kanonisch disjunktive Normalform (KDNF):

$$F(a,b,c,..) = \sum_{i=0}^{i=2n-1} (Fi * Di) = F0 * D0 + F1 * D1 + F2 * D2 + ... + Fn-1 * Dn-1$$

Die KDNF ist die Logiksumme aller Minterme, d.h. die Minterme der einzelnen Zeilen werden disjunktiv (mit ODER) verknüpft.

Kanonisch konjunktive Normalform (KKNF):

$$F(a,b,c,..) = \prod_{i=0}^{i=2n-1} (Fi + Mi) = (F0 + M0) * (F1 + M1) * (F2 + M2) * (.....) * (Fn-1 + Mn-1)$$

Die KKNF ist das Logikprodukt aller Maxterme, d.h. die Maxterme der einzelnen Zeilen werden konjunktiv (mit UND) verknüpft.

Normalformen der Logikfunktionen: DNF, KNF

Wenn die Logikfunktion F(a,b,c,...) unabhängig von eigenen Variablen oder anderen Funktionen ist, d.h. die Funktion für Fi nur die Werte 0 oder 1 annehmen kann, vereinfachen sich die Normalformen zu

Disjunktive Normalform (DNF):

$F(a,b,c,..) = \sum Di$ für i mit Fi = 1

Konjunktive Normalform:

$F(a,b,c,..) = \Pi Mi$ für i mit Fi = 0

Die Minterme und Maxterme einer Logikfunktion stehen in folgender Beziehung zueinander, wobei die Maxterme die negierten Minterme Mi = /Di sind:

(F0 * D0) + (F1 * D1) + (F2 * D2) + ... + (Fn-1 * Dn-1) =

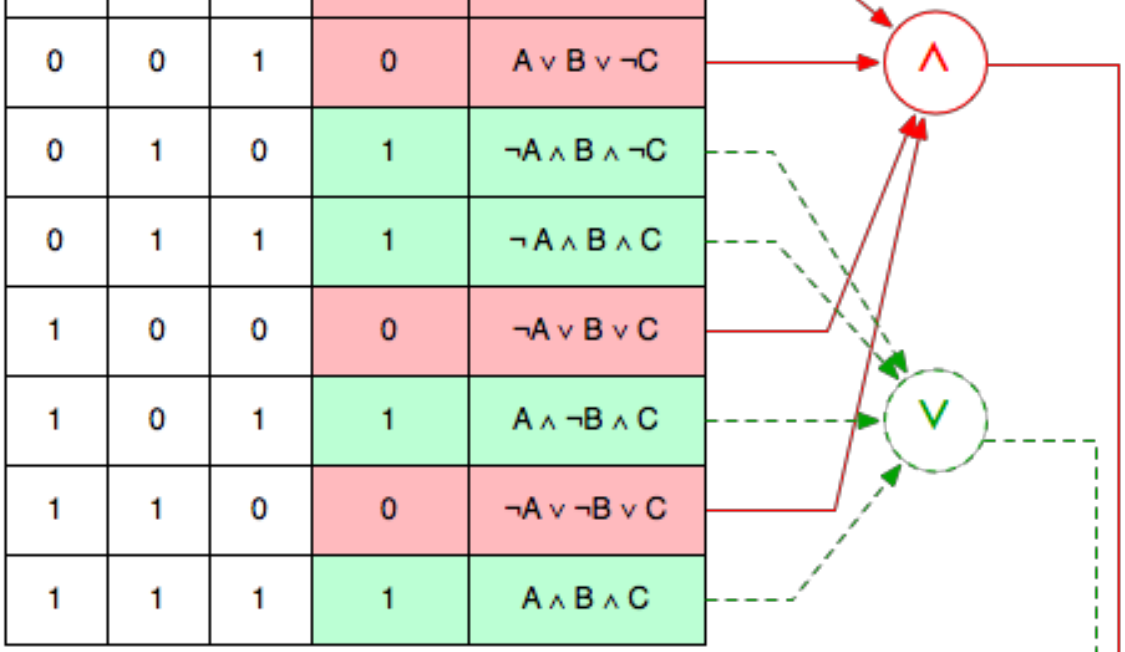
= (/F0 + /D0) * (/F1 + /D1) * (/F2 + /D2) * ... * (/Fn-1 + /Dn-1) =

= (/F0 + M0) * (/F1 + M1) * (/F2 + M2) * .... * (/Fn-1 + Mn-1)

Normalformen der Logikfunktionen: DNF, KNF

Beispiel

A	B	C	Ergebnis	Klausel
0	0	0	0	A ∨ B ∨ C
0	0	1	0	A ∨ B ∨ ¬C
0	1	0	1	¬A ∧ B ∧ ¬C
0	1	1	1	¬ A ∧ B ∧ C
1	0	0	0	¬A ∨ B ∨ C
1	0	1	1	A ∧ ¬B ∧ C
1	1	0	0	¬A ∨ ¬B ∨ C
1	1	1	1	A ∧ B ∧ C



DNF: (¬A ∧ B ∧ ¬C) ∨ (¬ A ∧ B ∧ C) ∨ (A ∧ ¬ B ∧ C) ∨ (A ∧ B ∧ C)

KNF: (A ∨ B ∨ C) ∧ (A ∨ B ∨ ¬C) ∧ (¬A ∨ B ∨ C) ∧ (¬A ∨ ¬B ∨ C)

Quelle: unbekannt

Rechengesetze der Bool‘schen Algebra

- Vertauschungsgesetz (Kommutativgesetz):

  $a * b = b * a$ $\qquad$ $a + b = b + a$

- Verbindungsgesetz (Assoziativgesetz):

  $a * b * c = a * (b * c) = (a * b) * c$
  $a + b + c = a + (b + c) = (a + b) + c$

- Verteilungsgesetz (Distributivgesetz):

  $a * b + a * c = a * (b + c)$
  $(a + b) * (a + c) = a + (b * c)$

- Umkehrungsgesetz (Inversionsgesetz):

  De Morgan’sche Regeln

  $y = a * b$ $\qquad$ $/ y = / (a * b) = / a + / b$
  $y = a + b$ $\qquad$ $/ y = / (a + b) = / a * / b$
  $y = / (a * b)$ $\qquad$ $/ y = a * b = / (/ a + / b)$
  $y = / (a + b)$ $\qquad$ $/ y = a + b = / (/ a * / b)$

  ... gelten auch für n Variable

Rechengesetze

- Klammerregel
- UND vor ODER

  (Punkt- vor Strichrechnung
- Existenz von 1- und 0-Elementen

  a + 0 = a **a * 0 = 0**
  a * 1 = a **a + 1 = 1**
- Existenz eines Komplimentes:

  a + / a = 1 **a * / a = 0**
- **Idempotenzgesetz**

  a * a = a **a + a = a**
- Absorptionsgesetz:

  a + a * b = a **a * (a + b) = a**
- Tautologie:

  / (/ a) = a

Rechengesetze

- weitere hilfreiche Umwandlungen:

  $(a + / b) * b = a * b$

  $a + / a * b = a + b$

- Verallgemeinerung der DeMorgan'schen Regeln nach Shannon:

  $/ F [(a, b, c, ... , n), * , +] =$

  $= F [(/ a, / b, / c, ... , / n), + , *]$

  alle Variablen durch ihre Negation sowie

  alle * durch + und umgekehrt ersetzen

Anwendung der Rechengesetze

Durch Anwendung der Rechenregeln können Logikfunktionen vereinfacht werden. Die Minimierung kann in folgenden Schritten durchgeführt werden:

1. Schritt: in disjunktive Normalform umwandeln, und Gesetz der Wiederholung angewenden

$a + a = a$ und $a * a = a$

2. Schritt: Gegebenenfalls weitere Rechenregeln angewenden

$a + a * b = a$ und $a * b + a * /b = a.$

3. Schritt: Anwendungsmöglichkeit der Kürzungsregeln prüfen

$a + /a * b = a + b$,

$/a + a * b = /a + b$

$a * b + /a * c + b * c = a * b + /a * c$

gegebenenfalls wiederholt anwenden.

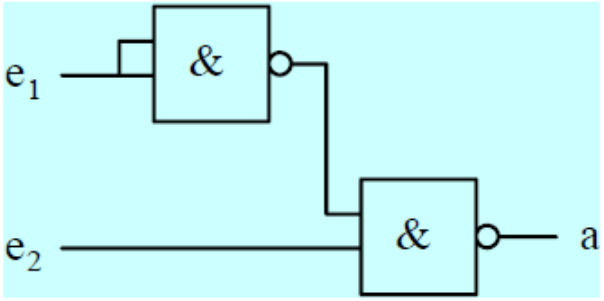
4. Schritt: Umformung der Logikfunktion in die von der Technik zur Realisierung geforderte Darstellung. Ausklammern von Variablen kann dabei u.U. zur weiteren Reduzierung von Variablen führen.

Beispiele

Beispiel 1: Umformung für TTL-Technik (NAND-Gatter)

$$a = e_1 \vee \overline{e_2} = \overline{\overline{e_1} \wedge e_2}$$

De Morgan Regel



Beispiel 2: Vereinfachung einer bool'schen Funktion
(Speicherplatzreduktion und Beschleunigung)

$$\begin{aligned} a &= \overline{(\overline{e_1} \vee \overline{e_2})} \vee (\overline{e_1} \wedge e_2) && \text{De Morgan} \\ &= (e_1 \wedge e_2) \vee (\overline{e_1} \wedge e_2) && \text{Distributivgesetz} \\ &= (e_1 \vee \overline{e_1}) \wedge e_2 && \text{Komplement} \\ &= e_2 \end{aligned}$$

Zusammenfassung

Mit Hilfe der angegebenen Grundgesetze lassen sich boolesche Funktionen vereinfachen und an bestimmte Operatoren anpassen.

Bei einer SPS ist dies aber nur noch bei der Programmierung in Kontaktplan relevant.

Aus der Wertetafel lassen sich die Booleschen Funktionen durch Bildung der disjunktiven und konjunktiven Normalformen (Min- und Maxterme) ableiten.

Mit dem Karnaugh-Diagramm lassen sie sich anschließend vereinfachen.

Dualsystem

Die Darstellung von Zahlen in diskreten Steuerungssystemen wird auf binäre Zustände zurückgeführt. Es stehen somit nur zwei Zustände (Ziffern) zur Verfügung. Das Zahlensystem mit binären Zuständen (0, 1) wird Dualzahlensystem bezeichnet.

Das Dualsystem hat die Basiszahl p = 2 und 10 Ziffern Z = {0, 1}.

$$\text{Dual-Zahlenwert} = \text{Ó}Z_j \overset{n}{*} 2^j$$

Dezimalzahl 206 als Dualzahl dargestellt:

$$\text{Dualzahl } 11001110 = 1*2^7 + 1*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 =$$

$$1*128 + 0*64 + 0*32 + 0*16 + 1*8 + 1*4 + 1*2 + 0*1$$

Zahlenformate

In der digitalen Steuerungstechnik werden unterschiedliche Zahlenformate verwendet, deren Zahlenumfang von der Anzahl der verwendeten Bits abhängt.

Datentyp	Beschreibung	Größe	Bereich	Std.-Anfangs-wert
BOOL	Boolesch	1	0 und 1	0
SINT	8-Bit-Integer	8	-128 bis 127	0
INT	Integer	16	-32768 bis 32767	0
DINT	Double Integer	32	-2.147.483.648 bis +2147.483.647	0
USINT	8-Bit-Integer ohne Vorzei-chen	8	0 bis 255	0
UINT	Integer ohne Vorzeichen	16	0 bis 65535	0
UDINT	Double Integer ohne Vorzei-chen	32	0 bis 4.294.967.295	0
REAL	Gleitkomma-Zahl	32	1.18×10^{-38} bis 3.40×10^{38}	0.0
TIME	Zeitdauer	32	über 24 Tage	t#0s
BYTE	Bitfolge der Länge 8	8	00 bis FF	0
WORD	Bitfolge der Länge 16	16	0000 bis FFFF	0
DWORD	Bitfolge der Länge 32	32	00000000 bis FFFFFFFF	0

Die Wertebereiche und die Bitgröße von elementaren Datentypen sind in der IEC 1131-3 beschrieben.

Die elementaren Datentypen sind in der Tabelle dargestellt.

Die kleinste darstellbare Zahl ist das Bit mit 0 oder 1. Ausgehend vom Oktalsystem (8 Ziffern mit 3 Bits) haben sich folgende Binär-Formate ergeben:

8 Bit = 1 Byte (B, BYTE)
16 Bit = 2 Byte = 1 Wort (W, WORD)
32 Bit = 4 Byte = 2 Wort = 1 Doppelwort (DW, DWORD)
64 Bit = 8 Byte = 4 Wort = 2 Doppelwort = Langwort (LW, LWORD)

Zahlenformate

Die Kenntnis der Zahlenformate ist wesentlich bei der Lösung digitaler Steuerungsaufgaben. Je nach Format ergeben sich unterschiedliche Bitmuster.

Beispiel: SINT (Short Interger, kurze Ganzzahl) verwendet 8 Bit

Dargestellt an einem Eingangsbyte ergibt sich folgende Wertigkeit der einzelnen Bits:

Bit	7	6	5	4	3	2	1	0
EB 4:	VZ	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Bei USINT (Unsigned, vorzeichenlos) wird das Bit 7 ebenfalls als Stellenwert verwendet. Der Zahlenumfang ist demnach 0 bis 255.

Gleitpunktzahlen verwenden 1 Doppelwort, das linke Byte bestimmt den Exponenten, davon Bit 31 für das Vorzeichen, die 3 anderen Byte bestimmen die Mantisse, davon wieder Byte 23 das Vorzeichen.

Zahlenformate : BCD

Binary Coded Decimal

In digitalen Steuerungssystemen haben außerdem Hexadezimalzahlen und BCD-Zahlen große Bedeutung. Die beiden Formate benötigen für jede Dezimalstelle 4 Bit:

Dezimal	Hexadezimal	BCD
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	0001 0000
11	B	0001 0001
12	C	0001 0010
13	D	0001 0011
14	E	0001 0100
15	F	0001 0101

Eingabe mit Wahlrad

von 0 bis 9

Literale

Nach IEC 1131-3 sind für die Darstellung von numerischen Daten, Zeichenfolgen (Strings) und Zeitdaten Literale erforderlich. Beim Eingeben von Werten müssen immer Literale verwendet werden.

Numerische Literale

Die verwendbaren numerischen Literale sind in der folgenden Tabelle dargestellt:

Typ	Beispiele
Integer-Literale	-12 0 123_456 +986
Gleitkomma-Literale	-12.0 0.0 0.4560 3.14159_26
Gleitkomma-Literale mit Exponenten	-1.34E-12 -1.34e-12 1.0E+6
Basis-2-Literale	INT#2#1111_1111
Basis-8-Literale	INT#8#377
Basis-16-Literale	INT#16#FF SINT#16#ff
Boolesche Werte FALSE und TRUE	FALSE TRUE
Boolesche Werte 0 und 1	0, 1

Literale

Zeichenfolge- (String-) Literale

Ein Zeichenfolge-Literal (String-Literal) ist eine Sequenz von null oder mehr Zeichen, die durch zwei einfache Anführungszeichen eingeschlossen sind.

Die verwendbaren Zeichenfolge-Literale sind in der nebenstehenden Tabelle dargestellt:

Typ	Beispiele
Leere Zeichenfolge	''
Zeichenfolge mit einem Leerzeichen	' '
Zeichenfolge mit Text	'dies ist ein Text'

Zeitdauer-Literale

Zeitdauer-Literale können in Stunden, Minuten, Sekunden, Millisekunden und in Kombination dieser Angaben dargestellt werden.

Die verwendbaren Zeitdauer-Literale sind in der nebenstehenden Tabelle dargestellt:

Typ	Beispiele
Kurzes Präfix	T#14ms t#14ms t#12m18s3.5ms T#25h_15m t#25h_15m
Langes Präfix	TIME#14ms time#14ms TIME#25h_15m time#25h_15m

Karnaugh Veitch Diagramm zur Vereinfachung von Logikfunktionen

Zur Vereinfachung von Logikfunktionen kann auch das KV-Diagramm (Karnaugh-Veitch-Diagramm) angewendet werden, wobei jeweils Terme der Funktionen zusammengefaßt werden, die sich nur in einer einzigen Variablen unterscheiden.

Veitch-Tabelle (KV-Diagramm) für 3 Variable:

WERTETABELLE

a	b	c	F(a,b,c)
0	0	0	1
0	0	1	1
0	1	0	x
0	1	1	0
1	0	0	0
1	0	1	a
1	1	0	b
1	1	1	0

VEITCH-TABELLE

	b	0	1	1	0
a	c	0	0	1	1
0		1	x	0	1
1		0	b	0	a

Karnaugh Veitch Diagramm für 4 und 5 Variablen

VEITCH-TABELLE

a	b	c \ d	0	1	1	0
		0	0	1	1	
0	0		1	0	1	0
0	1		0	1	0	1
1	1		1	0	0	1
1	0		0	0	1	0

VEITCH-TABELLE

a	b	e	0				1			
0	0		1	0	1	0	0	0	0	0
0	1		0	1	0	1	0	1	1	0
1	1		1	0	0	1	0	0	1	1
1	0		0	0	1	0	0	0	1	0
		c	0	1	1	0	0	1	1	0
		d	0	0	1	1	0	0	1	1

Beispiel Karnaugh Veitch Diagramm für 3 Variablen

a	b	c	F(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

a	b \ c	0	1
0	0	0	0
0	1	1	0
1	1	1	0
1	0	0	0

$F(a,b,c) = b^*/c$

$F(a,b,c) = /a * b * /c + a * b * /c = b * /c * (b + /b) = b * /c$

Vereinfachungsregeln

Es gelten folgende Vereinfachungsregeln:

- möglichst große rechteckige Blöcke mit Logik-1, Größe: 2^n, n = 1, 2, 3 ...

  Logikfunktion = Logikprodukt aus jenen Variablen, die entlang den Kanten nur mit einem Zustand (logisch 0 oder logisch 1) vorkommen.

- Bei mehreren Blöcken:

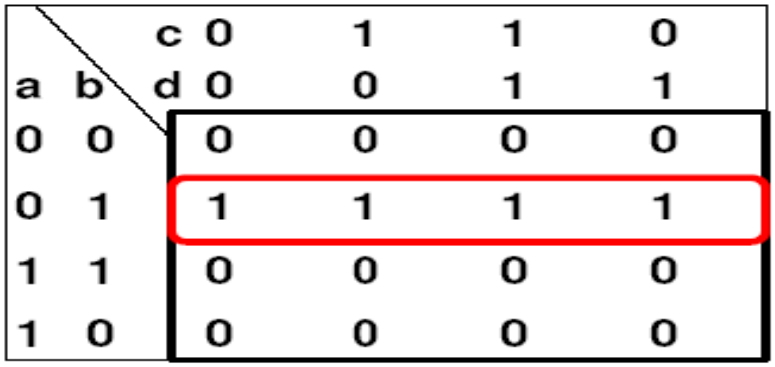
  Logikfunktion = Logiksumme der Logikprodukte der einzelnen Blocke.

- Eine Überlappung der Blocke ist erlaubt bzw. empfehlenswert.

- Unbestimmte Funktionswerte x können, müssen aber nicht in Blöcke eingeschlossen werden.

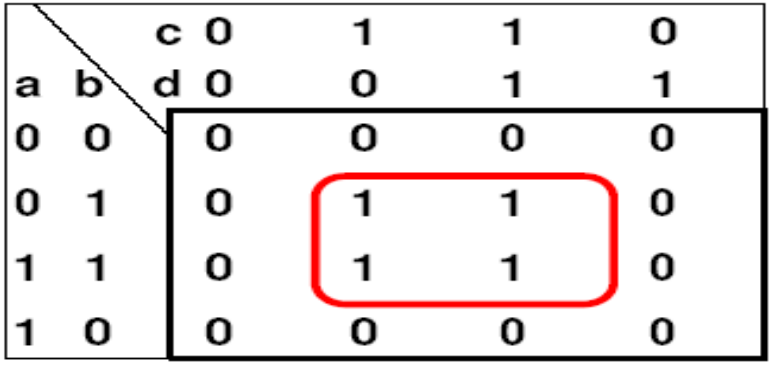
Beispiele

F(a,b,c,d) = /a * b



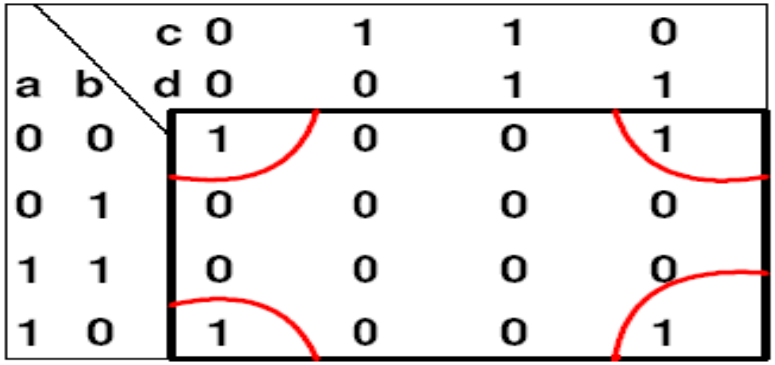
a b \ c d	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	1	1	1	1
1 1	0	0	0	0
1 0	0	0	0	0

F(a,b,c,d) = b * c



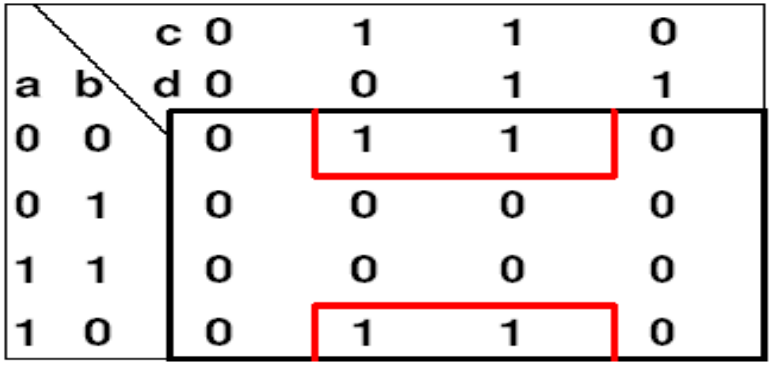
a b \ c d	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	0	1	1	0
1 1	0	1	1	0
1 0	0	0	0	0

F(a,b,c,d) = /b * /c



a b \ c d	0 0	0 1	1 1	1 0
0 0	1	0	0	1
0 1	0	0	0	0
1 1	0	0	0	0
1 0	1	0	0	1

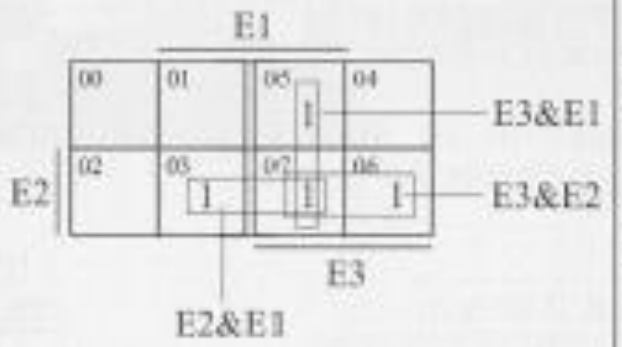
F(a,b,c,d) = /b * c



a b \ c d	0 0	0 1	1 1	1 0
0 0	0	1	1	0
0 1	0	0	0	0
1 1	0	0	0	0
1 0	0	1	1	0

Vereinfachung von Schaltfunktionen

Zeile	E4	E3	E2	E1	A
00_8	0	0	0	0	0
01_8	0	0	0	1	0
02_8	0	0	1	0	0
03_8	0	0	1	1	1
04_8	0	1	0	0	0
05_8	0	1	0	1	1
06_8	0	1	1	0	1
07_8	0	1	1	1	1



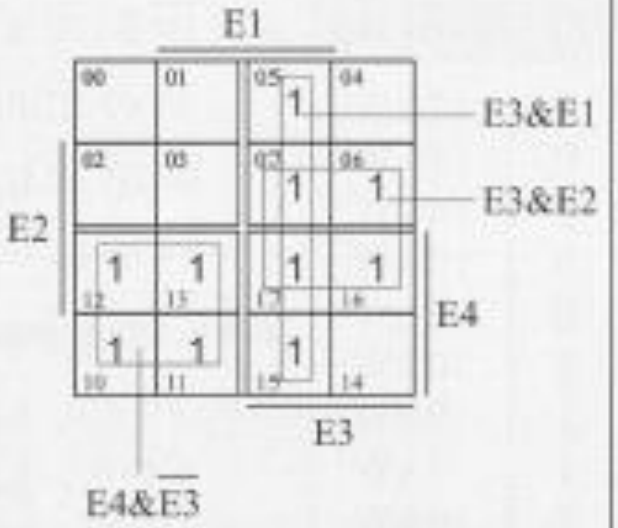
Minimierte DNF für den Ausgang A:

$$A = E2\ E1 \vee E3\ E1 \vee E3\ E2$$

Quelle: Wellenreuther, Zastrow: Automatisieren mit SPS. Übersichten und Übungsaufgaben

Vereinfachung von Schaltfunktionen

Zeile	E4	E3	E2	E1	A
00_8	0	0	0	0	0
01_8	0	0	0	1	0
02_8	0	0	1	0	0
03_8	0	0	1	1	0
04_8	0	1	0	0	0
05_8	0	1	0	1	1
06_8	0	1	1	0	1
07_8	0	1	1	1	1
10_8	1	0	0	0	1
11_8	1	0	0	1	1
12_8	1	0	1	0	1
13_8	1	0	1	1	1
14_8	1	1	0	0	0
15_8	1	1	0	1	1
16_8	1	1	1	0	1
17_8	1	1	1	1	1



Minimierte DNF für den Ausgang A:

$$A = E3\,E1 \vee E3\,E2 \vee E4\,\overline{E3}$$

Quelle: Wellenreuther, Zastrow: Automatisieren mit SPS. Übersichten und Übungsaufgaben

Nebenbedingungen, Sperrfunktion

In manchen Fällen haben einzelne Variable einer Logikfunktion durch vorgegebene Funktionswerte bestimmte Bedingungen (Eingangsbelegungen) zu erfüllen, während die übrigen frei wählbar sind.

Man bildet mit diesen Variablen eine Logikfunktion, die diese Bedingungen erfüllt, und bezeichnet diese Funktion als **Nebenbedingung** N(k,l,m,...).

Die Negation der Nebenbedingung schließt hingegen diesen Funktionswert aus und man bezeichnet diese Funktion daher als **Sperrfunktion** S(k,l,m,...) = /N().

Bei erlaubten Zuständen der Variablen (Eingangsbelegungen) nimmt die Sperrfunktion den Wert 0 an.

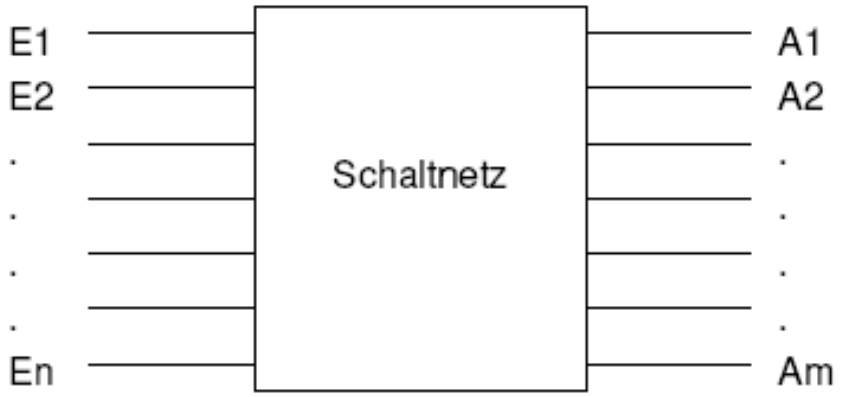
Eine Sperrfunktion entspricht einer „**Verriegelungsschaltung**“. (z.B. Linkslaufschütz und Rechtslaufschütz sind gegenseitig gegen gleichzeitige Betätigung verriegelt).

In der Steuerungstechnik gibt es „**Exklusivverriegelungen**“ (gegenseitige Verriegelungen) und „**Reihenfolgeverriegelungen**“.

Schaltnetze

In den Logikfunktionen von Schaltnetzen existieren lediglich Eingangssignale als Variable, Ausgänge oder Funktionswerte anderer Logikfunktionen kommen in der Logikfunktion nicht vor. Eine solche Verknüpfungsstruktur bewirkt, daß zu jedem beliebigen Zeitpunkt der Zustand der Ausgangssignale allein vom Zustand der Eingangssignale abhängt, oder anders ausgedrückt zu jedem Eingangszustand gibt es immer den gleichen Ausgangszustand.

Der Zustand der Ausgangssignale kann bei solchen Steuerungen nicht gespeichert werden, man spricht daher von Verknüpfungssteuerungen ohne Speicherverhalten. Eine Meldeleuchte wird beispielsweise nur solange leuchten, als an einem der Eingänge ein entsprechendes Signal anliegt.



Die Beschreibung der Logikfunktion gelingt vollständig mit der Wertetabelle, in der lediglich Funktionswerte mit logisch 0 oder logisch 1 vorkommen, sodaß die Logikfunktion als Disjunktive oder Konjunktive Normalform ermittelt werden kann.

Logikfunktionen von Schaltnetzen lassen sich häufig mit den oben beschriebenen Verfahren vereinfachen.

Funktionen

In der IEC 61131-3 werden verschiedene Typen von Standardfunktionen aufgeführt:

- Funktionen zur Typumwandlung
- Numerische Funktionen
- Arithmetische Funktionen
- Bitweise boolesche Funktionen
- Bit-Folge-Funktionen
- Funktionen für Auswahl
- Funktionen für Vergleich
- Funktionen für Zeichenfolgen

Funktionen entsprechen Schaltnetzen!

Standardfunktionen

- numerische Funktionen:
  ABS, SQRT, LN, LOG, EXP, SIN,
  COS, TAN, ASIN, ACOS, ATAN
- Funktionen zur Bitmanipulation:
  SHL, SHR, ROL, ROR
- Funktionen zur Auswahl:
  MAX, MIN
- Funktionen für Zeichenfolgen:
  REPLACE
- Funktionen zur Typwandlung:
  *_TO_**, TRUNC

Shift Left Multiplikation um 2 bei Binär

Rotate Left

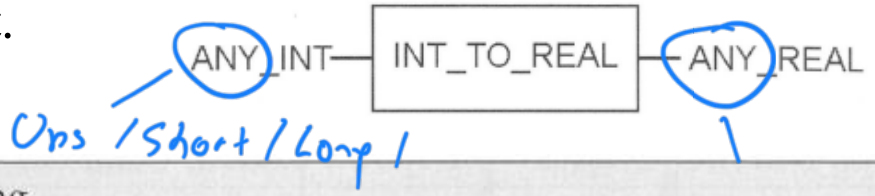
Typumwandlung

hinten Komma wegschneiden

Ergänzungen zu Umwandlungsfunktionen

Umwandlungsfunktionen nach DIN EN 61131-3

Nach der Norm DIN EN 61131-3 haben Typumwandlungsfunktionen die Form * _ TO _ **, wobei „*" der Typ der Eingangsvariablen IN ist und „**" der Typ der Ausgangsvariablen OUT ist.



Name	Beschreibung
BCD_TO_INT	Der Bitfolge-Datentyp BCD am Eingang wird in den Datentyp INTEGER umgesetzt.
INT_TO_BCD	Der Datentyp INTEGER am Eingang wird in den Bitfolge-Datentyp BCD umgesetzt.
INT_TO_REAL	Der Datentyp INTEGER am Eingang wird in den Datentyp REAL umgewandelt.
REAL_TO_INT	Der Datentyp REAL am Eingang wird in den Datentyp INTEGER umgesetzt. Die Umwandlung vom Typ REAL oder LREAL nach SINT, INT, DINT oder LINT muss auf die nächste ganze Zahl runden. Somit gilt: 1.4 = 1; 1.5 = 2; –1.4 = - 1; –1.6 = –2;
TRUNC	Eine Ausnahme in der Darstellung bildet die Funktion TRUNC. Der Datentyp REAL am Eingang wird dabei in den Datentyp INTEGER umgesetzt. Mit dieser Funktion werden die Nachkommastellen abgeschnitten. Somit gilt: 1.4 = 1; 1.5 = 1; -1.4 = –1; –1.6 = –1;

Deklaration auf Aufruf von Funktionen

Funktionsdeklaration:

Bsp.:

```
FUNCTION Differenz : INT
    VAR_INPUT
        w1, w2 : INT;
    END_VAR
    Differenz := w1-w2;
END_FUNCTION
```

Typ des Rückgabewertes

Funktionsaufruf:

Bsp.:

```
VAR
    i1 : INT := 12; i2 : INT := 7;
    i3 : INT;
END_VAR
...
i3 := Differenz(w1:=i1, w2:=i2); (* MIT Formalparam. *)
i3 := Differenz(i1, i2); (* OHNE Formalparam. *)
```

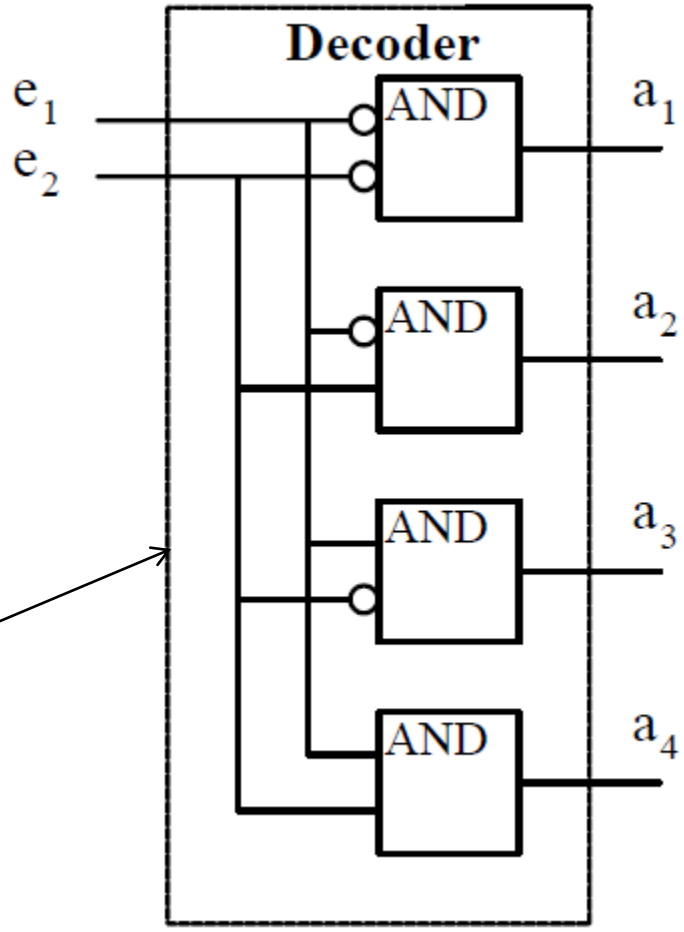
Vor einem Funktionsaufruf müssen keine *Instanzen* in der Deklaration erzeugt werden.

Beispiel: Decoder-Funktion

Wertetafel

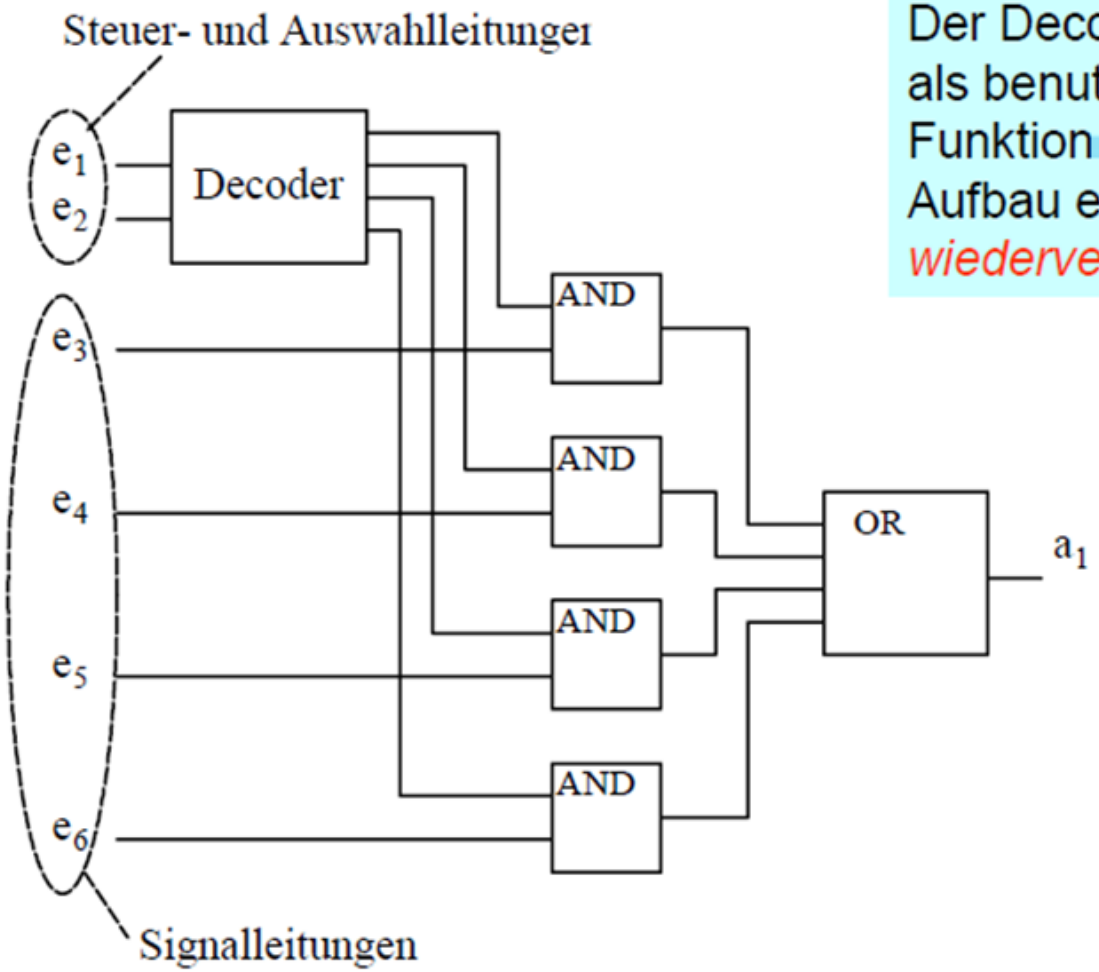
e_1	e_2	a_1	a_2	a_3	a_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

FB-Netzwerk des Decoders



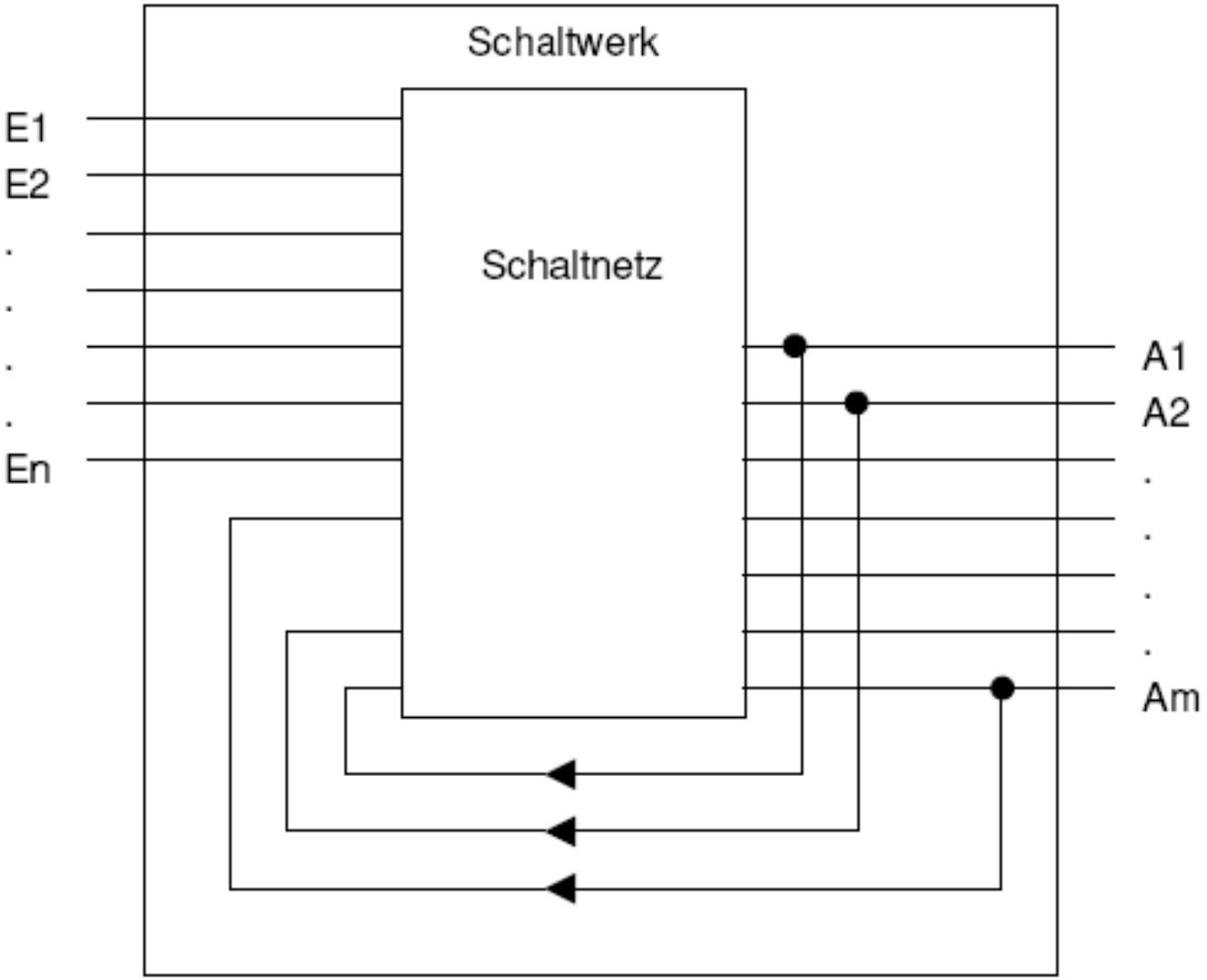
Decoder-Funktion

Beispiel: Einsatz der Decoder-Funktion in einem Multiplexer



Der Decoder selbst kann als benutzerdefinierte Funktion zum Aufbau eines Multiplexers *wiederverwendet* werden.

Schaltwerke



Übersicht

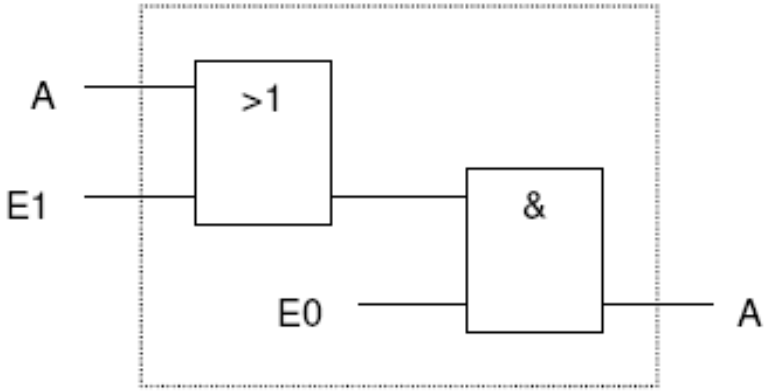
In der IEC 61131-3 werden verschiedene Typen von Standard-Funktionsbausteinen aufgeführt:

- Bistabile Elemente (Binäre Speicherelemente)
- Funktionsbausteine für Flankenerkennung
- Funktionsbausteine Zeitgeber
- Funktionsbausteine Zähler

Funktionsbausteine weisen Speicherverhalten auf. Es handelt sich somit um Schaltwerke!

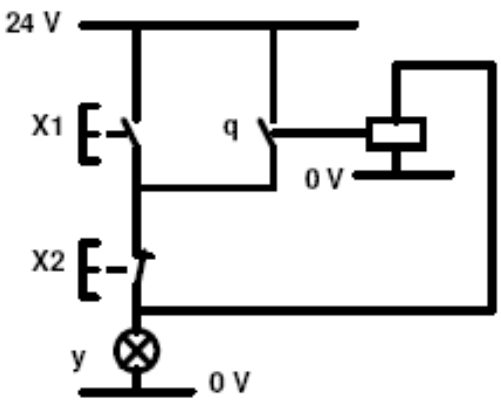
Binäre Schaltelemente

Verwendet man in einer Schützschaltung für E0 einen Öffnerkontakt, so stellt der nebenstehende Funktionsplan eine Selbsthalteschaltung dar.



Schützkontakt A und Taster E1 bilden in der elektrischen Schaltung einen Parallelzweig (ODER), gefolgt vom Taster E0 für vorrangig AUS in Serie (UND).

Wird festgelegt, „vorrangig Einschalten" gelten soll, dann ist der Ausgang A logisch 1, wenn E1 und E0 gleichzeitig logisch 1 sind. Die Schaltung für „Vorrangig Einschalten" läßt sich wie im Abschnitt 2.6 gezeigt entwickeln.



Signal X1 = 1 : Speicher wird gesetzt

Signal X2 = 1 : Speicher wird gelöscht.

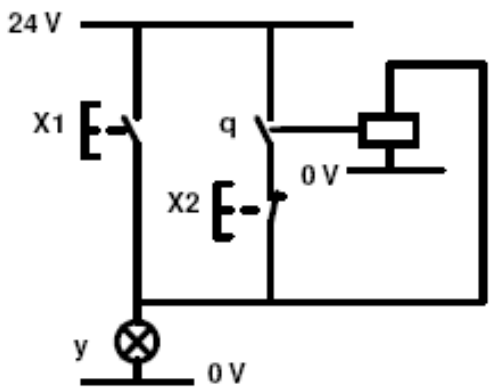
Hilfskontakt q = 1, wenn Speicher gesetzt

In der Schaltung dominiert Schalter X2,

daher **VORRANGIG AUSSCHALTEN**

(RS-Flip-Flop)

Binäre Schaltelemente



VORRANGIG EINSCHALTEN

Schalter X1 (Setzen) dominiert

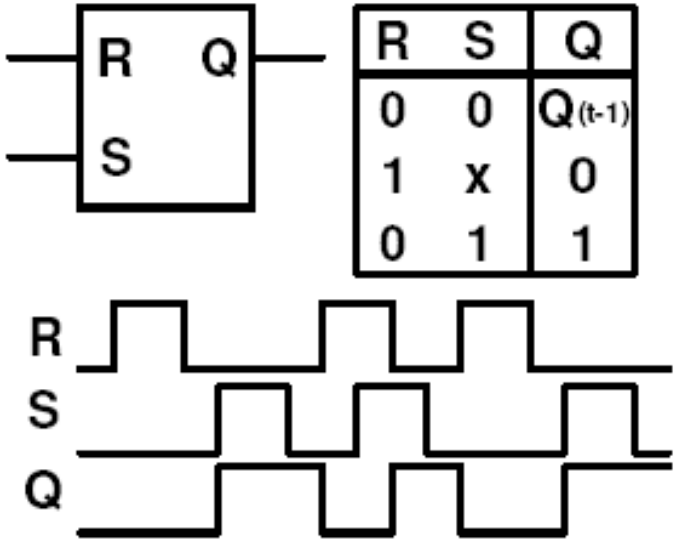
Ausschalten ist nur möglich, wenn X1 = 0

(SR-Flip-Flop)

Wird die Schaltung als Schaltwerk dargestellt, so besitzt sie 2 Eingänge (Setzen, Rücksetzen) und einen Ausgang (RS-Flip-Flop, SR-Flip-Flop).

RS-Flip-Flop

Symbol, Wertetabelle und Zeitdiagramm:



R	S	Q
0	0	$Q_{(t-1)}$
1	x	0
0	1	1

Wenn kein Eingangssignal anliegt, hängt der Zustand des Ausgangssignals von der Vorgeschichte ab (Zeitfolge = Sequenz).

Setzen (Set) ist nur möglich, wenn kein Rücksetz-Signal (Reset) anliegt.

Schaltungen mit Speicherelementen werden auch als „Sequentielle Logik“ bezeichnet.

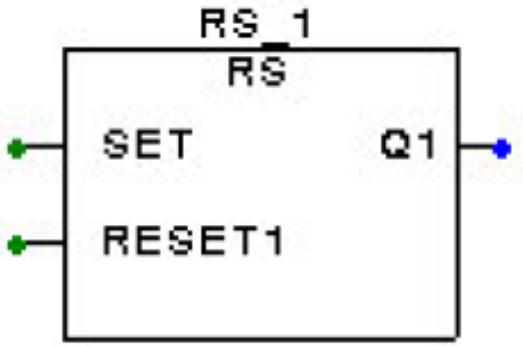
RS-Flip-Flop nach IEC 1131-3

Der bistabile Funktionsbaustein RS realisiert ein vorrangiges Rücksetzen des Ausganges Q1. Wenn der Eingang SET = TRUE ist, wird Ausgang Q1 gesetzt. Q1 bleibt gesetzt, selbst wenn SET FALSE wird. Q1 wird erst zurückgesetzt, wenn RESET1 = TRUE ist. Sind beide Eingänge TRUE, wird der Ausgang Q1 durch RESET1 auf FALSE gesetzt.

Wird der Funktionsbaustein zum ersten Mal aufgerufen, ist Q FALSE.

Parameter	Datentypen	Beschreibung
SET	BOOL	Wenn TRUE, wird Q1 gesetzt
RESET1	BOOL	Wenn TRUE, wird Q1 vorrangig zurückgesetzt
Q1	BOOL	Ausgangswert

Anmerkung: Alle Parameter können negiert werden.



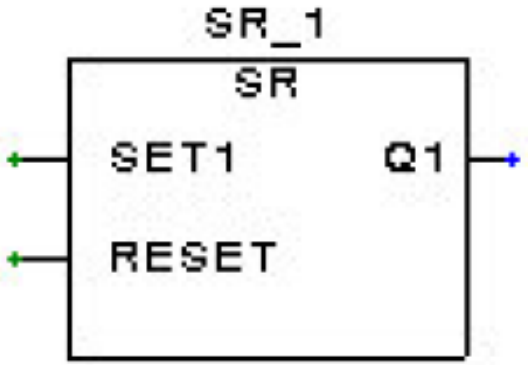
SR-Flip-Flop nach IEC 1131-3

Dieser bistabile Funktionsbaustein realisiert ein vorrangiges Setzen des Ausganges Q1. Wenn der Eingang SET1 = TRUE ist, wird Ausgang Q1 gesetzt. Q1 bleibt gesetzt, selbst wenn SET FALSE wird. Q1 wird erst zurückgesetzt, wenn RESET = TRUE ist. Sind beide Eingänge TRUE, wird der Ausgang Q1 durch SET1 auf TRUE gesetzt.

Wird der Funktionsbaustein zum ersten Mal aufgerufen, ist Q FALSE.

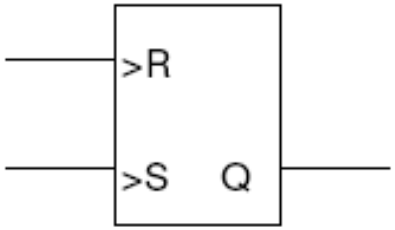
Parameter	Datentypen	Beschreibung
SET1	BOOL	Wenn TRUE, wird Q1 vorrangig gesetzt
RESET	BOOL	Wenn TRUE, wird Q1 zurückgesetzt
Q1	BOOL	Ausgang

Anmerkung: Alle Parameter können negiert werden.



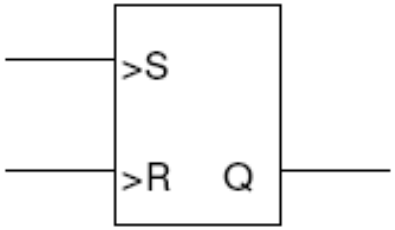
RS-Flip-Flop dynamisch

Häufig werden speichernde Elemente durch die steigende oder fallende Flanke z.B. eines Impulses getriggert. Man bezeichnet diese Signale als dynamische Eingangssignale. Der Eingang bleibt nur während des Zustandwechsels auf logisch 1, bei der nächstfolgenden Abfrage ohne Zustandswechsel wird der Eingang wieder logisch 0. Im Gegensatz dazu wird bei statischer Signalerkennung der Signalzustand über die ganze Signalbreite erkannt.



R	S	Q(t)
X	X	Q(t-1)
X	⌋	1
⌋	X	0
⌋	⌋	0

RS-Flip-Flop dynamisch, Reset dominiert, steigende Flanke



R	S	Q(t)
X	X	Q(t-1)
X	⌉	1
⌉	X	0
⌉	⌉	1

RS-Flip-Flop dynamisch, Set dominiert, fallende Flanke

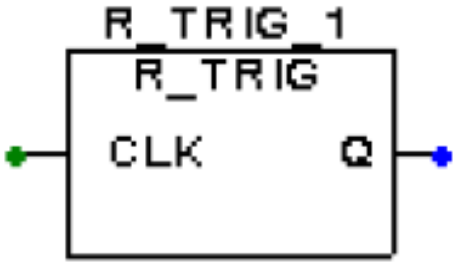
Flankenerkennung

Die folgenden Funktionsbausteine für Flankenerkennung sind verfügbar:

F_TRIG	Erkennung der fallenden Flanke
R_TRIG	Erkennung der steigenden Flanke

R_TRIG

Dieser Funktionsbaustein zur Flankenerkennung erkennt eine steigende Flanke. Wird eine steigende Flanke am Eingang CLK entdeckt, wechselt der Ausgang Q von FALSE zu TRUE. Q bleibt TRUE bis zur nächsten Ausführung des Funktionsbausteins.

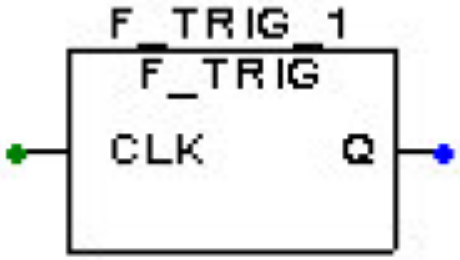


Wird der Funktionsbaustein zum ersten Mal aufgerufen, ist Q FALSE bis die erste Flanke erkannt wird.

Parameter	Datentypen	Beschreibung
CLK	BOOL	erkennt eine steigende Flanke
Q	BOOL	Wird eine steigende Flanke entdeckt, wechselt Q von FALSE zu TRUE.

Flankenerkennung

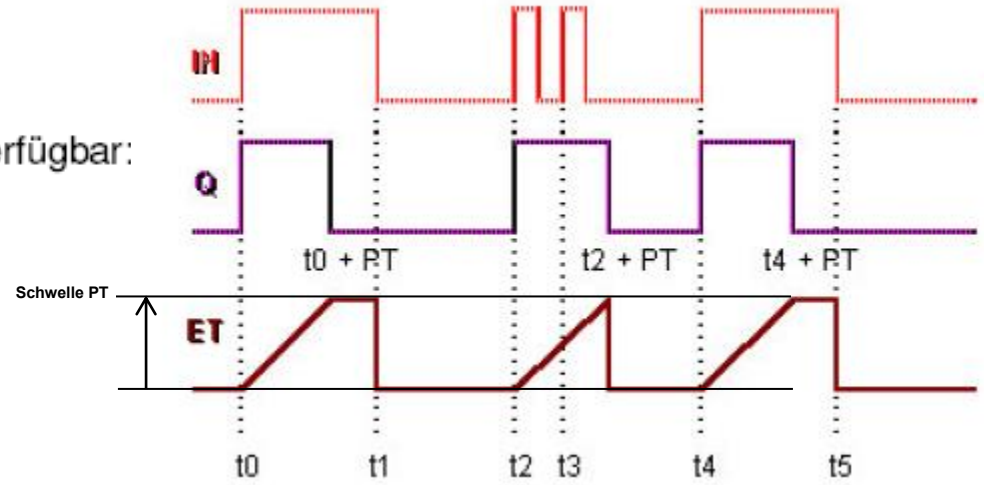
Dieser Funktionsbaustein zur Flankenerkennung erkennt eine fallende Flanke. Wird eine fallende Flanke am Eingang CLK entdeckt, wechselt der Ausgang Q von FALSE zu TRUE. Q bleibt TRUE bis zur nächsten Ausführung des Funktionsbausteins.



Wird der Funktionsbaustein zum ersten Mal aufgerufen, ist Q FALSE bis die erste Flanke erkannt wird.

Parameter	Datentypen	Beschreibung
CLK	BOOL	erkennt eine fallende Flanke
Q	BOOL	Wird eine fallende Flanke entdeckt, wechselt Q von FALSE zu TRUE.

Zeitfunktionen

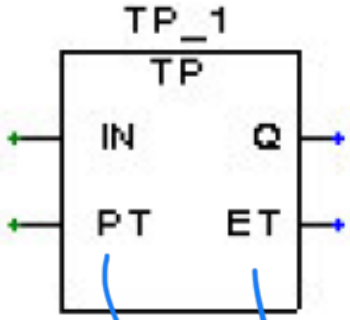


Die folgenden Zeitgeber sind nach IEC 1131-3 verfügbar:

TP	Puls
TON	Zeitgeber für Einschalt-Verzögerung
TOF	Zeitgeber für Ausschalt-Verzögerung

TP

Dieser Zeitgeber erzeugt einen Puls.



Wenn der Eingang IN von FALSE nach TRUE umschaltet, wird am Ausgang Q für die Dauer des Intervalls PT ein Puls generiert. Die bereits verstrichene Zeitspanne wird am Ausgang ET angezeigt. Wenn IN ein zweites Mal TRUE wird während PT noch andauert, hat dies keine Auswirkung auf die Dauer des am Ausgang Q erzeugten Pulses.

Parameter	Datentypen	Beschreibung
IN	BOOL	Wird eine steigende Flanke entdeckt, wird ein Puls erzeugt.
PT	TIME	voreingestelltes Zeitintervall für den Puls
Q	BOOL	TRUE, wenn IN = TRUE und ET < PT
		FALSE, wenn IN = FALSE und ET >= PT
ET	TIME	verstrichenes Zeitintervall

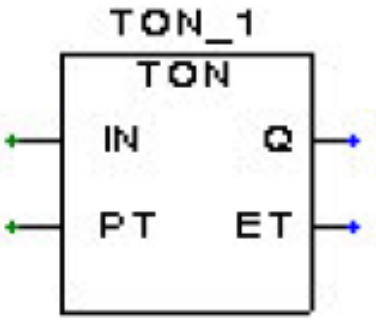
Preset time

Elapsed time

Anmerkung: Der Eingang IN und der Ausgang Q können negiert werden.

Zeitfunktionen: TON

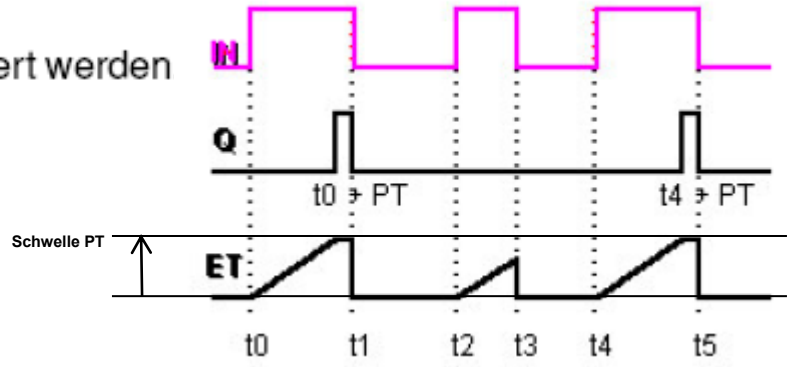
Dieser Zeitgeber realisiert eine Einschalt-Verzögerung.



Wenn der Eingang IN von FALSE nach TRUE umschaltet, wird das Einschalten des Ausgangs um das am Eingang PT definierte Zeitintervall verzögert. Nachdem die Zeit an PT abgelaufen ist, schaltet der Ausgang Q auf TRUE. Die bereits verstrichene Zeitspanne wird am Ausgang ET angezeigt.

Parameter	Datentypen	Beschreibung
IN	BOOL	Wird eine steigende Flanke erkannt, wird die Einschalt-Verzögerung gestartet.
PT	TIME	voreingestelltes Zeitintervall für die Verzögerung
Q	BOOL	TRUE, wenn IN = TRUE und ET < PT
		FALSE, wenn IN = FALSE und ET >= PT
ET	TIME	verstrichenes Zeitintervall

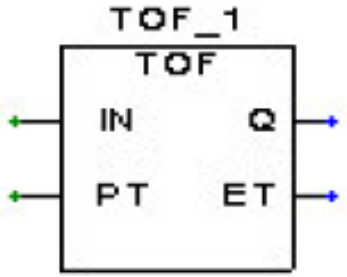
Anmerkung: Der Eingang IN und der Ausgang Q können negiert werden



Zeitfunktionen: TOF

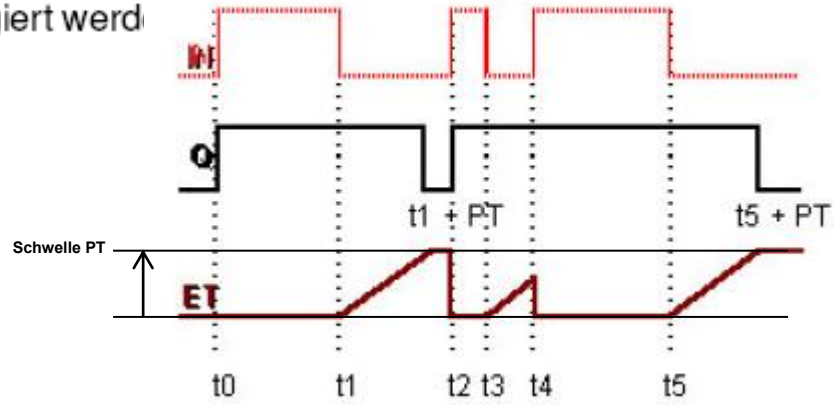
Dieser Zeitgeber realisiert eine Ausschalt-Verzögerung.

Wenn der Eingang IN von TRUE nach FALSE umschaltet, wird das Ausschalten des Ausgangs um das am Eingang PT definierte Zeitintervall verzögert. Nachdem die Zeit an PT abgelaufen ist, schaltet der Ausgang Q auf FALSE. Die bereits verstrichene Zeitspanne wird am Ausgang ET angezeigt.



Parameter	Datentypen	Beschreibung
IN	BOOL	Wird eine fallende Flanke erkannt, wird die Ausschalt-Verzögerung gestartet.
PT	TIME	voreingestelltes Zeitintervall für die Verzögerung
Q	BOOL	TRUE, wenn IN = TRUE und ET < PT FALSE, wenn IN = FALSE und ET >= PT
ET	TIME	verstrichenes Zeitintervall

Anmerkung: Der Eingang IN und der Ausgang Q können negiert werd



Angabe von Zeitdauern als Konstante:
T#14.7ms
TIME#3h_12m_4.3s

Einheiten:
d ..... Tag
h ..... Stunde
m .... Minute
s ..... Sekunde
ms .. Millisekunde

Zählfunktionen

Die folgenden Zähler sind nach IEC 1131-3 verfügbar:

CTU Aufwärts-Zähler

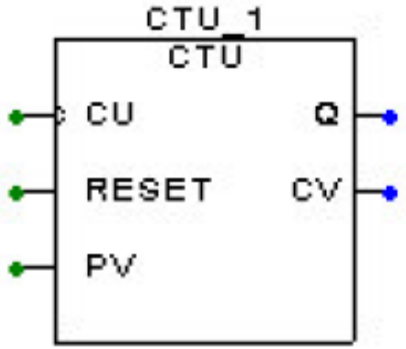
CTD Abwärts-Zähler

CTUD Auf-Abwärts-Zähler

CTU

Dieser Zähler zählt aufwärts. Wenn eine steigende Flanke am Eingang CU erkannt wird und RESET = FALSE, dann wird CV um eins erhöht. Wenn der letzte Zählwert (PV) erreicht ist, wird TRUE am Ausgang Q ausgegeben und der Funktionsbaustein zählt nicht mehr weiter.

Wenn RESET = TRUE ist, wird der Zähler mit 0 initialisiert. Um den Zählvorgang zu starten, muß der Eingang RESET FALSE sein. Sonst wird der Zähler stets neu initialisiert.



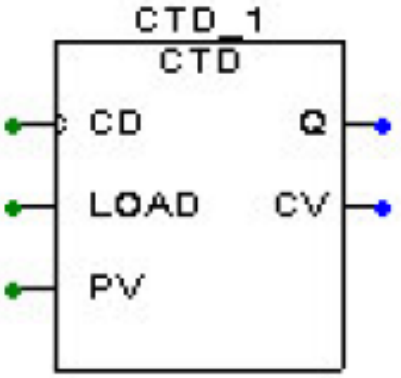
Parameter	Datentypen	Beschreibung
CU	BOOL	Wird eine steigende Flanke erkannt, wird CV um eins erhöht
RESET	BOOL	Wenn TRUE, wird der Zähler mit 0 initialisiert
		Wenn FALSE, wird gezählt
PV	INT	Voreinstellungswert
Q	BOOL	TRUE; wenn CV = PV
CV	INT	Zählergebnis

Anmerkung: Der Ausgang Q kann negiert werden.

Zählfunktionen

CTD

Dieser Zähler zählt abwärts. Wenn eine steigende Flanke am Eingang CD erkannt wird und LOAD = FALSE, dann wird CV um eins vermindert. Wenn der letzte Zählwert (0) erreicht ist, wird TRUE am Ausgang Q ausgegeben und der Funktionsbaustein zählt nicht mehr weiter.



Wenn LOAD = TRUE ist, wird der Zähler mit dem Wert des Einganges PV initialisiert. Um den Zählvorgang zu starten, muß der Eingang LOAD FALSE sein. Sonst wird der Zähler stets neu initialisiert.

Parameter	Datentypen	Beschreibung
CD	BOOL	Wird eine steigende Flanke erkannt, wird CV um eins vermindert.
LOAD	BOOL	Wenn TRUE, wird der Zähler mit PV initialisiert.
		Wenn FALSE, wird gezählt.
PV	INT	Voreinstellungswert
Q	BOOL	TRUE; wenn CV =0
CV	INT	Zählergebnis

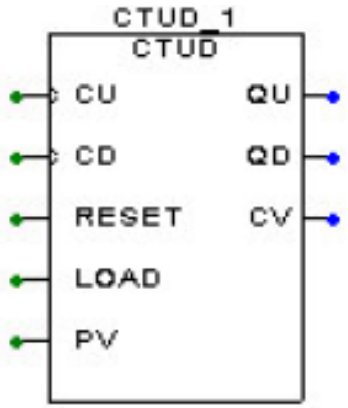
Anmerkung: Der Ausgang Q kann negiert werden.

Zählfunktionen

CTUD

Dieser Zähler zählt auf- und abwärts. Wenn eine steigende Flanke am Eingang CU erkannt wird, dann wird CV um eins erhöht. Wenn eine steigende Flanke am Eingang CD erkannt wird, dann wird CV um eins vermindert. Wenn CV = PV ist, wird TRUE am Ausgang QU ausgegeben. Wenn CV = 0 ist, wird TRUE am Ausgang QD ausgegeben.

Wenn der Eingang RESET = TRUE ist, wird der Zähler mit 0 initialisiert. Wenn der Eingang LOAD = TRUE ist, wird der Zähler mit PV initialisiert. Um den Zählvorgang zu starten, müssen die Eingänge RESET und LOAD FALSE sein. Sonst wird der Zähler stets neu initialisiert.



Parameter	Datentypen	Beschreibung
CU	BOOL	Wird eine steigende Flanke erkannt, wird CV um eins erhöht.
CD	BOOL	Wird eine steigende Flanke erkannt, wird CV um eins vermindert.
RESET	BOOL	Wenn TRUE, wird der Zähler mit 0 initialisiert.
		Wenn FALSE, wird gezählt.
LOAD	BOOL	Wenn TRUE, wird der Zähler mit PV initialisiert.
		Wenn FALSE, wird gezählt.
PV	INT	Voreinstellungswert
QU	BOOL	TRUE; wenn CV = PV
QD	BOOL	TRUE; wenn CV =0
CV	INT	Zählergebnis

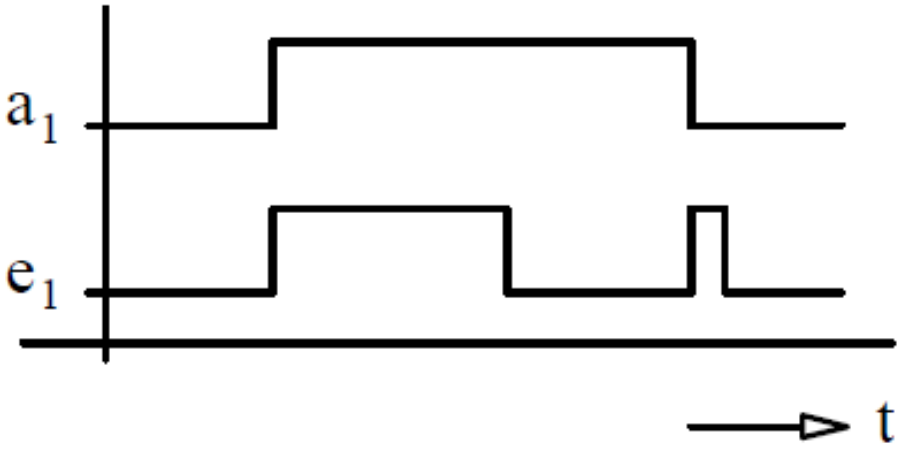
Anmerkung: Die Ausgänge QU und QD können negiert werden.

Beispiel: Ein-Aus- Steuerung mit Kurzkontaktgeber

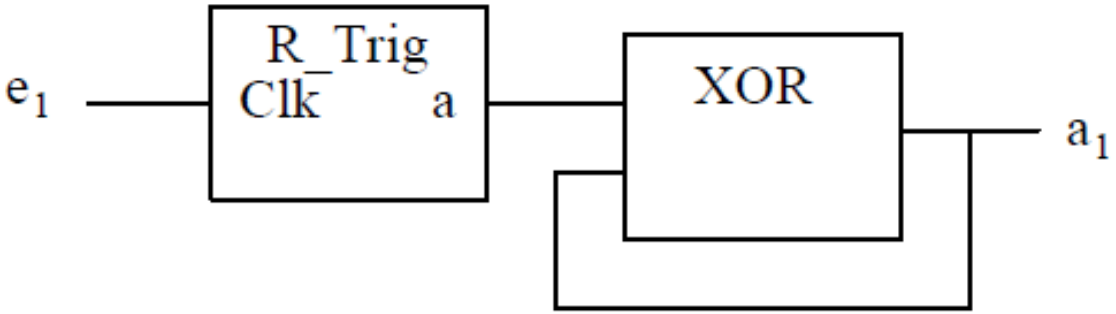
Ein – Ausschalten eines Ausgangs a1 über Tastendrücke auf einen Eingang e1:

Ein- und Aussteuerung mit einem Taster

Toggle-Switch

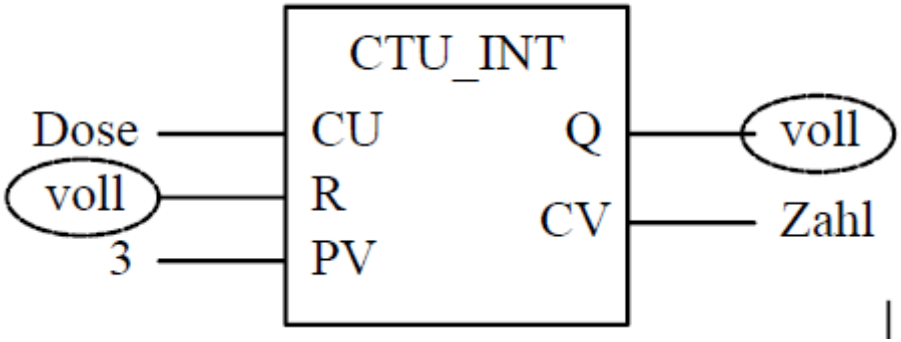


Lösung: Realisierung mit Standardfunktionen

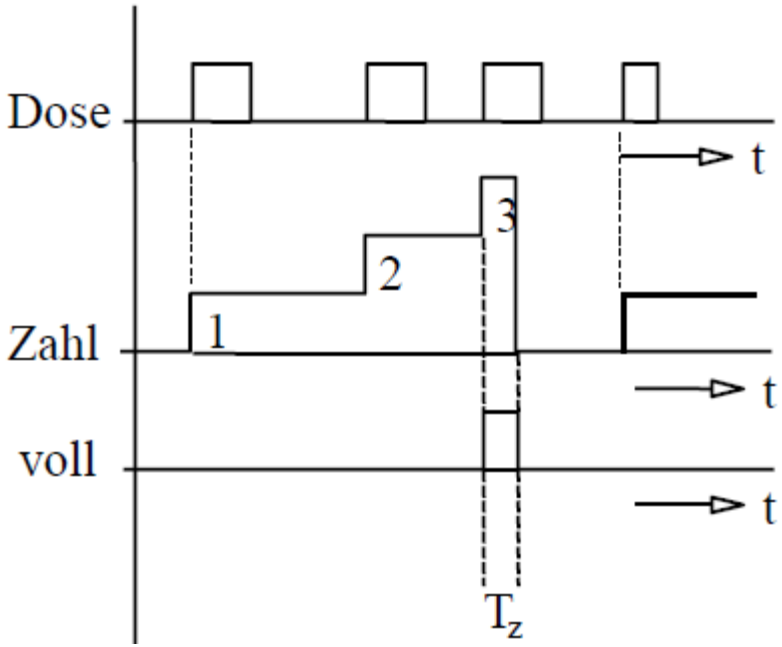


Beispiel Aufwärtszähler: Dosenzähler

Eine Lichtschranke zählt Dosendurchläufe (Variable Dose). Nach jeder dritten Dose soll eine Meldung „Voll“ (Variable voll) erfolgen.

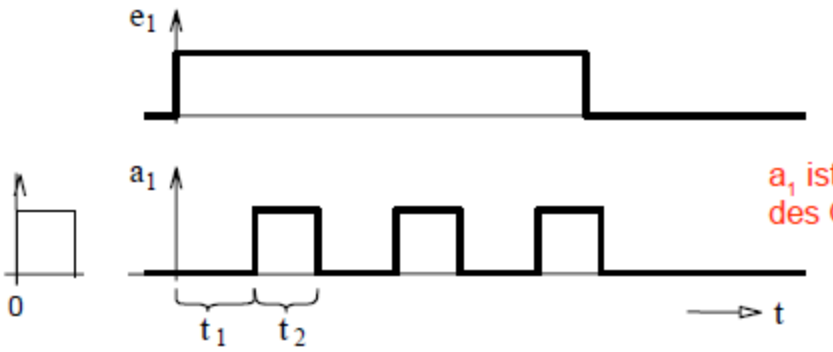



```
VAR    Dose,voll:BOOL;
       Zahl:INT;
END_VAR
```



Beispiel: Blinklicht

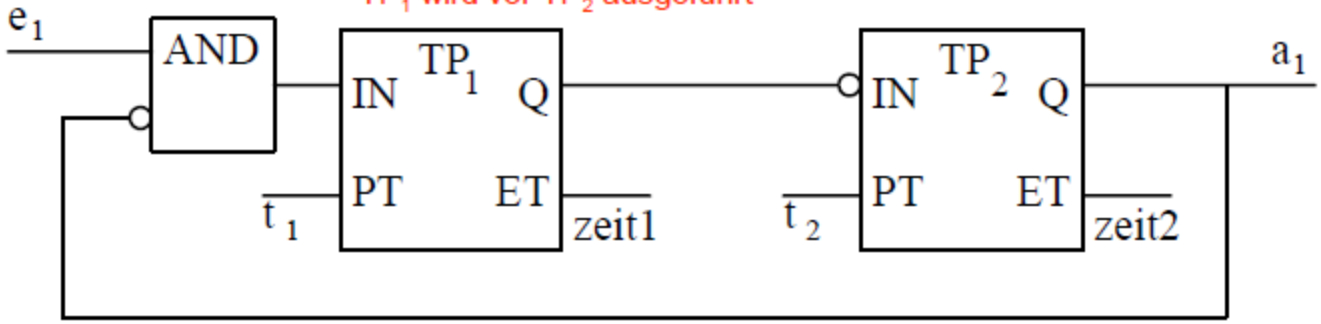
Blinklichtsteuerung aus elementaren Bausteinen: Wenn das Eingangssignal e_1 auf True wechselt (Flanke) soll das Blinklicht im Wechsel für die Zeitdauer t_1 ausgeschaltet und für die Dauer t_2 eingeschaltet werden. Wenn e_1 wieder auf False steht, soll das Blinklicht wieder ausgeschaltet werden.



a_1 ist beim Einschalten für die Dauer t_2 des Gesamtsystems einmalig 1!

```
VAR zeit1,zeit2:TIME;
END_VAR
```

TP_1 wird vor TP_2 ausgeführt



Nicht TP mit Ton oder Toff