PHILIPS

Computer Lehrbaukasten CL 1601/CL 1650

und Zusatzkästen CL 1602 CL 1603 CL 1604

A n l e i t u n g s b u c h

Computer-Lehrbaukästen	CL 1601
	CL 1650
und Zusatzkästen	CL 1602
	CL 1603
	CL 1604

Herausgegeben von der Deutschen Philips GmbH., Abt. Technische Spielwaren
2 Hamburg 1, Mönckebergstraße 7

1974

V O R W O R T

Computer in der Welt von heute -

und der PHILIPS Computer-Lehrbaukasten CL 1601

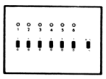
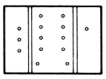
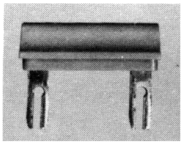
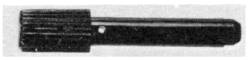
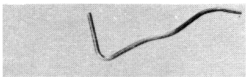
Seit Jahrtausenden ist der Mensch bestrebt, sich körperliche Arbeit durch Technik zu erleichtern. Der große Durchbruch geschah zu Beginn des vorigen Jahrhunderts: Die industrielle Revolution - möglich durch die Erfindung der Dampfmaschine einige Jahrzehnte zuvor. Heute erleben wir die zweite industrielle Revolution: Computer erweitern unsere geistige Leistungsfähigkeit in einem Ausmaß, das man vor gut 3o Jahren, als der erste programmgesteuerte Computer der Welt seine Arbeit aufnahm, noch für utopische Phantasterei gehalten hätte.

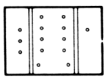
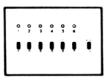
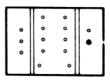
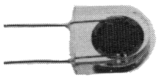
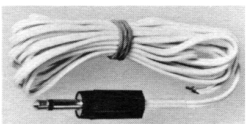
Computer führen in wenigen Sekunden wissenschaftliche Berechnungen durch, für die ein ganzes Team von Wissenschaftlern Monate oder Jahre brauchen würde -
Computer steuern und regeln vollautomatisch komplizierte Fabrikationsprozesse -
Computer untersuchen und überwachen Patienten -
Computer geben Unterricht -
Computer berechnen Satellitenbahnen -
Computer steuern Raketen -
Computer regeln Atomreaktoren -
Computer konstruieren Flugzeuge -
Auch im Alltag des Wirtschaftslebens spielen Computer von Tag zu Tag eine wichtigere Rolle. Von der Lohnabrechnung bis zur Produktionsplanung, von der Einwohnerregistrierung bis zur Steuerberechnung -
Computer rationalisieren die Arbeit und entlasten den Menschen.

Computer - Elektronengehirne - Elektronische Datenverarbeitungsanlagen - oder wie auch immer diese Wunderwerke von Präzision und Schnelligkeit genannt werden, sind den meisten Menschen fremd, unverständlich, ja sogar unheimlich.
Dieser Computer-Lehrbaukasten soll Ihnen helfen, in diese komplizierte Materie einzudringen.
Für alle Grundkästen wird im weiteren Text oft die Abkürzung CL 1601 gebraucht.

I n h a l t s v e r z e i c h n i s

Die Schaltungen, für die eine Eingabeeinheit und zwei Logik-Bausteine benötigt werden, können auch mit dem CL 1650 gebaut werden.

Teil und Symbol	Nr.	Bezeichnung	Inhalt CL 1601	/ 1650
	349.6502	Eingabeeinheit	1	1
	349.6501	Logik-Baustein	5	2
	349.1006	Elektrolyt-Kondensator 220 µF	1	-
	349.6019	Stromversorgungs-stecker	12	4
	349.6020	Stecker	60	30
	349.6021	Kontaktbügel für Stecker	60	30
	349.1017	Isolierter Draht	4 m	4 m

Teil und Symbol	Nr.	Bezeichnung	Inhalt CL 1602	/ 03	/ 04
	349.6501	Logik-Baustein	2	1	1
	349.6502	Eingabeeinheit	-	1	-
	349.6511	Relais-Baustein (Kennfarbe gelb)	-	-	1
	349.1006	Elektrolyt-Kondensator 470 µF (oder 640 µF)	1	1	-
		2200 µF	1	1	-
	349.1010	LDR	1	1	-
	349.6019	Stromversorgungsstecker	4	4	4
	349.6020	Stecker	30	30	30
	349.6021	Kontaktbügel	30	30	30
	349.1017	Isolierter Draht	5 m	5 m	5 m
	349.6026	Kabel mit Klinkenstecker	-	-	1

1. Allgemeine Bauanleitung

1.1. Eingabeeinheit

Die Eingabeeinheit nach Abbildung 1 übernimmt zwei wichtige Funktionen:

Alle Logik-Bausteine werden durch das eingebaute Stromversorgungsteil mit der notwendigen Betriebsspannung versorgt.

Über 6 Schalter lassen sich bei entsprechender Verdrahtung Informationen auf die Eingänge der Logik-Bausteine geben.

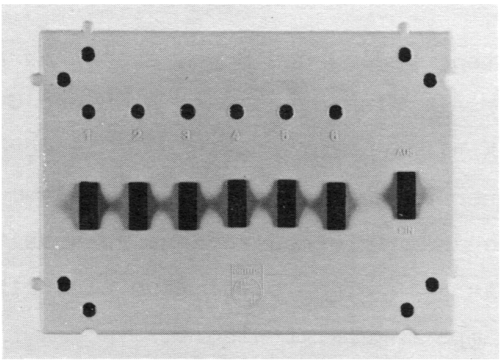


Abb. 1

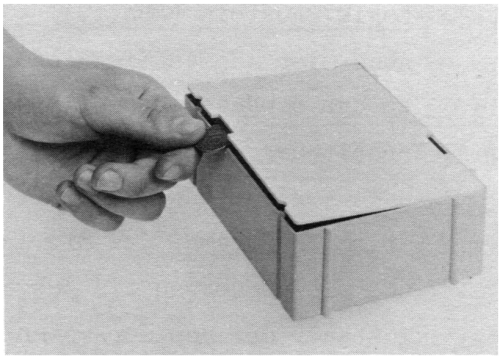


Abb. 2

Damit die Eingabeeinheit diese beiden Aufgaben erfüllen kann, müssen Sie zunächst den Batteriehalter im Gehäuseinneren mit 4 Stück 1,5 Volt Trokkenbatterien (z.B. Philips Babyzellen, Typ R 14 TR) versehen. Entfernen Sie deshalb zunächst mit einer Münze die Bodenplatte vom Gehäuse (Abb.2) und setzen Sie dann gemäß Abb. 3 die Batterien in den Halter, wobei Sie unbedingt auf die im Halter angegebene Polung achten müssen. Danach verschließen Sie die Eingabeeinheit wieder mit der Bodenplatte.

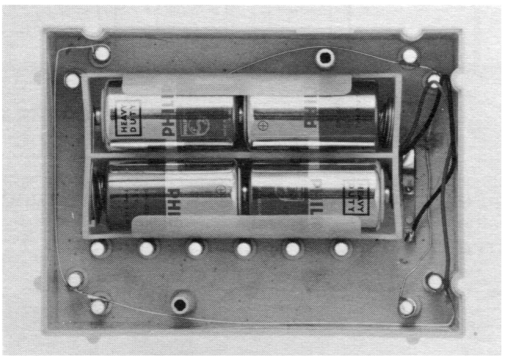


Abb. 3

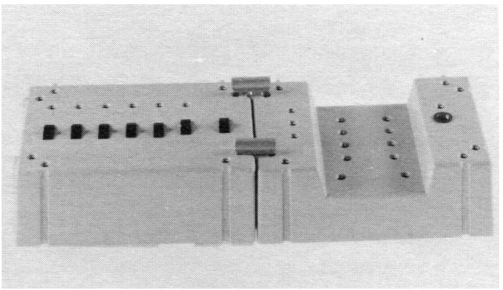


Abb. 4

Auf der Oberseite befindet sich ganz rechts der EIN-AUS-Schalter. In der EIN-Stellung verbindet er den Batteriehalter mit den Stromversorgungsbuchsen an der Gehäusekante. Bei Verwendung der Stromversorgungsstecker

als Verbindungselemente zwischen Eingabeeinheit und den Logik-Bausteinen ist durch die spezielle Anordnung der Buchsen, Nuten und Federn (Abb. 4) eine falsche Polung der Versorgungsspannung ausgeschlossen.

Neben dem EIN-AUS-Schalter befinden sich 6 weitere schwarze Schieber, die Eingabeschalter genannt werden. Sie sind den mit 1 bis 6 bezifferten Buchsen zugeordnet. Je nach Schalterstellung können über Drahtleitungen elektrische Signale (Informationen) in die Logik-Bausteine eingegeben werden.

1.2. Logik-Baustein (Abb. 5)

Auch hier können Sie wieder die Nuten, Federn und Stromversorgungsbuchsen an den Gehäusekanten erkennen. Über die Eingangsbuchsen A, B und C (Abb.6) erhält der Logik-Baustein alle elektrischen Informationen. Sie werden je nach Verdrahtung des tiefer gelegenen Programmierfeldes nach einem bestimmten Schema, das Sie später noch kennenlernen werden, verarbeitet, und sie versetzen die Ausgangsbuchse F in einen bestimmten elektrischen Zustand, der von der roten Signallampe angezeigt wird.

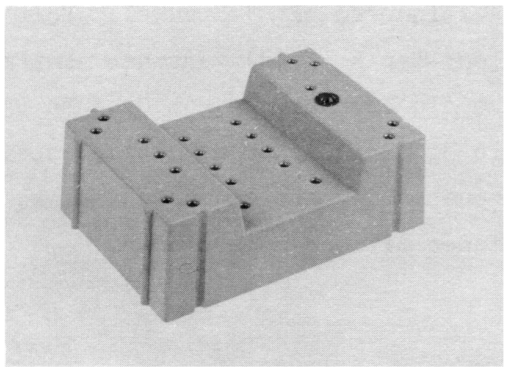


Abb. 5

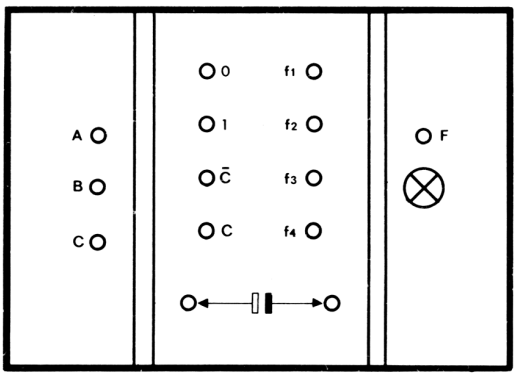


Abb. 6

Die linken 4 Buchsen im Programmierfeld (sie werden mit 0, 1, $\bar{C}$ und C bezeichnet) können beliebig mit den gegenüberliegenden 4 rechten Programmiereingängen f1, f2, f3 und f4 verbunden werden.

Bei einigen Anwendungsbeispielen ist es notwendig, die auf die Eingänge A, B und C gegebene Information gegenüber dem Ausgangssignal F zeitlich zu verzögern. Das kann durch einen Elektrolytkondensator erreicht werden, der in diesem Fall an die beiden unteren Buchsen im Programmierfeld anzuschließen ist. Bitte, achten Sie auf die in allen Schaltbildern und an allen Logik-Bausteinen angegebene Polung.

1.3. Elektrolytkondensator (Abb. 7)

Mit diesem Bauelement läßt sich eine Verzögerung vom Eingangs- zum Ausgangssignal erreichen. Allerdings muß der Kondensator in der richtigen Polarität an die betreffenden Buchsen im Programmierfeld des Logik-Bausteins angeschlossen werden (Abb. 8). Der Pluspol ist auf dem Kondensator deutlich mit dem Zeichen + versehen. Dieses Zeichen finden Sie in allen Schaltbildern und im Programmierfeld der Logik-Bausteine wieder.

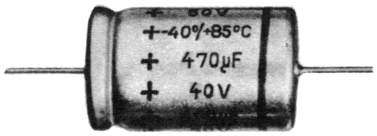


Abb. 7

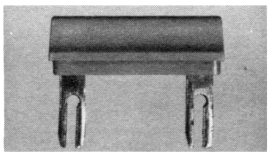


Abb. 9

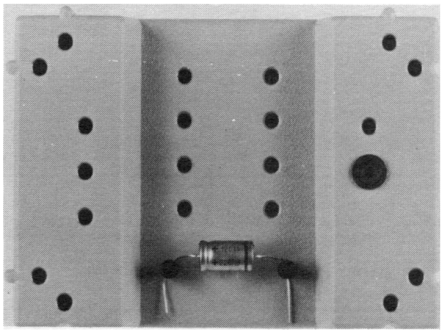


Abb. 8

1.4. Stromversorgungsstecker (Abb. 9)

Sie dienen als Verbindungselement zwischen dem Stromversorgungsteil der Eingabeeinheit und den Stromversorgungsbuchsen der Logik-Bausteine. Schieben Sie Nut und Feder zweier Bausteine wie in Abb. lo angegeben zusammen, und drücken Sie nach Abb. ll die Stromversorgungsstecker in die Buchsen. Falls ein Stecker zu leicht in die Stromversorgungsbuchsen gleitet, müssen Sie die Kontakte etwas aufbiegen. In den Verdrahtungsplänen sind die benötigten Stromversorgungsstecker durch einen breiten schwarzen Balken angedeutet. Schließen Sie nur die für das Spiel benötigte Anzahl von Logik-Bausteinen an die Eingabeeinheit an (Abb. 12).

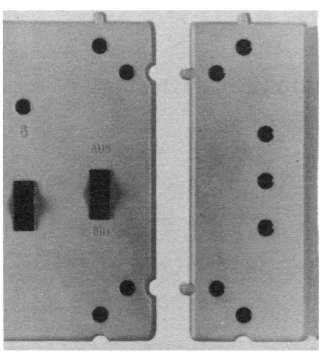


Abb.lo

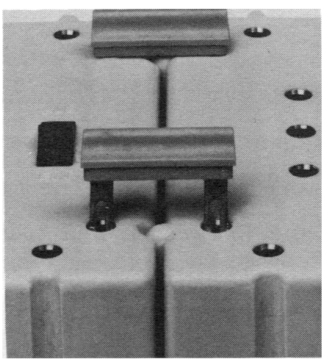


Abb.11

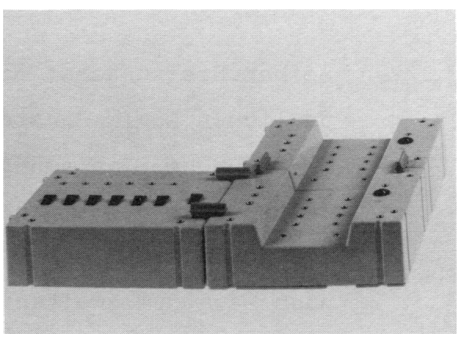


Abb.12

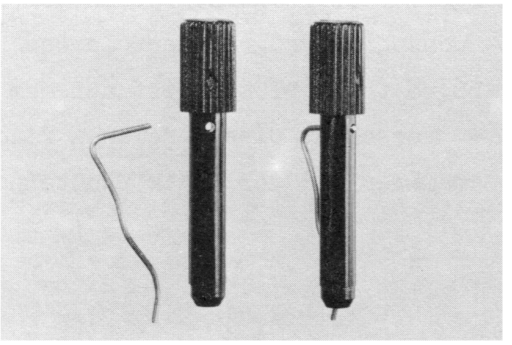


Abb.13

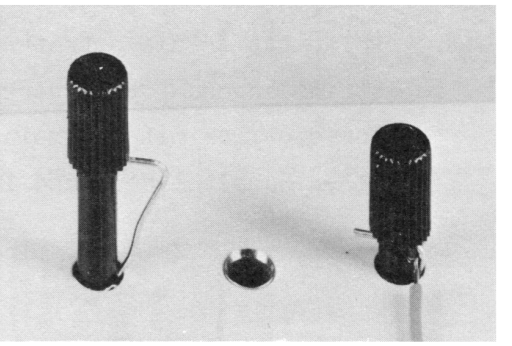


Abb.14

1.5. Stecker und Kontaktbügel (Abb. 13)

Außer bei der Stromversorgung werden alle anderen elektrischen Verbindungen mit Steckern und isolierten Schaltdrähten vorgenommen.

Stecken Sie das untere Ende des Kontaktbügels in den Schlitz des Kunststoffsteckers, und schieben Sie es durch das Loch an der Stirnseite des Steckerschaftes. Das abgewinkelte Drahtende können Sie nun durch das Querloch drücken. Mit diesen Handgriffen haben Sie den Stecker montiert und können ihn gem. Abb. 14 in die zu verbindenden Buchsen einsetzen. Die auf richtige Länge abgeschnittenen und an den Enden abisolierten Drähte können nun zwischen Kontaktbügel und Kunststoffteil gesteckt und eingeklemmt werden.

1.6. Verdrahtungspläne

Für alle Anwendungsbeispiele finden Sie im Anleitungsbuch einen übersichtlichen Verdrahtungsplan, der Ihnen in zwei Abbildungen alle Verbindungsleitungen des Programmierfeldes und die Verdrahtung der Ein- und Ausgangsbuchsen zeigt.

Reihen Sie zunächst nach diesem Plan die Eingabeeinheit mit den erforderlichen Logik-Bausteinen aneinander und verbinden Sie, wie im ersten Verdrahtungsplan angegeben, alle Bausteine mit Stromversorgungssteckern. Jetzt folgt die eigentliche Programmierung der Logik-Bausteine durch die Verdrahtung des Programmierfeldes. Zum Abschluß brauchen Sie nur noch nach dem zweiten Verdrahtungsplan die Eingangsbuchsen mit den Eingabeschaltern bzw. mit den Ausgangsbuchsen F zu verbinden, und der gesamte Aufbau ist einsatzbereit.

1.7. Papierschablonen

Für die wichtigsten logischen Grundfunktionen finden Sie am Schluß des Anleitungsbuches vorgedruckte Schablonen. Mit ihrer Hilfe lassen sich häufig wiederkehrende Schaltungen schneller verdrahten. Da Ein- und Ausgänge sowie das Programmierfeld getrennt verdrahtet werden, sind zwei Schablonen nötig. Schneiden Sie beide Teile aus dem Papierbogen und legen Sie die kleine Schablone auf das Programmierfeld. Mit einem spitzen Bleistift lassen sich jetzt alle gekennzeichneten Stellen leicht durchstoßen. Durch die entstandenen Löcher passen die erforderlichen vormontierten Stecker, und, wie auf der Schablone abgebildet, läßt sich die Verdrahtung ausführen.

Die große Schablone wird in gleicher Weise vorbereitet; sie erhält jedoch zusätzlich ein großes Loch für die rote Anzeigelampe.

1.8. Kontrolle des Logik-Bausteins

Verbinden Sie die Eingabeeinheit und einen Logik-Baustein und stellen Sie die Verdrahtung gem. Abb. 15 her. Achten Sie auf die Polarität des Elektrolytkondensators und kontrollieren Sie alle Verbindungen. Wenn Sie jetzt den EIN-AUS-Schalter der Eingabeeinheit in die Stellung EIN bringen, muß bei einem einwandfreien Logik-Baustein ständig die rote Lampe blinken.

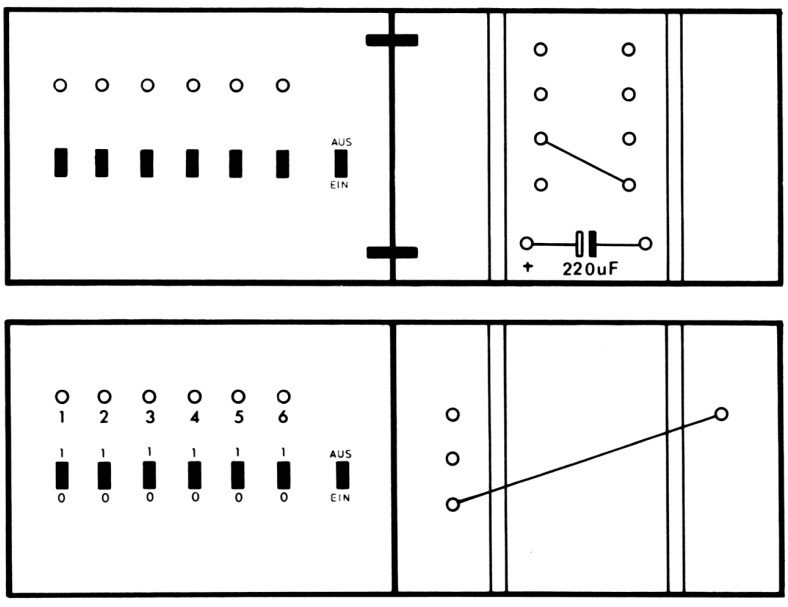
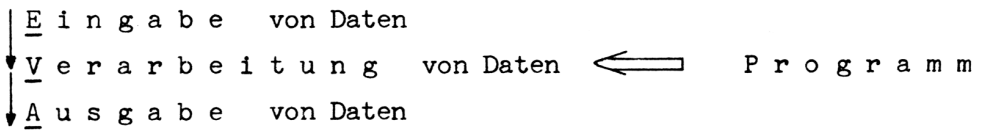


Abb.15

2. Begriffserklärungen

Vor dem ersten praktischen Einsatz des CL 1601 zunächst einige Begriffserklärungen.
Das Wort C o m p u t e r kommt aus dem Englischen. Zu deutsch heißt Computer R e c h n e r. Genaugenommen wird diese Bezeichnung dem Computer nicht gerecht, denn er kann mehr als nur rechnen. Er kann Daten verarbeiten. Und weil er das elektronisch macht, sagt man auch Elektronische Datenverarbeitungsanlage, abgekürzt EDVA. Dementsprechend wird Elektronische Datenverarbeitung kurz EDV genannt. Aber was ist Datenverarbeitung? Und was sind Daten? - Daten sind Informationen. Und Information ist alles das, was uns in irgendeiner Form eine Bedeutung übermittelt, also z.B. gesprochene oder geschriebene Wörter und Sätze, ferner Zahlen, Zeichnungen und Bilder. Einem Computer werden Daten zur Verarbeitung in Form von Zahlen bzw. Ziffern, Buchstaben und Zeichen (wie . , / + % usw.) eingegeben. Nach der Verarbeitung der Daten werden die Ergebnisse wiederum meist in Ziffern, Buchstaben und Zeichen ausgegeben, die zusammengenommen selbstverständlich irgendwelche sinnvollen Aussagen darstellen. Wie das bei großen Computern vor sich geht, werden Sie an anderer Stelle dieser Anleitung noch erfahren. Ähnlich wie bei großen Computern werden auch beim Philips Computer-Lehrbaukasten CL 1601 Daten eingegeben, verarbeitet und als Ergebnisse ausgegeben. Die Verarbeitung der Daten erfolgt aufgrund eines Programms. Auch darin stimmen große Computer und der CL 1601 überein. Veranschaulichen wir uns dieses Grundprinzip jeder Datenverarbeitung anhand einer schematischen Darstellung:



Der Ablauf ist leicht zu merken durch die Anfangsbuchstaben E V A.

Nun noch eine ganz besondere Kuriosität elektronischer Datenverarbeitung: Computer führen sämtliche Daten auf ganze 2 Zeichen zurück, auf die Zeichen 0 und 1. Wahrscheinlich können Sie sich im Moment nicht vorstellen, wie das möglich sein soll: Datenverarbeitung bis zu den kompliziertesten Aufgaben nur mit den beiden Zeichen 0 und 1? Aber es ist tatsächlich so, und Sie werden dieses Geheimnis bald ergründet haben - mit Ihrem CL 1601. Auch er arbeitet mit 0 und 1.
Vorerst einmal genug der Theorie. Jetzt soll Ihr Computer Daten verarbeiten.

3. Logische Grundfunktionen

Erster Einsatz des Philips Computer-Lehrbaukastens CL 1601

Bei den folgenden Versuchen können wir uns mit der Eingabeeinheit (mit Spannungsversorgung) und einem Logik-Baustein begnügen.

3.1. Darf ich nach der Party Auto fahren? - Was sagt der Computer dazu? Eine UND-Funktion

Sie sind auf einer feuchtfröhlichen Party. Ein Freund hat Sie mit seinem Wagen dorthin mitgenommen. Als zu vorgerückter Stunde die Heimfahrt beschlossen wird, ist Ihr Freund reichlich vollgetankt. Glücklicherweise ist er aber auch vernünftig: Er will sich mit Rücksicht auf das Maß seiner Alkoholisierung nicht ans Steuer setzen. Er fragt Sie, ob Sie in der Lage sind, bei der Heimfahrt das Steuer zu übernehmen. Daraufhin fragen Sie sich: 1. Bin ich nüchtern (Alkohol im Blut mit Sicherheit unter o,8 ‰)? 2. Habe ich meinen Führerschein überhaupt dabei? Sehen wir mal, was Ihr Computer dazu sagt. Wann sagt er Ihnen, Sie dürfen Auto fahren bzw. Sie dürfen nicht Auto fahren? Gewiß trauen wir Ihnen zu, die richtige Antwort auch ohne Hilfe des Computers zu finden. Aber darum geht es ja schließlich nicht.

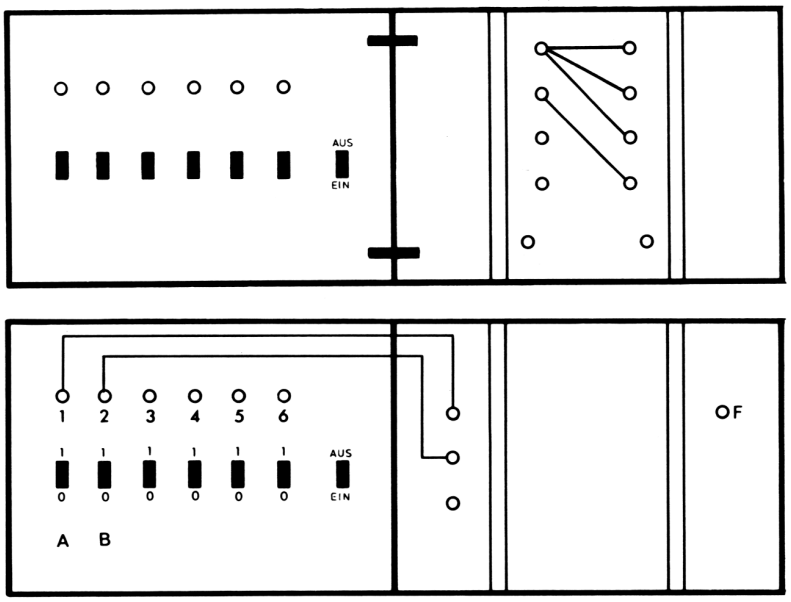


Abb.16

Verbinden Sie einen Logik-Baustein mit der Eingabeeinheit und verbinden Sie die Buchsen 1 und 2 der Eingabeeinheit mit den Buchsen A und B des Computerbausteins. Dann programmieren Sie den Baustein, indem Sie die Buchse 0 des Programmierfeldes mit den Buchsen f1, f2 und f3 des Pro-

grammierfeldes und 1 mit f4 verbinden (siehe Allgemeine Bauanleitung). Zur Durchführung unserer Datenverarbeitungsaufgabe brauchen wir von den Eingabeschaltern die Schalter A und B; sie sollten zu Beginn der Aufgabe auf 0 stehen. Schalten Sie jetzt den EIN-AUS-Schalter ein.

Der Schalter 1 wird der Frage "Bin ich nüchtern?" zugeordnet, der Schalter 2 der Frage "Habe ich meinen Führerschein dabei?". Auf die beiden Fragen gibt es als Antwortmöglichkeiten "JA" bzw. "NEIN". Treffen wir eine weitere Zuordnung: Für NEIN sagen wir 0 (Null), für JA sagen wir 1.

Ähnliche Zuordnungen nehmen wir für die Antwort vor: Für die Frage "Darf ich Auto fahren?" sei das rote Licht zuständig. Müssen wir die Frage mit NEIN beantworten, brennt das Licht nicht, können wir die Frage mit JA beantworten, leuchtet das Licht auf.

Schalten Sie jetzt anhand der folgenden Tabelle einmal alle Möglichkeiten durch, führen Sie also alle Kombinationen der Dateneingabe aus, und schreiben Sie dabei das jeweilige Ergebnis (ebenfalls in Form von 0 bzw. 1) in die Spalte für Datenausgabe:

Dateneingabe		Datenausgabe
Bin ich nüchtern?	Habe ich meinen Führerschein dabei?	Darf ich Auto fahren?
A	B	F
0	0	
0	1	
1	0	
1	1	

In welchen Fällen leuchtet die Lampe nicht auf?
Wann leuchtet sie auf?
Haben Sie für JA = "Lampe an" eine 1 eingetragen und für NEIN = "Lampe aus" eine 0 ?

Sieht Ihr Ergebnis auch so aus?

Dateneingabe		Datenausgabe
Bin ich nüchtern?	Habe ich meinen Führerschein dabei?	Darf ich Auto fahren?
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

In Worten ausgedrückt lautet das Ergebnis unserer Computerbefragung: Wenn ich nüchtern bin und wenn ich meinen Führerschein dabei habe, dann darf ich Auto fahren.
Wir haben es hier mit einer sogenannten UND-Funktion zu tun. UND-Funktion deshalb, weil eine Bedingung und eine zweite Bedingung erfüllt sein müssen, damit ein bestimmtes Ergebnis zustande kommt.
Die dazugehörige Tabelle, deren Ergebniswerte Sie mit Ihrem Computer ermittelt haben, heißt Funktionstabelle.
Wenn wir die eine Bedingung A nennen und die zweite Bedingung B und wenn wir das Ergebnis mit F (Lampe) bezeichnen, läßt sich die Funktionstabelle so darstellen:

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Bevor wir zum nächsten Experiment kommen, wollen wir noch einige weitere Zusammenhänge klären.
Sehen Sie sich das folgende Schaltbild an:

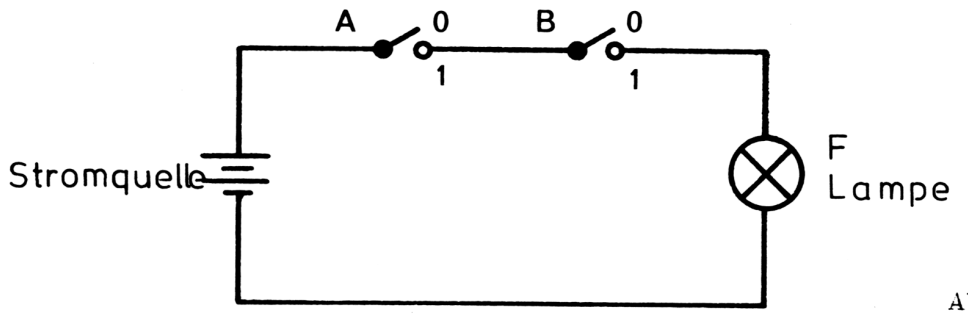


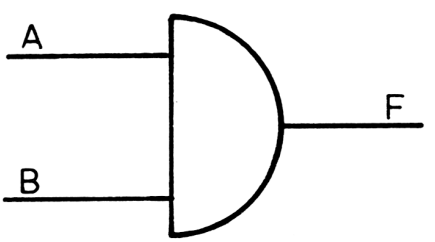
Abb.17

Eine Lampe ist mit einer Stromquelle verbunden. In den Stromkreis sind zwei Schalter eingebaut, und zwar hintereinander. Sie können leicht erkennen, daß die Lampe nur dann brennt, wenn Schalter A und Schalter B auf 1 - also eingeschaltet - sind.
Der Form nach haben wir es hier also ebenfalls mit einer UND-Funktion zu tun.
Damit haben wir eine UND-Funktion

sprachlich ausgedrückt,
in Form einer Funktionstabelle dargestellt und
in einer elektromechanischen Schaltung veranschaulicht.

Eine entsprechende Schaltung haben Sie auch bei Ihrem Computer-Lehrbaukasten verwirklicht. Deshalb konnten Sie auch die Ergebnisse ermitteln, die mit den Zusammenhängen bei der elektromechanischen Schaltung übereinstimmen.
Aber: Bei dem Computer-Lehrbaukasten handelt es sich um eine elektronische Schaltung. Wie die Elektronik den großen Computern zu deren enormen Leistungen verhilft, so bietet die Elektronik im Philips Computer-Lehrbaukasten ganz andere Experimentiermöglichkeiten als ein "elektromechanischer Spielcomputer" sie bieten würde.
Kommen wir aber noch einmal auf die Gemeinsamkeit zwischen der oben gezeigten elektromechanischen Schaltung und der elektronischen Schaltung in Ihrem Lehrcomputer zurück: Beiden Schaltungen gemeinsam ist ihre Logik: Wenn A und wenn B, dann (und nur dann) F. So können wir den Sachverhalt abstrakt ausdrücken.

Die besprochene UND-Funktion ist eine logische Funktion. Außer der UND-Funktion gibt es natürlich noch andere logische Funktionen. Sie werden sie noch kennenlernen.
Wenn eine logische Funktion in Form einer Schaltung (einer mechanischen, einer elektromechanischen oder einer elektronischen Schaltung) verwirklicht ist, spricht man von einer logischen Schaltung. Für logische Schaltungen gibt es bestimmte Symbole. So sieht z.B. das Symbol für die besprochene UND-Schaltung so aus:



Wenn Sie jetzt einmal einen kurzen Blick auf den Schaltplan der Logik-Bausteine Ihres Computer-Lehrbaukastens werfen (Kap. 8), dann sehen sie dieses und ähnliche Symbole. Lassen Sie sich aber nicht dadurch verwirren, daß Sie den Schaltplan noch nicht durchschauen. Zunächst einmal wollen wir weitere logische Funktionen und Schaltungen kennenlernen.

3.2. Internationales Ferienlager

Durch ein weiteres Beispiel können Sie sich mit einer UND-Funktion und ihrer logischen Schaltung vertraut machen.
Schüler bewerben sich um die Teilnahme an einem internationalen Ferienlager. Voraussetzung für die Teilnahme ist, daß jeder mindestens englisch und französisch spricht. Da nicht alle Jugendlichen zwei Fremdsprachen beherrschen, sollen Sie mit Hilfe des Computers entscheiden, wer die Ferien dort verbringen darf.

Versuchen Sie einmal, die vier möglichen Kombinationen, die sich aus diesen beiden Sprachen ergeben, aufzustellen:

englisch	französisch	Ferienlager
A	B	F
nein	nein	nein
ja	ja	nein
		ja

Vielleicht hilft Ihnen beim Ausfüllen der Funktionstabelle die sprachliche Formulierung der UND-Funktion: Wenn der Schüler englisch und französisch spricht, dann darf er am Ferienlager teilnehmen.

Die Funktionstabelle läßt sich einfacher aufstellen, wenn für die Beherrschung der Sprachen (JA) = 1, andernfalls (NEIN) = 0 vereinbart wird. Die Zusage für die Teilnahme (JA) wird ebenfalls durch 1 ausgedrückt, die Ablehnung (NEIN) durch 0.

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Was Sie schon selbst bei der ausführlichen Form der Funktionstabelle herausgefunden haben, drückt auch die Kurzform aus: Nur im vierten Fall der möglichen Kombinationen, also wenn der Schüler englisch und französisch beherrscht, wird eine Zusage erteilt.

Schalten Sie jetzt das Beispiel mit Ihrem Computer durch. Verbinden Sie dazu die Buchse 1 der Eingabeeinheit mit der Buchse A eines Logik-Bausteins und 2 mit B. Der Logik-Baustein wird programmiert, indem Sie die Buchse 0 des Programmierfeldes mit f1, f2 und f3, die Buchse 1 mit f4 verbinden.

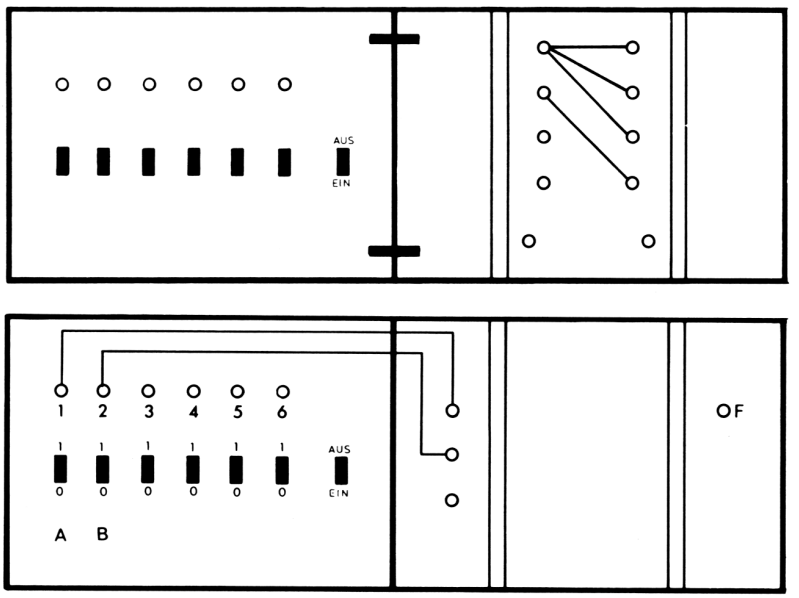


Abb.18

Schalter A	AUS	französisch	NEIN	(0)
Schalter A	EIN	französisch	JA	(1)
Schalter B	AUS	englisch	NEIN	(0)
Schalter B	EIN	englisch	JA	(1)
Lampe	AUS	nimmt nicht teil		(0)
Lampe	EIN	nimmt teil		(1)

Überqueren einer Straße

Nun ein letztes einfaches Beispiel zu einem logischen UND: Als Verkehrsteilnehmer ist Ihnen längst in Fleisch und Blut übergegangen, daß Sie eine Straße nur dann überqueren, wenn von links und von rechts kein Fahrzeug kommt. Überprüfen Sie doch einmal, ob Ihr Computer Ihnen denselben Rat gibt.

Bei diesem Beispiel sollen Sie aber weitgehend selbständig arbeiten, damit Sie prüfen können, ob Sie das Aufstellen der UND-Funktion beherrschen.

Einige Hilfen erhalten Sie allerdings noch:

Straße links frei	JA	(1)
	NEIN	(0)
Straße rechts frei	JA	(1)
	NEIN	(0)

Jetzt fällt es Ihnen bestimmt nicht schwer, die ausführliche Form der Funktionstabelle aufzustellen und die möglichen Kombinationen anzugeben.

Straße links frei	Straße rechts frei	Gehen
A	B	F

Zum Aufstellen der Kurzform der Funktionstabelle muß nur noch festgelegt werden, daß 1 GEHEN und 0 NICHT GEHEN zugeordnet werden soll.

A	B	F

Wenn Sie nach der folgenden Abbildung die Buchse 1 der Eingabeeinheit mit Buchse A und 2 mit B und im Programmierfeld des Logik-Bausteins 0 mit f1, f2, f3 und 1 mit f4 verbinden, können Sie mit Ihrem Computer überprüfen, ob Sie ihre Funktionstabelle richtig aufgestellt haben.

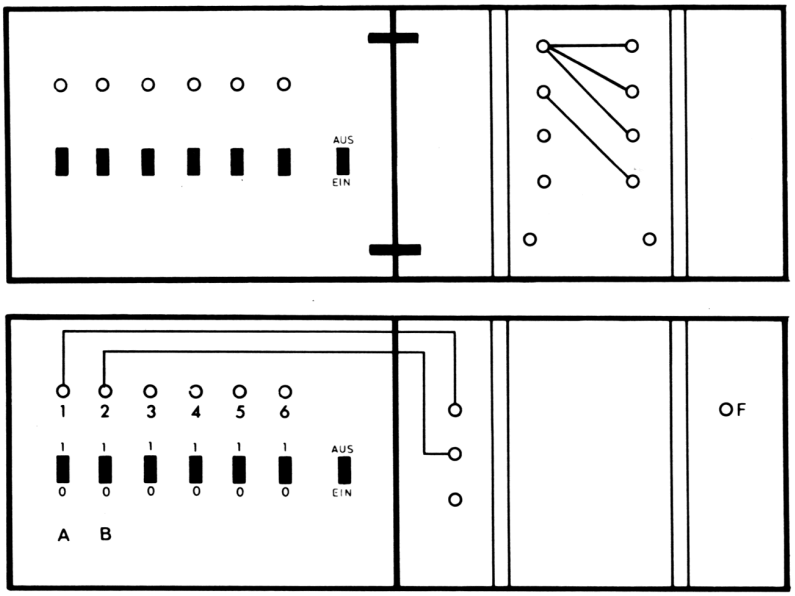


Abb.19

Wenn die Lampe aufleuchtet, gibt der Computer den Weg für die Straßenüberquerung frei.

3.3. Wann ist es hell im Zimmer? - Eine ODER-Funktion

Auch diese recht einfache Frage soll uns der Computer beantworten. Wir gehen dabei von zwei Ausgangsfragen aus:

Brennt im Zimmer eine Lampe?
Ist es draußen taghell?

Wir benötigen für die Beantwortung dieser Frage wiederum die Eingabeeinheit und einen Logik-Baustein. Programmieren Sie, wie auf der Abbildung angegeben.

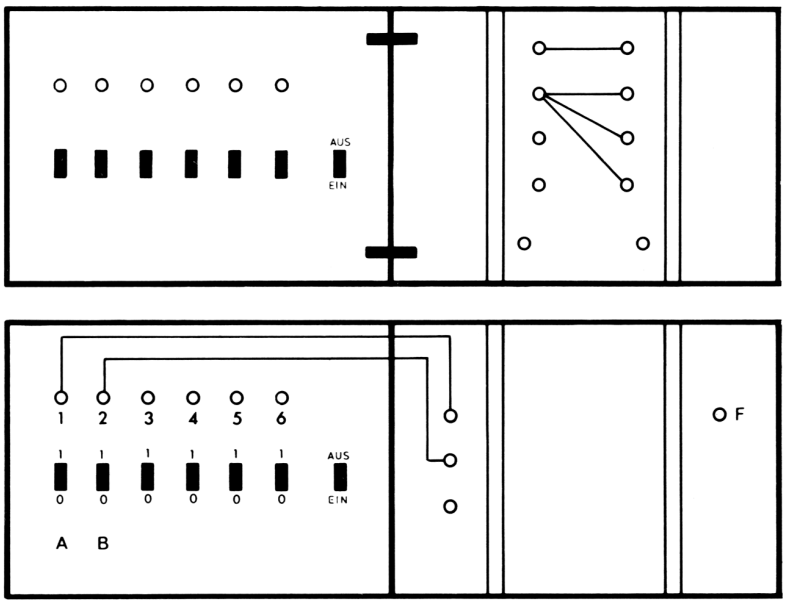


Abb.2o

Der Schalter A entspricht der Frage: "Brennt im Zimmer eine Lampe?" Der Schalter B entspricht der Frage: "Ist es draußen taghell?" Für ein NEIN auf diese Fragen gelte wiederum 0, für ein JA = 1. Was sagt nun der Computer dazu? Wann ist es hell im Zimmer? Tragen Sie die Antworten des Computers in die Tabelle ein.

Dateneingabe		Datenausgabe
Brennt im Zimmer eine Lampe?	Ist es draußen taghell?	Ist es hell im Zimmer?
A	B	F
0	0	
0	1	
1	0	
1	1	

Und dies sind die Werte, die Ihr Computer ermittelt hat und durch die Lampe (0 = NEIN, 1 = JA) angezeigt hat:

Dateneingabe		Datenausgabe
Brennt im Zimmer eine Lampe?	Ist es draußen taghell?	Ist es hell im Zimmer?
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

In Worten ausgedrückt lautet das Ergebnis: Wenn im Zimmer eine Lampe brennt oder wenn es draußen taghell ist, dann ist es hell im Zimmer. Wenn wir uns das Ergebnis der Datenverarbeitung durch den Computer ansehen, müssen wir uns sprachlich noch etwas genauer ausdrücken: Im Zimmer ist es auch dann hell, wenn im Zimmer eine Lampe brennt und wenn es draußen taghell ist.
Es handelt sich hier um eine ODER-Funktion. Genau gesagt, um eine Einschließende ODER-Funktion, d.h. das Ergebnis - im Zimmer ist es hell - kommt zustande, wenn die eine oder die andere bzw. auch wenn die beiden der genannten Bedingungen erfüllt sind.

Wie schon bei der UND-Funktion fassen wir auch hier das Ergebnis in abstrakter Form in einer Funktionstabelle zusammen:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Die ODER-Funktion läßt sich elektromechanisch durch folgendes Schaltbild veranschaulichen:

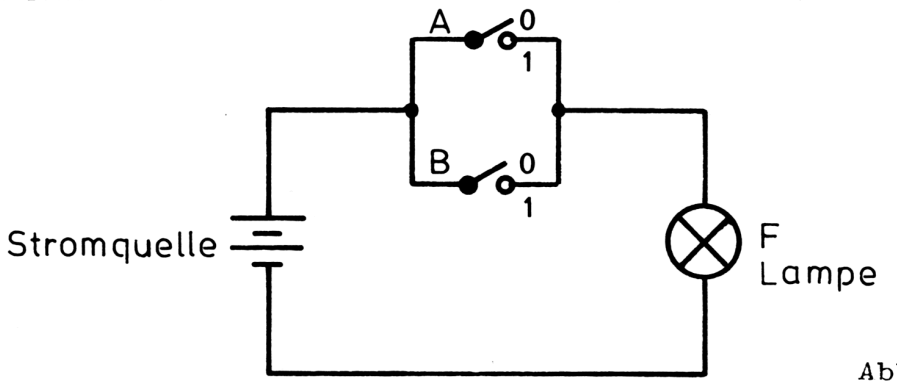
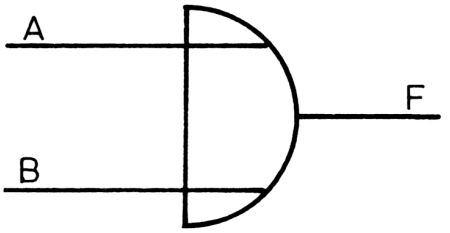


Abb.21

Wie in Abb.21 ist hier eine Lampe mit einer Stromquelle verbunden. In den Stromkreis sind zwei Schalter eingebaut, aber dieses Mal sind die Schalter parallel angeordnet. Sie können aus der Darstellung ersehen, daß die Lampe bereits brennt, wenn nur einer der beiden Schalter A oder B eingeschaltet ist. Natürlich brennt die Lampe auch dann, wenn beide Schalter eingeschaltet sind.
Schließlich gibt es auch für die ODER-Funktion ein Schaltsymbol:



Wir haben also eine zweite logische Schaltung kennengelernt. Bei der UND-Funktion haben wir bei ihrer technischen Verwirklichung von UND-Schaltung gesprochen. Dementsprechend haben wir es mit einer ODER-Schaltung zu tun, wenn eine ODER-Funktion in Form einer Schaltung verwirklicht ist.

3.4. Wann muß ich an einer Ampel halten?

Als Teilnehmer am Straßenverkehr wissen Sie selbstverständlich, daß ROT oder GELB oder ROT und GELB an einer Verkehrsampel zum Halten zwingen. Sie sollen an diesem Beispiel auch nur versuchen, die Funktionstabelle für eine ODER-Funktion aufzustellen.
Sie können wählen, ob Sie zuerst die Tabelle aufstellen und diese dann mit Ihrem Computer überprüfen oder ob der Computer das Aufstellen übernehmen soll.

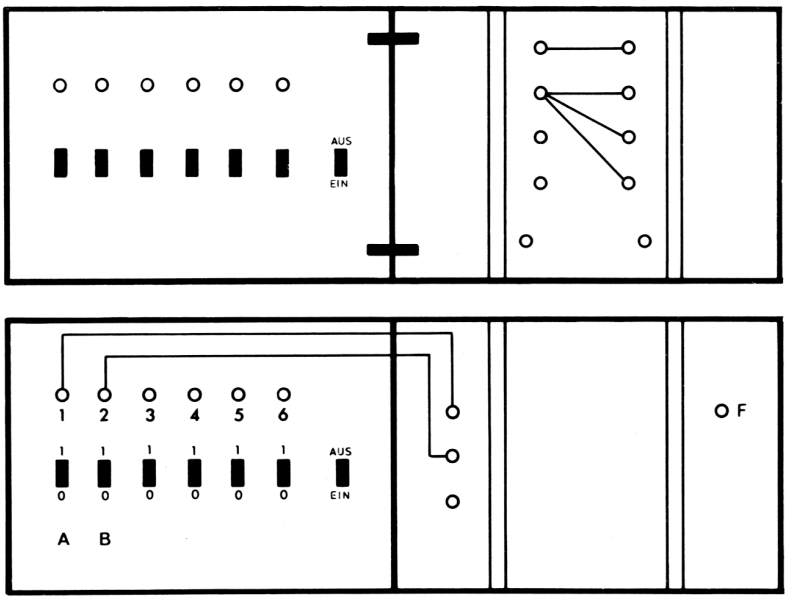


Abb.22

Bevor Sie mit dem Computer arbeiten, muß noch die Fragestellung festgelegt werden:

Schalter A:	Ampel ROT	(A)	JA	1
			NEIN	0
Schalter B:	Ampel GELB	(B)	JA	1
			NEIN	0
Datenausgabe:	Halten	(L)	JA	1
			NEIN	0

Jetzt können Sie auch die ausführliche Funktionstabelle entwickeln.

Ampel ROT	Ampel GELB	Halten
A	B	F

Bitte, verwenden Sie in der Kurzform nur 0 und 1.

A	B	F

Leuchtet die Lampe der Datenausgabe auf, so müssen Sie vor der Ampel anhalten.

Wenn Ihnen das Aufstellen der Funktionstabelle und das Überprüfen Spaß gemacht haben, versuchen Sie vielleicht auch noch, ohne weitere Hilfe die nächste ODER-Funktion zu erproben. Sie können dazu dieselbe Schaltung und Programmierung einsetzen wie bei der vorigen Funktion.

Wann ist Abblendlicht einzuschalten? Die Straßenverkehrsordnung schreibt vor, Abblendlicht einzuschalten, wenn es dunkel oder neblig ist. Funktionstabelle:

A	B	F

3.5. Wie komme ich nach München? - Eine Exklusiv-ODER-Funktion

Sie wollen von Hamburg nach München reisen. Dafür stellen Sie sich zwei Fragen:

1. Reise ich mit dem Auto?
2. Reise ich mit dem Flugzeug?

Wollen Sie das Ergebnis aus diesen beiden Fragen von Ihrem Computer erfahren, müssen Sie ihn so programmieren (Frage 1: Schalter A; Frage 2: Schalter B):

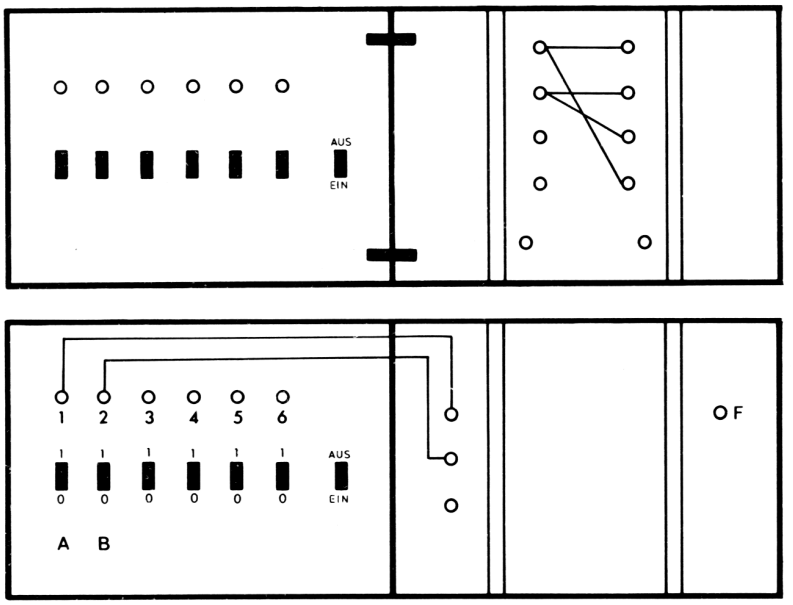


Abb.23

Hier wieder die entsprechende Tabelle, in die Sie die Computer-Antworten eintragen:

Dateneingabe		Datenausgabe
Reise ich mit dem Auto?	Reise ich mit dem Flugzeug?	Komme ich nach München?
A	B	F
0	0	
0	1	
1	0	
1	1	

Das ist die Lösung:

Dateneingabe		Datenausgabe
Reise ich mit dem Auto?	Reise ich mit dem Flugzeug?	Komme ich nach München?
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Sie kommen also nach München, wenn Sie entweder mit dem Auto oder mit dem Flugzeug reisen. Sollten Sie vorhaben, mit dem Auto und mit dem Flugzeug zu reisen, dann sagt Ihnen der Computer - und natürlich auch Ihre eigene Überlegung - daß Sie nicht nach München kommen. Denn mit Auto und Flugzeug zugleich, können Sie nicht reisen.

Im Unterschied zum einschließlichen ODER haben wir es hier mit einem ausschließlichen ODER zu tun. Die logische Funktion heißt dementsprechend ausschließliche ODER-Funktion bzw. Exklusiv-ODER-Funktion. Nur dann kommt das Ergebnis zustande, wenn entweder die eine Bedingung erfüllt ist oder die andere.

Die Funktionstabelle in Kurzform:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Da wir es auch hier mit einer logischen Schaltung zu tun haben, veranschaulichen Sie sich die Zusammenhänge wieder anhand der entsprechenden elektromechanischen Schaltung:

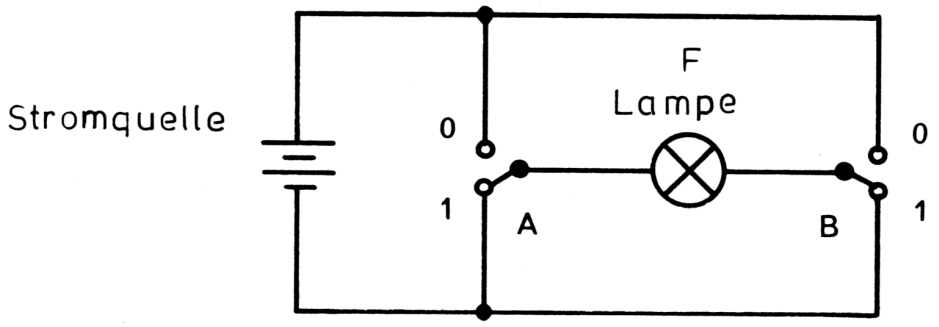
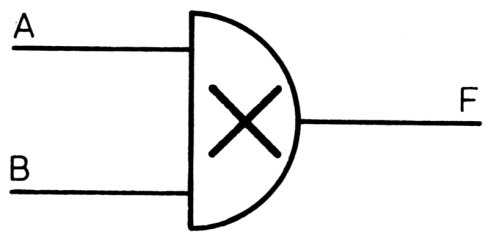


Abb.24

Und dies ist das Symbol für die Exklusiv-ODER-Schaltung:



3.6. Sommerurlaub

Kurz vor Ihrem Sommerurlaub, den Sie im Ausland verbringen möchten, haben Sie sich immer noch nicht für ein Ziel entscheiden können. Im Reisebüro teilt man Ihnen bedauernd mit, daß für die vorgesehene Zeit nur noch Plätze in Jugoslawien und auf Mallorca frei sind. Jetzt müssen Sie sich schnell entschließen: Entweder Sie fliegen nach Jugoslawien oder nach Mallorca.
Natürlich kann Ihnen der Computer diese schwere Entscheidung nicht abnehmen, aber er kann Ihnen die Antwort geben, ob Sie den Urlaub im Ausland verbringen werden oder nicht. (Vorausgesetzt, Sie finden keine andere Möglichkeit, ins Ausland zu reisen.)

Sie merken sicherlich schon, daß auch dieser Fall ein Beispiel für eine Exclusiv-ODER-Funktion ist. Denn es gibt nur die Möglichkeit, nach Jugoslawien oder nach Mallorca zu reisen. Für beide Ziele können Sie sich eben nicht gleichzeitig anmelden. Es gilt also:

Reise ich nach Jugoslawien? (Schalter A)
Reise ich nach Mallorca? (Schalter B)

Wenn Sie den Logik-Baustein wie auf der folgenden Abbildung programmieren, gibt Ihnen der Computer die entsprechende Antwort.

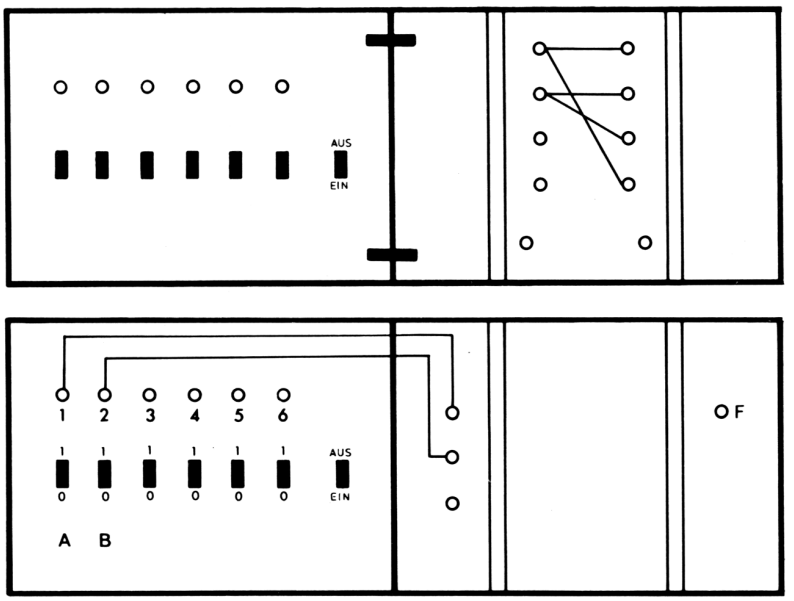


Abb.25

Dateneingabe		Datenausgabe
Jugoslawien	Mallorca	Auslandsaufenthalt
A	B	F
0	0	
0	1	
1	0	
1	1	

Das Aufleuchten der Lampe (Datenausgabe) bedeutet, in welchem Fall Sie Ihren Urlaub im Ausland verbringen werden. Ganz bestimmt gelingt es Ihnen, die Kurzform der Funktionstabelle zu ermitteln. Sie brauchen die Eingabebefehle und den Ausgang nur in der Form 0 oder 1 einzusetzen.

A	B	F

Sicherlich fallen Ihnen noch weitere Beispiele zu einem Exclusiv-ODER ein. Versuchen Sie doch einmal, dazu die Funktionstabellen aufzustellen und diese mit dem Computer zu überprüfen. Der umgekehrte Weg ist natürlich auch möglich.

3.7. Wann ist die Waage im Gleichgewicht?

Für die hier gezeigte Balkenwaage haben wir zwei gleichschwere Gewichte zur Verfügung.

Selbstverständlich ist leicht zu erkennen, in welchen Fällen sich diese Waage im Gleichgewicht befindet. Probieren wir die verschiedenen Möglichkeiten mit unserem Computer durch. Programmieren Sie dazu wie folgt:

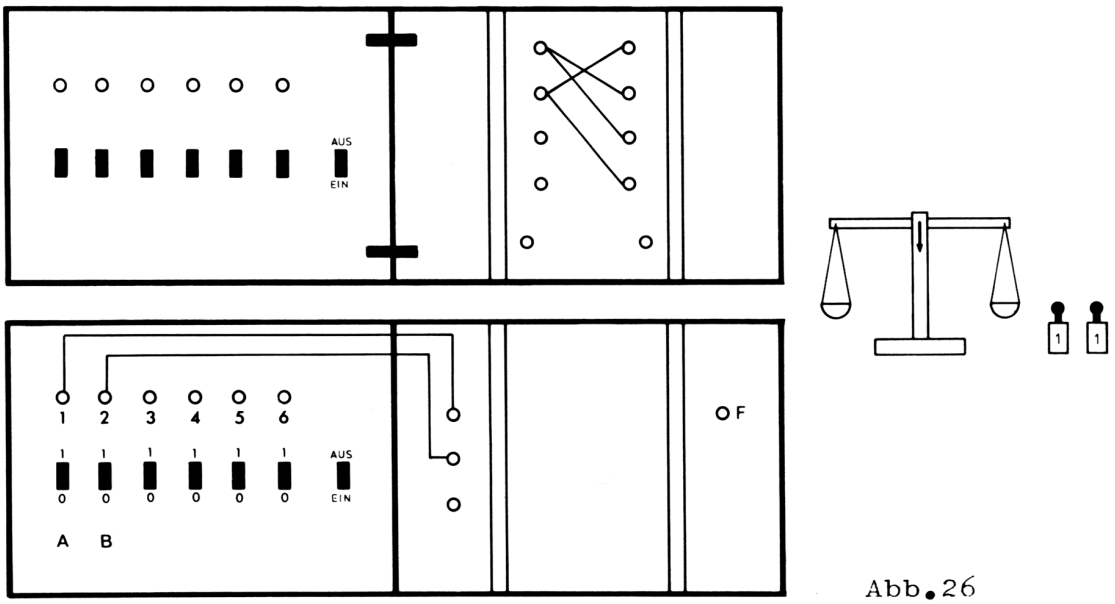


Abb.26

Treffen wir nun die notwendigen Zuordnungen:

Kein Gewicht auf der Waagschale : 0
Gewicht auf der Waagschale : 1
Waage nicht im Gleichgewicht : 0
Waage im Gleichgewicht : 1

Ermitteln Sie nun, in welchen Fällen sich die Waage im Gleichgewicht befindet:

Dateneingabe		Datenausgabe
Liegt ein Gewicht auf der linken Waagschale?	Liegt ein Gewicht auf der rechten Waagschale?	Ist die Waage im Gleichgewicht?
A	B	F
0	0	
0	1	
1	0	
1	1	

Die Lösung ist sehr einleuchtend:

Dateneingabe		Datenausgabe
Liegt ein Gewicht auf der linken Waagschale?	Liegt ein Gewicht auf der rechten Waagschale?	Ist die Waage im Gleichgewicht?
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Die Waage ist nur dann im Gleichgewicht, wenn beide Seiten gleich stark belastet (oder nicht belastet) sind. Mit anderen Worten: Wenn die linke Waagschale nicht belastet ist und die rechte Waagschale nicht belastet ist bzw. wenn die linke Waagschale belastet ist und wenn die rechte Waagschale belastet ist (gleiche Gewichte), dann befindet sich die Waage im Gleichgewicht. Es handelt sich hier um die sog. Äquivalenz-Funktion, was zu deutsch soviel wie Gleichheitsfunktion heißt.
Hier wieder die entsprechende Funktionstabelle in abstrahierter Form:

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Und dies ist die
Äquivalenz-Schaltung
in elektromechanischer
Form:

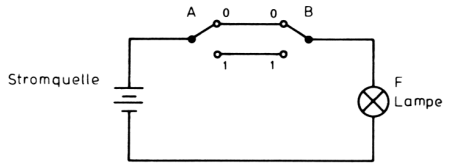
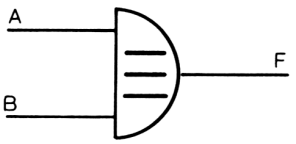


Abb.27

Das Schaltsymbol für die
Äquivalenz-Schaltung
sieht so aus:



3.8. Bleibt die Erde bei Regen trocken? - Identität und Negation

Was für eine törichte Frage! - Aber lassen Sie uns trotzdem kurz bei dieser Frage verweilen, um die zwei einfachsten logischen Funktionen kennenzulernen.

Stellen wir uns die Frage zunächst anders herum: Wird die Erde naß, wenn es regnet? Wollen wir die Antwort auf diese "schwierige" Frage von unserem Computer haben, dann müssen wir ihn so programmieren:

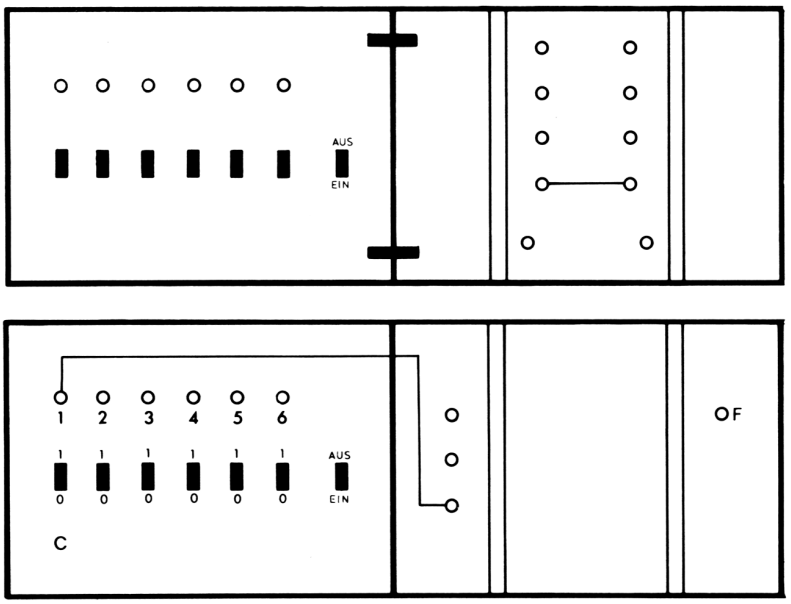


Abb.28

Wenn wir unsere Ergebnisse wieder in eine Tabelle eintragen und dabei wieder für NEIN = 0 und für JA = 1 setzen, dann ergibt sich folgendes Bild:

Regnet es?	Wird die Erde naß?
C	F
0	0
1	1

Mit anderen Worten: Wenn es regnet, dann wird die Erde naß, bzw.: Wenn es nicht regnet, dann wird die Erde nicht naß. Diese logische Funktion ist die sog. Identitäts-Funktion, zu deutsch könnte man sagen, Übereinstimmungsfunktion.

Als Schaltbild sieht das so aus:

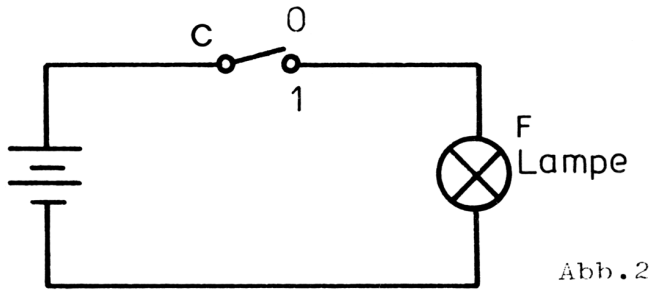


Abb.29

Wenn der Schalter geschlossen ist, dann brennt die Lampe, und wenn der Schalter nicht geschlossen ist, dann brennt die Lampe nicht.
Wie steht es nun mit der Frage: "Bleibt die Erde trocken, wenn es regnet?" Programmieren Sie den Logik-Baustein jetzt wie folgt:

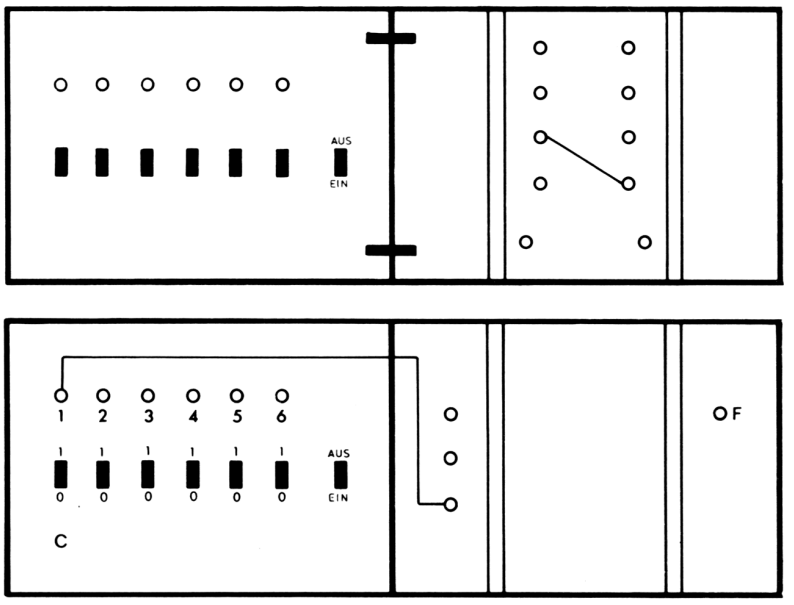


Abb.3o

Die zugehörige Funktionstabelle:

Regnet es?	Bleibt die Erde trocken?
C	F
0	1
1	0

Wenn es nicht regnet, dann bleibt die Erde trocken. Wenn es regnet, dann bleibt die Erde nicht trocken. Bei dieser logischen Funktion handelt es sich um eine Negations-Funktion, was soviel heißt wie Verneinungsfunktion.

Die Funktionstabellen für Identität und Negation:

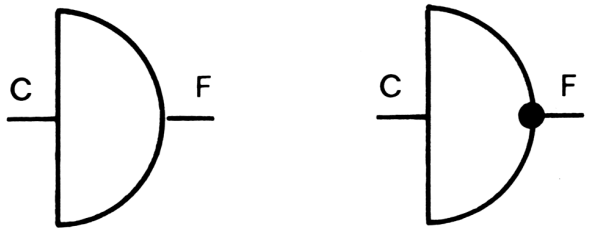
Identität

C	F
0	0
1	1

Negation

C	F
0	1
1	0

und die Schaltsymbole:



3.9. Welches ist der richtige Weg?

Ein Wanderer trifft an einer Wegegabelung auf einen Mann. Über diesen Mann ist bekannt, daß er einer von zwei Brüdern ist, die je eine sehr ausgeprägte Eigenschaft besitzen: Einer der Brüder lügt immer, der andere sagt stets die Wahrheit. Dem Wanderer - er kennt nämlich den Weg nicht - stellt sich das Problem durch Befragung des Mannes, von dem er nicht weiß, ob er der Lügner (L) oder der Wahrheitssager (W) ist, auf den richtigen Weg zu gelangen.

Fragt er Bruder W, ob dies der richtige Weg sei, indem er gleichzeitig auf den richtigen Weg weist, so wird der mit JA antworten. Zeigt er auf den flaschen Weg, so wird die Antwort NEIN sein.

Dieser Bruder W verändert also die mit der Frage gegebene Information nicht, sondern er bestätigt sie oder bestätigt sie auch nicht.

Richtiger Weg	Bruder W
C	F
0	0
1	1

Es handelt sich also um eine Identitätsfunktion. - Bruder L dagegen würde die Information verändern: Stellt ihm der Wanderer dieselbe Frage und zeigt dabei auf den richtigen Weg, dann antwortet L mit NEIN. Weist er bei dieser Fragestellung auf den falschen Weg, lautet die Antwort JA.

Richtiger Weg	Bruder L
C	F
0	1
1	0

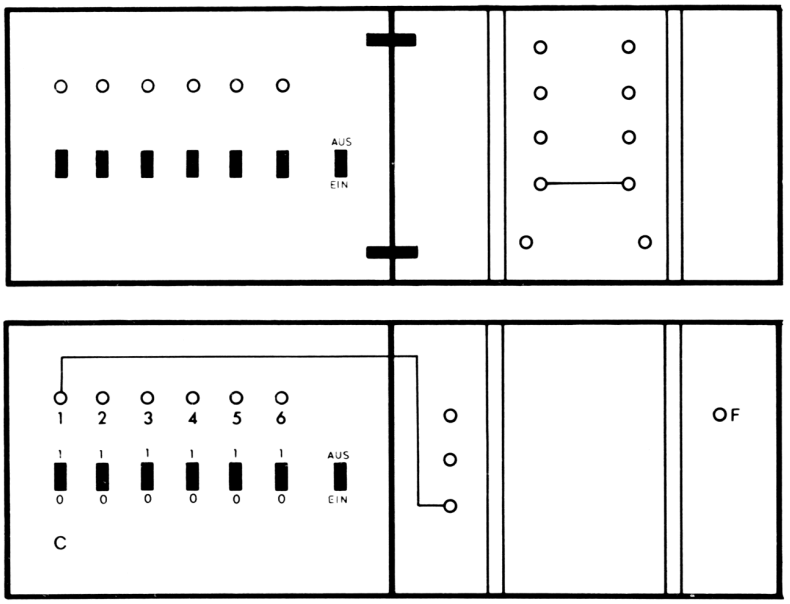


Abb. 31
Identität

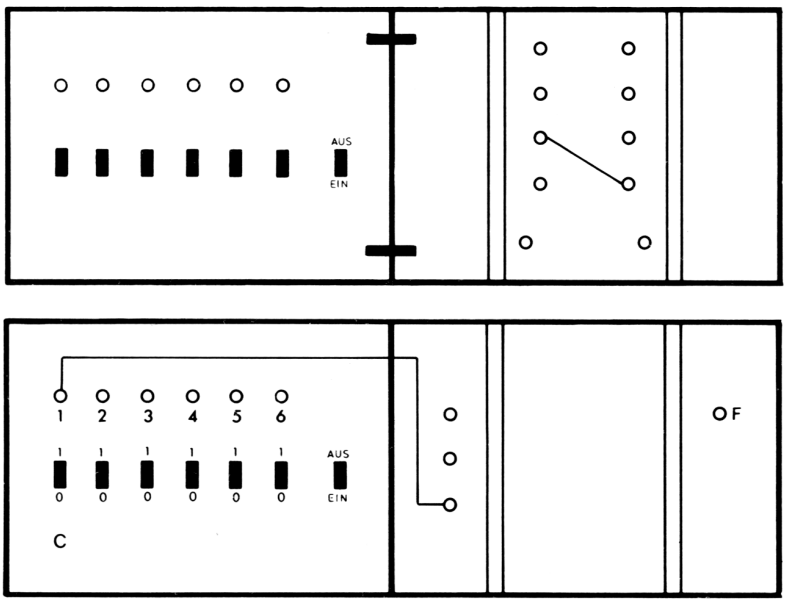


Abb. 32
Negation

Er kehrt also die gegebene Information um, so daß es sich um eine Negationsfunktion handelt. Der Ausgang des Logik-Bausteins stellt die Umkehrung des Eingangs dar.

Der arme Wanderer muß sich also eine Frage einfallen lassen, auf die er eine eindeutige Antwort bekommt. Das ist nur möglich, wenn eine Frage beide Brüder gleichzeitig einschließt. Sie muß also lauten: "Würde dein Bruder zustimmen, daß dies der richtige Weg ist?", wobei er auf irgendeinen Weg deutet.

Bruder W antwortet, sofern es sich um den richtigen Weg handelt, mit NEIN. Denn sein Bruder L würde dann nicht zustimmen. Bruder L dagegen antwortet im gleichen Fall auch mit NEIN. Denn da er immer lügt, wird er die Antwort von W wieder umkehren.
Durch diese geschickte Frage gelangt der Wanderer doch noch an sein Ziel.

3.1o. Eine Badewanne wird gefüllt

Für das Füllen einer Badewanne kommen folgende Eingangsbedingungen in Frage:

Der Abfluß ist geöffnet	(A = 0)
Der Abfluß ist geschlossen	(A = 1)
Der Kaltwasserhahn ist geschlossen	(B = 0)
Der Kaltwasserhahn ist geöffnet	(B = 1)
Der Warmwasserhahn ist geschlossen	(C = 0)
Der Warmwasserhahn ist geöffnet	(C = 1)

Bitte, achten Sie darauf, daß wir bei den Wasserhähnen sprachlich anders verfahren als bei den Stromschaltern unserer vorangegangenen Beispiele: Wenn wir sagen, ein Stromschalter ist geöffnet, dann heißt das: Es fließt kein Strom (0). Wenn wir dagegen von einem Wasserhahn sagen, daß er geöffnet ist, so bedeutet das: Das Wasser fließt (1).
Wollen Sie nun mit Hilfe eines Logik-Bausteins erfahren, unter welchen Bedingungen sich die Badewanne mit Wasser füllt, so nehmen Sie folgende Programmierung vor:

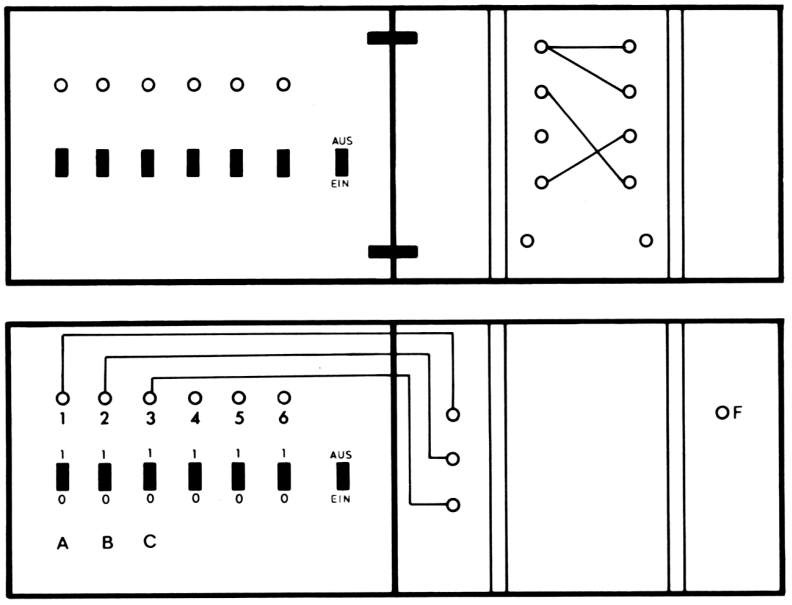


Abb.33

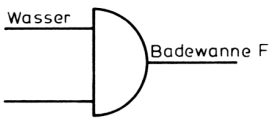
Tragen Sie die Ergebnisse wieder in die Tabelle ein:

Abfluß geschlossen?	Kaltwasserhahn geöffnet?	Warmwasserhahn geöffnet?	Wird Badewanne gefüllt
A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

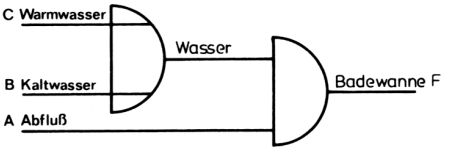
Das Ergebnis:

Abfluß geschlossen?	Kaltwasserhahn geöffnet?	Warmwasserhahn geöffnet?	Wird Badewanne gefüllt
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Das Ergebnis ist ganz einleuchtend, wenn Sie sich die Zusammenhänge sprachlich vor Augen halten: Wenn der Abfluß geschlossen ist und wenn der Kaltwasserhahn oder der Warmwasserhahn geöffnet ist, dann wird die Wanne mit Wasser gefüllt. Mit anderen Worten: Der Abfluß muß auf jeden Fall geschlossen sein, und es muß in jedem Fall Wasser laufen. Damit haben wir eine UND-Funktion.



Das Wasser fließt, wenn der Kaltwasserhahn geöffnet ist oder wenn der Warmwasserhahn geöffnet ist oder auch wenn beide geöffnet sind. Damit sind Kaltwasserhahn und Warmwasserhahn in einer ODER-Funktion verbunden. So ergibt sich insgesamt folgendes Schaltbild:



Zum Abschluß dieses Beispiels noch die Funktionstabelle in allgemeiner Form:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

3.11. Ein Lagerfeuer wird entzündet

Ein weiteres Beispiel für die Verknüpfung einer UND-Funktion mit einer ODER-Funktion:

Sie möchten auf einem freien Platz ein kleines Lagerfeuer entzünden. Voraussetzung dafür sind in jedem Fall, daß Brennmaterial und Streichhölzer vorhanden sind. Diese beiden Eingangsbedingungen stellen in diesem Fall eine UND-Funktion dar.

Nun müssen Sie das Feuer aber nicht unbedingt mit Streichhölzern entzünden, sondern Sie können natürlich auch ein Feuerzeug einsetzen. Dies ist ein Einschließendes ODER. Denn es müssen entweder Streichhölzer oder ein Feuerzeug vorhanden sein. (Beides ist natürlich auch möglich und in der Praxis manchmal sehr nützlich.) Fassen wir also alle Eingangsbedingungen zusammen:

Brennmaterial nicht vorhanden	0	Schalter A
Brennmaterial vorhanden	1	Schalter A
Streichhölzer nicht vorhanden	0	Schalter B
Streichhölzer vorhanden	1	Schalter B
Feuerzeug nicht vorhanden	0	Schalter C
Feuerzeug vorhanden	1	Schalter C

Das Aufleuchten der Lampe des Logik-Bausteins zeigt an, unter welchen Bedingungen Sie ein Feuer entzünden können. Doch zunächst gilt es, den Logik-Baustein zu programmieren und mit der Eingabeeinheit zu verbinden.

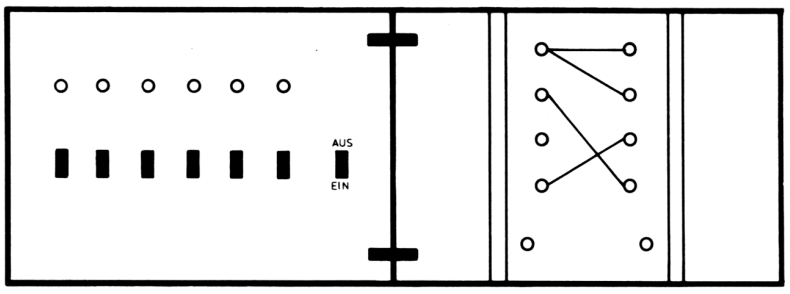


Abb. 34

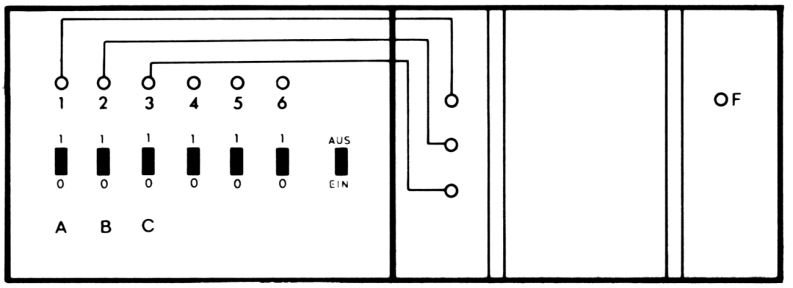


Abb. 34

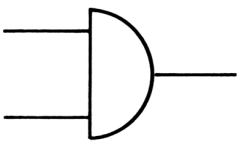
Die Ausgabedaten können Sie in die Funktionstabelle eintragen. (Lampe AUS = 0, Lampe EIN = 1)

Brennmaterial vorhanden?	Streichhölzer vorhanden?	Feuerzeug vorhanden?	Feuer brennt
A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Formuliert lautet das Ergebnis: Wenn Brennmaterial und Streichhölzer oder ein Feuerzeug vorhanden sind, läßt sich das Lagerfeuer entzünden. In der Kurzform sieht die Funktionstabelle so aus:

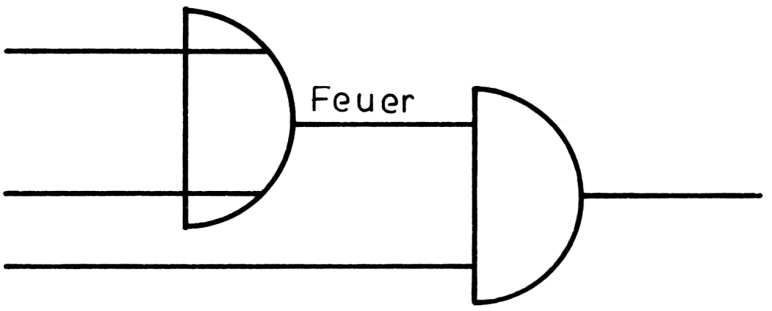
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Versuchen wir auch für dieses Beispiel, die Schaltsymbole für die UND-Funktionen einzubeziehen.



Sicherlich bereitet es Ihnen keine Mühe, die Eingänge und den Ausgang dieses UND-Symbols - bezogen auf den beschriebenen Fall - zu beschriften.

Verknüpft man die UND-Funktion mit der ODER-Funktion und stellt diese Verknüpfung durch die Symbole dar, entsteht folgendes Bild:



Wie müssen die Eingänge und der Ausgang beschriftet werden?

Bevor wir nun weiter in die Computerlogik eindringen und bevor wir mit mehreren Logik-Bausteinen experimentieren, lassen Sie uns noch einige grundsätzliche Betrachtungen über Datenverarbeitung anstellen.

4. Aufbau eines Computers

4.1. Ist der Mensch ein Computer?

Diese Frage drängt sich uns unweigerlich auf, wenn wir uns die erstaunlichen Ähnlichkeiten zwischen menschlichem Denken und Datenverarbeitung durch einen Computer veranschaulichen.
In dem Abschnitt über Begriffserklärungen hatten wir schon besprochen, was Daten sind. Daten sind Informationen. Information ist alles das, was uns in irgendeiner Form eine Bedeutung übermittelt, also z.B. gesprochene oder geschriebene Wörter und Sätze, Zahlen, Zeichnungen und Bilder. Datenverarbeitung ist die Verarbeitung von Eingabedaten mittels eines Programms zu Ausgabedaten. Statt Dateneingabe können wir auch Datenaufnahme sagen: Ein Computer nimmt Daten auf, die ihm eingegeben werden, und auch Sie selbst nehmen Daten auf.

Was geschieht z.B. jetzt im Augenblick? Sie lesen den Text dieser Anleitung. Sie nehmen die Buchstaben bzw. Wörter, die Zahlen und Abbildungen auf. Das ist kurz gesagt Datenaufnahme. Dafür, also für Sehen und Lesen, haben Sie ein spezielles Instrumentarium: Ihre Augen. Ebenso ist Hören Datenaufnahme, durch die Ohren als Datenaufnahmeinstrumente. Wir können auch die anderen drei unserer fünf Sinne als Funktionen mit Datenaufnahme verstehen: Fühlen, Schmecken, Riechen. Unsere Instrumente dazu: Haut, Zunge, Nase bzw. die entsprechenden Nerven.
Und was geht beim Sprechen vor sich? - Es werden Daten ausgegeben. Sprechen ist also eine Funktion der Datenausgabe. Das entsprechende Datenausgabeinstrument: Der Mund. Daten geben wir auch aus, wenn wir etwas schreiben oder zeichnen. Im allgemeinen ist unsere rechte Hand - mit Schreib- oder Zeichenstift - dabei das Datenausgabeinstrument. Wir haben noch weitere Möglichkeiten, uns verständlich zu machen. Bekanntlich kann auch Gestikulieren etwas Bestimmtes bedeuten. Gestikulieren ist also ebenfalls Datenausgabe, und zwar durch das Instrumentarium der Hände. Dann haben wir noch das Gesicht als Ausdrucksmöglichkeit, indem wir durch die Mimik Informationen, also Daten, weitergeben.

Zwischen der Datenaufnahme und der Datenausgabe liegt die eigentliche Datenverarbeitung. Wir können auch von Datenverarbeitung im engeren Sinn sprechen, wenn wir Datenaufnahme, Datenverarbeitung (im engeren Sinn) und Datenausgabe zusammen als Datenverarbeitung im weiteren Sinn verstehen. Wo findet die Datenverarbeitung im engeren Sinn bei uns statt? - Im Gehirn. - Und was ist das nun im einzelnen, Datenverarbeitung im

engeren Sinn? - Veranschaulichen Sie sich das am besten anhand Ihrer gegenwärtigen Situation. Sie lesen dieses Anleitungsbuch, Sie lesen den Text und schauen sich die Abbildungen an. Sie wissen bereits: Das bedeutet zunächst einmal Datenaufnahme. Einen Teil des Inhalts dieses Buches nehmen Sie in Ihr Gedächtnis auf, Sie speichern Daten. Aber das ist noch das wenigste. Was Sie insgesamt bei der Datenverarbeitung tun, bezeichnen wir als "denken". Und Denken ist eine überaus komplizierte Angelegenheit. Wir wollen dem Begriff Denken nicht bis auf den Grund gehen, das würde viel zu kompliziert werden; wir wollen uns statt dessen mit einigen Andeutungen zufrieden geben.

Funktionen des Denkens, also der Datenverarbeitung im engeren Sinn, sind:

Ordnen:

F		A
D		B
C	⟹	C
A		D
E		E
B		F

Vergleichen: $6 \stackrel{?}{=} 8$

Kontrollieren: $3 \times 4 = 13$ Kontrolle: Korrektur $3 \times 4 = 12$

Schlußfolgern: Der Badewannenabfluß ist geschlossen und der Kaltwasserhahn ist geöffnet. Daraus folgt: Die Badewanne wird mit Wasser gefüllt.

Entscheiden: Beschäftige ich mich auch weiterhin mit meinem Philips Computer-Lehrbaukasten? - Ja/Nein.

Mit diesen Beispielen für Funktionen des Denkens wollen wir es genug sein lassen. Fassen wir diese Funktionen unter einem neuen Oberbegriff zusammen: Steuern.
Unter der Funktion Steuern wollen wir hauptsächlich jene Funktionen verstehen, die dazu dienen, Datenverarbeitung durchzuführen bzw. Verarbeitung von Daten zu steuern. Das mag Ihnen im Moment etwas undurchsichtig erscheinen, klarer wird es jedenfalls noch im Zusammenhang mit Datenverarbeitung durch den Computer.
Eine Funktion der Datenverarbeitung im engeren Sinn bleibt aber noch zu erwähnen: Rechnen.
Die drei Hauptfunktionen der Datenverarbeitung im engeren Sinn sind also: Speichern, Steuern, Rechnen.

Wir sprachen vorhin davon, daß Sie sich einen Teil dieses Anleitungsbuches im Gedächtnis merken werden. Wenn Sie sich ausgiebig mit dem Anleitungsbuch und Ihrem Philips Computer-Lehrbaukasten befassen, werden Ihnen die wichtigsten Zusammenhänge und sicherlich auch manch ein Detail

im Gedächtnis haften bleiben. Aber Sie werden sich den Text nicht wortwörtlich einverleiben, das wäre auch nicht unbedingt sinnvoll. Gewiß ist es eine nützliche Einrichtung, daß wir nicht jede Kleinigkeit für alle Zeiten im Gedächtnis behalten, das würde unsere Gedanken nur belasten. Auf der anderen Seite aber gibt es viele Dinge, die wir gerne behalten würden. Das ist jedoch nicht immer ohne weiteres möglich, da wir erstens in einer bestimmten Zeit nur eine bestimmte Menge von Daten aufnehmen können und da wir zweitens auch vergeßlich sind. Der Schwierigkeit kann abgeholfen werden, denn unser Gedächtnis ist ja nicht die einzige Möglichkeit zur Speicherung von Daten.
Nehmen wir an, Sie hören einen interessanten Vortrag; vielleicht machen Sie sich Notizen. Oder ein Freund teilt Ihnen seine neue Telefonnummer mit; Sie notieren sich die Nummer. Oder aber Sie haben nach einiger Zeit Dinge aus diesem Anleitungsbuch, die Sie besonders interessieren, nicht mehr ganz gegenwärtig. Was werden Sie tun? Sie nehmen sich die Anleitung wieder zur Hand und lesen nach. In allen drei angeführten Fällen speichern Sie Daten auf externen Datenspeichern im Unterschied zu Ihrem internen Datenspeicher, Ihrem Gedächtnis. Natürlich können Sie zur externen Datenspeicherung auch Tonbandgerät und Tonbänder benutzen. In jedem Fall: Durch externe Datenspeicherung können wir unser Gedächtnis entlasten.

Nicht nur die Datenspeicherung können wir uns durch Hilfsmittel erleichtern, auch die Datenverarbeitung muß nicht in allen Einzelheiten in unserem Gehirn vor sich gehen. Einfache Hilfsmittel zur externen Datenverarbeitung sind z.B. Rechenschieber oder die Registrierkasse mit Additionswerk. Externe Datenverarbeitung betreiben Sie aber auch, wenn Sie mit Ihrem Philips Computer-Lehrbaukasten experimentieren. Darüber hinaus stehen uns heute Hilfsmittel zur Verfügung, die komplizierteste Datenverarbeitungsaufgaben vollautomatisch durchführen, und das in unvorstellbaren Geschwindigkeiten: Computer.

4.2. Ein großer Computer wird vorgestellt

Befassen wir uns ein wenig mit dem Aufbau eines großen Computers. Nicht zuletzt deshalb, weil es für Sie interessant ist zu wissen, in welcher Beziehung die Funktionen Ihres Computer-Lehrbaukastens zu den Funktionen einer großen Datenverarbeitungsanlage stehen.
Wie Sie bereits wissen, beginnt jede Datenverarbeitung mit der Datenaufnahme. Nachdem wir bei der Datenverarbeitung des Menschen von Datenaufnahme gesprochen haben, wollen wir beim Computer wieder Dateneingabe

sagen, um damit anzudeuten, daß sich der Computer die Daten nicht selbst besorgt, sondern daß der Mensch ihm die Daten eingibt. Das geht natürlich nicht auf beliebigen Notizzetteln oder etwa im zwanglosen Gespräch mit dem Computer. Wir können zwar mit Datenträgern wie Notizzetteln und Büchern etwas anfangen, der Computer aber braucht seine Daten auf maschinell lesbaren Datenträgern. Der bekannteste maschinell lesbare Datenträger ist die Lochkarte.

Daten, die für den Computer lesbar auf Lochkarten verschlüsselt bzw., wie man auch sagt, codiert sind, werden über einen Lochkartenleser in den Computer eingegeben. Ein anderer, maschinell lesbarer Datenträger ist der Lochstreifen. Der Inhalt von Lochstreifen wird über einen Lochstreifenleser eingelesen. Außer der Dateneingabe über Lochkarten und Lochstreifen ist auch eine direkte Dateneingabe über eine Tastatur möglich. Eine solche Tastatur ist mit Zahlen- und Buchstabentasten versehen.

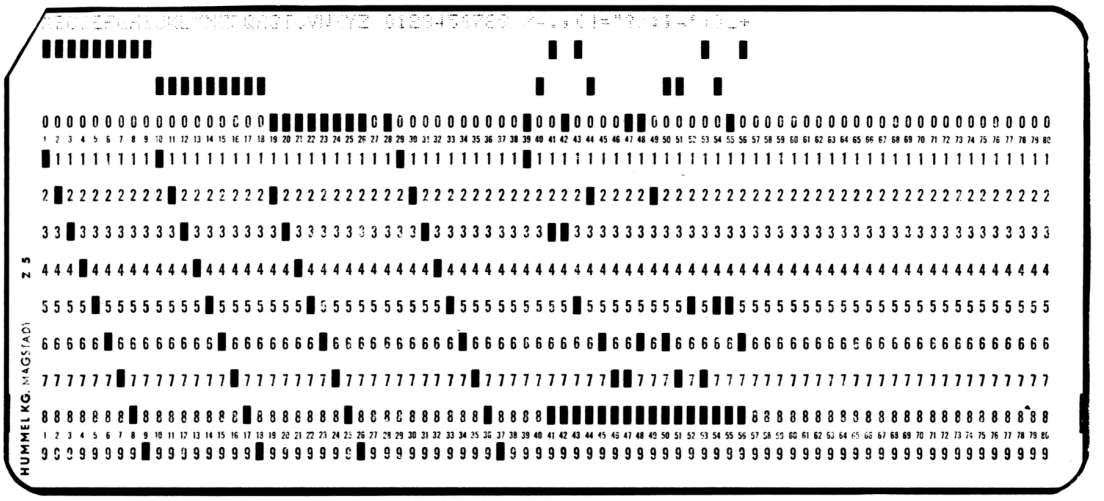
Wie beim Menschen gibt es auch beim Computer Datenausgabe. Computer können Daten beispielsweise über Lochkartenstanzer auf Lochkarten ausgeben. Außerdem können Ausgabedaten durch Lochstreifenstanzer in Lochstreifen gestanzt werden, und schließlich können Daten auch in Schrift, also Buchstaben, Ziffern und in Zeichen wie . , ; + % usw. über Schnelldrukker auf Listen ausgegeben werden.

Die eigentliche Datenverarbeitung, also die Datenverarbeitung im engeren Sinn, findet beim Computer in der sog. Zentraleinheit statt. Die Zentraleinheit ist gewissermaßen das Gehirn des Computers. In der Zentraleinheit hat die Elektronik ihren Hauptwirkungsbereich. Hier finden, oft in Bruchteilen von Sekunden, die kompliziertesten Datenverarbeitungsvorgänge statt - mit einer Zuverlässigkeit, wie sie der Mensch niemals erreichen könnte. Wenn wir die Zentraleinheit als das Gehirn des Computers verstehen, so entspricht der Zentralspeicher dem Gedächtnis. Im Zentralspeicher werden Daten intern gespeichert.

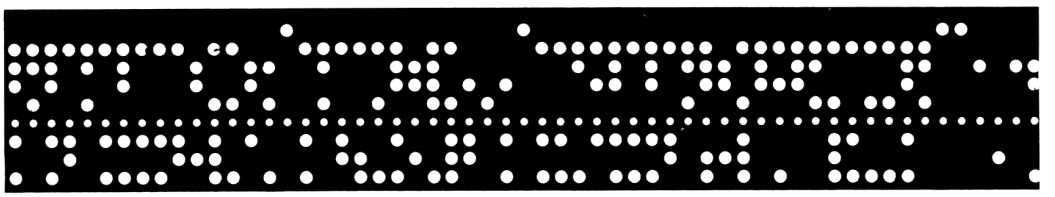
Das Rechenwerk ist für alle Rechenoperationen zuständig. Dabei mag folgender Sachverhalt erstaunlich erscheinen: Das Rechenwerk verfährt auch mit Buchstaben wie mit Zahlen, da Buchstaben in einem Computer wie Zahlen verschlüsselt werden. Aber das sei nur am Rande vermerkt. Die Rechenoperationen beruhen zum größten Teil wiederum auf logischen Funktionen, von denen Sie ja schon einige kennengelernt haben.

Das Steuerwerk schließlich führt gewissermaßen Regie bei der Datenverarbeitung. Das Steuerwerk sorgt dafür, daß Daten eingelesen, nach Programm verarbeitet und ausgegeben werden.

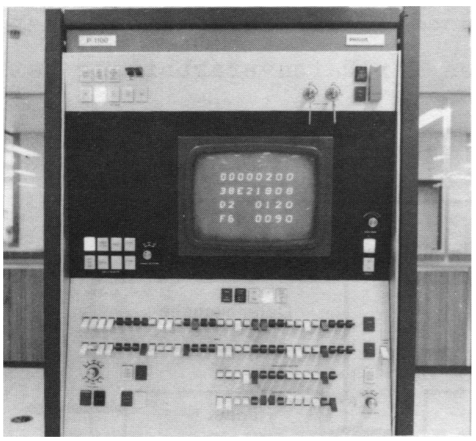
Der Mensch kann sein Gedächtnis durch externe Datenspeicherung entlasten. Auch ein Computer braucht nicht alle Daten in seinem Gedächtnis, dem Zentralspeicher, aufzubewahren. Auch der Computer kann sich externer Datenspeicher bedienen. Er kann Daten z.B. über Magnetbandstationen auf Magnetbänder abgeben, um sie im Bedarfsfall von dort wieder abzurufen. Auch Magnetplatten sind Datenträger für externe Datenspeicherung, die entsprechenden Maschineneinheiten sind Magnetplattenstationen. Hat ein Computer Daten auf Lochkarten oder Lochstreifen ausgegeben, so können diese Daten für eine neue Datenverarbeitungsaufgabe über Lochkarten- bzw. Lochstreifenleser als Eingabedaten wieder eingelesen werden. So gesehen, können auch Lochkarte und Lochstreifen als externe Datenspeicher verstanden werden.



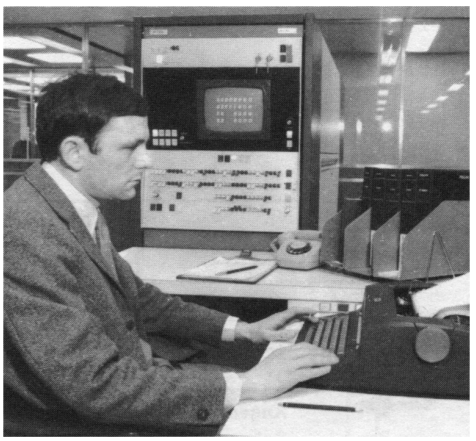
Der Grundtyp der Lochkarte. Durch ein, zwei oder drei Lochungen je Spalte werden die in der EDV (Elektronische Datenverarbeitung) verwendeten Zeichen codiert.



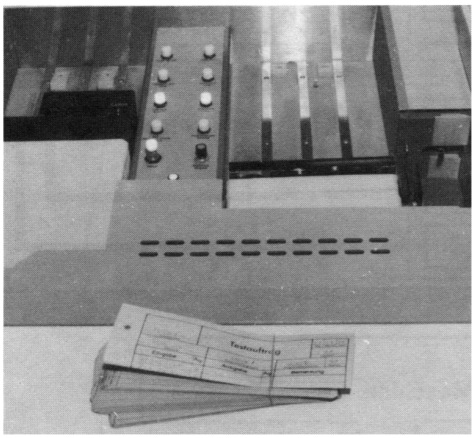
Ein Lochstreifen.



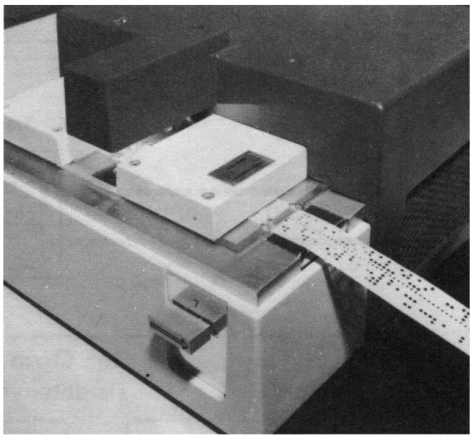
Frontseite Zentraleinheit
- das Gesicht des Computers



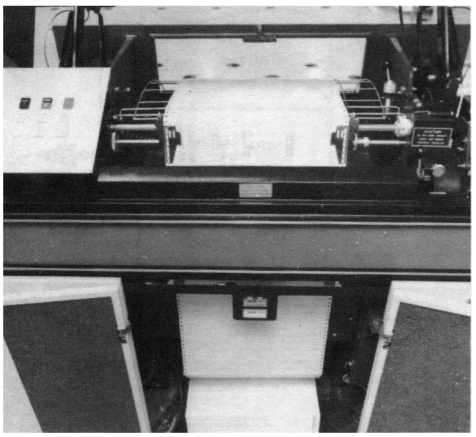
Kommunikation
Mensch - Maschine



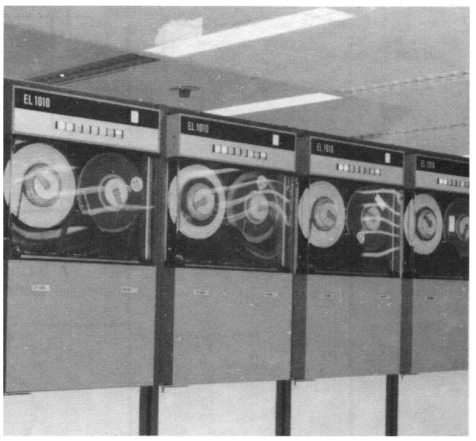
Lochkartenleser



Lochstreifenleser

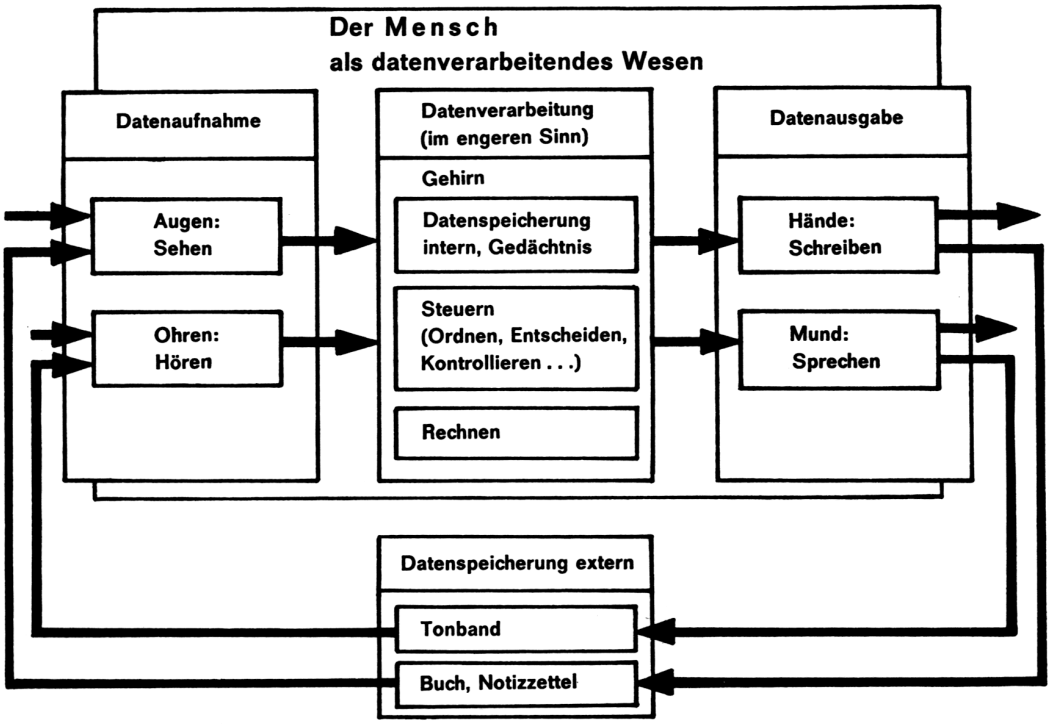


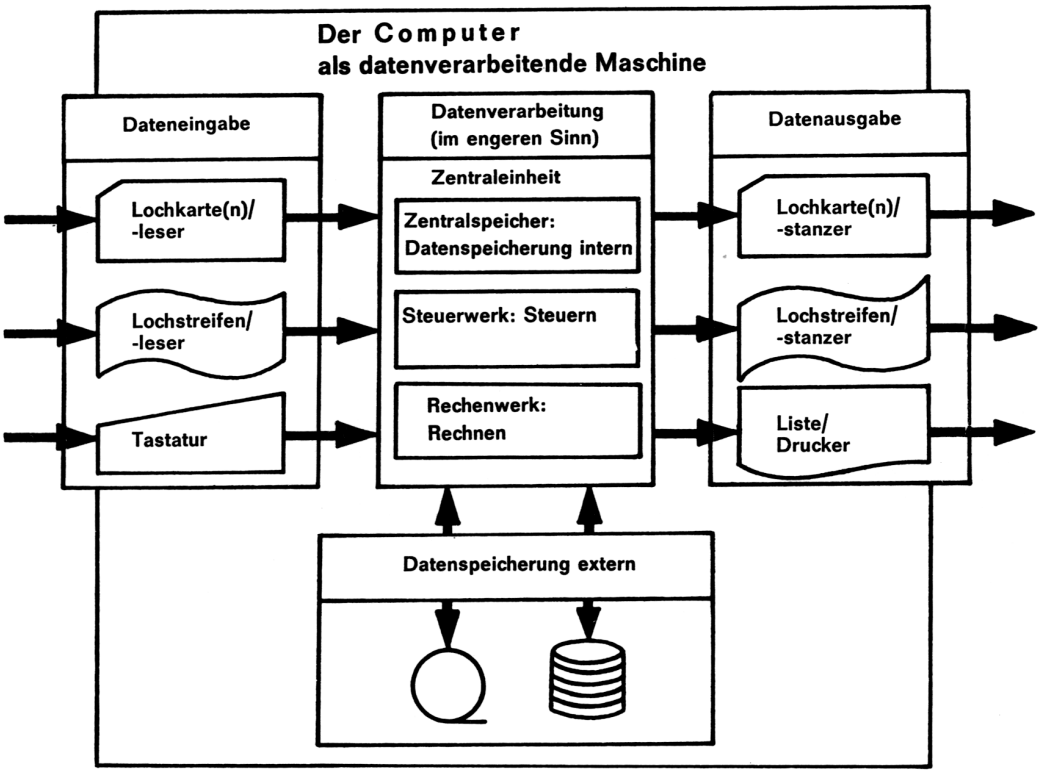
Ein Schnelldrucker



Magnetbandstation

Die beiden Abbildungen zeigen Ihnen noch einmal in schematischer Übersicht die Funktionen und Einheiten für Datenverarbeitung bei Mensch und Computer.





4.3. Das Programm

Übertragen wir nun einem Computer eine bestimmte Datenverarbeitungsaufgabe, beispielsweise die Lohnabrechnung eines Betriebes. Dazu benötigt der Computer Daten, wie die Namen der Lohnempfänger, Zahl der geleisteten Arbeitsstunden, Stundenlohn, Lohnsteuersätze usw. Diese Daten stellen wir auf Lochkarten zur Verfügung, und der Computer kann an die Arbeit gehen. Oder? - Nein, so einfach geht es nun doch nicht, denn noch "weiß" der Computer gar nicht, was er mit den Daten anfangen soll; er "weiß" ja nicht, wie eine Lohnabrechnung vor sich geht; er "weiß" sogar nicht einmal, daß er eine Lohnabrechnung machen soll. Daten allein genügen also nicht. Der Computer braucht ein Programm: Ein Programm, das ihm in allen Einzelheiten Anweisungen gibt, wie er mit welchen Daten verfahren soll. Bevor der Computer das Programm nicht kennt, kann er die Aufgabe nicht durchführen.

Und wie ist das beim Menschen? - Auch der Mensch braucht bestimmte Programme, um Datenverarbeitung durchführen zu können. Wenn wir die Lohnabrechnung z.B. durch einen Buchhalter vornehmen lassen, so ist es der Buchhalter, der nach einem Programm arbeitet, nämlich nach einem Lohnabrechnungsprogramm, das er kennt, das er also im Gedächtnis hat.
So wie der Mensch sein Programm, nach dem er eine Datenverarbeitungsaufgabe durchführen will, im Gedächtnis hat, muß auch der Computer sein Programm im Gedächtnis, also im Zentralspeicher, haben. Natürlich muß das Programm erst entworfen, dann z. B. auf Lochkarten übertragen und schließlich über den Lochkartenleser eingelesen werden - in den Zentralspeicher. Dann kann der Computer mit der Datenverarbeitung beginnen.

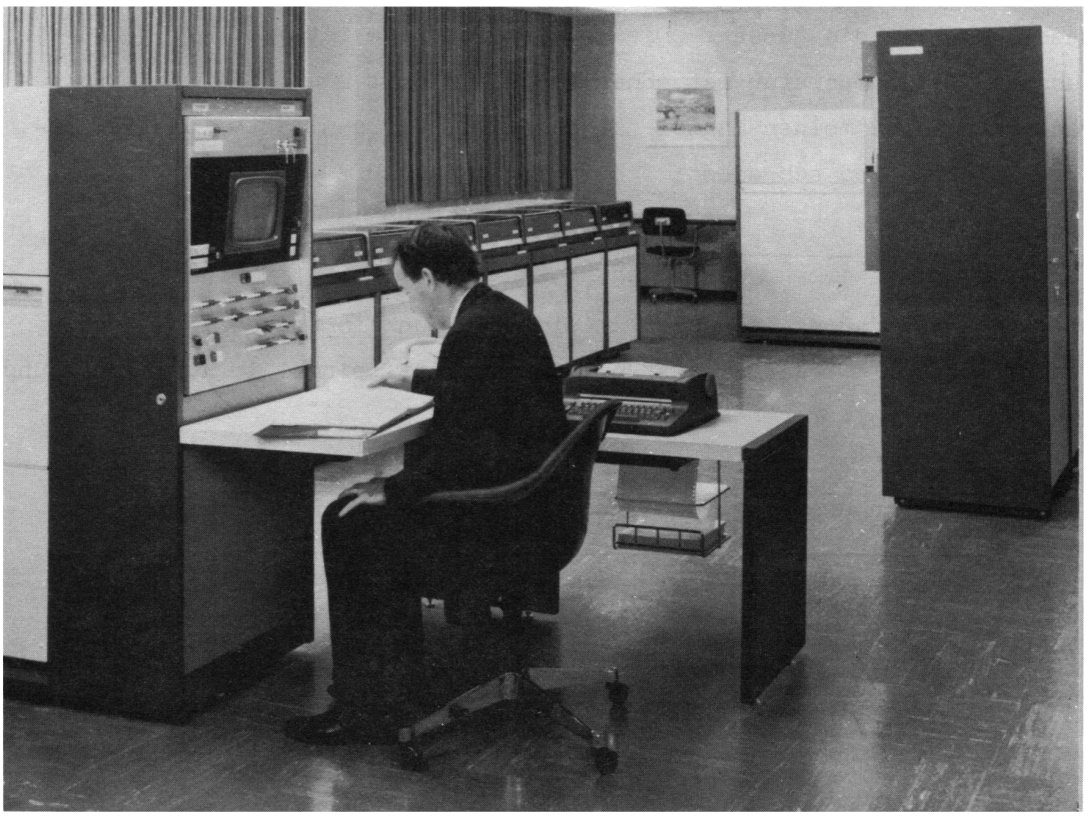
Sie wissen nun - in groben Zügen - was Datenverarbeitung ist. Sie kennen den Menschen als datenverarbeitendes Wesen, und Sie haben einen Eindruck vom Computer als datenverarbeitende Maschine. Beide, Mensch und Computer, haben Ähnlichkeit miteinander. Beide brauchen zur Datenverarbeitung Daten und Programme. - Gibt es überhaupt noch einen wesentlichen Unterschied zwischen Mensch und Computer? Kann eim Computer etwa denken? Oder ist der Mensch nichts weiter als ein Computer?
Ehe wir diese schwierige Frage beantworten können, müssen wir uns darüber im klaren sein, was wir unter "Denken" verstehen wollen. Wir können die Frage kurz beantworten, ohne in philosophische Schwierigkeiten zu geraten. Wenn wir die Fähigkeit, Verknüpfungen im Sinne der Datenverarbeitung

durchzuführen, als "Denken" bezeichnen wollen, so können wir jedem komplexen Datenverarbeitungssystem (Mensch, Tier und Computer) die Fähigkeit zu denken zugestehen. Dabei abstrahieren wir allerdings vom Bewußtsein. Bewußtsein aber als die Tatsache, von sich selbst zu wissen, zu wollen, zu empfinden und einen Spielraum zu freier Entscheidung zu haben, kann kein Computer für sich in Anspruch nehmen. Auch nicht das gigantischste künstliche Supergehirn der Zukunft. Wenn wir also Denken als bewußtes Denken, als Bewußtsein von sich selbst und als Fähigkeit zu empfinden und zu wollen, auffassen, kann kein Computer denken. Und ganz abgesehen davon: Ihr eigenes Gehirn ist allein rein schaltungstechnisch komplizierter als sämtliche heute in der Welt arbeitenden Computer zusammen.

4.4. Der Philips Computer-Lehrbaukasten CL 1601

Hier liegen im Prinzip die gleichen Zusammenhänge vor wie beim großen Computer. Bei Ihrem CL 1601 erfolgt die Dateneingabe über die sechs Eingabeschalter. Die Dateneingabe erfolgt direkt, d.h. die Daten werden nicht auf irgendwelchen Belegen eingegeben.
Die Datenausgabe erfolgt über die Lampen der Logik-Bausteine. Diese Art der Datenausgabe ist vergleichbar mit Datenausgabe durch optische Anzeige (ebenfalls Lampen oder Bildschirm) bei großen Computern.
Zwischen Dateneingabe und Datenausgabe liegt die eigentliche Datenverarbeitung. Sie erfolgt beim CL 1601, wie bei den großen Computern, nach der entsprechenden Programmierung in den IC's (Integrierten Schaltkreisen) der Logik-Bausteine. Einen Teil der Steuerwerk- und Rechenwerk-Funktionen können Sie selbst durch entsprechende Schaltvorgänge an den Eingabeschaltern übernehmen.
Schließlich bliebe noch die externe Datenspeicherung zu erwähnen. Selbstverständlich kann die externe Datenspeicherung bei dem CL 1601 nicht so komfortabel vor sich gehen wie bei großen Computern. Beim CL 1601 übernehmen Sie selbst die externe Datenspeicherung, wenn Sie z.B. durch die Anzeigelampen ausgegebene Ergebnisse auf einem Blatt Papier notieren, also extern speichern, um sie für einen neuen Datenverarbeitungsvorgang wieder über die Eingabeschalter einzugeben.
Ähnlich wie ein großer Computer, so braucht auch Ihr CL 1601 sein Programm. Das Programm geben Sie ein, indem Sie die entsprechende Verdrahtung vornehmen. Die Programmierung der Logik-Bausteine des CL 1601 legt jeweils bestimmte logische Beziehungen fest, die eben so lange bestehen

bleiben wie die Verdrahtung beibehalten wird. Ein Programm für einen großen Computer kann und muß selbstverständlich mehr vollbringen: Das Programm gibt dem Computer eine Abfolge einzelner Befehle - das können mehrere hundert oder tausend sein - nach denen der Computer vollautomatisch die anstehenden Daten verarbeitet. Mit dem CL 1601 können Sie solche umfangreichen Aufgaben nicht ausführen, aber Sie lernen im Spiel und im Experiment Computer-Logik und Computer-Arithmetik kennen.



5. Logische Funktionen mit zwei Eingängen

5.1. Das Mantel-Problem

An einem alltäglichen Entscheidungsproblem soll einmal gezeigt werden, wie viele Möglichkeiten es gibt, Elemente einer logischen Funktion miteinander zu verbinden. Das "Problem" ist folgendes: Sie möchten spazierengehen und nach Möglichkeit den Mantel zu Hause lassen. Allerdings wollen Sie weder frieren noch naß werden. Sie machen also Ihre Entscheidung, den Mantel zu Hause zu lassen, von zwei Bedingungen abhängig: Die Wettervorhersage muß gut sein, und es muß wärmer als 10° sein. Mit anderen Worten:

Wenn die Wettervorhersage gut ist und wenn die Temperatur höher als 10° C ist, dann lasse ich meinen Mantel zu Hause.

Bezeichnen wir die Aussage "Die Wettervorhersage ist gut" mit A und die Aussage "Die Temperatur ist höher als 10° C" mit B und die Schlußfolgerung "Ich lasse meinen Mantel zu Hause" mit F. Dann läßt sich die Aussage abgekürzt so schreiben:

Wenn A und wenn B, dann F.

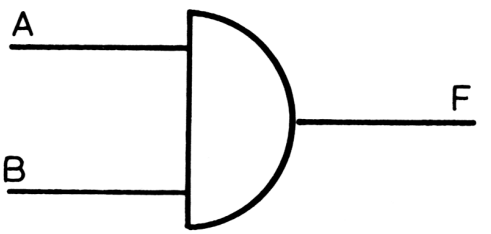
Oder noch kürzer: Wenn A und B, dann F.

Sie erkennen, daß es sich hier um eine UND-Funktion handelt. Hier die Funktionstabelle - einmal ausführlich mit Text, einmal in Kurzform - und das Schaltsymbol:

A B F

Wettervorhersage gut?	Temperatur über 10°C?	Mantel zu Hause lassen?
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



Mit einem Logik-Baustein läßt sich die UND-Funktion so programmieren:

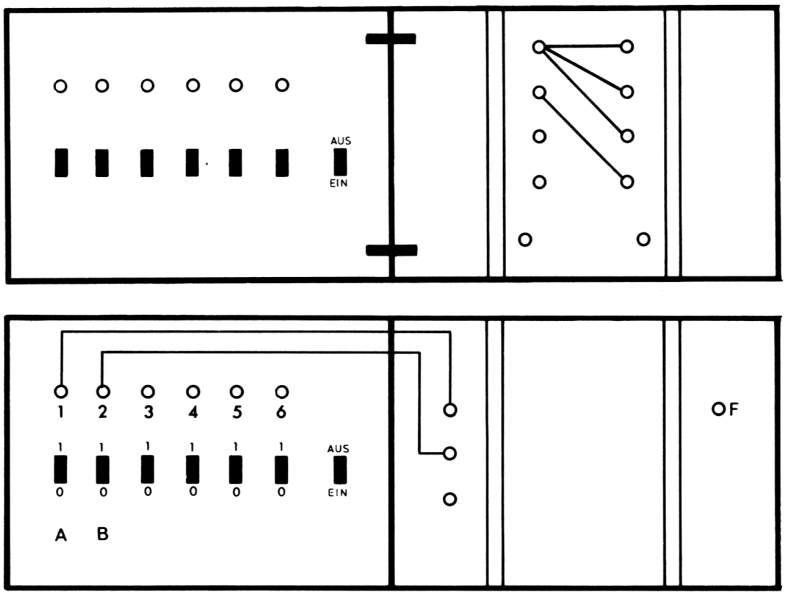


Abb.35

Formulieren wir die Zusammenhänge jetzt einmal anders: Wenn das Wetter nicht gut ist oder wenn die Temperatur nicht über 10° C ist, dann lasse ich meinen Mantel nicht zu Hause.

Im Grunde genommen haben wir die gleichen Zusammenhänge, nur haben wir die einzelnen Elemente der Aussage umgekehrt (negiert) und erhalten damit eine ODER-Funktion. Die Umkehrung bzw. die Verneinung der einzelnen Aussageelemente wollen wir folgendermaßen kennzeichnen:

Wettervorhersage nicht gut: $\bar{A}$

Temperatur nicht über 10° C: $\bar{B}$

Ich lasse den Mantel nicht zu Hause: $\bar{F}$

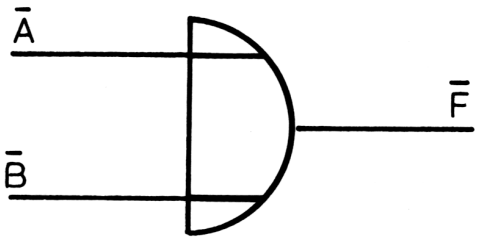
Der Querstrich über den Eingangs- bzw. Ausgangs-Symbolen deutet an, daß das Signal negiert (verneint, umgekehrt) ist. Man liest z.B. $\bar{A}$ = A quer.

Die Funktionstabellen und das Schaltsymbol:

$\bar{A}$, $\bar{B}$, $\bar{F}$

Wettervorhersage nicht gut?	Temperatur nicht über 10° C?	Mantel nicht zu Hause lassen?
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

$\bar{A}$	$\bar{B}$	$\bar{F}$
0	0	0
0	1	1
1	0	1
1	1	1



Um keine Mißverständnisse aufkommen zu lassen, müssen wir noch erwähnen, wie 0 und 1 bzw. im Falle von Fragestellungen NEIN und JA hier zu verstehen sind. Wenn wir z.B. fragen: "Ist die Wettervorhersage nicht gut?" und wir antworten darauf JA bzw. 1, so soll das heißen: "Ja, es stimmt, die Wettervorhersage ist nicht gut". Beantworten wir dageben die Frage mit NEIN oder 0, dann heißt das: "Nein, es stimmt nicht, die Wettervorhersage ist doch gut." Die dazugehörige Programmierung sieht so aus:

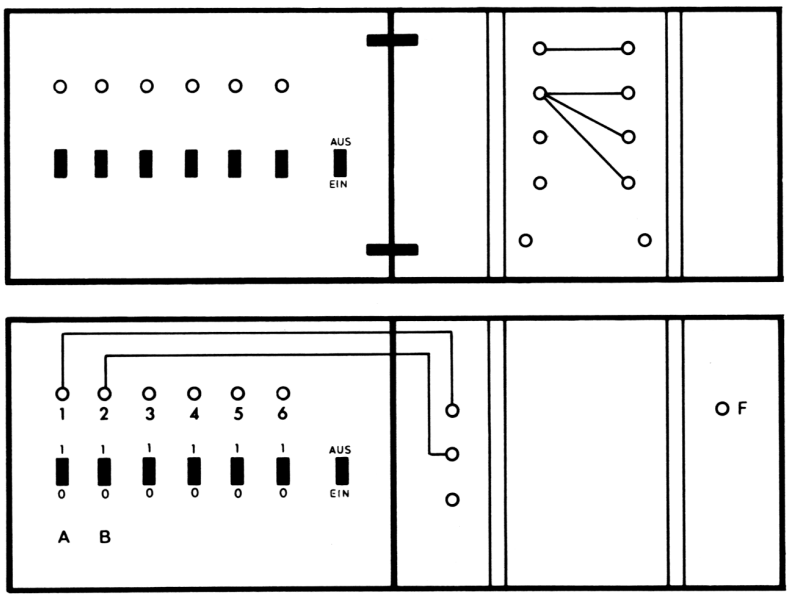


Abb.36

Wir haben einen wichtigen Grundsatz der Logik kennengelernt: Eine UND-Funktion entspricht einer ODER-Funktion, wenn die einzelnen Elemente der UND-Funktion negiert, also verneint werden.

Bevor wir weitere Kombinationsmöglichkeiten zu diesem Beispiel durchdenken, noch einige neue Begriffe:
Eine logische Funktion hat als Elemente Bedingungen und ein Ereignis.
Im Falle unseres Mantel-Problems sind die Bedingungen die gute Wettervorhersage und die Mindesttemperatur; das Ereignis ist, ob ich meinen Mantel mitnehme.
Wenn wir von der sprachlichen Ausdrucksmöglichkeit für eine logische Funktion absehen und uns eine Schaltung bzw. das entsprechende Schalt-

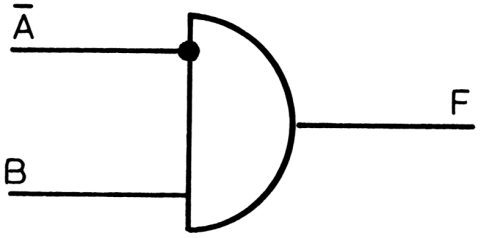
symbol vornehmen, dann sprechen wir nicht von Bedingungen und Ereignis, sondern von Eingängen und Ausgang. Eingänge und Ausgang sind veränderliche, also variable Größen. Sie können die Werte 0 und 1 haben. Eine veränderliche Größe wird kurz auch eine Variable genannt. Was wir eben als Bedingungen bezeichnet haben, können wir nunmehr als Eingangsvariable bezeichnen; was wir als Ereignis gekennzeichnet haben, stellt eine Ausgangsvariable dar. Diese Ausdrucksweise können wir nun ein für allemal verwenden, ganz gleich ob es sich bei einer logischen Funktion um eine sprachliche Darstellung oder um eine elektrische oder elektronische Schaltung handelt.

Die vorhin gewonnene Erkenntnis über den Zusammenhang zwischen UND-Funktion und ODER-Funktion drücken wir mit den neuen Bezeichnungen nun so aus: Eine UND-Funktion entspricht einer ODER-Funktion der negierten Variablen.
Das klingt reichlich abstrakt, läßt sich aber insbesondere im Zusammenhang mit unserem konkreten Beispiel des Mantel-Problems durchaus begreifen. Die Eingangsvariablen und die Ausgangsvariable unseres Mantelproblems und deren Negationen (Verneinungen, Umkehrungen) lassen sich beliebig miteinander kombinieren. Es gibt insgesamt acht Kombinationsmöglichkeiten, von denen wir zwei schon kennengelernt haben, nämlich: A, B, F und $\bar{A}$, $\bar{B}$, $\bar{F}$. Streifen wir kurz auch noch die anderen Möglichkeiten:

$\bar{A}$, B, F

Wettervorhersage nicht gut?	Temperatur über 10° C?	Mantel zu Hause lassen?
A	B	F
0	0	0
0	1	1
1	0	0
1	1	0

$\bar{A}$	B	F
0	0	0
0	1	1
1	0	0
1	1	0



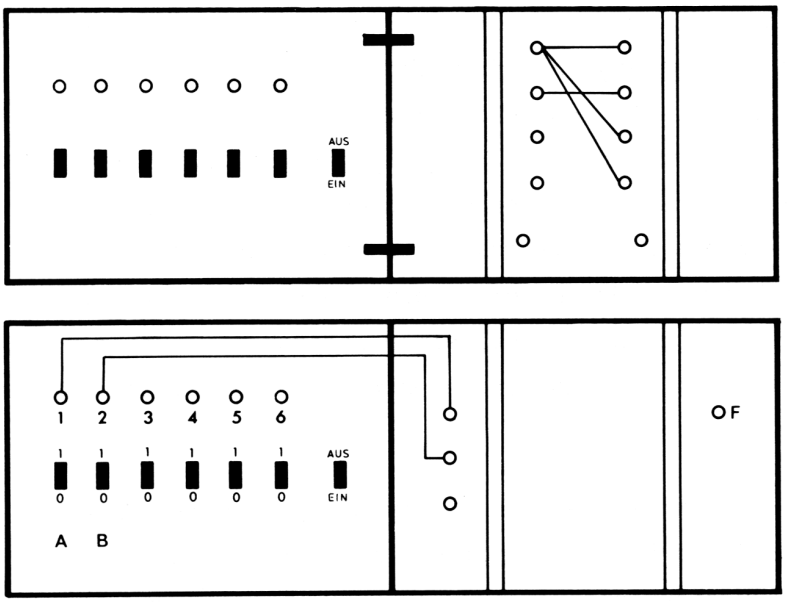


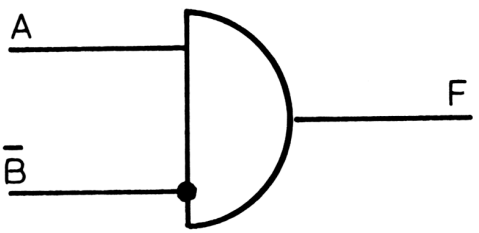
Abb.37

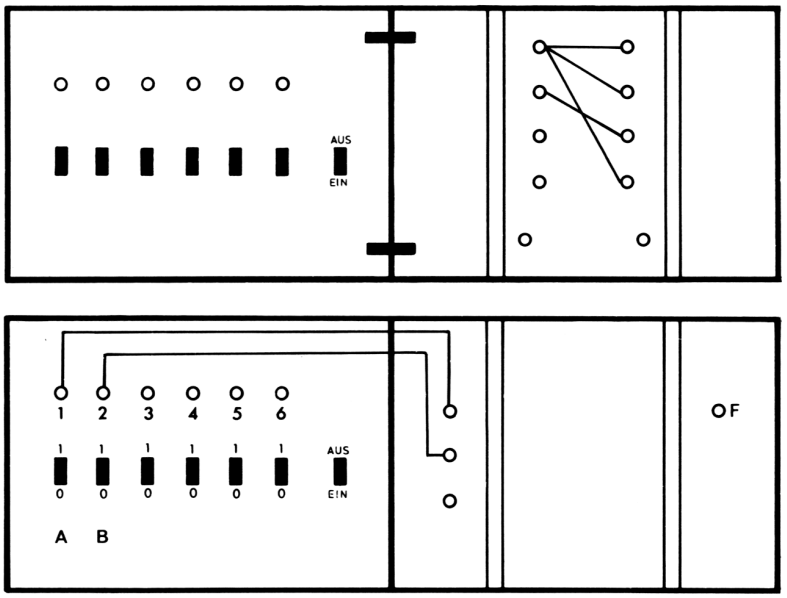
Es handelt sich hier um eine UND-Funktion mit einem negierten Eingang.

A, $\bar{B}$, F

Wettervorhersage gut?	Temperatur nicht über 10°C?	Mantel zu Hause lassen?
A	$\bar{B}$	F
0	0	0
0	1	0
1	0	1
1	1	0

A	$\bar{B}$	F
0	0	0
0	1	0
1	0	1
1	1	0





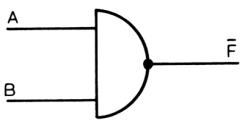
Auch dies ist eine <u>UND-Funktion mit einem negierten Eingang</u>.

Abb.38

A, B, $\bar{F}$

Wettervorhersage gut?	Temperatur über 10°C?	Mantel nicht zu Hause lassen?
A	B	$\bar{F}$
0	0	I
0	1	1
1	0	1
1	1	0

A	B	$\bar{F}$
0	0	1
0	1	1
1	0	1
1	1	0



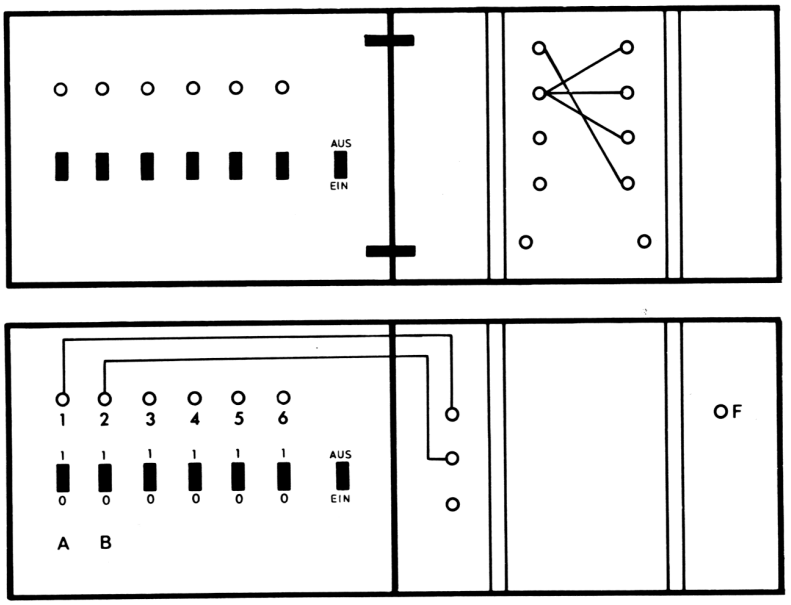


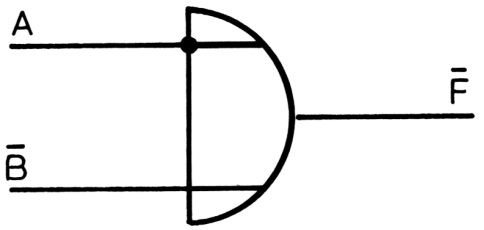
Abb.39

Diese logische Funktion hat den eigenartigen Namen NAND-Funktion. NAND ist eine Abkürzung von "not and" (englisch), was zu deutsch "nicht und" heißt. Eine NAND-Funktion kann man also als eine UND-Funktion mit negiertem Ausgang betrachten.

$A, \bar{B}, \bar{F}$

Wettervorhersage gut?	Temperatur nicht über 10° C?	Mantel nicht zu Hause lassen?
A	$\bar{B}$	$\bar{F}$
0	0	1
0	1	1
1	0	0
1	1	1

A	$\bar{B}$	$\bar{F}$
0	0	1
0	1	1
1	0	0
1	1	1



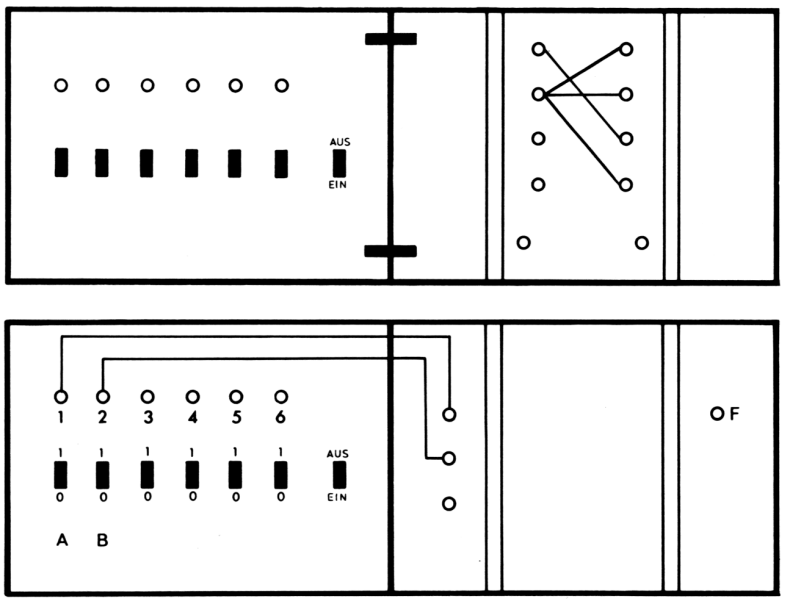


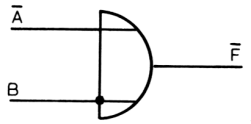
Abb. 4o

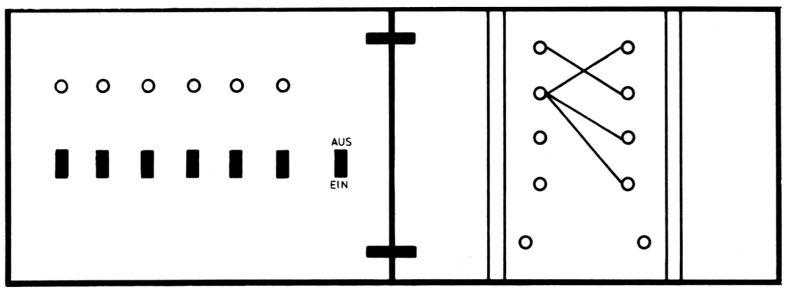
Hier haben wir es mit einer ODER-Funktion mit einem negierten Eingang zu tun.

$\bar{A}$, B, $\bar{F}$

Wettervorhersage nicht gut?	Temperatur über 10° C?	Mantel nicht zu Hause lassen?
$\bar{A}$	B	$\bar{F}$
0	0	1
0	1	0
1	0	1
1	1	1

$\bar{A}$	B	$\bar{F}$
0	0	1
0	1	0
1	0	1
1	1	1





Dies ist ebenfalls eine ODER-Funktion mit einem negierten Eingang.

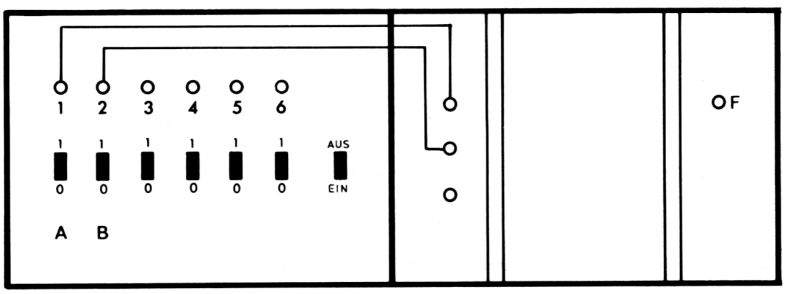
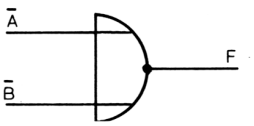


Abb.41

$\bar{A}$, $\bar{B}$, F

Wettervorhersage nicht gut?	Temperatur nicht über 10° C?	Mantel zu Hause lassen?
$\bar{A}$	$\bar{B}$	F
0	0	1
0	1	0
1	0	0
1	1	0

$\bar{A}$	$\bar{B}$	F
0	0	1
0	1	0
1	0	0
1	1	0



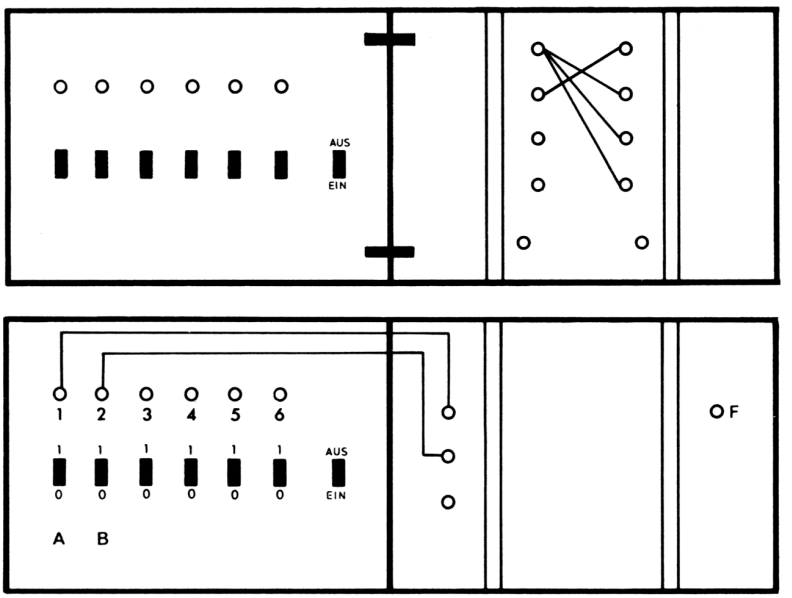


Abb.42

Dies ist das Gegenstück zur NAND-Funktion: die NOR-Funktion. NOR ist die Abkürzung von "not or" (englisch), was zu deutsch "nicht oder" heißt. Eine NOR-Funktion ist also eine ODER-Funktion mit negiertem Ausgang.

Durch Negieren einer oder mehrerer der Variablen wurde aus unserer UND-Funktion eine UND-Funktion mit einem negierten Eingang, eine NAND-Funktion, eine ODER-Funktion, eine ODER Funktion mit einem negierten Eingang sowie eine NOR-Funktion. Von solchen Umformungen macht man in der Computer-Technik häufig Gebrauch, wenn man mit wenig logischen Verknüpfungselementen möglichst vielfältige Schaltungen verwirklichen will.

In diesem Sinne ist auch der Aufbau der Logik-Bausteine Ihres CL 1601 zu verstehen. Wenn Sie sich jetzt einmal den Schaltplan eines Logik-Bausteins im Kapitel ansehen, so erkennen Sie, daß sehr viele Negationen eingebaut sind. Machen Sie sich nichts daraus, wenn Sie den Schaltplan nicht ganz durchschauen. Nur so viel sei gesagt: Die Logik-Bausteine des CL 1601 sind so konstruiert - bei Verwendung von standardisierten integrierten Schaltkreisen, die auch im Groß-Computer Verwendung finden - daß möglichst viele Programme zu verwirklichen sind.

6. Logische Funktionen mit 3 Eingängen

Wir haben bisher logische Funktionen und logische Schaltungen mit zwei Eingängen kennengelernt:

UND-Funktion	bzw.	UND-Schaltung
ODER-Funktion	"	ODER-Schaltung
Exclusiv-ODER-Funktion	"	Exclusiv-ODER-Funktion
Äquivalenz-Funktion	"	Äquivalenz-Schaltung
NAND-Funktion	"	NAND-Schaltung
NOR-Funktion	"	NOR-Schaltung

Die Identitäts-Funktion bzw. Identitäts-Schaltung sowie die Negations-Funktion bzw. Negations-Schaltung hatten jeweils nur einen Eingang.

6.1. UND-Funktion

Die genannten Funktionen bzw. Schaltungen können - mit Ausnahme der Identitäts- und Negations-Funktion - auch mit mehr als zwei Eingängen vorkommen. Im folgenden geben wir Ihnen eine Übersicht über logische Funktionen mit drei Eingängen.

Selbstverständlich sind auch hierzu sprachliche Beispiele denkbar, wie z.B. für die UND-Funktion mit 3 Eingängen: Wenn ich durstig bin und wenn ich Geld bei mir habe und wenn ich an einem Wirtshaus vorbeikomme, dann trinke ich Bier.

Die Ausgangsvariable F kann also nur den Wert 1 annehmen, wenn alle drei Eingangsvariablen ebenfalls den Wert 1 haben. Auf unser Beispiel übertragen bedeutet das, daß ich nur Bier trinken kann (F = 1), wenn ich durstig bin (A = 1) und Geld besitze (B = 1) und an einem Wirtshaus vorbeikomme (C = 1). Nimmt nur eine der Eingangsvariablen den Wert 0 an, so muß die Ausgangsvarialbe F ebenfalls 0 sein.

Bei drei variablen Eingängen nimmt die Zahl der Möglichkeiten zu, und die Funktionstabelle muß diese größere Zahl auch ausdrücken.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Programmieren Sie Ihren Logik-Baustein und verbinden Sie ihn wie auf der folgenden Abbildung mit der Eingabeeinheit.

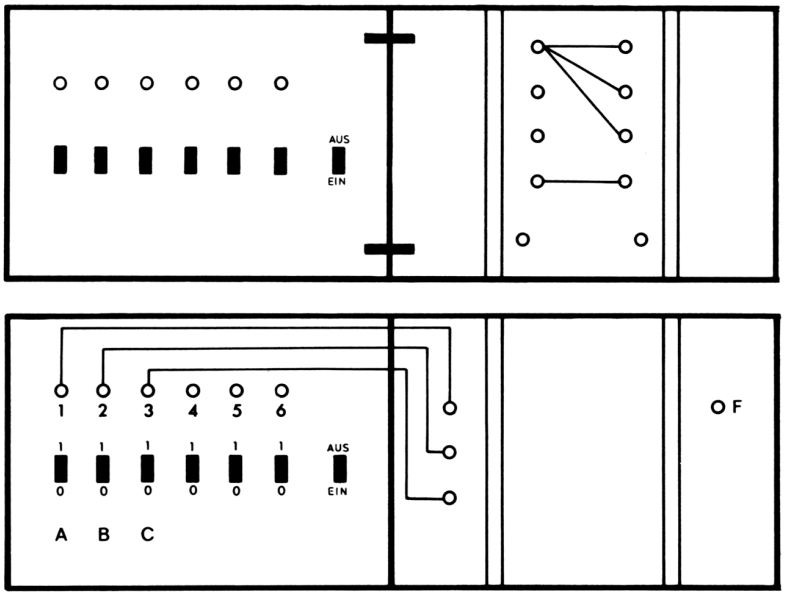
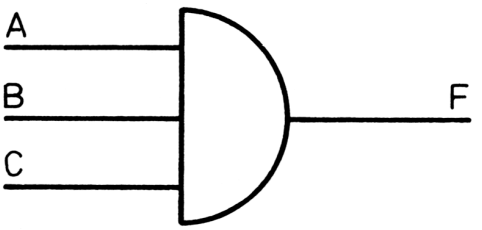


Abb.43

Probieren Sie alle in der Funktionstabelle angegebenen Möglichkeiten und tragen Sie die angezeigten Werte ein.

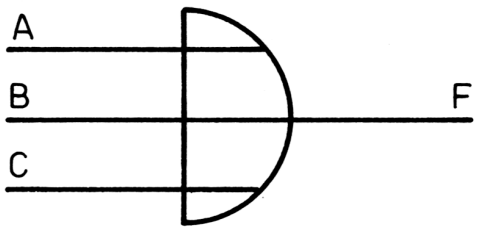
Schalter A —— Variable A
Schalter B —— Variable B
Schalter C —— Variable C

Auch für eine UND-Funktion mit drei Eingängen gibt es ein Schaltsymbol:



6.2. Einschließende ODER-Funktion

Symbol ODER-Funktion:



Sie möchten einem Freund eine kurze schriftliche Mitteilung zukommen lassen. Sie zücken also Ihren Federhalter und suchen Schreibpapier. Haben Sie eine einfache Postkarte oder eine Ansichtskarte oder Briefpapier gefunden, so können Sie die Mitteilung zu Papier bringen. Das

ist natürlich auch möglich, wenn Sie z.B. Briefpapier und eine Ansichtskarte zur Hand haben. Ist dagegen nichts greifbar, dann müssen Sie sich erst Papier beschaffen.
Für die Funktionstabelle bedeutet das: Hat mindestens eine Eingangsvariable den Wert 1, dann ist F auch 1. Versuchen Sie doch wieder einmal, die Funktionstabelle für dieses Beispiel aufzustellen. Dazu noch ein Hinweis: Bei drei Eingangsvariablen existieren immer acht mögliche Kombinationen.

A	B	C	F

Wenn Sie nach der folgenden Abbildung verdrahten, können Sie überprüfen, ob die Funktionstabelle richtig aufgestellt wurde.

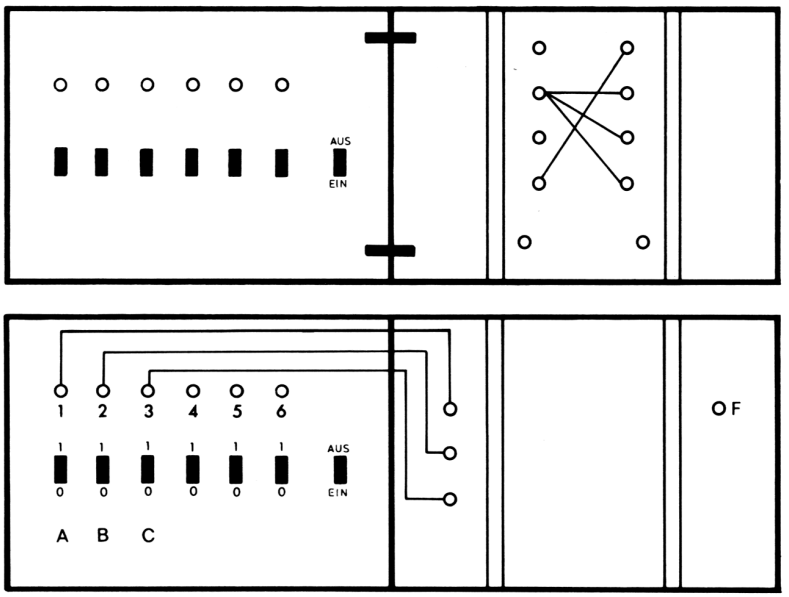
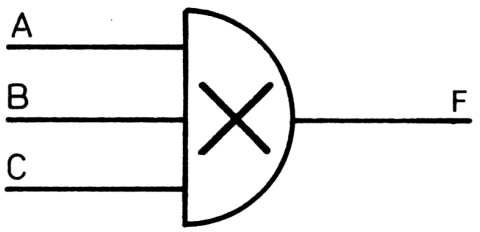


Abb.44

6.3. Exklusiv-ODER-Funktion

Symbol Exklusiv-ODER-Funktion:



Als Bastler stehen Sie vor dem Problem, ein Gehäuse für ein elektronisches Gerät herzustellen. Drei verschiedene Materialien stehen Ihnen dafür zur Verfügung: Holz, Metall oder Kunststoff.
Zu diesem Problem soll zuerst Ihr Computer in Betrieb gesetzt werden, und anschließend wenden wir uns der sprachlichen Lösung zu.

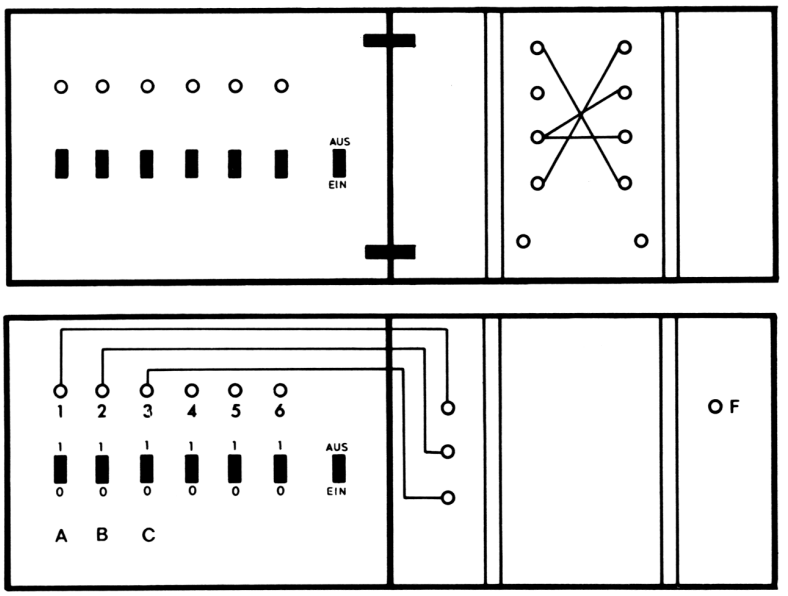


Abb.45

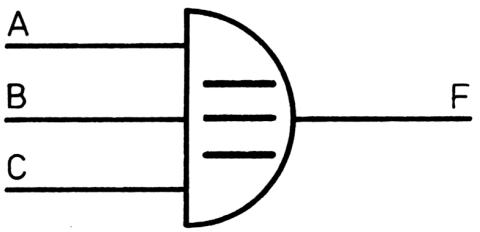
Tragen Sie bitte auch die Werte in die Funktionstabelle ein.

A	B	C	F

Die Auswertung der Funktionstabelle ist nun sprachlich kein Problem mehr. Nur wenn höchstens eine der Eingangsvariablen den Wert 1 erhält, ist auch der Ausgang 1. Für Sie als Bastler bedeutet das, Sie bekommen Ihr Gehäuse nur dann fertiggestellt, wenn Sie entweder Holz oder Metall oder Kunststoff verwenden. Zwei oder drei Materialien können Sie nicht verwenden.

6.4. Äquivalenz-Funktion

Symbol Äquivalenz-Funktion:



"Alles oder nichts" - das ist die einfachste Erklärung für eine Äquivalenz-Funktion mit drei Eingängen.
Wie Sie der Funktionstabelle entnehmen und mit Ihrem Computer überprüfen können, ist der Ausgang F nur dann 1, wenn alle Eingangsvariablen den Wert 0 oder 1 besitzen. Bei den übrigen sechs Möglichkeiten ist der Ausgang 0.

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

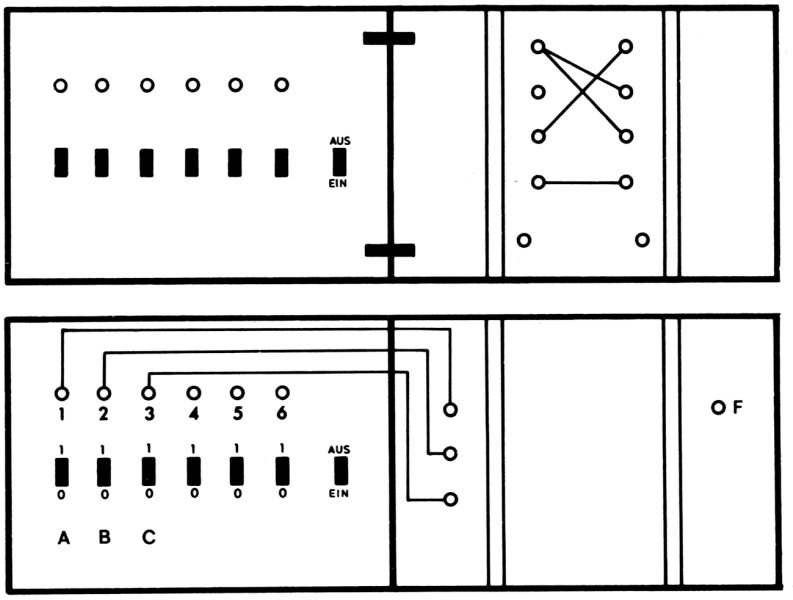
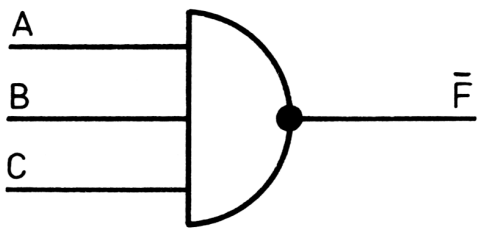


Abb.46

6.5. NAND-Funktion

Symbol NAND-Funktion:



Den Besuch eines großen internationalen Fußballspiels machen Sie von drei Bedingungen abhängig:

1. Das Wetter ist gut (A).
2. Ein Freund kommt mit (B).
3. Es sind noch Eintrittskarten vorhanden (C).

Sie nehmen sich fest vor, nicht hinzugehen, wenn nicht alle drei Bedingungen erfüllt sind.

Bevor Sie die Funktionstabelle betrachten und überprüfen, überlegen Sie einmal die Bedeutung der Werte 0 und 1: Ist eine Bedingung erfüllt, gilt 1, ist sie nicht erfüllt, bedeutet das 0. Bei der Ausgangsvariablen F ist das umgekehrt: Nicht zum Fußball gehen, bedeutet 1, hingehen erhält den Wert 0 - heißt also JA. Denn die Frage wurde in der Verneinungsform gestellt, und ein NEIN (0) als Antwort darauf - doppelte Nerneinung - heißt JA.

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Überprüfen Sie, ob Ihr Computer zu demselben Ergebnis gelangt.

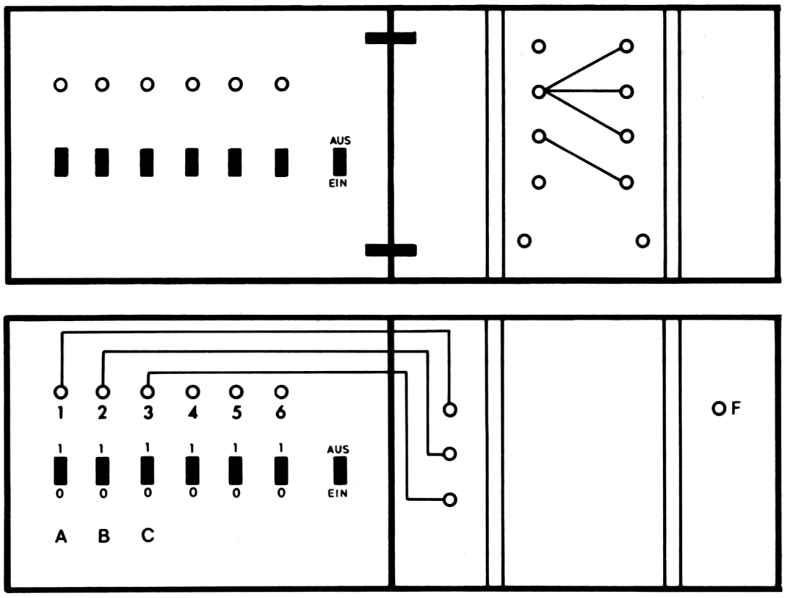
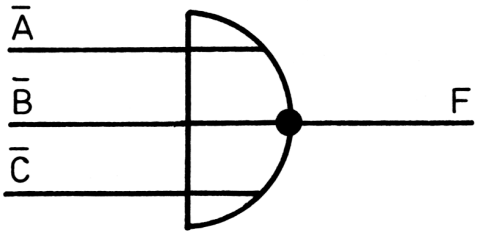


Abb.47

6.6. NOR-Funktion

Symbol NOR-Funktion:



Vielleicht mögen Sie gern mit einem kleinen Boot fahren, vielleicht auch nicht. Nehmen wir einmal an, das letztere sei der Fall, und Sie haben eine deutliche Abneigung gegen Bootsfahrten. Von einem Freund, der das nicht weiß, werden Sie dazu eingeladen für das nächste Wochenende. Sie machen eine Zusage von drei Bedingungen abhängig:

1. Es darf nicht regnen ($\bar{A}$).
2. Es darf nicht windig sein ($\bar{B}$).
3. Sie wollen nicht weit vom Ufer entfernt fahren ($\bar{C}$).

Wenn alle negierten Bedingungen mit NEIN erfüllt sind (0) - dadurch hebt sich die doppelte Verneinung wieder auf - sagen Sie JA zur Bootsfahrt (I).

$\bar{A}$	$\bar{B}$	$\bar{C}$	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Die Funktionstabelle macht es deutlich: Wenn es nicht regnet oder nicht stürmt oder nicht zu weit vom Ufer entfernt gefahren wird, dann erteilen Sie eine Zusage.

Nach der folgenden Schaltung können Sie die Tabelle überprüfen. Die Lampe leuchtet nur (F = 1), wenn $\bar{A}$ oder $\bar{B}$ oder $\bar{C}$ nicht 1 sind.

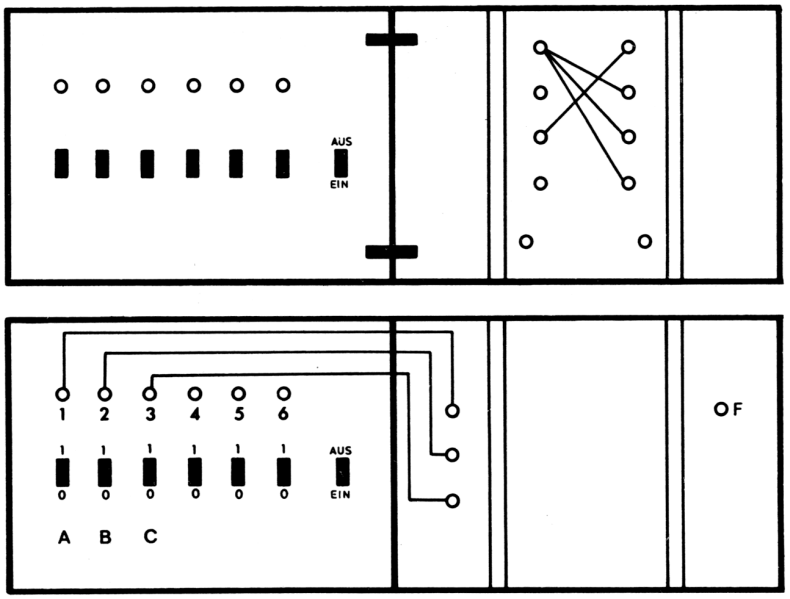
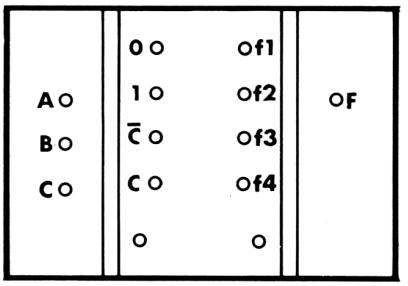


Abb.48

7. Programmierung der Logik-Bausteine

Sie haben eine Menge logischer Funktionen kennengelernt. Nun ist es an der Zeit, daß wir Ihnen das Schema erklären, nach dem Sie die Programmierung der Logik-Bausteine, also die Verdrahtung des Programmierfeldes, selbst ableiten können. Dazu wollen wir die einzelnen Buchsen des Programmierfeldes mit Bezeichnungen versehen, wie in der folgenden Abbildung aufgeführt:



Die Buchsen f1 bis f4 sind in bestimmter Weise - wie aus den folgenden Schemata zu ersehen ist - den Wertekombinationen der Eingangsvariablen von logischen Funktionen zugeordnet. Haben wir es beispielsweise mit einer logischen Funktion mit drei Eingangsvariablen zu tun, so ist die Buchse f1 den ersten beiden Zeilen der Funktionstabelle, also mit den Wertekombinationen 000 und 001, zugeordnet. Je nach dem, welche Ausgangswerte bei F entsprechend der logischen Funktion anliegen, ist die Buchse f1 mit der Buchse 0 bzw. C, $\bar{C}$ oder 1 zu verbinden.

Programmierung mit den Eingangsvariablen A, B und C

Programmierfeldeingänge (Zuordnung)	Eingänge A	B	C	Ausgang F
f1	0	0	0	
	0	0	1	
f2	0	1	0	
	0	1	1	
f3	1	0	0	
	1	0	1	
f4	1	1	0	
	1	1	1	

Verbinden Sie den Programmiereingang f .... mit ...., wenn der Ausgang F .... ist.

0	C	$\bar{C}$	1
0	0	1	1
0	1	0	1

Angenommen, Sie wollen das Wissen eines Bekannten testen und das Ergebnis mit einem Logik-Baustein anzeigen. Sie stellen die Frage: "Was hat Thomas A. Edison erfunden?" Ihr Bekannter kann jetzt eine lange Reihe von Erfindungen aufzählen, die Sie jedoch nicht mit dem Logik-Baustein

verarbeiten können. So stellen Sie Ihre allgemeine Frage etwas differenzierter und unterteilen sie:

1. Teilfrage: (A) "Ist Thomas A. Edison der Erfinder des Phonografen?"
2. " : (B) "Ist Thomas A. Edison der Erfinder des Tonbandgerätes?"
3. " : (C) "Ist Thomas A. Edison der Erfinder der Kohlefadenlampe?"

Auf diese Fragen kann nur mit JA oder NEIN geantwortet werden. Ferner stellen Sie die Forderung, daß nur bei 3 richtigen Antworten die Lampe des Logik-Bausteins aufleuchten soll. Jetzt müssen Sie Ihren Logik-Baustein programmieren.

Ordnen Sie zunächst die drei Teilfragen in der angegebenen Reihenfolge den Eingangsvariablen A, B und C zu, und legen Sie für die möglichen Antworten JA und NEIN fest, daß der betreffende Eingabeschalter auf 1 (JA) bzw. auf 0 (NEIN) zu stellen ist. Daraus ergeben sich folgende 8 verschiedene Möglichkeiten für die Stellung der 3 Eingabeschalter.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Setzen Sie in die Spalte F überall dort eine 1 ein, wo nach der eingangs geforderten Bedingung die Lampe brennen soll.

Da die Teilfragen

(A) "Ist Thomas A. Edison der Erfinder des Phonografen?" mit JA (1)
(B) "Ist Thomas A. Edison der Erfinder des Tonbandgerätes?" mit NEIN(0)
(C) "Ist Thomas A. Edison der Erfinder der Kohlefadenlampe?" mit JA (1)

richtig beantwortet sind, müssen Sie überall dort eine 1 für F setzen, wo alle Eingangswerte mit der Lösung übereinstimmen. Sind nur eine oder zwei Fragen richtig beantwortet, schreiben Sie in die Spalte F eine 0.

Ihre vollständige Funktionstabelle muß nun so aussehen:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Aus dem Programmierschema für 3 Eingangsvariablen können Sie jetzt die Verdrahtung des Logik-Baunsteins ableiten. Da die ersten beiden Zeilen dem Programmiereingang f1 zugeordnet sind und F die Kombination 00 angenommen hat, ist f1 mit 0 zu verbinden. In den nächsten 2 Zeilen ist der Ausgang auch 00, also wird f2 wieder mit 0 verbunden. f3 (5. und 6. Zeile) ist mit C zu verbinden, da F in diesen beiden Zeilen die Werte 01 annimmt. Die Ausgangsfunktion der letzten beiden Möglichkeiten bestimmt der Programmiereingang f4; da F 00 ist, muß f4 auch mit 0 verbunden werden. Und so sieht die Verdrahtung des Logik-Bausteins aus:

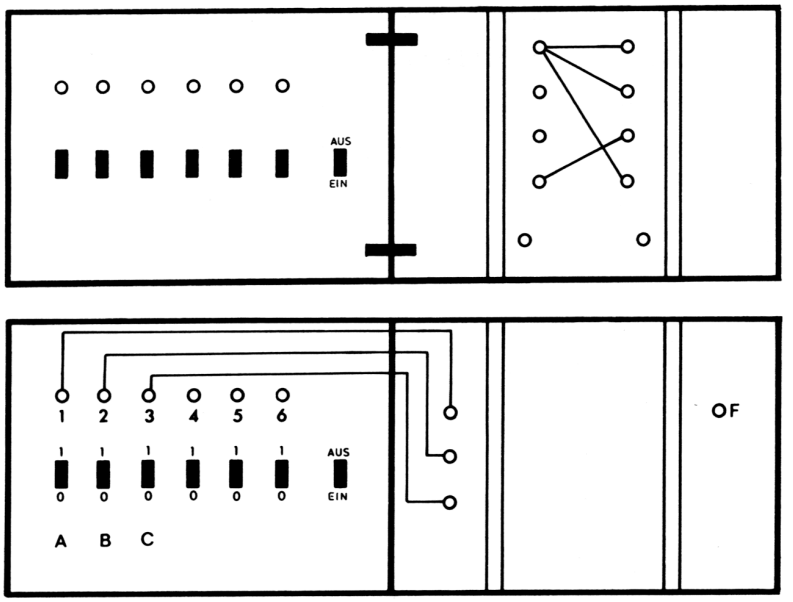


Abb.49

Jetzt können Sie Ihrem Bekannten die Fragen stellen. Er muß sein Wissen dem Logik-Baustein durch Betätigung der Eingabeschalter mitteilen, die Elektronik in dem Logik-Baustein verarbeitet nach Ihrer Programmierung die eingegebenen Informationen und zeigt 3 richtige Antworten durch das Aufleuchten der Lampe an.

Sinngemäß lassen sich auch andere Fragen, die mit JA oder NEIN beantwortet werden können, in eine Funktionstabelle verwandeln. Mit Hilfe des entsprechenden Programmierschemas, das je nach Anzahl der Eingangsvariablen und der dabei verwendeten Eingangsbuchsen von dem Grundschema für 3 Eingangsvariablen abweicht, können Sie die aufgestellte Funktionstabelle auf den Logik-Baustein übertragen. Dazu noch ein Beispiel mit 3 Eingangsvariablen:

Frage A:	"Ist der Mount Everest 8848 m hoch?"	JA (1)	NEIN (0)
Frage B:	"Ist die Zugspitze 2964 m hoch?"	JA (1)	NEIN (0)
Frage C:	"Ist der Feldberg (Schwarzw.) 1768 m hoch?"	JA (1)	NEIN (0)

Bedingung: Wenn zwei oder drei Fragen richtig beantwortet werden, soll die Lampe des Logik-Bausteins leuchten.
Richtig gelöst sind alle Fragen, wenn A = 1, B = 1 und C = 0 ist. Es ergibt sich die folgende Funktionstabelle und Programmierung:

A	B	C	F	
0	0	0	0	
0	0	1	0	→ f1 wird 0
0	1	0	1	
0	1	1	0	→ f2 wird $\bar{C}$
1	0	0	1	
1	0	1	0	→ f3 wird $\bar{C}$
1	1	0	1	
1	1	1	1	→ f4 wird 1

Die Verdrahtung von Eingabeeinheit und Logik-Baustein sieht so aus:

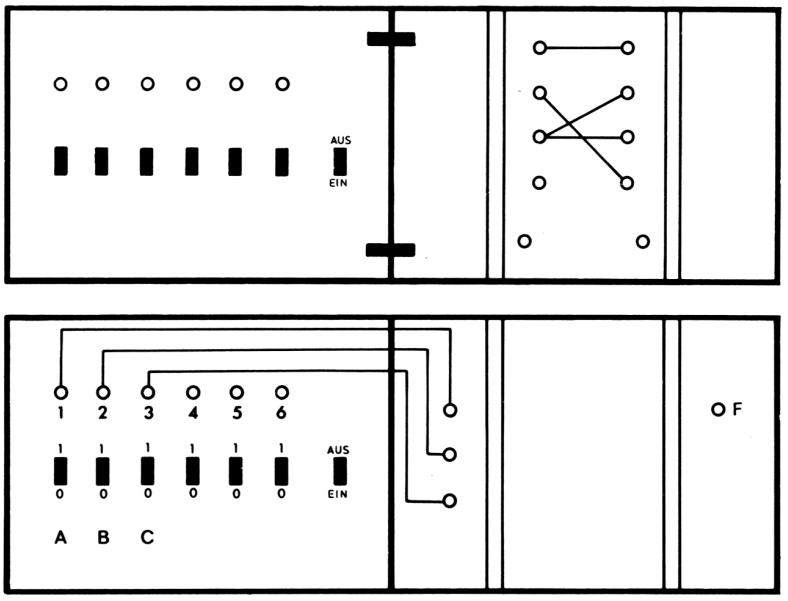


Abb.5o

Durch dieses Programmierschema sind Sie wie gesagt in der Lage, für die 3 Eingangsvariablen jede gewünschte Funktion zu programmieren. Sie können also auch z.B. die Forderung stellen, daß in der Spalte F jede dritte Ziffer 1 sein soll oder nur die zweite und die fünfte eine 0 und alle anderen 1. Dann müssen Sie wieder ermitteln, mit welchen Buchsen f1, f2, f3 und f4 zu verbinden sind.

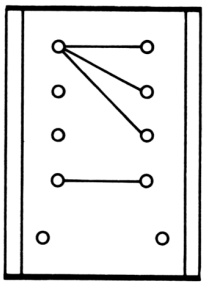
Weitere Beispiele:

UND-Funktion

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

→ f1 = 0
→ f2 = 0
→ f3 = 0
→ f4 = C

Verdrahtung

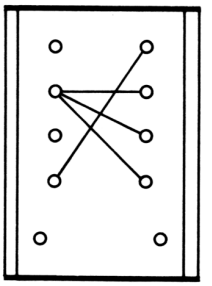


ODER-Funktion

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

→ f1 = C
→ f2 = 1
→ f3 = 1
→ f4 = 1

Verdrahtung

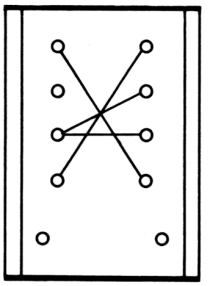


Exklusiv-ODER-Funktion

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = C
→ f2 = $\bar{C}$
→ f3 = $\bar{C}$
→ f4 = 0

Verdrahtung



Einige Seiten vorher hatten wir Ihnen eine Übersicht über die verschiedenen logischen Funktionen mit drei Eingangsvariablen gezeigt. Vielleicht versuchen Sie einmal, die Verdrahtung für die Äquivalenz-Funktion, die NAND-Funktion und die NOR-Funktion selbst herauszufinden; das Ergebnis können Sie dann später kontrollieren.

Äquivalenz-Funktion

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

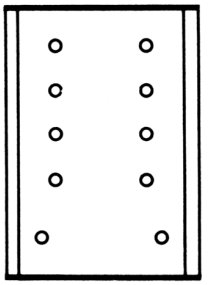
f1 =

f2 =

f3 =

f4 =

Verdrahtung



NAND-Funktion

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

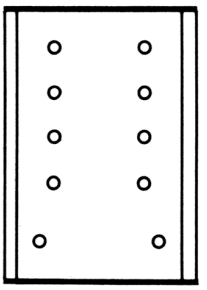
f1 =

f2 =

f3 =

f4 =

Verdrahtung



NOR-Funktion

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

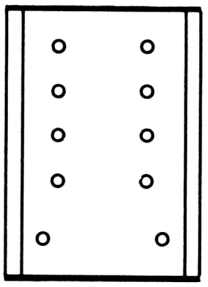
f1 =

f2 =

f3 =

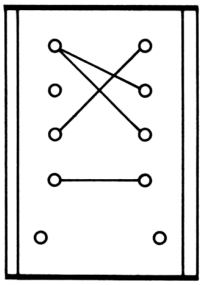
f4 =

Verdrahtung

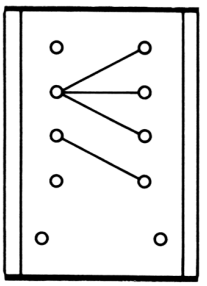


Auflösungen der vorgenannten Programmierungen:

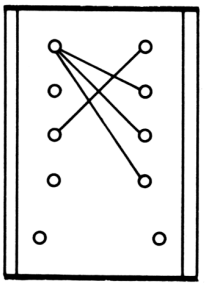
Äquivalenz-Funktion



NAND-Funktion



NOR-Funktion



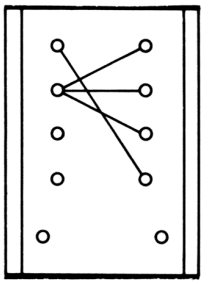
Wie bei der Verdrahtung bei logischen Funktionen mit zwei Eingangsvariablen bzw. mit einer Eingangsvariablen zu verfahren ist, ersehen Sie aus den folgenden Erläuterungen mit je einem Beispiel.

Programmierung mit den Eingangsvariablen A und B
(C bleibt frei)

Programmierfeldeingänge (Zuordnung)	Eingänge		Ausgang
	A	B	F
f1	0	0	
f2	0	1	
f3	1	0	
f4	1	1	

Verbinden Sie den Programmiereingang f .... mit ...., wenn der Ausgang F .... ist.

0	1
0	1



Beispiel:

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

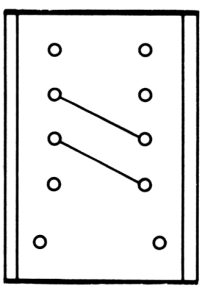
f1 = 1
f2 = 1
f3 = 1
f4 = 0

Programmierung mit den Eingangsvariablen B und C
(A bleibt frei)

Programmierfeldeingänge (Zuordnung)	Eingänge		Ausgang
	B	C	F
f3	0	0	
	0	1	
f4	1	0	
	1	1	

Verbinden Sie den Programmiereingang f .... mit ...., wenn der Ausgang F .... ist.

0	C	$\bar{C}$	1
0	0	1	1
0	1	0	1



Beispiel:

B	C	F
0	0	1
0	1	1
1	0	1
1	1	0

f3 = 1

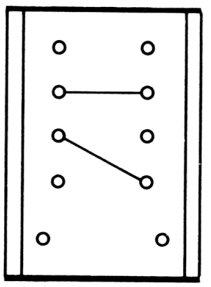
f4 = $\bar{C}$

Programmierung mit den Eingangsvariablen A und C
(B bleibt frei)

Programmier-feldeingänge (Zuordnung)	E i n g ä n g e		Ausgang
	A	C	F
f2	0	0	
	0	1	
f4	1	0	
	1	1	

Verbinden Sie den Programmiereingang f .... mit ...., wenn der Ausgang F .... ist.

0	C	$\bar{C}$	1
0	0	1	1
0	1	0	1



Beispiel:

A	C	F
0	0	1
0	1	1
1	0	1
1	1	0

f2 = 1

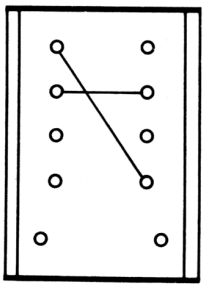
f4 = $\bar{C}$

Programmierung mit der Eingangsvariablen A
(B und C bleiben frei)

Programmier-feldeingänge (Zuordnung)	Eingang	Ausgang
	A	F
f2	0	
f4	1	

Verbinden Sie den Programmiereingang f .... mit ...., wenn der Ausgang F .... ist.

0	1
0	1



Beispiel:

A	F
0	1
1	0

f2 = 1

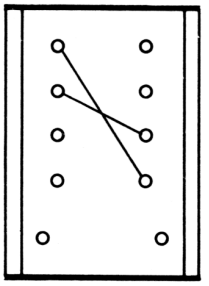
f4 = 0

Programmierung mit der Eingangsvariablen B
(A und C bleiben frei)

Programmier-feldeingänge (Zuordnung)	Eingang	Ausgang
	B	F
f3	0	
f4	1	

Verbinden Sie den Programmiereingang f .... mit ...., wenn der Ausgang F .... ist.

0	1
0	1



Beispiel:

B	F
0	1
1	0

f3 = 1

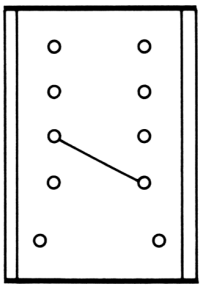
f4 = 0

Programmierung mit der Eingangsvariablen C
(A und B bleiben frei)

Programmier-feldeingang (Zuordnung)	Eingang	Ausgang
	C	F
f4	0	
	1	

Verbinden Sie den Programmiereingang f4 mit ...., wenn der Ausgang F .... ist.

C	$\bar{C}$
0	1
1	0



Beispiel:

C	F
0	1
1	0

$f4 = \bar{C}$

Stellen Sie anhand der folgenden Verdrahtungen die Funktionstabellen auf.

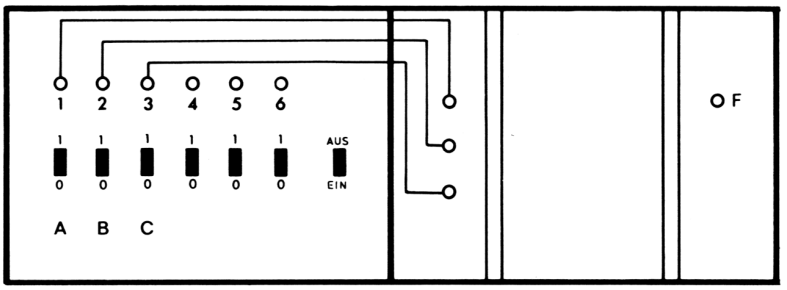


Abb.51

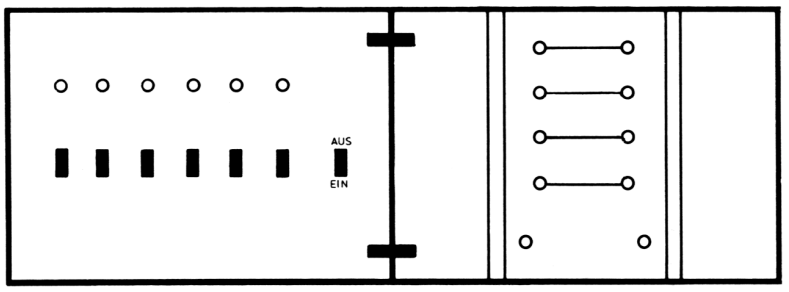


Abb.52

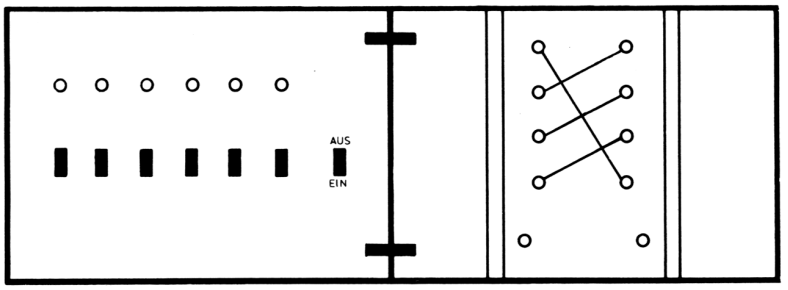
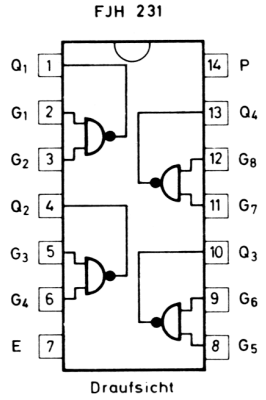


Abb.53

8. Aufbau der Logik-Bausteine

In jedem der Logik-Bausteine Ihres CL 1601 sind drei integrierte Schaltungen eingebaut. Integrierte Schaltungen, die auch in großen Computern Verwendung finden, sind Bauelemente, die in sich die verschiedensten elektronischen Einzelteile enthalten. Ein großer Vorteil der integrierten Schaltung, auch kurz IC genannt (Abkürzung von englisch "integrated circuit"), besteht neben ihren geringen Abmessungen darin, daß sich mit ihnen die unterschiedlichsten logischen Funktionen verwirklichen lassen. Sehen wir uns als Beispiel eine integrierte Schaltung aus einem Logik-Baustein näher an.
Die folgende Abbildung zeigt die Gehäuse-Anschlüsse der integrierten Schaltung FJH 231:



Wie Sie leicht erkennen können, enthält die integrierte Schaltung FJH 231 vier NAND-Schaltungen. Je nach dem wie nun die verschiedenen Anschlüsse untereinander oder mit anderen Bauteilen verbunden werden, lassen sich die unterschiedlichen logischen Schaltungen verwirklichen. Wir wollen uns im Rahmen dieser Ausführungen zwar nicht mit Elektronik beschäftigen, doch sei für besonderes Interessierte ein Beispiel einer NAND-Schaltung aus der integrierten Schaltung FJH 231 angeführt:

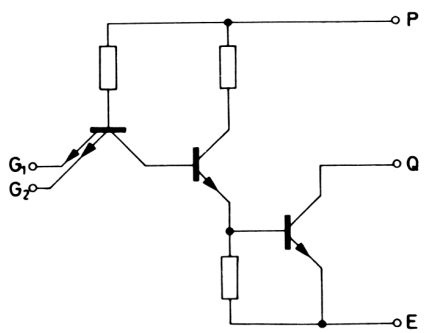
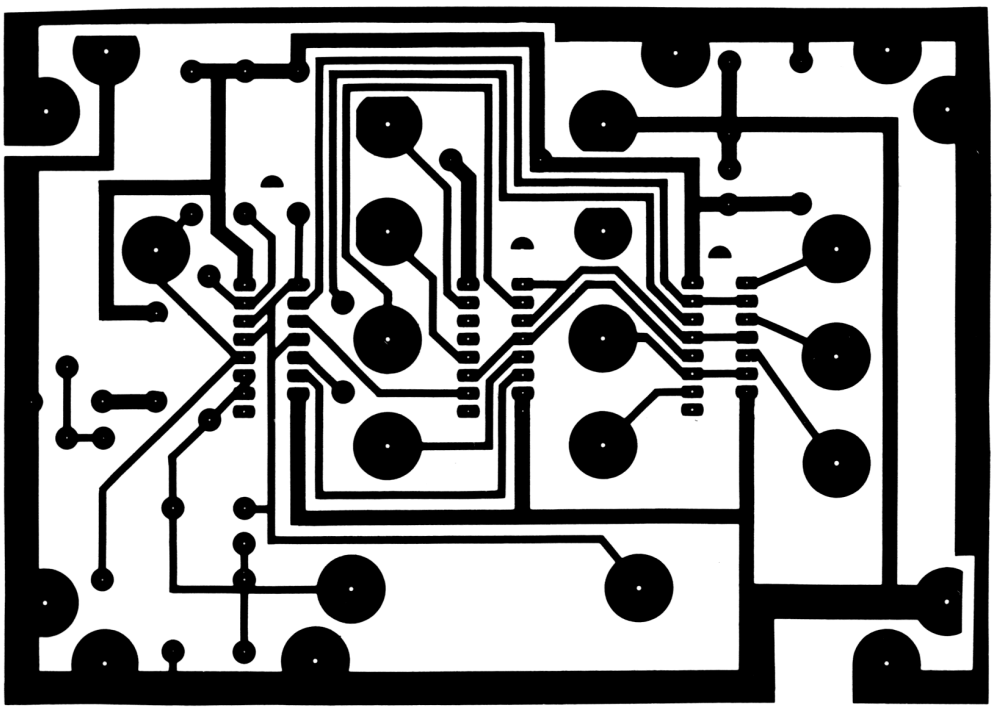


Abb. 54

Dieses NAND besteht aus mehreren Transistoren, es hat daher seinen Namen TTL-NAND = Transistor-Transistor-Logik-NAND.
Die folgende Abbildung zeigt Ihnen die gedruckte Leiterplatte, die zusammen mit den integrierten Schaltungen Hauptbestandteil der Logik-Bausteine ist.



Hier können Sie deutlich die Anschlußstellen der drei integrierten Schaltungen FJH 241, FJH 151 und FJH 231 erkennen.

Wenn Sie es ganz genau wissen wollen, beschäftigen Sie sich ein wenig mit dem folgenden logischen Schaltplan. Er zeigt Ihnen im Zusammenhang die logischen Schaltungen der drei integrierten Schaltkreise sowie die Anschlüsse zur Ein- und Ausgabe (A, B, C bzw. F) und zur Programmierung (0, 1, $\bar{C}$, C bzw. f1, f2, f3 und f4). Im rechten Teil der Abbildung erkennen Sie die vier NAND-Schaltungen der integrierten Schaltung FJH 231 wieder.

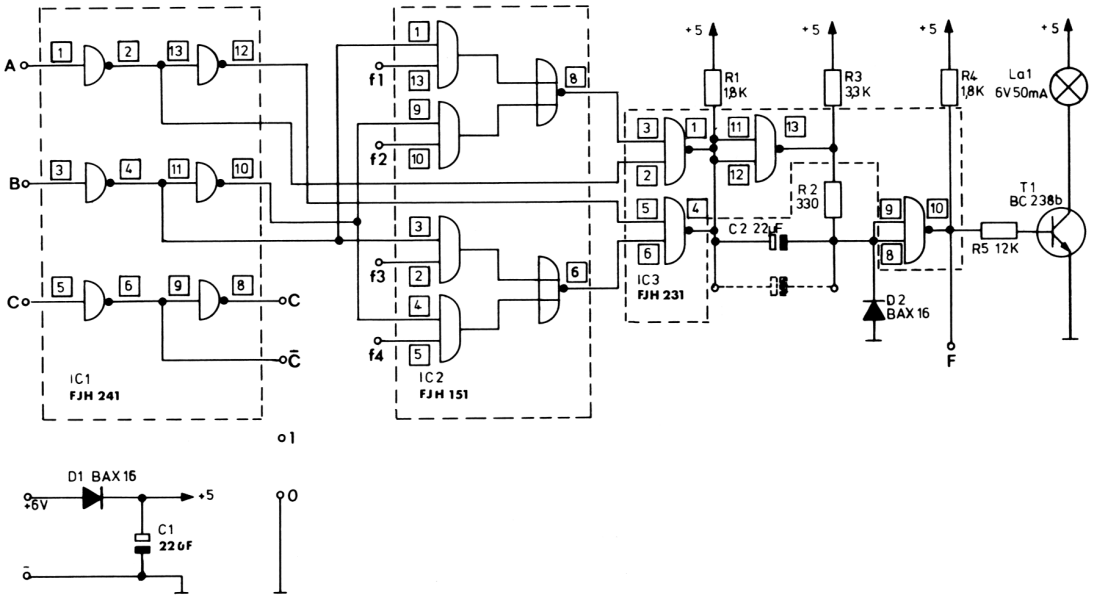


Abb.55

Sollten Sie mit der Elektronik nicht vertraut sein, so zerbrechen Sie sich deshalb nicht den Kopf, denn wie Sie gewiß schon gemerkt haben, ist die Kenntnis der Elektronik für die Durchführung unserer logischen Experimente nicht erforderlich.

9. Blinkschaltungen

Wir programmieren zunächst einen einfachen Blinkgenerator. Er besteht erstens aus einem rückgekoppelten Inverter, d.h. einer NICHT-Schaltung, deren Ausgang wiederum mit dem Eingang verbunden, also rückgekoppelt ist, und zweitens aus einer Verzögerung.

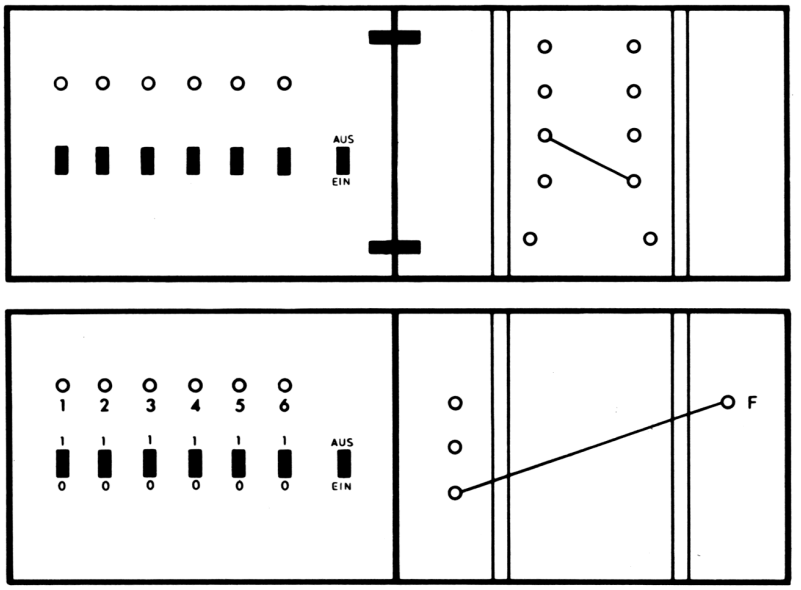


Abb.56

Betrachten Sie zunächst noch einmal die Funktionstabelle für eine Negation.

C	F
0	1
1	0

Hieraus geht hervor, daß beim Eingang C = 0 der Ausgang F = 1 ist und umgekehrt. Da in dieser Blinkschaltung der Ausgang F mit dem Eingang C rückgekoppelt ist, muß F immer C sein. Wenn also F = 1 ist, ist auch C = 1, das bedeutet aber nach der Funktionstabelle, daß beim Eingang C = 1 der Ausgang F = 0 wird. F = 0 bedeutet wiederum, Eingang C = 0, das bedeutet, daß der Ausgang nun wieder 1 wird. Sie erkennen, daß diese Schaltung nicht stabil ist. Sie schwingt. Da in dem Baustein eine Verzögerung eingebaut ist (siehe hierzu Kapitel 1o), erscheint nachdem der Eingang umgepolt wurde, das neue Ausgangssignal erst nach etwa 2o ms. Im rückgekoppelten Inverter unserer Schaltung liegt also 2o ms, nachdem das Signal C am Eingang anliegt, das entgegengesetzte Signal $\bar{C}$ an F und damit wiederum am Eingang (durch die Rückkopplung). Weitere 2o ms später geht das Signal bei F wieder auf C und erscheint somit auch am Eingang,

so daß insgesamt 4o ms das Signal am Ausgang einmal von beispielsweise 1 nach 0 und von 0 nach 1 gewechselt hat. Wir erkennen dies daran, daß in dieser Zeit die Lampe 2o ms leuchtet und 2o ms nicht leuchtet. Da sich dies 25-mal in der Sekunde wiederholt, beobachten wir ein rasches Flimmern der Lampe.

Wollen wir nun eine Blinkschaltung bauen, die so langsam blinkt, daß wir jedes Aufleuchten und Verlöschen der Lampe gut verfolgen können, so wählen wir eine größere Verzögerung des Ausgangssignals und schalten den Kondensator von 22o uF an die Kondensatorbuchsen des Logik-Bausteins. Die Verzögerung erhöht sich dadurch auf etwa 2oo ms, die Lampe leuchtet und erlischt daher nur noch etwa 2 1/2 mal in der Sekunde.

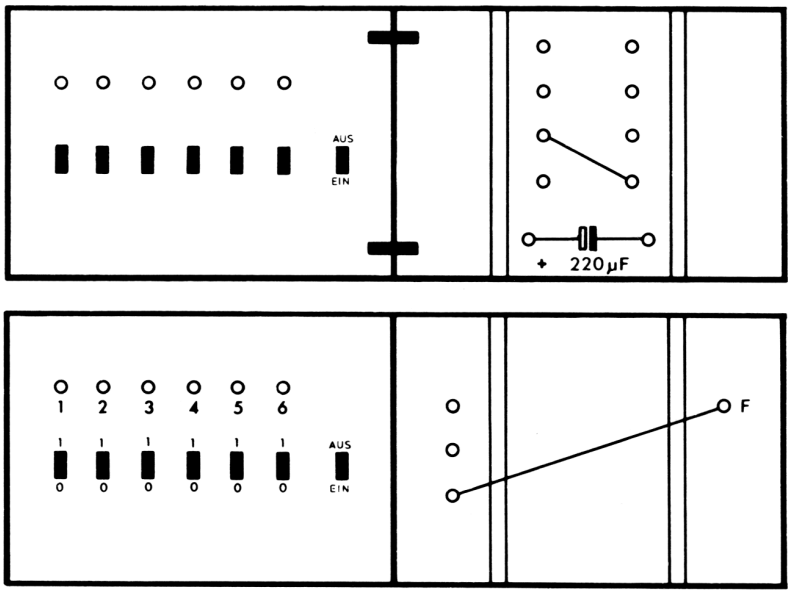


Abb.57

Nehmen Sie sich nun einen zweiten Logik-Baustein und programmieren Sie ihn folgendermaßen mit dem ersten zusammen:

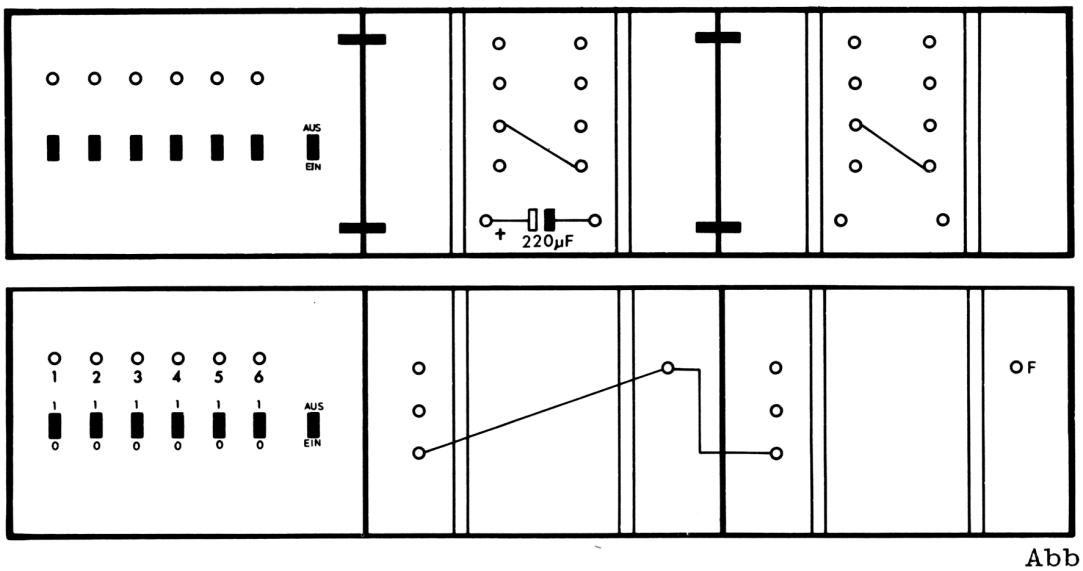


Abb.58

Wir haben einem Blinkgenerator eine weitere Inversionsschaltung (Inverter) angebaut. Wenn wir den Hauptschalter wieder einschalten, dann leuchten die Lampen des Generators und des Inverters abwechselnd auf, also im Gegentakt.

Wenn wir jetzt bei dem Inverter im Programmierfeld den Draht von $\bar{C}$ nach C verlegen, also auf diese Weise f4 mit C verbinden, dann leuchten beide Lampen gleichzeitig auf. Wir haben die Inversionsschaltung zu einer Identitätsschaltung umgebaut.

Nehmen Sie nun noch einen weiteren Baustein zur Hand und programmieren Sie ihn ebenfalls als Inversionsschaltung. Programmieren Sie auch den zweiten Baustein wieder als Inversionsschaltung:

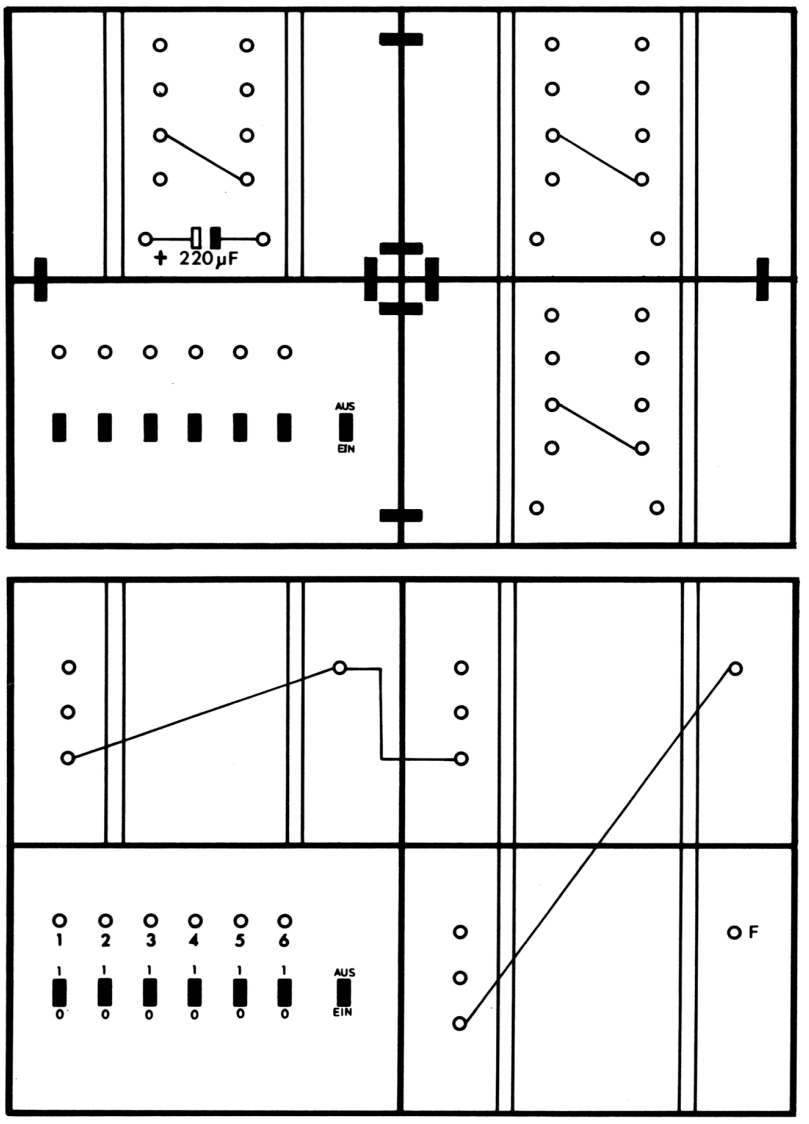


Abb.59

Sie erkennen, daß der zweite Inverter jetzt im Takt mit dem Generator blinkt, der erste Inverter im Gegentakt.

Das macht uns übrigens anschaulich, daß doppelte Verneinung eine Bejahung gibt; beide Inverter bilden sozusagen eine NICHT-NICHT-Schaltung.

Wir können dieses Spiel fortsetzen, bis wir alle fünf Logik-Bausteine aneinandergeschaltet haben. Von den fünf Bausteinen blinken nur der erste, dritte und fünfte im Takt und der zweite und vierte im Gegentakt. Wollen wir erreichen, daß z.B. der erste Baustein zusammen mit dem vierten und fünften blinkt und der zweite und dritte Baustein im Gegentakt blinken, so schalten wir den dritten und fünften Baustein nicht als Inverter, sondern als einfache Identitäts-Schaltung, d.h. wir verbinden f4 mit C:

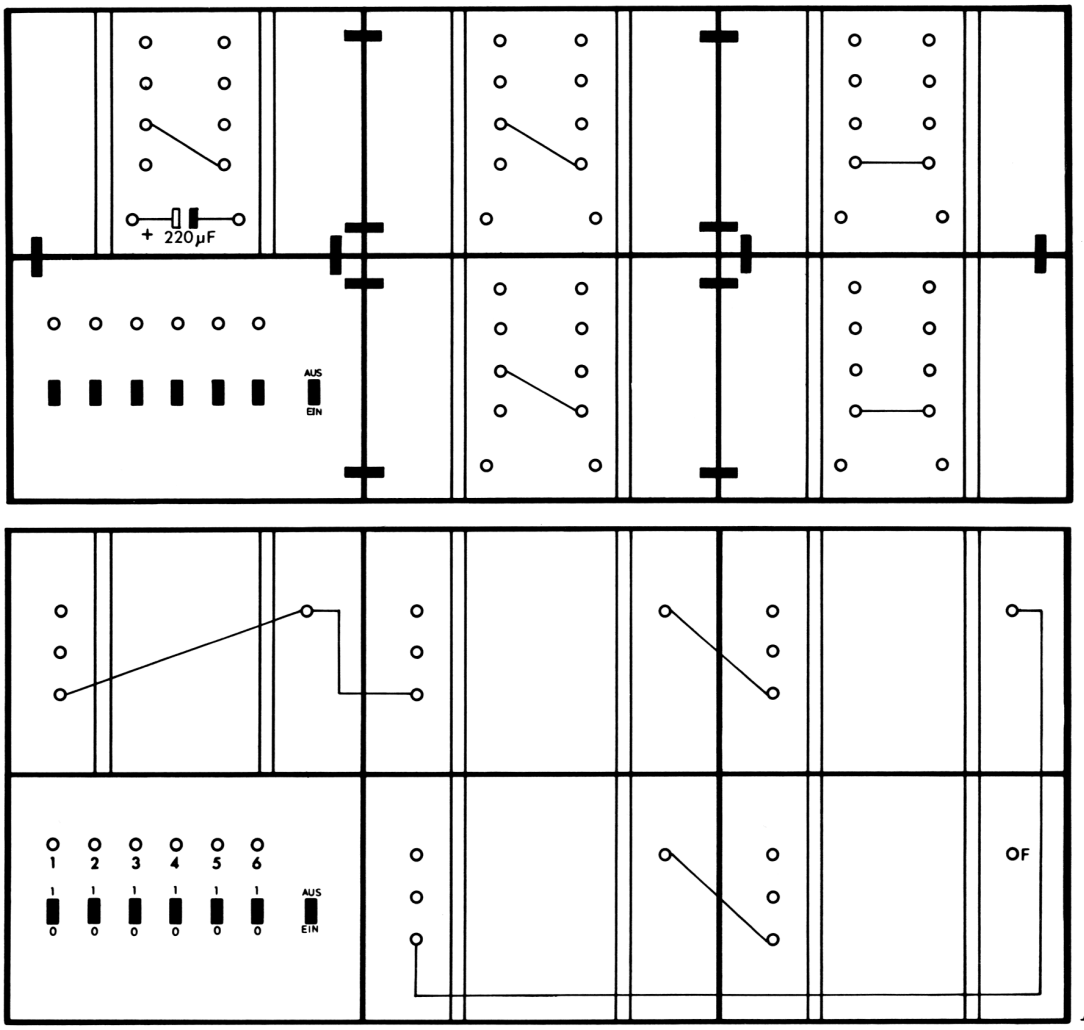


Abb.6o

Durch wahlweises Umstecken der Programmierverbindungen von f4 an die Klemmen C oder $\bar{C}$ des Programmierfeldes können wir eine beliebige Verteilung der im Takt mit dem Blinkgenerator und im Gegentakt blinkenden Lampen erreichen. Bei der zuletzt gezeigten Anordnung erhalten wir den Eindruck, daß die roten Lampen zwischen den Bausteinen 2 und 3 oben und 4 und 5 unten hin- und herspringen. Ebenso können wir sie auch von rechts nach links und zurück oder über Kreuz springen lassen. Oder wir lassen alle fünf Lampen im Gleichtakt blinken.

1o. Demonstration der Baustein-Laufzeiten

Jeder Baustein benötigt eine bestimmte Zeit, um ein oder mehrere Eingangssignale an den Ausgang weiterzugeben. Diese Zeit ist von den technischen Eigenschaften der Schaltung abhängig, z.B. von der Art der verwendeten Transistoren, von der Zahl der hintereinandergeschalteten Transistoren usw. Die Zeit, die ein oder mehrere Impulse brauchen, um vom Eingang eines Gatters an den Ausgang zu gelangen - mit dem der logischen Schaltung entsprechenden Ausgangswert - nennt man die Baustein-Laufzeit.

Wie wir bei den Versuchen mit dem Blinkgenerator gesehen haben, beträgt die Baustein-Laufzeit eines Logik-Bausteins ungefähr 2o ms. Sie läßt sich durch außen anzuschließende Kondensatoren verlängern.

Wir hatten eine NICHT-NICHT-Schaltung kennengelernt, also eine Identitätsschaltung, die das Eingangssignal unverändert bzw. doppelt invertiert an den Ausgang weitergibt, allerdings um die interne Baustein-Laufzeit von rund 2o ms verzögert. Wenn wir nun eine oder mehrere dieser NICHT-NICHT-Schaltungen zwischen den Ausgang F und den Eingang C unseres Blinkgenerators legen, an den wir jetzt keinen zusätzlichen Kondensator außen anschließen, so können wir erkennen, wie sich die Baustein-Laufzeiten der einzelnen Bausteine sichtbar addieren.

Blinkgenerator + Verzögerungsglieder	Gesamt-verzögerungszeit	Blinktakt Hz (Hertz = Takte/sec.)
0	ca. 2o ms	ca. 25 Hz
1	ca. 4o ms	ca. 12 Hz
2	ca. 6o ms	ca. 8 Hz
3	ca. 8o ms	ca. 6 Hz
4	ca. 1oo ms	ca. 5 Hz

Baustein 1 bildet also den Blinkgenerator (rückgekoppelter Inverter), die Bausteine 2 bis 5 wirken als Verzögerungsglieder. Jede einzelne Baustein-Laufzeit können wir durch einen außen angeschlossenen Kondensator noch verlängern.

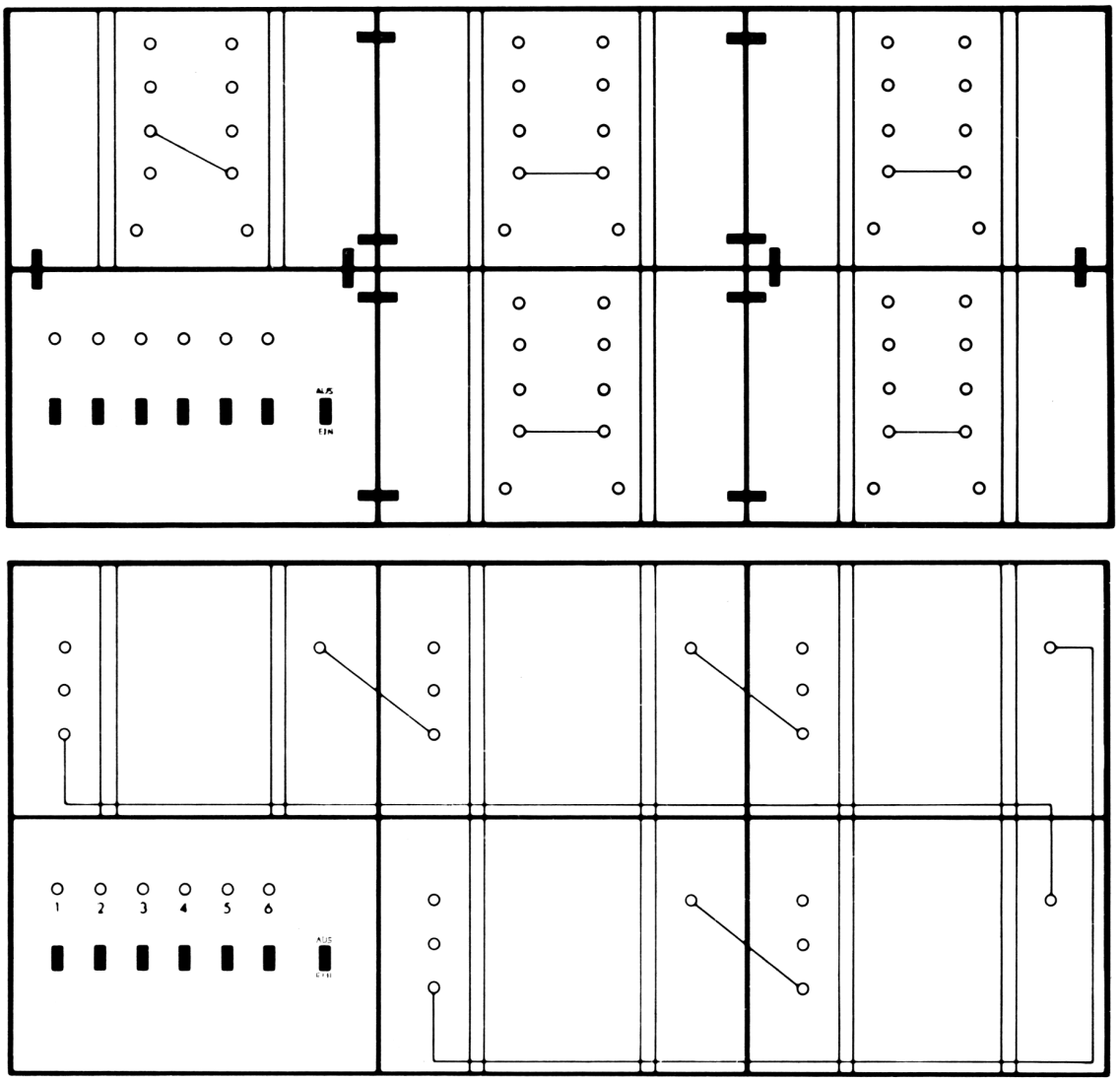


Abb.61

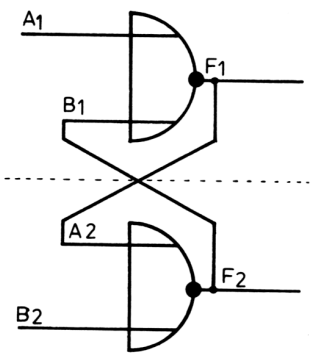
Je mehr Bausteine wir in die Rückkopplung des Blinkgenerators einbauen, um so größer wird die gesamte Laufzeit und um so langsamer blinkt der Generator. Die Anordnung mit 5 Bausteinen sieht dann so aus:

11. Flipflops

Um Informationen in einem Computer speichern zu können, bedient man sich außer dem Magnetbandspeicher häufig vollelektronischer Speicherelemente, die im Gegensatz zu den Bandgeräten ohne mechanisch bewegliche Teile auskommen. Sie können nahezu die gleichen Arbeiten übernehmen und haben als wesentlichen Vorteil eine viel kürzere Zugriffszeit.
Ein solches Speicherelement ist das Flipflop, das die Eigenschaft besitzt, duale Informationen, die kurzzeitig an seinem Eingang liegen, über einen beliebigen Zeitraum speichern zu können. Das Fassungsvermögen des Einzelspeichers beträgt 1 bit, das bedeutet, daß nur die kleinste duale Einheit (0 oder 1) gespeichert werden kann.

11.1. Speicher-Flipflop (R-S-Flipflop)

Die einfachste Form aller Flipflops ist das sogenannte Speicher-Flipflop. Es läßt sich z.B. aus zwei kreuzweise rückgekoppelten NOR-Schaltungen aufbauen, wie das folgende Schaltbild zeigt.



Bauen Sie zunächst diese Schaltung aus zwei Logik-Bausteinen auf. Aus der Funktionstabelle für eine NOR-Schaltung ergibt sich nach dem Programmierschema für zwei Eingangsvarialbe (A und B) - Kapitel 7 - folgende Verdrahtung:

Funktionstabelle NOR:

A	B	F	
0	0	1	⟶ f1 = 1
0	1	0	⟶ f2 = 0
1	0	0	⟶ f3 = 0
1	1	0	⟶ f4 = 0

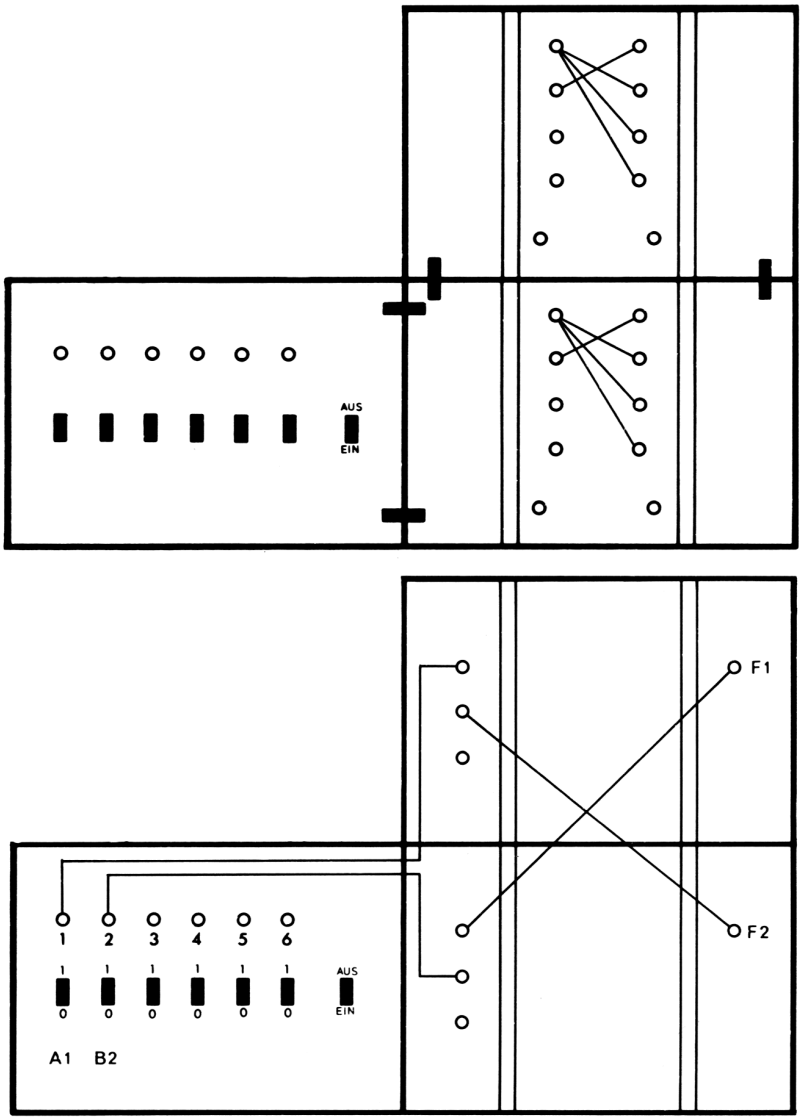


Abb.62

Sie können sich anhand der logischen Schaltsymbolik veranschaulichen, wie das Flipflop arbeitet. Die Funktionstabelle für beide kreuzweise gekoppelten NOR-Schaltungen unterstützt Sie bei diesen Überlegungen.

	Schalter/Eing. A1	B2	Ausg. F1
1.	0	0	0
2.	0	1	1
3.	0	0	1
4.	1	0	0
5.	0	0	0
6.	1	1	0
7.	0	0	1
8.	0	0	0

1. NOR			2. NOR		
A1	B1	F1	A2	B2	F2
0	1	0	0	0	1
0	0	1	1	1	0
0	0	1	1	0	0
1	1	0	0	0	1
0	1	0	0	0	1
1	0	0	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1

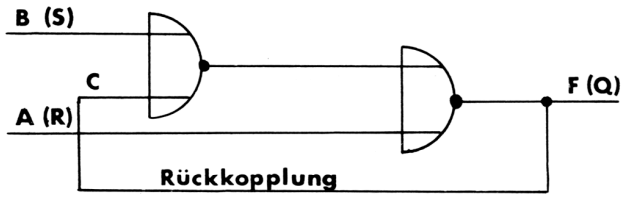
Anmerkung zu 7.: 1. NOR ist schneller als 2. NOR.
Anmerkung zu 8.: 2. NOR ist schneller als 1. NOR.

Gehen wir einmal von folgender Situation aus:
1.) Am Ausgang F1 liegt ein 0-Signal, und die Eingänge A1 und B2 sind ebenfalls auf 0. Aus dem rechten Teil der Tabelle können Sie entnehmen, daß der Ausgang der zweiten NOR-Schaltung auf 1 liegt, da durch die Rückkopplungsleitung von F1 auf A2 beide Eingänge 0 sind. 1 an F2 und aufgrund der Rückkopplung gleichzeitig an B1 bedeutet für F1 = 0. Dieser Zustand wird als stabil bezeichnet.
2.) Wird jetzt der Schalter B2 auf 1 gelegt, ist F2 = 0. Durch die direkte Drahtverbindung zwischen F2 und B1 wird F1 = 1, da beide Eingänge der 1. NOR-Schaltung 0 sind. Dieser Zustand des Flipflops wird ebenfalls als stabil bezeichnet. Der Ausgangswert F1 = 1 bleibt auch bestehen, wenn, wie in Position 3, der Schalter B2 wieder auf 0 gesetzt wird. Damit hat die Schaltung die Information 1 gespeichert.
Die Information können Sie nur löschen, indem Sie A1 auf 1 schalten (4.). F1 ist dann 0, dementsprechend auch A2 und B2, so daß F2 = 1 wird.
5.) Legen Sie nun den Eingang A1 wieder auf 0. Die Information F1 = 0 bleibt erhalten.
Werden A1 und B2 = 1 (Pos. 6.), so ist für diese Zeit F1 zwar 0, bei der nachfolgenden Informationseingabe (A1 = 0 und B2 = 0 in Pos. 7. und 8.) stellt sich jedoch der Ausgang F1 auf einen nicht voraussehbaren Wert ein, der nur durch die Schaltzeiten der NOR-Bausteine bedingt ist. Arbeitet z.B. die erste NOR-Schaltung schneller, dann tritt der Ausgangswert F1 = 1 ein (7.). Ist dagegen die zweite NOR-Schaltung schneller, so wird F1 = 0 (siehe Pos. 8.). Sie können die Funktionstabelle auf diese Eigenart hin überprüfen, wenn Sie durch den Anschluß des 220 µF Elektrolytkondensators die Verzögerungszeit einer NOR-Schaltung künstlich verlängern. Sind beide Verzögerungszeiten gleich, dann blinkt die Schaltung im schnellen Rhythmus, sofern A1 und B2 gleichzeitig von 1 auf 0 geschaltet werden. Durch dieses undefinierte Verhalten bezeichnet man die Pos. 6. bis 8. als nicht stabil. Sie dürfen bei dem Betrieb eines Flipflops nicht auftreten.
Aus diesen Überlegungen läßt sich für das Speicher-Flipflop folgende stark vereinfachte Funktionstabelle ableiten:

A1	B2	F1
0	0	0 1
0	1	1
1	0	0
1	1	?

Zur Auswertung dieser Tabelle noch einige wichtige Anmerkungen: Bei allen Flipflops wird die Wertigkeit der Ausgangsvariablen F1 für den Zeitpunkt angegeben, wenn die einzustellenden Informationen nicht mehr an den Eingängen liegen und, wie in unserem Beispiel A1 und B2, wieder 0 sind. Daraus ergibt sich: Wenn A1 = 0 und B2 = 1, ist der Ausgang F1 = 1. Diese Information bleibt auch bestehen, wenn A1 und B2 wieder 0 werden. Ist A1 = 1 und B2 = 0, wird F1 = 0 und bleibt in diesem Zustand, auch wenn A1 und B2 wieder 0 werden. Aus diesem Grund finden Sie auch in der ersten Spalte der Funktionstabelle die zwei Möglichkeiten 0 und 1 für die Ausgangsvariable F1. In der letzten Zeile steht unter F1 ein Fragezeichen. Das bedeutet: Dieser Zustand ist nicht stabil, wenn A1 und B2 wieder auf 0 gehen.

In den vorangegangenen Überlegungen haben wir das Speicher-Flipflop mit zwei Logik-Bausteinen verwirklicht. Dadurch ergaben sich zwei Ausgänge, wobei jedoch nur der Zustand des Ausgangs F1 betrachtet wurde. Wenn das Logikschaltbild etwas umgezeichnet wird, erkennen Sie, daß bei nur einem Ausgang das Speicher-Flipflop sich mit einem Logik-Baustein realisieren läßt.



Um nach der Funktionstabelle die Programmierung finden zu können, trennen wir zunächst die Rückkopplungsleitung von F nach C auf.

A (R)	B (S)	C	F (Q)	
0	0	0	0	
0	0	1	1	→ f1 = C
0	1	0*	1	
0	1	1**	1	→ f2 = 1
1	0	0**	0	
1	0	1*	0	→ f3 = 0
1	1	0**	0	
1	1	1*	0	→ f4 = 0

Wenn Sie nun nur die Eingangsvariablen A und B betrachten, erkennen Sie, daß bei A und B = 0, F = 0 oder 1, bei A = 0 und B = 1 F nur 1 sein kann. Wird B wieder 0, bleibt die Information F = 1 erhalten. Bei A = 1 und B = 0 kann F nur 0 werden, und auch bei A und B = 1 wird F = 0. Dieser letzte Zustand ist jedoch nicht stabil. Durch die Rückkopplung von F auf C sind die in der Funktionstabelle mit einem Stern versehenen Zustände nicht möglich und gehen innerhalb der Zweiergruppe in die mit zwei Sternen versehenen Zustände über.
Aus der gefundenen Programmierung ergibt sich folgende Verdrahtung:

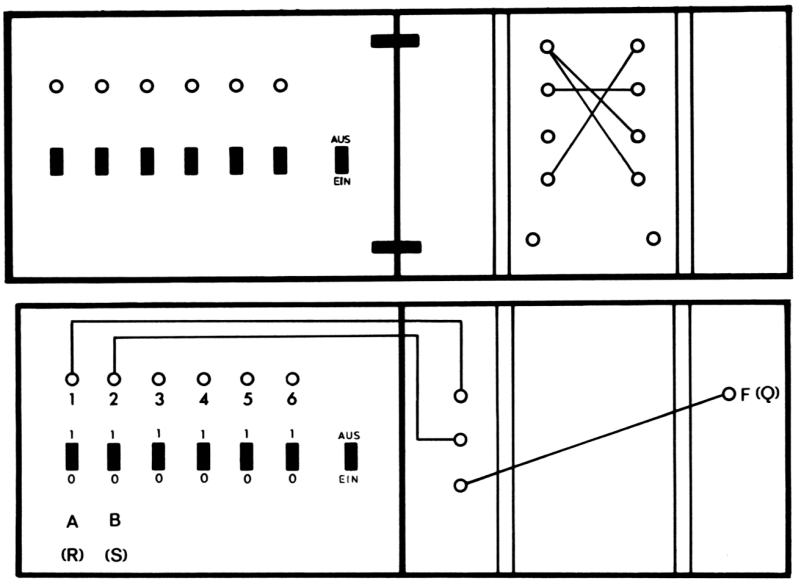


Abb.63

Zu Beginn des Experimentes sollen die beiden Schalter A und B auf 0 stehen. Je nach der Endsituation eines vorangegangenen Schaltexperimentes steht der Ausgang F auf 1 oder 0. Unabhängig davon hat der Ausgang F jedoch den Wert 1, wenn Sie den Schalter B auf 1 stellen. Schalten Sie den Schalter B dann wieder auf 0, werden Sie sehen, daß der Wert F = 1 erhalten bleibt; die 1 ist gespeichert. Auch wenn Sie den Schalter B wieder auf 1 legen, ändert sich der Speicherzustand nicht.

Betätigen Sie nun den Schalter A. Sobald Sie ihn auf 1 stellen, wird die 1 am Ausgang F wieder gelöscht; wir können auch sagen, der Ausgang F hat den Wert 0. Dieser Wert 0 bleibt auch erhalten, wenn Sie den Schalter A wieder auf 0 zurückstellen, mit anderen Worten, unser Speicher-Flipflop hat jetzt den Wert 0 gespeichert.
Die Schalterstellung A und B = 1 ergeben an F zwar 0, wenn aber beide Schalter wieder auf 0 gesetzt werden, stellt sich ein Zustand ein, der nicht berechenbar ist und dem Zufall überlassen werden muß.

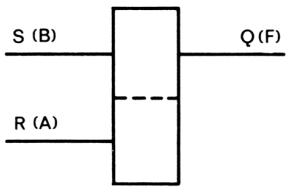
Die Operation B (S) = 1 (Betätigung des Schalters B) wird auch mit "set" (setzen) und A (R) = 1 mit "reset" (rücksetzen) bezeichnet. Daher die zunächst in Klammern gesetzten Eingangsbezeichnungen R und S.
Außerdem spricht man bei einem Flipflop von dem Ausgang Q, wobei der stabile Ruhezustand, der in unserem Beispiel bei A (R) = B (S) = 0, die Werte 0 oder 1 annehmen kann. Aus diesen neuen Gesichtspunkten kommen wir zu der Funktionstabelle. Die in Klammern gesetzten Ein- und Ausgangsbezeichnungen beziehen sich auf die Eingänge des Logik-Bausteins.

R (A)	S (B)	Q (F)
0	0	0 1
0	1	1
1	0	0
1	1	?

Die Funktionstabelle ist folgendermaßen zu lesen:

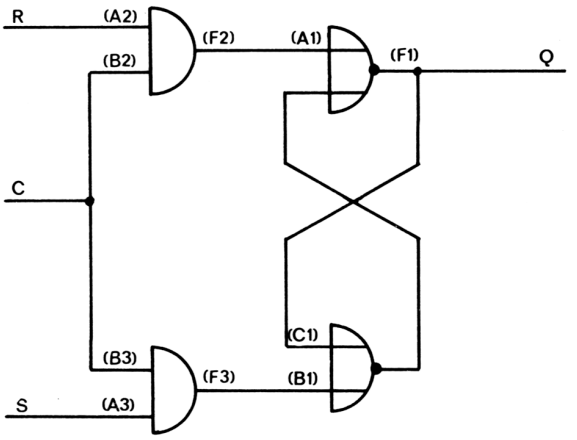
Bei R = S = 0 behält Q seinen alten Wert (Q = 0 oder 1).
R = 0 und S = 1 bewirkt Q = 1.
R = 1 und S = 0 bewirkt Q = 0.
R = 1 und S = 1 bewirkt Q = 0, solange die Eingangssignale erhalten bleiben. Werden die Eingangssignale jedoch gleichzeitig wieder von 1 auf 0 gesetzt, so stellt sich ein nicht vorhersehbarer Zustand ein (Q = ?).

Für ein Speicher-Flipflop sieht das Logik-Symbol folgendermaßen aus:



11.2. Auffang-Flipflop

Ein abgewandeltes Speicher-Flipflop ist das sogenannte Auffang-Flipflop, das vielfach auch als getaktetes R-S-Flipflop bezeichnet wird. Es unterscheidet sich von dem Speicher-Flipflop nur durch die beiden vorgeschalteten UND-Schaltungen.



Beide UND-Schaltungen bewirken, daß die Signale R und S nur dann an den Eingängen des Flipflops wirksam werden können, wenn der Takteingang C (C von englisch "clock") auf 1 geht. Bei dieser Schaltungsart des Speicher-Flipflops kann man die Eingänge R und S als sogenannte Vorbereitungseingänge betrachten, die das logische Verhalten des Flipflops vorbestimmen, wobei der Eingang C den Zeitpunkt festlegt, wann die an den Vorbereitungseingängen liegende Information gespeichert werden soll. In Computern findet man diese Art des Flipflops überall dort, wo nur zu einem genau definierten Zeitpunkt Informationen gleichzeitig in bestimmte Speicher gegeben werden sollen, während andere Speichergruppen für diese Informationen gesperrt bleiben.

C	R	S	Q
0	0	0	0 1
0	0	1	0 1
0	1	0	0 1
0	1	1	0 1
1	0	0	0 1
1	0	1	1
1	1	0	0
1	1	1	?

Und so sieht die Schaltung des Auffang-Flipflops mit 3 Logik-Bausteinen aus:

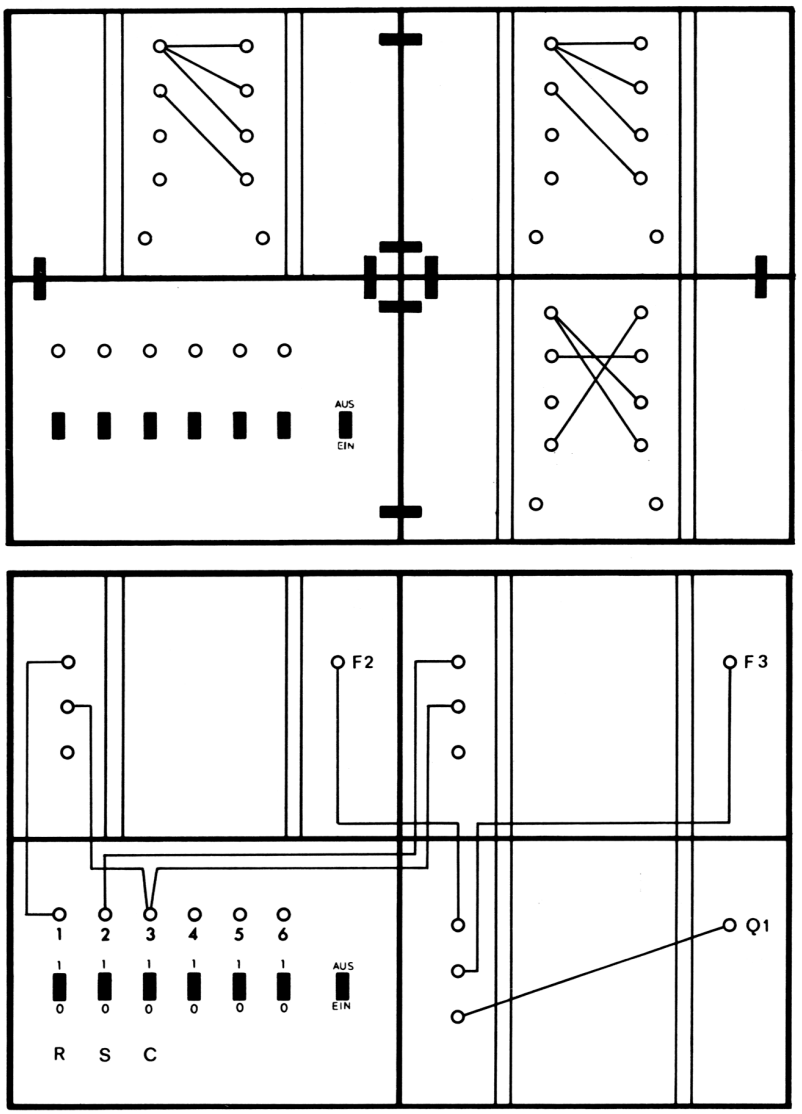
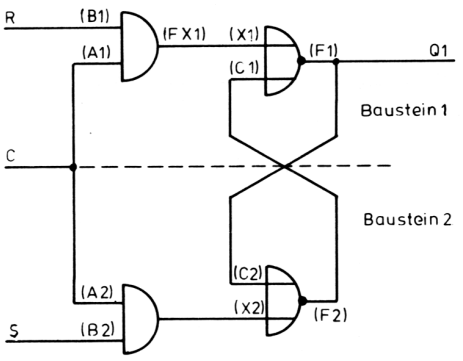


Abb.64

Wenn Sie das Auffang-Flipflop mit nur zwei Bausteinen realisieren wollen, müssen Sie sich die Gesamtschaltung als zwei symmetrisch aufgeteilte Schaltungshälften vorstellen, die aus in Reihe geschalteten UND- und NOR-Gattern bestehen.



Um die Funktionstabelle für die Reihenschaltung einer UND- und NOR-Schaltung aufstellen zu können, brauchen Sie zunächst nur das Logikschema für den ersten Baustein zu betrachten. Aus den Einzeltabellen für das UND und NOR mit zwei Eingängen ergibt sich folgende Funktionstabelle, aus der Sie leicht die Programmierung ableiten können:

UND

A1	B1	FX1
0	0	0
0	1	0
1	0	0
1	1	1

NOR

X1	C1	F1
0	0	1
0	1	0
1	0	0
1	1	0

A1	B1	C1	F1	
0	0	0	1	
0	0	1	0	→ $f1 = \bar{C}$
0	1	0	1	
0	1	1	0	→ $f2 = \bar{C}$
1	0	0	1	
1	0	1	0	→ $f3 = \bar{C}$
1	1	0	0	
1	1	1	0	→ $f4 = 0$

Da beide Schaltungshälften die gleiche Funktion aufweisen, erhalten Sie mit den zusätzlich anzubringenden Rückkopplungsleitungen F2 - C1 und F1 - C2 den Verdrahtungsplan eines Auffang-Flipflops mit zwei Logik-Bausteinen.

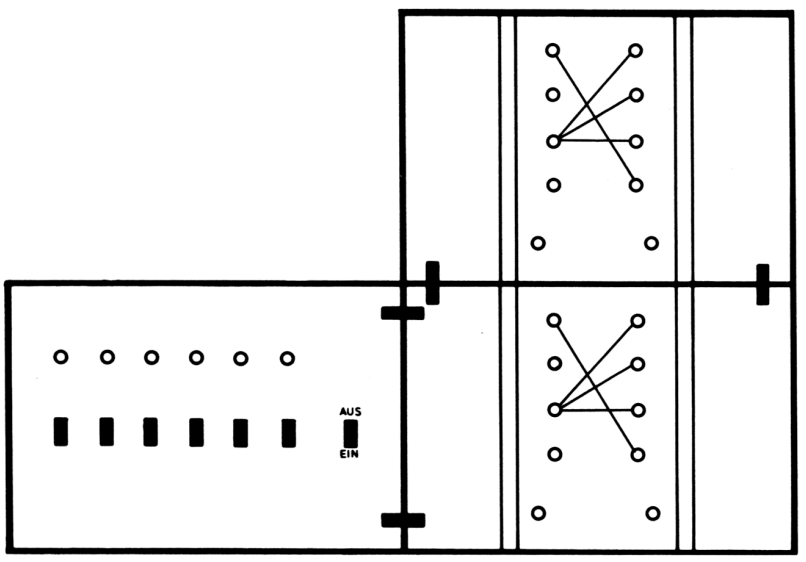


Abb.65

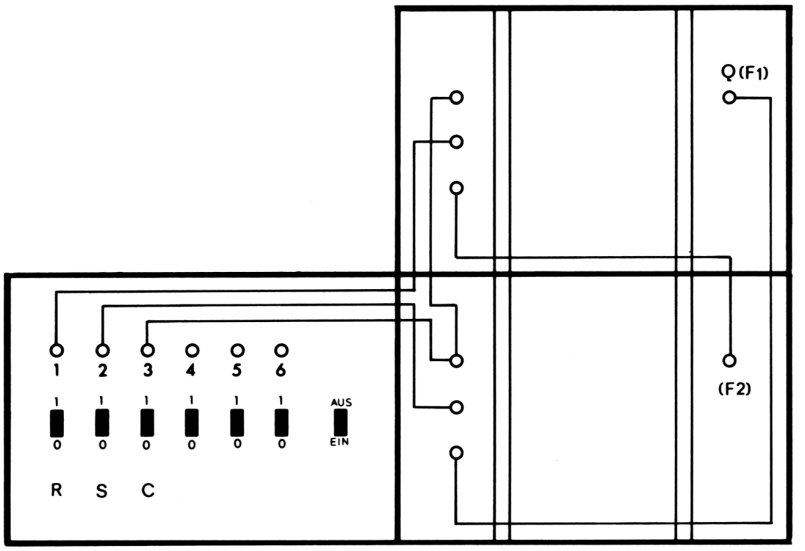
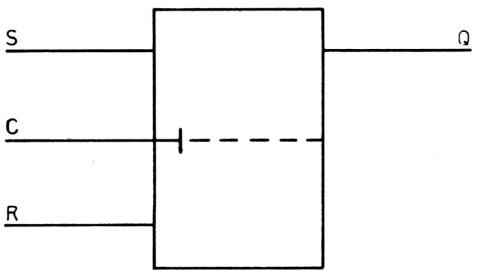


Abb. 65

Funktionstabelle des
Auffang-Flipflops

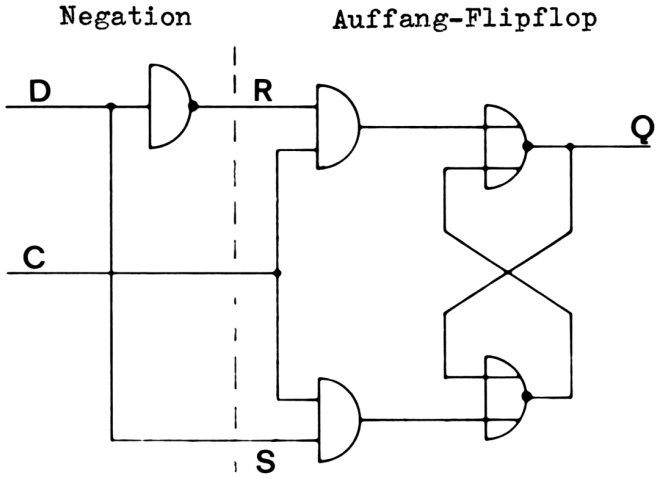
C	R	S	Q1
0	0	0	0 1
0	0	1	0 1
0	1	0	0 1
0	1	1	0 1
1	0	0	0 1
1	0	1	1
1	1	0	0
1	1	1	?

Logiksymbol des
Auffang-Flipflops



D-Auffang-Flipflop 1

Ein anderes Auffang-Flipflop ist das sogenannte D-Auffang-Flipflop (Delay-Auffang-Flipflop, englisch delay = Verzögerung), das in vier verschiedenen Ausführungsformen möglich ist. Betrachten Sie zunächst den Logikschaltplan des D-Auffang-Flipflop 1 - kurz DFF1 genannt.



Wie Sie erkennen können, besteht es aus dem getakteten RS-Flipflop (Auffang-Flipflop) und einer im (R)-Eingang liegenden Negation. Außerdem ist der negierte (R)-Eingang direkt mit (S) verbunden und mit der neuen Bezeichnung D versehen. Die Speicherinformation am D-Eingang kann das DFF1 wie auch beim Auffang-Flipflop nur dann übernehmen, wenn der Takteingang C = 1 ist.

Liegt bei C = 1 z.B. D auf 0, ist auch (S) = 0, durch die Negation wird (R) jedoch 1. Der Ausgang Q ist wie beim Auffang-Flipflop 1. Geht der Takteingang auf 0 zurück, bleibt die Information Q = 1 erhalten, auch wenn an D ein anderes Signal vorhanden ist. Bei C = 1 und D = 1 ist (S) = 1 und (R) = 0, daraus resultiert der Ausgang Q = 1.

Prüfen Sie diesen logischen Zusammenhang anhand der Verdrahtung eines D-Auffang-Flipflops 1 mit drei Logik-Bausteinen. Falls Sie noch den Auffang-Flipflop aus 2 Logik-Bausteinen aufgebaut haben, brauchen Sie nur noch die Negation in die Leitung (R) einzufügen und den Eingang der Negation mit S zu verbinden.

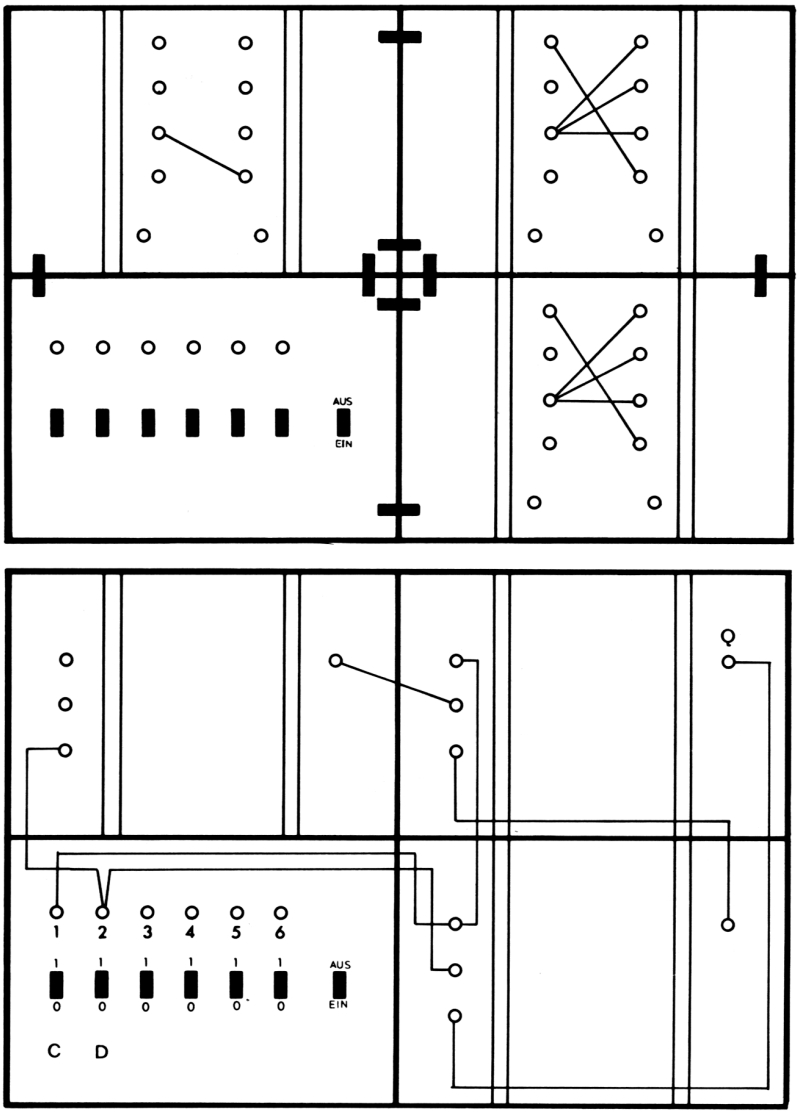
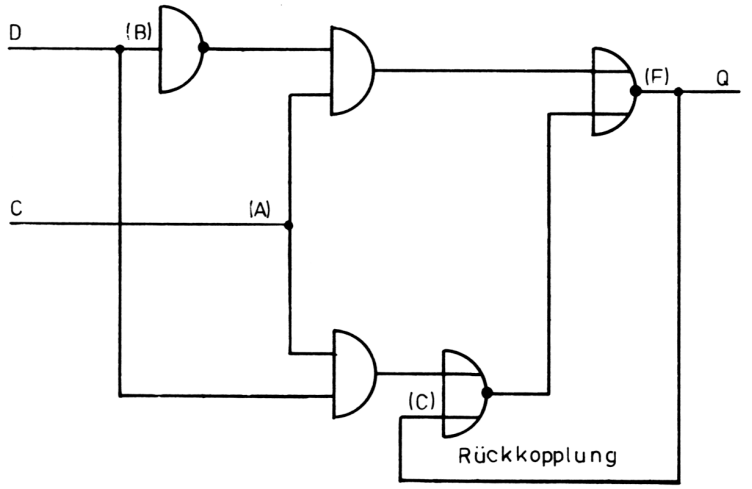


Abb.66

Sie erhalten dann eine Schaltung, die dieser Funktionstabelle entspricht:

C	D	Q
0	0	0
		1
0	1	0
		1
1	0	0
1	1	1

Versuchen Sie nun, diese Schaltung auf nur einen Logik-Baustein zu übertragen. Der Übersicht wegen zeichnen wir den Logikschaltplan so um, daß entsprechend dem zu programmierenden Logik-Baustein 3 Eingänge und ein Ausgang zu erkennen sind.



Alle in Klammern gesetzten Ein- und Ausgangsbezeichnungen beziehen sich auf die Ein- und Ausgangsbuchsen des Logik-Bausteins. Um die Programmierung finden zu können, müssen Sie sich zunächst die Rückkopplungsleitung von (F) nach (C) aufgetrennt denken. So können Sie aus der Funktionstabelle für das D-Auffang-Flipflop in bekannter Weise die Programmierung ableiten.

Funktionstabelle DFF1

C	D	Q
0	0	0
		1
0	1	0
		1
1	0	0
1	1	1

Programmierung

(A) C	(B) D	(C)	(F) Q	
0	0	0	0	
0	0	1	1	→ f1 = C
0	1	0	0	
0	1	1	1	→ f2 = C
1	0	0^{++}	0	
1	0	1^{+}	0	→ f3 = 0
1	1	0^{+}	1	
1	1	1^{++}	1	→ f4 = 1

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen, dann sind die mit + versehenen Zustände nicht möglich. Sie gehen in den mit ++ gekennzeichneten Zustand über.

Wenn Sie gemäß der gefundenen Programmierung den Baustein verdrahten und die Rückkopplungsleitung von (F) nach (C) anschließen, erhalten Sie das D-Auffang-Flipflop 1 mit nur einem Logik-Baustein.

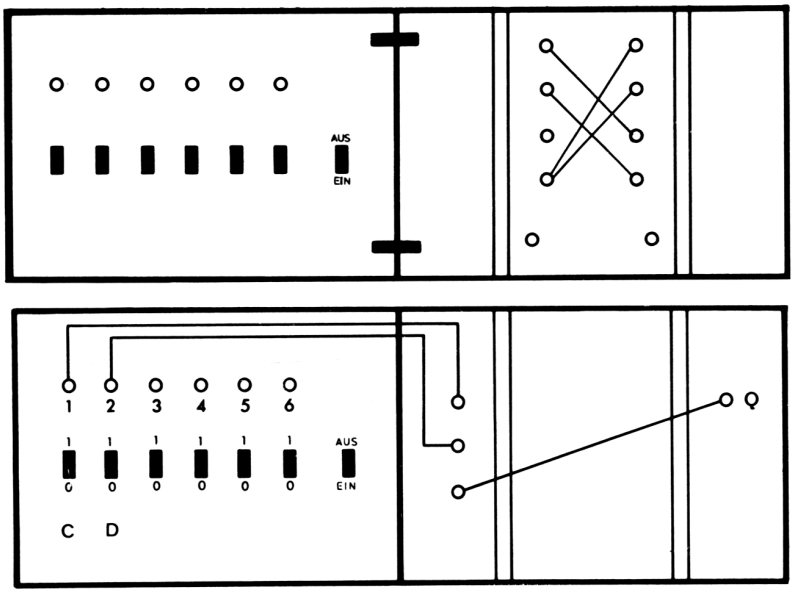
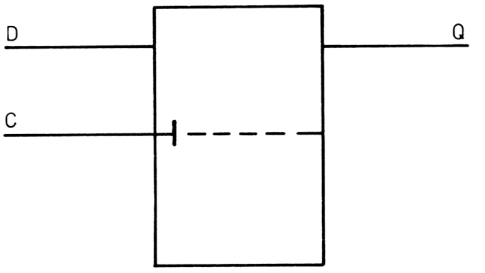


Abb.67

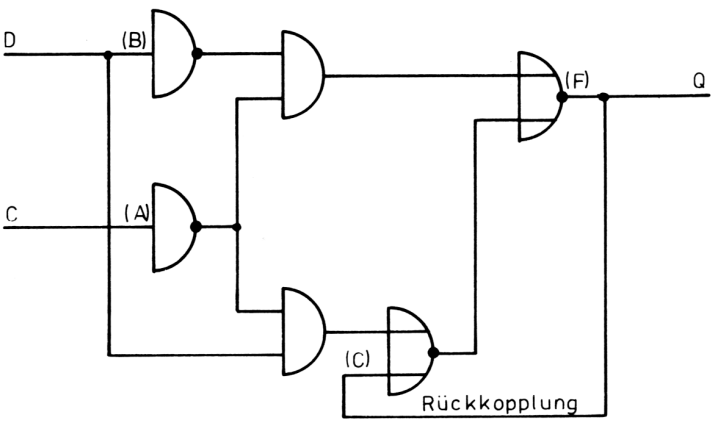


Logiksymbol
für das DFF1

D-Auffang-Flipflop 2 (DFF2)

Fast die gleichen Eigenschaften wie das DFF1 hat auch das D-Auffang-Flipflop 2. Es unterscheidet sich lediglich vom DFF1 in der Inversion des Takteingangs C. Nur wenn C = 0 ist, kann eine Information gespeichert werden. Bei C = 1 ist eine Informationseingabe nicht möglich, das Signal am Eingang D hat keinen Einfluß auf das Flipflop.

Logikschaltplan
des DFF2



Funktionstabelle DFF2

C	D	Q
0	0	0
0	1	1
1	0	0 1
1	1	0 1

Programmierung

(A) C	(B) D	(C)	(F) Q	
0	0	0^{++}	0	
0	0	1^{+}	0	→ f1 = 0
0	1	0^{+}	1	
0	1	1^{++}	1	→ f2 = 1
1	0	0	0	
1	0	1	1	→ f3 = C
1	1	0	0	
1	1	1	1	→ f4 = C

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen, dann sind die mit $^{+}$ versehenen Zustände nicht möglich. Sie gehen in den mit $^{++}$ gekennzeichneten Zustand über.

Und hier die
Verdrahtung des DFF2

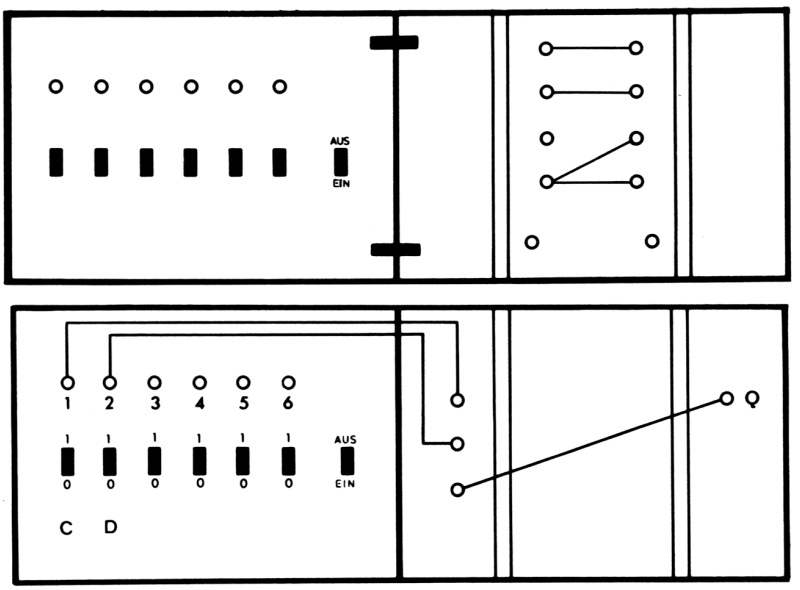
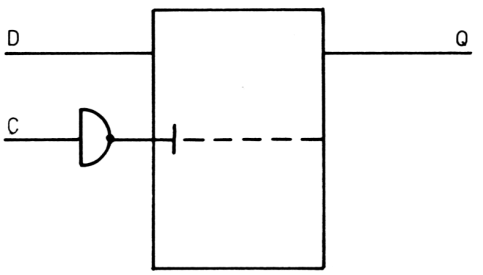


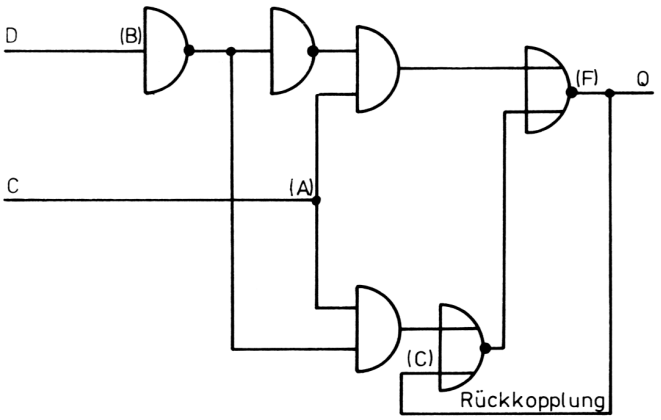
Abb.68

Logiksymbol



D-Auffang-Flipflop 3 (DFF3)

Soll bei einem geöffneten D-Eingang (Takteingang C = 1) die Eingangsinformation 0 am Ausgang Q eine 1 hervorrufen und umgekehrt, muß vor dem D-Eingang des DFF1 eine Negation geschaltet werden. Wir erhalten dann das DFF3 mit folgendem Logikschaltplan:



Daraus ergibt sich die Funktionstabelle und die Programmierung für einen Logik-Baustein.

Funktionstabelle DFF3

C	D	Q
0	0	0
		1
0	1	0
		1
1	0	1
1	1	0

Programmierung

(A) C	(B) D	(C)	(F) Q	
0	0	0	0	
0	0	1	1	→ f1-C
0	1	0	0	
0	1	1	1	→ f2-C
1	0	0^{+}	1	
1	0	1^{++}	1	→ f3-1
1	1	0^{++}	0	
1	1	1^{+}	0	→ f4-0

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen, dann sind die mit $^{+}$ versehenen Zustände nicht möglich. Sie gehen in den mit $^{++}$ gekennzeichneten Zustand über.

Und so sieht die Verdrahtung des DFF3 aus:

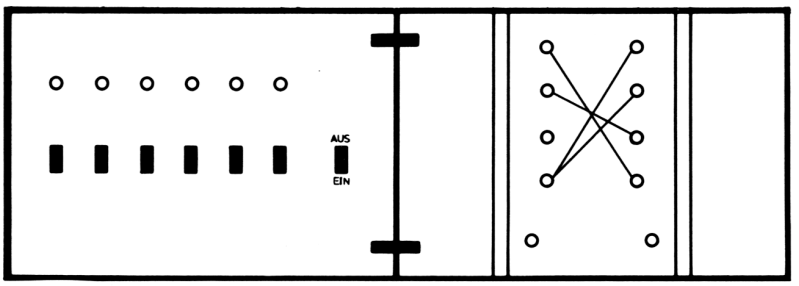


Abb.69

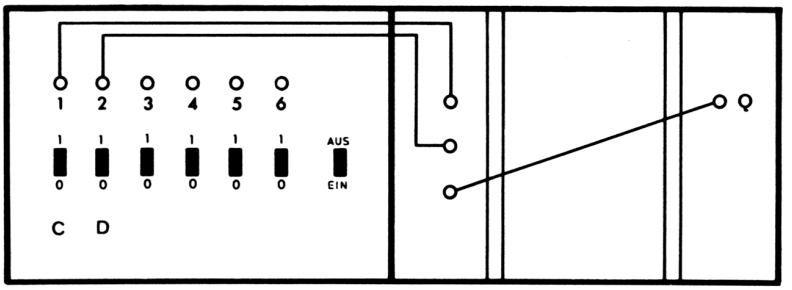
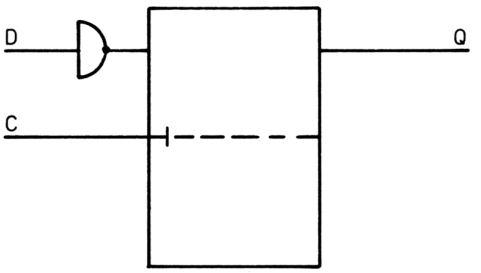


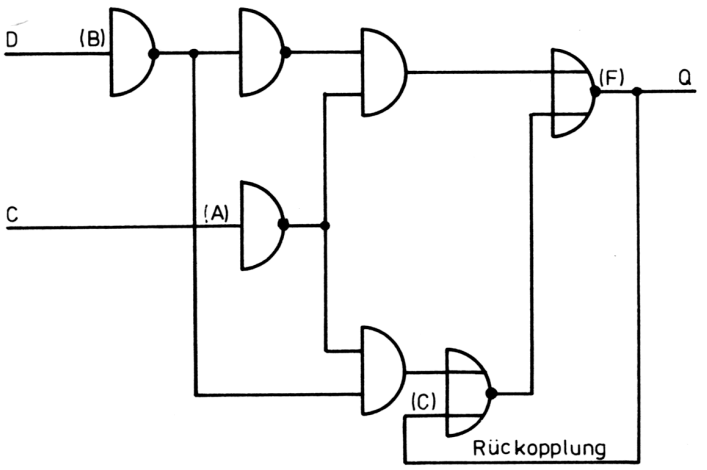
Abb. 69

Logiksymbol



D-Auffang-Flipflop 4 (DFF4)

Werden der D- und der C-Eingang des DFF1 invertiert, dann erhalten wir folgenden Logikschaltplan mit der dazugehörenden Funktionstabelle:



Funktionstabelle DFF4

C	D	Q
0	0	1
0	1	0
1	0	0 1
1	0	0 1

Programmierung

(A) C	(B) D	(C)	(F) Q	
0	0	0^{+}	1	
0	0	1^{++}	1	→ f1 = 1
0	1	0^{++}	0	
0	1	1^{+}	0	→ f2 = 0
1	0	0	0	
1	0	1	1	→ f3 = C
1	1	0	0	
1	1	1	1	→ f4 = C

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen, sind die mit $^{+}$ versehenen Zustände nicht möglich. Sie gehen in den mit $^{++}$ gekennzeichneten Zustand über.

Die Verdrahtung des so programmierten DFF4 sieht so aus:

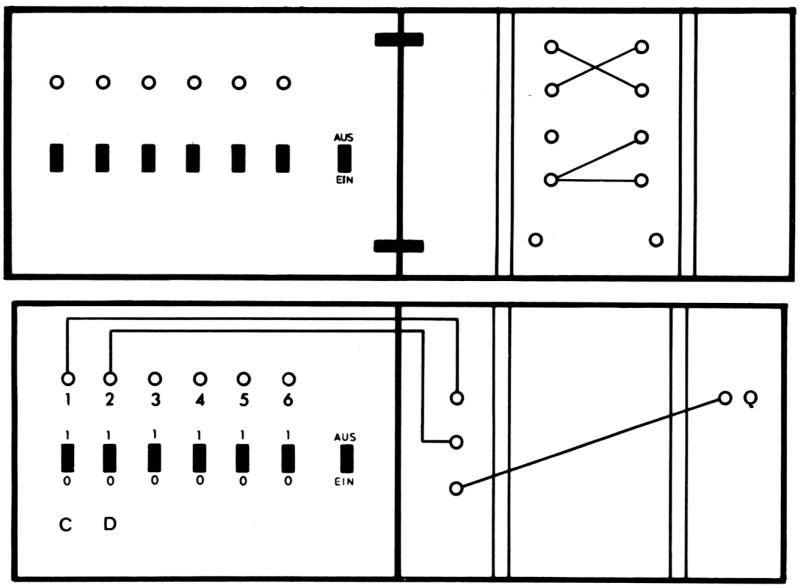
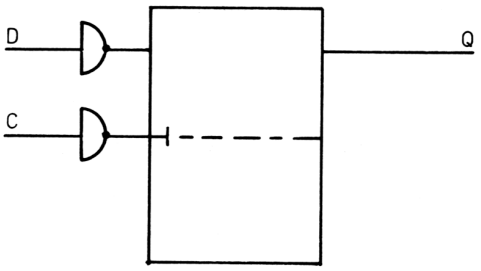


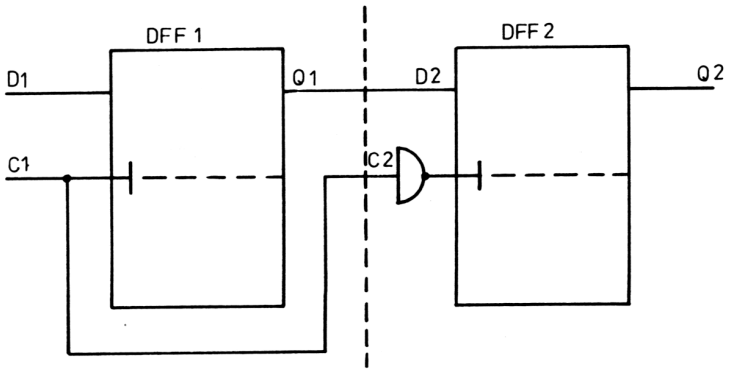
Abb.7o

Logiksymbol



11.3. 2-Speicher-Flipflop (Binärzähler)

In den Kapiteln 11.1. und 11.2. haben Sie das Speicher-Flipflop (RS-Flipflop) und das Auffang-Flipflop in verschiedenen Variationen kennengelernt. Es gibt jedoch noch eine weitere Flipflop-Schaltung, die ein sehr interessantes Verhalten zeigt: das sogenannte 2-Speicher-Flipflop (andere Bezeichnungen sind Zähl-Flipflop, Binäruntersetzer). Dieses Speicher-Flipflop besteht in seiner einfachsten Ausführung aus zwei unterschiedlichen D-Auffang-Flipflops, die, wie im Logikschema angegeben, miteinander verbunden sind.



Mit Hilfe der Funktionstabelle für das DFF1 und DFF2 können Sie leicht die Abhängigkeit des Ausgangssignals Q2 von den Eingangsinformationen C1 und D1 erkennen.

Funktionstabelle DFF1

C1	D1	Q1
0	0	0 1
0	1	0 1
1	0	0
1	1	1

Funktionstabelle DFF2

C2	D2	Q2
0	0	0
0	1	1
1	0	0 1
1	1	0 1

Für den Fall, daß der Takteingang C1/C2 von 0 auf 1 geht, wird die an D1 liegende Information zunächst vom DFF1 übernommen. Ein Beispiel: Ist C1/C2 = 1 und D1 auch 1, dann ist auch Q1 = 1. Dieser Wert kann vom DFF2 erst dann übernommen werden, wenn der Taktimpuls an C wieder von 1 auf 0 zurückgeht, da die Inversion des C2-Eingangs einen Einfluß des an D2 liegenden Eingangssignals bei C2 = 1 auf den Ausgang Q2 verhindert. Untersuchen Sie diesen Sachverhalt anhand der nachfolgenden Schaltung.

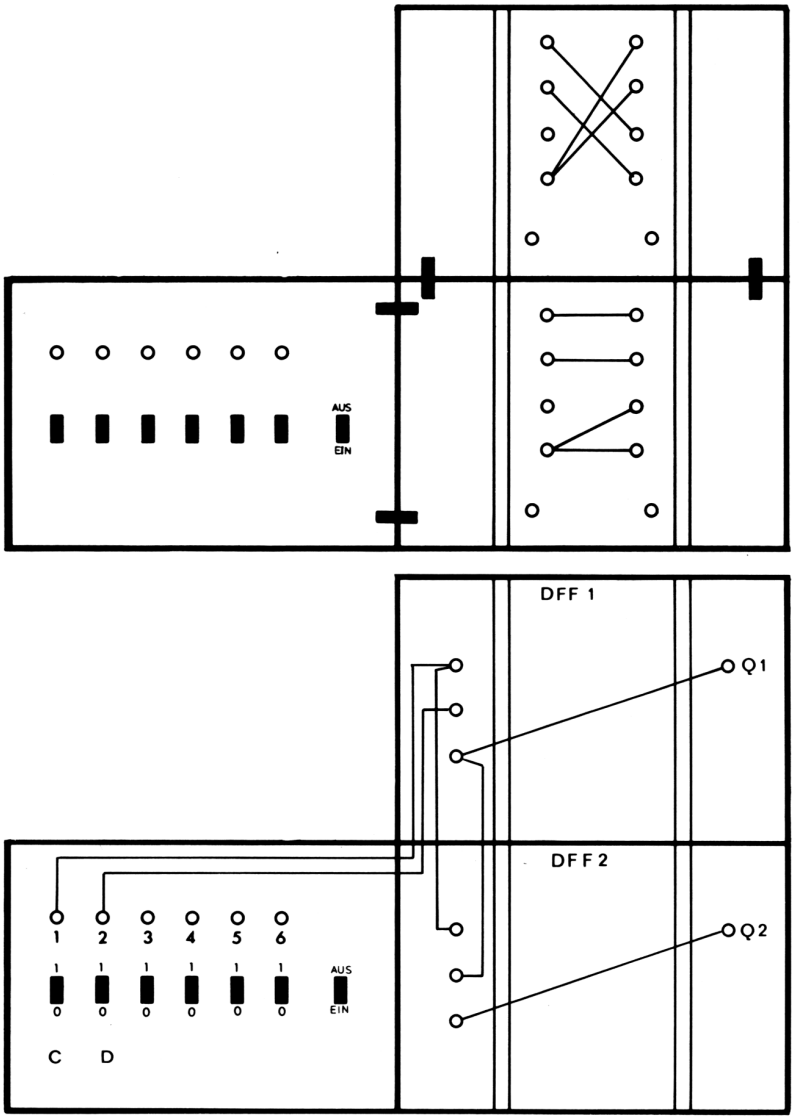
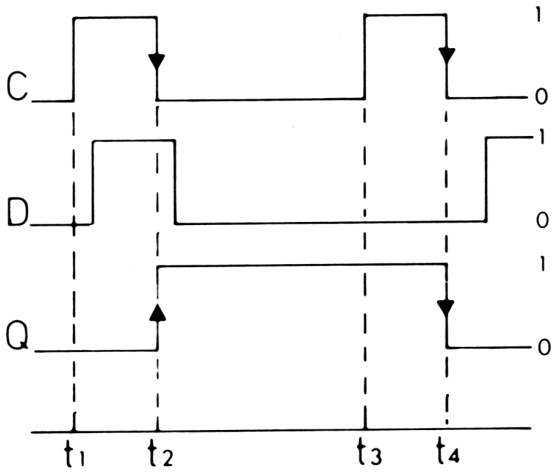


Abb.71

Da zunächst bei C = 1 das an D liegende Signal in den ersten Speicher (Master, Zwischenspeicher) gelangt und erst bei C = 0 vom zweiten D-Auffang-Flipflop (Slave, Hauptspeicher) übernommen wird, spricht man von einer Zwischenspeicherung des Signals. In einer Funktionstabelle ausgedrückt, sieht das Verhalten des 2-Speicher-Flipflops so aus:

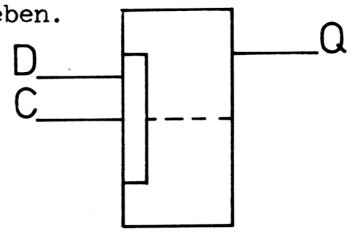
D	Q2
0	0
1	1

Bedingung: Nur in dem Augenblick, wenn C = 1 ist, beeinflußt die Information an D das Verhalten des Flipflops. Die Übergabe an den Ausgang Q2 erfolgt nur dann, wenn C wieder auf 0 zurückgeht.
Dieser Sachverhalt läßt sich auch zeichnerisch darstellen:

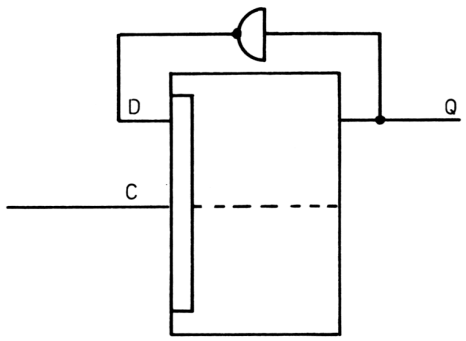


Wenn z.B. zum Zeitpunkt tl das an C liegende Signal auf 1 geht, wird der erste Speicher geöffnet. Nur die letzte im Zeitraum von tl bis t2 am Eingang D liegende Information wird zwischengespeichert und zum Zeitpunkt t2 an den Ausgang Q weitergegeben. Sie bleibt solange erhalten, bis bei t3 der Taktimpuls wieder auf 1 geht und den ersten Speicher öffnet. Liegt eine Informationsänderung vor, so wird diese zur Zeit t4 an den Ausgang Q weitergegeben.

Logiksymbol des 2-Speicher-Flipflops



Die Zwischenspeicherung eines Signals wird überall dort notwendig, wo zur gleichen Zeit eine Information gespeichert und eine andere abgegeben werden soll. Bei der Realisierung einer Zählschaltung z.B. muß der Binäruntersetzer aus einem 2-Speicher-Flipflop aufgebaut sein. Dieser Baustein, der nur noch den Takteingang C besitzt, hat die Eigenschaft, daß bei jedem Taktimpuls das Ausgangssignal in den entgegengesetzten Zustand springt. Einen solchen Binäruntersetzer erhalten Sie, wenn Sie das Ausgangssignal des 2-Speicher-Flipflops invertieren und auf den D-Eingang zurückführen.



Durch die Rückführung des invertierten Q-Signals liegt immer eine dem Ausgangssignal entgegengesetzte Information an D, die beim nächsten Taktimpuls von der 0 auf 1 gehenden Taktflanke in den Zwischenspeicher gegeben und mit der aktiven Flanke (C geht von 1 auf 0) vom Hauptspeicher (Q) übernommen wird.
Sie können diese Vorgänge leichter verstehen, wenn Sie den nachfolgenden Binäruntersetzer aus 3 Logik-Bausteinen aufbauen.

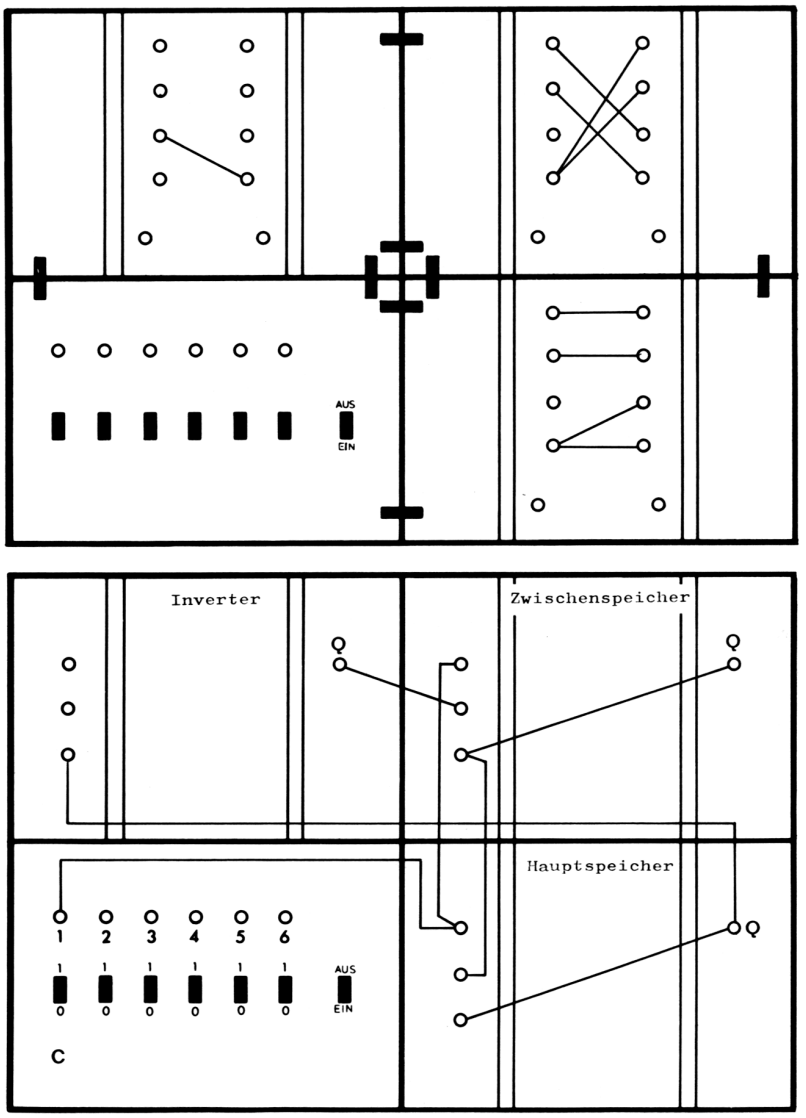
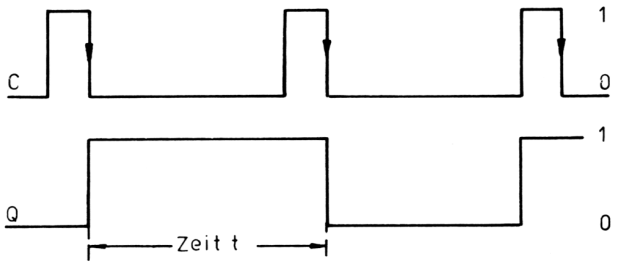


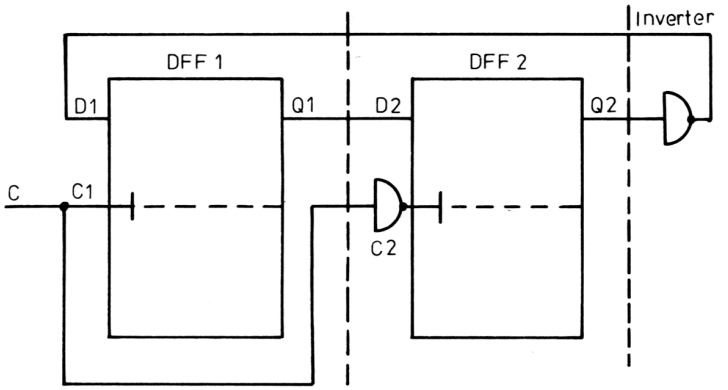
Abb.72

Schalten Sie den Taktschalter C von 0 auf 1. Sie erkennen, daß die Information am Ausgang des Inverters vom Zwischenspeicher übernommen wird und erst dann in den Hauptspeicher gelangt, wenn Sie C wieder auf 0 zurücksetzen. Wenn Sie nur die Anzeigelampe des Hauptspeichers betrachten, sehen Sie, daß der Ausgangszustand des Hauptspeichers (Ausgang Q) nur umkippt, wenn der Taktimpuls C von 1 auf 0 geht.



Aus der Eingangsinformation 0 und 1 (2 Zustände in der Zeit t) ist entweder ein 0- oder ein 1-Signal (1 Zustand in der Zeit t) geworden. 2 Taktimpulse ergeben also einen Ausgangsimpuls. Das Signal ist also durch den Faktor 2 geteilt, oder anders ausgedrückt, um den Faktor 2 untersetzt worden.

Dieser Zähl-Flipflop soll nun mit 2 Logik-Bausteinen realisiert werden. Betrachten Sie dazu nochmals den logischen Schaltungsaufbau.



Der zusätzlich in die Leitung D1 - Q2 geschaltete Inverter kann entfallen, wenn Sie das DFF1 durch ein DFF3 ersetzen, da dieses Flipflop einen invertierten D-Eingang besitzt. Sie erhalten dann einen Binäruntersetzer, der nur aus 2 Bausteinen besteht.

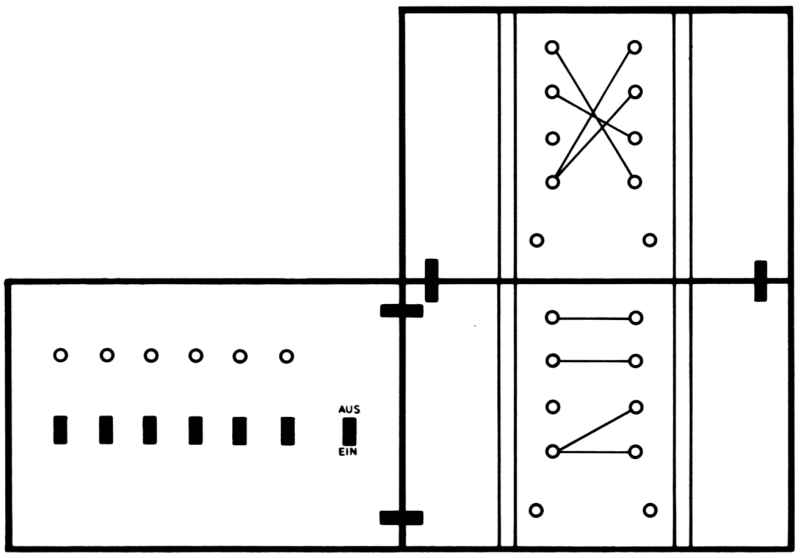


Abb.73

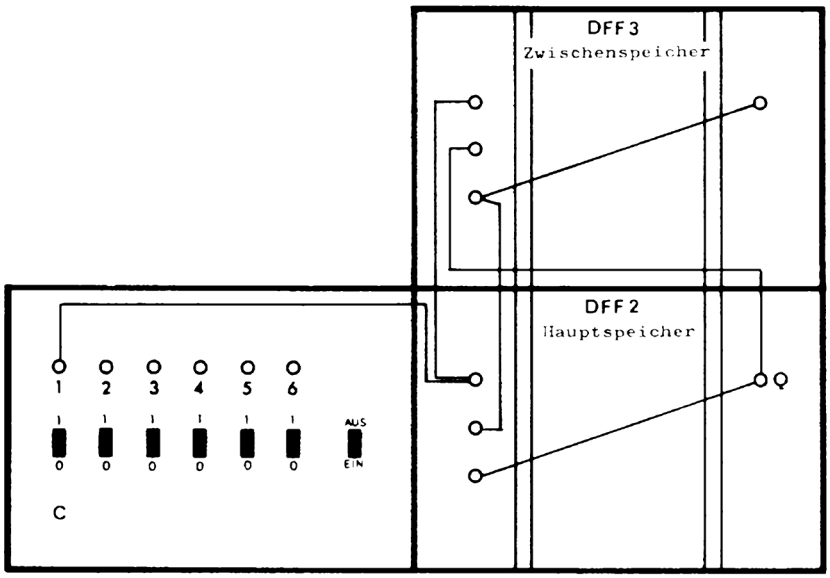
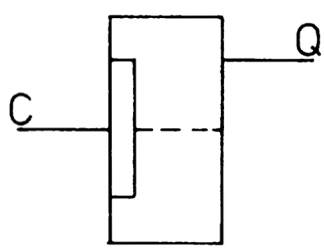


Abb. 73

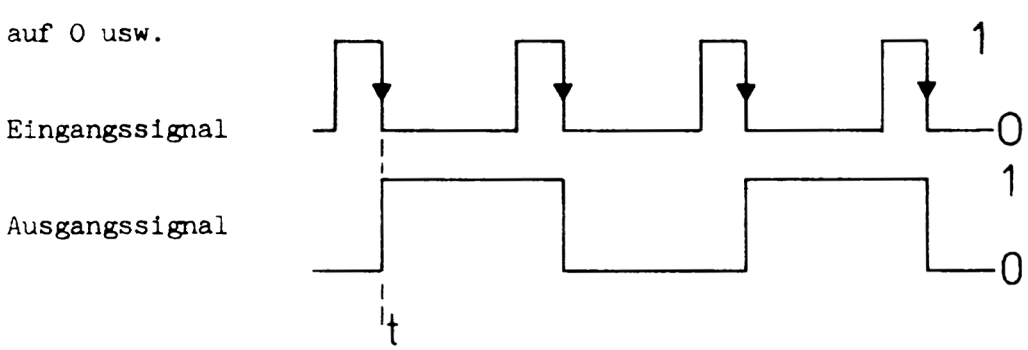
Logiksymbol des
Binäruntersetzers



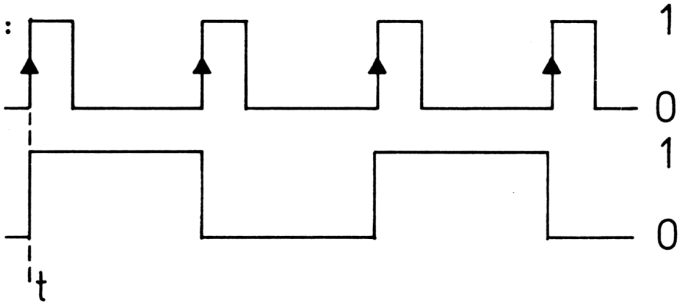
Im folgenden Beispiel finden Sie eine andere Schaltungsvariante für einen Binärzähler.

Generator mit Frequenzuntersetzer

Die Frequenz des Generators (Kapitel 10) läßt sich durch einen außen anzubringenden Kondensator verändern. Je größer die Kapazität wird, desto kleiner wird die Frequenz. Diese Frequenz kann mit einem Binärzähler noch zusätzlich verringert werden, und zwar im Verhältnis 1:2. Der Binärzähler wirkt also wie ein Frequenzuntersetzer.
Nach einem Eingangsimpuls (vom Generator) liegt der Ausgang des Binärzählers auf 0. Nach zwei Eingangsimpulsen liegt er auf 1, nach 3 wieder auf 0 usw.



Eine andere zusätzliche Zuordnung zwischen dem Eingangs- und dem Ausgangssignal sieht so aus:



Bei der ersten Möglichkeit trifft der Aufstieg des Ausgangssignals im Zeitpunkt t mit dem Abfall des Eingangssignals zeitlich zusammen. Bei dem zweiten Beispiel treffen zum Zeitpunkt t1 der Anstieg des Eingangs- und des Ausgangssignals zeitlich aufeinander.
Die Programmierung der beiden Logikbausteine zum Binärzähler wurde für die erste Möglichkeit gewählt.

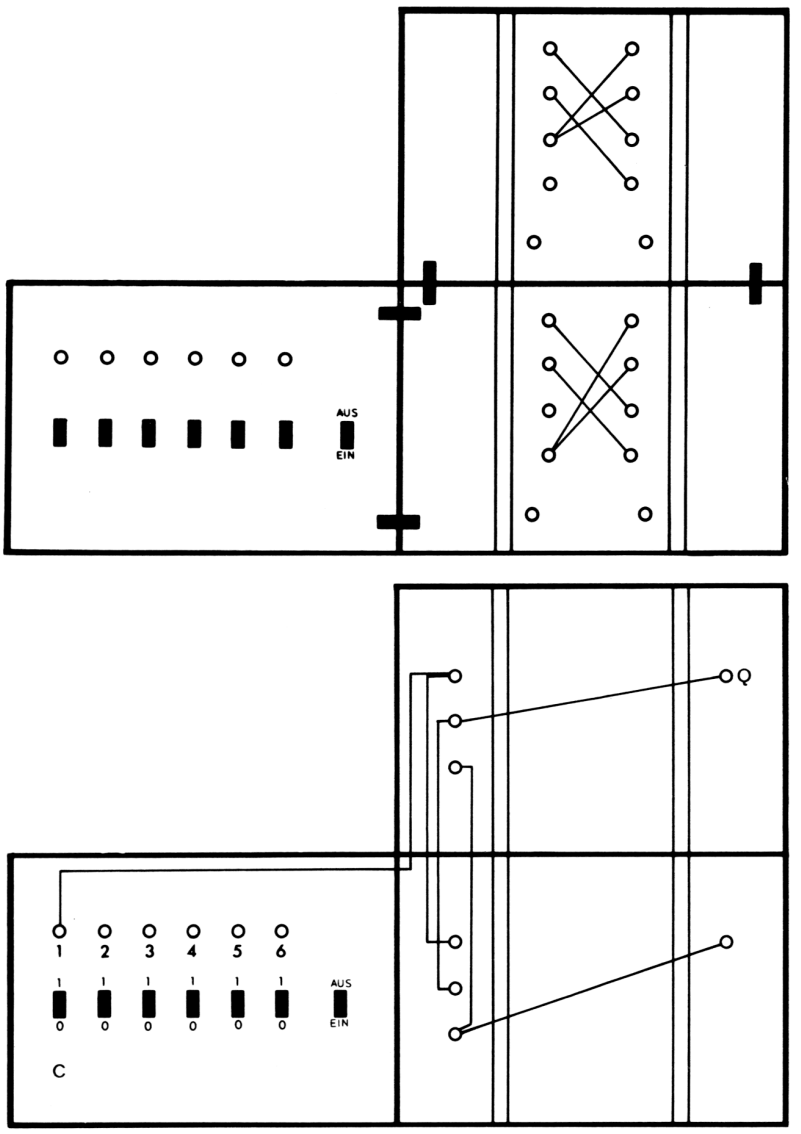


Abb.74

Die Funktion des Binärzählers wird aus den Funktionstabellen der beiden Logik-Bausteine deutlich.

A	B	C	F1	F2
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

Aus der Verdrahtung der Logik-Bausteine können Sie erkennen, daß

der Ausgang F1 mit B und
der Ausgang F2 mit C verbunden sind.

Die Eingangsvariable A entspricht dem Eingang des Binärzählers.

Dann können folgende Zustände auftreten:
Wenn sein Eingang 0, der Ausgang F1 = 1 und B = 0 sind, so ist dieser Zustand nicht stabil. Denn

F1 ≠ B
F2 = C

Wird B = F1 = 1, so wird der Zustand des Binärzählers stabil. Denn jetzt ist

F1 = B und
F2 = C.

Wird der Eingang dann 1, so gilt

A	B	C	F1	F2
1	1	0	1	1

Aus diesem nicht stabilen Zustand - F2 ≠ C - wird der stabile Zustand

A	B	C	F1	F2
1	1	1	1	1

Man kann so das Ausgangssignal F1 als Funktion des Eingangssignals ermitteln.
Verbinden Sie nun den Generator nach der folgenden Abbildung mit dem Binärzähler.

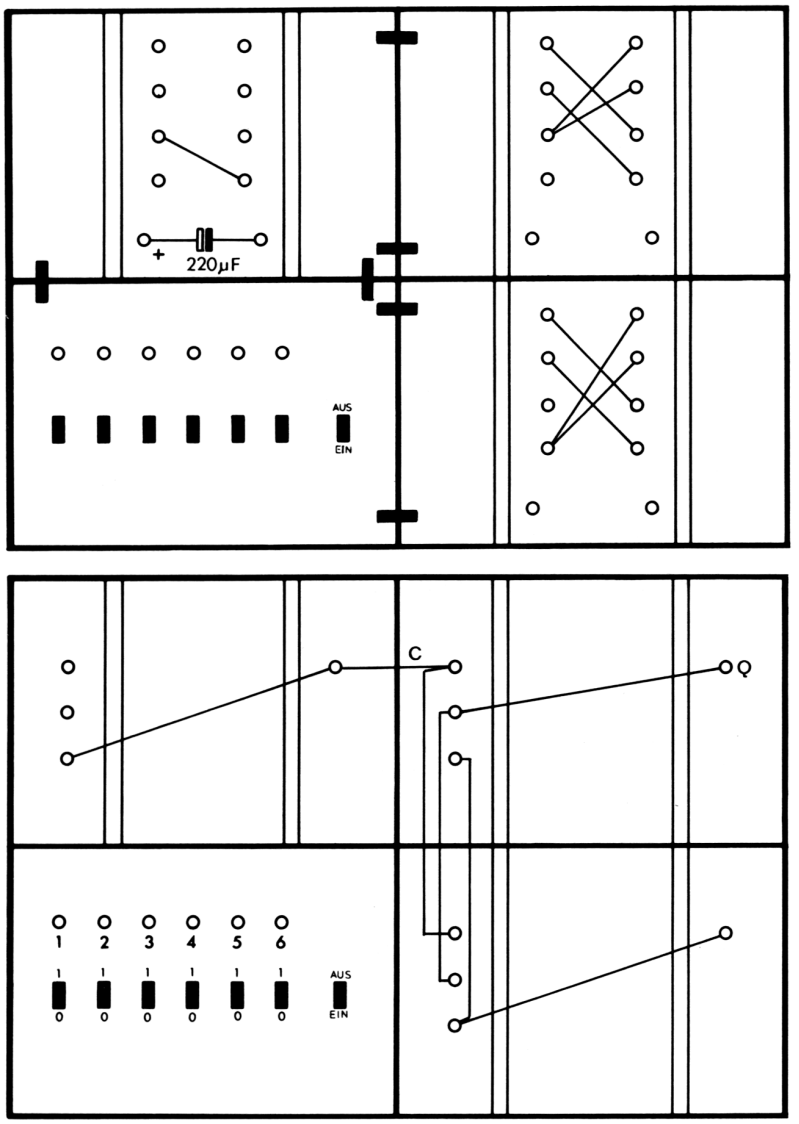


Abb.75

Sie erkennen, daß die Ausgangsfrequenz halb so groß ist wie die Frequenz des Generators.

Wenn Sie den Ausgang des Binäruntersetzers mit dem Eingang eines weiteren Zähl-Flipflops verbinden, können Sie das Ausgangssignal ein weiteres Mal untersetzen.

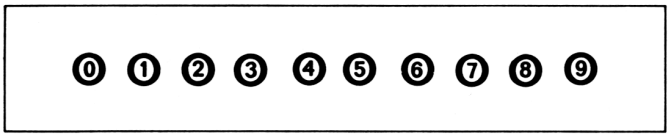
12. Das Dualsystem

12.1. Einführung

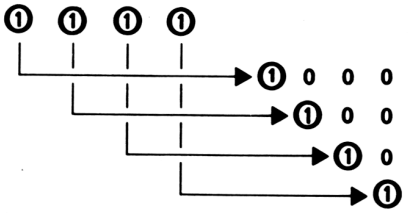
Computer haben merkwürdige Rechengewohnheiten - wenigstens kommt es uns so vor, wenn wir sie neu kennenlernen. So rechnen Computer z.B. mit dem einfachsten Zahlensystem, das es gibt, mit dem Dualsystem (Zweiersystem; duo (lateinisch): zwei; dualis: zweifach). Das Dualsystem kennt nur 2 Ziffern, nämlich die Ziffern

Ein ganzes Zahlensystem aus nur zwei Ziffern? - Ja, tatsächlich! Und Computer rechnen damit.
Wenn Sie diese Broschüre nicht nur überfliegen, können auch Sie im Dualsystem rechnen, so sicher wie mit dem Ihnen vertrauten Dezimalsystem. (Fast so sicher.)
Halten wir uns zunächst noch einmal die Prinzipien vor Augen, nach denen das Dezimalsystem aufgebaut ist. Wir verstehen dann das Dualsystem besser, denn es hat die gleichen Aufbauprinzipien als Grundlage.

Das Dezimalsystem (Zehnersystem; decem (lateinisch): zehn; decimalis: zehnfach) kennt lo verschiedene Ziffern:



Der Wert einer Dezimalziffer hängt nicht nur von ihr selbst ab, sondern auch von der Position, die sie innerhalb einer Zahl einnimmt. Hier ein Beispiel, die vierstellige (also aus 4 Ziffern bestehende) Zahl 1111:

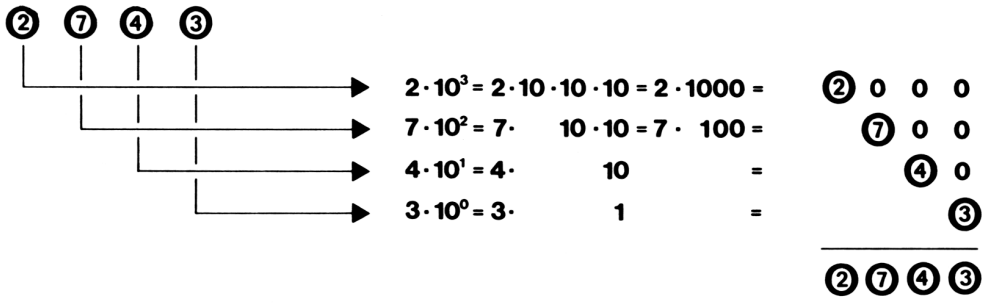


Von links beginnend hat also die erste 1 einen Wert von looo, die zweite 1 einen Wert von loo, die dritte 1 hat einen Wert von lo und die vierte 1 schließlich ist 1 wert. Jede 1 ist also lo-mal soviel wert wie die rechts neben ihr stehende 1. Mit anderen Worten: Das Dezimalsystem hat die Basis lo.

Das Aufbauprinzip des Dezimalsystems können wir uns an einer Zehnerpotenztabelle übersichtlich veranschaulichen.
Für den Fall, daß es Ihnen nicht mehr geläufig ist: z.B. 10^3 = 3. Potenz von 10; Basis: 10; Hochzahl (= Exponent): 3. Und noch etwas: Jede Zahl hoch 0 ist gleich 1; also auch $10^0 = 1$; $2^0 = 1$.

Ziehen wir gleich noch ein konkretes Beispiel hinzu, die Zahl 2743, und "errechnen" wir anhand der Zehnerpotenzentabelle den Gesamtwert dieser Zahl:

10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1



Die Gesamtzahl ist also die Summe der Werte der einzelnen Ziffernstellen (Ziffer mal Stellenwert).
Soweit die kleine Rekapitulation des vertrauten Dezimalsystems.

Nun zur Zahlenwelt des Computers, zum Dualsystem. - Warum rechnen Computer mit Dualzahlen? - Ganz einfach deshalb, weil es technisch am einfachsten ist. Man unterscheidet bei elektromechanischen und elektronischen Bauteilen, die in einem Computer verwendet werden, nur zwischen zwei verschiedenen Zuständen: Durch einen Draht fließt entweder Strom oder es fließt kein Strom; ein Schalter (z.B. ein Transistor) ist entweder offen oder geschlossen; ein Magnetring ist entweder rechts herum magnetisiert (Richtung des magnetischen Nordpols) oder links herum usw. Wollten wir Computer im Dezimalsystem rechnen lassen, so müßten wir 10 verschiedene Zustände ermöglichen (beispielsweise 0, 1, 2, ... 9 Volt), was zwar technisch möglich, aber erheblich komplizierter wäre.
Von den beiden verschiedenen Zuständen nennen wir den einen einfach 0 und den anderen 1. Damit haben wir die beiden Ziffern des Dualsystems.
Wenn wir uns im folgenden mit dem Dualsystem befassen, wollen wir dabei das Dezimalsystem immer zum Vergleich und zur klaren Orientierung hinzuziehen, da es nun einmal das uns vertrauteste Zahlensystem ist.

Zunächst sind die Dualzahlen 0 und 1 mit den beiden Dezimalzahlen 0 und 1 gleichbedeutend:

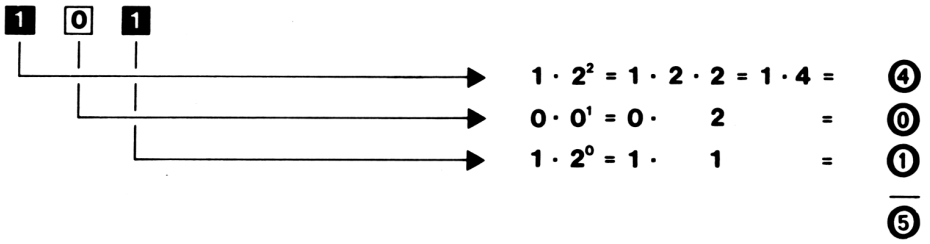
Dualsystem				Dezimalsystem
	0	=	0	
	1	=	1	

Während wir im Dezimalsystem 10 Ziffern (bzw. 10 einstellige Zahlen) haben, gibt es im Dualsystem nur die beiden Ziffern (bzw. einstelligen Zahlen) 0 und 1. Im Dezimalsystem ist 10 (Zehn) die erste zweistellige Zahl, dementsprechend heißt die erste zweistellige Zahl im Dualsystem "Zwei". Die "Zwei" aber sieht genauso aus wie die "Zehn" des Dezimalsystems: 10.

Für 10 im Dualsystem sagen wir aber weder "Zwei" noch "Zehn", sondern "Eins-Null". 10 ("Eins-Null") im Dualsystem entspricht also der 2 ("Zwei") des Dezimalsystems. Der 3 des Dezimalsystems entspricht die 11 ("Eins-Eins") des Dualsystems. Für die Dualzahl, die der Dezimalzahl 4 entspricht, werden bereits drei Stellen beansprucht: 100 ("Eins-Null-Null").

Suchen wir nach der nächstgrößeren Dualzahl anhand einer Potenzentabelle, wie wir sie vom Dezimalsystem her kennen. Wie schon gesagt: Das Dezimalsystem hat die Basis 10 (Zehn), und jede Ziffernstelle ist eine Potenz der Basis 10 (Zehn). Dementsprechend hat das Dualsystem die Basis - schreiben wir sie als Dezimalzahl - 2 (Zwei): jede Ziffernstelle ist also eine Potenz zur Basis 2 (Zwei). Und so sieht die Zweierpotenzentabelle zum Dualsystem aus - mit deren Unterstützung wir gleich die Dualzahl 101 ("Eins-Null-Eins") in die entsprechende Dezimalzahl umrechnen:

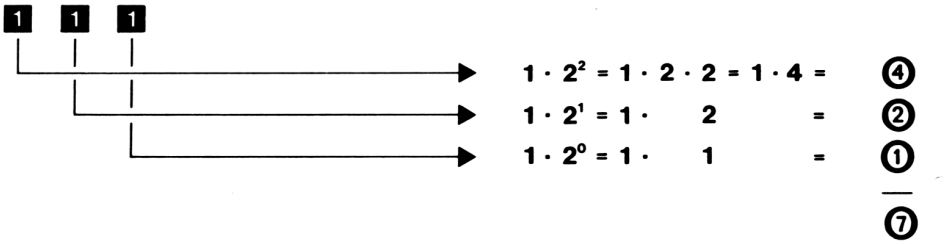
2^4	2^3	2^2	2^1	2^0
16	8	4	2	1



101 ("Eins-Null-Eins") im Dualsystem entspricht also 4 + 0 + 1 = 5 im Dezimalsystem.

Hier eine weitere Umwandlung, dargestellt in ihren einzelnen Schritten:

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1



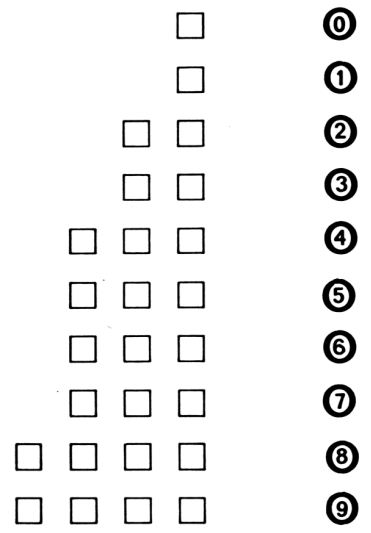
Übrigens sei nebenbei bemerkt: Besteht die Gefahr, Zahlen verschiedener Zahlensysteme zu verwechseln, so kann man die Zahlen mit einer "Tiefzahl" kennzeichnen, die die Basis des Systems angibt:

101_2	=	5_{10}
	aber	
101_2	≠	101_{10}

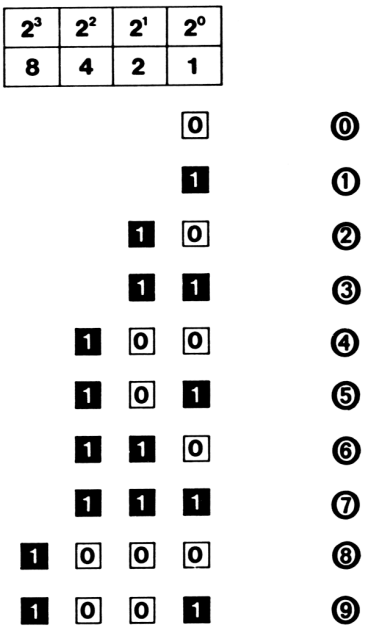
So kann also z.B. "Eins-Null-Eins" nicht mit "Hunderteins" verwechselt werden.
Hier nun eine Aufgabe für Sie:
Schreiben Sie in die Kästchen die Dualzahlen, die den Dezimalzahlen 0 bis 9 entsprechen (in jedes Kästchen eine Dualziffer). Wenn Sie sich an die Zweierpotenzentabelle halten, wird es Ihnen nicht schwerfallen.

Aufgabe

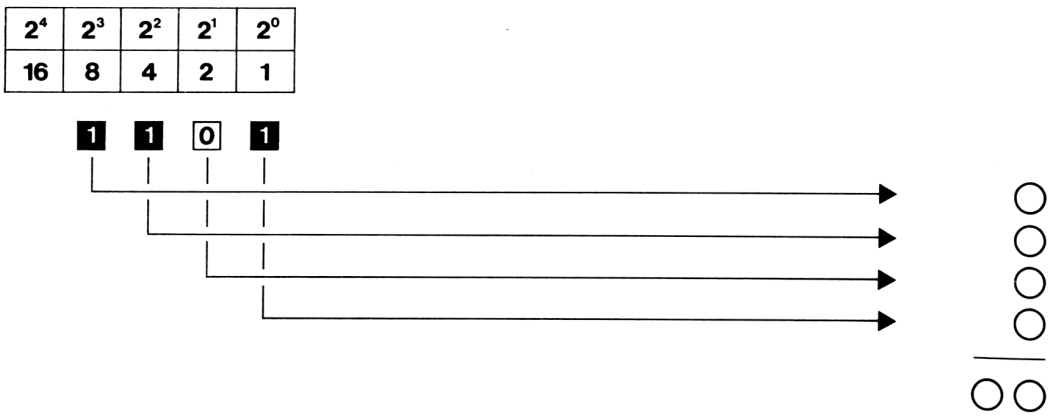
2^3	2^2	2^1	2^0
8	4	2	1



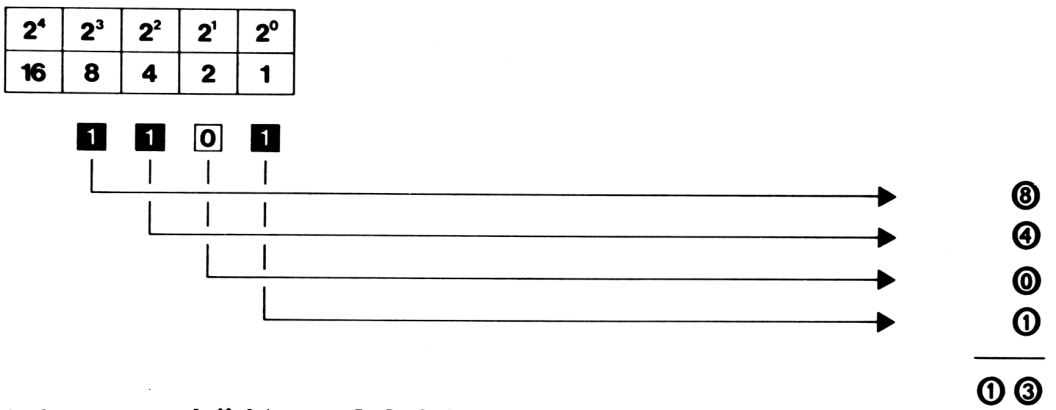
Lösung



Welcher Dezimalzahl entspricht die Dualzahl 1101? - Errechnen Sie die Dezimalzahl:

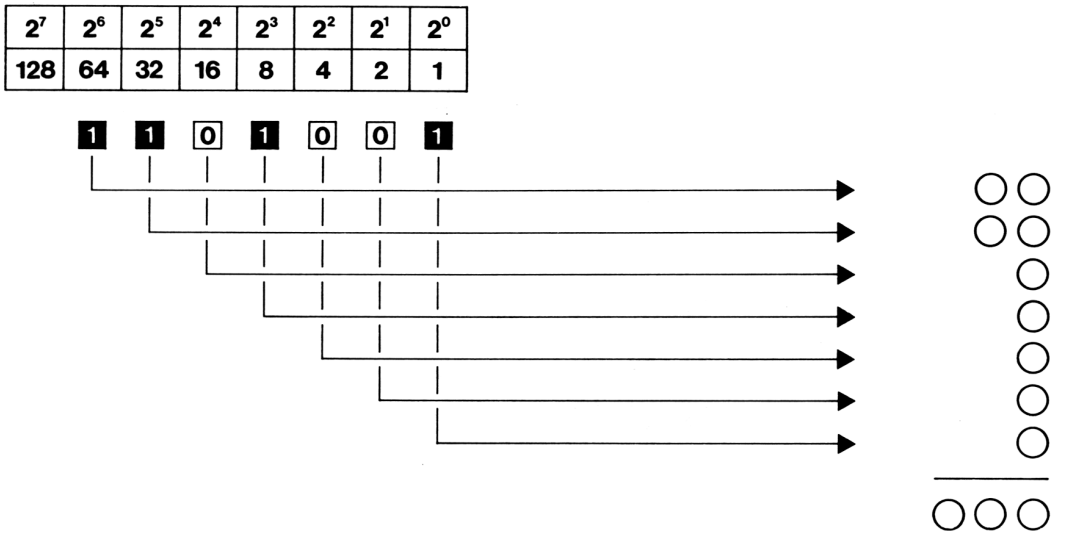


Und hier die Lösung:

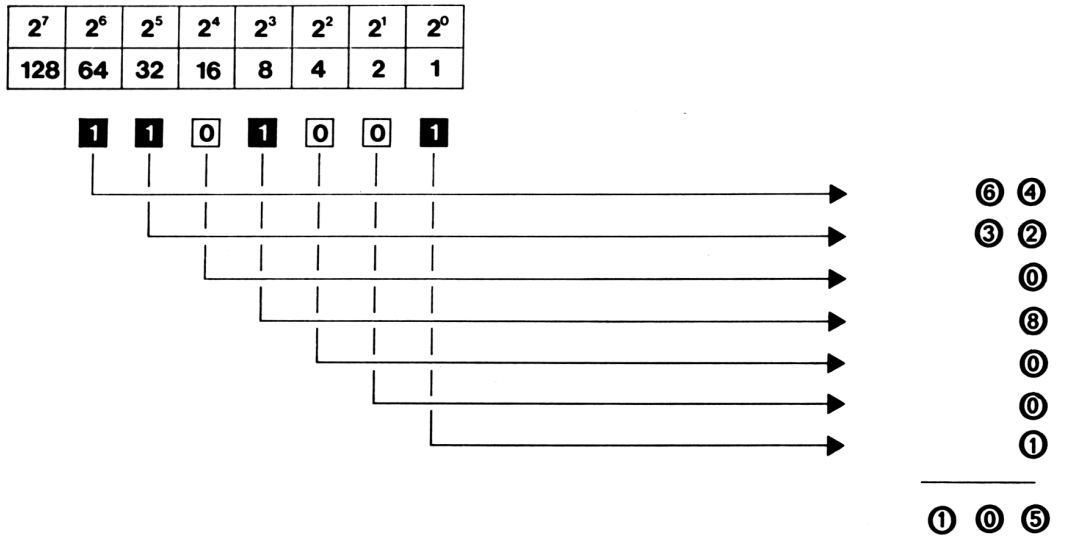


Anders ausgedrückt: $1\ 1\ 0\ 1_2 = 13_{10}$

Wie heißt die Dezimalzahl, die der Dualzahl 1 1 0 1 0 0 1 entspricht?



So ist es richtig:



Also: $1\,1\,0\,1\,0\,0\,1_2 = 105_{10}$

Und wie wird nun umgekehrt zu einer Dezimalzahl die entsprechende Dualzahl gefunden? Z.B :

$$86_{10} = ?_2$$

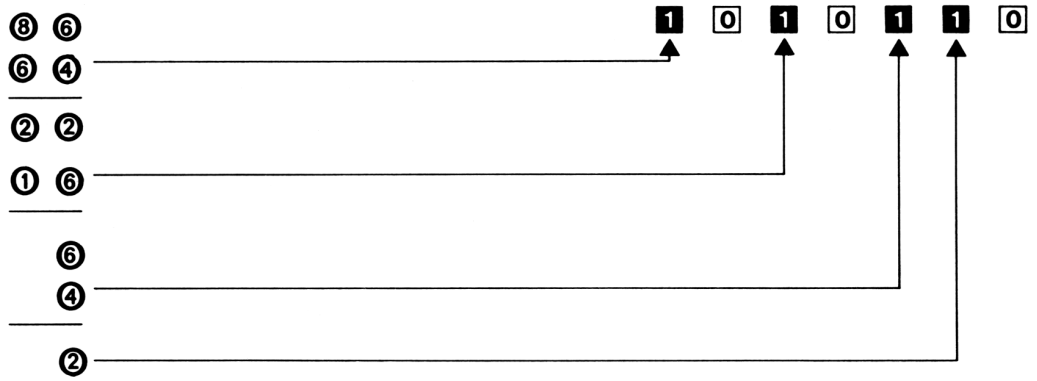
Sollten Sie sich die Umrechnung schon zutrauen, obwohl wir sie noch nicht behandelt haben, dann versuchen Sie es doch einmal. Benutzen Sie dazu die Zweierpotenzentabelle:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Das Ergebnis: $86_{10} = 1\,0\,1\,0\,1\,1\,0_2$

Haben Sie das Ergebnis selbst gefunden? - Wenn nein, macht es nichts.
Wenn ja, um so beachtlicher!

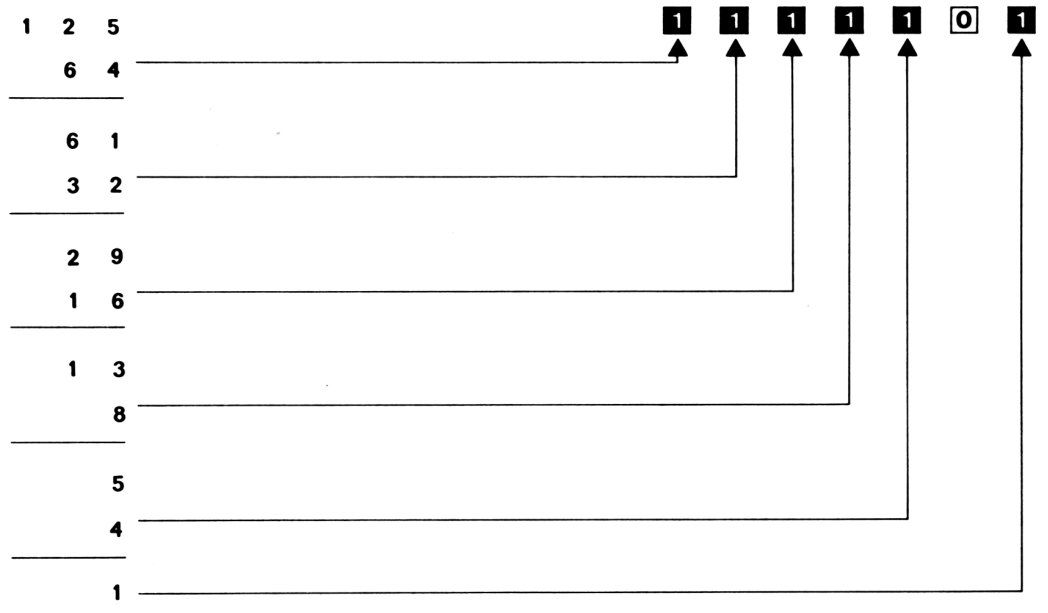
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



Zunächst wird die größte Zweierpotenz festgestellt, die in 86 enthalten ist, das ist 64. Also wird an der Wertstelle 64 eine 1 notiert. Bleibt ein Rest von 22; 32 ist nicht in 22 enthalten; deshalb wird eine 0 für die Wertstelle 32 notiert. Aber 16 ist in 22 enthalten; also eine 1 an die Wertstelle 16. 22 - 16 = 6; 6 enthält nullmal die 8, aber einmal die 4 und einmal die 2, womit der Rest untergebracht wäre und für die verbleibende Wertstelle 1 nur noch eine 0 zu notieren bliebe.

Ermitteln Sie nun bitte die
Dualzahl zur Dezimalzahl 125:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



Damit wollen wir die Umwandlung von Dezimalzahlen in Dualzahlen und von Dualzahlen in Dezimalzahlen abschließen.

Sie kennen jetzt den Aufbau des Dualsystems und staunen nachträglich vielleicht, wie klar dieses System im Grunde ist.

Abschließend noch ein wichtiger Begriff:
Eine duale Ziffernstelle - egal, ob sie die Ziffer 0 oder 1 enthält - wird auch Bit genannt. Bit ist eine Abkürzung des englischen Ausdrucks binary diget, was binäre Ziffer - duale Ziffer bzw. Ziffernstelle heißt. Ein Bit ist auch jedes technische Element, das die Zustände 0 oder 1 annehmen kann. So ist z.B. das Speicher-Flipflop ein Speicherelement, das ein Bit speichern kann, und dieses Bit kann entweder 0 oder 1 sein.

Wird Bit als Maßeinheit verwendet, so wird es klein geschrieben: bit. Die Zahl 1001 beispielsweise ist 4 bit groß. Das ist wiederum gleichbedeutend mit: Die Zahl 1001 besteht aus 4 Bits.

12.2. Zahlenraten

Fertigen Sie sich fünf Karten an, die die Zahlen 1 - 31 wie folgt enthalten:

Karte 1			
3	17	25	13
7	29	23	9
11	21	31	5
15	27	19	1

Karte 2			
3	14	7	27
18	31	23	15
6	26	11	19
1o	22	3o	2

Karte 3			
5	14	7	2o
28	22	3o	12
6	31	23	29
13	15	21	4

Karte 4			
9	28	12	24
15	25	31	1o
11	29	3o	14
26	13	27	8

Karte 5			
17	25	19	24
22	29	27	21
28	26	31	18
2o	3o	23	16

Bauen Sie nun aus Ihren Logik-Bausteinen fünf Speicher-Flipflops und verbinden Sie diese Speicher-Flipflops mit der Eingabeeinheit gemäß Abbildung.

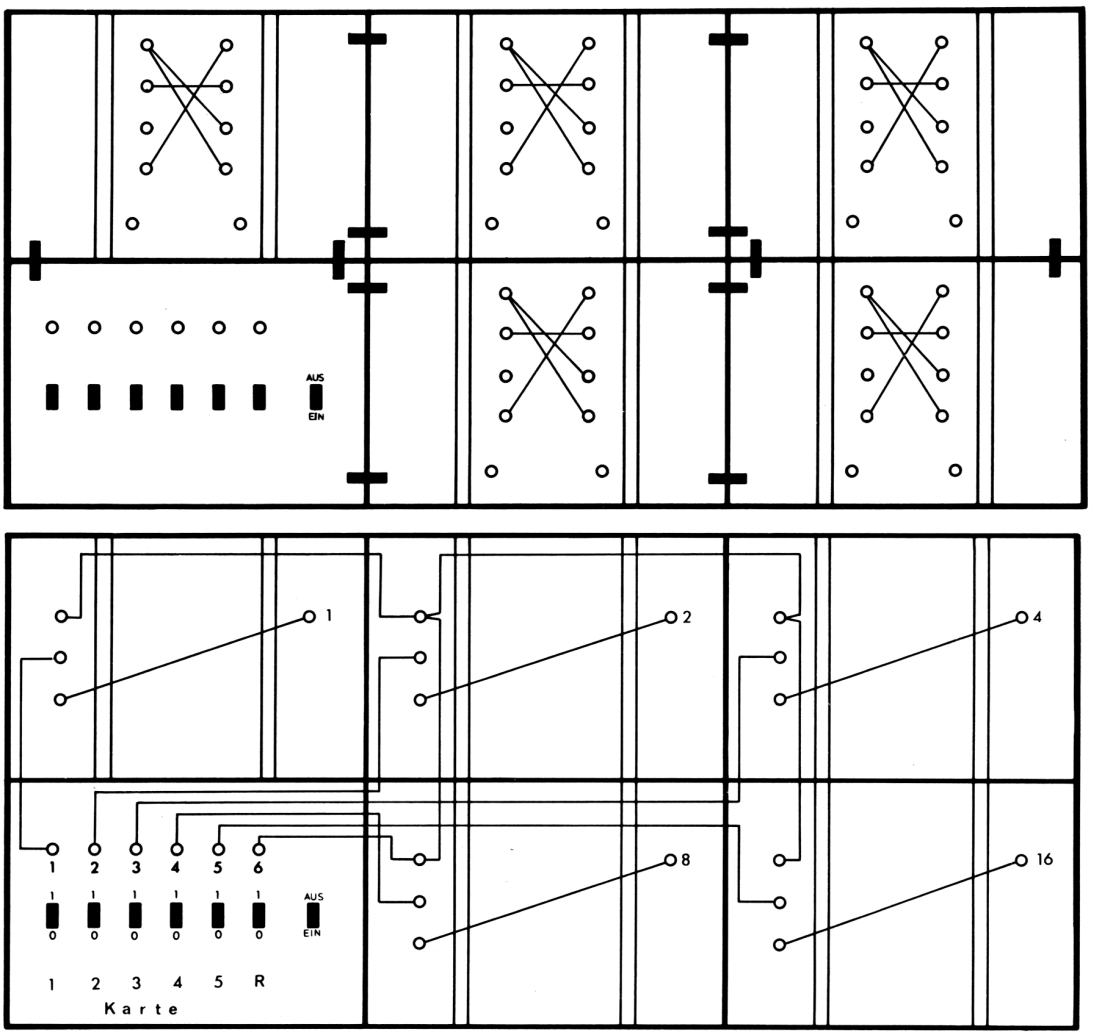


Abb.76

Jedes Speicher-Flipflop hat einen eigenen Set-Schalter, alle fünf Speicher-Flipflops haben zusammen einen Reset-Schalter. Den fünf Speicher-Flipflops ordnen wir die Wertigkeiten 1, 2, 4, 8 und 16 zu, entsprechend den Stellenwertigkeiten im Dualsystem. Dabei spielt es keine Rolle, daß wir die Stellenwertigkeiten von oben nach unten und nicht, wie im Kapitel über das Dualsystem, von rechts nach links aufgeführt haben.

Bitten Sie nun einen Mitspieler - am besten jemanden, der weder Ihren Computer noch das Dualsystem kennt - sich eine Zahl zwischen 1 und 31 zu denken. Legen Sie dem Mitspieler dann der Reihe nach die Karten 1 bis 5 vor und fragen ihn jeweils, ob die gezeigte Karte die gedachte Zahl enthält. Antwortet der gefragte Mitspieler mit JA, so wird kurz der Schalter, der der entsprechenden Karte zugeordnet ist, auf 1 gesetzt (set). Nach Abfrage aller Karten wird die gedachte Zahl von dem Computer dual angezeigt. Wie Sie die angezeigte Zahl ins Dezimalsystem übertragen, haben

Sie bereits kennengelernt. Das Ergebnis ist jedenfalls die vom Mitspieler gedachte Zahl.
Vielleicht führen Sie das Experiment zunächst einmal mit sich allein durch. Sicher werden auch Sie erstaunt sein, wieso Sie durch dieses Verfahren auf die Lösung kommen können. Und dies ist das Geheimnis der Zahlenkarten:
Auf den Karten sind die Zahlen so geordnet, daß alle diejenigen, die in der dualen Darstellung in der ersten Stelle eine 1 haben, auf Karte 1 abgebildet sind. Diejenigen Zahlen, die in der dualen Darstellung eine 1 in der zweiten Stelle haben, sind auf Karte 2 enthalten. Alle Zahlen mit einer 1 an der dritten dualen Stelle sind auf Karte 3 usw. Wenn nun bekannt ist, auf welchen Karten die gedachte Zahl vorhanden ist,
so ist die duale Darstellung bekannt und kann ins Dezimalsystem umgerechnet werden.

dezimal	dual
1	1
2	1 o
3	1 1
4	1 o o
5	1 o 1
6	1 1 o
7	1 1 1
8	1 o o o
9	1 o o 1
1 o	1 o 1 o
1 1	1 o 1 1
1 2	1 1 o o
1 3	1 1 o 1
1 4	1 1 1 o
1 5	1 1 1 1
1 6	1 o o o o
1 7	1 o o o 1
1 8	1 o o 1 o
1 9	1 o o 1 1
2 o	1 o 1 o o
2 1	1 o 1 o 1
2 2	1 o 1 1 o
2 3	1 o 1 1 1
2 4	1 1 o o o
2 5	1 1 o o 1
2 6	1 1 o 1 o
2 7	1 1 o 1 1
2 8	1 1 1 o o
2 9	1 1 1 o 1
3 o	1 1 1 1 o
3 1	1 1 1 1 1
	5 4 3 2 1 Stelle

12.3. Rechnen mit Dualzahlen ist denkbar einfach, denn es gibt ja nur die beiden Ziffern 0 und 1. Nur weil es Ihnen ungewohnt ist, werden Sie vielleicht anfangs etwas mehr Mühe haben als beim Rechnen mit Dezimalzahlen (nebenbei: denken Sie immer an die Lesart der Dualzahlen, also z.B.: 1 + 1 = 10; Eins + Eins = Eins-Null).

Addition

Bevor wir Dualzahlen addieren, erinnern wir uns zuerst an die Dezimaladdition. Zwei oder mehr Summanden werden zu einer Summe addiert. Wenn die Summenbildung auf einer Dezimalstelle den Wert 9 überschreitet, entsteht ein Übertrag auf die linke Nachbarstelle; bei dieser Nachbarstelle wird der Übertrag mitaddiert, ein Beispiel:

	1	5	7	Summand	A
+	5	4	3	Summand	B
	1	1		Überträge	
	7	(1) 0	(1) 0	Summe	

Bei der Addition von Dualzahlen wird entsprechend verfahren. Ein Übertrag auf die linke Nachbarstelle entsteht allerdings bereits, wenn die Summenbildung auf einer Dualstelle den Wert 1 überschreitet.

	16	8	4	2	1			
		1	1	1	0	Summand	A	(14)
+		1	1	0	1	Summand	B	(13)
	1	1				Überträge		
=	1	(1) 1	(1) 0	1	1	Summe		(27)

Nun versuchen Sie doch einmal, die folgende Aufgabe selbst zu lösen. Sehen Sie sich bitte nicht gleich die nachfolgende Lösung an, sondern probieren Sie erst einmal selbst. Zunächst können Sie Ihre eigene Lösung dadurch kontrollieren, indem Sie zu den Dualzahlen die Dezimalzahlen errechnen und die Summen vergleichen:

	32	16	8	4	2	1		
			1	1	1	1	Summand	A
+		1	1	0	1	0	Summand	B
							Überträge	
=							Summe	

	32	16	8	4	2	1			
			1	1	1	1	Summand	A	(15)
+		1	1	0	1	0	Summand	B	(26)
+	1	1	1	1			Überträge		
=	1	0	1	0	0	1	Summe		(41)

Subtraktion

Auch hier zunächst eine kurze Erinnerung an das Dezimalsystem. Von einem Minuenden (das ist die Zahl, von der eine andere Zahl abgezogen, also subtrahiert wird)wird ein Subtrahend (die Zahl, die abzuziehen ist) subtrahiert. Das ergibt die Differenz. Wenn auf einer Dezimalstelle der Wert des Subtrahenden größer ist als der Wert des Minuenden, muß von der linken Nachbarstelle "geborgt" werden; die geborgte Zahl wird als Übertrag auf der linken Nachbarstelle mit subtrahiert. Ein Beispiel:

	1 (1)	8	Minuend	A
–		9	Subtrahend	B
	1			
=		9	Differenz	

Die Subtraktion von Dualzahlen geschieht nach denselben Regeln.

	16	8	4	2	1			
	1 (1)	0 (1)	0	1	1	Minuend	A	(19)
–		1	1	1	0	Subtrahend	B	(14)
	1	1						
=			1	0	1	Differenz		(5)

Wollen Sie diese Aufgabe selbst lösen?
Sie können sich ja wieder dadurch kontrollieren, indem Sie zu den Dualzahlen die Dezimalzahlen errechnen und dann die Ergebnisse der Rechnungen vergleichen.

	64	32	16	8	4	2	1		
	1	0	1	0	1	0	1	Minuend	A
–		1	0	1	0	1	0	Subtrahend	B
								Überträge	
=								Differenz	

12.4. Addition

Im Kapitel 12.3. haben Sie das Rechnen mit Dualzahlen gelernt. Da das Dualsystem nur die Werte 0 und 1 kennt, lassen sich auch Ihre elektronischen Logik-Bausteine zu einem kleinen Rechner programmieren.
Sollen z.B. 2 einstellige Dualzahlen mit Logik-Bausteinen addiert werden, so müssen Sie sich, um die Programmierung zu finden, zunächst noch einmal die Rechenregeln der Addition ansehen.
Im Dualsystem ergibt

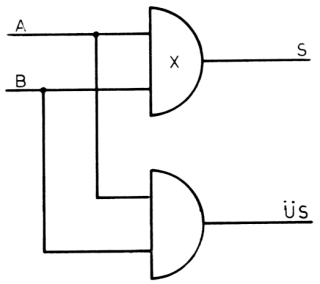
1. Dualzahl	+	2. Dualzahl	=	Summe	und Übertrag
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

Wird die erste Dualzahl mit A und die zweite mit B bezeichnet, läßt sich aus diesem logischen Zusammenhang eine Funktionstabelle für die Summe S und den Übertrag ÜS aufstellen.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

A	B	ÜS
0	0	0
0	1	0
1	0	0
1	1	1

Sie erkennen, daß bei der Addition die Summe S mit der Exklusiv-ODER-Funktion und der Übertrag ÜS mit der UND-Funktion gebildet werden können.



Diese einfachste Art eines Rechners nennt man Halbaddierer. Die Programmierung der zwei Logik-Bausteine läßt sich nach dem Ihnen bekannten Schema (Kapitel 7) leicht finden.

Programmierung Baustein S

A	B	F	
0	0	0	→ f1 = 0
0	1	1	→ f2 = 1
1	0	1	→ f3 = 1
1	1	0	→ f4 = 0

Programmierung Baustein ÜS

A	B	F	
0	0	0	→ f1 = 0
0	1	0	→ f2 = 0
1	0	0	→ f3 = 0
1	1	1	→ f4 = 1

Schaltung des Halbaddierers

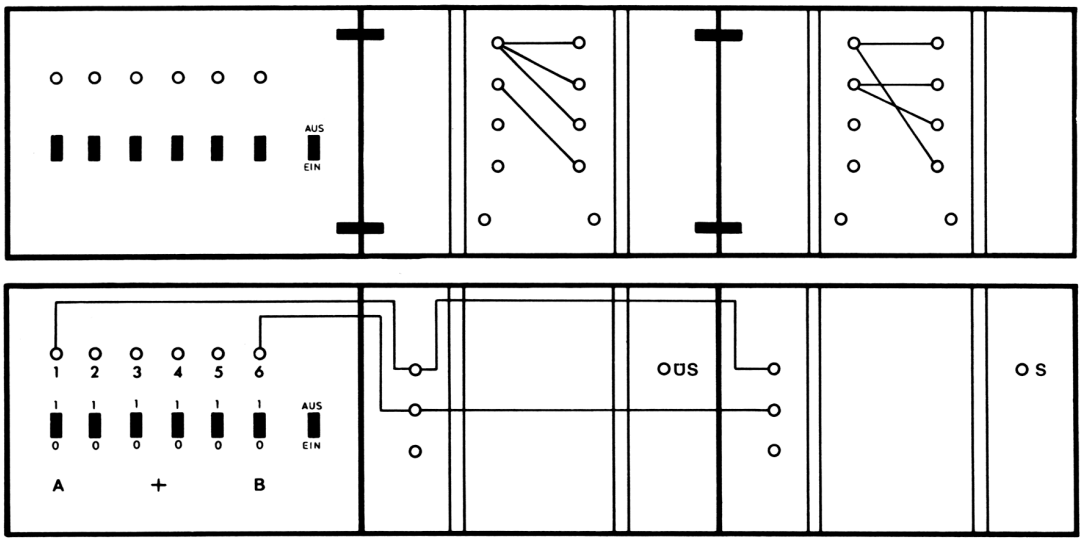


Abb.77

Beispiel:

Addieren Sie mit der Schaltung die Dualzahlen 0 + 1, 1 + 1, 1 + 0, 0 + 0 und überprüfen Sie das Ergebnis.

Wie Sie festgestellt haben, lassen sich mit dieser Schaltung jedoch nur einstellige Dualzahlen addieren.

Wenn beispielsweise neben den Dualziffern A und B auch noch der Übertrag Ü aus dem vorangegangenen Rechengang berücksichtigt werden soll, muß der Halbaddierer zu einem Volladdierer ausgebaut werden. Mit Hilfe der Funktionstabellen für die Summe S und den Übertrag ÜS läßt sich die Programmierung für den Volladdierer finden.

Funktionstabelle für Summe S

1. Dualziffer + 2. Dualziffer + Übertrag A B Ü A B C	Summe S F	
0 0 0	0	
0 0 1	1	→ f1 = C
0 1 0	1	
0 1 1	0	→ f2 = $\bar{C}$
1 0 0	1	
1 0 1	0	→ f3 = $\bar{C}$
1 1 0	0	
1 1 1	1	→ f4 = C

Funktionstabelle für den Übertrag ÜS

1. Dualziffer + 2. Dualziffer + Übertrag			Übertrag	
A	B	Ü	ÜS	
A	B	C	F	
0	0	0	0	
0	0	1	0	→ f1 = 0
0	1	0	0	
0	1	1	1	→ f2 = C
1	0	0	0	
1	0	1	1	→ f3 = C
1	1	0	1	
1	1	1	1	→ f4 = 1

Und so sieht die Verdrahtung des Volladdierers aus:

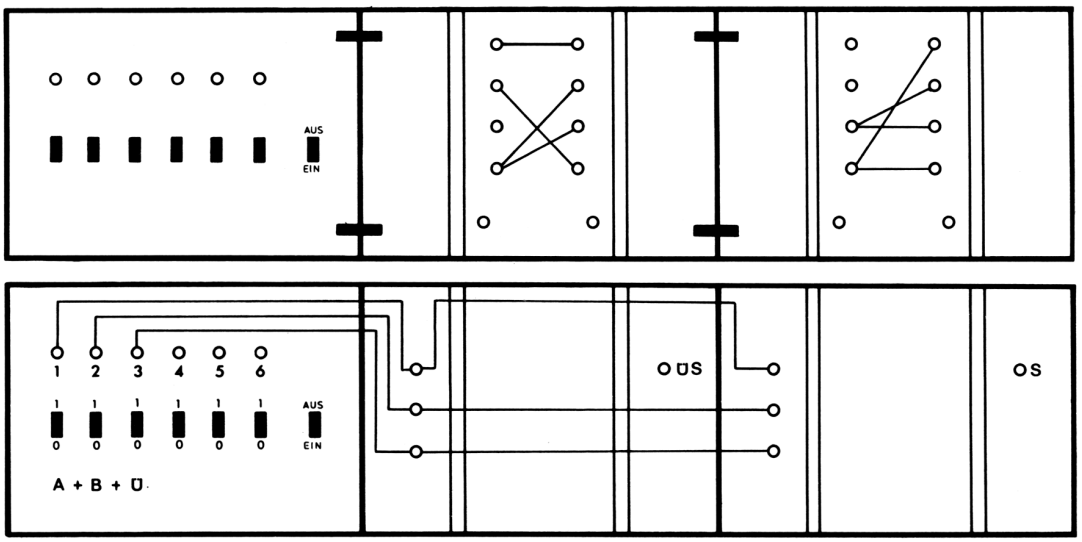
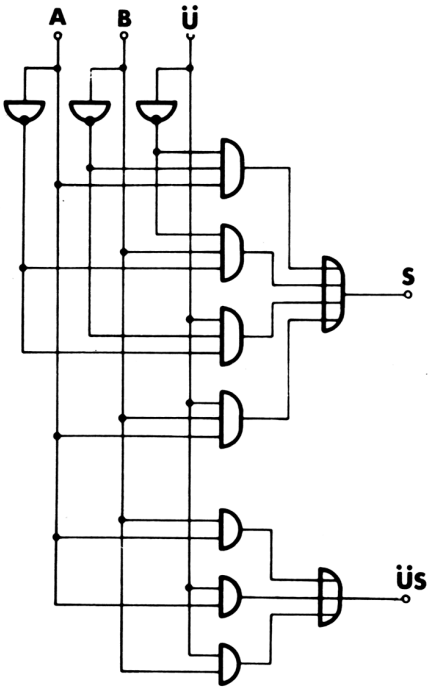


Abb.78

Logikplan des Volladdierers



Ein Beispiel:
Es sollen die Dualzahlen 1011 und 111 addiert werden (11 + 7 dezimal). Wir beginnen mit der rechten Dualstelle, der Stelle mit der geringsten Wertigkeit. Dabei bedeutet A die jeweilige Stelle der ersten Dualzahl, B bedeutet die jeweilige Stelle der zweiten Dualzahl. ÜS bedeutet ein Übertrag, der bei der Summenbildung entsteht, Ü ein Übertrag, der am Eingang mit zu berücksichtigen ist, der also dem Übertrag der vorangegangenen Addition entspricht. S ist die Summe der beiden addierten Dualziffern.

Erster Schritt:	A = 1,	B = 1,	Ü = 0	-	Ergebnis:	S = 0,	ÜS = 1
zweiter "	: A = 1,	B = 1,	Ü = 1	-	"	: S = 1,	ÜS = 1
dritter "	: A = 0,	B = 1,	Ü = 1	-	"	: S = 0,	ÜS = 1
vierter "	: A = 1,	B = 0,	Ü = 1	-	"	: S = 0,	ÜS = 1
fünfter "	: A = 0.	B = 0,	Ü = 1	-	"	: S = 1,	ÜS = 0

Gesamtergebnis: 10010
(18 dezimal)

Bei dem hier gezeigten Additions-Rechenwerk müssen Sie dem Computer etwas unter die Arme greifen. Die Ergebnisse und die jeweiligen Überträge jeder Ziffernstelle werden extern zwischengespeichert; d.h. sie müssen auf einem Zettel notiert werden. Die Summenziffern notieren Sie, um so schrittweise das Endergebnis zu erhalten. (Selbstverständlich von rechts nach links, entsprechend der Stellenwertigkeit der Ziffern.) Die Ausgangsüberträge ÜS notieren Sie, um sie bei der Addition der nächsthöheren Ziffernstelle mit zu berücksichtigen.

Einen Addierer, bei dem die Dualzahlen parallel eingegeben werden und der die Summe aus 2 zweistelligen Dualzahlen direkt anzeigt, erhalten Sie, wenn ein Halbaddierer und ein Volladdierer nach der folgenden Abbildung in Reihe geschaltet werden.

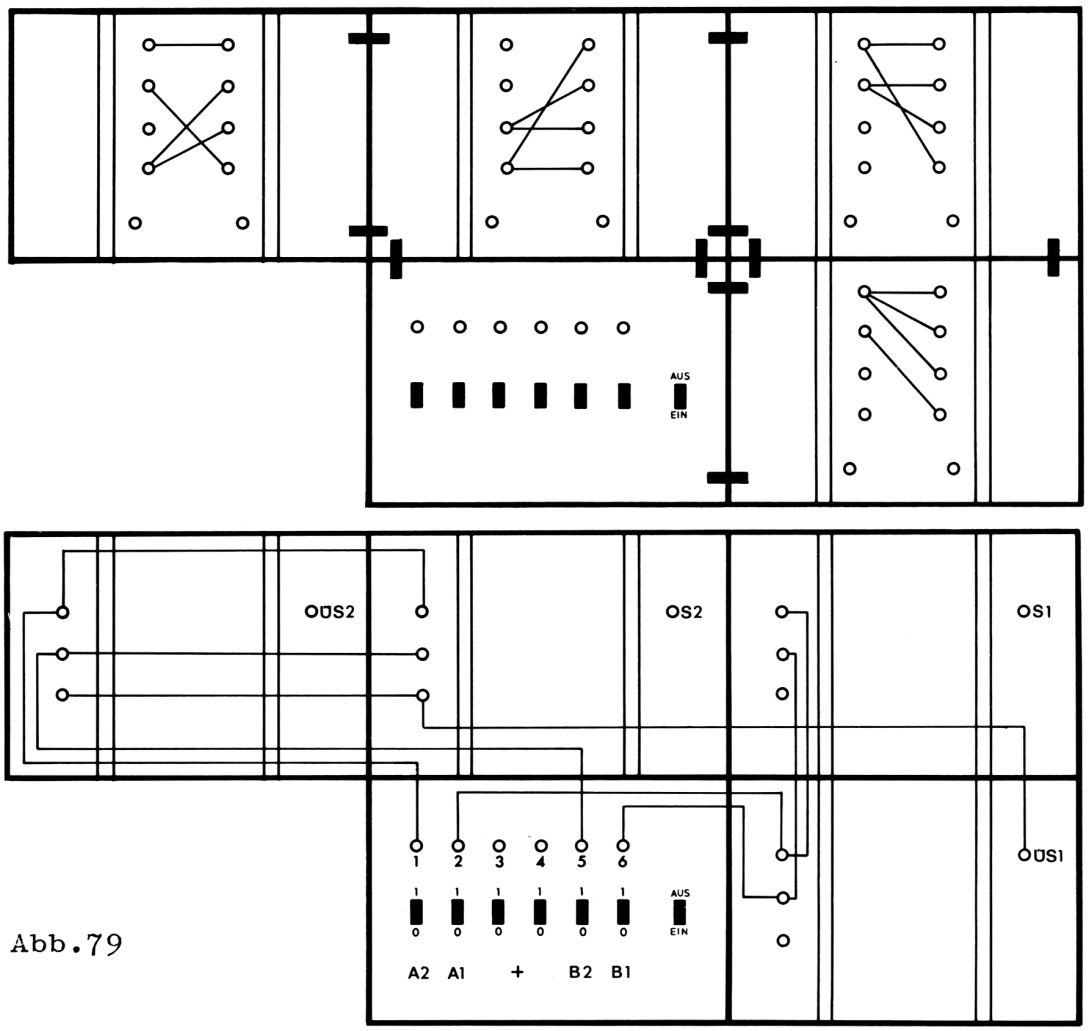


Abb.79

Wenn Sie das Ergebnis der Addition ablesen wollen, dürfen Sie den Übertrag ÜS1 nicht mit berücksichtigen, da er bei der Summenbildung des Volladdierers (S2, ÜS2) mit in die Anzeige eingeht.

Ein kleines Rechenbeispiel:
Mit A1 und A2 geben Sie die erste Dualzahl - z.B. 10 - ein und mit B1 und B2 die zweite - angenommen sie sei 11. Im Dezimalsystem entspricht das der Rechnung 2 + 3 = 5. Die Zahl 5 im Dezimalsystem entspricht 101 im Dualsystem. Sie stellen fest, daß die Anzeigelampe S1 = 1, S2 = 0 und ÜS1 = 1 sind und können damit das Ergebnis direkt ablesen. Die Umrechnung vom Dual- in das Dezimalsystem wird erleichtert, wenn Sie den Ausgängen S1 den Wert 1 (dezimal), S2 den Wert 2 und ÜS3 den Wert 4 zuordnen. Da in diesem Beispiel S1 = 1 (dual) und ÜS3 = 1 (dual) sind, ist das Ergebnis 1 + 4 = 5. Dieser Addierer in Parallelbetrieb kann durch Reihenschaltung von weiteren Volladdierern in seiner Kapazität vergrößert werden.

12.5. Subtraktion

Kurz zusammengefaßt hier noch einmal die Rechenregeln für das Subtrahieren von Dualzahlen:

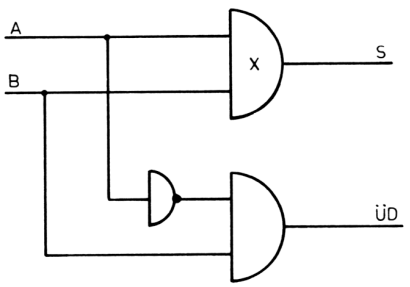
1. Dualzahl	-	2. Dualzahl	=	Differenz	Übertrag
0	-	0	=	0	0
0	-	1	=	1	1
1	-	0	=	1	0
1	-	1	=	0	0

Daraus lassen sich die Funktionstabellen für die Differenz D und für den Übertrag ÜD ableiten, wobei in den Tabellen die erste Dualzahl mit A und die zweite mit B bezeichnet werden.

A	B	D
0	0	0
0	1	1
1	0	1
1	1	0

A	B	ÜD
0	0	0
0	1	1
1	0	0
1	1	0

Die Differenz kann wie bei der Addition mit der Exklusiv-ODER-Funktion gebildet werden, der Übertrag ÜD dagegen aus einem UND mit negiertem A-Eingang. So sieht der Logikplan des Halbsubtrahierers aus:



Soll dieser Logikplan mit zwei Logik-Bausteinen verwirklicht werden, ergibt sich folgende Programmierung:

Programmierung Baustein D

A	B	F	
0	0	0	→ f1 = 0
0	1	1	→ f2 = 1
1	0	1	→ f3 = 1
1	1	0	→ f4 = 0

Programmierung Baustein ÜD

A	B	F	
0	0	0	→ f1 = 0
0	1	1	→ f2 = 1
1	0	0	→ f3 = 0
1	1	0	→ f4 = 0

Verdrahtungsplan des Halbsubtrahierers

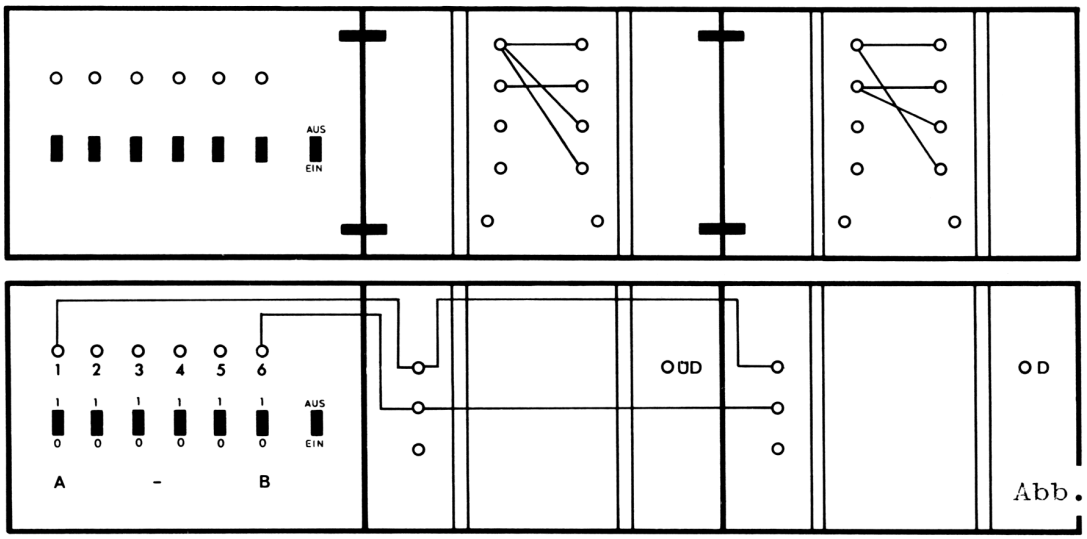


Abb.8o

Kontrollieren Sie die Schaltung mit Hilfe der Rechenregeln für die Subtraktion. Sollen neben der eigentlichen Subtraktion A minus B noch der Übertrag Ü aus dem vorangegangenen Rechenvorgang berücksichtigt werden, muß die Programmierung für einen Vollsubtrahierer gefunden werden.

Funktionstabelle und Programmierung für die Differenz D

1. Dualziffer A A	- 2. Dualziffer B B	- Übertrag Ü C	Differenz D F	
0	0	0	0	
0	0	1	1	→ f1 = C
0	1	0	1	
0	1	1	0	→ f2 = $\bar{C}$
1	0	0	1	
1	0	1	0	→ f3 = $\bar{C}$
1	1	0	0	
1	1	1	1	→ f4 = C

Funktionstabelle und Programmierung für den Übertrag ÜD

1. Dualziffer A A	- 2. Dualziffer B B	- Übertrag Ü C	Übertrag ÜD F	
0	0	0	0	
0	0	1	1	→ f1 = C
0	1	0	1	
0	1	1	1	→ f2 = 1
1	0	0	0	
1	0	1	0	→ f3 = 0
1	1	0	0	
1	1	1	1	→ f4 = C

Die Schaltung des Vollsubtrahierers aus 2 Logik-Bausteinen:

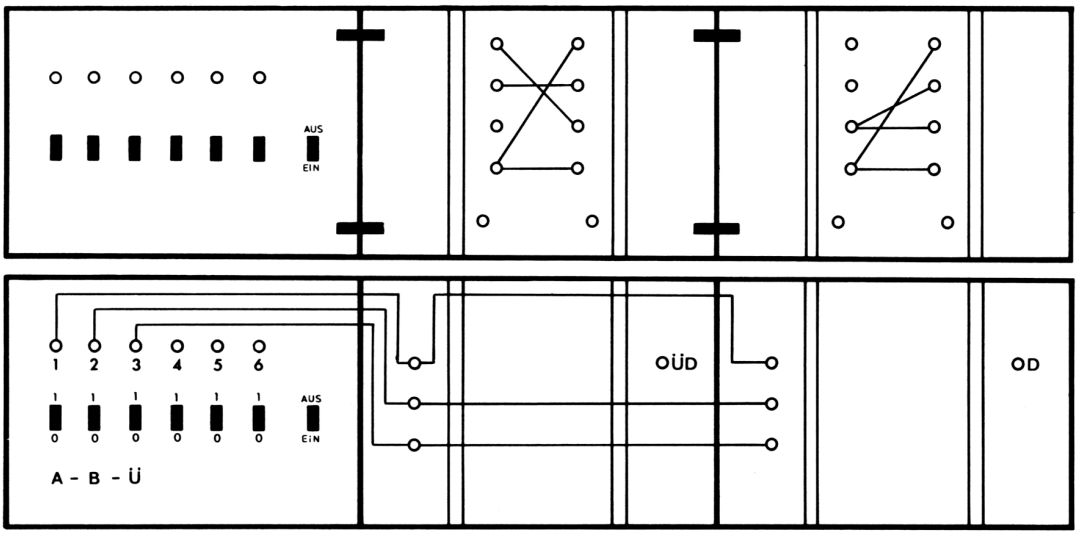


Abb.81

Beispiel: Es soll die Dualzahl 0111 (7) von der Dualzahl 1011 (11) abgezogen werden.

Erster Schritt:	A = 1,	B = 1,	Ü = 0	-	Ergebnis:	D = 0,	ÜD = 0
zweiter "	: A = 1,	B = 1,	Ü = 0	-	"	: D = 0,	ÜD = 0
dritter "	: A = 0,	B = 1,	Ü = 0	-	"	: D = 1,	ÜD = 1
vierter "	: A = 1,	B = 0,	Ü = 1	-	"	: D = 0,	ÜD = 0

Endergebnis: 0100 (4)

D bedeutet hier Differenz, ÜD ist der Übertrag, der bei der Differenzbildung entsteht; alle übrigen Bezeichnungen entsprechen denen der Addition.

Sie können sich denken, daß bei großen Computern die Ergebnisse der einzelnen Ziffernstellen sowie die Überträge im Computer selbst gespeichert werden. Daß Sie bei den Experimenten "externe Speicherung" mit hinzuziehen, tut dem Prinzip jedoch keinen Abbruch.

Die direkte Subtraktion von 2 zweistelligen Dualzahlen kann mit der Reihenschaltung aus einem Halb- und einem Vollsubtrahierer verwirklicht werden. Wie bei jeder Subtraktion von Dualzahlen muß der Minuend A größer als der Subtrahend B sein, da sonst das Ergebnis negativ wird und nicht ohne weiteres wieder in das Dezimalsystem umgewandelt werden kann. Der Parallelsubtrahierer in der folgenden Schaltung zeigt einen negativen Wert durch das Aufleuchten der Lampe ÜD2 an.

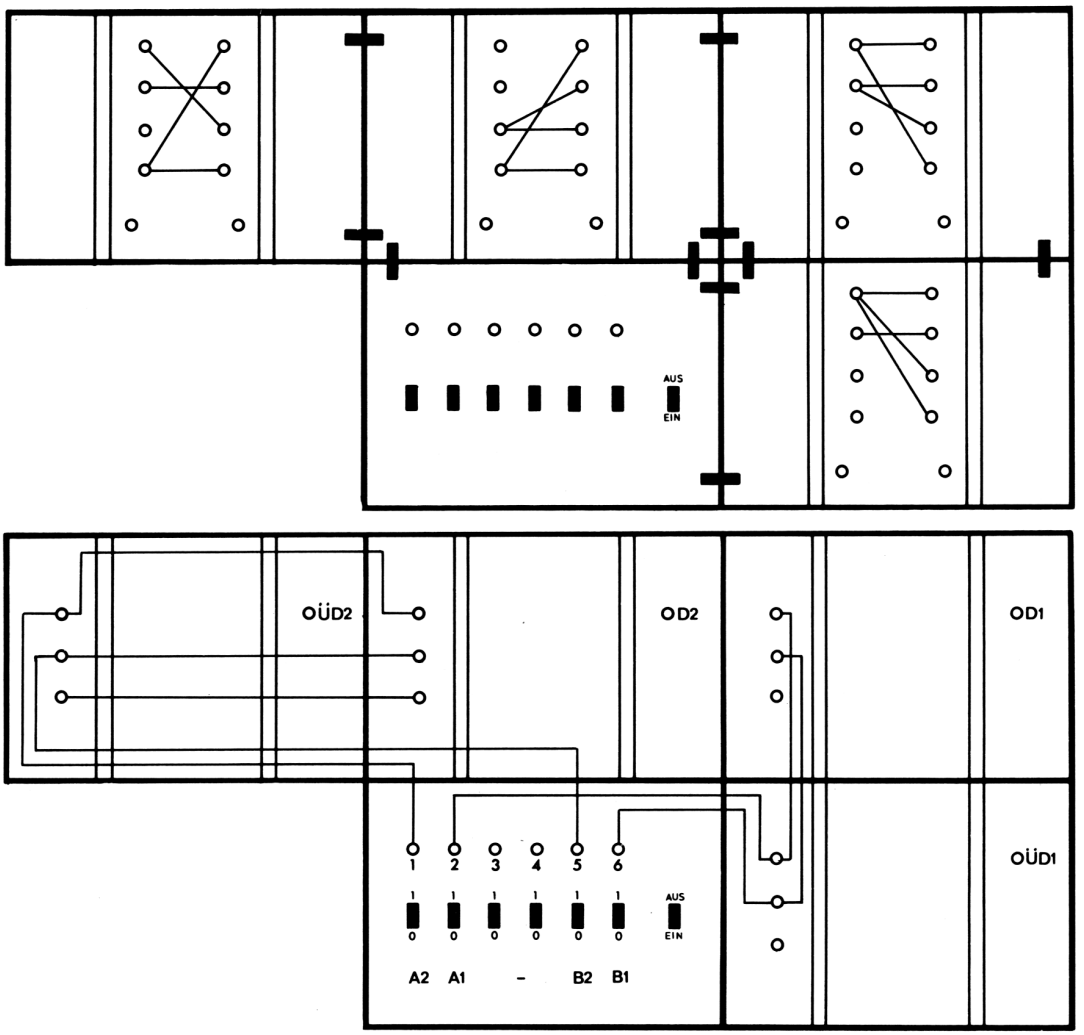


Abb.82

Beispiel:

dual	dezimal	ÜD2	D2	D1	
11 - 10 = 001	3 - 2 =	- 0	+ 0	+ 1 =	+ 1
01 - 01 = 000	1 - 1 =	- 0	+ 0	+ 0 =	0
10 - 01 = 001	2 - 1 =	- 0	+ 0	+ 1 =	+ 1
11 - 01 = 010	3 - 1 =	- 0	+ 2	+ 0 =	+ 2
11 - 00 = 011	3 - 0 =	- 0	+ 2	+ 1 =	+ 3

Sie können das Ergebnis leichter in das Dezimalsystem übertragen, wenn Sie dem Ausgang D1 die dezimale Ziffer +1 und D2 +2 zuordnen. Zeigt der Ausgang ÜD2 die duale Ziffer 1 an, ist das Ergebnis negativ, dabei ist die Wertigkeit dieser Anzeige (dezimal) -4.

Hier einige Beispiele, bei denen ein negativer Differenzbetrag auftritt:

dual	dezimal	ÜD2	D2	D1	
00 - 11 = 101	0 - 3 =	- 4	+ 0	+ 1 =	- 3
01 - 11 = 110	1 - 3 =	- 4	+ 2	+ 0 =	- 2
01 - 10 = 111	1 - 2 =	- 4	+ 2	+ 1 =	- 1

Wie auch beim Addierer im Parallelbetrieb läßt sich diese Subtrahierschaltung durch weitere Vollsubtrahierer vergrößern. Da für jede Dualstelle 2 Logik-Bausteine gebraucht werden, können Sie sich vorstellen, daß für den Rechenvorgang größerer Werte der Aufwand an Rechenwerken nicht mehr zu vertreten ist. Aus diesem Grund finden Sie in großen Computern fast nur sequentielle Addier- und Subtrahierschaltungen, die den Rechenvorgang schrittweise ausführen und die Teilergebnisse zwischenspeichern.

12.6. Sequentielle Addier- und Subtrahierschaltung

Um auch mit geringem Schaltungsaufwand große Zahlen addieren und subtrahieren zu können, bedient man sich sogenannter sequentieller Rechenwerke. Die Dualzahlen werden hier nicht wie bei dem Paralleladdierer und -subtrahierer in ihrem gesamten Wert in das Rechenwerk gegeben, sondern die einzelnen Dualziffern der Zahlen A und B werden zeitlich nacheinander addiert bzw. subtrahiert, wobei das jeweilige Ergebnis von einem Speicher übernommen wird.
Der nachfolgende Verdrahtungsplan zeigt eine Addierschaltung, bei der die Überträge ÜS in einen Zwischenspeicher gelangen und durch die Rückführung auf den Volladdierer im nachfolgenden Rechenvorgang automatisch berücksichtigt werden.

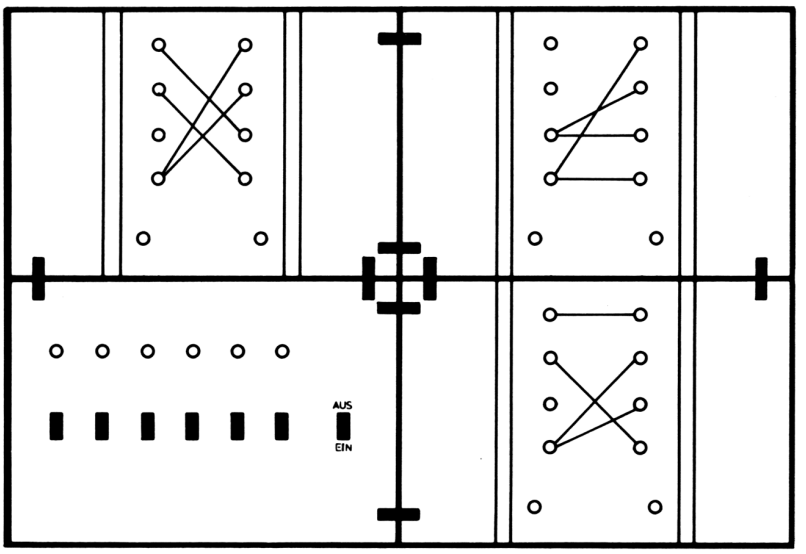


Abb.83

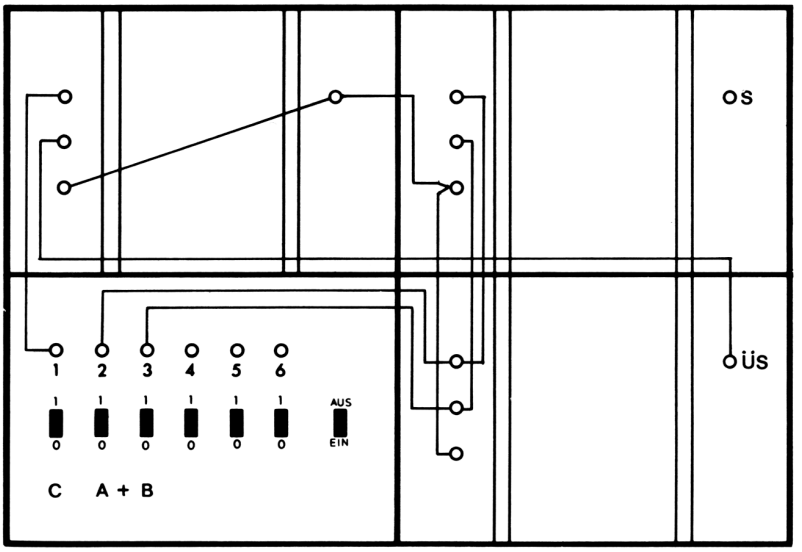


Abb. 83

Die jeweilige Speicherung des Übertrages ÜS ermöglicht das D-Auffang-Flipflop 1 (DFF1). In der Versuchsanordnung ist der linke Logik-Baustein als Flipflop programmiert, der rechte Schaltungsteil stellt den bekannten Volladdierer mit seinen Ausgängen S und ÜS dar.
Die Arbeitsweise ist folgende: Über die Schalter A und B geben Sie die ersten beiden Dualziffern (beachten Sie, daß hiermit die letzte Stelle der zu addierenden Dualzahlen gemeint ist) auf den Volladdierer, und es erscheint als erstes Teilergebnis die Summe S und der Übertrag ÜS. Als nächstes müssen Sie den Zwischenspeicher öffnen, indem Sie den Taktschalter C von 0 auf 1 setzen. Der Übertrag ÜS kann jetzt vom DFF1 übernommen werden. Wenn Sie nun den Schalter C wieder von 1 auf 0 zurückschalten, hat das D-Auffang-Flipflop den Übertrag gespeichert. Da der Ausgang des Speichers direkt mit dem Übertragseingang des Volladdierers verbunden ist, wird bei der nächsten Zifferneingabe der Übertrag automatisch berücksichtigt. Die jeweilige Summe S müssen Sie allerdings auf einem Zettel notieren.

Rechnen Sie folgendes Additionsbeispiel durch: 1011 + 0111
(dezimal 11 + 7)

Hier die einzelnen Rechenschritte:

Schalter A		Schalter B		Summe S	ÜS	Schalter C
1	+	1	=	0	1	0 - 1 - 0
1	+	1	=	1	1	1 - 0
0	+	1	=	0	1	1 - 0
1	+	0	=	0	1	

Das Ergebnis ist also 10010 (dezimal 18).

Die sequentielle Subtrahierschaltung arbeitet nach dem gleichen Prinzip.
Auch hier ein Rechenbeispiel: 1011 - 0111 (dezimal 11 - 7)

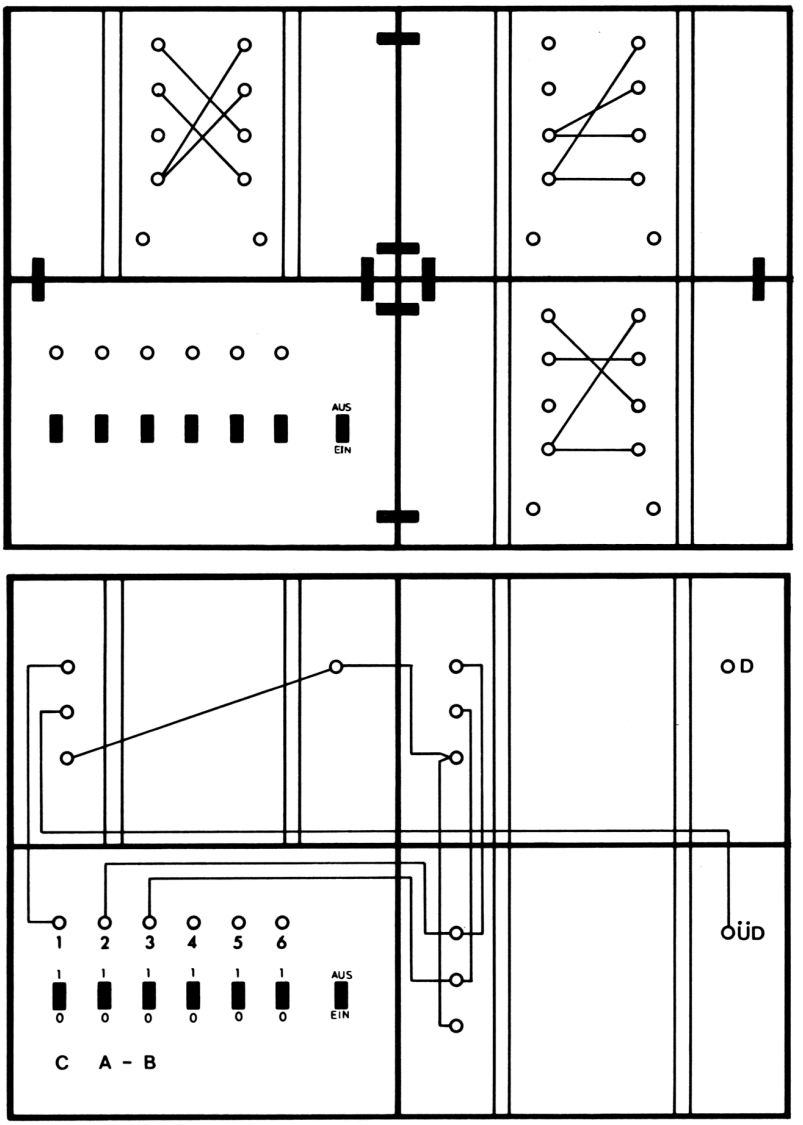


Abb.84

Schalter A		Schalter B		Differenz D	ÜD	Schalter C
1	-	1	=	0	0	0 - 1 - 0
1	-	1	=	0	0	1 - 0
0	-	1	=	1	1	1 - 0
1	-	0	=	0	0	

Ergebnis: 00100 (dezimal 4)

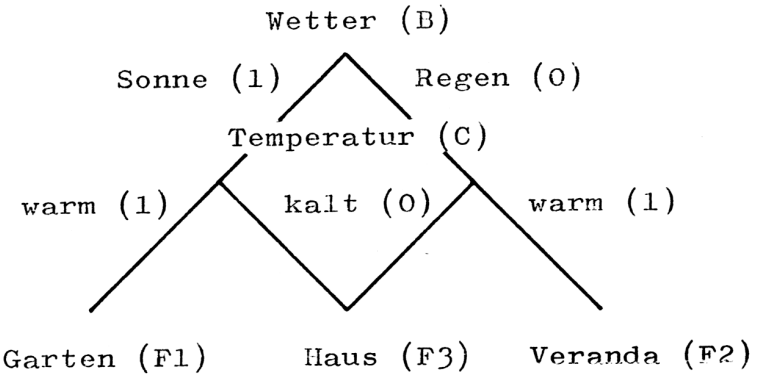
13. Anwendungsmöglichkeiten des CL 1601

13.1. Ein Party-Problem

Wir wollen eine Gartenparty feiern. Das ungewisse Wetter stellt uns vor folgendes Entscheidungsproblem:

1. Wenn die Sonne scheint und wenn es warm ist, dann feiern wir im Garten.
2. Wenn es regnet und wenn es warm ist, dann feiern wir in der Veranda.
3. Wenn es kalt ist, dann feiern wir in jedem Fall im Haus.

Wir können uns dieses Entscheidungsproblem übersichtlich als "Entscheidungsbaum" aufzeichnen.



Programmieren Sie Ihren Computer folgendermaßen:

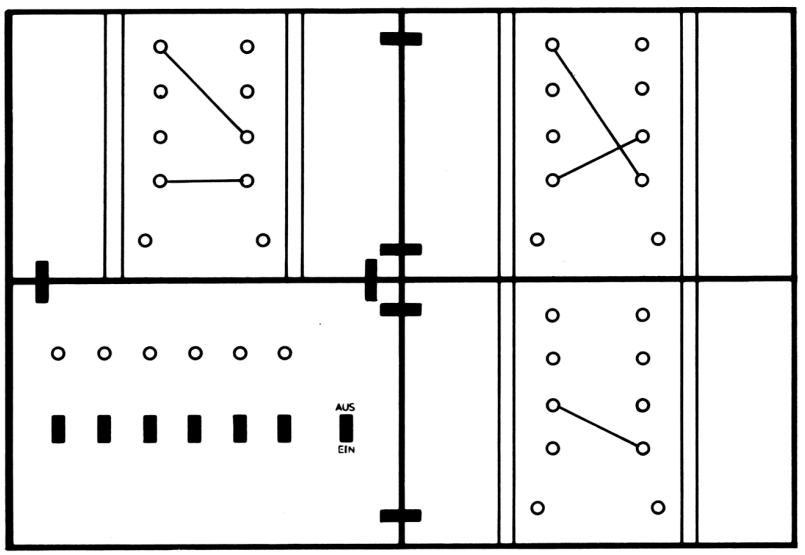


Abb.85

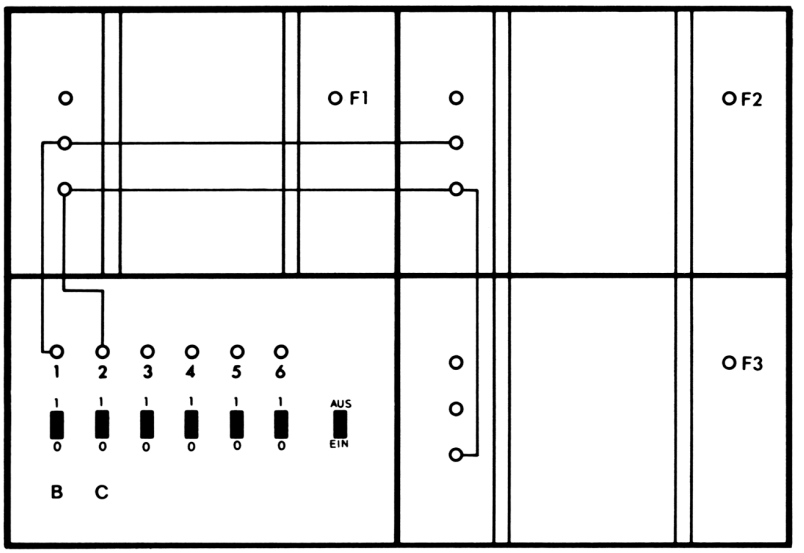


Abb.85

Und nun schalten Sie alle Möglichkeiten durch und ermitteln Sie, in welchen Fällen Sie Ihre Party im Garten, in der Veranda oder im Haus feiern werden.

Wetter B	Temperatur C	Garten F1	Veranda F2	Haus F3
0	0			
0	1			
1	0			
1	1			

Das ist also das Ergebnis:

B	C	F1	F2	F3
0	0	0	0	1
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

Wir können diese Funktionstabelle auch als Entscheidungstabelle bezeichnen, denn sie gibt uns an, wie wir uns je nach den Bedingungen bei unserem Problem entscheiden müssen. Bei dieser Gelegenheit sei auch noch ein weiterer Begriff für Funktionstabelle erwähnt: Wahrheitstabelle.

13.2. Wettervorhersage

Das Wetter läßt sich mit einiger Wahrscheinlichkeit richtig vorhersagen, wenn mehrere Bedingungen bekannt sind. Wir wissen:
Im Sommer wird bei steigendem Luftdruck und bei steigender Temperatur das Wetter schön; bei fallendem Luftdruck und fallenden Temperaturen wird das Wetter schlecht. Im Winter dagegen wird bei steigendem Luftdruck und fallenden Temperaturen das Wetter schön; bei fallendem Luftdruck und steigenden Temperaturen wird das Wetter schlecht.
Nun werden Sie mit Ihrem Computer mit Sicherheit nicht die Wettervorhersage der professionellen "Wetterpropheten" in den Schatten stellen können. Zudem wollen wir der Einfachheit halber vereinbaren, daß eine Wettervorhersage nicht möglich ist, wenn
im Sommer der Luftdruck steigt und die Temperatur fällt - oder umgekehrt - und wenn im Winter der Luftdruck steigt und die Temperatur steigt - oder umgekehrt.

Unsere Vorhersagetabelle sieht demnach so aus:

Jahreszeit A	Luftdruck B		Temperatur C		Wetter wird	
Sommer 1	steigt	1	steigt	1	schön	F1 = 1
	fällt	0	fällt	0	schlecht	F2 = 1
Winter 0	steigt	1	fällt	0	schön	F1 = 1
	fällt	0	steigt	1	schlecht	F2 = 1

Programmieren Sie Ihren Computer wie in der Abbildung dargestellt.

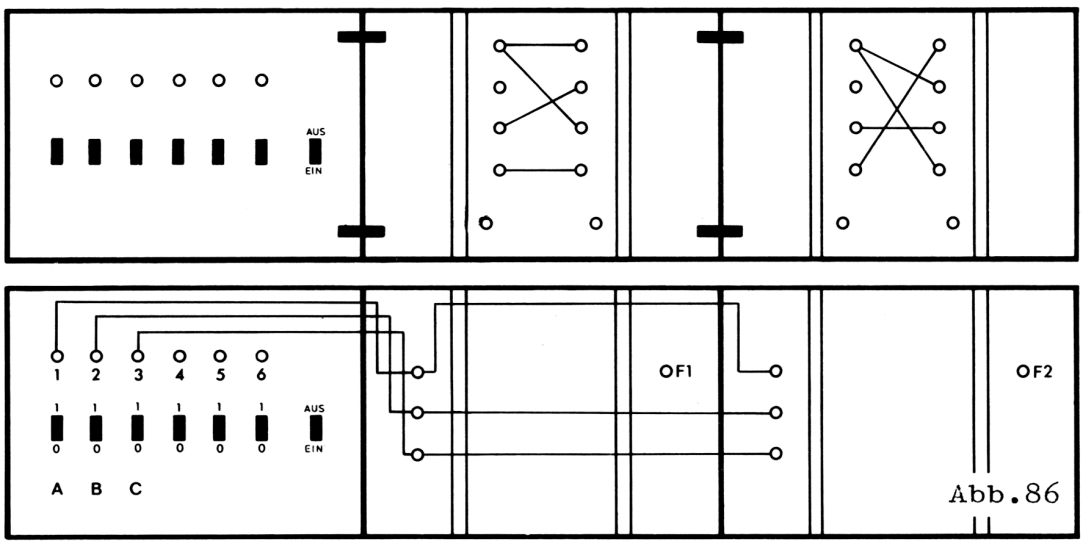


Abb.86

Und jetzt schalten Sie nach folgender Vereinbarung die verschiedenen Bedingungen durch:

A = 0 : Winter
A = 1 : Sommer
B = 0 : Luftdruck fällt
B = 1 : Luftdruck steigt
C = 0 : Temperatur fällt
C = 1 : Temperatur steigt

Tragen Sie die verschiedenen Wettervorhersagen in die Tabelle ein:

Jahreszeit A	Luftdruck B	Temperatur C	Wetter schön F1	Wetter schlecht F2
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Die Lösungen:

	Jahreszeit A	Luftdruck B	Temperatur C	Wetter schön F1	Wetter schlecht F2
+	0	0	0	0	0
	0	0	1	0	1
	0	1	0	1	0
+	0	1	1	0	0
	1	0	0	0	1
+	1	0	1	0	0
+	1	1	0	0	0
	1	1	1	1	0

Anmerkung: + keine Vorhersage möglich

13.3. Kreditprüfung

Ein Problem, das in allen Banken und Sparkassen täglich auftritt und eine Entscheidung verlangt:
Ein Kunde möchte von seinem Konto Geld abheben, auf dem Konto aber befindet sich kein Guthaben mehr. Wegen seiner Häufigkeit wird dieses Problem gern als Musterbeispiel zur Erläuterung von Entscheidungstabellen (entspricht unseren Funktionstabellen) herangezogen. Die verschiedenen Fragestellungen sind diese:

1. Ist das Kreditlimit noch nicht überschritten, d.h. ist ein Kredit ohne weiteres möglich (Kredit möglich)?

2. Wenn nein: Ist die Zahlungsweise des Kunden aus Erfahrung gut, so daß auch bei Überschreiten der Kreditgrenze kein Risiko besteht (Zahlungsweise gut)?

3. Wenn nein: Erteilt der Leiter der Kreditabteilung in diesem außergewöhnlichen Fall eine Sondergenehmigung (Sondergenehmigung)?

Müssen alle drei Fragen mit NEIN beantwortet werden, ist der Kreditantrag abzulehnen. Wenn jedoch bereits eine Frage mit JA beantwortet wird, so kann der Kredit überwiesen bzw. ausgezahlt werden.

Programmieren Sie nun Ihren Computer.

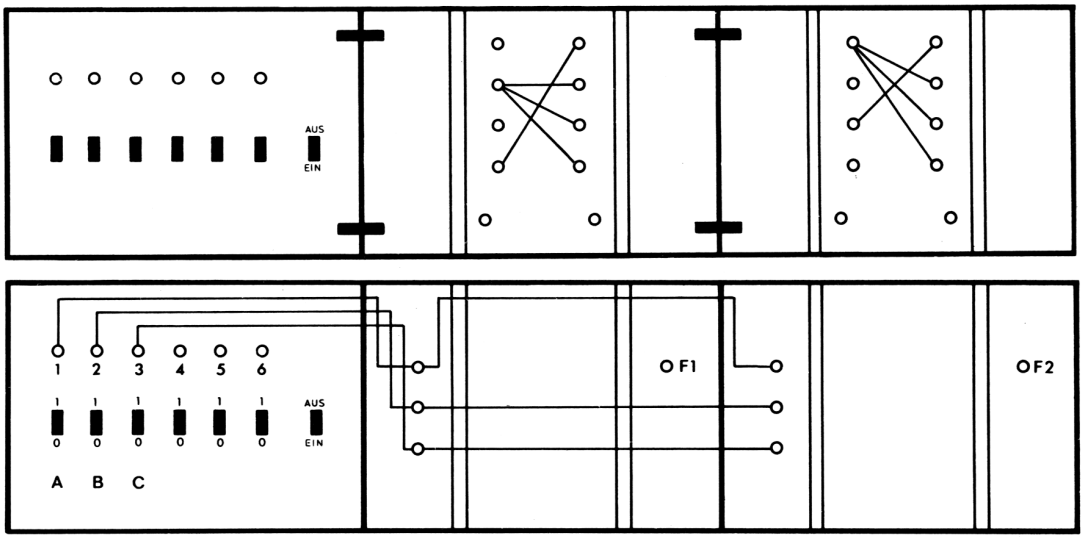


Abb.87

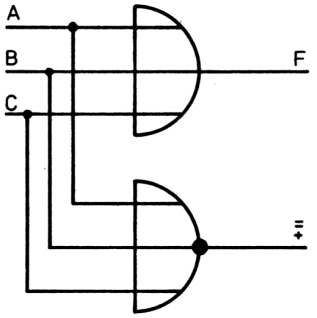
Wenn Sie nun die verschiedenen Möglichkeiten "durchschalten", erkennen Sie leicht, daß das Kreditproblem folgender Entscheidungstabelle entspricht:

A	Kredit möglich?	J	N	N	N
B	Zahlungsweise gut?		J	N	N
C	Sondergenehmigung erteilt?			J	N
F1	Kredit freigegeben	X	X	X	
F2	Kredit abgelehnt				X

JA (J) = 1
NEIN (N) = 0
Entscheidung (X) = Lampe brennt (F = 1)

Sie erkennen aus der Tabelle, daß die Kreditgewährung eine ODER-Bedingung darstellt, denn nur eine der drei Bedingungen A, B und C muß erfüllt sein, damit der Kredit gewährt wird. Da bereits eine Bedingung ausreicht, um einen Kredit zu bekommen, haben wir die Fälle, bei denen mehrere Bedingungen erfüllt sind (einschließendes ODER), der Einfachheit halber hier weggelassen. Die drei Bedingungen zusammen mit dem Ereignis "Auftrag abgelehnt" stellen eine NOR-Funktion dar: Wenn ein Kredit nicht möglich ist und wenn die Zahlungsweise nicht gut ist und wenn die Sondergenehmigung nicht erteilt wird, dann wird der Auftrag abgelehnt.

Den Zusammenhang können wir durch folgendes Schaltbild darstellen:

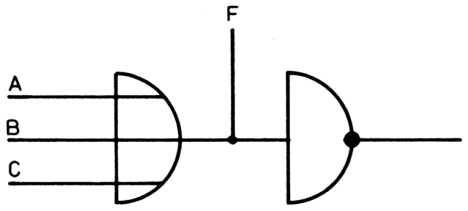


Versuchen Sie nun bitte, die beiden Funktionstabellen (ODER, NOR) gemäß der Entscheidungstabelle aufzustellen.

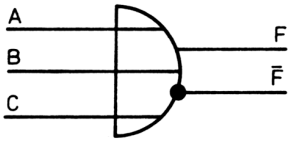
A	B	C	F1

A	B	C	F2

Wenn Sie sich jetzt die Ergebnisse noch einmal ansehen und feststellen, daß die beiden Rubriken "Auftrag angenommen" bzw. "Auftrag abgelehnt" immer einander entgegengesetzte Werte haben, so läßt sich leicht einsehen, daß auch folgende Darstellungsweise für die Schaltsymbolik möglich ist.



oder vereinfacht



Wir haben es hier mit einer logischen Funktion mit drei Eingängen und zwei Ausgängen zu tun. Die Ausgänge sind einander entgegengesetzt, man spricht deshalb auch von komplementären Ausgängen.
Demnach können wir auch bei unserem Computer den zweiten Ausgang einfach als komplementären Ausgang programmieren. Wir verdrahten ihn als Inverter:

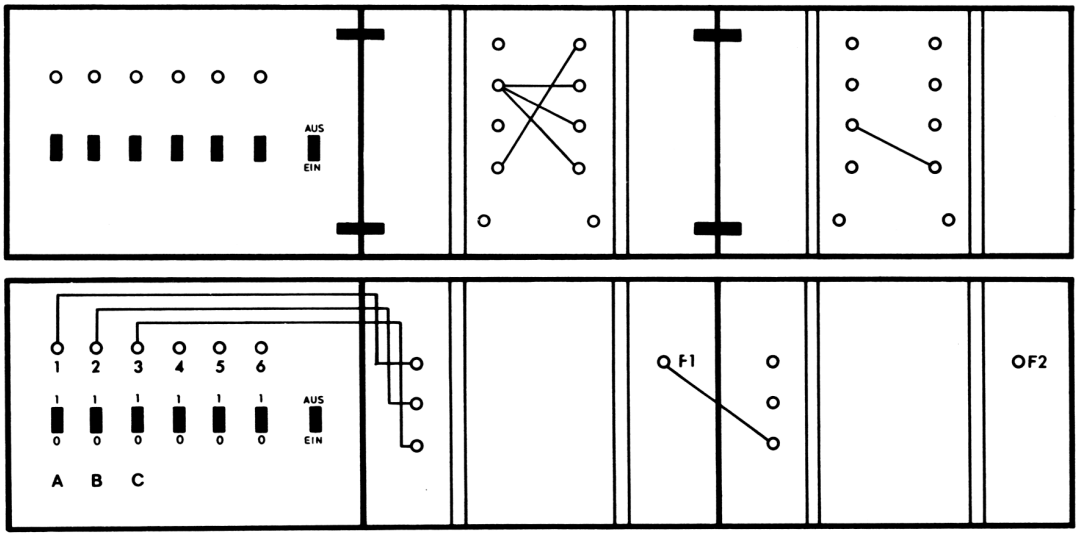


Abb.88

13.4. Platzreservierung

Computer werden heutzutage auch bei der Platzreservierung im Eisenbahnverkehr eingesetzt. Das wollen wir auch tun, allerdings verständlicherweise in etwas vereinfachter Form.
Nehmen wir an, Sie wünschen eine Platzreservierung für eine bestimmte Fahrt in der 2. Klasse. Das Personal, das für die Platzreservierung zuständig ist, hat folgende Anweisungen:

> Wenn in der 2. Klasse noch ein Platz frei ist, dann für den Reisenden reservieren.
>
> Ist in der 2. Klasse kein Platz frei, dann fragen, ob eine Reservierung in der 1. Klasse akzeptiert wird.
> Wenn JA: Ist dort noch ein Platz frei?
> Wenn JA: In der 1. Klasse reservieren.
>
> Wenn in der 1. Klasse kein Platz mehr frei ist, fragen, ob für einen anderen Zug gebucht werden soll.
> Wenn JA, wird eine Reservierung für diesen Ausweichzug vorgenommen.
>
> Wird der Ersatzzug nicht gewünscht, so ist dem Reisenden zu empfehlen, ohne Platzkarte rechtzeitig beim Eintreffen des Zuges auf dem Bahnsteig zu sein.

Stellen wir die Zusammenhänge in einer Entscheidungstabelle dar, ergeben sich folgende Verhältnisse:

A	2. Klasse verfügbar?	J	N	N	N
B	1. Klasse akzeptiert?		J	N	N
C	Ersatzzug akzeptiert?			J	N
F1	Buchen 2. Klasse	X			
F2	Buchen 1. Klasse		X		
F3	Buchen Ersatzzug			X	
F4	Reisen ohne Platzkarte				X

JA (J) = 1
NEIN (N) = 0
Entscheidung (X) = Lampe brennt (F = 1)

Zu den gleichen Ergebnissen kommen Sie, wenn Sie 4 Logik-Bausteine wie folgt programmieren.

A	B	C	F1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = 0
→ f2 = 0
→ f3 = $\bar{C}$
→ f4 = 0

A	B	C	F2
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	0
1	1	1	0

→ f1 = 0
→ f2 = $\bar{C}$
→ f3 = 0
→ f4 = 0

A	B	C	F3
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = C
→ f2 = 0
→ f3 = 0
→ f4 = 0

A	B	C	F4
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = $\bar{C}$
→ f2 = 0
→ f3 = 0
→ f4 = 0

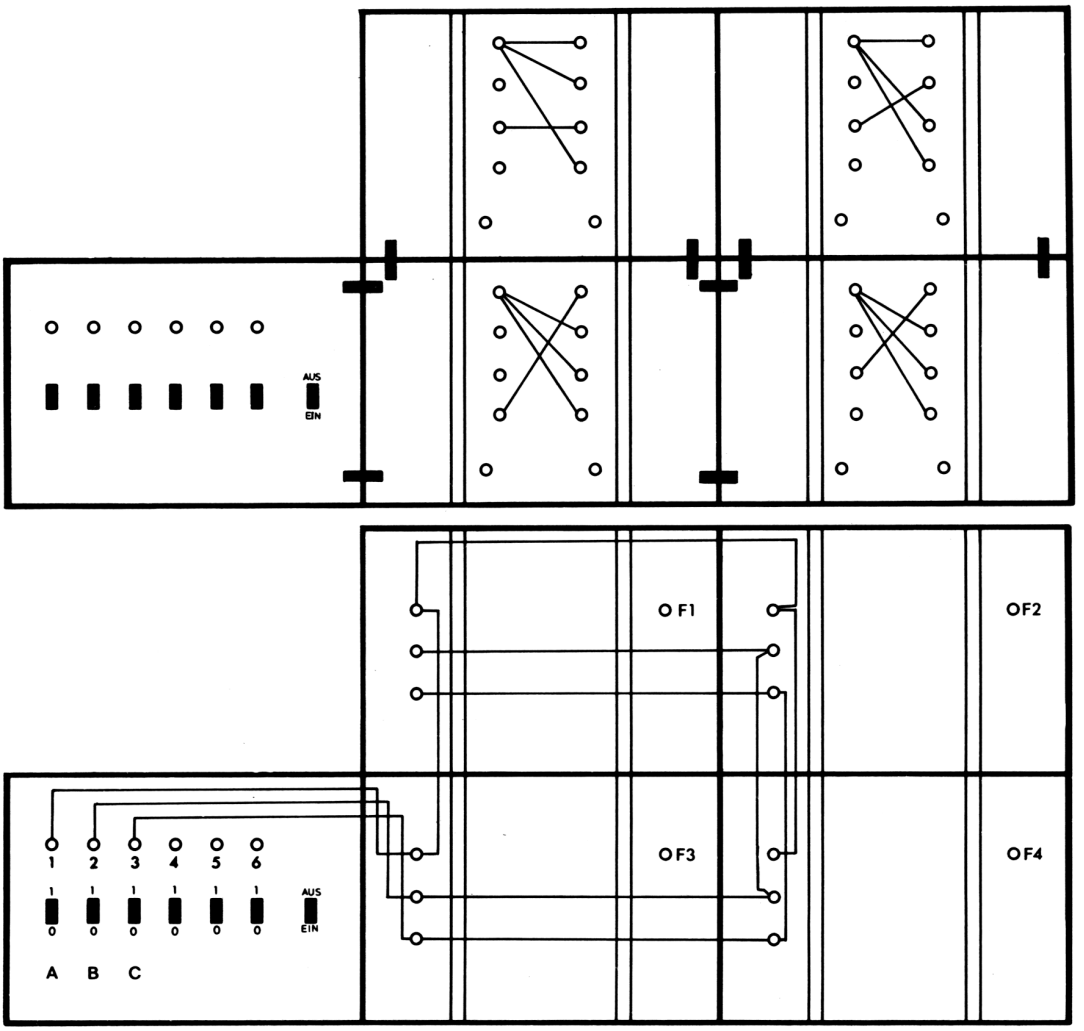


Abb.89

13.5. Ein Tresor wird geknackt

Ein sequentielles Kombinationsschloß aus Flipflops.

Programmieren Sie die Computer-Bausteine wie auf der Abbildung dargestellt als sogenannte D-Auffang-Flipflops 1. Setzen Sie nun alle Schalter auf 0 und bringen Sie die Schalter 1 - 5 der Eingabeeinheit einmal auf 1 und dann wieder auf 0. Damit sind alle Ausgänge der Versuchsanordnung auf 0 gesetzt. Wenn Sie jetzt den Schalter 6 auf 1 setzen und nacheinander die Schalter 1 - 5 kurzzeitig auf 1 und anschließend wieder auf 0 bringen, können Sie beobachten, daß die eingegebene Information - das Signal 1 also - von Logik-Baustein zu Logik-Baustein bzw. von Flipflop zu Flipflop weitergegeben wird, bis es beim 5. Flipflop angelangt ist und alle Lampen leuchten.

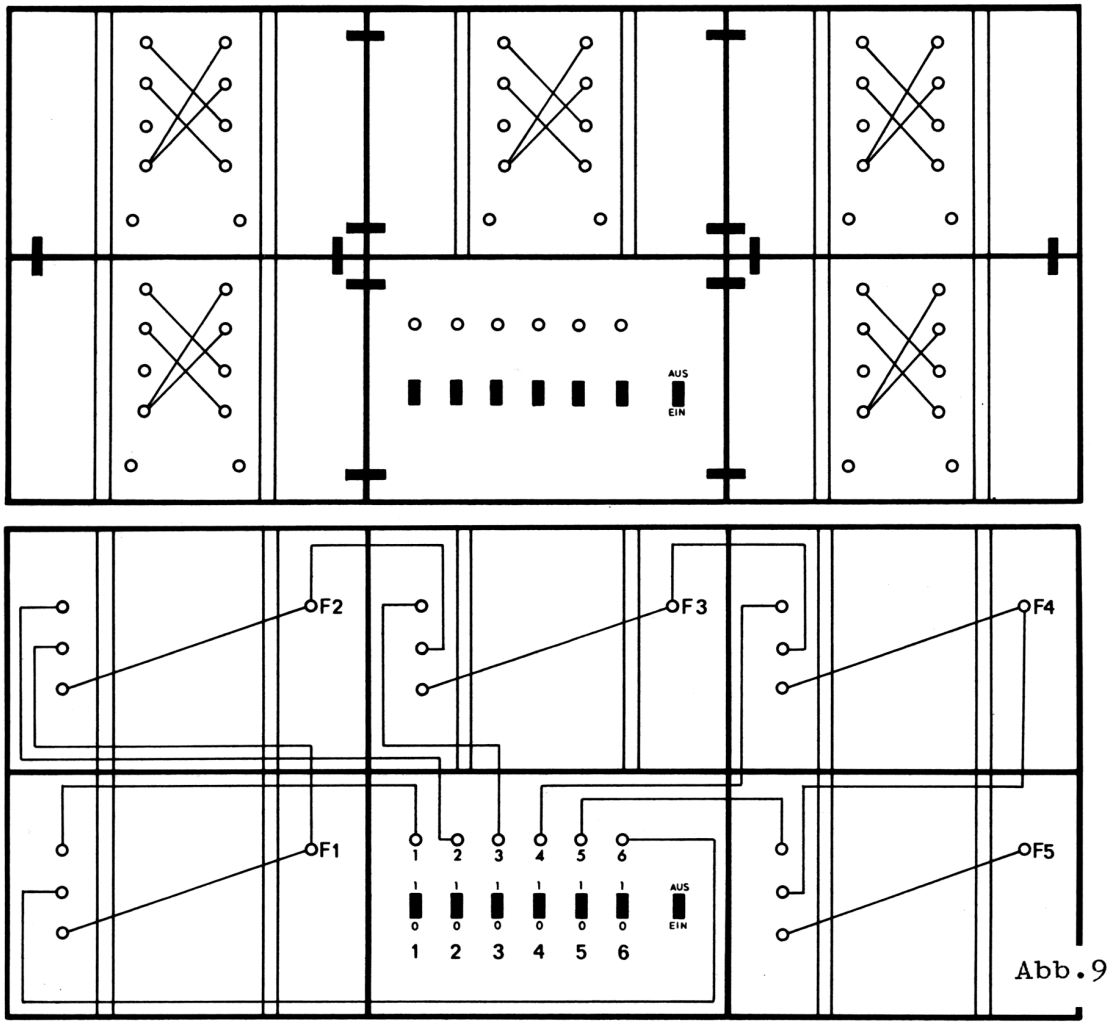


Abb.9o

Betätigen Sie die Schalter in anderer Reihenfolge, so wird das Signal nicht weitergegeben. Wir bauen unsere Versuchsanordnung jetzt zu einem Kombinationsschloß um, das nur derjenige betätigen kann, der die Kombination, d.h. die Reihenfolge, in der die Schalter zu bedienen sind, kennt.

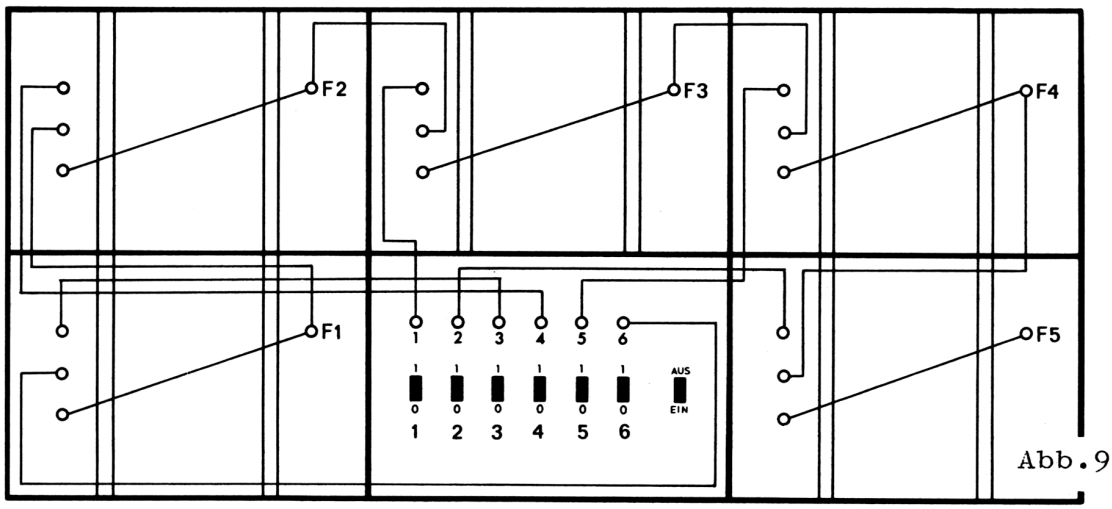


Abb.91

Die Schaltungsanordnung soll das Schloß zu einem Tresor sein. Lassen Sie einen Freund versuchen, ob er diesen Tresor knacken kann. Das Kombinationsschloß gibt den Zugang zum Tresor frei, wenn alle Lampen der Logik-Bausteine brennen. Solange Ihr Freund die Verdrahtung nicht durchschaut, ist es reine Glückssache, zum Ergebnis zu gelangen, denn es gibt insgesamt 12o Möglichkeiten für die Reihenfolge, in der die Schalter durchgeschaltet werden können. Sie selbst kennen natürlich die richtige Kombination - im abgebildeten Beispiel 3 4 1 5 2 - und können somit den Tresor jederzeit öffnen.
Diese Kombination können Sie beliebig ändern, indem Sie die Eingänge A der Logik-Bausteine 1 - 5 immer anderen Schaltern zuordnen. Der Schalter, der mit Logik-Baustein 1 verbunden ist, muß stets zuerst betätigt werden, dann der mit Logik-Baustein 2 usw.

13.6. Knobeln

Haben Sie schon einmal geknobelt? Sicher kennen Sie dieses Spiel und wissen, daß man sich dabei verschiedener Hilfsmittel wie Streichhölzer, Würfel usw. bedienen kann. Oder Sie nehmen eine Münze. Ihr Mitspieler setzt auf "Adler" oder "Zahl", und Sie werfen das Geldstück in die Luft. Ist beispielsweise die Zahl sichtbar und Ihr Partner hat auch auf "Zahl" gesetzt, so ist er der Gewinner des Spiels.
Diese Art des Knobelns läßt sich auch elektronisch verwirklichen. Betrachten Sie dazu den nachfolgenden Verdrahtungsplan.

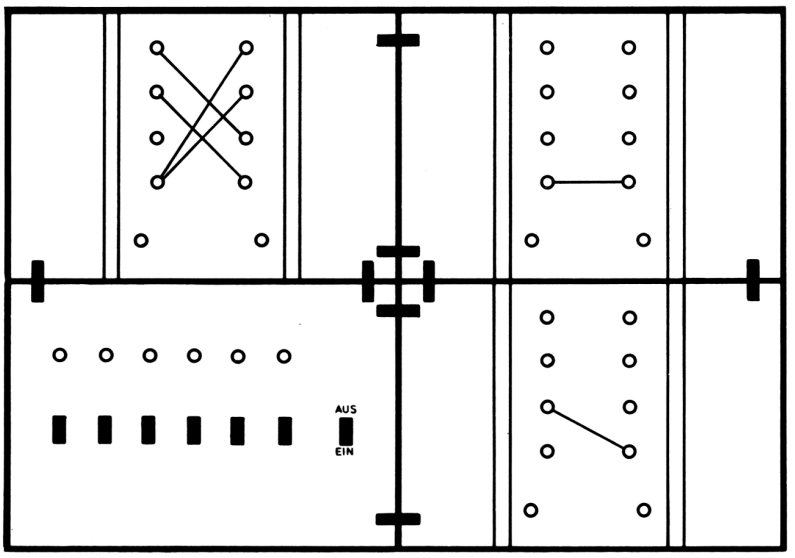


Abb.92

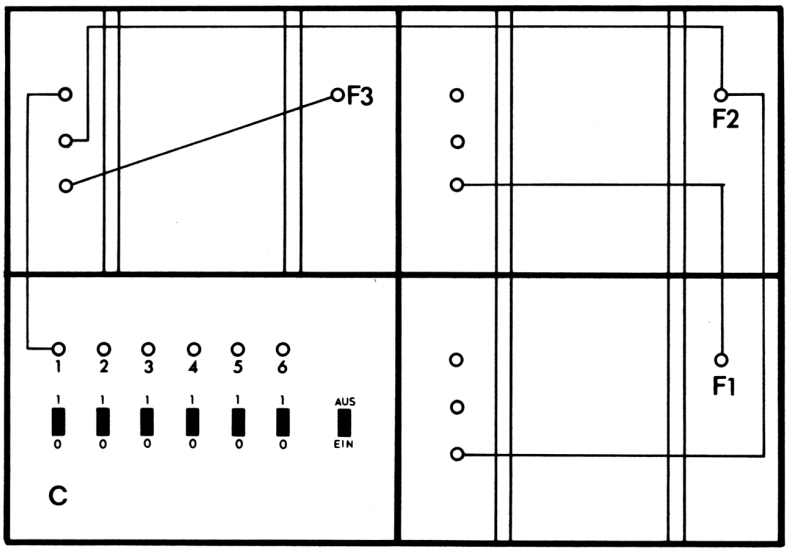


Abb.92

Die Logik-Bausteine 1 und 2 sind hier als Generator geschaltet, wobei der Ausgang F2 mit dem Eingang D eines D-Auffang-Flipflops 1 verbunden ist. Wird der Schalter C auf 1 gestellt, beeinflußt die Ausgangswechselspannung des Generators das Verhalten des DFF1. Wenn Sie jetzt den Schalter C wieder in die 0-Stellung bringen, speichert das Flipflop den Wert, der zuletzt am D-Eingang vorhanden war. Da die Generatorfrequenz sehr hoch ist und Sie den Schaltzustand nicht mit dem Auge verfolgen können, ist es ganz dem Zufall überlassen, ob die Lampe (F3) brennt oder ob sie dunkel bleibt.

13.7. Würfeln

Würfelspiele gibt es in zahlreichen Variationen. Doch sicherlich haben Sie noch nie elektronisch gewürfelt, wie Sie es in diesem Abschnitt tun werden.

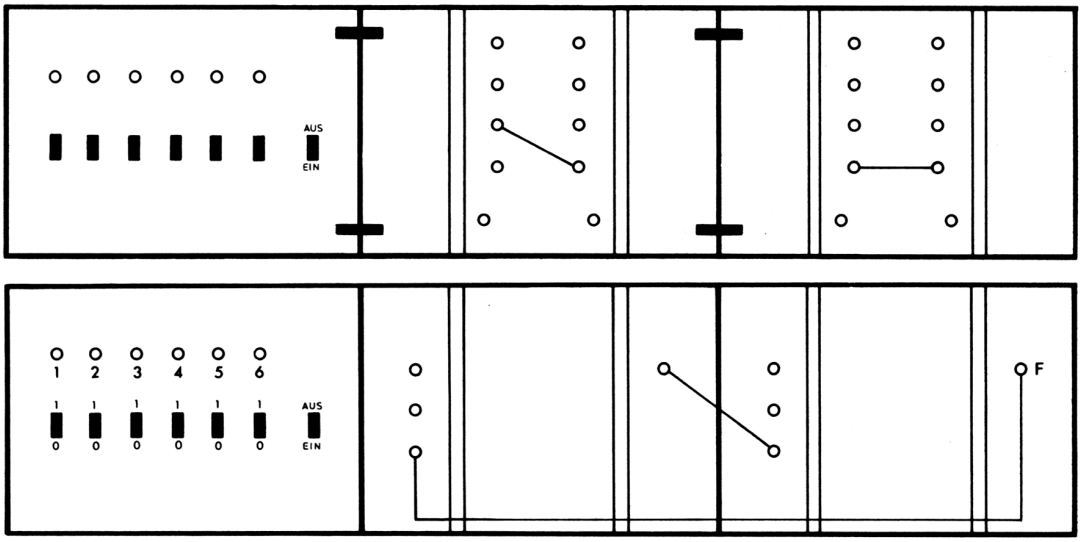


Abb.93

Grundlage dieses Würfelspiels ist ein Generator, der aus zwei Logik-Bausteinen (Inversions- und Identitätsschaltung) besteht.

Die Generatorfrequenz wird durch die Laufzeit der Logik-Bausteine bestimmt. Der Ausgang des Generators ist mit den Eingängen der drei Logik-Bausteine angeschlossen, die als D-Auffang-Flipflops geschaltet sind. Ein DFF1, das haben Sie inzwischen erfahren, speichert am Eingang B die Information, sobald der Eingang A den Wert 0 annimmt. Das Ergebnis des Würfelns hängt also ausschließlich davon ab, ob Sie bei der Betätigung der Würfelschalter den Wert 0 am Ausgang des Generators oder den Wert 1 treffen. Da das bei der Generatorfrequenz nicht mehr mit dem Auge erfaßt werden kann, ist es wie beim Spiel mit richtigen Würfeln immer ein Zufall, welche Zahl Sie treffen.
Doch programmieren Sie zunächst einmal die gesamte "Würfelmaschine", wie in der folgenden Abbildung dargestellt.

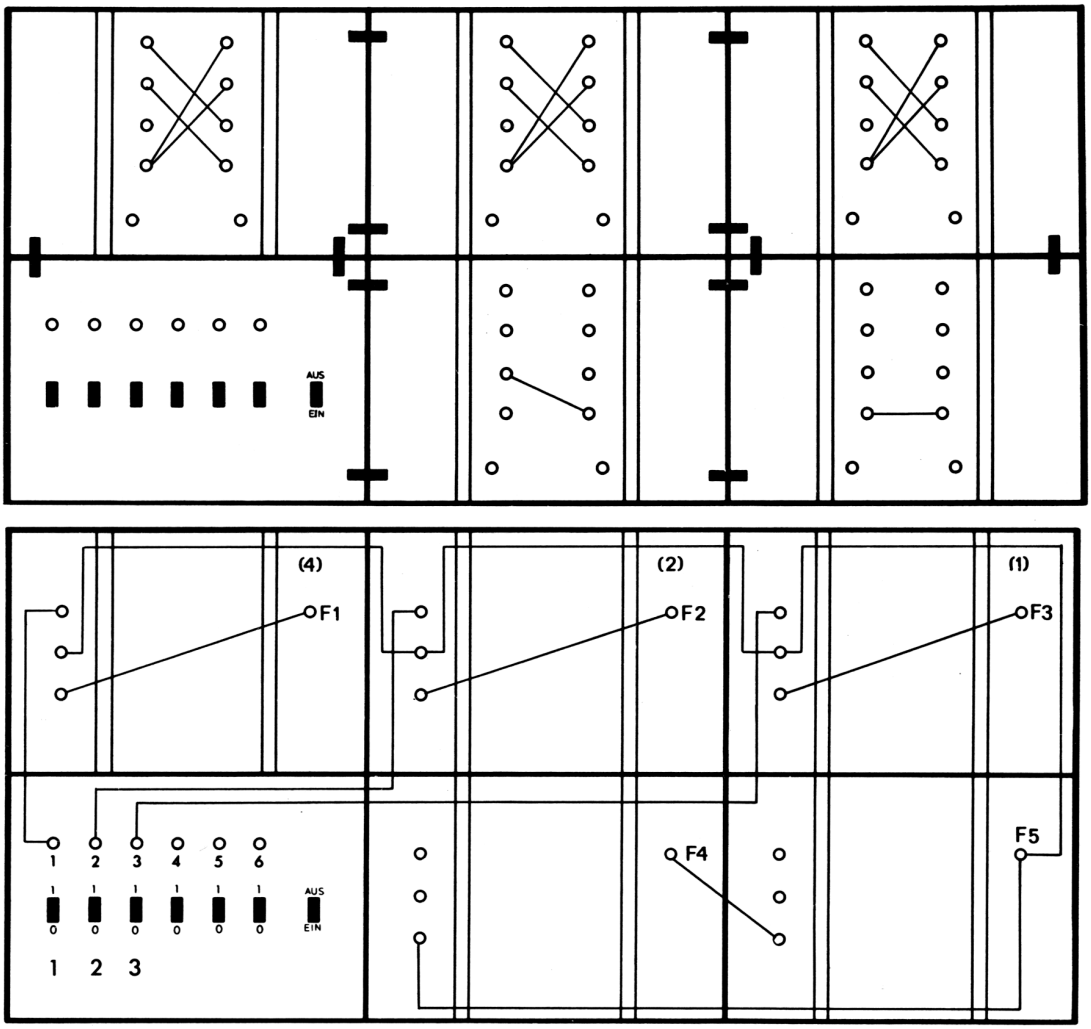


Abb.94

Nach dem Betätigen des Hauptschalters blinkt der Ausgang des Generators, und die Ausgänge der drei DFF1 leuchten. Durch die Schalter 1 bis 3 können Sie nun die "Becher schütteln", d.h. das Würfeln beginnt. Nacheinander oder gleichzeitig müssen dann die Schalter auf STOP (0) gestellt werden, und das Ergebnis kann an den Ausgängen der D-Auffang-Flipflops abgelesen werden. F1 hat den Wert 4, F2 besitzt den Wert 2 und F3 hat den Wert 1. Wenn Sie die Werte der Ausgänge F1 bis F3 addieren, erhalten Sie das Ergebnis. Dabei stellt sich heraus, daß Ihr "Würfel" nicht, wie sonst üblich, von 1 bis 6 zählt, sondern von 0 bis 7. Wenn Sie sich schwer daran gewöhnen können, sollten Sie festlegen, daß z.B. die 0 einem ungültigen Wurf entspricht und bei 7 noch einmal wiederholt werden darf. Sie können natürlich auch bei 0 wiederholen lassen.

Ein Spielvorschlag für mehrere Personen: Jeder Teilnehmer erhält die gleiche Zahl von Zündhölzern und wer die höchste Zahl in einer Runde würfelt, darf ein Streichholz fortlegen. Wer zuerst alle Hölzer ablegen kann, hat gewonnen.

13.8. Bauer, Ziege, Wolf und Kohlkopf

In einer schon fast klassischen Denksportaufgabe will ein Bauer eine Ziege, einen Wolf und einen Kohlkopf über einen Fluß transportieren. Es steht ihm jedoch nur ein kleines Boot zur Verfügung, in dem er jeweils nur ein Objekt mitnehmen kann. Es ergeben sich daraus Probleme. Denn wenn der Bauer z.B. den Wolf hinüberrudert, frißt die Ziege inzwischen den Kohlkopf. Wenn der Wolf und die Ziege zurückgelassen werden, frißt sie der Wolf auf. Der Bauer muß also so geschickt das Übersetzen ausführen, daß keine Gefahr auftreten kann. Eine Gefahr soll auch immer dann ausgeschlossen werden, wenn der Bauer zugegen ist.

Die Stellung 0 der Schalter 1 bis 4 (für Bauer, Ziege, Wolf und Kohlkopf) bedeutet, daß sich das Objekt am linken Ufer befindet; 1 heißt, daß es sich am rechten Ufer aufhält. Droht am linken Ufer oder am rechten Gefahr, so leuchtet dafür die Lampe des entsprechenden Logik-Bausteins auf. Während der Überfahrt besteht keine Gefahr, da der Bauer immer anwesend ist. Für die Anwesenheit des Bauern am linken Ufer sieht die Funktionstabelle folgendermaßen aus:

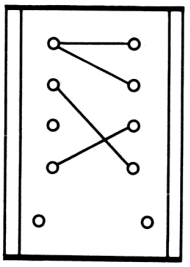
Vorbedingung: Bauer links (0) Teilbedingungen

Ziege A	Wolf B	Kohl C	Gefahr links F2	Gefahr rechts F1
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1	0 +)	0
1	1	1		0
1	0	1		1
1	1	0		1
1	1	1		1

+) Es droht keine Gefahr, solange der Bauer dort anwesend ist.

Aus dieser Funktionstabelle kann die Programmierung für den Logik-Baustein entnommen werden, der auf die Gefahr am rechten Ufer hinweist, wenn sich der Bauer am linken befindet.

Gefahr rechts F1	
0 0	f1 = 0
0 0	f2 = 0
0 1	f3 = C
1 1	f4 = 1



Für die Anwesenheit des Bauern am rechten Ufer und die dabei entstehenden Gefahren gilt eine entsprechende Funktionstabelle.

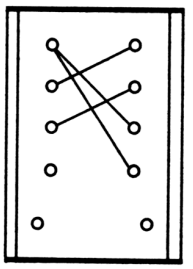
Vorbedingung: Bauer rechts (1) Teilbedingungen

Ziege A	Wolf B	Kohl C	Gefahr links F2	Gefahr rechts F1
0	0	0	1	
0	0	1	1	
0	1	0	1	
0	1	1	0	0 +)
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	0	

+) Wie auf dem linken Ufer droht auch hier keine Gefahr, wenn der Bauer anwesend ist.

Dieser Logik-Baustein wird nach den Ergebnissen der Funktionstabelle programmiert und enthält somit die Teilbedingungen für die Gefahr am linken Ufer, wenn der Bauer rechts ist.

Gefahr links F2	
1 1	f1 = 1
1 0	f2 = $\bar{C}$
0 0	f3 = 0
0 0	f4 = 0



Die beiden Logik-Bausteine für die Teilbedingung "Gefahr am linken Ufer" und "Gefahr am rechten Ufer" müssen nun noch mit zwei weiteren verbunden werden, die die Verknüpfung der Gefahren an beiden Ufern mit dem Aufenthalt des Bauern herstellen.

	Bauer A	Teilbedingung Gefahr rechts B	Gefahr rechts F3	
Bauer links	0	0	0	f1 = 0
	0	1	1	f2 = 1
Bauer rechts	1	0	0 +	f3 = 0
	1	1	0	f4 = 0

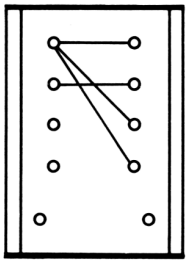
+) Keine Gefahr, da der Bauer ebenfalls rechts ist.

Funktionstabelle für den Aufenthalt des Bauern und die Teilbedingung für Gefahr links:

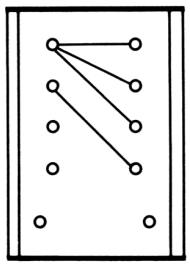
Bauer A	Teilbedingung Gefahr links B	Gefahr links F4	
0	0	0 +	f1 = 0
0	1	0	f2 = 0
1	0	0	f3 = 0
1	1	1	f4 = 1

+) Keine Gefahr, da der Bauer ebenfalls links ist.

Programm Logik-Baustein 3



Programm Logik-Baustein 4



Wenn Sie die vier Logik-Bausteine bereits programmiert haben, brauchen Sie diese nur noch nach der folgenden Abbildung mit der Eingabeeinheit zu verbinden.

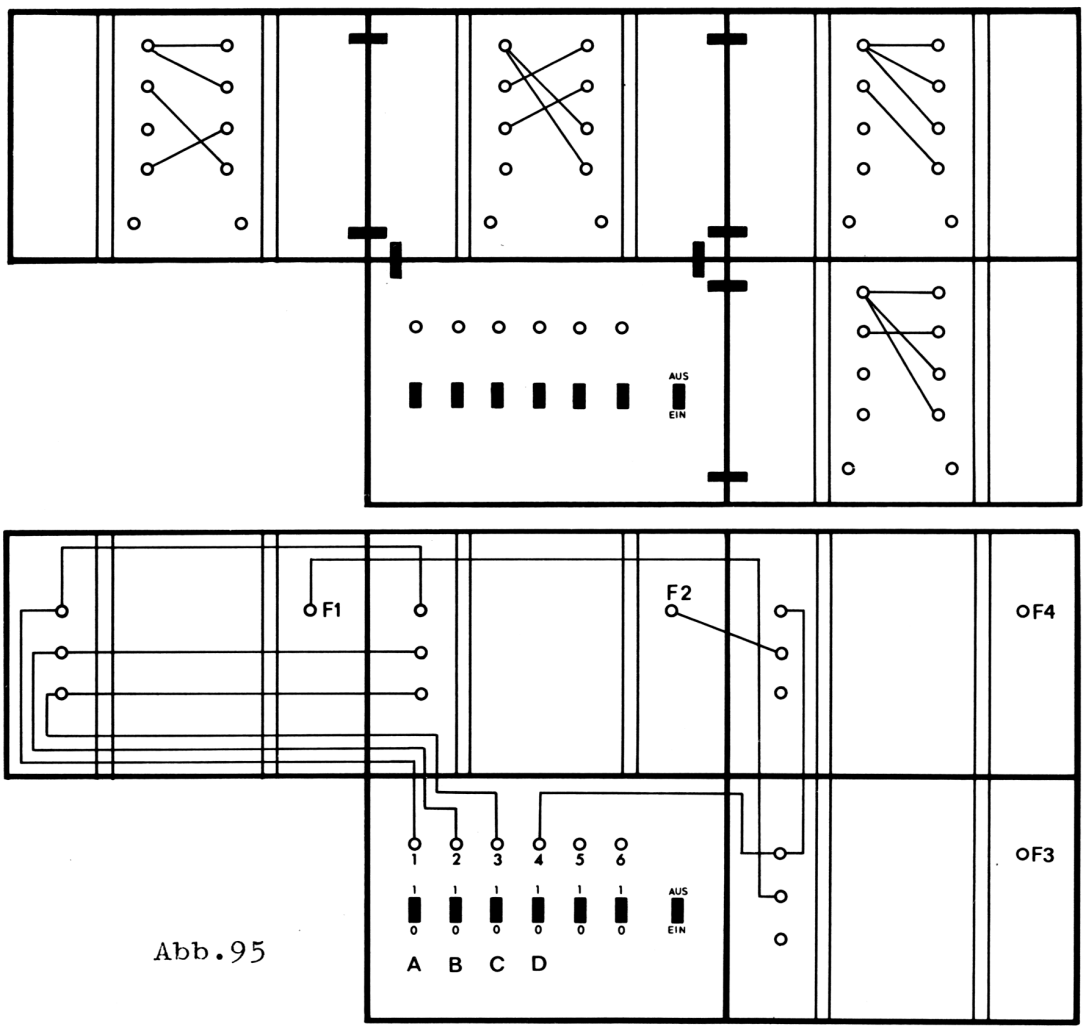


Abb.95

Mit dieser Schaltung können Sie nun versuchen, die Lösung zu finden, die es dem Bauern ermöglichte, seine Güter gefahrlos von einem Ufer zum anderen zu transportieren.

Gefahr droht immer, wenn die Lampen F3 oder F4 aufleuchten (F1 und F2 bleiben unberücksichtigt.)

Eine Lösung sieht so aus (Ausgang vom linken Ufer; A, B, C, D = 0):

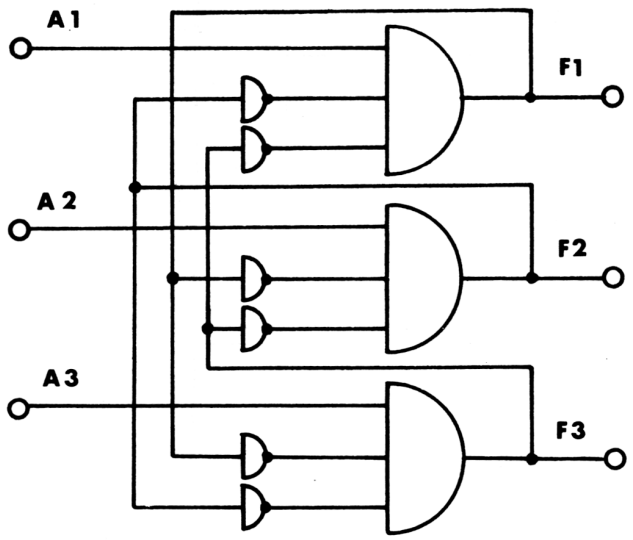
1.	Bauer fährt mit der Ziege nach rechts.	(D = 1, A = 1)
2.	Bauer kehrt leer zurück.	(D = 0)
3.	Bauer fährt mit dem Kohlkopf nach rechts.	(D = 1, C = 1)
4.	Bauer nimmt die Ziege nach links zurück.	(D = 0, A = 0)
5.	Bauer bringt Wolf nach rechts.	(D = 1, B = 1)
6.	Bauer kehrt leer zurück.	(D = 0)
7.	Bauer fährt die Ziege nach rechts.	(D = 1, A = 1)

Diese Lösung ist nicht die einzig mögliche. Sicherlich gelingt es Ihnen, die anderen mit Ihrem Computer zu ermitteln.

13.9. Reaktionstest mit einer Verriegelungsschaltung

Mit dieser Schaltung können Sie testen, ob Sie schneller sind als Ihre Freunde.

Dazu ist eine Schaltung notwendig, bei der mit drei Schaltern jeweils eine zugehörige Lampe eingeschaltet werden kann. Das allein reicht natürlich noch nicht, denn so würden schließlich alle drei Lampen leuchten. Sobald also die erste Lampe eingeschaltet wird, müssen die anderen gesperrt werden. Die Programmierung muß so gewählt werden, daß durch den Ausgang des einen die Eingänge der anderen Bausteine verriegelt werden. Das geschieht durch Inverter.



Sie erkennen aus dieser Schaltung, daß es sich jeweils um eine UND-Funktion mit 2 negierten Eingängen handelt. Für jede dieser Funktionen kann eine Tabelle aufgestellt werden.

A1	F2	F3	F1
A2	F1	F3	F2
A3	F1	F2	F3
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Sie erkennen aus dieser Tabelle, daß z.B. F1 nur 1 sein kann, wenn lediglich A1 = 1 ist. Entsprechendes gilt für F2 und F3. Daraus ergibt sich, daß die Programmierung aller drei Logik-Bausteine gleich ist. Durch die Verknüpfung der Ausgänge mit jeweils zwei Eingängen wird erreicht, daß z.B. F2 und F3 verriegelt werden, wenn A1 zuerst geschaltet wurde. Wenn allerdings zufällig zwei oder drei Schalter gleichzeitig gedrückt werden, leuchten die entsprechenden Ausgänge zusammen auf.
Hier die vollständige Schaltung.

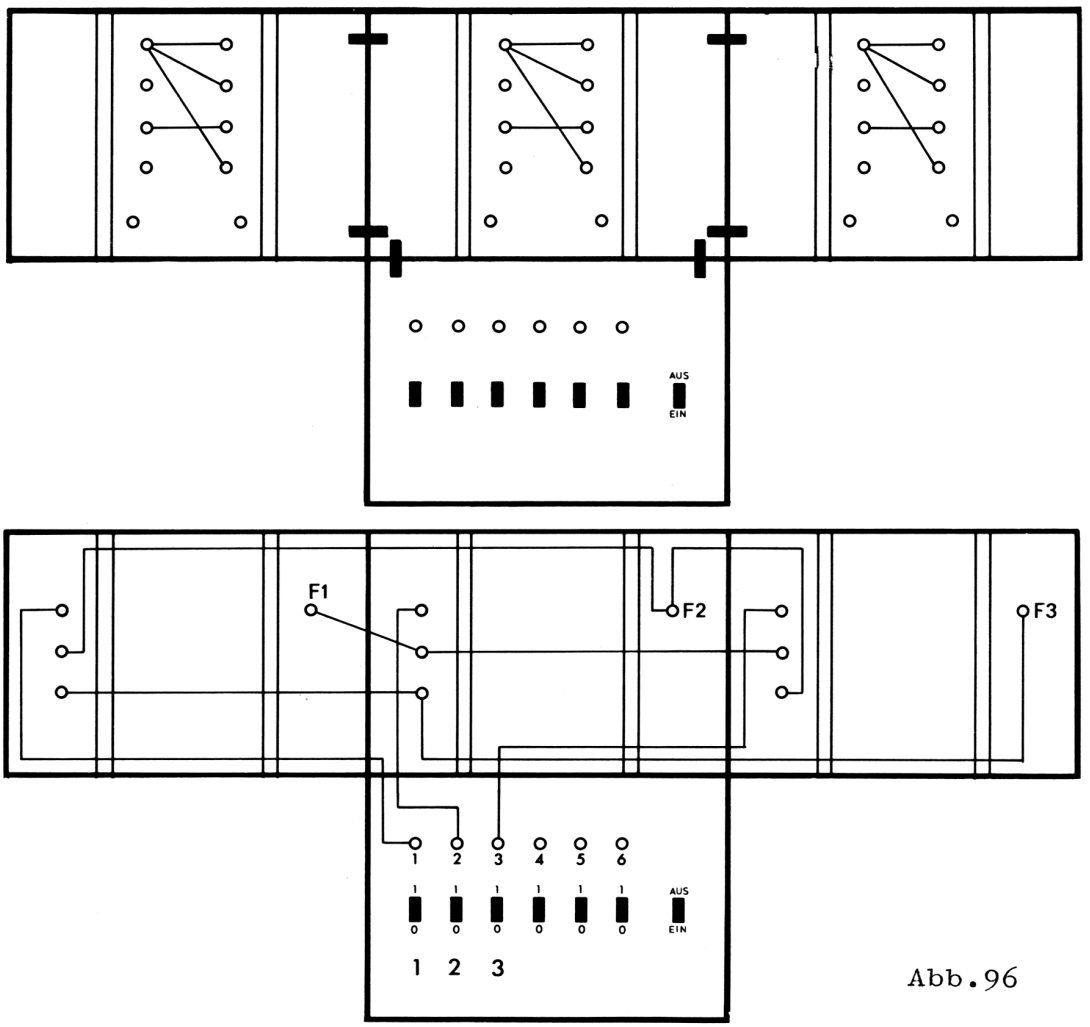


Abb.96

13.1o. Wer gewinnt gegen den Computer?

Ein Zahlenspiel

An diesem Spiel können zwei Spieler teilnehmen. Einer bedient sich dabei des Computers, und der andere tritt gegen diese "Mannschaft" an.

Die Spielregeln: Der erste Spieler nennt eine Zahl von 1 bis 3, und der nächste zählt eine Zahl, ebenfalls von 1 bis 3, hinzu. Abwechselnd addieren die Spieler eine dieser Zahlen, und wer schließlich 33 oder mehr erreicht, der hat verloren.
Zur Erläuterung hier ein Spiel ohne die Hilfe des Computers:

Spieler	A	fängt an mit		z.B.	1	
Spieler	B	addiert	2	=	3	
	A	+	3		6	
	B		3		9	
	A		1		1o	
	B		3		13	
	A		3		16	
	B		3		19	
	A		2		21	
	B		3		24	
	A		1		25	
	B		1		26	
	A		3		29	
	B		2		31	
	A		1		32	
	B		1		33	B hat verloren.

Wenn Sie dieses Spiel mit dem Computer spielen, sind Sie nicht zu schlagen, sofern Sie Ihren Gegenspieler beginnen lassen. Wenn Sie selbst anfangen, macht Sie der Computer auf Fehler Ihres Gegenspielers aufmerksam, und dann können Sie ebenfalls nicht mehr verlieren. Der Computer sagt Ihnen nämlich immer die richtige Zahl, die Sie addieren müssen.

Zu dieser Schaltung benötigen Sie vier Logik-Bausteine, die zu zwei Binärzählern programmiert werden. Baustein 1 und 2 sowie 3 und 4 stellen je einen Binärzahler dar. Das Ergebnis, das Sie jeweils addieren müssen, wird an den Ausgängen F2 und F4 - und zwar dual - angezeigt. Dabei ist der dezimale Wert für F2 -2- und für F1 -1-.
Sie müssen während des Spiels den Schalter S so oft von 0 nach 1 und zurück nach 0 schieben, wie es der von Ihnen oder Ihrem Mitspieler addierten Zahl entspricht (z.B. +3 = dreimal Schalter S betätigen).
Aus der nachfolgenden Tabelle können Sie entnehmen, welche Zahl Sie bei den Ausgangszuständen F2 und F4 addieren müssen.

F2	F4			
0	0	=	0	bedeutet, daß Sie eigentlich eine 0 hinzuzählen müßten. Da das nicht erlaubt ist, wählen Sie eine beliebige Zahl. In dieser Situation ist Ihr Gegner im Vorteil.
1	1	=	3	bedeutet, daß Sie eine 3 addieren müssen. Jetzt kann Sie der Gegner nicht mehr schlagen.
1	0	=	2	eine 2 muß addiert werden, und Sie sind nicht mehr zu schlagen.
0	1	=	1	Sie müssen eine 1 hinzuzählen, um zu gewinnen.

Programmieren Sie bitte zunächst Ihre Logik-Bausteine, bevor wir Ihnen ein Beispiel mit Computerunterstützung bringen.

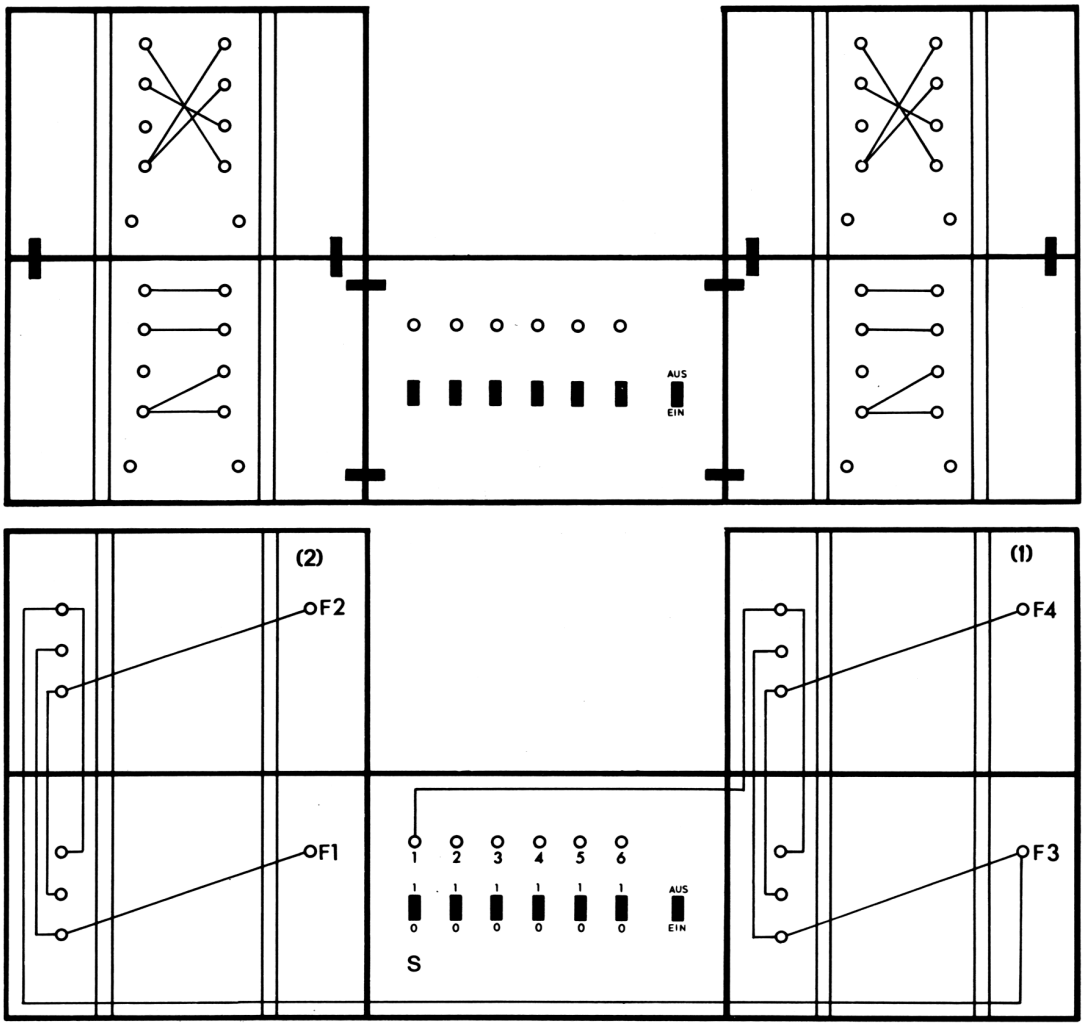


Abb.97

Und hier das Spiel: Zunächst müssen Sie die Flipflops mit dem Schalter S auf die Ausgangsposition 00 stellen. Alle Lampen sind in diesem Fall dunkel.

Der Spieler A spielt mit dem Computer, Spieler B allein. - A fängt an:

Spieler	wählt	Spieler A schaltet	Computer zeigt .. an	A muß .. sagen
A	2	2 x	1 0	-
B	+ 1 = 3	1 x	0 1	1
A	+ 1 = 4	1 x	0 0	-
B	+ 3 = 7	3 x	0 1	1
A	+ 1 = 8	1 x	0 0	-
B	+ 3 = 1o	2 x	1 0	2
A	+ 3 = 12	2 x	0 0	-
B	+ 1 = 13	1 x	1 1	3
A	+ 3 = 16	3 x	0 0	-
B	+ 3 = 19	3 x	0 1	1
A	+ 1 = 2o	1 x	0 0	-
B	+ 2 = 22	2 x	1 0	2
A	+ 2 = 24	2 x	0 0	-
B	+ 2 = 26	2 x	1 0	2
A	+ 2 = 28	2 x	0 0	-
B	+ 3 = 31	3 x	0 1	1
A	+ 1 = 32	1 x	0 0	

B muß an dieser Stelle 33 oder mehr sagen und hat deshalb verloren.

Das Prinzip dieses Spiels besteht darin, dem Gegenspieler eine Zahl weiterzugeben, die ein Vielfaches von 4 beträgt. Wenn Sie einmal eine solche Zahl erreicht haben, so müssen Sie nur noch die Zahl, die der Mitspieler nennt, auf 4 ergänzen, und damit erreichen Sie stets ein Vielfaches von 4. Der Computer zeigt diese Zahl an. Er zählt auch jeweils nur bis 4, gleichgültig, ob Sie bereits bei 8, 12, 16, 2o usw. sind.

Wenn Sie die Schaltung nach der folgenden Abbildung mit dem fünften Logik-Baustein versehen, kann die Anzeige erweitert und vereinfacht werden. Jetzt zählt jede Lampe als 1. Die Zahl der aufleuchtenden Lampen der Logik-Bausteine 2, 4 und 5 gibt die Zahl an, die Sie addieren müssen.

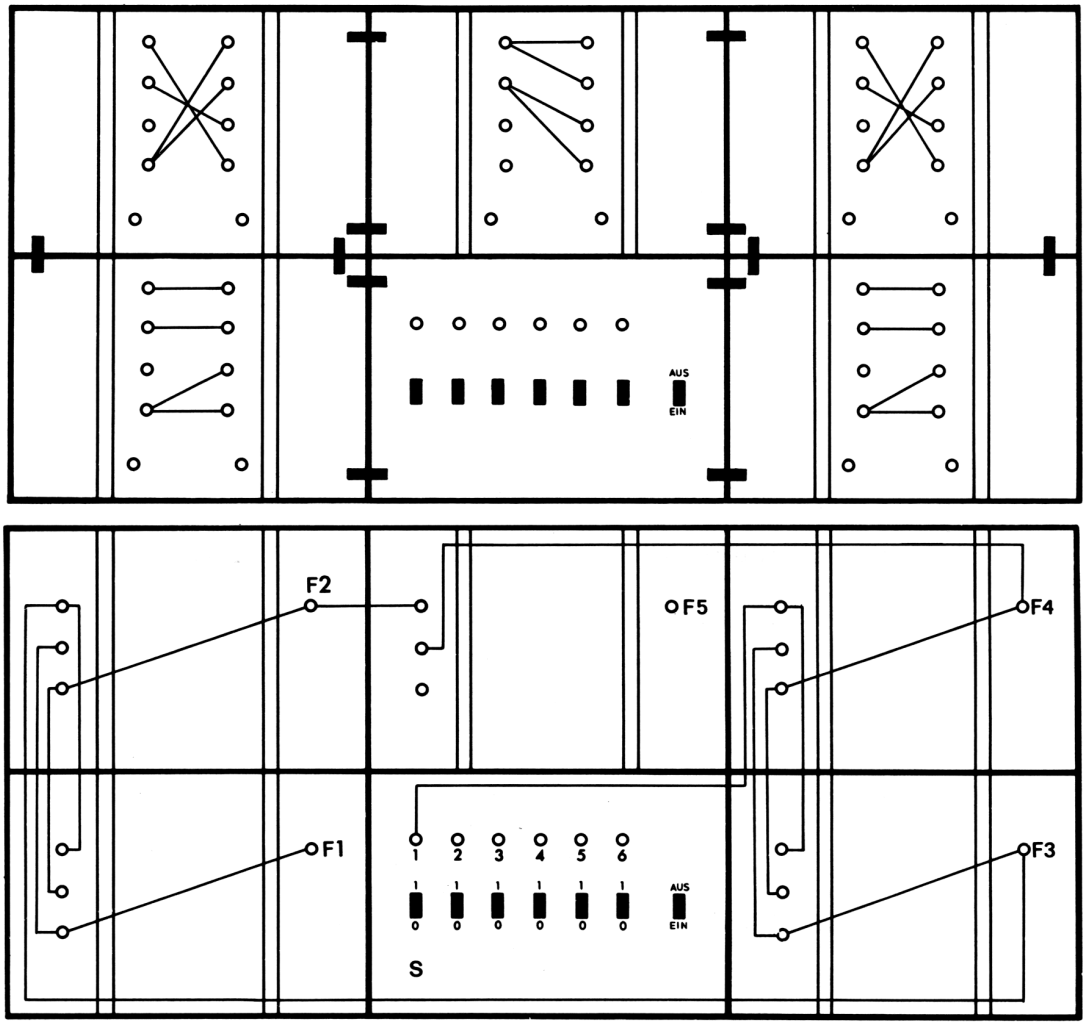


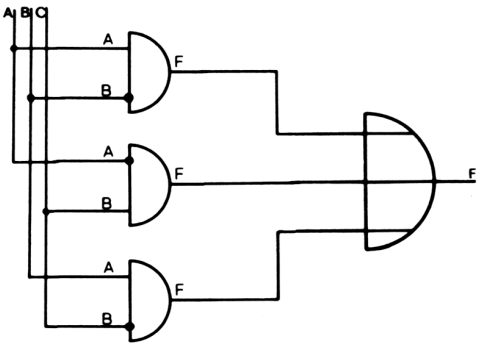
Abb.98

Der Computer-Lehrbaukasten CL 1601 ist der Grundkasten eines ausbaufähigen Computer-Lehrprogramms.

Wir werden uns erlauben, Ihnen Zusatzkästen und weitere Spiele sowie neue Schaltungen nach Erscheinen anzubieten.

13.11. Automatische Sicherung

In einem kleinen Handwerksbetrieb stehen drei Maschinen mit annähernd gleicher Leistung. Der Netzstrom ist so abgesichert, daß höchstens zwei der drei Aggregate gleichzeitig eingeschaltet werden können. Um zu vermeiden, daß beim Zuschalten der dritten Maschine die Hauptsicherung anspricht, wird eine automatische Sperre benötigt. Sie blockiert die Stromzufuhr zur dritten Maschine, wenn bereits zwei eingeschaltet sind. Aus der nachstehenden Schaltung können Sie erkennen, daß es sich um drei UND-Glieder mit jeweils einem negierten Eingang handelt, deren Ausgänge auf ein ODER geschaltet werden.



Die Funktionstabelle läßt erkennen, daß der Ausgang F immer dann 1 sein muß, wenn zwei Eingänge den Wert 1 führen. Dann wird die Verriegelung wirksam (F = 1).

A	B	C	F	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	
1	1	1	0	+)

+) Diese Möglichkeit darf nicht auftreten, da bereits nach zwei eingeschalteten Verbrauchern die Verriegelung einsetzt.

Die vollständige Schaltung und Programmierung entnehmen Sie der folgenden Abbildung.

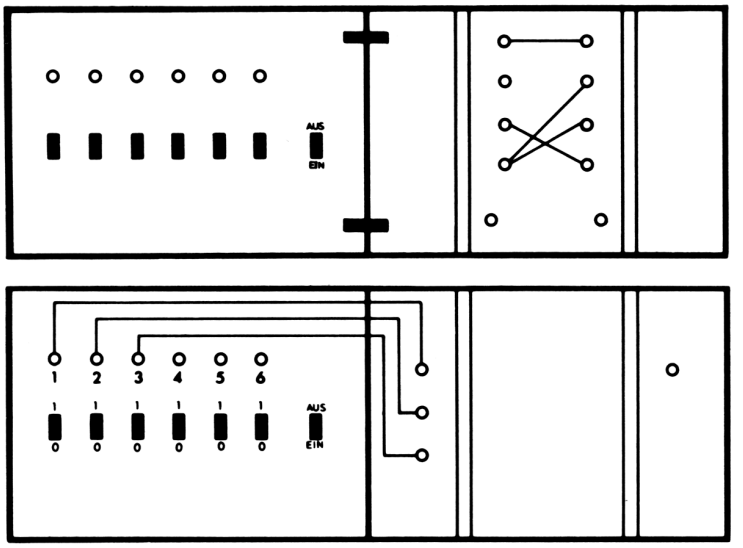


Abb. 99

Beim Ausprobieren der Schaltung wird Ihnen aufgefallen sein, daß die Sperre nicht wirkt (F = 0), wenn A, B und C den Wert 1 führen. Das bedeutet, daß die Sperre zwar wirkt, solange nur 2 Maschinen in Betrieb sind, sie aber aufgehoben wird, wenn der dritte Verbraucher trotzdem eingeschaltet wird. Deshalb muß die Funktionstabelle so abgeändert werden, daß auch für A, B und C = 1 F = 1 wird.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Wenn Sie die Programmierung nach dieser Funktionstabelle ausführen, erfüllt die automatische Sicherung die an sie gestellte Bedingung.

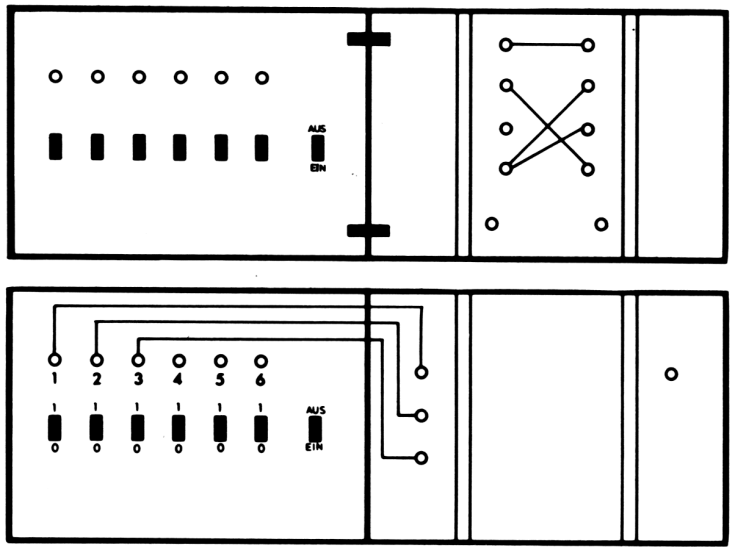


Abb. 100

Sicherlich bereitet Ihnen das Aufstellen einer Funktionstabelle oder das Programmieren der Logik-Bausteine nach einer vorgegebenen Funktionstabelle keine Schwierigkeiten mehr.
Sie haben deshalb mit den folgenden Beispielen die Möglichkeit zu prüfen, wie weit Sie schon mit dem PHILIPS Computer-Lehrbaukasten selbständig arbeiten können. Dabei ist entweder die Funktionstabelle anhand der Datenausgabe am Logik-Baustein ganz oder teilweise zu ergänzen oder die Programmierung aus der Funktionstabelle abzuleiten. Die Ergebnisse finden Sie am Ende dieses Abschnittes.

13.12. Schwierigkeiten bei der Geburtstagseinladung

Fritz hat Geburtstag. Er möchte gern seine Freunde Anton (A), Bernd (B) und Christian (C) einladen. Leider ergeben sich Schwierigkeiten, weil Anton und Christian nur gemeinsam kommen. Da Anton mit Bernd im Streit liegt, kann er sie beide nicht zusammen einladen. Welche Möglichkeiten ergeben sich, mit seinen Freunden zu feiern (F = 1)?

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

(Lösung am Ende dieses Kapitels)

Wenn Sie anhand der folgenden Schaltung mit einem Logik-Baustein und der Eingabeeinheit die Lösung gefunden haben, können Sie die Funktionstabelle vervollständigen.

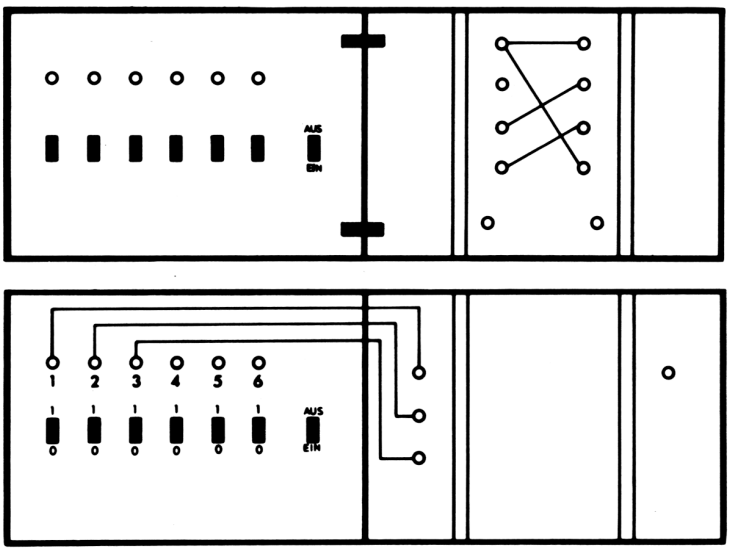


Abb. 101

13.13. Temperaturabhängige Regelung einer Heizungsanlage

In einem Einfamilienhaus soll eine Zentralheizung mit Warmwasseraufbereitung installiert werden. Der Bauherr verlangt, daß die Anlage durch drei Thermostate (A, B und C) geregelt wird. Der Kesselthermostat A soll den Brenner einschalten, wenn die Wassertemperatur unter 80° absinkt. Thermostat B mißt die Außentemperatur, C die Zimmertemperatur. Sinkt die Außentemperatur unter 18° C, und die Zimmertemperatur unter 20° C, so soll der Brenner ebenfalls in Betrieb gesetzt werden. Das bedeutet:

Der Brenner soll zünden (F = 1), wenn 1. $A < 80^{\circ}$ C (A = 1)
oder 2. $B < 18^{\circ}$ C (B = 1) und
$C < 20^{\circ}$ C (C = 1)

Das Zeichen $<$ bedeutet: kleiner als.

A	B	C	F

(Lösung am Ende dieses Kapitels)

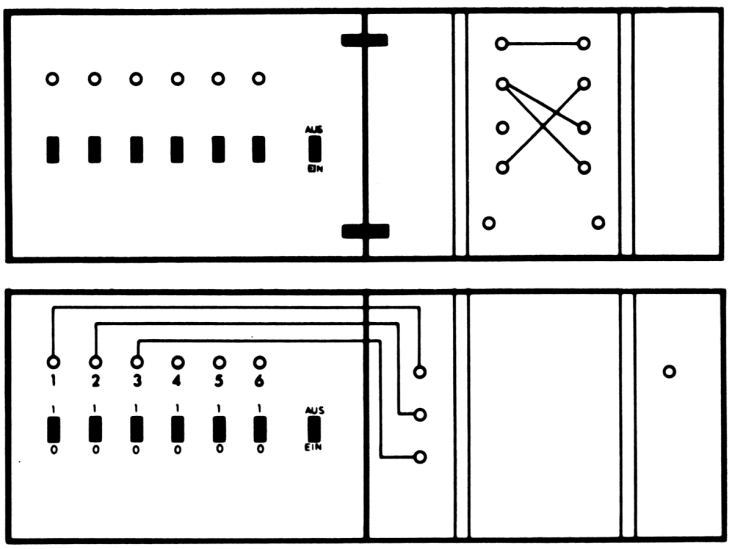


Abb. 102

13.14. Überwachung einer Fließbandfertigung

In einer Fertigungsstraße arbeiten drei Maschinen A, B und C in dieser Reihenfolge hintereinander. Um Schäden an der Anlage zu vermeiden, soll sie stillgelegt werden (F = 0), wenn eine Maschine ausfällt (A, B, C = 0) und mindestens eine vor ihr im Arbeitsablauf noch in Betrieb ist. Die Fertigungsstraße braucht nicht ausgeschaltet zu werden, wenn A steht, B und C aber weiterarbeiten oder wenn A und B stehen, C dagegen läuft. Fallen alle Maschinen aus, soll die gesamte Anlage abgeschaltet werden (F = 0).

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Aus dieser Funktionstabelle leiten Sie die Programmierung des einen Logik-Bausteins ab. Sollten Sie nicht ganz sicher sein, können Sie noch einmal auf Seite 7-1 nachschlagen.

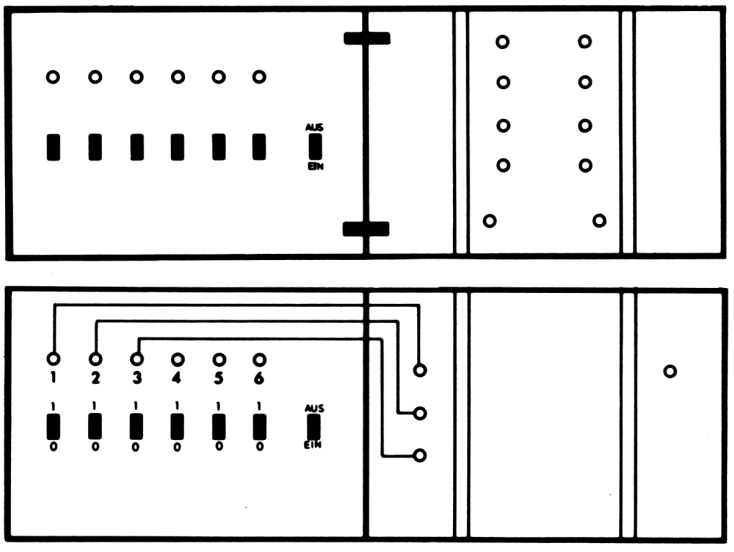


Abb. 103

13.15. Planung einer Ferienreise

Drei Klassenkameraden - Arno (A), Bertram (B) und Claus (C) - überlegen, ob sie in den Sommerferien eine gemeinsame Fahrt (F) unternehmen sollen. Schon bei der Festlegung des Reiseziels ergeben sich Differenzen. Dabei stellt sich heraus, daß Claus sich stets mit seiner Meinung Arno anschließt, Bertram dagegen immer entgegengesetzter Ansicht ist als Arno. Claus möchte es sich allerdings auch nicht mit Bertram verderben, und deshalb verspricht er, ihm zuzustimmen, wenn er einen Vorschlag annimmt.

Wer kann mit wem verreisen?

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Können Sie nach dieser Funktionstabelle die Programmierung und zusätzlich die Schaltung verwirklichen?

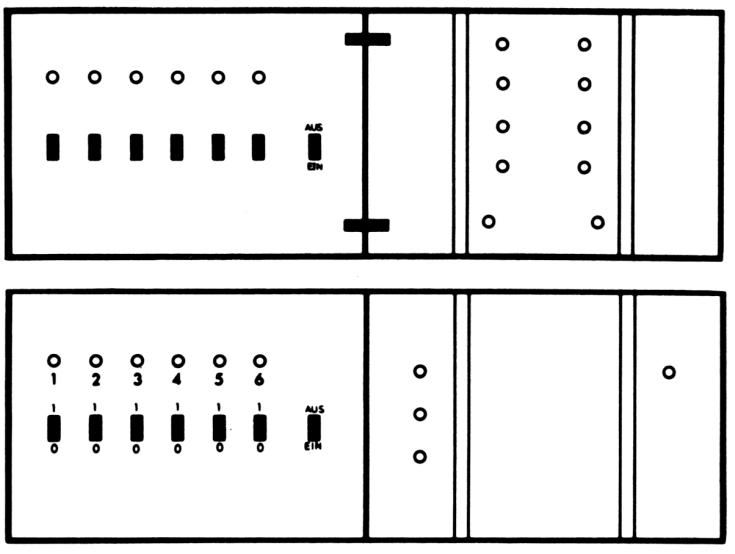


Abb. 1o4

(Lösung am Ende des Kapitels)

Lösungen zum Kapitel 13:

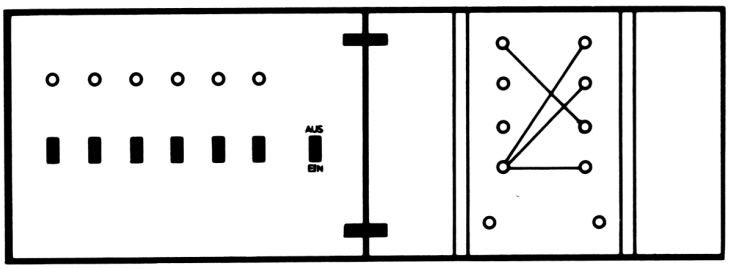
13.12

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

13.13

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

13.14



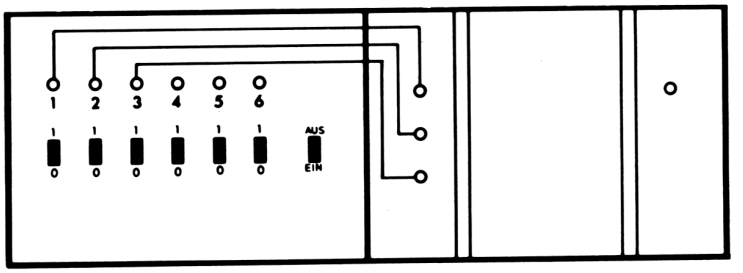
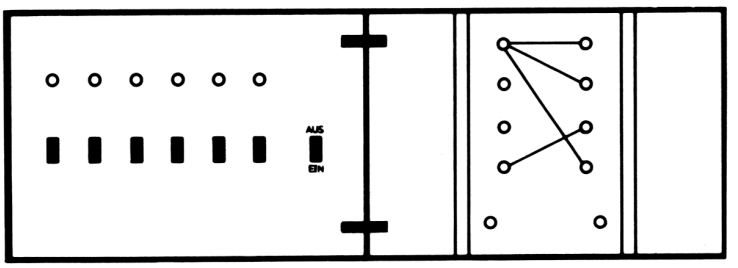


Abb. 1o5

13.15



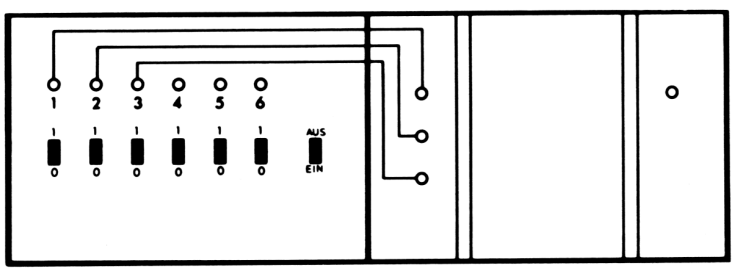


Abb. 1o6

14. Funktionen mit mehr als 3 Eingängen

Logische Funktionen mit 2 oder 3 Eingängen und deren Realisierungen haben Sie in den Kapiteln 3, 5 und 6 des Anleitungsbuches zum PHILIPS Computer-Lehrbaukasten kennengelernt. Sofern die Zahl der Eingangsvariablen 3 nicht überschreitet, läßt sich jede Funktion mit einem Logik-Baustein und einer Eingabeeinheit darstellen. Erst wenn mehr Eingangsbedingungen logisch verknüpft werden sollen, müssen weitere Logik-Bausteine eingesetzt werden. Nebenbei bemerkt: Die PHILIPS-Logik-Bausteine sind ja so aufwendig konstruiert, daß nicht nur je eine Grundfunktion mit maximal 3 Eingängen geschaltet werden kann, sondern auch eine Folge oder Kombination einiger Funktionen.

Wie nun mehr als drei Eingangsvariable durch Logik-Bausteine miteinander verbunden werden können, können Sie in den folgenden Beispielen erproben.

14.1. UND mit 4 bis 6 Eingangsvariablen

Sie wissen, daß bei einer UND-Funktion alle Bedingungen erfüllt sein müssen, bevor der Ausgang F = 1 ist. Für ein UND mit z. B. 6 Eingängen läßt sich natürlich auch eine Funktionstabelle aufstellen, doch aus zwei Gründen soll auf die komplette Tabelle verzichtet werden: Zum einen ergeben sich bei 6 Variablen $2^6 = 64$ Möglichkeiten, so daß eine solche Tabelle zu umfangreich wird. Wir wollen uns für die UND-Funktion auf nur 3 der 64 Möglichkeiten beschränken:

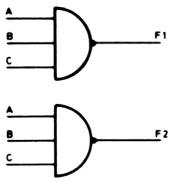
A	B	C	D	E	G	F
0	0	0	0	0	0	0
.			.			.
.			.			.
.			.			.
1	1	1	1	1	0	0
1	1	1	1	1	1	1

Der zweite Grund besteht darin, daß die Programmierung und Schaltung des kompletten Beispiels aus einer Gesamtfunktionstabelle nicht zu entnehmen ist, sondern es muß für jeden Logik-Baustein eine eigene Wertetafel aufgestellt werden.

Wie sieht nun der Aufbau eines UND mit z. B. 6 Eingängen aus? Ein Logik-Baustein kann nur maximal 3 Eingänge logisch verknüpfen, also müssen 2 Bausteine parallel eingesetzt werden.

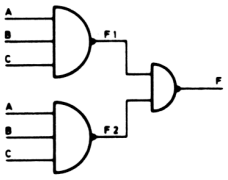
A	B	C	F1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

D	E	G	F2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Jeder der beiden Bausteine wird als UND programmiert, wie Sie es bereits mehrfach durchgeführt haben (vergl. Kap. 6.1.). Sie arbeiten zunächst noch völlig unabhängig voneinander. Deshalb wird diesen beiden Bausteinen ein weiterer nachgeschaltet, der ebenfalls als UND (mit 2 Eingängen) programmiert ist. Erst wenn alle 6 Eingänge den Zustand 1 führen, liegen an F1 und F2 ebenfalls 1-Signale. Sie rufen in dem nachgeschalteten UND F = 1 hervor.

F1	F2	F
0	0	0
0	1	0
1	0	0
1	1	1



Nach der folgenden Programmierung und Verdrahtung läßt sich das UND mit 6 Eingängen aufbauen.

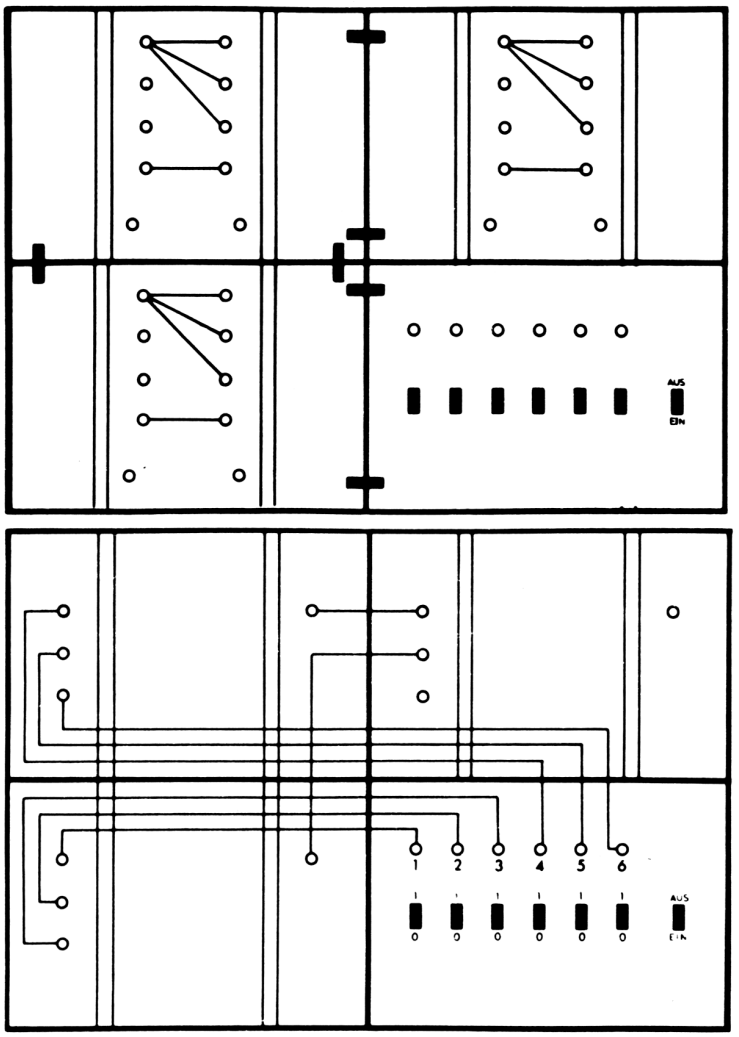


Abb. 1o7

Eine zusätzliche Aufgabe: Lassen sich UND-Funktionen mit 4 Eingangsvariablen mit nur zwei Logik-Bausteinen realisieren? Versuchen Sie, die Programmierung und Verdrahtung für diese Beispiele selbst zu finden. Bedenken Sie, daß im zweiten Logik-Baustein noch Eingänge frei sind. (Lösung am Ende des Kapitels)

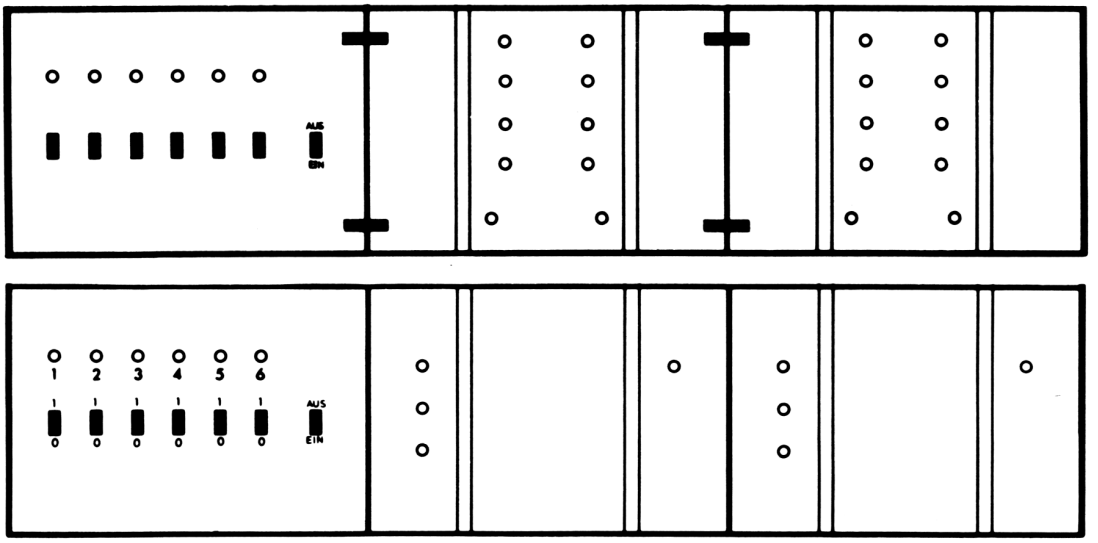
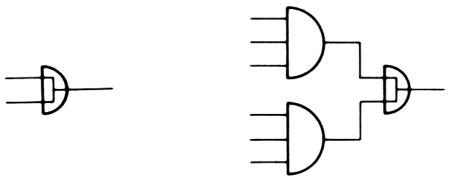


Abb. 1o8

Bei den UND-Funktionen mit mehr als 3 Eingängen haben Sie jeweils die Ausgänge auf einen nachgeschalteten Logik-Baustein gelegt, um die gesamte Schaltung zu realisieren. Durch die besondere Schaltungstechnik der PHILIPS-Computer-Bausteine kann der nachgeschaltete Logik-Baustein durch Parallelschaltung der Ausgänge eingespart werden. Man spricht dann von einem "Wired AND", einem "verbundenen UND". Für dieses Wired AND gibt es ein eigenes Logiksymbol, das die galvanische Verbindung der Ausgänge ausdrückt.



Die Programmierung und Verdrahtung für ein Wired AND mit 6 Eingängen können Sie der folgenden Abbildung entnehmen.

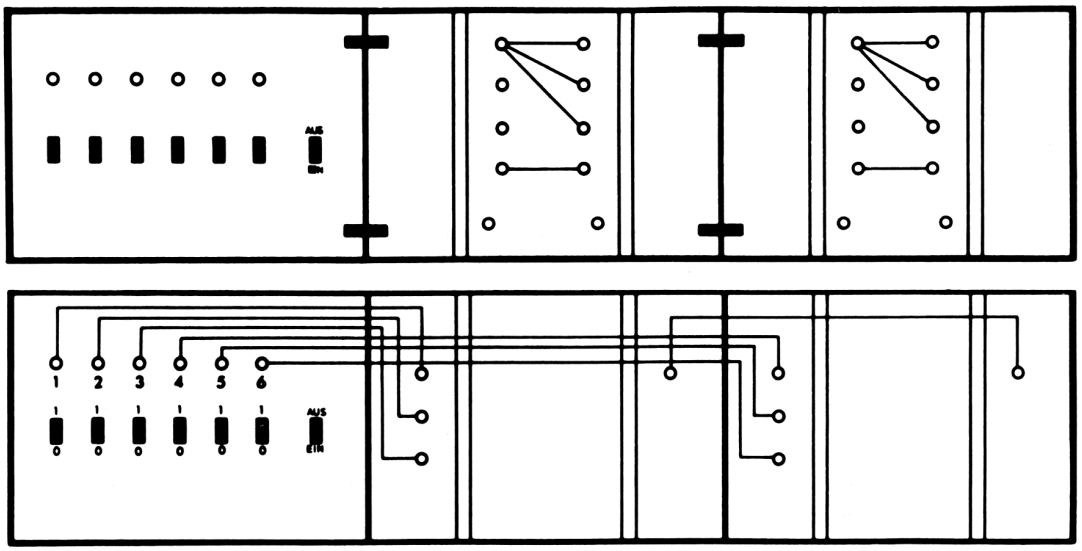
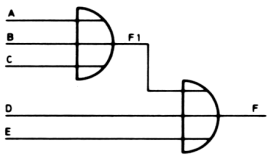


Abb. 109

In dieser Weise können maximal 4 Bausteine untereinander verbunden werden. Bei mehr als vier Bausteinen kann jedoch im ungünstigsten Fall durch zu hohen Ausgangsstrom ein Baustein zerstört werden.

14.2. ODER mit 4 bis 6 Eingangsvariablen

Ähnlich wie eine UND-Funktion läßt sich auch ein ODER mit mehr als 3 Eingangsvariablen ausführen. Die verwendeten Logik-Bausteine müssen ebenfalls nachgeschaltet werden. An einer ODER-Funktion mit 5 Eingängen soll das verdeutlicht werden: Baustein 1 wird wie ein ODER mit 3 Eingängen programmiert und verdrahtet (vergl. Kap. 6.2). Der zweite Logik-Baustein wird ebenso programmiert. Die Eingänge A und B werden mit den beiden verbleibenden Eingangsvariablen belegt, der Eingang C wird mit F des ersten Bausteins verbunden.



Vergegenwärtigen Sie sich, bitte, noch einmal die ODER-Funktion: F ist 1, wenn A oder B oder C oder D oder E den Zustand 1 führt. Aus der Logikschaltung können Sie entnehmen, daß diese Bedingungen erfüllt werden. Führt z. B. A den Zustand 1, so ist F1 = 1, und am Eingang C des nachgeschalteten Bausteins liegt ein 1-Signal. Damit ist F = 1.

Und hier die vollständige Programmierung und Verdrahtung:

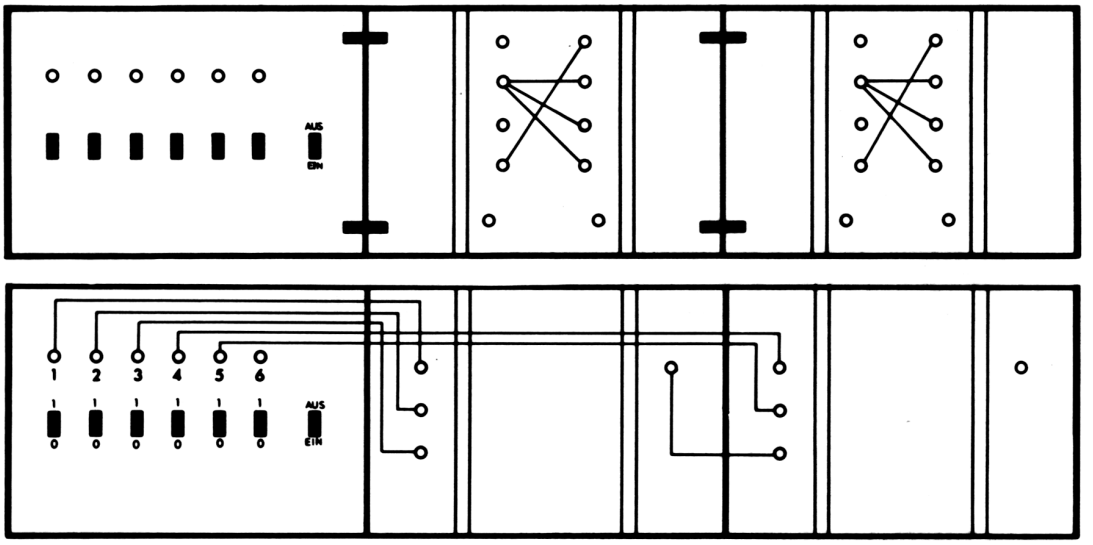


Abb. 11o

Falls es Ihnen Freude macht, können Sie noch ein paar Aufgaben lösen:

1. Stellen Sie die Gesamtfunktionstabelle für eine ODER-Funktion mit 5 Eingangsvariablen auf - es gibt $2^5 = 32$ Möglichkeiten - und überprüfen Sie die Tabelle mit Ihrem Aufbau!

	A	B	C	D	E	F
1.						
2.						
3.						
4.						
5.						
6.						
7.						
8.						
9.						
1o.						
11.						
12.						
13.						
14.						
15.						
16.						
17.						
18.						
19.						
2o.						
21.						
22.						
23.						
24.						
25.						
26.						
27.						
28.						
29.						
3o.						
31.						
32.						

(Lösung am Ende des Kapitels)

2. Bauen Sie ein ODER mit 6 Eingangsvariablen auf!

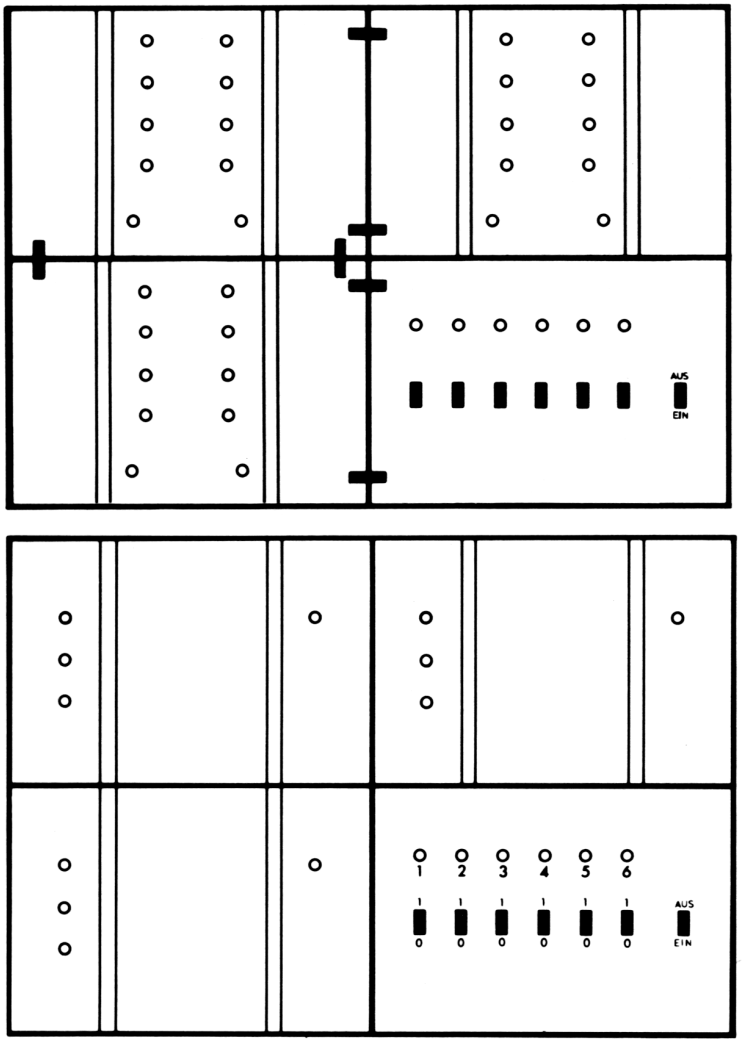


Abb. 111

(Lösung am Ende des Kapitels)

14.3. NAND mit 6 Eingangsvariablen

Ebenso wie UND und ODER lassen sich NAND-Funktionen mit mehr als 3 Eingangsvariablen realisieren. Der entscheidende Unterschied besteht allerdings darin, daß nur die beiden vorgeschalteten Bausteine als NAND programmiert werden dürfen. Wie die Programmierung des dritten aussehen muß, soll an diesem Beispiel ausführlich erklärt werden. Sie können bestimmt die Funktionstabelle für ein NAND ableiten (vergl. Kap. 6.5). Daraus entnehmen Sie, daß F immer dann 1 ist, wenn mindestens eine Eingangsvariable den Zustand 0 führt. Erst wenn alle 1 sind, ist F = 0. Der nachgeschaltete Baustein kann also nicht als NAND programmiert werden, da sonst die Ausgangssignale der beiden vorgeschalteten als UND mit negiertem Ausgang verknüpft würden. Der dritte Baustein muß folgende Bedingung erfüllen: Wenn F1 und F2 = 0 sind, muß F auch 0 sein. Die Funktionstabelle dafür:

A (F1)	B (F2)	F
0	0	0
0	1	1
1	0	1
1	1	1

Sie sehen, daß die Tabelle der einer ODER-Funktion mit 2 Eingängen entspricht.

Und nun die vollständige Schaltung:

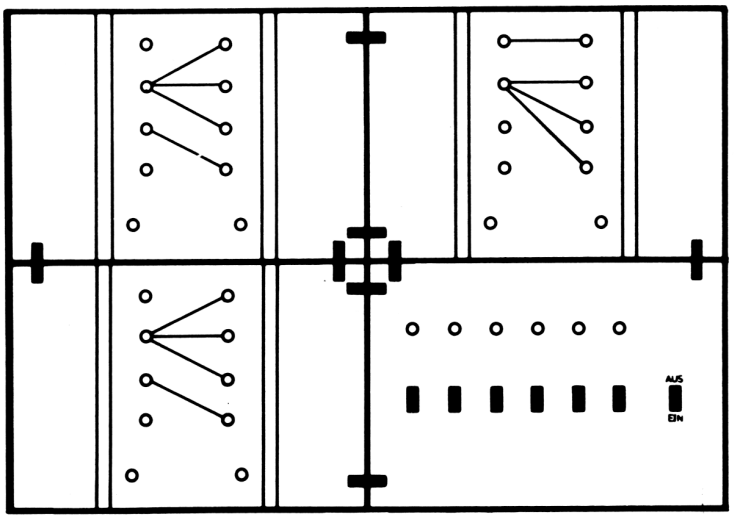


Abb. 112 a

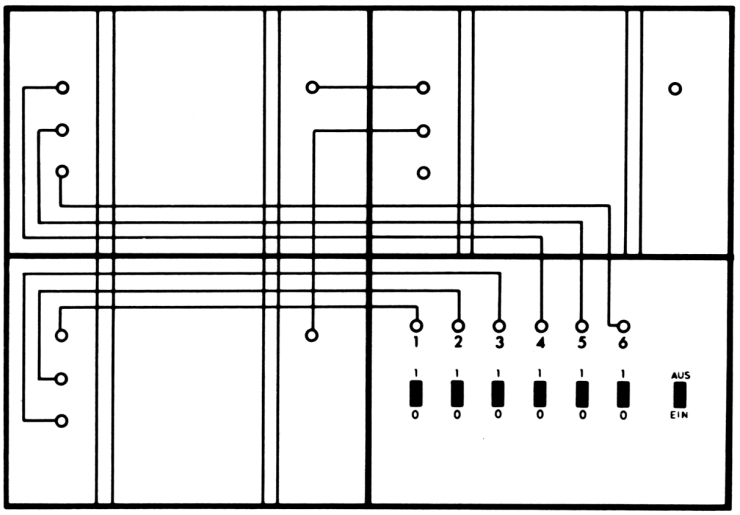


Abb. 112 b

Eine Aufgabe zum Durchdenken und Ausprobieren: Läßt sich eine NAND-Funktion mit 5 Eingangsvariablen ebenso wie das ODER mit 5 Eingängen aus 2 Logik-Bausteinen und einer Eingabeeinheit aufbauen?

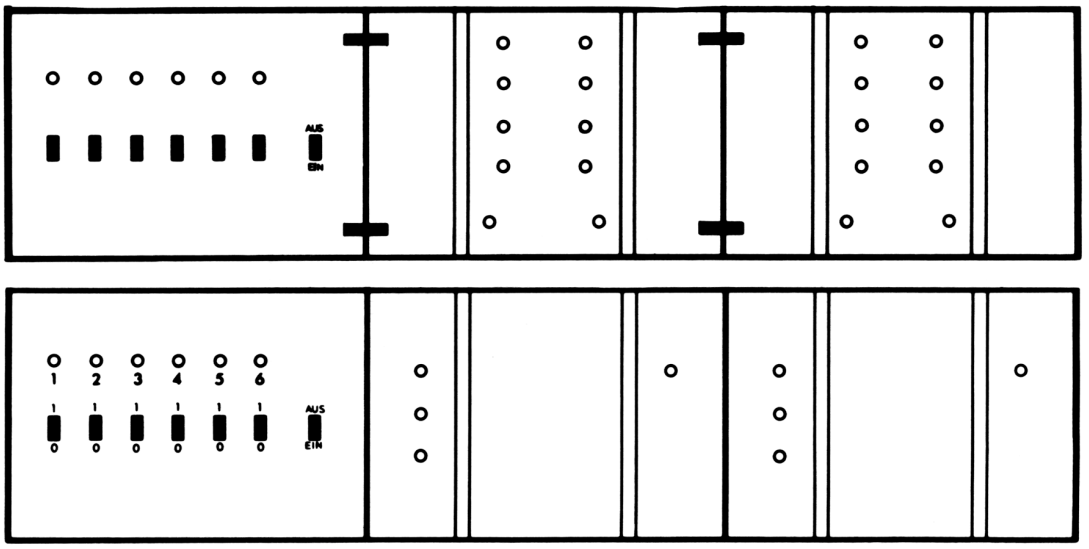


Abb. 113

(Lösung am Ende dieses Kapitels)

14.4. NOR mit 6 Eingangsvariablen

Um ein NOR mit 6 Eingangsvariablen aufzubauen, soll - wie beim NAND - ebenfalls nur der nachgeschaltete Logik-Baustein genauer betrachtet werden. An ihn ist die Bedingung geknüpft, daß F nur dann 1 sein darf, wenn F1 und F2 gleich 1 sind. (F1 und F2 haben übrigens immer dann am Ausgang ein 1-Signal, wenn alle Eingänge 0 sind.)

Die Funktionstabelle für den dritten Baustein sieht so aus:

A (F1	B (F2)	F
0	0	0
0	1	0
1	0	0
1	1	1

Sie erkennen sofort, daß diese Tabelle ein UND mit 2 Eingangsvariablen darstellt. Die beiden vorgeschalteten Logik-Bausteine müssen natürlich als NOR programmiert werden. Damit steht die gesamte Schaltung für ein NOR mit 6 Eingangsvariablen fest:

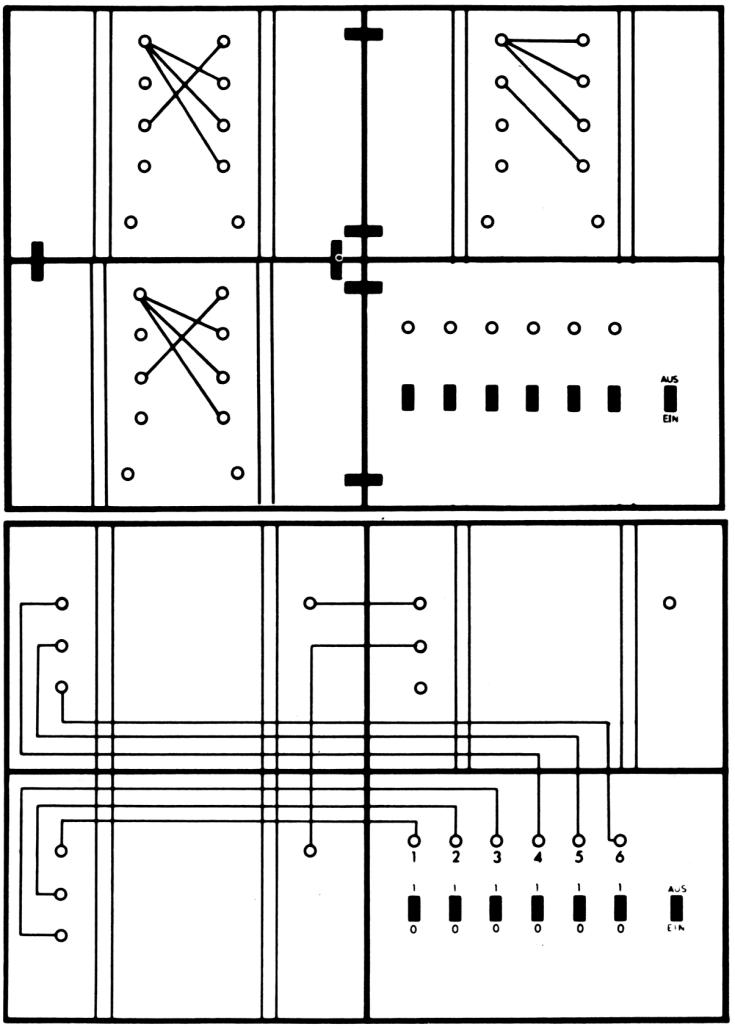


Abb. 114

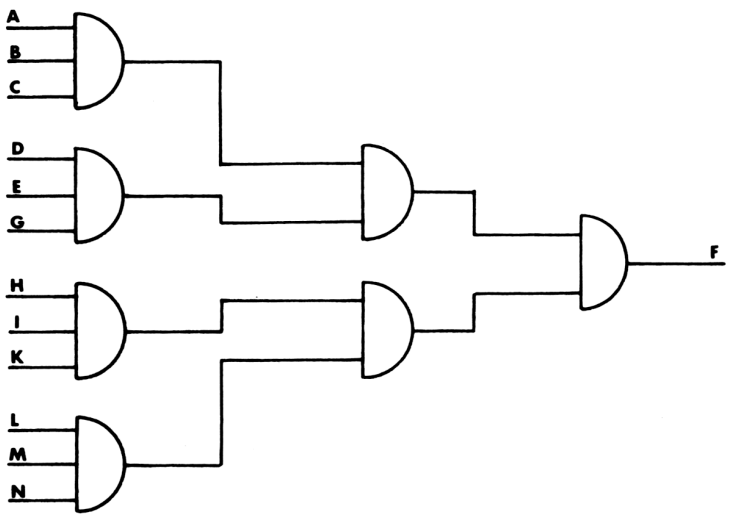
14.5. Logische Schaltungen mit 12 Eingangsvariablen

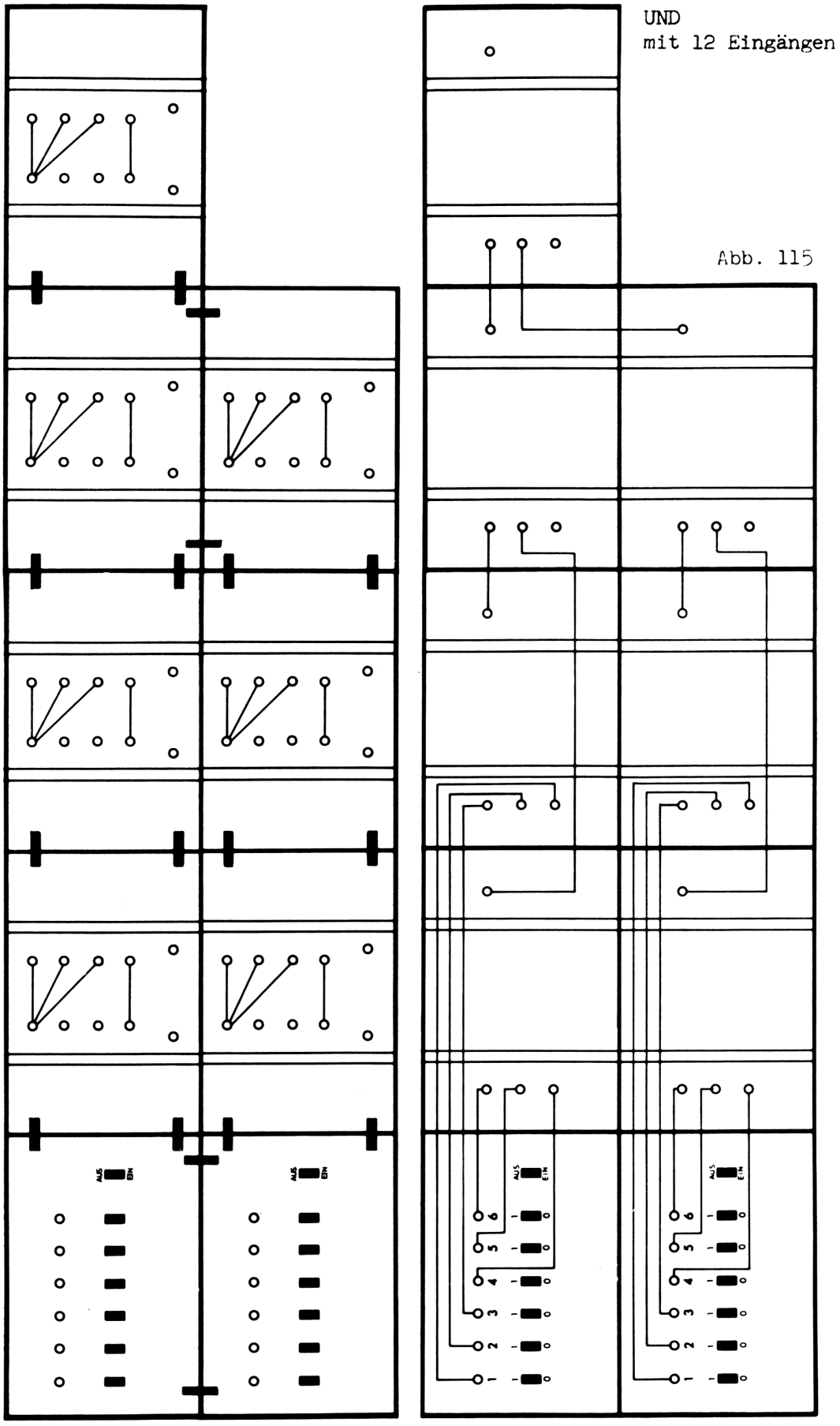
Ebenso wie Beispiele mit 6 Eingangsvariablen auftauchen, sind solche möglich, bei denen diese Zahl noch überschritten wird. So ist z.B. eine UND-Funktion mit 12 Eingangsvariablen oder ein NAND mit 10 denkbar. Für UND-, ODER-, NAND- und NOR-Funktionen sollen im folgenden die Programmierungen und Verdrahtungen mit 12 Eingängen dargestellt werden. Selbstverständlich lassen sich auch sprachliche Beispiele finden, doch da sie sich im Prinzip nicht von denen für z. B. 3 Eingänge unterscheiden, soll darauf verzichtet werden.

Zu diesen schon ziemlich umfangreichen Schaltungen ein wichtiger Hinweis:

> Wird eine der Buchsen f1 bis f4 im Programmierfeld nicht mit 0, 1, $\bar{C}$ oder C verbunden, so nimmt sie automatisch den Zustand 1 an. Alle 1-Verbindungen werden deshalb bei den folgenden Programmierungen nicht mitgezeichnet.

UND mit 12 Eingängen



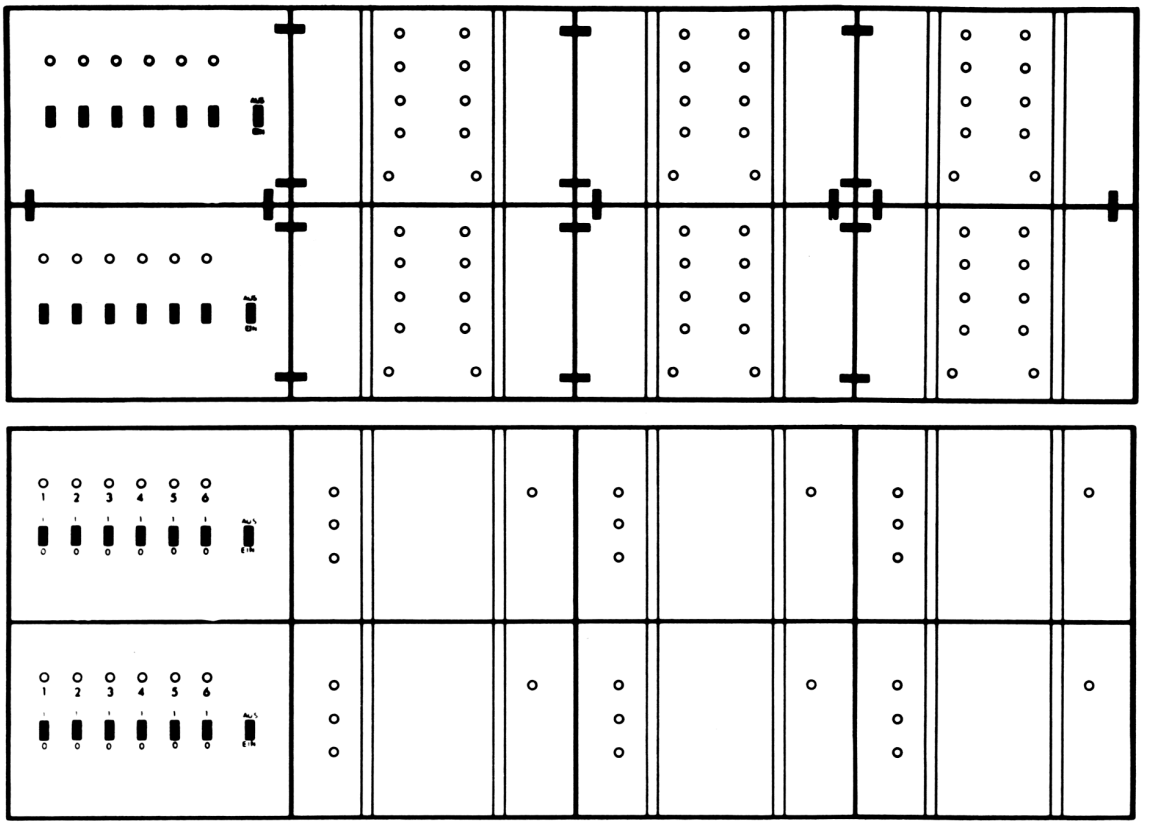


UND
mit 12 Eingängen

Abb. 115

Aufgabe:

1. Das UND mit 12 Eingängen läßt sich mit einem Baustein weniger aufbauen.



2. Durch das Wired AND kann die Schaltung mit 4 Bausteinen realisiert werden.

Abb. 116

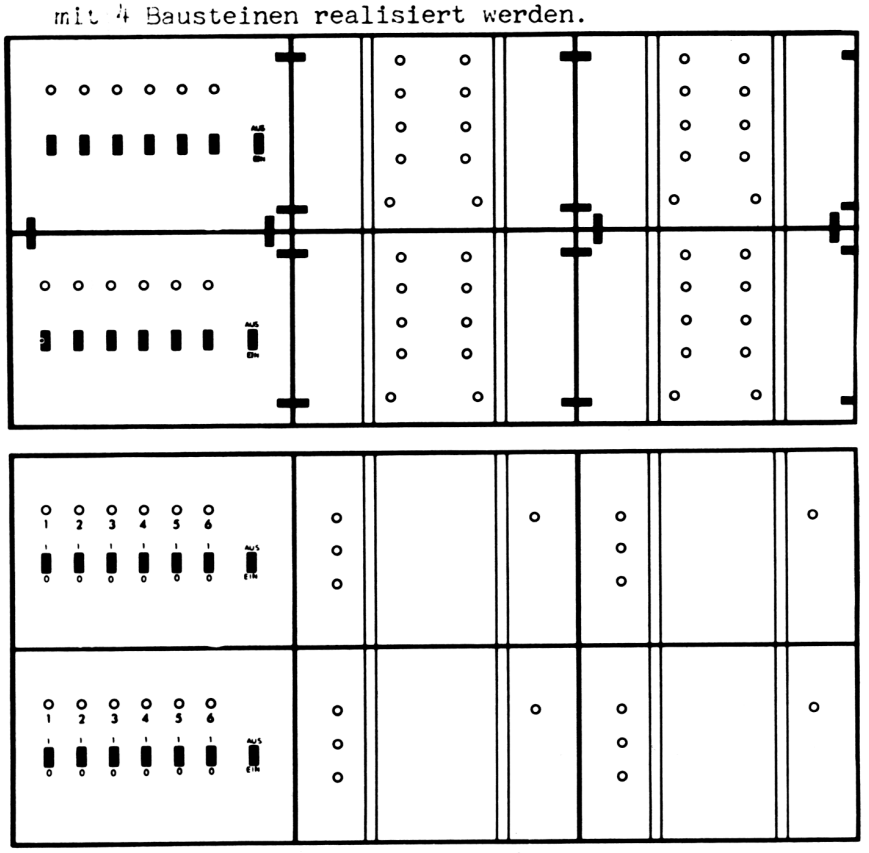


Abb. 117

(Lösungen am Ende dieses Kapitels)

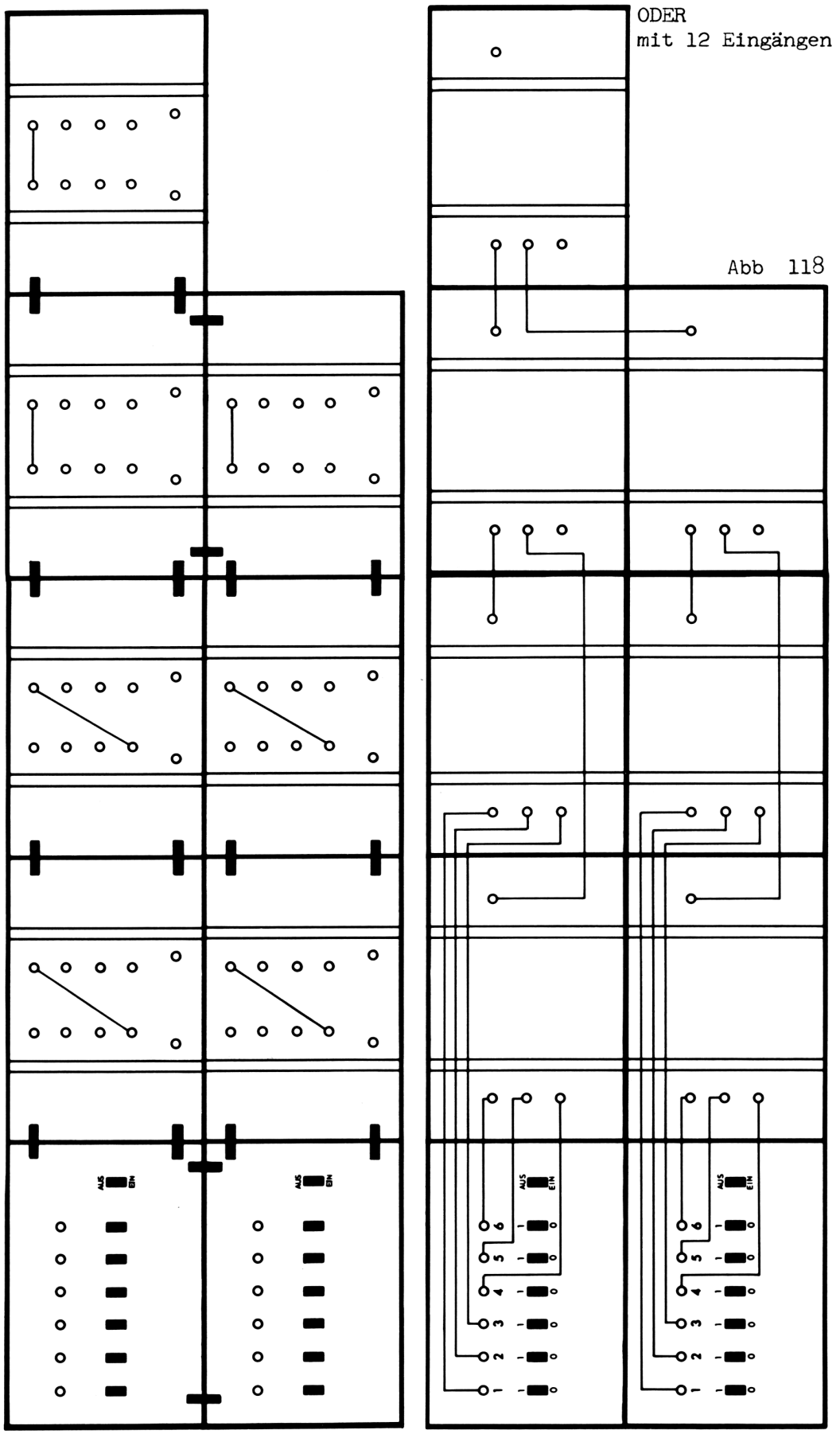
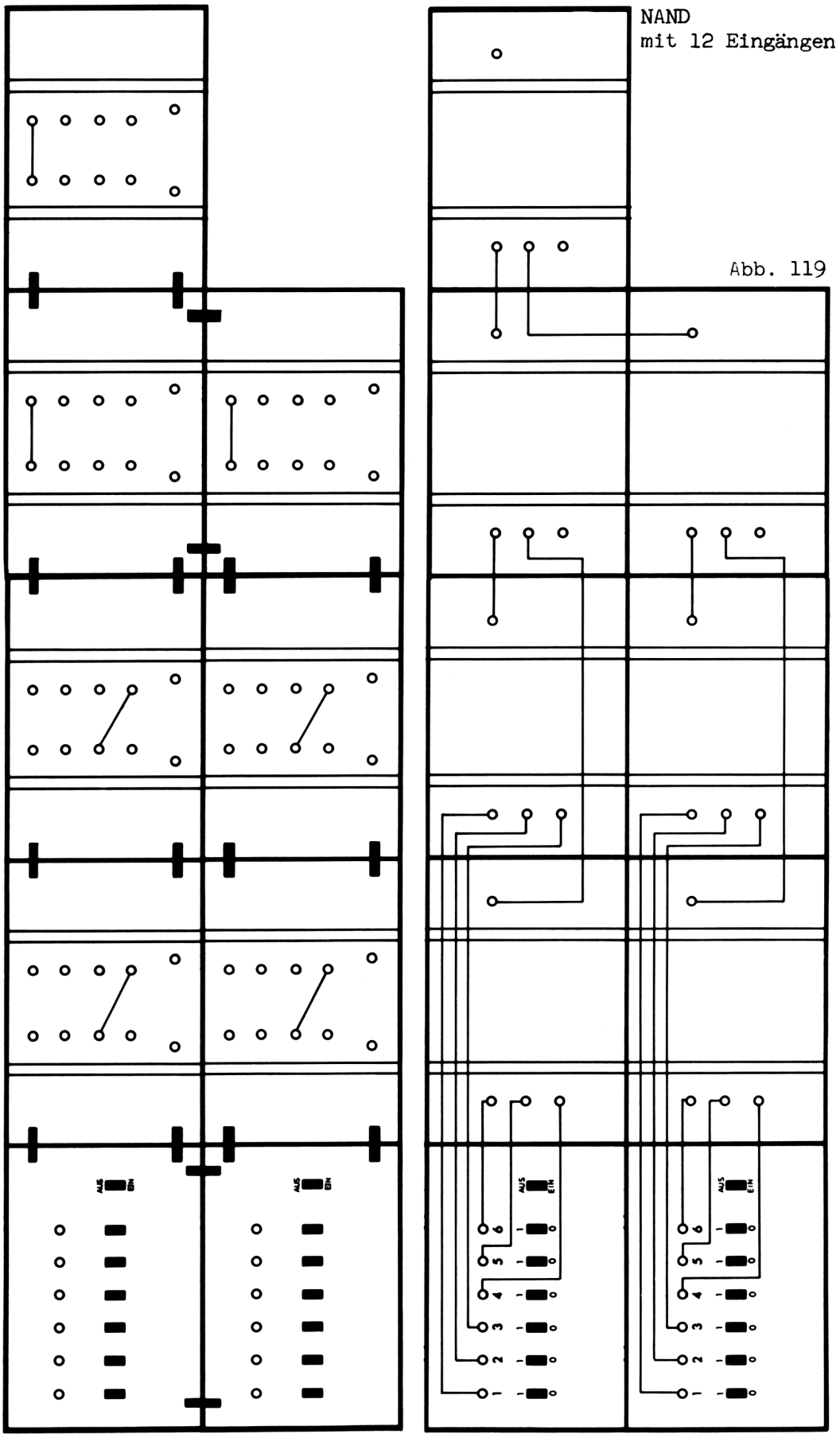
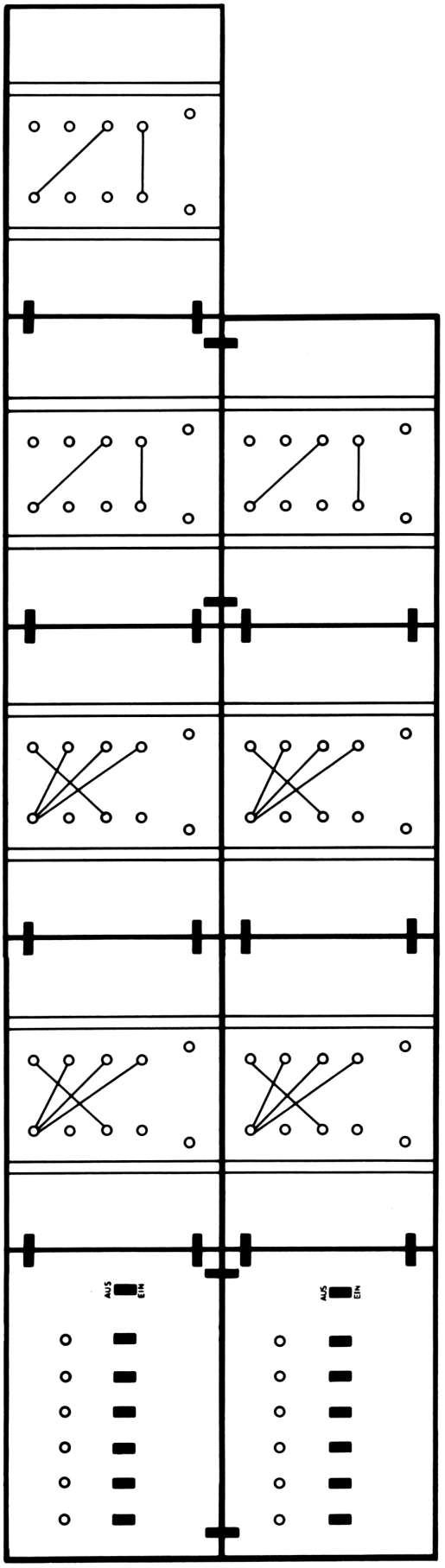


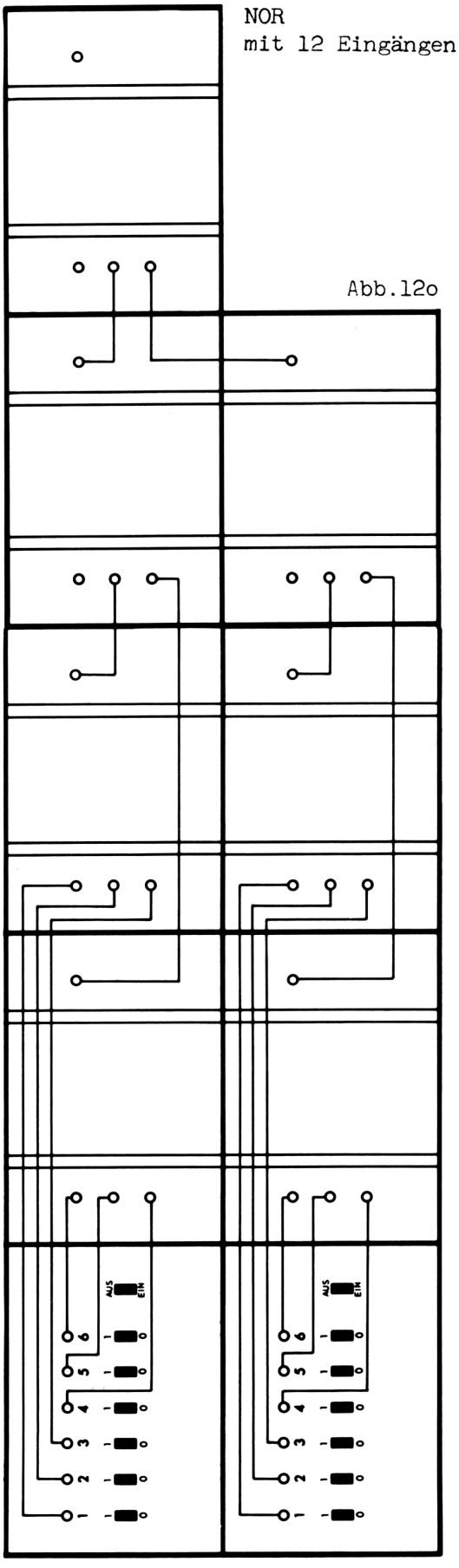
Abb 118



NAND
mit 12 Eingängen

Abb. 119





NOR
mit 12 Eingängen

Abb.12o

Aufgabe: 3. Das NOR läßt sich ebenfalls durch Wired AND vereinfachen.

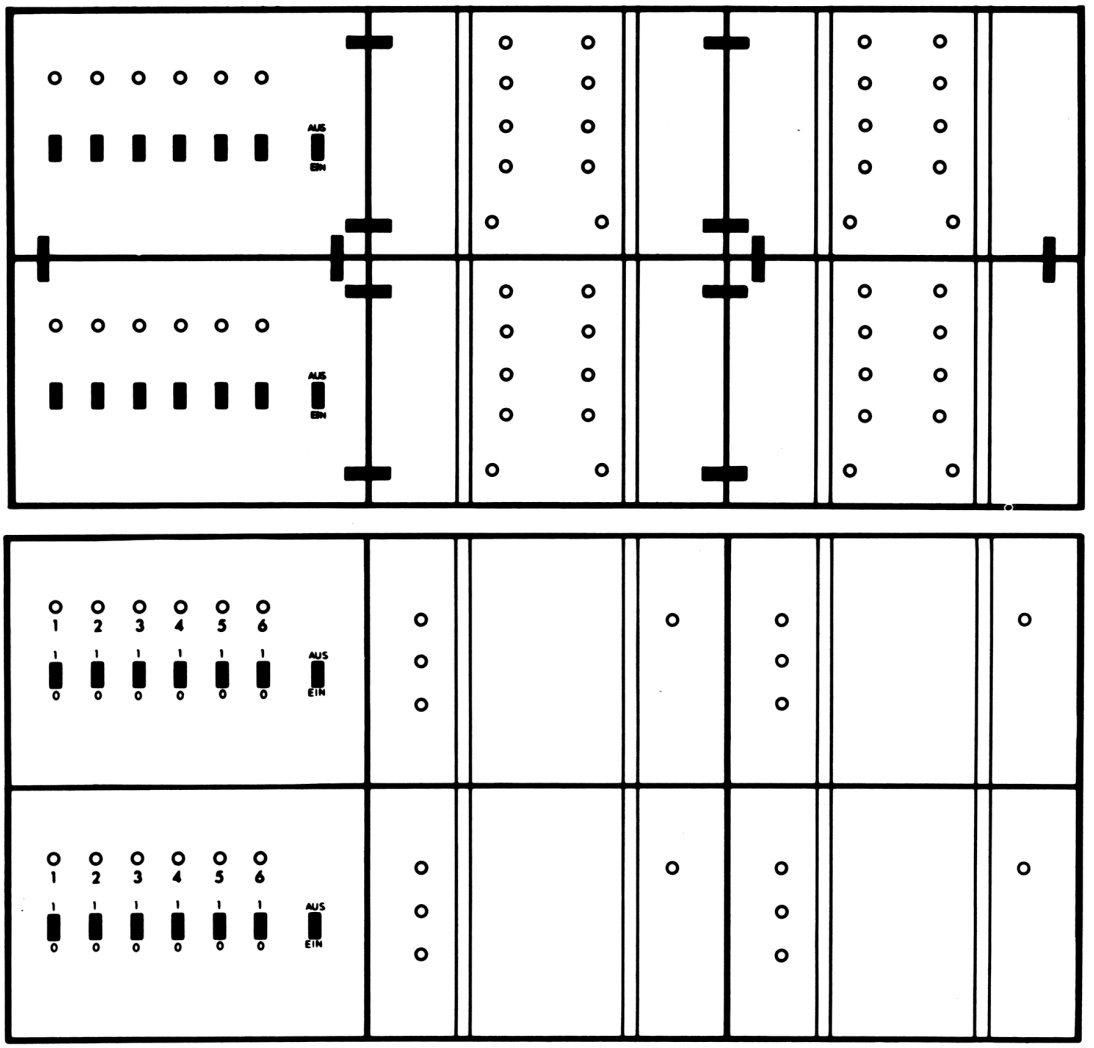


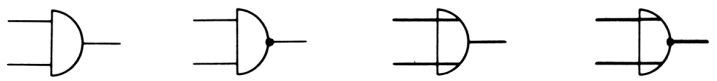
Abb.121

Hoffentlich ist Ihnen das Programmieren und Verdrahten nicht zu langwierig vorgekommen. Haben Sie einmal bedacht, wie lang eine Funktionstabelle für 12 Eingangsvariable ausfallen müßte? Sie enthält immerhin 2^{12} Möglichkeiten = 4096! Alle diese Kombinationen zu prüfen, ist nun überhaupt nicht notwendig, zumal es sich noch um die Grundfunktionen handelt und Sie ohnehin wissen, wann F = 1 sein soll.

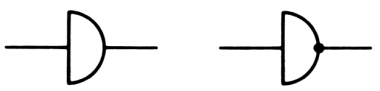
Mit diesem "Rüstzeug" sollte es Ihnen nun eigentlich keine Schwierigkeiten mehr bereiten, die folgenden aufwendigeren Beispiele zum PHILIPS Computer-Lehrbaukasten durchzuspielen und zu verstehen.

Zur besseren Übersicht finden Sie noch einmal eine Zusammenfassung der wichtigsten Funktionstabellen. Anweisungen zur Programmierung entnehmen Sie Kapitel 7.

		F			
A	B	UND	NAND	ODER	NOR
0	0	0	1	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	1	0



	F	
A	Identität	Negation (NICHT)
0	0	1
1	1	0

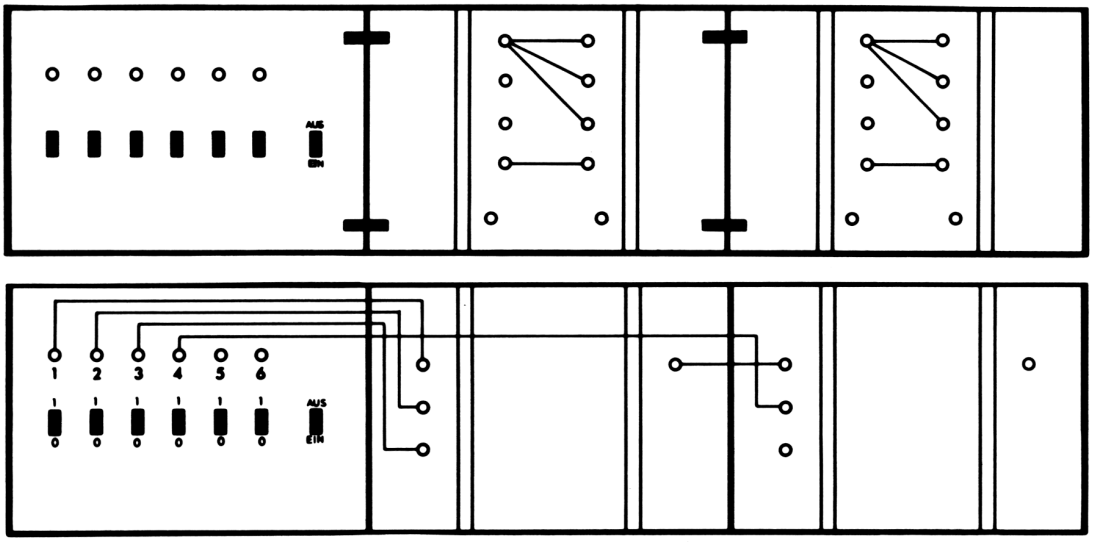


Lösungen zum Kapitel 14:

14.1.

A	B	C	Fl
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

D	E	Fl	F
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



14.2. 1.

	A	B	C	D	E	F
1.	0	0	0	0	0	0
2.	0	0	0	0	1	1
3.	0	0	0	1	0	1
4.	0	0	0	1	1	1
5.	0	0	1	0	0	1
6.	0	0	1	0	1	1
7.	0	0	1	1	0	1
8	0	0	1	1	1	1
9.	0	1	0	0	0	1
1o.	0	1	0	0	1	1
11.	0	1	0	1	0	1
12.	0	1	0	1	1	1
13.	0	1	1	0	0	1
14.	0	1	1	0	1	1
15.	0	1	1	1	0	1
16.	0	1	1	1	1	1
17.	1	0	0	0	0	1
18.	1	0	0	0	1	1
19.	1	0	0	1	0	1
2o.	1	0	0	1	1	1
21.	1	0	1	0	0	1
22.	1	0	1	0	1	1
23.	1	0	1	1	0	1
24.	1	0	1	1	1	1
25.	1	1	0	0	0	1
26.	1	1	0	0	1	1
27.	1	1	0	1	0	1
28.	1	1	0	1	1	1
29.	1	1	1	0	0	1
3o.	1	1	1	0	1	1
31.	1	1	1	1	0	1
32.	1	1	1	1	1	1

14.2. 2.

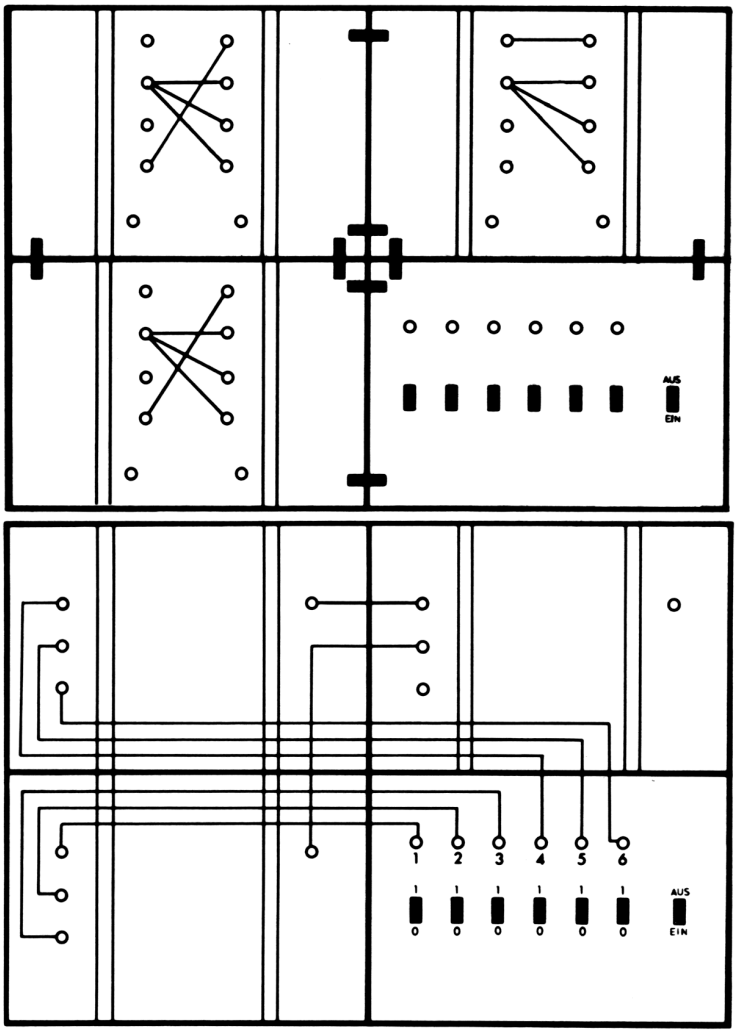


Abb.123

14.3.

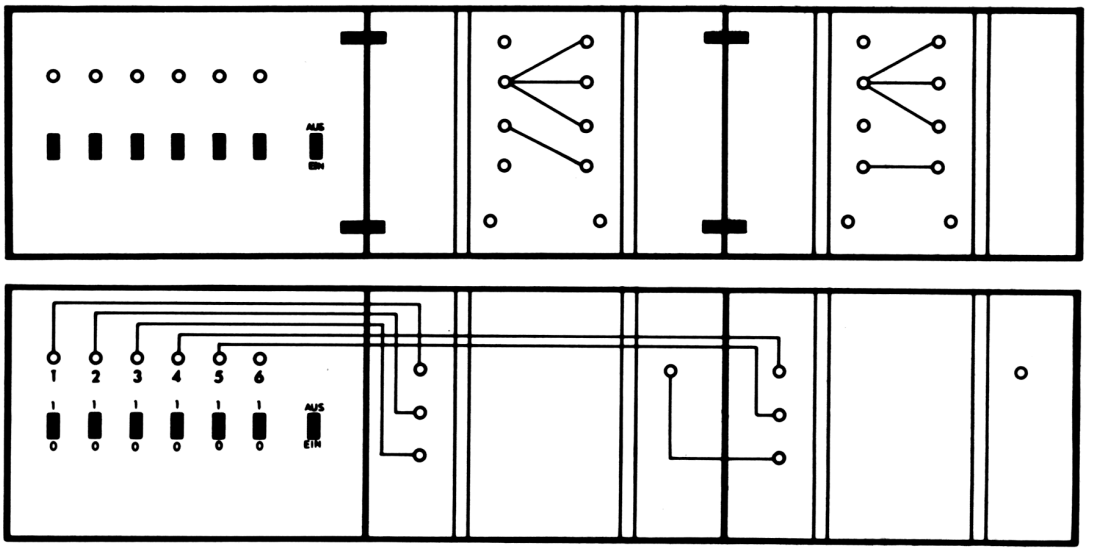


Abb.124

14.5. 1.

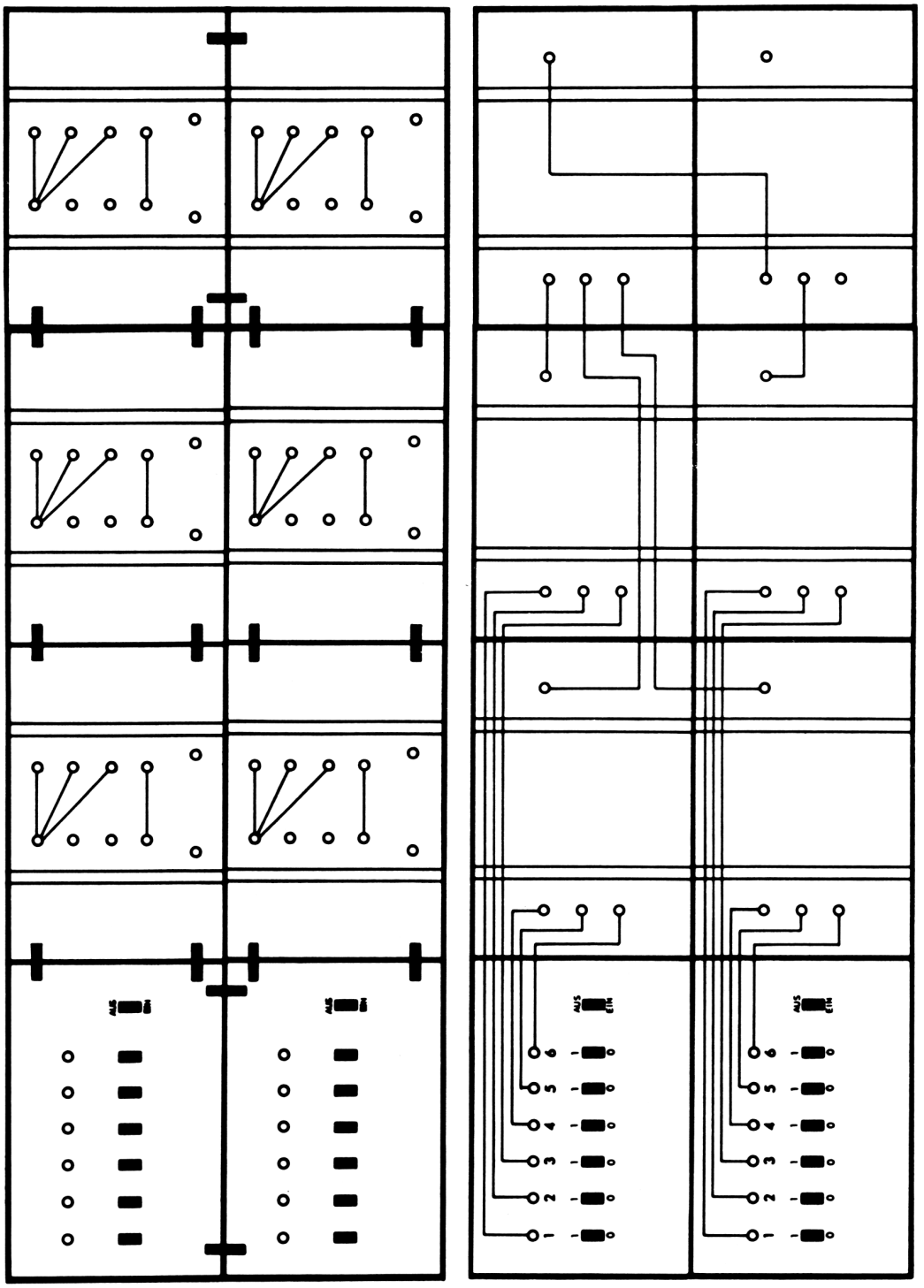


Abb.125

14.5.2

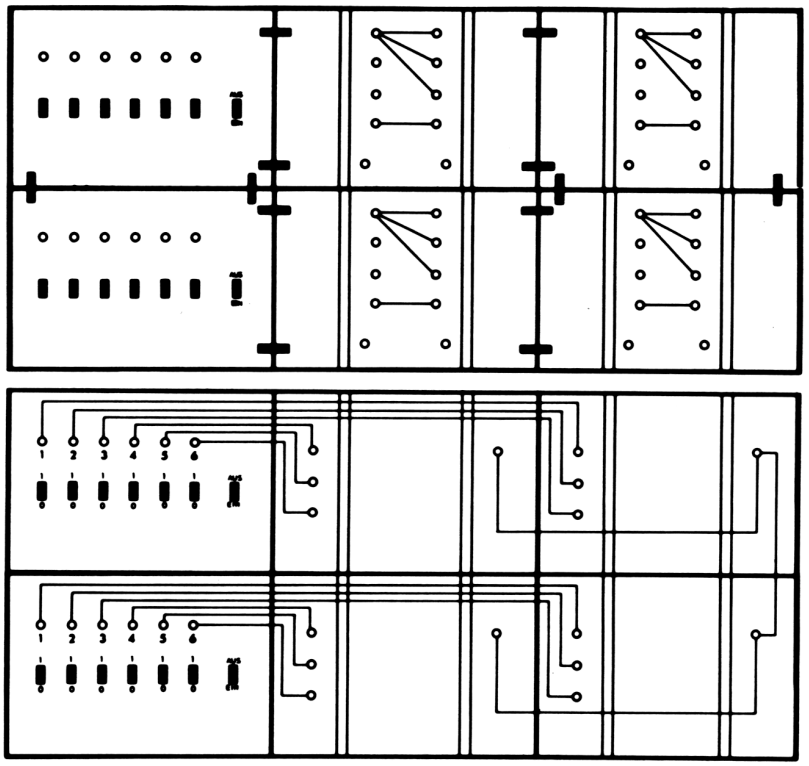


Abb.126

14.5.3

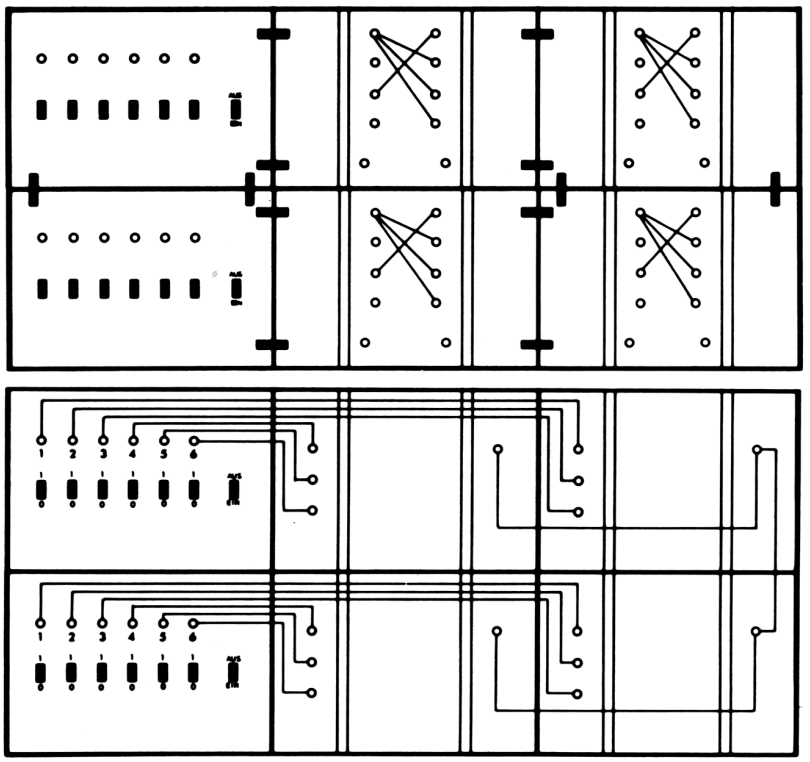


Abb.127

15. NAND und NOR als universelle Funktionen

In Kapitel 5/6 haben Sie die logischen Funktionen bzw. Schaltungen UND, ODER und deren Negationen NAND, NOR kennengelernt. Dabei wurden diese Funktionen durch Programmierung eines Logik-Bausteins dargestellt. Für den Bau von Computern ist es aus fertigungstechnischen Gründen von großer Bedeutung, möglichst aus einem Gatter und dessen Kombinationen alle logischen Funktionen zu verwirklichen. Als Grundschaltungen werden dafür die NAND- und NOR-Funktionen benutzt.

Wenden wir uns zunächst der NAND-Schaltung zu, um die Kombinationsmöglichkeiten zu untersuchen:
Stellen Sie sich bitte vor, Sie sollen nur aus NAND-Bausteinen - das entspricht der Forderung nach wirtschaftlicher Fertigung - verschiedene Grundfunktionen realisieren. Sie programmieren also alle Bausteine als NAND und erzielen durch eine entsprechende Verdrahtung die gewünschte Funktion.

15.1. NICHT aus NAND

Für die Realisierung benötigen Sie die Eingabeeinheit und einen Logik-Baustein, der wie in Abb.128 programmiert wird.

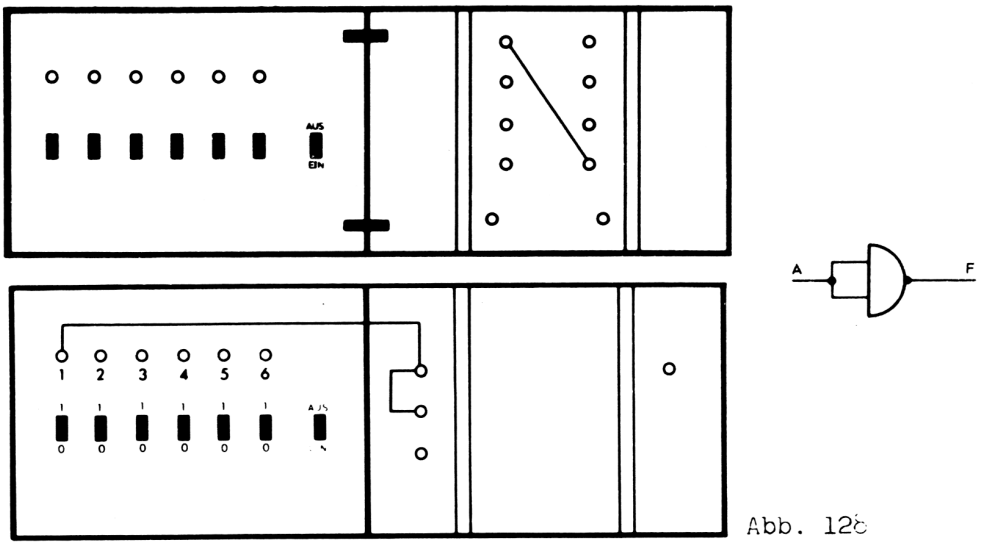


Abb. 128

Wenn Sie das Logiksymbol betrachten, fällt Ihnen auf, daß mit einem Schalter der Eingabeeinheit gleichzeitig beide Eingänge des Logik-Bausteins ein 0- bzw. 1-Signal erhalten, da die beiden Eingänge A und B des Logik-Bausteins parallel geschaltet sind. Deshalb können die in der Funktionstabelle mit einem Sternchen gekennzeichneten Zustände der NAND-Funktion nicht auftreten. Ein 0-Signal am Schalter der Eingabeeinheit bewirkt ein 1-Signal am Ausgang und umgekehrt. Damit haben Sie die Funktion eines NICHT erzielt.

A1 A	B	F
0	0	1
0	1	1 +
1	0	1 +
1	1	0

15.2. UND aus NAND

Soll mit NAND-Bausteinen eine UND-Funktion dargestellt werden, soll das durch die Kombination zweier als NAND programmierter Bausteine erreicht werden. Das entspricht einer doppelten Verneinung, die wieder eine positive Aussage ergibt.
Die UND-Funktion aus NAND kann aus einer Eingabeeinheit und zwei in Reihe geschalteten Logik-Bausteinen realisiert werden.

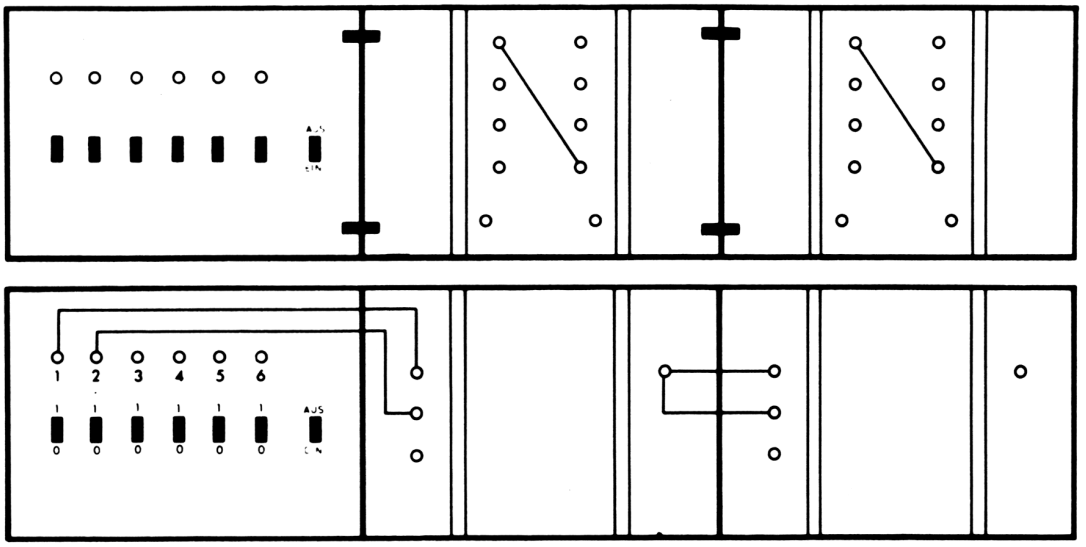
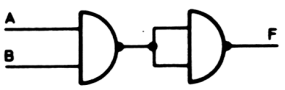


Abb. 129



Sie erkennen, daß der erste Baustein als NAND arbeitet, während der zweite das bekannte "NICHT aus NAND" erfüllt.

15.3. ODER aus NAND

Zur Realisierung eines ODER aus NAND führen Sie folgende Verdrahtung durch:

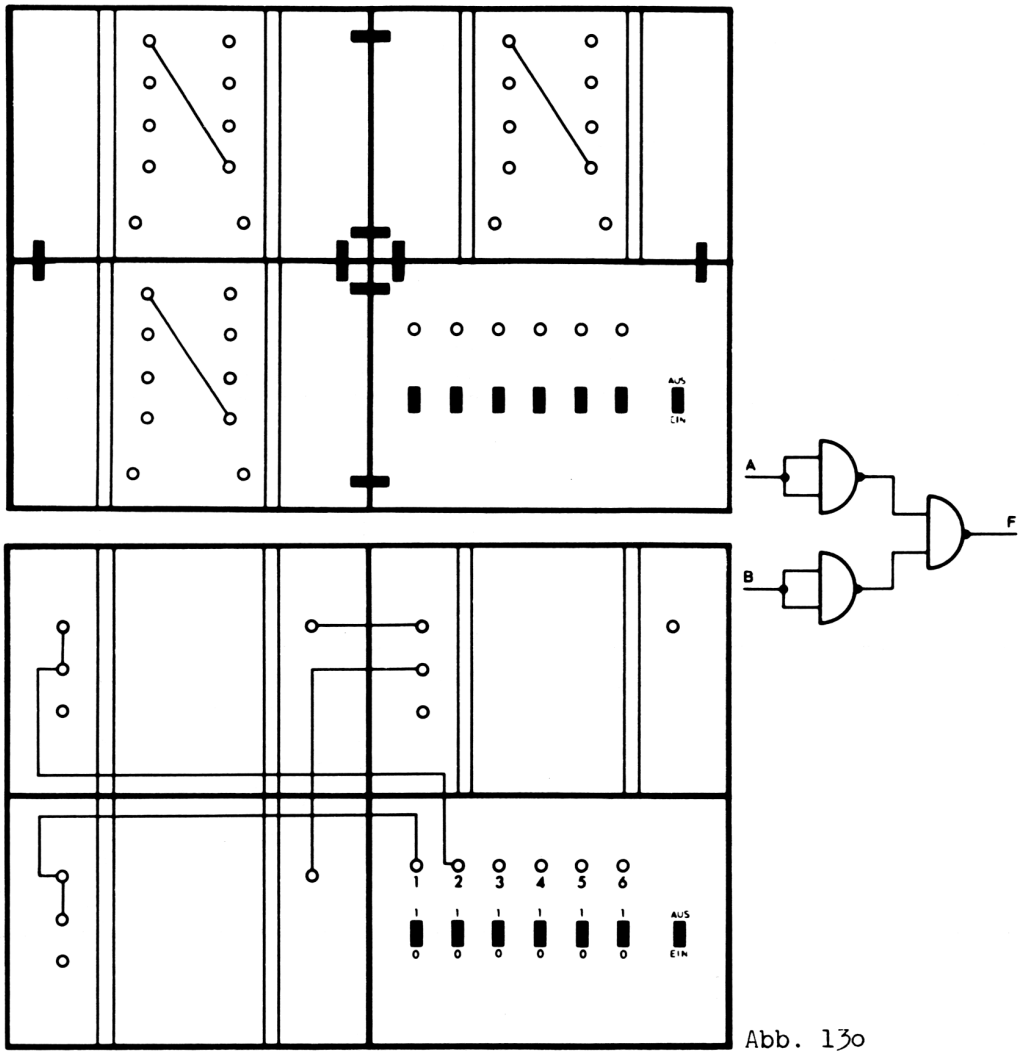


Abb. 13o

Um diese Schaltung zu verstehen, ist es am einfachsten, für jeden Logik-Baustein eine getrennte Funktionstabelle aufzustellen. Die Funktionstabellen für die Bausteine I und II entsprechen der für NICHT aus NAND: 0 am Schalter 1 ergibt am Eingang A des Bausteins III ein 1-Signal und an seinem Ausgang F = 0, wenn Schalter 2 ebenfalls auf 0 liegt. Führt dagegen mindestens ein Schalter ein 1-Signal, erhält der betreffende Eingang des Bausteins III 0 und F wird gemäß der NAND-Funktion = 1.

Die Funktionstabelle für die gesamte Schaltung entspricht also der einer ODER-Funktion.

15.4. NOR aus NAND

Um eine NOR-Schaltung aus NAND zu erhalten, kann von ODER aus NAND ausgegangen werden. Diese ODER-Schaltung muß dann noch zusätzlich negiert, d.h. mit einem NICHT kombiniert werden.

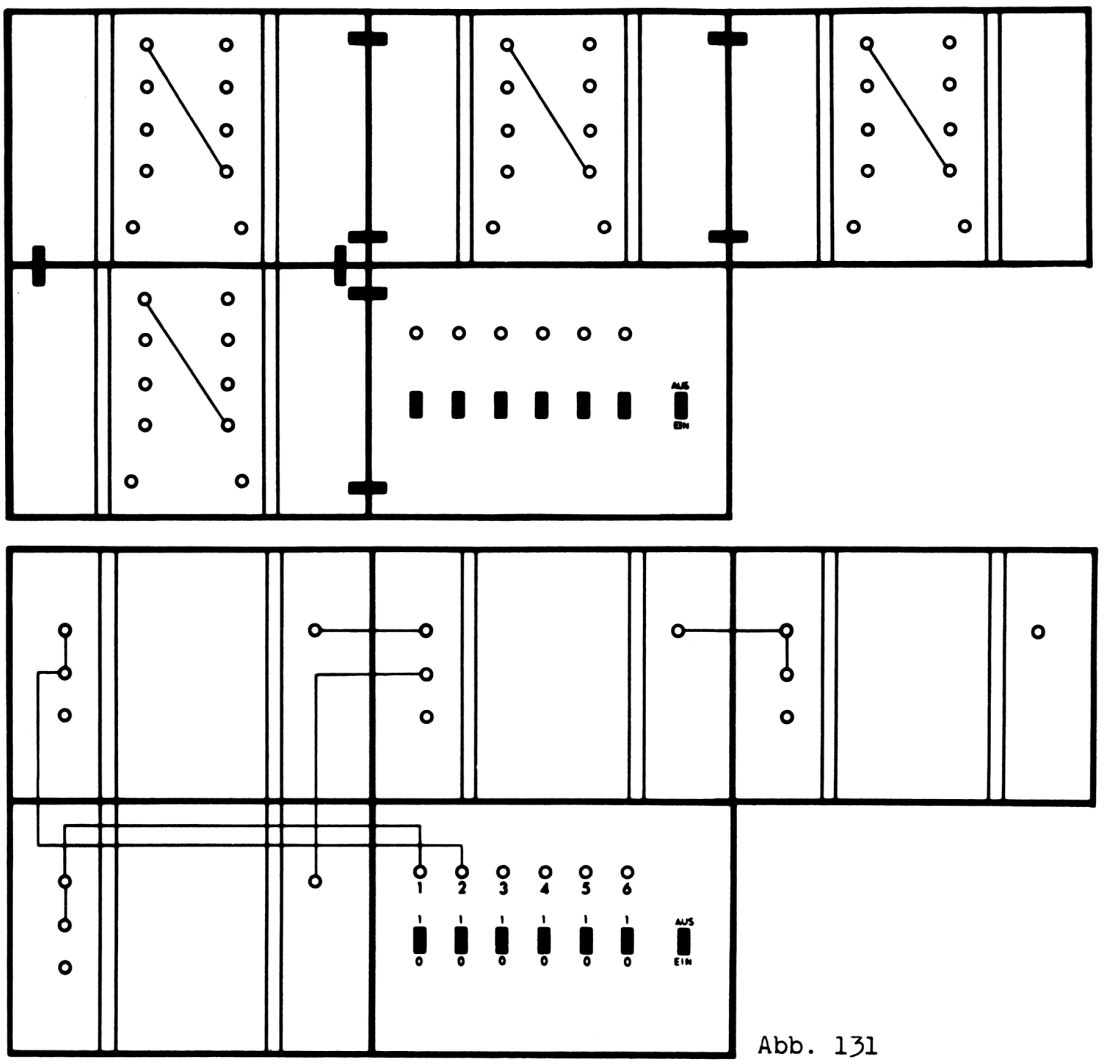
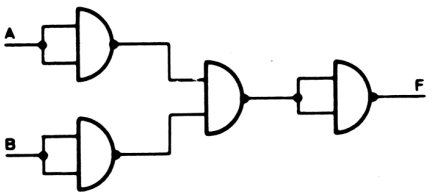


Abb. 131



Aufgabe: Das NOR aus NAND kann durch Wired AND vereinfacht werden.
Wie müssen die Bausteine verdrahtet sein?

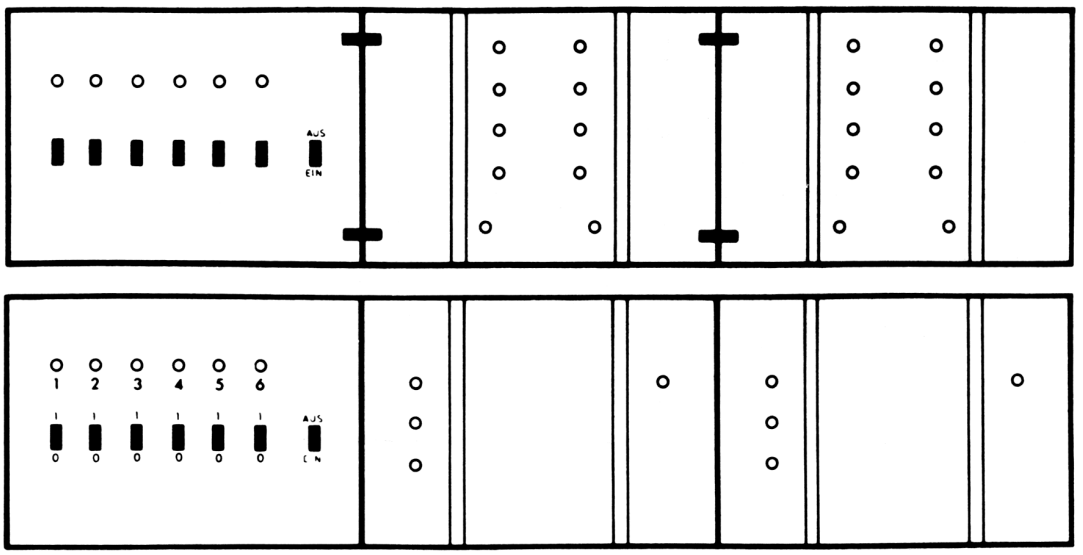


Abb. 132

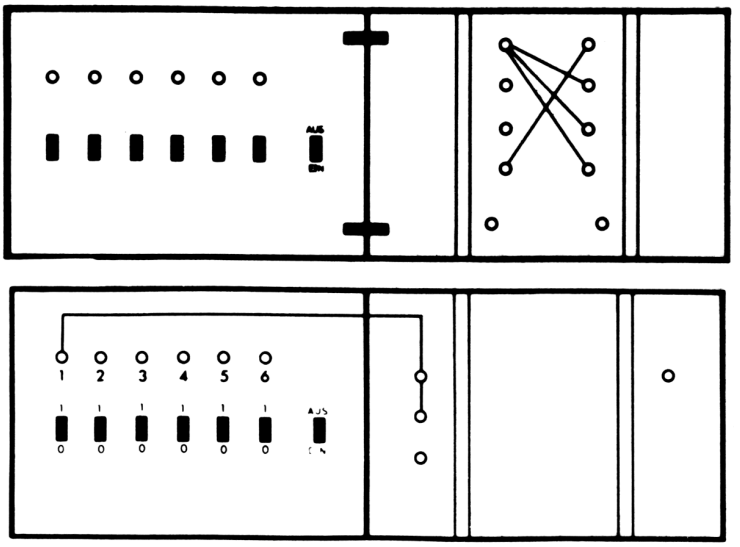
(Lösung am Ende des Kapitels)

Sie haben erfahren, daß aus NAND-Gattern diese vier Grundfunktionen aufgebaut werden können. In entsprechender Weise lassen sich alle anderen ebenfalls darstellen.

Ebenso wie das NAND-Gatter kann auch das NOR-Gatter dazu dienen, alle logischen Funktionen aus einer Grundschaltung zu verwirklichen. Dazu müssen alle Bausteine als NOR programmiert werden. - Die gewünschte Funktion erreichen Sie wieder durch eine entsprechende Verdrahtung.

15.5. NICHT aus NOR

Die NOR-Funktion ist eine Schaltung mit negiertem Ausgang der ODER-Funktion. Zur Realisierung von NICHT aus NOR benötigen Sie eine Eingabeeinheit und einen Logik-Baustein, der wie in Abb. 133 programmiert wird.



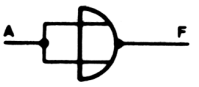


Abb. 133

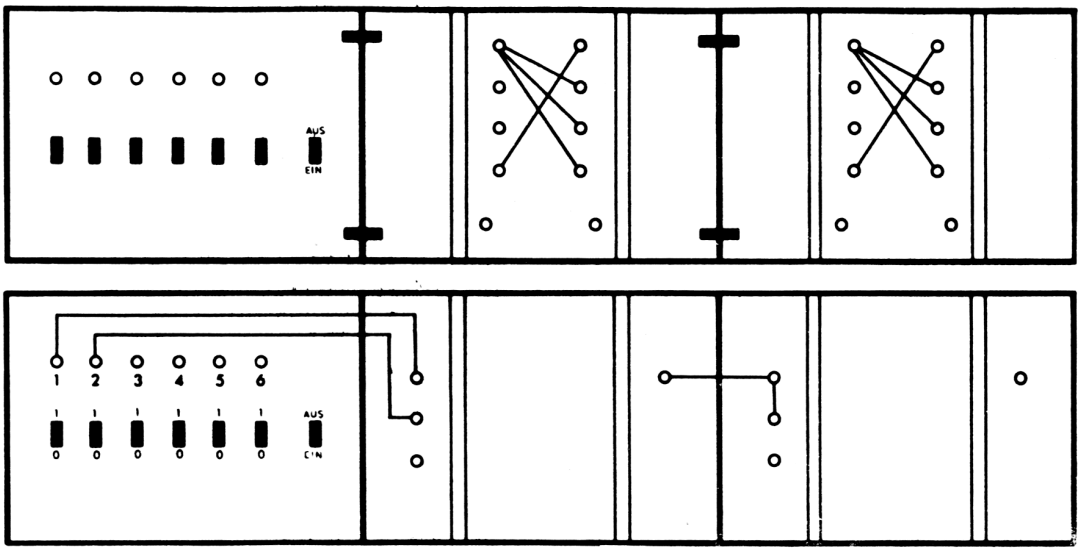
A1 A	B	F
0	0	1
0	1	0 +
1	0	0 +
1	1	0

Die beiden Eingänge des Logig-Bausteins werden wieder parallel geschaltet und mit einem Schalter der Eingabeeinheit verbunden. Dadurch können nur die Zustände A und B = 0 oder 1 auftreten. Die mit einem Sternchen bezeichneten entfallen. Das 0-Signal am Schalter der Eingabeeinheit bewirkt ein 1-Signal am Ausgang des Logik-Bausteins und umgekehrt.

15.6. ODER aus NOR

Verbindet man den Ausgang des ersten NOR-Bausteins mit den beiden Eingängen des zweiten, so erhalten diese beiden Eingänge nach der NOR-Funktionstabelle nur dann ein 1-Signal, wenn von den Schaltern A und B ein 0-Signal ausgeht. Durch die NOR-Programmierung des zweiten Logik-Bausteins wird dieses Signal negiert, und der Ausgang des zweiten Bausteins zeigt deshalb ein 0-Signal. Geht nur von einem der beiden Schalter ein 1-Signal aus, wird durch die Negierung am zweiten Logik-Baustein auch ein 1-Signal erzeugt. Das entspricht der ODER-Funktion.

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



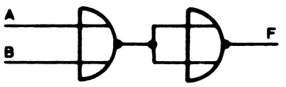
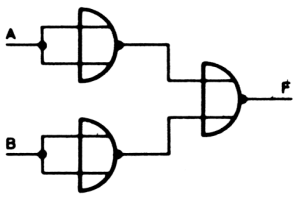


Abb. 134

15.7. UND aus NOR

Sicherlich bereitet es Ihnen keine Schwierigkeiten, das UND aus NOR anhand der Symbolschaltung zu erkennen. Hilfe bietet ein Vergleich mit der Schaltung NICHT aus NOR.



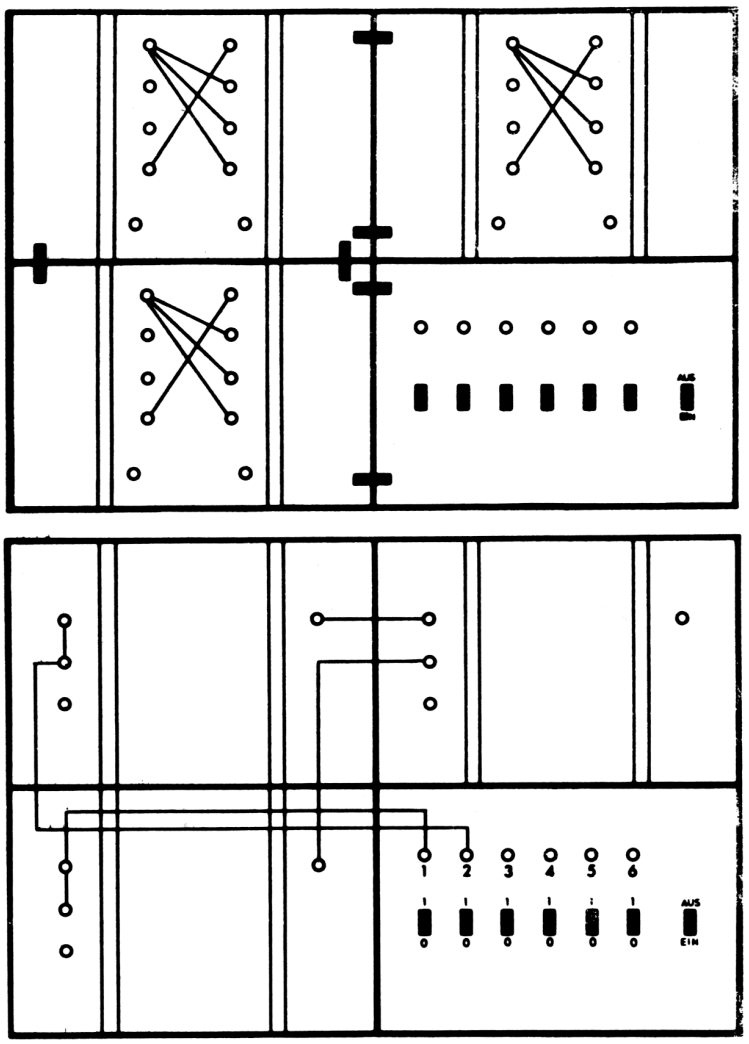


Abb. 135

15.8. NAND aus NOR

Das NAND aus NOR ist am leichtesten zu verstehen, wenn Sie den Ausgang des UND aus NOR nur noch einmal durch NICHT aus NOR negieren.

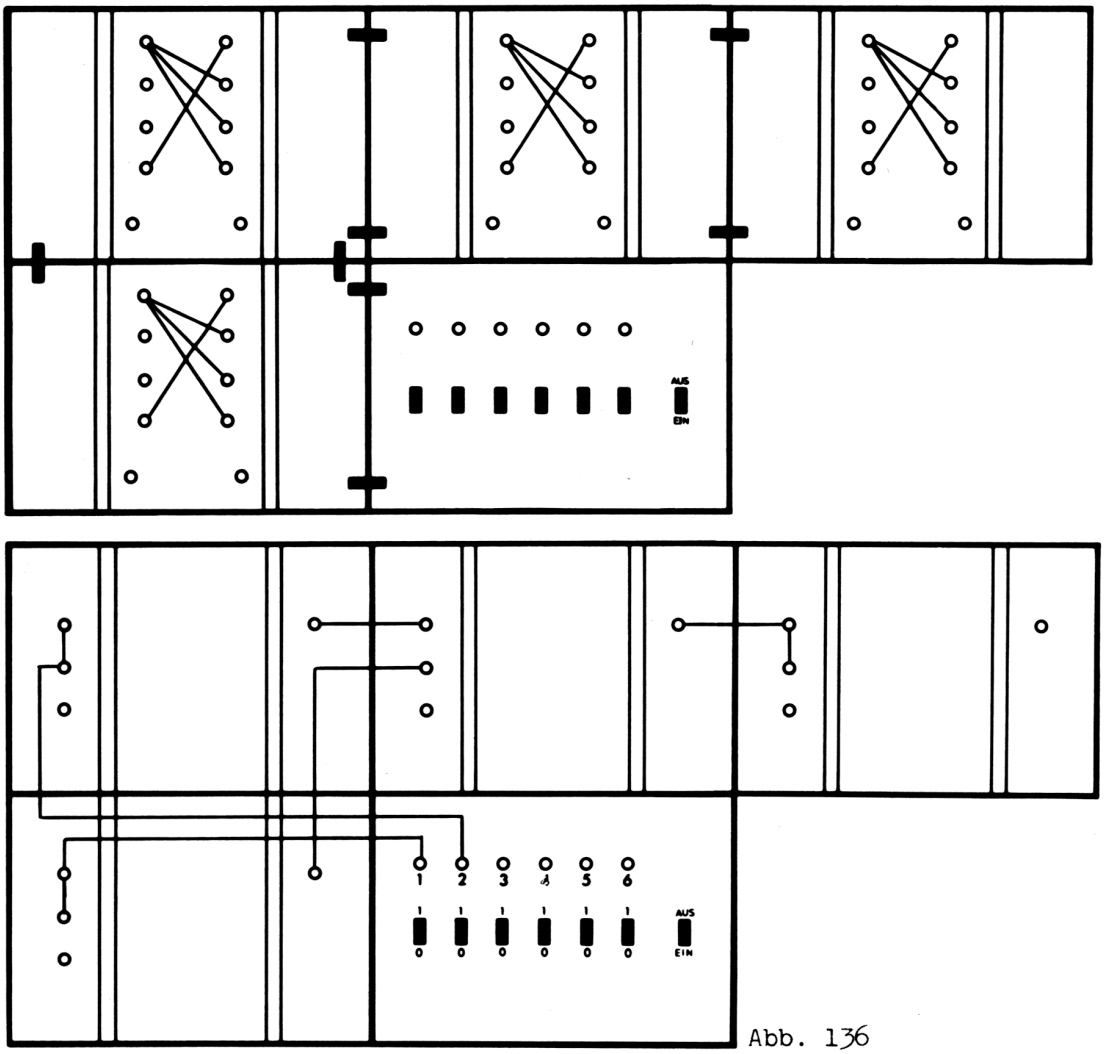
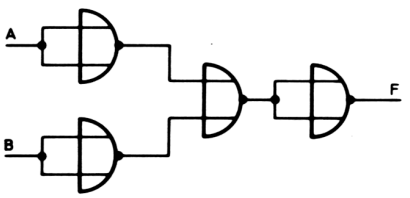


Abb. 136



Lösung zum Kapitel 15

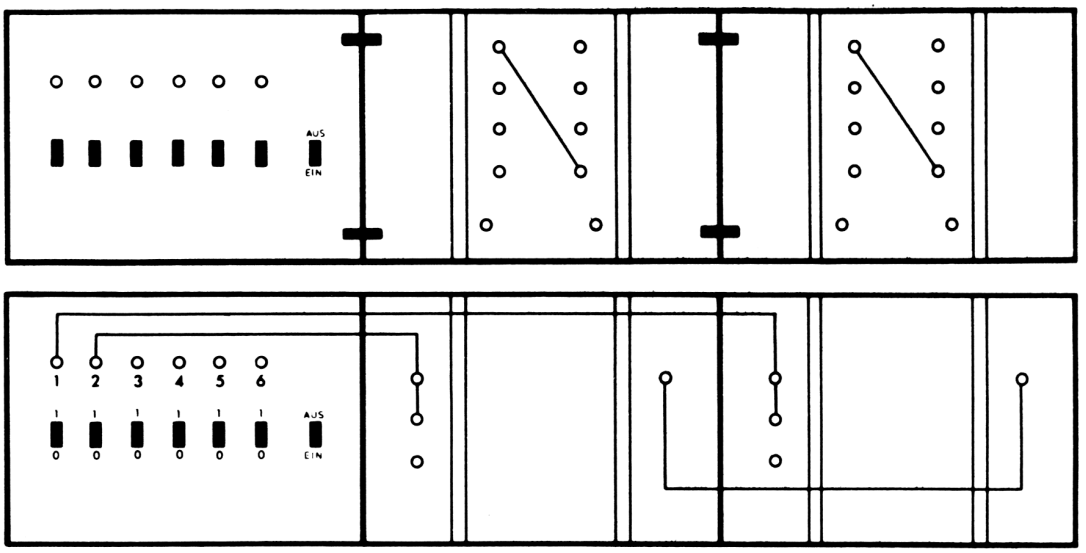


Abb. 137

16. Anwendungsmöglichkeiten für CL 1601/CL 1602/CL 1603

16.1. Prüfeinrichtung für Logik-Bausteine

In der industriellen Produktion gewinnt die automatische Prüfung und Kontrolle immer mehr an Bedeutung. Nicht nur, daß komplizierte Geräte mit geringer menschlicher Hilfe einer exakten und unbestechlichen Endkontrolle unterzogen werden, auch ein Dauertest ausgewählter Prüfexemplare wird überwiegend von Automaten ausgeführt. Einen solchen Dauerprüfautomat für 2 Logik-Bausteine stellt die folgende Schaltung dar. Durch Vergleich mit einem Normbaustein werden 2 Prüflinge kontrolliert. Für eine Funktionsprüfung eines Logik-Bausteins ist es notwendig, alle 3 Eingänge mit wechselnden Signalen zu belegen, um dann festzustellen, ob das Ausgangssignal entsprechend der Programmierung auftritt. Da es nun sehr aufwendig ist, die Eingangssignale durch Schalter über einen längeren Zeitraum einzugeben, sollen Logik-Bausteine das übernehmen. Dazu wird ein Generator mit einem Binäruntersetzer (vergl. Kap. 11, Seite 24 ff) aufgebaut. Der Normbaustein und die beiden Prüflinge können beliebig, müssen aber gleich programmiert werden. Sie erhalten gleichzeitig dieselben Eingangssignale und zeigen deshalb auch dieselben Ausgangssignale. Die Ausgänge der Prüflinge und des Normbausteins werden auf die Eingänge eines als Äquivalenz programmierten Bausteins gegeben. Dort muß F solange gleich 1 sein, wie alle Prüflinge dasselbe Ausgangssignal besitzen. Hat einer einen Defekt, so erhält der Äquivalenz-Baustein nicht mehr gleichartige Signale, und sofort ist F = 0.

Hier die Programmierung und Verdrahtung eines Beispiels:
Bei dieser Prüfschaltung zeigt der Äquivalenz-Baustein F = 1, da die Prüflinge mit dem Normbaustein in ihrer Funktion übereinstimmen, d.h. ohne Fehler waren.
Im Prüfling PI soll durch Änderung der Verdrahtung ein Fehler simuliert werden. Verbinden Sie dazu f2 mit 0.

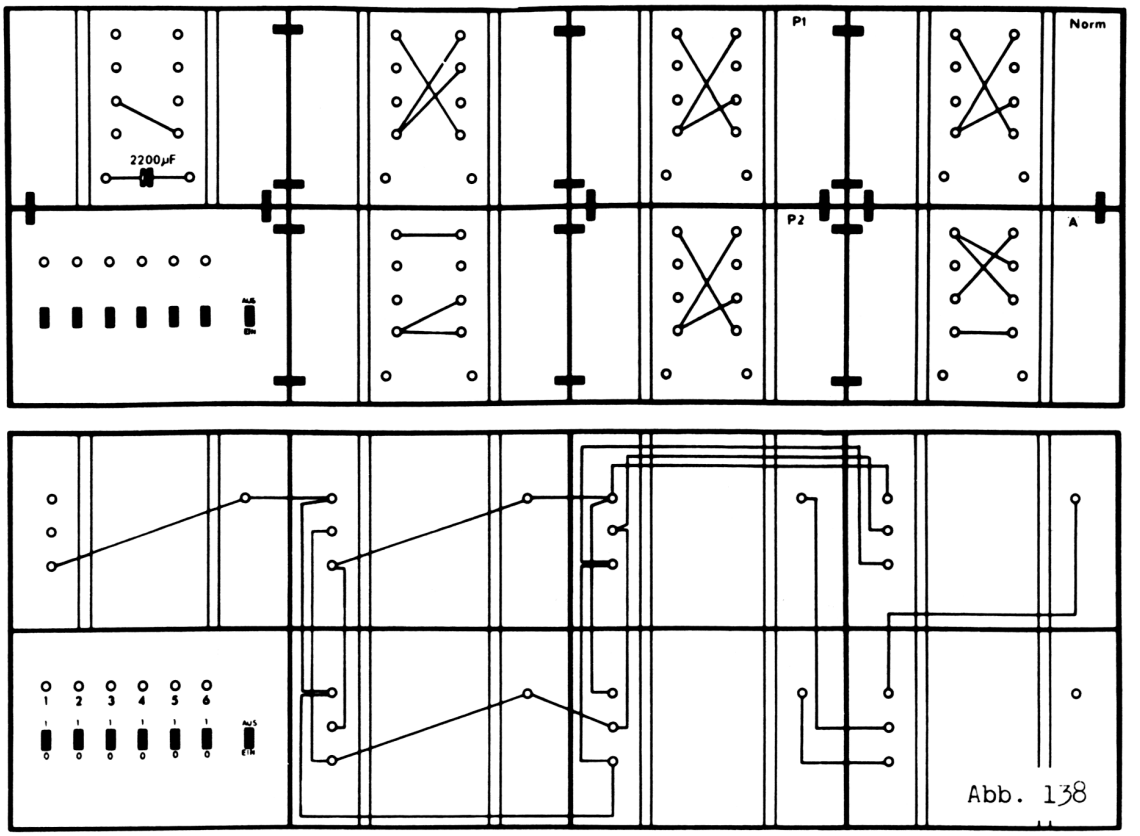


Abb. 138

Bei dem vorliegenden Aufbau wird der Fehler in dem Logik-Baustein PI nur kurzfristig durch ein optisches Signal am Äquivalenz-Baustein - Lampe erlischt - angezeigt, das um so kürzer ausfällt, je größer die Frequenz des Generators ist. Um diese Schwäche auszugleichen, soll anstelle des Äquivalenz-Bausteins einer eingesetzt werden, der ein Verhalten entsprechend der folgenden Funktionstabelle aufweist:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$f1 = C$

$f2 = 1$

$f3 = 1$

$f4 = \bar{C}$

Wenn Sie nun den Ausgang des so programmierten Logik-Bausteins mit dem Eingang B eines Speicher-Flipflops verbinden (vergl. Kap. 11.1), wird eine Unregelmäßigkeit an den Ausgängen der Prüflinge - was ja einem Fehler entspricht - gespeichert. Erst durch Betätigen des Schalters 1, der mit dem Eingang A des Speichers verbunden ist, kann die Fehleranzeige gelöscht werden.

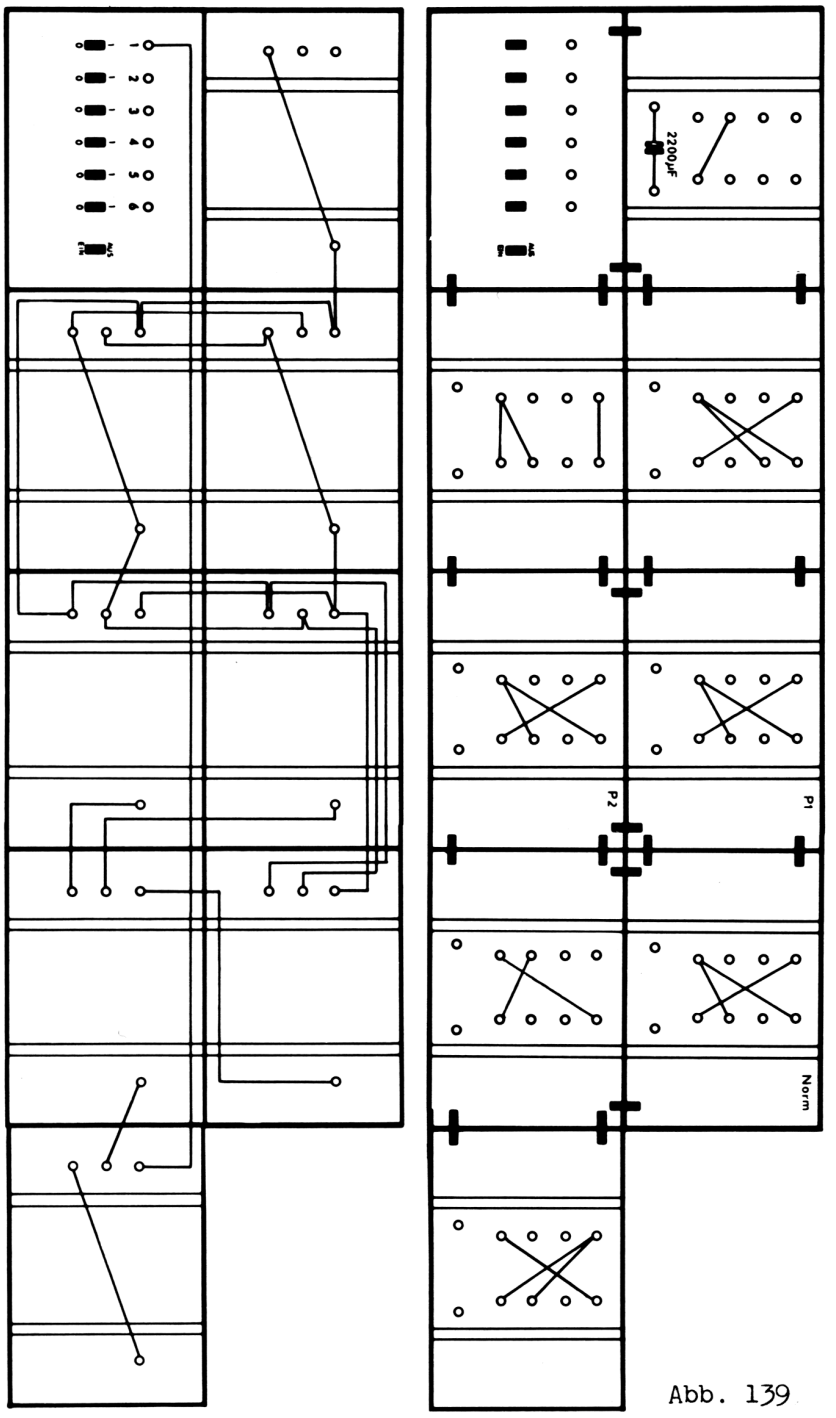


Abb. 139

Auch diese Prüfeinrichtung ist noch nicht vollkommen. Wenn Sie nämlich folgenden Fehler an PI einprogrammieren, werden Sie feststellen, daß er nicht angezeigt wird.

Programmierung PI:

f1 = 0
f2 = 1
f3 = C
f4 = 0

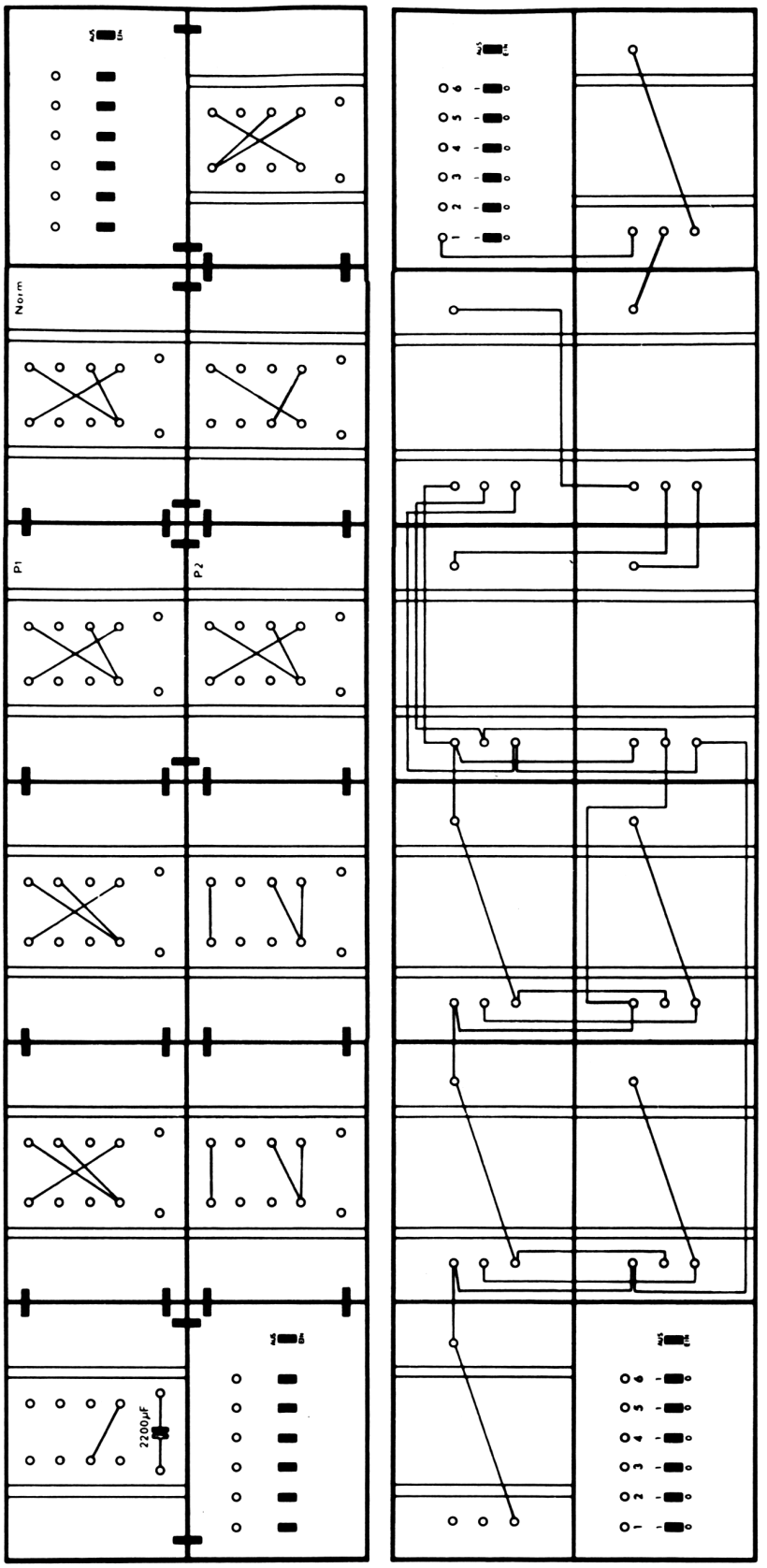


Abb. 14o

Durch den Generator mit einem Binäruntersetzer können nur 4 der 8 möglichen Signalkombinationen an den Eingängen der Prüflinge auftreten, und zwar:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Es fehlen:

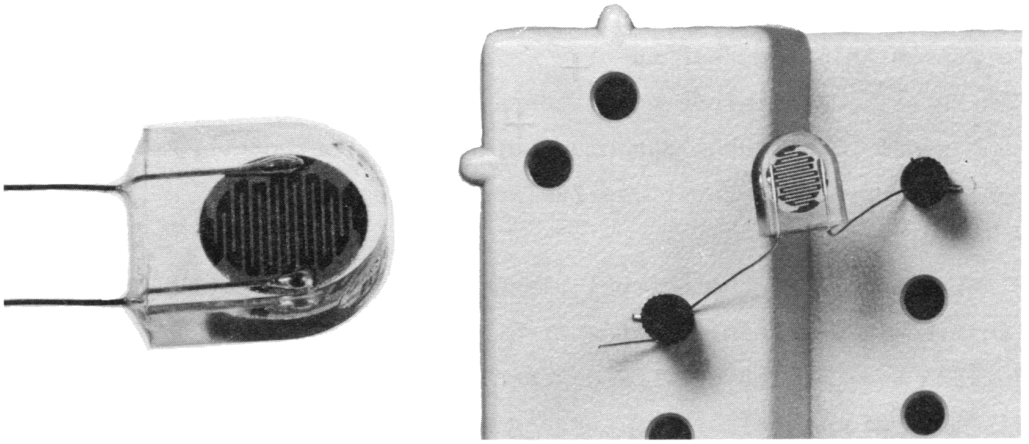
A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

Um auch die fehlenden Eingangskombinationen zu erzeugen, wird ein weiterer Binäruntersetzer benötigt (s. Abb. 14o).

Bei Schaltungen mit mehr als 8 Logik-Bausteinen sollte stets eine zweite Eingabeeinheit angeschlossen werden, um eine sichere Spannungsversorgung zu gewährleisten.

16.2. Dämmerungsanzeiger

Mit einem Logik-Baustein und einem LDR (light dependent resistor = lichtabhängiger Widerstand) läßt sich leicht ein Dämmerungsanzeiger aufbauen.



Ein LDR besitzt bei Dunkelheit einen relativ großen Widerstand ($> 10\ k\Omega$), der mit zunehmender Helligkeit immer kleiner wird. Schaltet man einen solchen LDR zwischen Eingang A und 0 des Programmierfeldes eines als Identität ausgelegten Logik-Bausteins, nimmt bei Dunkelheit der Ausgang F = 1 an. Denn durch den hohen Widerstand erhält Eingang A keine Verbindung mit "0". Das bewirkt an A eine "1", so, als ob A nicht belegt wäre. Und nicht belegte Eingänge nehmen automatisch den Zustand 1 an.
Fällt dagegen Licht auf die gestreifte Seite des LDR, wird der Widerstand so gering, bis schließlich A = 0 wird. Dann ist auch F = 0.

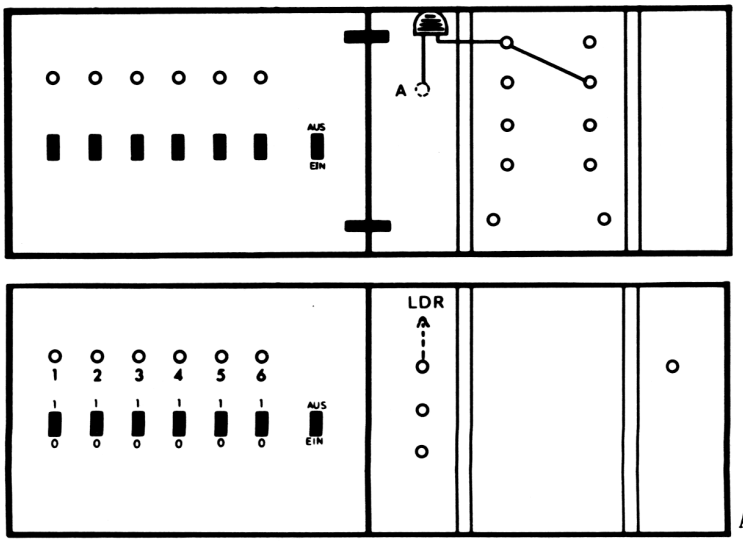


Abb. 141

16.3. Lichtschranken-Zähler

Bei automatischen Fertigungsanlagen ist es notwendig, produzierte Gegenstände beim Verlassen des Bandes zu zählen. Dazu werden heute Zähleinrichtungen verwendet, bei denen Menschen nur noch die Stückzahlen ablesen müssen. Einen solchen Zähler können Sie nach der folgenden Anleitung aufbauen.
Das Prinzip des Zählers beruht darauf, daß die Produkte auf dem Band einen Lichtstrahl durchlaufen, der dadurch unterbrochen wird. Jede Unterbrechung setzt das Zählwerk in Tätigkeit.

Die Lichtschranke besteht bei diesem Beispiel aus dem LDR und einem Logik-Baustein. Solange Licht auf den LDR fällt, ist der Ausgang F = 0. Wird der LDR dagegen abgeschirmt, erscheint durch die Inversionsschaltung am Ausgang ein 1-Signal. Es wird auf die 1. Stufe des Zählwerks geleitet, das aus Binärzählern besteht (vergl. Kap. 11.3). Diese erste Stufe ist die 3. Stelle des Zählwerks. Bei jeder weiteren Unterbrechung schaltet der Zähler um eins weiter, bis schließlich 1 - 1 - 1 erreicht ist. Das bedeutet dezimal 7. Dann beginnt das Zählwerk wieder bei 0 - 0 - 0. Durch weitere Binärzähler kann das Zählwerk beliebig vergrößert werden.

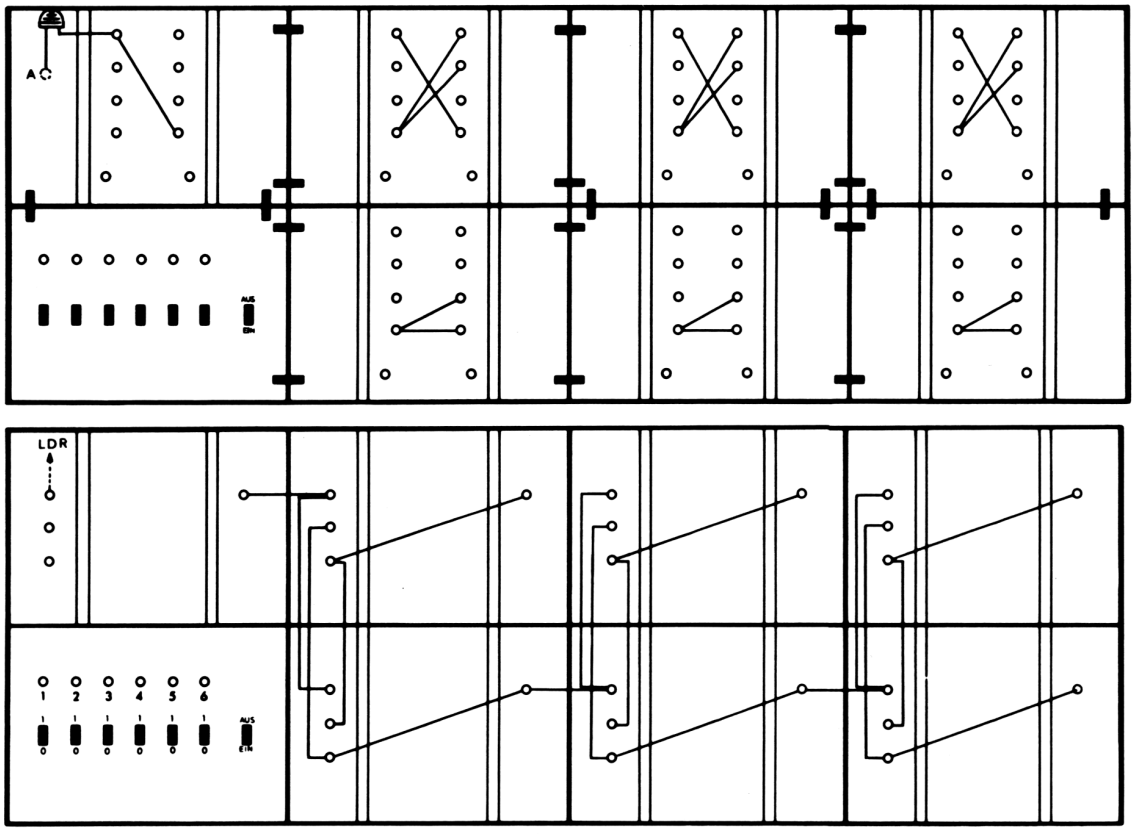
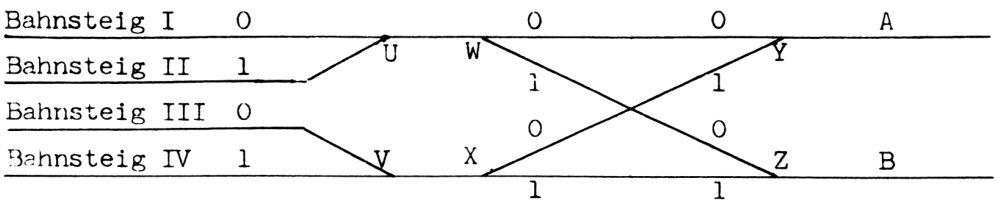


Abb. 142

16.4. Kopfbahnhof mit 4 Gleisen

Von den vier Bahnsteiggleisen (I, II, III, IV) eines Kopfbahnhofs besteht über 6 Weichen eine Verbindung zu zwei Hauptgleisen (A und B). Die Bedingung, daß jedes der beiden Hauptgleise A und B von den 4 Bahnsteiggleisen erreicht werden kann, wird durch entsprechende Weichenstellung realisiert.



Der Computer hat die Aufgabe, für die Ausfahrt von den Bahnsteiggleisen "Freie Fahrt" anzuzeigen, wenn durch die Weichenstellung eine durchgehende Verbindung zu den Hauptgleisen besteht.
Für I und II hängt "Freie Fahrt" von der Stellung 0 oder 1 der Weichen U, W, Y, Z ab, für III und IV von der Stellung der Weichen V, X, Y, Z.
Da die Logik-Bausteine nur 3 Eingänge besitzen, müssen die Durchfahrten W, Y, Z bzw. X, Y, Z und die Ausfahrten U bzw. V gesondert betrachtet werden.
Aus der Funktionstabelle für W, Y, Z können Sie die Bedingungen für F1 = 1 (Durchfahrt) ersehen.

Analog gilt für X, Y, Z:

A	B	C	F
W	Y	Z	F1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

A	B	C	F
X	Y	Z	F2
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Im folgenden muß eine Verknüpfung zwischen U und F1 bzw. V und F2 hergestellt werden.
Da die Ausfahrt von I und II durch die Weiche U geregelt wird, ergibt sich für Gleis I nur eine Ausfahrtmöglichkeit bei der Kombination U = 0 und F1 = 1. Die Funktionstabelle für I drückt das aus:

A	B	F
U	F1	I
0	0	0
0	1	1
1	0	0
1	1	0

Entsprechend gilt für Gleis II:

A	B	F
U	F1	II
0	0	0
0	1	0
1	0	0
1	1	1

Über die Weiche V ergibt sich die Ausfahrtmöglichkeit für Gleis III und IV bei F2 = 1.
Bestimmen Sie, bei welcher Bedingung für Gleis III bzw. IV die Ausfahrt freigegeben werden kann.

A	B	F
V	F2	III
0	0	
0	1	
1	0	
1	1	

A	B	F
V	F2	IV
0	0	
0	1	
1	0	
1	1	

Falls es Ihnen schwerfällt: Die Schaltung liefert das richtige Ergebnis, und Sie können die Werte für F an den Ausgängen III und IV ablesen.

Für den Aufbau der gesamten Steuerungsanlage benötigen Sie eine Eingabeeinheit und sechs Logik-Bausteine. Durch die Eingabeschalter 1 - 6 wird die Stellung der Weichen U - Z (0 oder 1) simuliert. Aus der Schaltung können Sie ersehen, daß Schalter 1 und 2 (Weiche U und V) direkt mit den Eingängen der Bausteine I/II bzw. III/IV verbunden sind, die die Anzeige "Freie Fahrt" für jedes Gleis übernehmen. Schalter 3 - 6 stellen die Weichen W, X, Y und Z dar und erzeugen entsprechend der Funktionstabelle die Ausgangssignale F1 und F2. Die Ausgänge von F1 und F2 werden als zweite Eingangsvariable auf die Anzeige-Bausteine I und II bzw. III und IV gegeben.

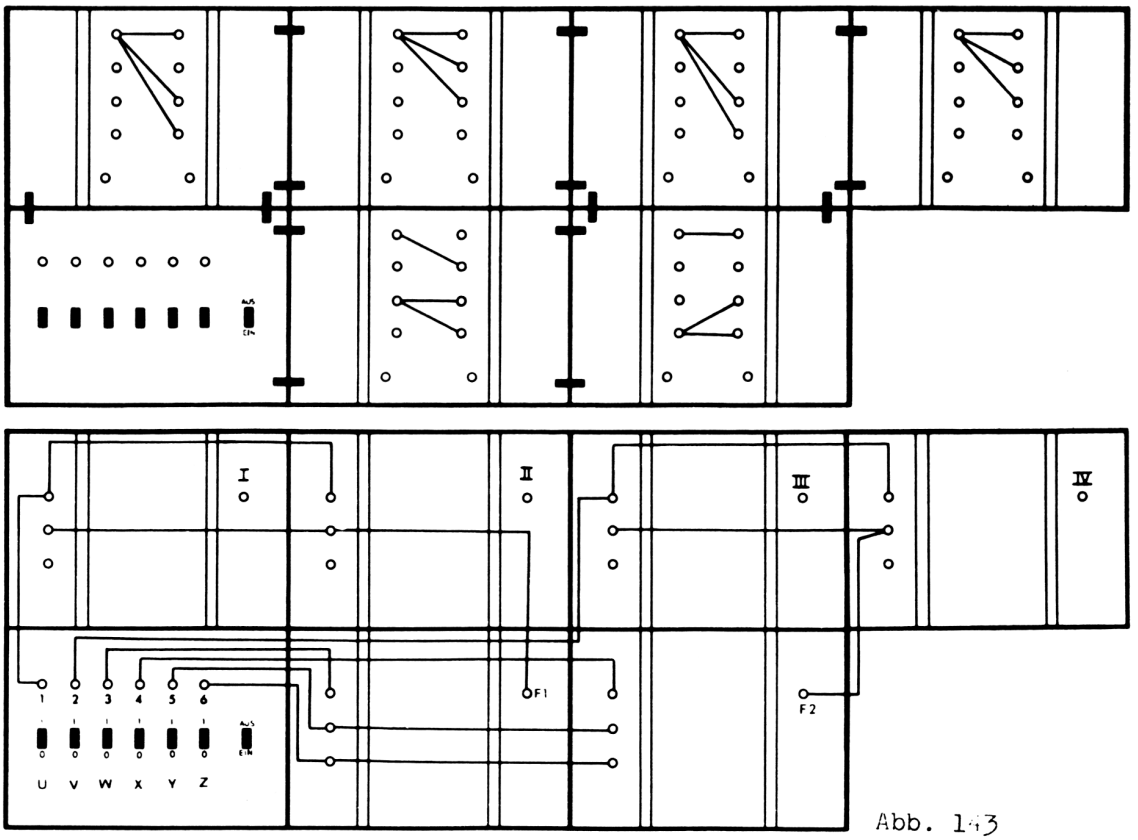


Abb. 143

Bei der Überprüfung der Schaltung werden Sie erkennen, daß tatsächlich nur dann für eines der Bahnsteiggleise I - IV "Freie Fahrt" auf dem entsprechenden Baustein angezeigt wird, wenn eine durchgehende Verbingung zu einem der Hauptgleise A oder B besteht. Allerdings kann es bei dieser Schaltung beim Kreuzen der Züge von W nach Z und X nach Y zu einem Zusammenstoß kommen, wenn folgende Weichenkombinationen geschaltet werden:

W	= 1	Z	= 0
X	= 0	Y	= 1

Um zu vermeiden, daß im Kreuzungsbereich Komplikationen entstehen können, soll ein weiterer Baustein so in die Schaltung eingebaut werden, daß kein Gleis ein Ausfahrtsignal erhält, wenn W = 1 und X = 0 auftreten. Dafür erhalten Sie die Funktionstabelle:

W	X	Fx
0	0	0
0	1	0
1	0	1
1	1	0

Dieser Ausgang Fx wird auf die C-Eingänge der 4 Anzeigebausteine gelegt. Für diese müssen nun neue Funktionstabellen aufgestellt werden, die aussagen, daß alle Gleise gesperrt werden (F = 0), wenn Fx = C = 1 auftritt.

Für Gleis I gilt:

A	B	C	F
U	Fl	Fx	F I
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	0
1	1	1	0

$f1 = 0$

$f2 = \bar{C}$

$f3 = 0$

$f4 = 0$

In der ersten Schaltung erhielt Gleis I "Freie Fahrt" bei U = 0 und Fl = 1. Diese Kombination tritt jetzt zweimal auf, nämlich in Zeile 3 und 4. Aber nur in Zeile 3 ist Fx = 0, und darum ist auch nur dann F = 1.

Hier nun die Tabellen für die Gleise II - IV:

A	B	C	F
U	Fl	Fx	F II
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

A	B	C	F
V	F2	Fx	F III
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	0
1	1	1	0

A	B	C	F
V	F2	Fx	F IV
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Die geänderte Programmierung und Verdrahtung können Sie der folgenden Abbildung entnehmen:

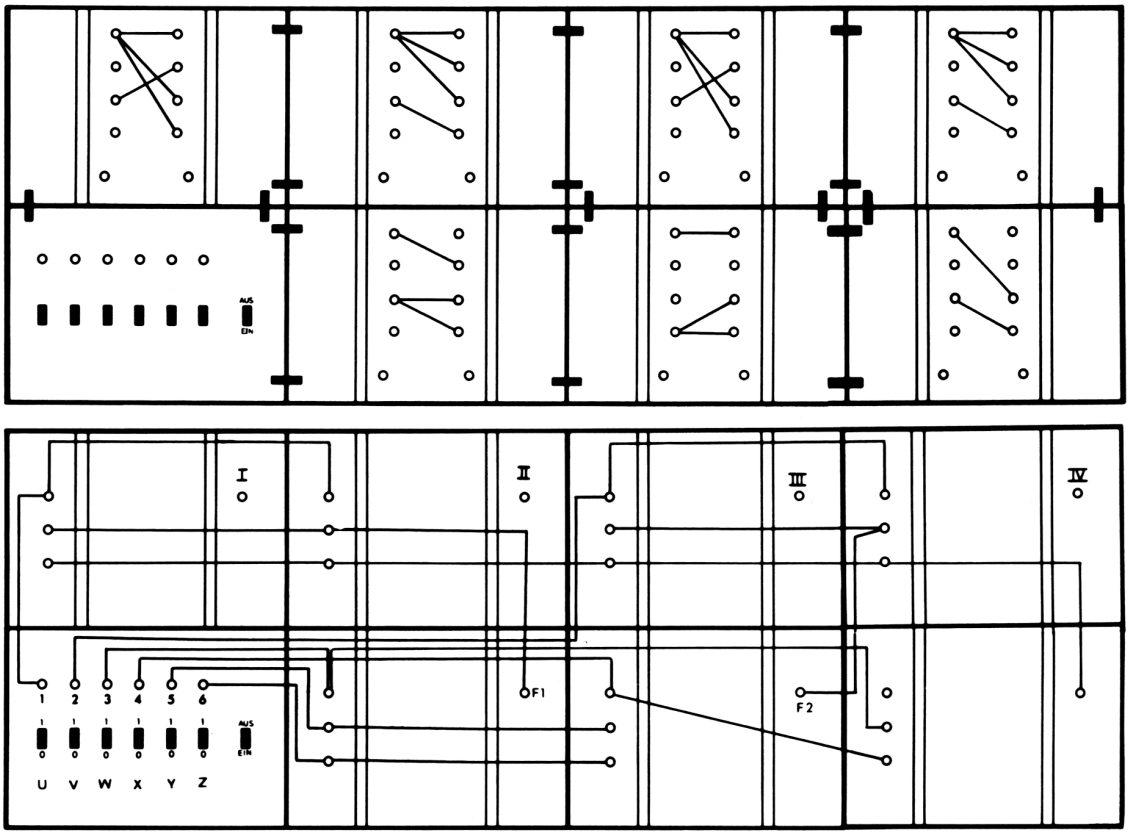
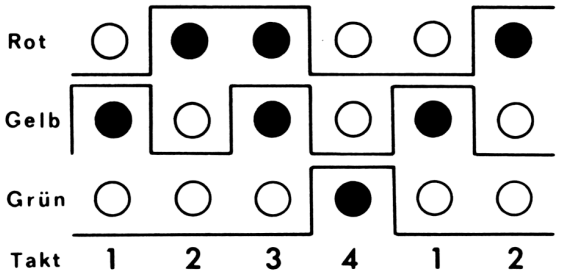


Abb. 144

16.5. Verkehrsampel

Die Lichtfolge an einer Verkehrsampel soll automatisch ablaufen. Dabei sollen die drei Lampen Rot, Gelb, Grün nach folgendem Schema ein- und ausgeschaltet werden:



Ein Generator mit zwei parallel geschalteten Kondensatoren (2.2oo µF und 47o µF) erzeugt ca. alle 3 Sekunden einen Impuls. Dieser Impuls wird über einen Binärzähler im Verhältnis 1 : 2 untersetzt und schaltet so direkt das rote Licht ein - das Ausgangssignal des Generators zeigt inzwischen wieder 0 (vergl. Schema - Takt 2).

Beim nächsten 1-Signal des Generators (Gelb) steht durch die Untersetzung das Ausgangssignal des Binärzählers (Rot) ebenfalls auf 1 (Rot - Gelb).

Gehen beide Ausgangssignale auf 0, also nicht gelb und nicht rot, wird das grüne Licht eingeschaltet (Takt 4). Die Programmierung für den Logik-Baustein, auf dem die Anzeige für Grün erfolgen soll, erhält man nach folgender Funktionstabelle:

A	B	F
Rot	Gelb	Grün
0	0	1
0	1	0
1	0	0
1	1	0

Mit vier Logik-Bausteinen und einer Eingabeeinheit als Stromversorgungsteil können Sie nach Abb.145 die Schaltung für die automatische Ampelanlage aufbauen.

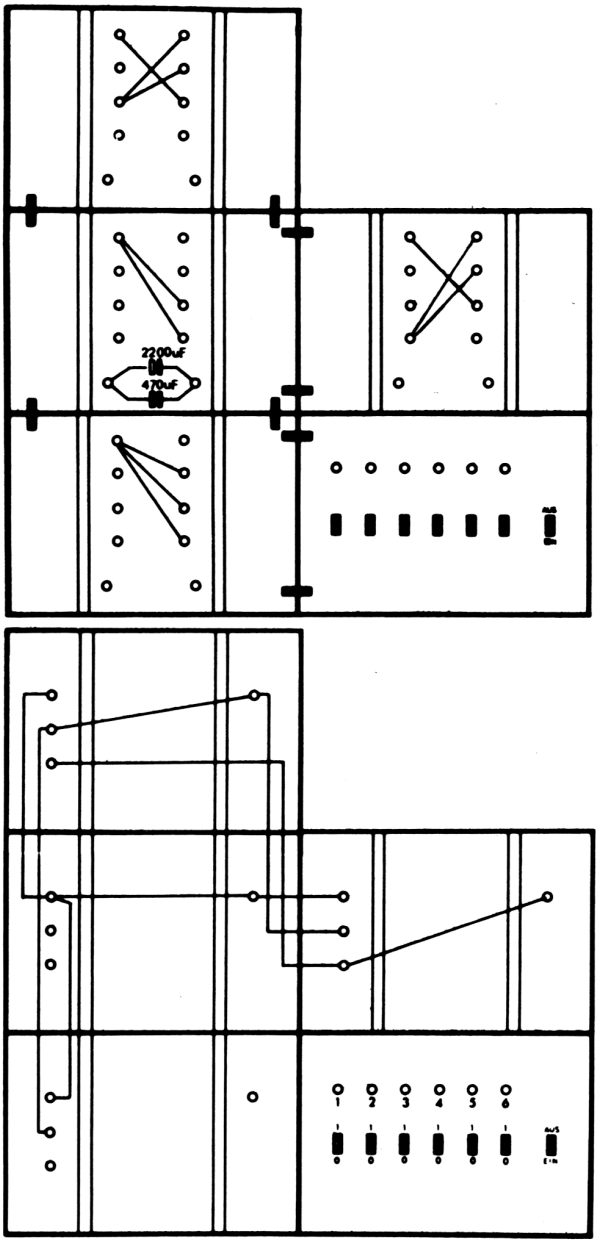


Abb. 145

16.6. Verkehrsampel mit Stop

An Baustellen ist es oft notwendig, Verkehrsampeln zu installieren, die eine den Verkehrsverhältnissen angepaßte Grün- bzw. Rotpause haben.

Die Schaltung für diese Anlage entspricht im Grundaufbau der automatischen Verkehrsampel. Am Generator-Baustein, der eine Programmierung nach Abb.146 erhält, erfolgt die Rückkopplung jedoch auf den Eingang C. Die Eingänge A und B werden zusätzlich in Parallelschaltung mit einem Schalter der Eingabeeinheit verbunden.

Bei Betätigung des Schalters erhalten die Eingänge A und B des Generators ein 1-Signal, der dann erst gestartet wird. Die Funktionstabelle für den Generator muß deshalb folgendermaßen aussehen:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Bei Rückkopplung C = F schwingt der Generator nicht; er zeigt am Ausgang konstant 0. Erst wenn durch den Schalter die Eingänge A, B ein 1-Signal erhalten, wird F = 1. Aufgrund von F = 1 wird C = 1. Verzögert um die Impulsversetzungszeit wird dann F = 0, dadurch auch C wieder = 0 usw. Die Anlage arbeitet jetzt im gleichen Taktschema wie im vorigen Beispiel.

Wird der Schalter bei F = 1 am Generator (Gelb) in Nullstellung gebracht, wird die Schwingung unterbrochen, und F nimmt den Wert 0 an. Das 1-Signal wird aber über den Frequenzuntersetzer noch weitergegeben, so daß das rote Licht noch eingeschaltet wird. In dieser Stellung stoppt die Anlage und zeigt konstant Rot.

Wird umgekehrt die Schwingung des Generators bei Stellung Rot - Gelb unterbrochen, stoppt die Anlage bei Grün.

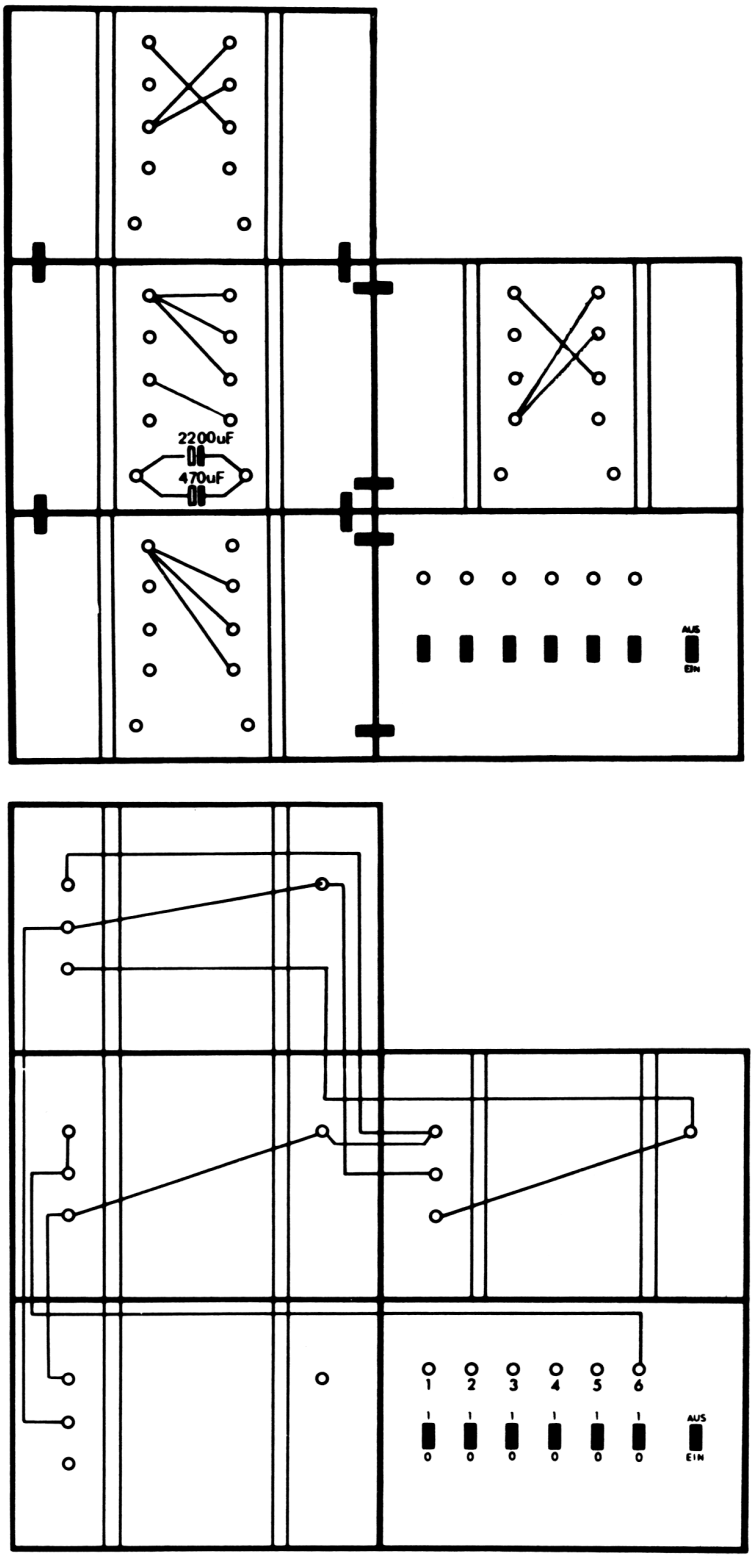


Abb. 146

16.7. Automatische Anzeige mit Stop

Bei den bisherigen Beispielen haben Sie ein bestimmtes Problem in eine logische Funktion übergeführt und aus der Funktionstabelle die Programmierung für den Logik-Baustein abgeleitet. Die Signalkombination, bei der F den Wert 1 annimmt, läßt sich durch entsprechendes Schalten auf der Eingabeeinheit "abfragen". Dieses "Abfragen" kann der Computer auch selbst übernehmen, und zwar viel schneller.

Dazu muß ein Generator die Signale auf einen Binärzähler geben - in unserem Beispiel auf drei hintereinandergeschaltete Zählwerke. Die Ausgangssignale der Binärzähler dienen als Eingangsvariable für einen Logik-Baustein als Anzeige. Wählen wir für die Anzeige die Signalkombination 1 - 0 - 1, so ergibt sich für den Logik-Baustein eine Programmierung nach folgender Funktionstabelle:

A	B	C	F
F1	F2	F3	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Der Generator übernimmt nun mit Hilfe des Zählwerks selbsttätig das Abfragen und zeigt auf dem Ausgang des entsprechend programmierten Anzeige-Bausteins bei dieser Signalkombination F = 1 an. Da der Generator so schnell abfragt, daß die Anzeige F = 1 nur für Bruchteile von Sekunden erscheint, muß der Generator gestoppt werden, damit die Signalkombination abgelesen werden kann. Dazu wird das 1-Signal auf einen zusätzlichen Logik-Baustein gegeben, der als Torschaltung zwischen dem Generator und dem Eingang des ersten Binär-Zählers geschaltet ist. Wird beim Abfragen die vorgegebene Signalkombination erreicht, bewirkt das 1-Signal am Eingang A der Torschaltung eine 1 am Ausgang. Die Ausgangsimpulse des Generators sind dadurch wirkungslos und können nur wieder durchgelassen werden, wenn der Schalter 1 der Eingabeeinheit kurzzeitig auf 1 gelegt wird.

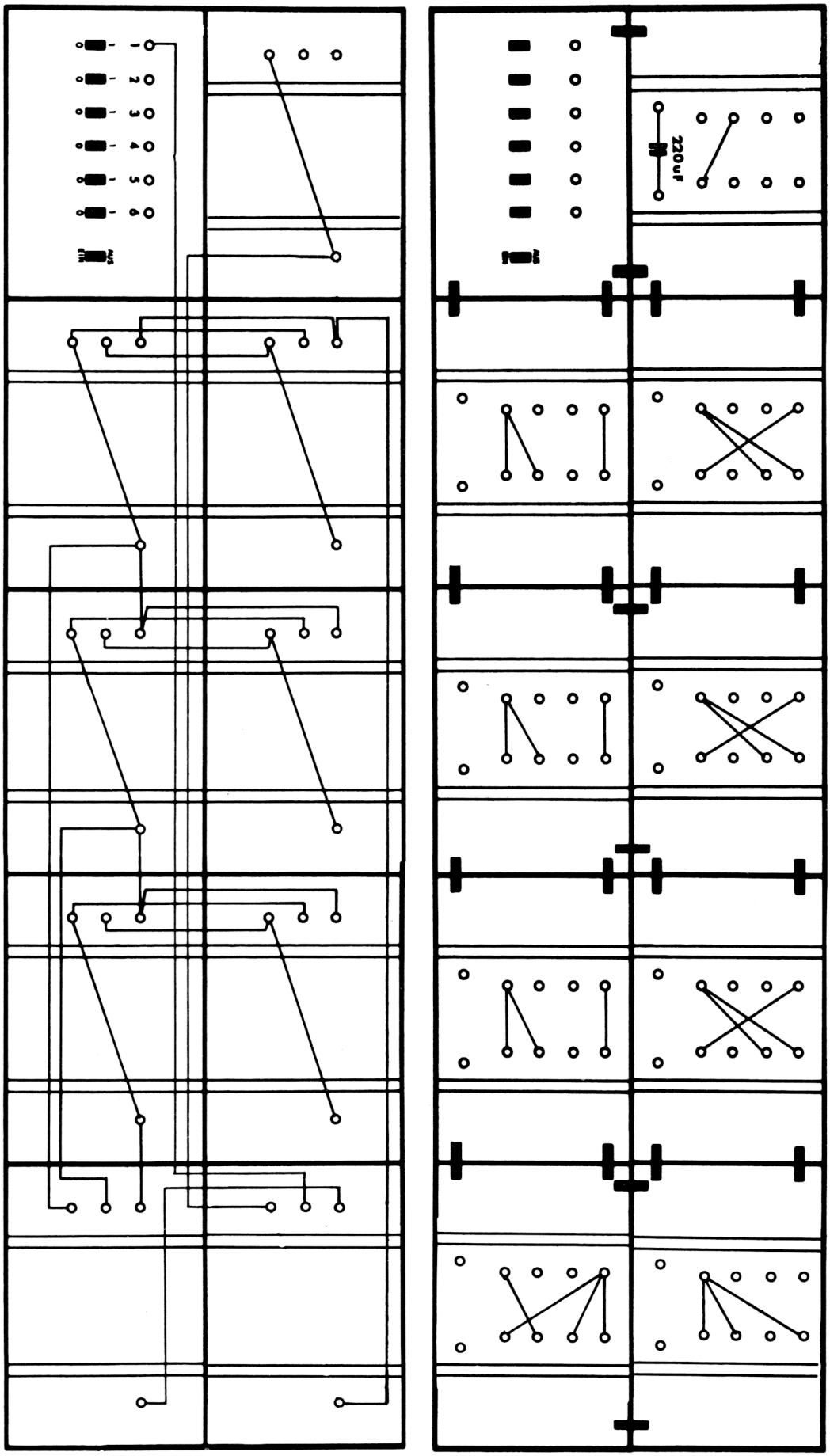
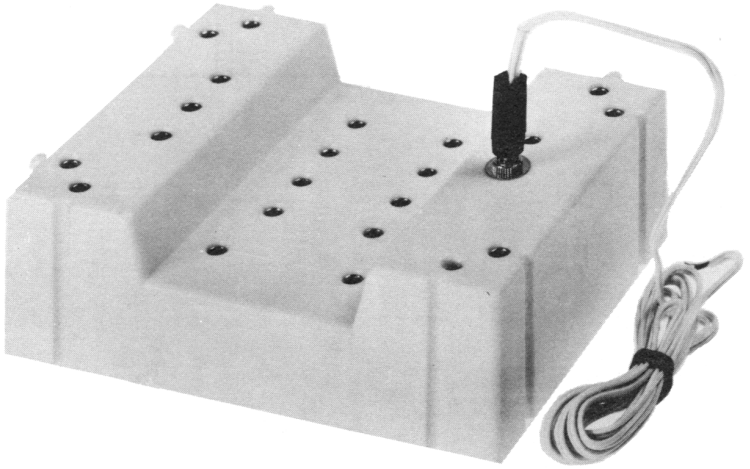


Abb. 147

17. Relais-Baustein (CL 1604)

Äußerlich unterscheidet sich der Relais-Baustein vom Logik-Baustein durch die Farbe, die fehlende Anzeigelampe und eine zusätzliche Spezialbuchse. Die Elektronik im Innern dagegen ist mit dem Logik-Baustein fast identisch. Wenn Sie die beiden Schaltbilder einmal miteinander vergleichen, fällt auf, daß die Anzeigelampe durch ein Relais ersetzt ist, zu dem parallel die Diode D3 liegt. Ein Relais ist ein elektromagnetischer Schalter, der mit einem geringen Steuerstrom einen großen Arbeitsstrom schaltet. Die beiden Anschlüsse dieses Schalters werden über die Spezialbuchse und den dazugehörigen Stecker mit Kabel herausgeführt.



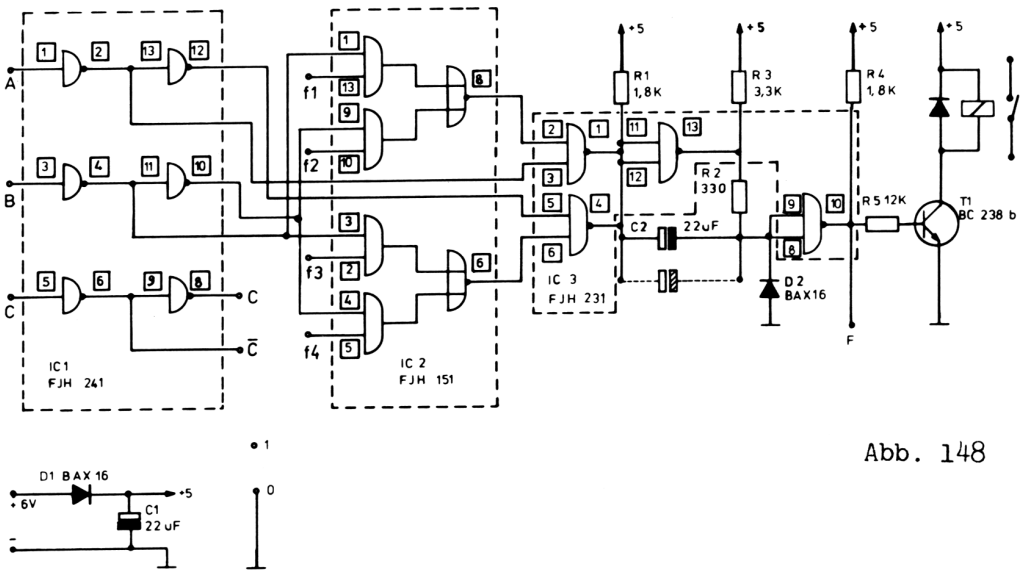


Abb. 148

17.1. Identität mit dem Relais-Baustein

Sie sollen hier nicht weiter mit theoretischen Einzelheiten belastet werden, sondern diesen neuen Baustein gleich einmal ausprobieren.

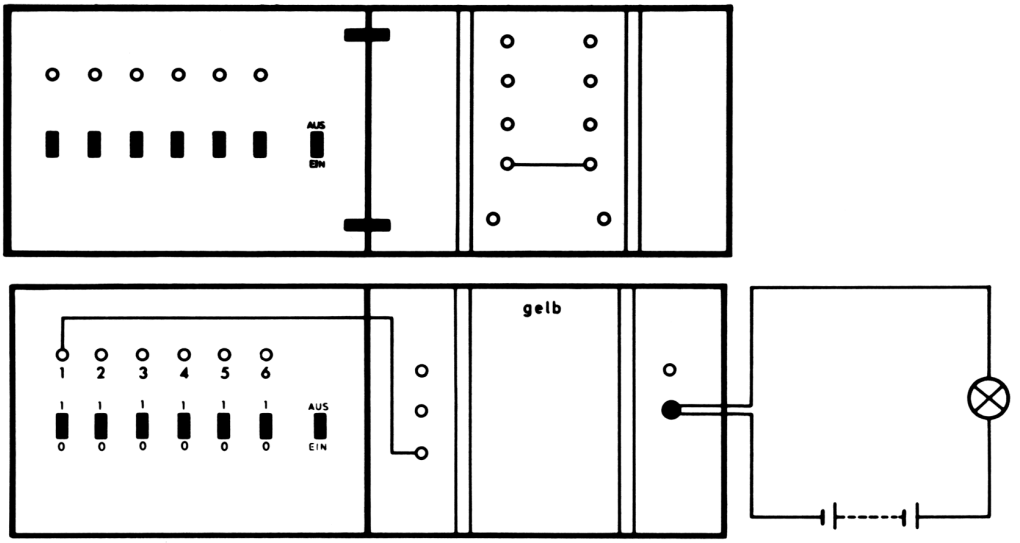
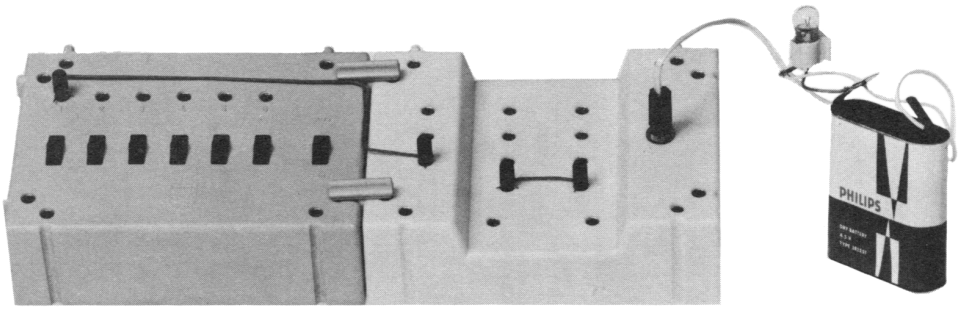


Abb. 149



Sie erkennen, daß das Programmierfeld wie bei einer Identitätsfunktion eines Logik-Bausteins belegt und entsprechend auch der Eingang C mit dem Schalter 1 verbunden wurde. Soweit ist Ihnen alles vertraut. Neu ist nun, daß die eine Ader des Kabels mit dem Minuspol einer Stromquelle (z. B. Flachbatterie) verbunden ist. Vom Pluspol führt eine Leitung zu einer Glühlampe mit Fassung und vom anderen Anschluß der Fassung über die zweite Ader in den Baustein.

Betätigen Sie nun den Schalter 1, liegt am Eingang C ein 1-Signal, und gleichzeitig leuchtet die externe Lampe auf. Dabei vernehmen Sie ein Klicken, das vom Magneten im Relais herrührt. In diesem Moment wird der Arbeitsstromkreis eingeschaltet. Führt der Eingang C anschließend ein 0-Signal, weil der Schalter wieder auf 0 gestellt wurd, erlischt mit einem Klicken die Lampe.

17.2. Negation mit dem Relais-Baustein

Ändern Sie, bitte, die Programmierung des Relais-Bausteins zur Negation ab.

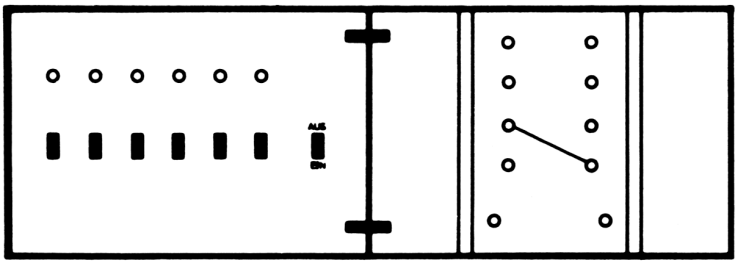
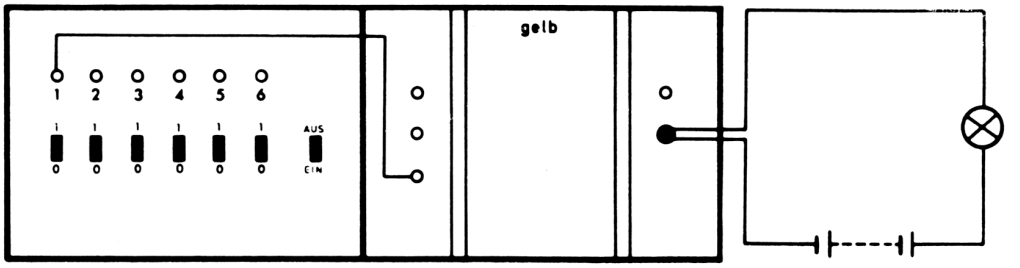


Abb. 15o



Jetzt leuchtet die Lampe solange, wie der Eingang C ein 0-Signal erhält. Bei einer 1 dagegen erlischt sie, so wie Sie es auch bei der Negations-Programmierung der Logik-Bausteine kennengelernt haben. Diese beiden Schaltungen mit dem Relais-Baustein sollten Ihnen nur deutlich machen, daß er zunächst einmal alle die Aufgaben erfüllen kann, die auch die Logik-Bausteine bisher erledigten. Sie können das überprüfen, indem Sie sämtliche Funktionen nacheinander programmieren, die Abhängigkeit des Ausgangssignals von den Eingangssignalen feststellen und mit den Funktionstabellen vergleichen.

Darüber hinaus liegt die Bedeutung dieses Bausteins darin, daß Sie jetzt direkt mit Ihrem Computer-Lehrbaukasten Vorgänge außerhalb der Bausteine beeinflussen können. So läßt sich z. B.die Lampe durch einen Summer oder eine Glocke ersetzen, das Relais kann eine Spielzeugeisenbahn nach vorher festgelegten Bedingungen anhalten

oder abfahren lassen, Lichtsignale stellen usw. Wir wollen Ihnen im folgenden einige Beispiele anführen, die Sie vielleicht auch zum weiteren selbständigen Probieren anregen.
Sie können übrigens die externe Batterie auch durch eine andere Stromquelle ersetzen. Allerdings darf die Spannung 24 V, die Stromstärke 2 A nicht übersteigen.

17.3. Einfache Alarmanlage

Die nächsten Beispiele sollen Ihnen zeigen, wie Sie mit einer Alarmanlage Ihr Hab und Gut schützen können. Den Alarmfall können Sie entweder optisch (also mit einer Lampe) oder akustisch (mit einem Summer) oder auch durch beide Signale anzeigen lassen.

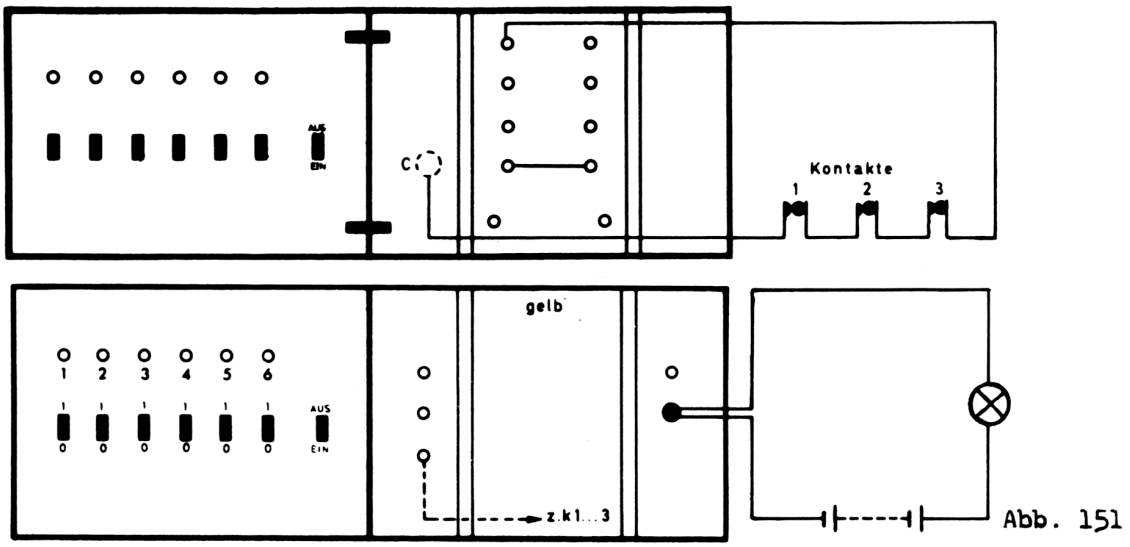


Abb. 151

Bei dieser Anlage werden die zu sichernden Fenster, Türen usw. direkt mit dem Relais-Baustein verbunden. Die Kontakte lassen sich einfach aus Alu-Folie herstellen - je ein Stück an der Tür und genau gegenüber am Rahmen. Das Kabel vom Eingang C wird mit dem metallischen Ende mit einer Heftzwecke auf die Alu-Folie gedrückt. Von der gegenüberliegenden Seite führt das Kabel - entsprechend befestigt - zum nächsten Kontakt oder zur Buchse 0 des Programmierfeldes.
Solange die Tür geschlossen ist, besteht eine durchgehende Verbindung von 0 nach C, also erhält C ein 0-Signal. Wird die Tür geöffnet, erhält C ein 1-Signal, weil dieser Eingang jetzt offenliegt und damit automatisch den Zustand 1 annimmt. Aufgrund der Identitätsprogrammierung erscheint an F ebenfalls 1, und je nachdem, was Sie angeschlossen haben: Die Lampe leuchtet oder der Summer ertönt.

17.4. Alarmanlage mit Speicher

Die einfache Alarmanlage hat noch eine Schwäche: Der Alarmfall wird nur solange angezeigt, wie das Fenster oder die Tür tatsächlich geöffnet ist. Wenn die Unterbrechung wieder aufgehoben ist, können Sie nachträglich nichts mehr feststellen.
Bei der im folgenden beschriebenen Anlage wird der Relais-Baustein als Speicher-Flipflop programmiert. Eine einmalige Unterbrechung der Alarmleitung wird dann so lange gespeichert, bis durch den Schalter der Eingabeeinheit der Eingang B ein 1-Signal erhält.

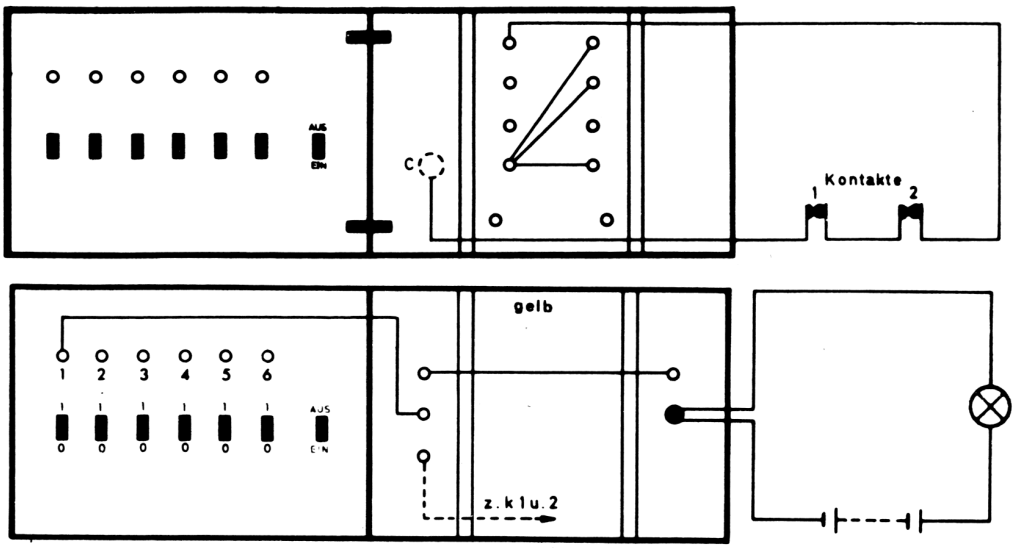


Abb. 152

17.5. Alarmanlage mit Verzögerung

Eine Alarmanlage soll jedes unbefugte Öffnen eines Fensters oder einer Tür anzeigen. Sie sollte aber so aufgebaut sein, daß derjenige, der die gesicherte Tür berechtigt öffnet, einige Sekunden Zeit hat, die Anlage abzuschalten. In der Weise können Sie Ihre Alarmvorrichtung erweitern.

An die Kondensator-Buchsen des Relais-Bausteins wird ein Elko von 2200 µF angeschlossen. Bitte, achten Sie auf die richtige Polung. Im übrigen kann die Anlage so verwendet werden wie beim letzten Aufbau.
Tritt nun der Alarmfall ein, d.h., der Eingang C erhält ein 1-Signal, wird die Baustein-Laufzeit so lange verzögert, bis der Kondensator aufgeladen ist. Das sind ca. 3 - 4 Sekunden. Erst dann ist der Ausgang F = 1, und die Anlage zeigt den Alarm an.

Wenn Sie diese Anlage in der Nähe der Eingangstür versteckt installieren, haben Sie etwa 3 - 4 Sekunden Zeit, durch Betätigen des Schalters den Alarm zu unterbinden. Sie dürfen allerdings nicht vergessen, die Anlage anschließend wieder betriebsbereit zu machen.

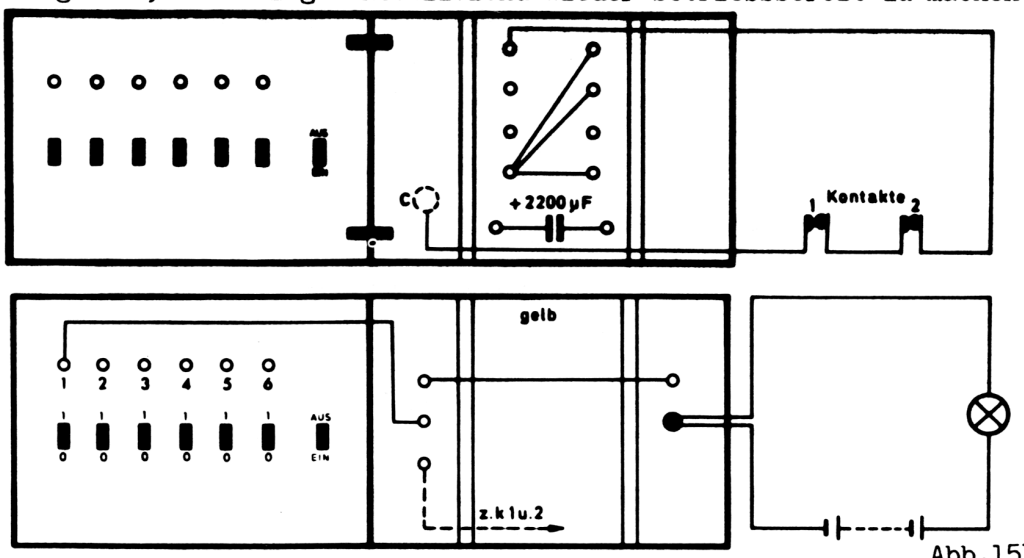


Abb.153

17.6. Lichtschranken-Sicherung

Während die Alarmanlage bei den vorigen Schaltungen stets durch Kontakte eingeschaltet wurde, soll sie jetzt durch eine Lichtschranke in Funktion treten. Ein LDR wird zwischen dem Eingang A und der Buchse 0 des Programmierfeldes eines Logik-Bausteins angeschlossen. Wird dieser LDR dann so am Fenster oder einer Tür in einem Papprohr installiert, daß kein Streulicht, sondern nur das Licht einer gegenüber angebrachten Lichtquelle darauf fällt, wird dieser Strahl beim Öffnen sofort unterbrochen. Durch die Identität des Logik-Bausteins ist F = 1, und dieses 1-Signal setzt die Alarmanlage am Relais-Baustein in Betrieb.

Durch die Verzögerung mit einem Kondensator können Sie natürlich selbst eintreten und innerhalb der angegebenen Zeit durch Betätigen des Schalters den Alarm verhindern.

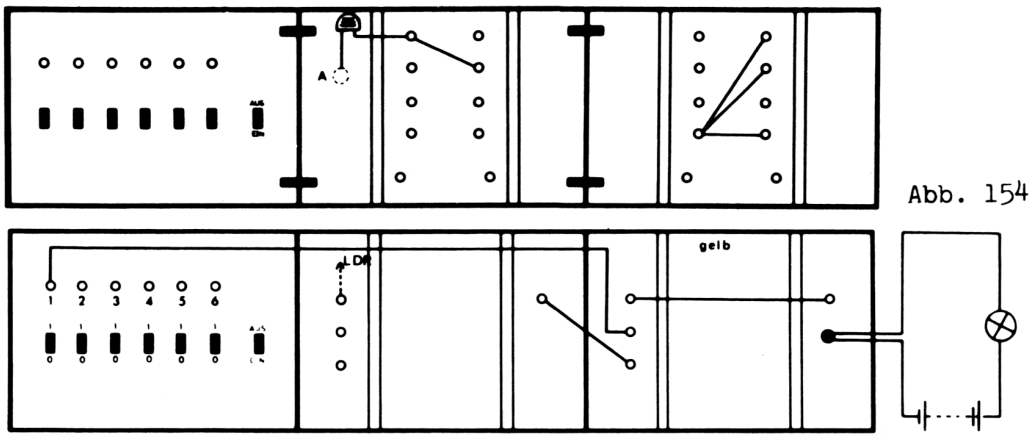


Abb. 154

17.7. Steuerung einer Verpackungsmaschine 1

In einer Konservenfabrik soll der Computer die Steuerung einer Verpackungsmaschine übernehmen. Je 8 Dosen werden in einen Karton gepackt. Das bedeutet, daß zu Beginn und nach jeder 8. Dose ein Karton bereitgestellt werden muß.
Für diese Schaltung benötigen Sie zwei Eingabeeinheiten, die eine ausreichende Stromversorgung garantieren. Dazu kommen sechs Logik-Bausteine, die zu drei Binärzählern programmiert werden. Vor das "Zählwerk" wird ein Baustein als Inverter mit einem lichtabhängigen Widerstand (LDR) geschaltet.
In der Praxis fällt über das Laufband ein Lichtstrahl auf diesen lichtabhängigen Widerstand. Wird der Lichtstrahl durch eine Konservendose, die auf dem Laufband transportiert wird, unterbrochen, ergibt das ein 1-Signal, das vom Zählwerk registriert wird.

Die Ausgänge F1 - F3 der drei Binärzähler werden als Eingangsvariable auf einen Relais-Baustein gegeben. Nach der vorgegebenen Programmierung - 0 auf den drei Ausgängen - wird bei diesem Zustand (F = 1) durch das Relais die Freigabe eines neuen Kartons ausgelöst.

F1	F2	F3	F
0	0	0	1
0	1	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Für das angegebene Beispiel darf F (Relais) nur dann den Wert 1 annehmen, wenn F1, F2 und F3 = 0. Daraus ergibt sich die NOR-Programmierung für den Relais-Baustein.

Natürlich sind auch andere Kombinationen von F1, F2, F3 für F = 1 denkbar. Dazu müßten Sie anhand einer neuen Funktionstabelle festlegen, wann F (Relais) den Wert 1 annehmen soll und den Relais-Baustein entsprechend programmieren.
Der Zähler wird in unserem Beispiel über den lichtabhängigen Widerstand (LDR) durch Anstrahlen mit einer Lampe und manuelle Unterbrechung des Lichteinfalls mit einem Stück Karton in Betrieb gesetzt. Die Funktion des Relais kann dadurch deutlich gemacht werden, daß an den Ausgang F ein zusätzlicher Stromkreis mit Glühlampe und Batterie angeschlossen wird.

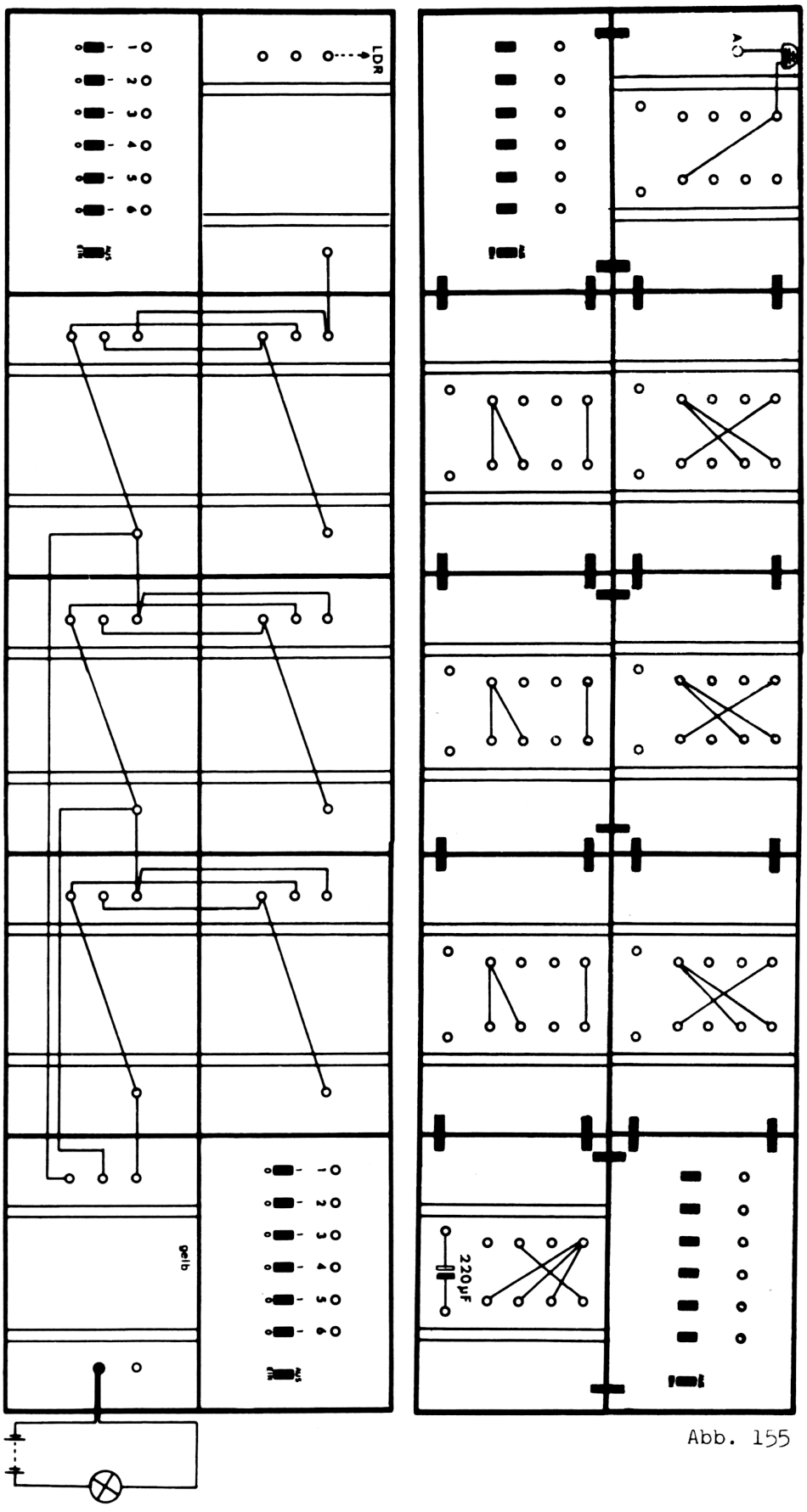


Abb. 155

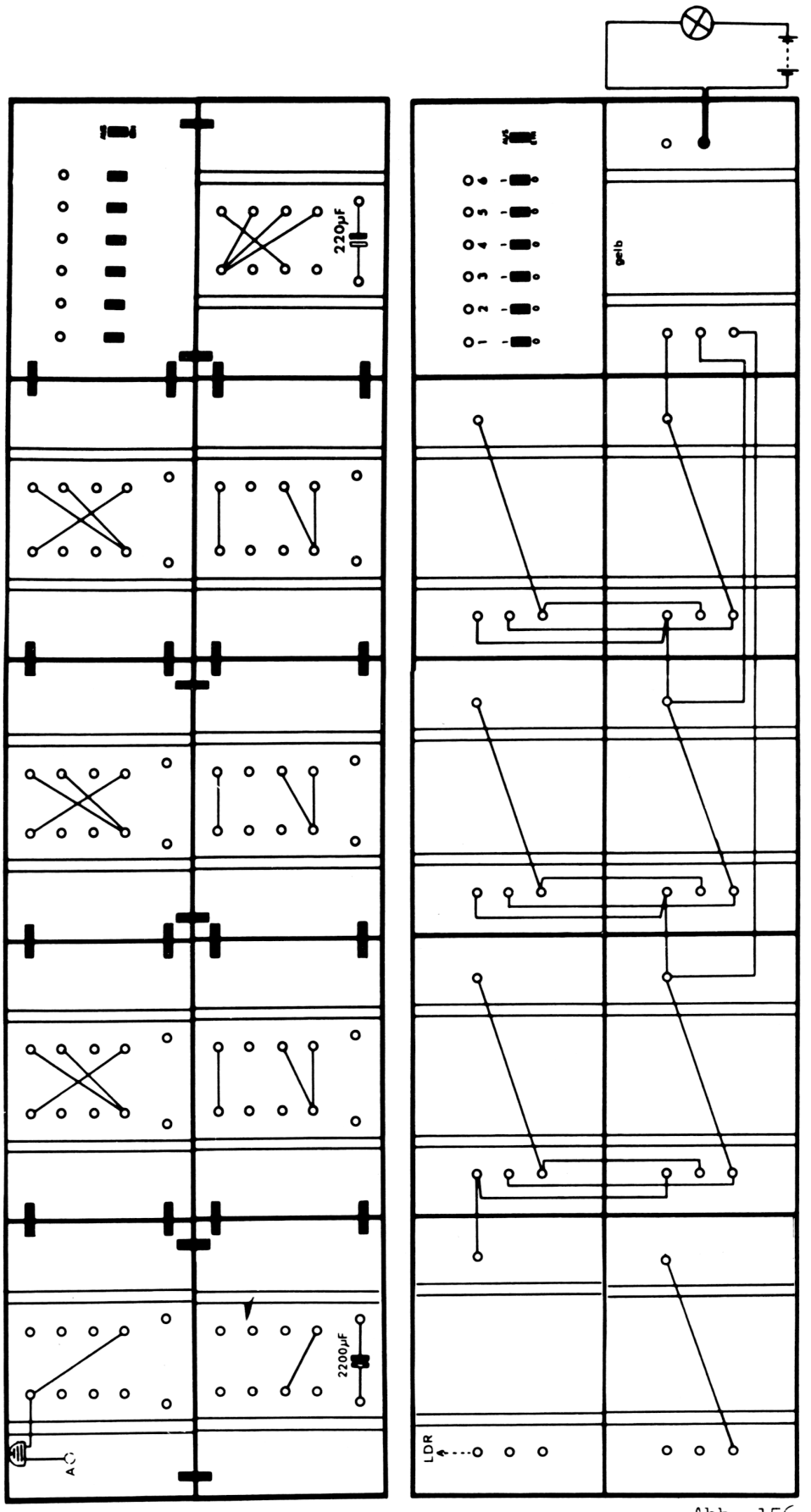


Abb. 156

Wenn das Relais anspricht, leuchtet die Glühlampe auf. Das entspricht der Bereitstellung eines Kartons in der Praxis.

17.8. Steuerung einer Verpackungsmaschine 2

Da bei manueller Unterbrechung des Lichteinfalls auf den LDR der Geschwindigkeit Grenzen gesetzt sind, kann zusätzlich ein Generator eingesetzt werden. Durch Auflegen des LDR auf die Ausgangsanzeige des Generators können Sie die Geschwindigkeit simulieren, mit der die Anlage arbeitet. (Abb. 156)

17.9. NIM-/Marienbadspiel

Das NIM- oder Marienbadspiel wird von zwei Spielern gespielt. Dazu werden 16 Streichhölzer in 4 Reihen in einer Formation aufgebaut, wie es die Abbildung zeigt. Nun müssen die beiden Spieler abwechselnd

I
III
IIIII
IIIIIII

Streichhölzer entfernen. Dabei dürfen nur aus einer Reihe Hölzer fortgenommen werden - es muß mindestens eins, es dürfen alle Hölzer einer Reihe entfernt werden. Wer das oder die letzten Hölzer aufnehmen kann, hat das Spiel gewonnen.
Die Strategie des Spiels besteht also darin, dem Gegenspieler eine Konstellation zu hinterlassen, die es ihm unmöglich macht, zu gewinnen.

Um beim Spiel jeweils die Konstellation überschauen zu können, werden die Hölzer einer Reihe in dualer Schreibweise notiert.

Reihe	
I	001
II	011
III	101
IV	111
	224

Bildet man nun daraus die dezimalen Spaltensummen - hier 2, 2, 4 - ergibt jede Spaltensumme eine gerade Zahl. Dann befindet sich der am Zug befindliche Spieler in Verliererposition. Das bedeutet, der Spieler, der das Spiel eröffnen muß, wird nicht gewinnen, wenn sein Gegenspieler die Strategie kennt und keinen Fehler macht. Es ist nämlich für ihn unmöglich, dann wieder nur gerade Spaltensummen zu hinterlassen.
Der im folgenden dargestellte Spielverlauf zeigt das:

X beginnt	Y findet vor	X findet vor	Y findet vor
001	001	001	001
011	011	011	001
101	001	001	001
111	111	011	011
224	124	024	014
und zieht 4 Hölzer aus Reihe III	und zieht 4 Hölzer aus Reihe IV	und zieht 2 Hölzer aus Reihe II	und zieht 2 Hölzer aus Reihe IV

X findet vor	Y findet vor	X findet vor	Y findet vor
001	000	000	000
001	001	000	000
001	001	001	000
001	001	001	001
004	003	002	001
und zieht 1 Holz aus Reihe I	und zieht 1 Holz aus Reihe II	und zieht 1 Holz aus Reihe III	und zieht das letzte Holz. Y gewinnt !

Da Spieler X beginnt, muß er verlieren, weil er Y keine geraden Spaltensummen hinterlassen kann. Y wird zum Schluß, wenn er keinen Fehler macht, immer eine ungerade Spaltensumme vorfinden - in diesem Falle 0, 0, 1 - kann das letzte Holz aufnehmen und gewinnt damit.

Die Kombinationen, die gerade Spaltensummen liefern, können Sie mit dem Computer-Lehrbaukasten ermitteln. Dazu werden die drei Spaltensummen mit den Buchstaben A, B, C bezeichnet, wobei A

immer der letzten Stelle der Dualzahl entspricht, B der zweiten, C der ersten Stelle; der Index gibt die Reihe an. Daraus läßt sich folgendes Schema entwickeln:

CBA	C	B	A
001			A1
011		B2	A2
101	C3	B3	A3
111	C4	B4	A4

Nach diesem Schema werden die Schalter von 2 Eingabeeinheiten wie in Abb.157 den vier Streichholzreihen zugeordnet.

A1	B2 A2	C3 B3 A3	C4 B4 A4
1	2 1	4 2 1	4 2 1
Reihe I	Reihe II	Reihe III	Reihe IV

Jeder Schalter nimmt damit eine dezimale Wertigkeit ein, und zwar

A = 1
B = 2
C = 4

Wird der Schalter A1 in 1-Stellung gebracht, bedeutet es, daß ein Holz der ersten Reihe gezogen wird - hier ist nach der Formation der Hölzer auch nur diese Möglichkeit gegeben.
Wird Schalter B2 in 1-Stellung gebracht, so bedeutet es, daß zwei Hölzer aus Reihe II gezogen wurden. Durch zusätzliche 1-Stellung des mit A2 bezeichneten Schalters wird der Wert 3 eingegeben. Entsprechend gilt das für die Schalter der Reihen III und IV. Die Schalter C3 und C4 haben die dezimale Wertigkeit 4.

Eine Gewinnkombination kann sich nur ergeben, wenn die Summen der Spalten A, B und C geradzahlig sind. Deshalb muß auch für jede Spalte eine eigene Funktionstabelle aufgestellt werden. Das bedeutet:

FA kann nur dann den Wert 1 annehmen, wenn entweder alle A-Eingänge = 0 bzw. zwei oder vier = 1 sind.

FB ist immer dann 1, wenn alle B-Eingänge = 0 oder zwei = 1 sind.

FC ist nur 1, wenn beide Eingänge = 0 oder = 1 sind.

Wie Sie der Funktionstabelle für FA entnehmen, läßt sich FA mit einem Logik-Baustein nicht realisieren, da 4 Eingangsvariable miteinander verknüpft werden müssen. Deshalb werden A2, A3, A4 auf die Eingänge A, B, C eines Logik-Bausteins gelegt. Der Ausgang ist Fz.
Fz darf nur dann = 1 sein, wenn alle Eingangsvariablen = 0 oder zwei = 1 sind. Daraus ergibt sich folgende Funktionstabelle für Fz:

A	B	C	F
A2	A2	A4	Fz
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A1 wird mit Fz in einem weiteren Logik-Baustein verknüpft. Der Ausgang ist FA. FA darf nur = 1 sein bei folgenden Kombinationen:

B	C	F
Fz	A1	FA
0	0	0
0	1	1
1	0	1
1	1	0

0 am Eingang B (Fz) bedeutet immer, daß die Summe aus A2 bis A4 ungerade ist. Kommt durch A1 eine duale 1 hinzu, ist die Gesamtsumme der Spalte A wieder gerade, und FA wird 1.

Ein 1-Signal am Eingang B (Fz) bedeutet eine geradzahlige Summe aus A2 bis A4. Kommt keine weitere duale Zahl hinzu, bleibt die Spaltensumme gerade, und FA wird = 1.

A1	A2	A3	A4	FA
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
-	-	-	-	-
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

A	B	C	F
FA	FB	FC	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

A	B	C	F
B3	C3	F	FG
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

A	B	C	F
B2	B3	B4	FB
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

B	C	F
C3	C4	FC
0	0	1
0	1	0
1	0	0
1	1	1

Aus den Spielbedingungen ergibt sich, daß die Gewinnanzeige nur dann leuchten darf (F = 1), wenn alle Spaltensummen geradzahlig, also A = B = C = 1 sind.

Die Gewinnanzeige F = 1 muß jedoch unterdrückt werden, wenn in Reihe III die Kombinationen 6 und 7 (4+2, 4+2+1) auf der Eingabeeinheit geschaltet werden, da diese Reihe nur 5 Hölzer enthält!
Die Information F wird deshalb auf C eines weiteren Logikbausteins (FG) gegeben, der das Gewinn-Signal F = 1 nur anzeigt, wenn B (Eingabe Reihe III = 4 Hölzer) und A (Eingabe Reihe III = 2 Hölzer) nicht gleichzeitig gedrückt (also 1) sind.

Danach ergibt sich der Versuchsaufbau nach Abb. 157.

Vor Spielbeginn müssen alle Schalter in 0-Stellung stehen. Wenn Sie nun auf der Eingabeeinheit die Anzahl der vom Gegner entfernten Hölzer durch 1-Stellung der betreffenden Schalter eingeben, können Sie selbst durch Probieren an den Schaltern herausfinden, wieviel Hölzer Sie in welcher Reihe entfernen müssen. Bei richtiger Konstellation leuchtet die Lampe der Gewinnanzeige (FG = 1) auf.

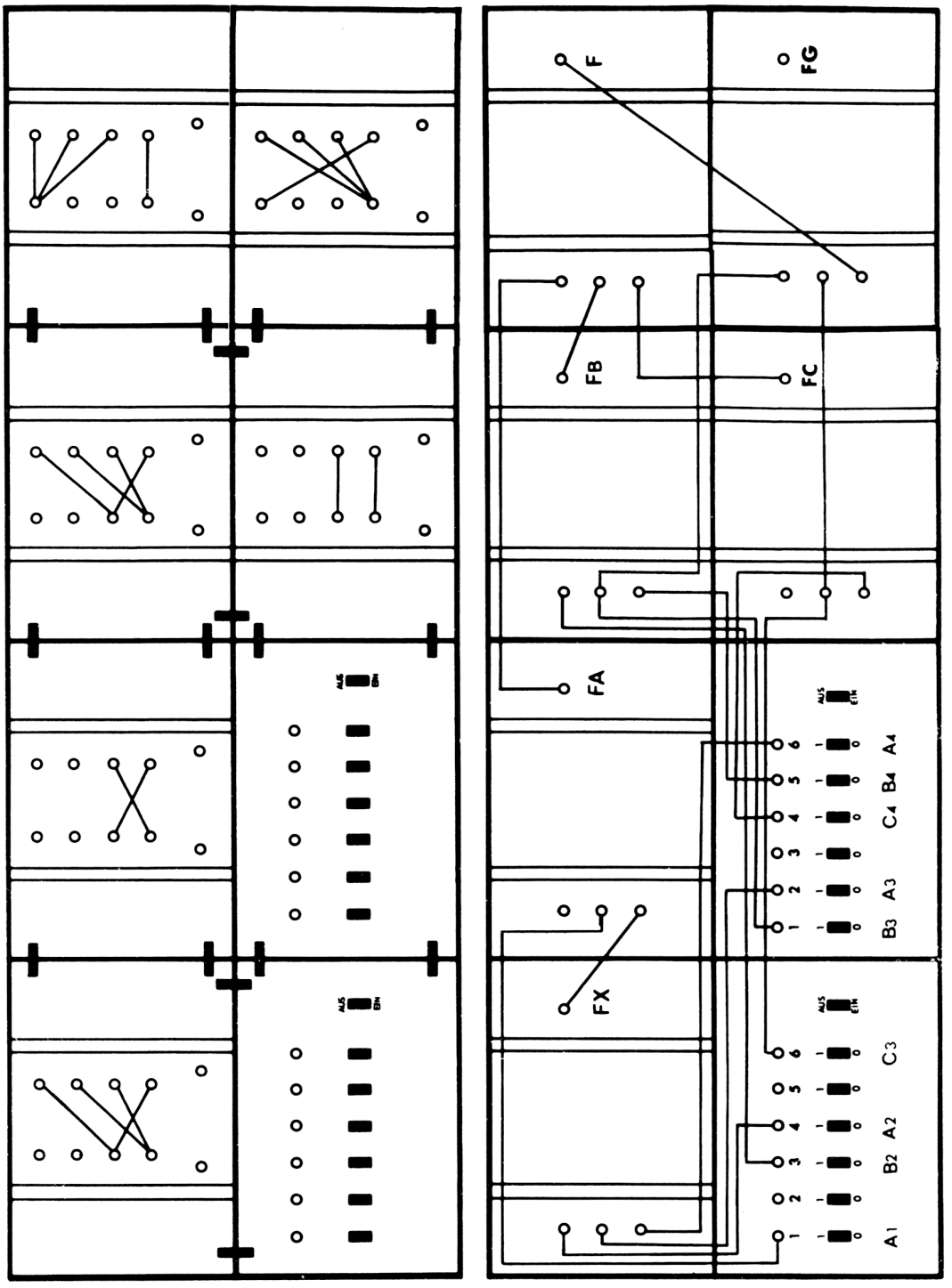
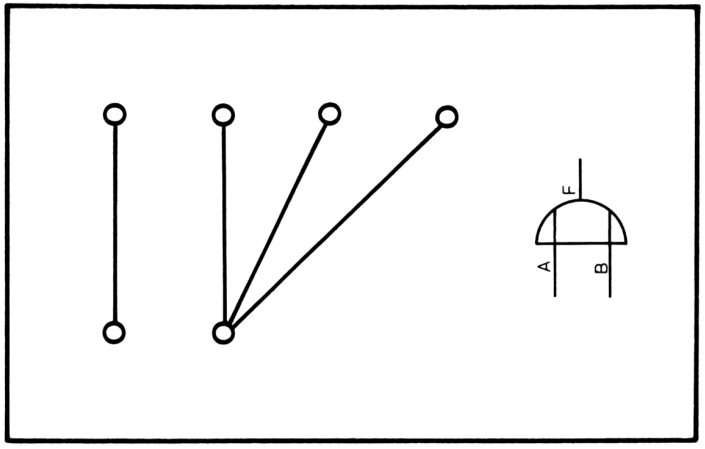
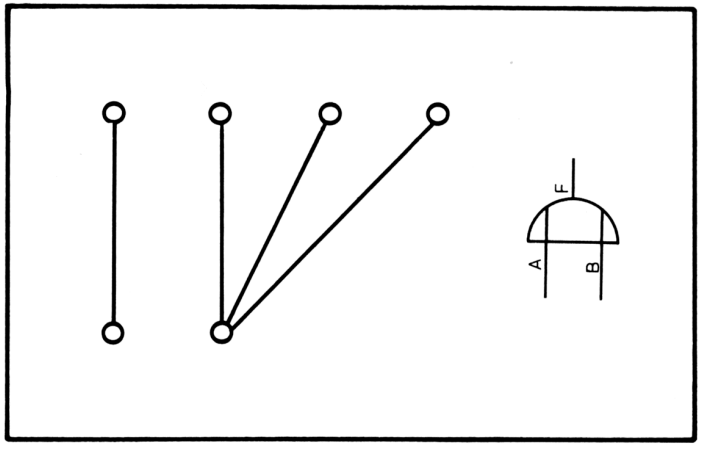
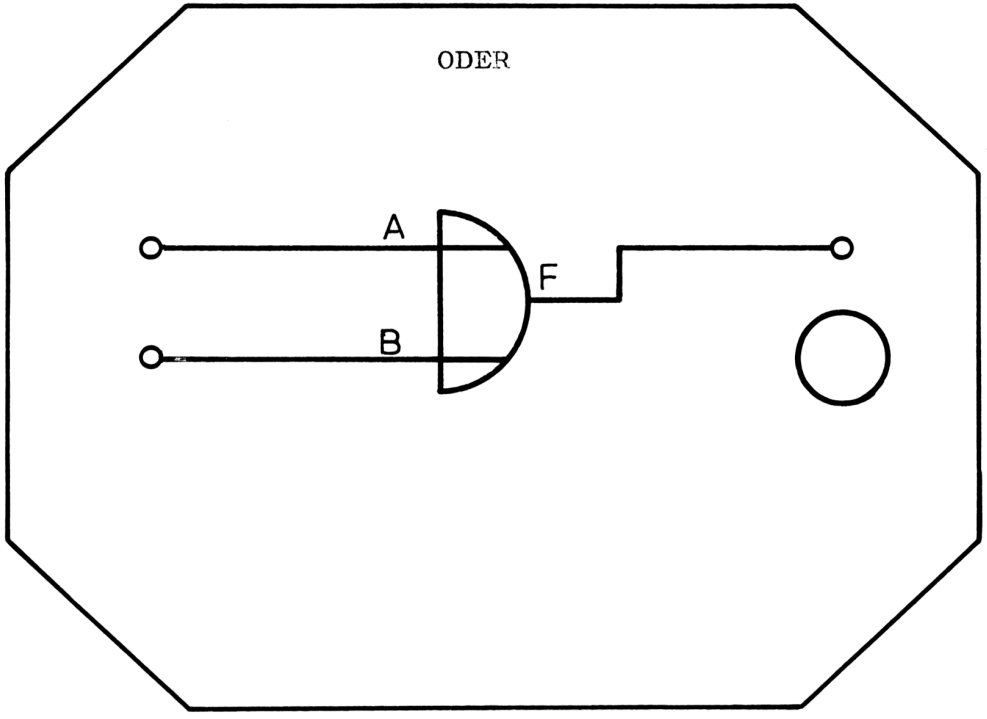
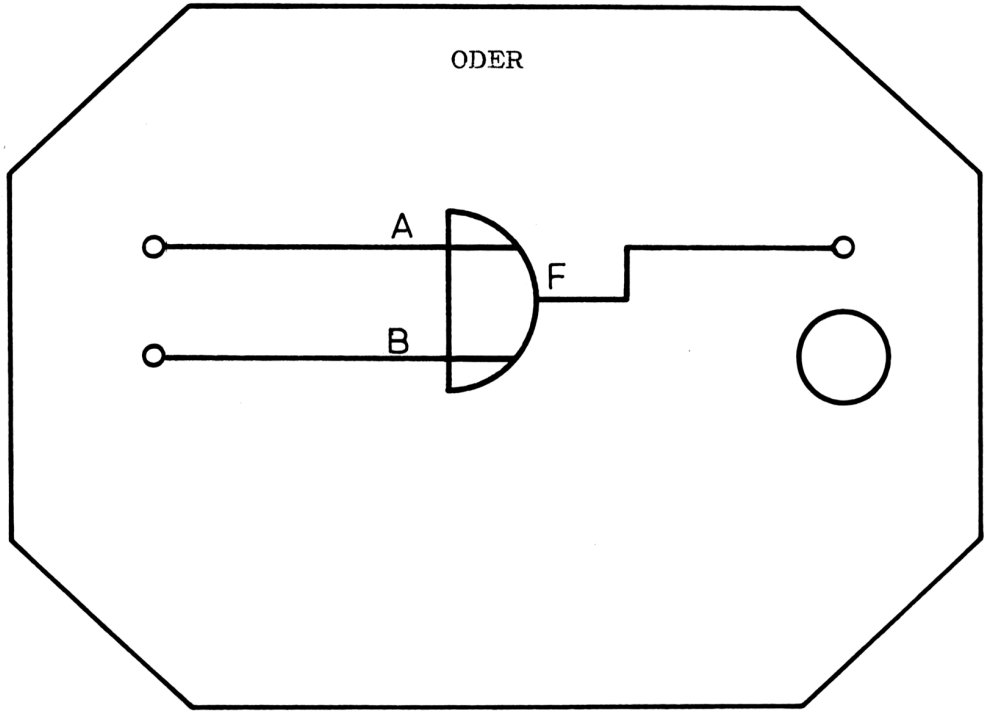


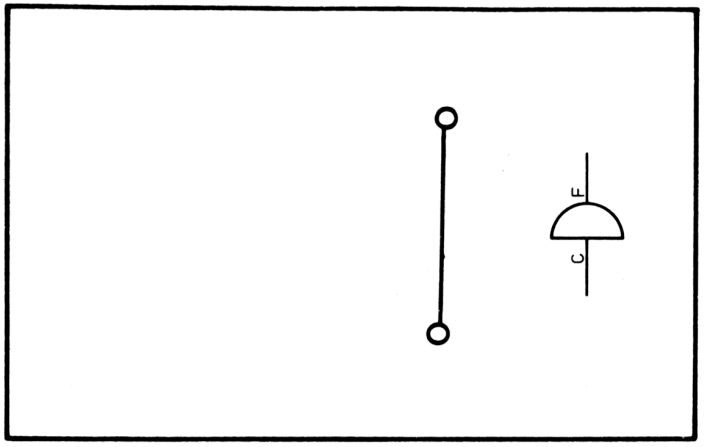
Abb. 157

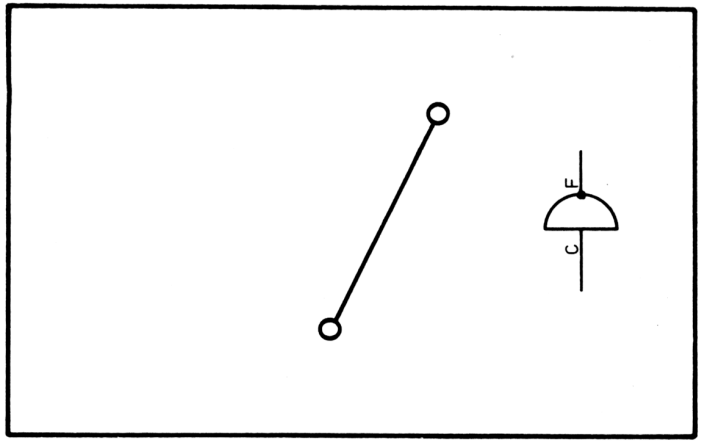

F
A
B


F
A
B

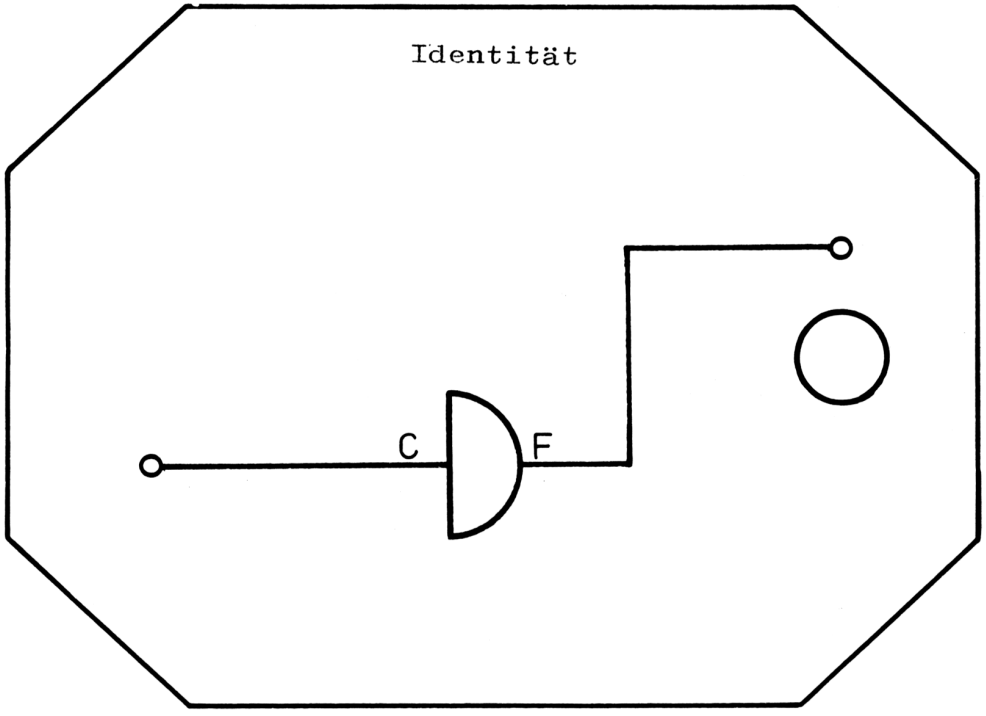

ODER
A
B
F


ODER
A
B
F

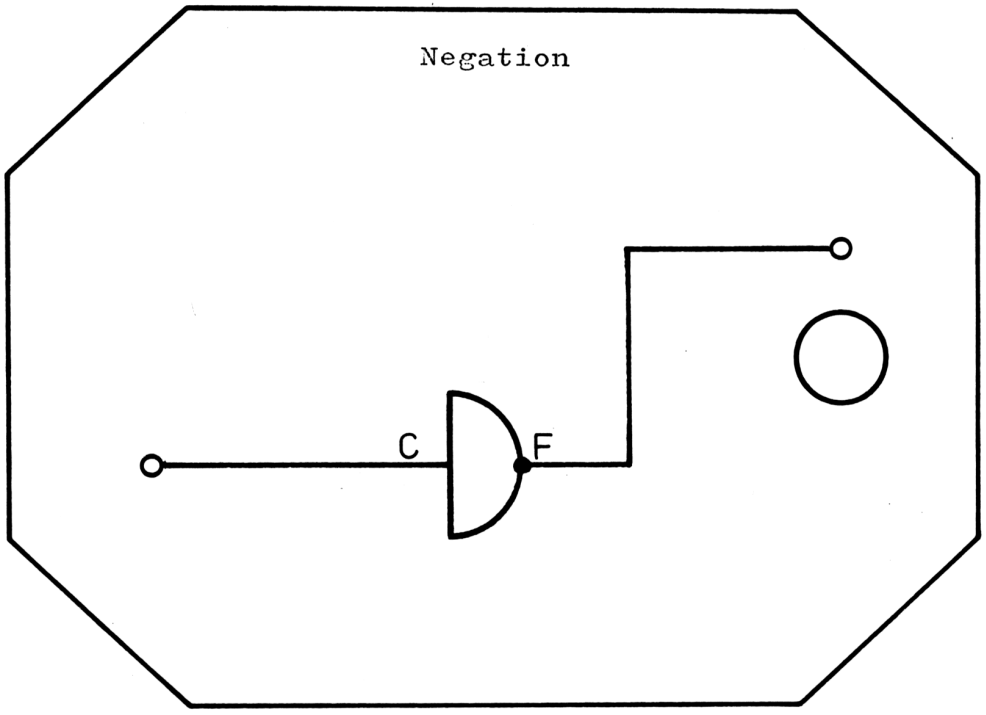


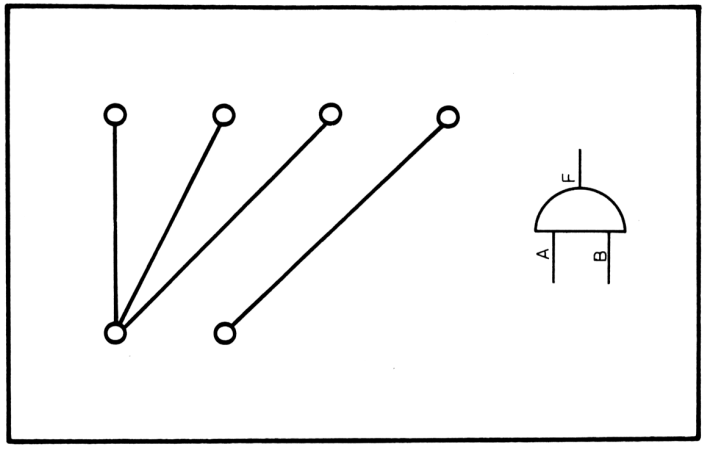


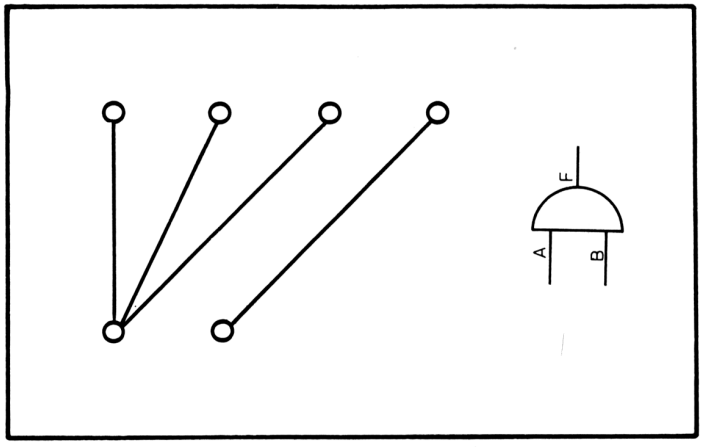
Identität



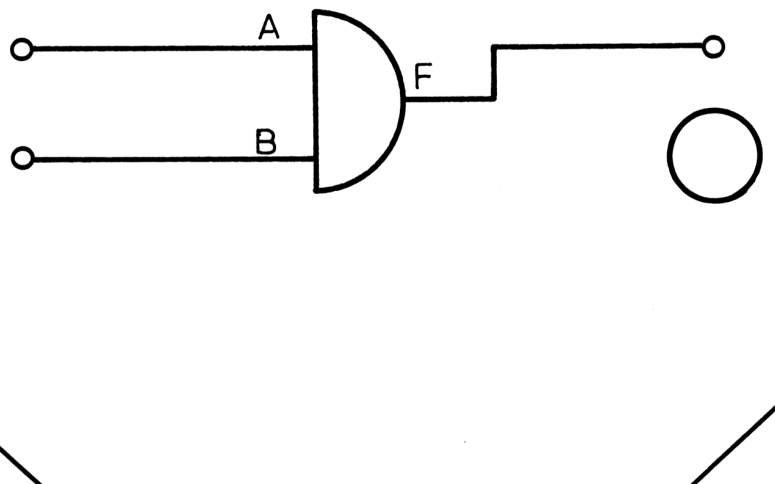
Negation



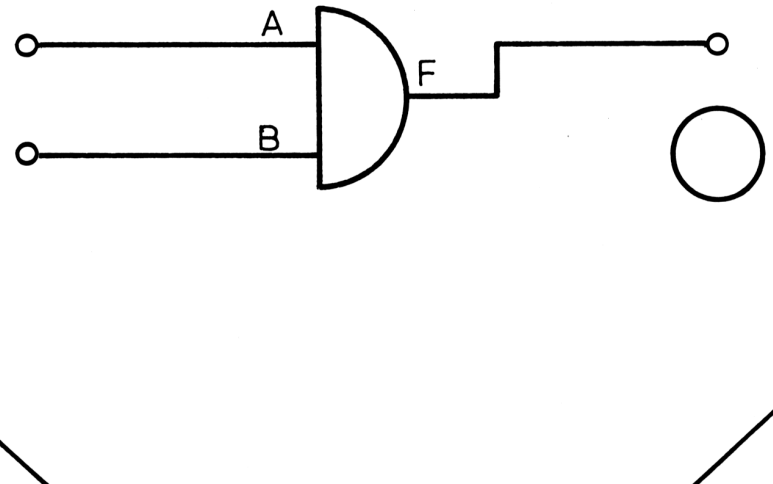


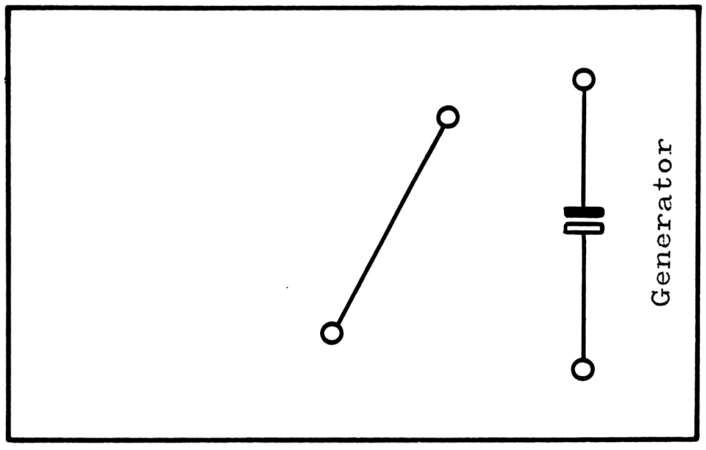


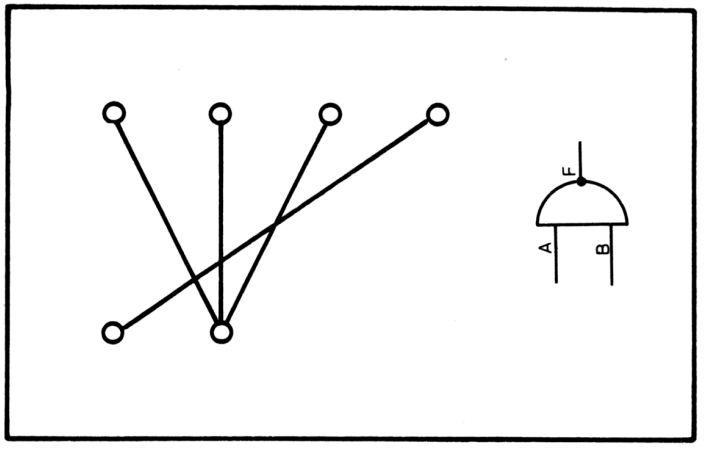
UND

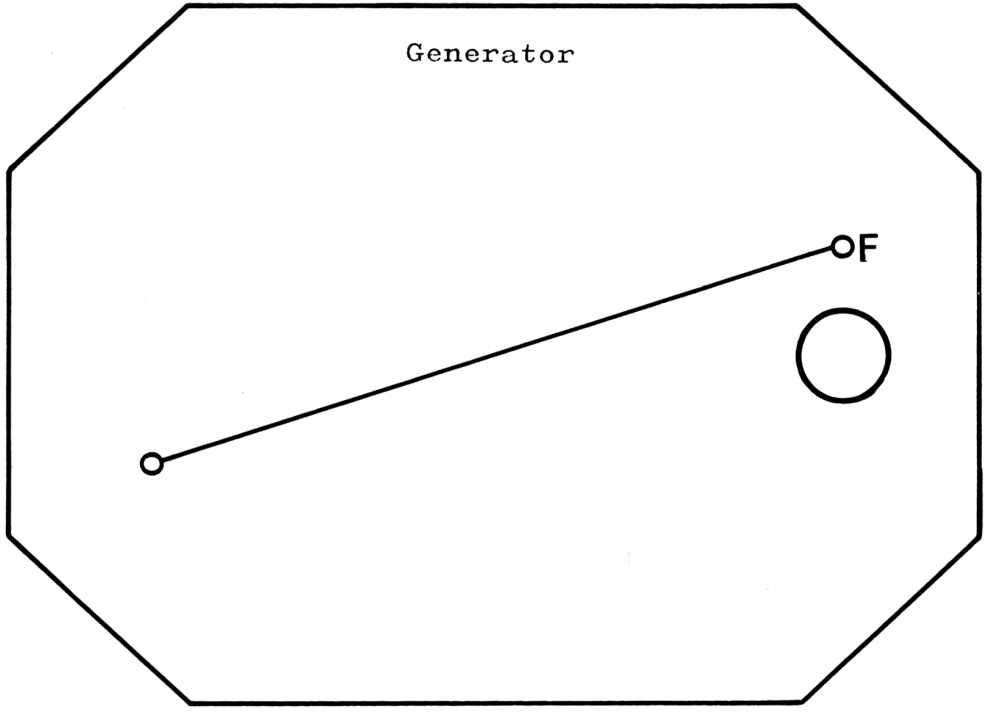


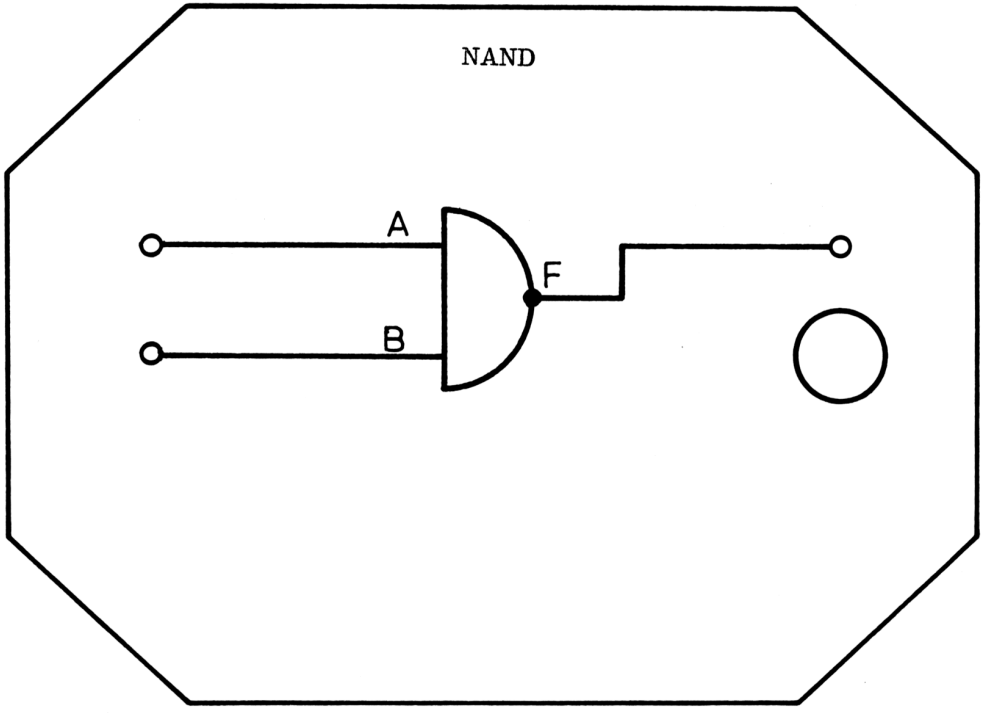
UND




Generator


F
A
B


Generator
F

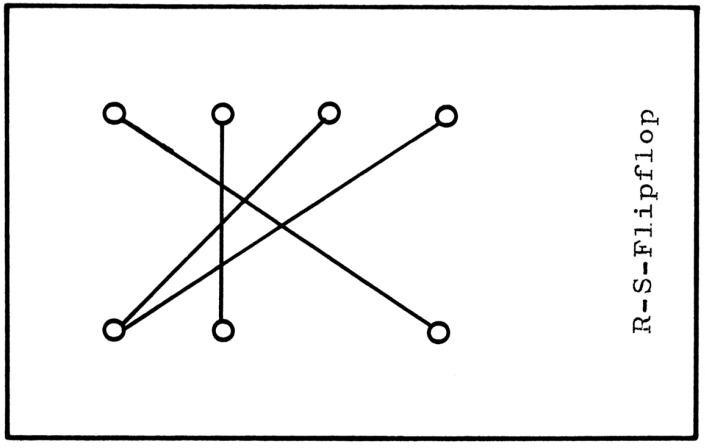

NAND
A
B
F

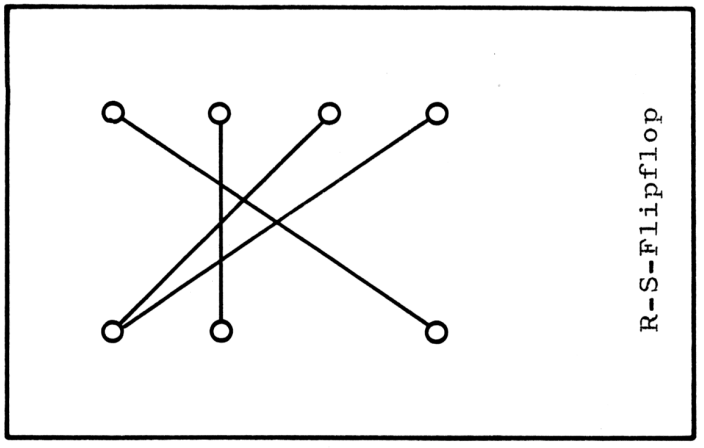
C
D
Q
DFF1

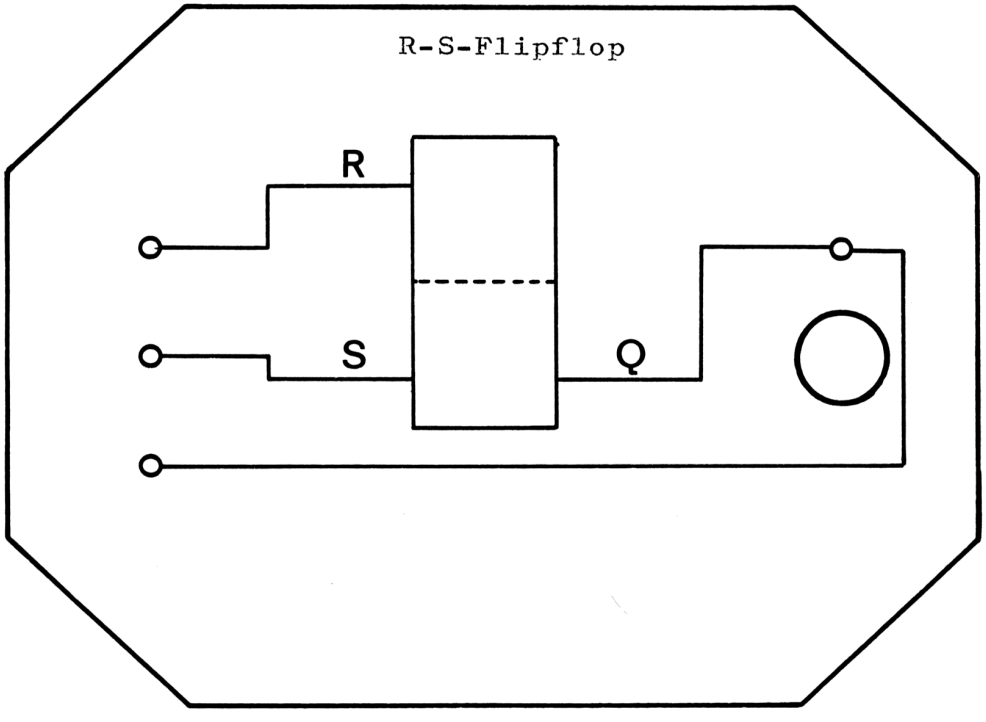
DFF1

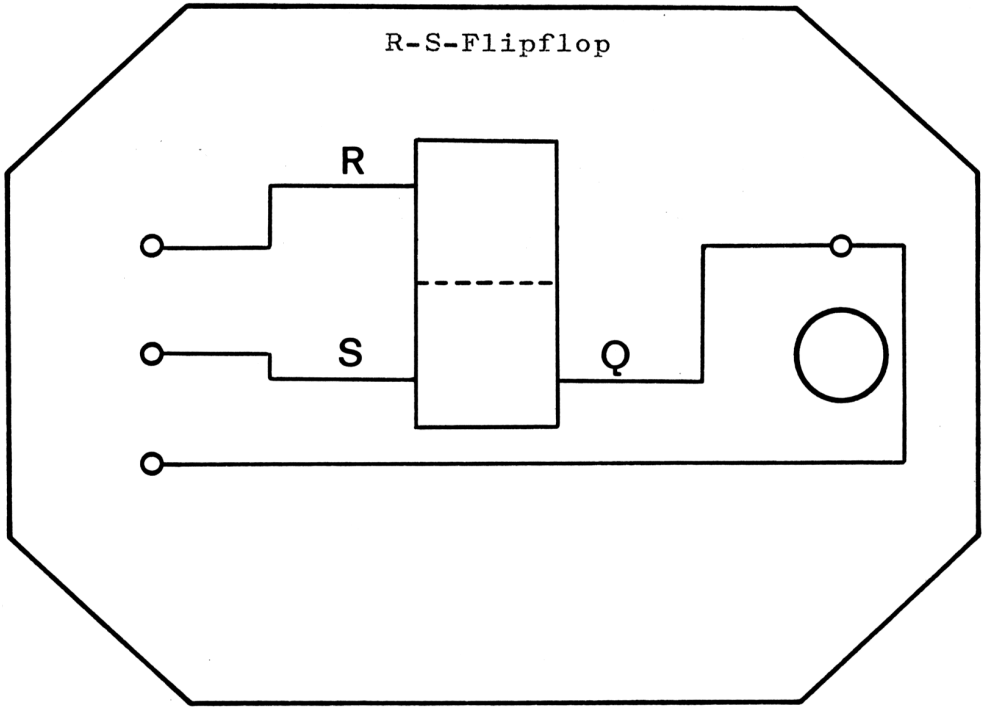
C
D
Q
DFF1

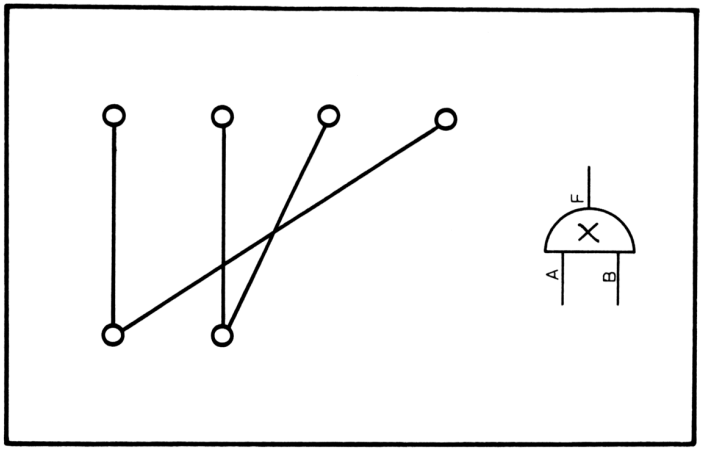
DFF1

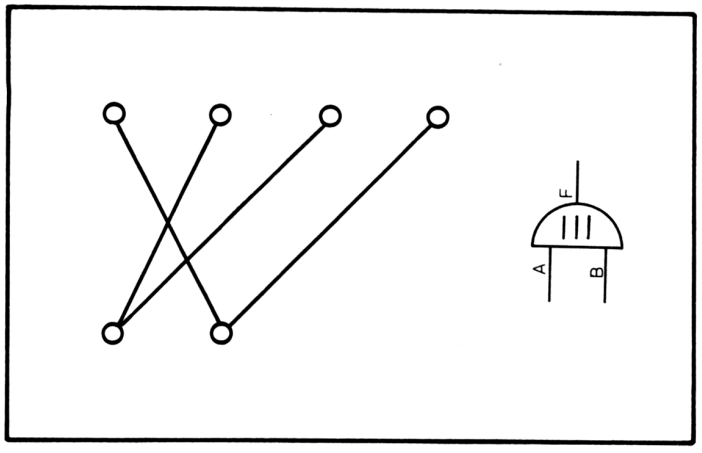

R-S-Flipflop


R-S-Flipflop

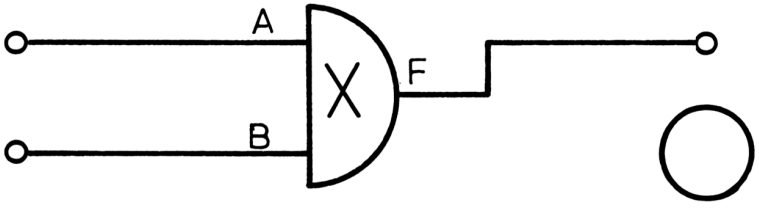

R-S-Flipflop
R
S
Q


R-S-Flipflop
R
S
Q

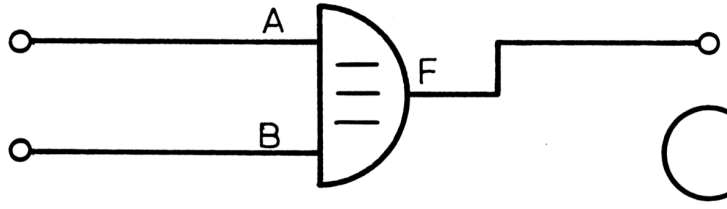


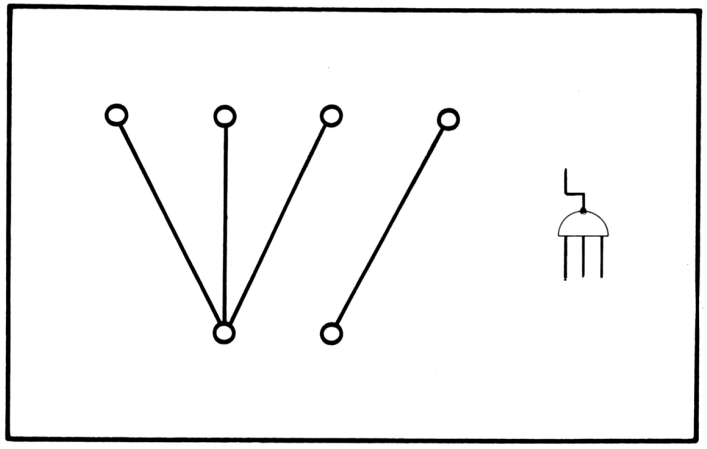


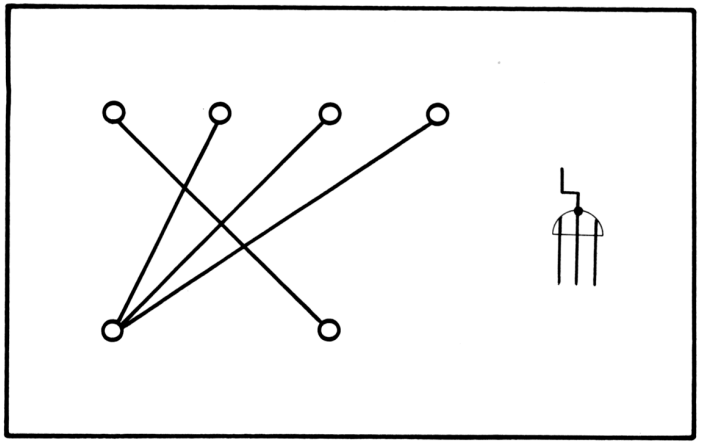
Exklusiv-ODER

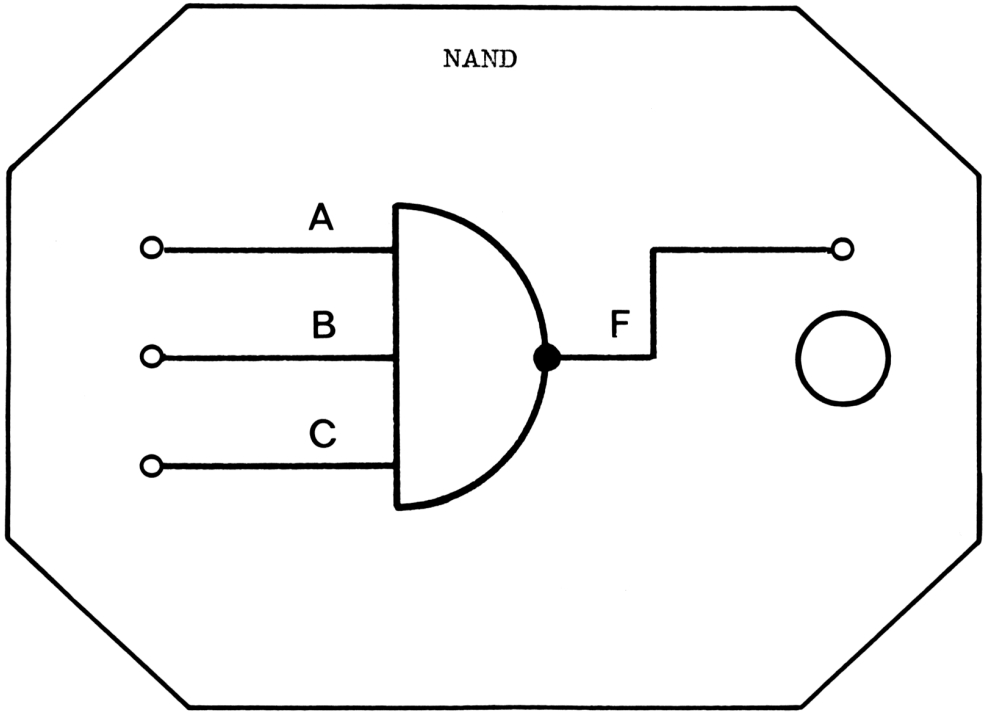


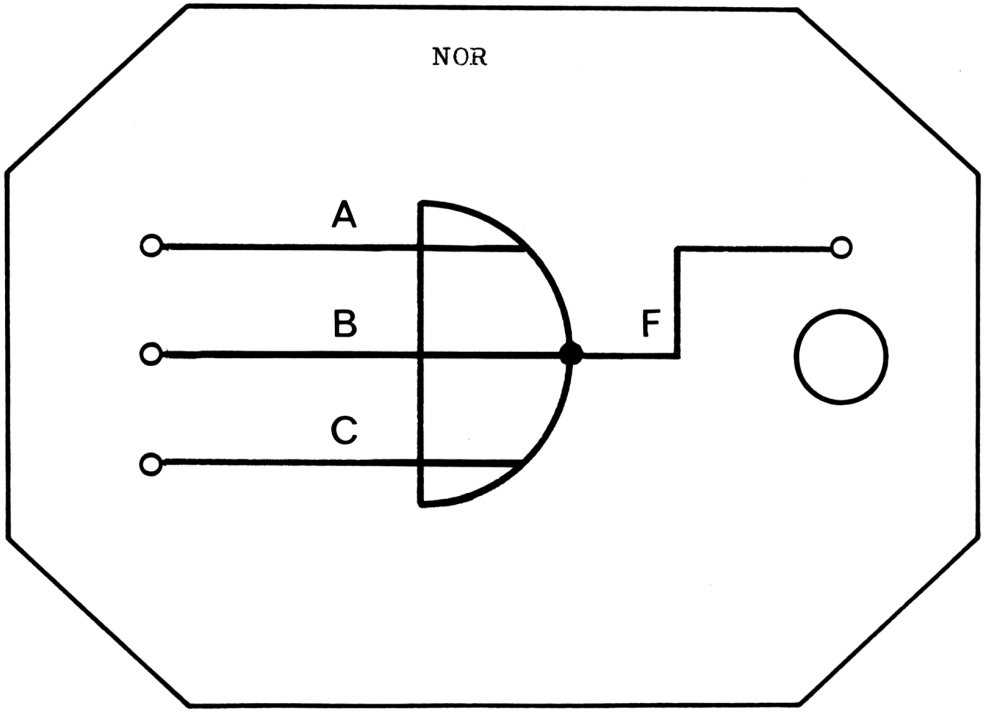
Äquivalenz

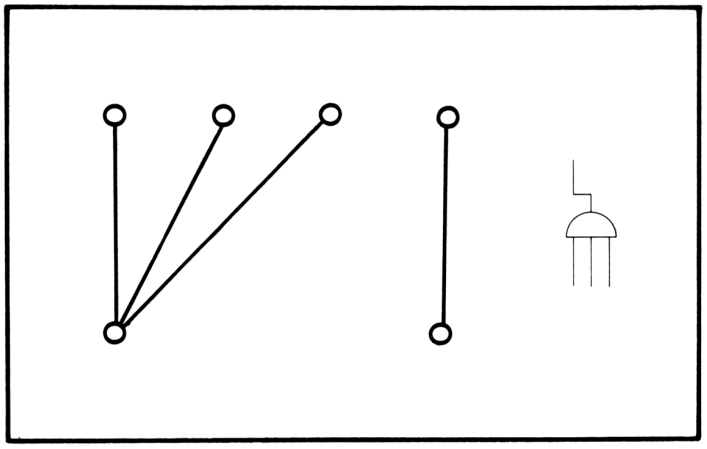


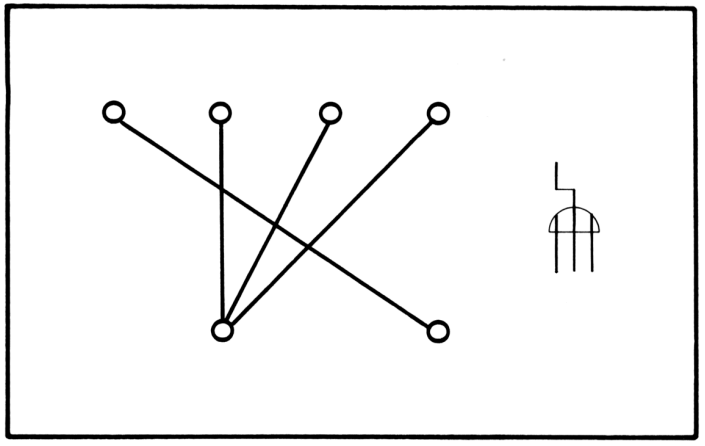




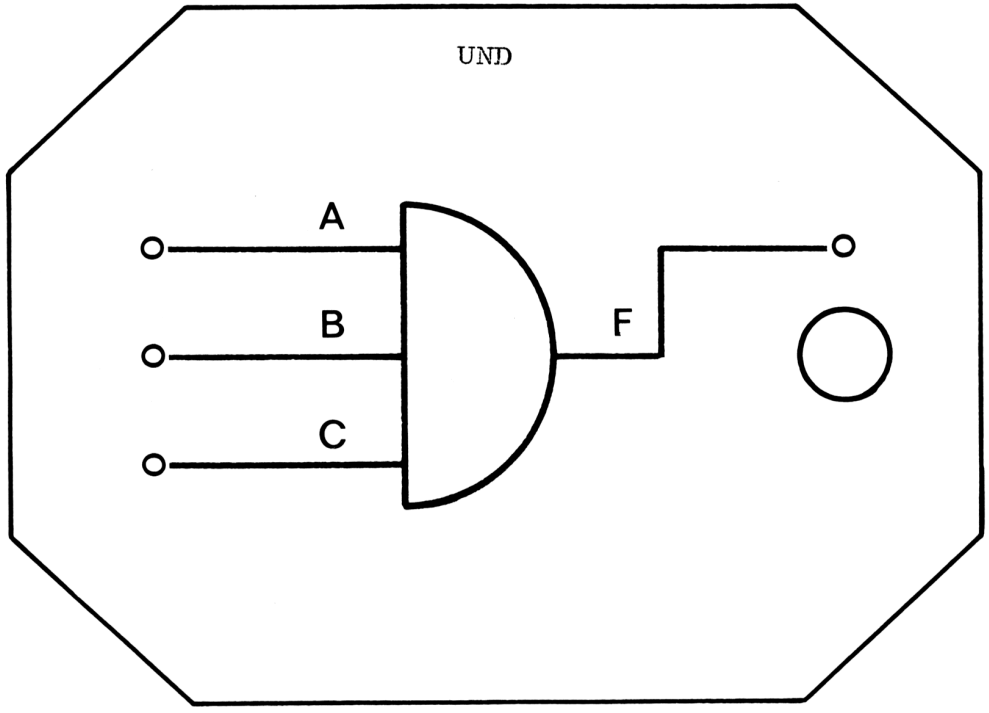

NAND
A
B
C
F


NOR
A
B
C
F





UND



ODER