Binäre Verknüpfungen

Inhalt

Grundlage der Binären Verknüpfungen

Grundlage der binären Verknüpfungen ist das Duales Zahlensystem, das für EDV-Systeme allgemein übliche Binäre Zahlensystem, bestehend aus „0" und „1", als sog. Stellenwertsystem auf der Basis von 2. Bedeutet: „0" ist der nichtaktive, der Ruhezustand, „1" der aktive Zustand der auslösende. Dies gilt bei => Positiver Logik.

Diese Vorgabe aus der positiven Logik, „1" = Aktiv, lässt sich bei elektronischen Schaltungen überwiegend einsetzen, es sei denn… stellen Sie sich einen entfernten Schalt-Kontakt vor (ein NC, **N**ormally **C**losed, Öffner), der Anlagen definiert aus(!)schalten soll. Dabei fliesst i.d.R. im Ruhezustand (Anlage ein) Strom über die Leitungen zum Kontakt und zurück. Öffnet man den Kontakt, was einer Stromunterbrechung gleichkommt, wird ein Signal ausgelöst, dass die Anlage abschaltet. Das gleiche passiert allerdings auch, wenn die Leitung oder der Kontakt, oder die Verbindungsstelle… unterbrochen wird, oder oxidiert etc. Dann würde jedesmal die Anlage abgeschaltet (Sicherheitsfunktion). Würde man dies so realisieren, dass in „Ruhe" ein „0" – Signal anliegt und die Ausschaltfunktion löst ein „1" Signal aus, so könnte man im Leitungsfehlerfall (Leitung unterbrochen etc.) die Anlage nicht abschalten – das wäre gefährlich! Daher werden Einschaltsignale mit einem „1" Signal ausgelöst, Ruhe „0" und Ausschaltsignale mit einer „0" und Ruhe „1". Am Eingang der Logik, z.B. einer SPS (**S**peicher**p**rogrammierbare **S**teuerung) muss dieses Signal invertiert werden, damit weiterhin (intern der SPS) mit positiver Logik gearbeitet werden kann. Kann man Anlagen nicht einschalten, wäre es kein Problem (es funktioniert dann eben nicht...), den Fehler könnte man suchen. Diese Vorgehensweise nennt man Aktiven Ausschaltschutz, Leitungssicherung, Kabelbruchsicherheit… und ist absolut sinnvoll!

Aufbau Duales Zahlensystem

Das Duale Zahlensystem, Binärsystem, ist Grundlage sämtlicher digitalen Schaltkreise, mögen Sie auch noch so groß sein, es gibt definiert nur „0" oder „1". Wie bei allen Zahlensystemen ist die Wertigkeit der ersten Spalte, ganz Rechts das LSB, **L**east **S**ignificant **B**it immer die „1", die nächste Wertigkeit - die nächste Spalte „heisst" immer so wie das Zahlensystem heisst, das entspricht der Wertigkeit, daher 2.

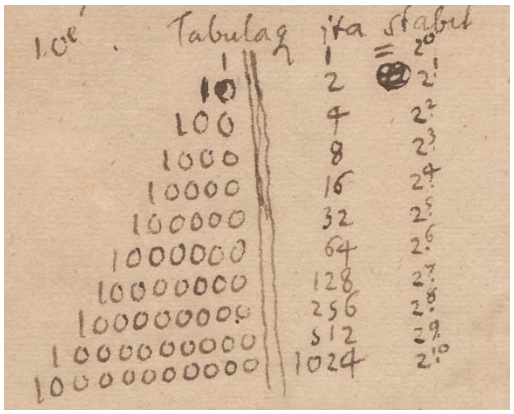
Darauffolgende Spalten lassen sich relativ einfach errechnen: Vorhergehende Spalte * Zahlensystem.

Daher ergeben sich folgende steigende Wertigkeiten, 1, 2, 4, 8, 16, 32, 64, 128, … das höchste, (wertigste) Bit ist das MSB, **M**ost **S**ignificant **B**it. Bedeutet, bei einem 8 Bit System ist die Wertigkeit des LSB = 1, des MSB = 128.

Das Duale Zahlensystem ist damit das wichtigste Zahlensystem – nicht nur innerhalb der Digitaltechnik / Computertechnik. Würde Computertechnik das uns bekannte Zehner Zahlensystem nutzen, so müsste pro Spalte - auf einer Leitung, einem Bit, zwischen zehn verschiedenen Spannungspotentialen exakt unterschieden werden was sehr aufwendig wäre und einen hohen Aufwand erzeugen, was die Referenz betrifft. (Ist 2 V nun 1,9 V „schon", oder erst 1,999 V oder 1,98 V?) Bei nur zwei Zuständen muss nur zwischen Spannung oder keine Spannung unterschieden werden => einfacher und.. erheblich(!) schneller!

Heute bekannt, in den Anfangszeiten der EDV war dies eine Revolution und damals in den von-Neumann „Gesetzen" (der von-Neumann Architektur) festgelegt – nein, nicht an einer Tür angeschlagen…

Aus Wikipedia eines der ersten Darstellungen der „Dualen Theorie", Gottfried Wilhelm Leibnitz, aus 1697:



Als „bessere" Tabelle, eine ganz einfache Folgetabelle aufsteigender Dualzahlen, ebenfalls aus Wikipedia:

Dezimalzahlen 0 bis 15 im Dualsystem	
Wertigkeit:	**8 4 2 1**
Null:	0 0 0 0
Eins:	0 0 0 1
Zwei:	0 0 1 0
Drei:	0 0 1 1
Vier:	0 1 0 0
Fünf:	0 1 0 1
Sechs:	0 1 1 0
Sieben:	0 1 1 1
Acht:	1 0 0 0
Neun:	1 0 0 1
Zehn:	1 0 1 0
Elf:	1 0 1 1
Zwölf:	1 1 0 0
Dreizehn:	1 1 0 1
Vierzehn:	1 1 1 0
Fünfzehn:	1 1 1 1

Aus dieser Ansicht folgt auch manchmal die Benennung „8-4-2-1" – Tabelle. Pro Spalte wird nur ein Wert benutzt, das bedeutet, heutige (Jahr 2020) 64-Bit Systeme besitzen 64 Spalten. Ein Ein- und Ausgabesystem, das mit einer CPU im 128 Bit-Modus kommuniziert, hat daher 128 Bit, Leitungen, Pins…, für Ein- UND 128 Bit, Leitungen, Pins… für Ausgangssignale im Parallelen Modus. Klar, damit sind schon mal 256 Pins des CPU Sockels belegt. In der oben stehenden „kleinen" Tabelle ist allerdings nur bis zur Zahl (Dezimalzahl) 15 aufgelistet. Nebenbei… $15_{Dezimal} = 1111_{Dual} = F_{Hexadezimal}$.

Binäre Verknüpfungen und Abfragen

Wenn binäre Signalzustände (0 oder 1) von Variablen über eine Funktion miteinander verbunden werden, so spricht man von Verknüpfungen.

Aus der Negation (NICHT) und den beiden Grundverknüpfungen UND bzw. ODER lassen sich sämtliche Verknüpfungen realisieren, auch wenn sie noch so kompliziert sind. Eine Erleichterung beim Programmieren einer SPS stellt die Verknüpfung XOR (E**x**clusiv-**O**DE**R**, besser **OR**) dar, die in den SPS bereits als fertige Verknüpfung vorhanden ist. Die Verknüpfungen UND, ODER usw. werden als Operatoren bezeichnet. Diese Operatoren verknüpfen die Operanden (Eingänge, Merker, Zähler oder Zeiten) miteinander.

Variablennegation

Durch Negation (NOT, NICHT) wird der Signalwert einer Variablen invertiert (0 => 1, 1 => 0).
Eine Funktionstabelle ist die Tabelle der Dualzahlen (vgl. oben) auf sämtliche Eingänge bezogen, hier die Negation, mit einem Eingang (!) der nur „0" oder „1" aufweisen kann. Der Ausgang ist demzufolge invertiert. (Eingang = E, Ausgang A, die Bezeichnungen sind jedoch frei wählbar)

E	A
0	1
1	0

Mit Hilfe einer Funktionstabelle wird damit der Zusammenhang zwischen der Eingangsvariablen E und der Ausgangsvariablen A übersichtlich dargestellt. Und, es werden mit Sicherheit alle möglichen Eingangsvariablen aufgelistet.

UND, AND

Haben alle (!) Eingangsvariablen einer UND-Verknüpfung den Signalwert "1", so hat die Ausgangsvariable den Signalwert "1".

Funktionstabelle:

E2	E1	A1
0	0	0
0	1	0
1	0	0
1	1	1

Die ODER-Verknüpfung, Disjunktion (OR, ODER)

Hat eine (oder mehr) der Eingangsvariablen einer ODER-Verknüpfung den Signalwert "1", so hat die Ausgangsvariable den Signalwert "1".

Funktionstabelle:

E2	E1	A1
0	0	0
0	1	1
1	0	1
1	1	1

Die Exclusiv-ODER-Verknüpfung (XOR)

Der Ausgangssignalwert einer Exclusiv-ODER-Verknüpfung, mit zwei Eingangsvariablen, hat dann den Signalwert "1", wenn die beiden Eingangsvariablen unterschiedliche Signalwerte haben.

Funktionstabelle:

E2	E1	A1
0	0	0
0	1	1
1	0	1
1	1	0

Würde der Ausgang invertiert, so ergibt sich ein positiver Ausgang, eine „1" dann, wenn beide Eingänge den gleichen Wert haben, „0" oder „1".

Die Negation einer Verknüpfung

Durch Negation wird der Signalwert einer Variablen invertiert. UND-Verknüpfungen und ODER-Verknüpfungen werden durch die Operation NOT negiert. Eine UND-Verknüpfung bezeichnet man dann als NAND-Verknüpfung, eine ODER-Verknüpfung als NOR-Verknüpfung. Interessanterweise lassen sich alle logischen Funktionen mit NAND's realisieren - die nur aus ganz wenigen Halbleiterbausteinen aufgebaut werden. Im Idealfall kann man mit einem Transistor eine NAND Funktion realisieren. Für ODER, UND.. werden 4 Transistoren benötigt. Damit ergibt sich nicht nur eine Einsparung was die Anzahl der Schaltglieder umfasst, sondern auch die Komplexität der Anordnung unterschiedliche Bausteine fertigen zu müssen, entfällt. Lieber eine Variante, auch in höherer Stückzahl, als fünf verschiedene.

NAND-Verknüpfung

Funktionstabelle einer NAND-Verknüpfung

E2	E1	A1
0	0	1
0	1	1
1	0	1
1	1	0

NOR-Verknüpfung

Funktionstabelle einer NOR-Verknüpfung

E2	E1	A1
0	0	1
0	1	0
1	0	0
1	1	0

Negierte Exclusiv-ODER-Verknüpfung (XOR)

Funktionstabelle einer negierten Exclusiv-ODER-Verknüpfung (XOR) (oben bereits angesprochen…)

E2	E1	A1
0	0	1
0	1	0
1	0	0
1	1	1

Das Verknüpfungsergebnis VKE

Der Signalzustand "0" oder "1" des Ausgangs einer binären Grundoperation / Verknüpfung wird in einem Register in der CPU abgelegt, ganz nah an der Recheneinheit, welches man als Verknüpfungsergebnis (VKE) bezeichnet. Das Verknüpfungsergebnis wird verändert durch Abfrageanweisungen (z.B. "NAND", " NOR", "XOR" usw.)

Abgefragt wird das Verknüpfungsergebnis durch die Zuweisung "=". Die erste Operation nach einer Zuweisung bezeichnet man als Erstabfrage. Die besondere Bedeutung einer Erstabfrage besteht darin, dass die CPU das Abfrageergebnis dieser Anweisung direkt als VKE übernimmt. Das vorherige Verknüpfungsergebnis geht dabei verloren. Ein neues Netzwerk beginnt stets mit einer Erstabfrage.

Daher - nochmal zurück zu von-Neumann die Reihenfolge der Bearbeitung einer Addition:

1. Erster Operand holen (im Register ablegen, Operand, Hex 8B 45 F8)
2. Zweiter Operand holen (im Register ablegen, Accu, Hex 8B 55 FC)
3. Befehl holen (Hexadezimalcode 01 D0) decodieren (= übersetzen) = Add

Ergebnis fertig! Dies ist auch zugleich der Flaschenhals der von-Neumann Architektur, diese Reihenfolge muss zwingend so eingehalten werden, wurde festgelegt und jeder hält sich dran… ,der einen IBM Kompatiblen Computer herstellt und der nach handelsüblichen Programmen funktionieren soll. (ok, wer macht das - Zeitrechnung 2020 nicht?)

Das bedeutet auch, EDV, Computer, Handy (..) etc. alles, was irgendwie mit Computer zu tun hat… Computer können nur addieren und eben nur zwei (Binär)Zahlen. Alles muss auf dieses Problem

zurück geführt werden, auch.. 25345632967 ^{666} * 9808209387654 ^{889}. Nur eben schnell!
Bzw. Computer berechnen das Ergebnis des o.g. Problems „fast" so schnell wie 3 * 9 (dabei können wir Menschen noch mithalten).

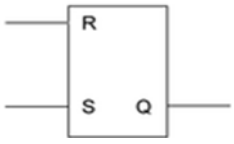
Mithilfe der Befehle "CLR" und "SET" (nur in der Anweisungsliste) kann das Verknüpfungsergebnis ohne Operand fest auf Signalzustand "0" oder "1" gesetzt werden.

Zusammengesetzte binäre Grundverknüpfungen

In Steuerungsprogrammen (z.B. SPS) kommen nicht nur die reinen Elemente NICHT, UND, ODER, NAND, NOR und XOR vor. In vielen Fällen setzt sich eine logische Funktion aus der Kombination mehrerer unterschiedlicher binärer Grundverknüpfungen zusammen. Die beiden Grundstrukturen "UND-vor-ODER" und "ODER-vor-UND" kommen immer wieder vor. (Erstellen Sie hierzu einmal die Wahrheitstabelle, einmal statisch, wie oben und einmal dynamisch, also, springen Sie durcheinander)

UND-vor-ODER Verknüpfung

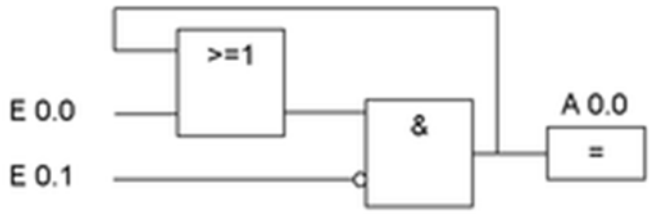
Beispiel einer UND-vor-ODER Verknüpfung, einfachere Darstellung des obigem… :



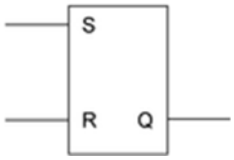
RS-Glied

Die Verknüpfungsergebnisse der UND-Verknüpfung werden zusammen mit dem Eingang E 0.1 ODER-verknüpft. Die Signalzustände an den Eingängen E 0.0 und E 0.1 (das wäre die SPS-Schreibweise) ergeben hier die Funktion eines RS-Gatters, auch setzdominantes FlipFlop genannt.

Folgende Seite, das Digitalschaltbild dazu.

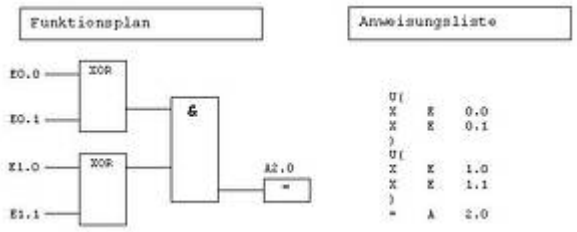


ODER-vor-UND Verknüpfung



SR-Glied

Die Verknüpfungsergebnisse der UND-Verknüpfung werden zusammen mit dem Eingang E 0.1 ODER-verknüpft. Die Signalzustände an den Eingängen E 0.0 und E 0.1 ergeben hier die Funktion eines SR-Gatters, auch rücksetzdominantes FlipFlop genannt.

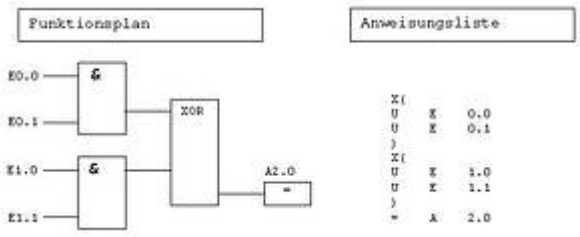


Oben haben wir nun die Symbolik der Digitaltechnik gesehen, den **Fu**nktions**p**lan FUP, eine SPS „versteht" allerdings auch **A**n**w**eisungs**l**iste (Befehlsliste), AWL und KOP, **Ko**ntakt**p**lan.

Exclusiv-ODER vor einer UND-Verknüpfung

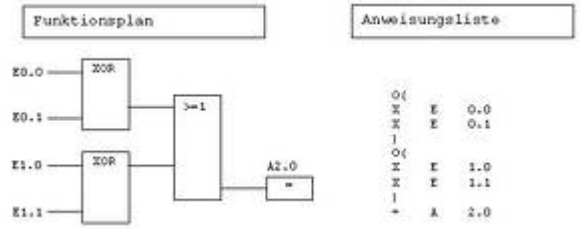
Verknüpft man im Funktionsplan FUP Exclusiv-Oder-Verknüpfungen (XOR-Verknüpfungen) durch eine UND-Verknüpfung, gilt für die Anweisungsliste AWL, dass die Exclusiv-ODER-Verknüpfungen (eigentlich) in Klammern stehen müssten.

Folgende Seite, die Logik dazu.



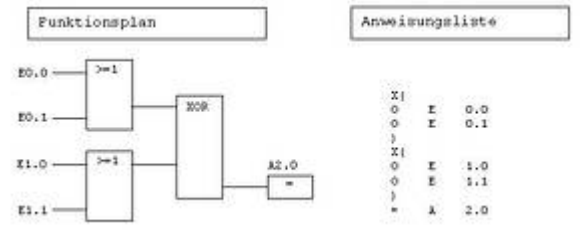
UND-Verknüpfungen vor Exclusiv-ODER

Verknüpft man im Funktionsplan FUP UND-Verknüpfungen durch eine Exclusiv-ODER-Verknüpfung (XOR-Verknüpfungen), gilt für die Anweisungsliste AWL, dass die UND-Verknüpfungen in Klammern stehen müssen.



Exclusiv-ODER vor ODER-Verknüpfung

Verknüpft man im Funktionsplan FUP Exclusiv-ODER-Verknüpfungen (XOR-Verknüpfungen) durch eine ODER-Verknüpfung, gilt für die Anweisungsliste AWL, dass die Exclusiv-ODER-Verknüpfungen in Klammern stehen müssen.



ODER vor Exclusiv-ODER-Verknüpfung

Verknüpft man im Funktionsplan FUP ODER-Verknüpfungen durch eine Exclusiv-ODER-Verknüpfung (XOR-Verknüpfungen), gilt für die Anweisungsliste AWL, dass die ODER-Verknüpfungen in Klammern stehen müssen.

Verwendung von Merkern

In der Relaistechnik gibt es Selbsthaltungen, in der Digitaltechnik gibt es ein RS-Glied, in der SPS Technik „heisst" das gleiche eben Merker. In der SPS-Programmiernorm IEC 6113-3 sind Merker als Operanden benannt - die einen festen Speicherplatz haben. In der Praxis wird manchmal auf die

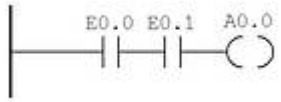
Verwendung von Merkern (nach Möglichkeit) verzichtet. Begründung: Werden bei der Programmierung eines umfangreicheren Anwenderprogrammes Merker verwendet, führt dies dazu, dass die mit Merkern programmierten Bausteine (FB, FC) nicht mehr bibliotheksfähig sind. Merker haben in einem Anwenderprogramm bausteinübergreifend Gültigkeit. Das ist manchmal schwierig handhabbar.

Merkern werden logische Signalzustände zugewiesen, die in festen Speicheradressen im Automatisierungssystem hinterlegt werden. Werden Merker doppelt verwendet, können Programmierfehler entstehen, deren Behebung sehr zeitaufwändig sein können. Deshalb verzichtet man bei der Programmierung von umfangreicheren Anwenderprogrammen auf die Verwendung von Merkern. Fast niemand programmiert allerdings 2020 in Maschinensprache, viele verwenden Hochsprachen – die dann allerdings wieder in Maschinensprache hinunter-übersetzt werden müssen, die ALU **A**rithmetisch-**L**ogische Einheit (**U**nit) versteht es sonst nicht immer. Dies benötigt Zeit und Platz! Schreiben Sie also ein Virusprogramm oder einen Wurm…, verwenden Sie Maschinensprache, das funktioniert!

Die dadurch bibliotheksfähig bleibenden Codebausteine können beliebig oft verwendet werden, da die in ihnen verwendeten Variablen frei zuweisbar sind. Die in Merkern abgelegten Signalzustände haben Gültigkeit, auch außerhalb des Bausteins in dem sie verwendet werden. Es gibt auch ergänzend sog. remanente Merker die bei Spannungsausfall die in ihnen gespeicherten Ergebnisse nicht verlieren. Nichtremanente Merker haben nach einem Neustart oder einem erneuten Anlaufen der SPS den Signalzustand "0". ABER.. es soll ja kein SPS Seminar werden.

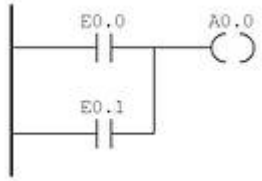
Abschließend (ergänzend) etwas SPS- Technik (KOP und FUP)

UND

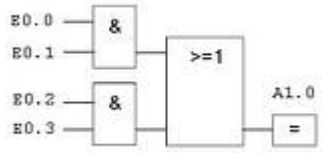


Kontaktplan, fast selbsterklärend, beide Kontakte E 0.0 und E 0.1 müssen geschlossen werden, damit A 0.0 einen Ausgang, eine „1" zeigt.

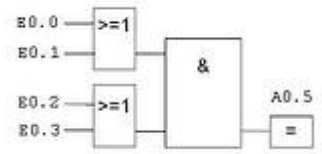
ODER



Die UND vor ODER-Verknüpfung in FUP



Die ODER vor UND-Verknüpfung in FUP



Operation mit dem Bitoperanden U - die UND-Verknüpfung in AWL

U E 0.1
U E 0.2
= A 2.3

Operation mit dem Bitoperanden O - die ODER-Verknüpfung in AWL

O E 0.1
O E 0.2
= A 2.3

UND-Verknüpfungen vor einer ODER-Verknüpfung in AWL

U E 0.1
U E 0.2
O
U E 0.5
U E 0.6
= A 2.3