
Experiments in Digital Technology

1st Edition

© hps SystemTechnik

Lehr- + Lernmittel GmbH
Altdorfer Strasse 16
88276 Berg / Germany

Phone: + 49 751 / 5 60 75 80
Telefax: + 49 751 / 5 60 75 17
Internet: <http://www.hps-systemtechnik.com>
E-mail: export@hps-systemtechnik.com

Order no.: V 0160

All rights reserved. No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior permission of hps SystemTechnik.

9.9.9

Foreword

Teaching in the field of „Digital Technology“ can be conducted for the most part with practical methods in experiment classes.

This manual „Experiments in Digital Technology“ offers, in concise form, all the teaching material which needs to be covered in basic and further training courses following the reorganisation of the electrical engineering vocations. Some of the experiments progress beyond the actual syllabus of vocational training colleges in part (e. g. experiments with the ALU and other components) so that they are also suitable for pupils at technical schools or electrotechnical courses at technical grammar schools.

The manual contains experiments with different degrees of difficulty for all the subjects offered to provide teachers with a reasonable choice.

In addition, the Digital Training System offers the teacher the possibility of adding his own experiments to the individual subject fields in order to select the right emphasis of content for his class.

The instructions have been compiled in such a way that, on the one hand, the pupil can be taught new knowledge and skills step by step, whereby independent planning, execution and evaluation of the experiments is aimed at, and on the other hand already available knowledge can be deepened and consolidated.

The chapter **Fundamental principles** contains an introduction to the respective subject.

The appropriate **Experiments section** contains the tasks and notes on their execution. Circuits, tables and diagrams are often provided in part to save the pupil time-consuming writing.

There is a detailed **Solutions section** at the back of the book with the answers to the set tasks and questions to allow the student to check his own work.

Notes:

List of Contents

Foreword	1	3. Bistable multivibrators	29
1. Basic logic circuits	3	3.1 Fundamental principles	29
1.1 Fundamental principles	3	3.1.1 General	29
1.1.1 Forms of representation and aids	3	3.1.2 Asynchronous flipflops	29
1.1.2 Laws of switching algebra	5	3.1.3 Synchronous flipflops	31
1.1.3 Complete disjunctive normal form	6	3.2 Experiments section	33
1.1.4 Complete conjunctive normal form	6	3.2.1 RS flipflop consisting of NOR gates	33
1.1.5 KV diagrams	7	3.2.2 RS flipflop consisting of NAND gates	34
1.1.6 Logic functions with NOR and NAND elements	8	3.2.3 Clock state controlled RS flipflops	35
1.2 Experiments section	9	3.2.4 RS flipflops with dominant S or R input ...	36
1.2.1 Important binary logic operations	9	3.2.5 D flipflops	37
1.2.2 Laws of switching algebra	10	3.2.6 Single edge controlled RS flipflop	38
1.2.3 Disjunctive and conjunctive normal form ..	15	3.2.7 Two state controlled D flipflop	41
1.2.4 Circuit design with the aid of KV diagrams	17	3.2.8 Single edge controlled JK flipflop	42
1.2.5 Representation of switching networks in NAND and NOR technology	19	4. Monostable multivibrators	47
1.2.6 Equivalence	22	4.1 Fundamental principles	47
1.2.7 Antivalence	23	4.2 Experiments section	48
1.2.8 Working with TTL components	24	4.2.1 The monoflop of the Digital Training System	48
2. Schmitt triggers	27	4.2.2 Delay circuits	49
2.1 Fundamental principles	27	5. Code converters, coders	53
2.2 Experiments section	28	5.1 Fundamental principles	53
		5.1.1 General	53
		5.1.2 8421-BCD / three-excess code converter .	53

5.2 Experiments section	56	7.1.4 Modulo-n counters	90
5.2.1 8421-BCD / Decimal code converter	56	7.1.5 Programmable counters	91
5.2.2 8421-BCD / 7-segment code converter	58	7.2 Experiments section	92
5.2.3 Coding circuits	61	7.2.1 Asynchronous up counters	92
6. Arithmetic circuits	63	7.2.2 Asynchronous down counters	94
6.1 Fundamental principles	63	7.2.3 Asynchronous reversing counters	96
6.1.1 General	63	7.2.4 Asynchronous modulo-n counters	97
6.1.2 Semi adder	63	7.2.5 Synchronous counters	99
6.1.3 Full adder	64	7.2.6 Programmable counters	102
6.1.4 Correction addition for decimal numbers	65	8. Register circuits	105
6.1.5 Subtractor for dual numbers	66	8.1 Fundamental principles	105
6.2 Experiments section	68	8.1.1 General	105
6.2.1 Semi adder	68	8.1.2 Shift registers	105
6.2.2 Full adder	70	8.1.3 Universal shift registers	106
6.2.3 Adding circuits for the 8421-BCD code	72	8.2 Experiments section	107
6.2.4 Semi subtractor	77	8.2.1 JK shift registers	107
6.2.5 Full subtractor	78	8.2.2 Shift registers with parallel data input	109
6.2.6 Subtractor for dual numbers	81	8.2.3 Serial data transfer	111
6.2.7 2-bit parallel multiplication circuit	83	9. Multiplex mode	115
6.2.8 Arithmetic unit for 4-bit dual numbers	85	9.1 Fundamental principles	115
7. Counting circuits	87	9.2 Experiments section	117
7.1 Fundamental principles	87	10. Arithmetic logic unit	121
7.1.1 General	87	10.1 Fundamental principles	121
7.1.2 Asynchronous counters	87	10.1.1 General	121
7.1.3 Synchronous counters	88		

10.1.2 Interaction of ALU and accumulator	121	12.2.2 Digital-Analog conversion	144
10.1.3 Arithmetic logic unit 74HC/HCT181	122	12.2.3 Signal transfer	146
10.2 Experiments section	124	12.2.4 Digital function generator	148
10.2.1 Arithmetic operations	124		
10.2.2 ALU with accumulator	126	Solutions section	S 1
11. Memory circuits	129	1. Basic logic circuits	S 1
11.1 Fundamental principles	129	2. Schmitt triggers	S 9
11.1.1 General	129	3. Bistable multivibrators	S 11
11.1.2 Extension of the word width	129	4. Monostable multivibrators	S 17
11.1.3 Extension of the number of memory locations	130	5. Code converters, coders	S 19
11.1.4 Memory components RAM 8 x 4 and EEPROM 8 x 4	131	6. Arithmetic circuits	S 25
11.2 Experiments section	132	7. Counting circuits	S 37
11.2.1 Writing and reading memory components	132	8. Register circuits	S 45
11.2.2 Word width	134	9. Multiplex mode	S 51
11.2.3 Memory capacity	137	10. Arithmetic logic unit	S 53
12. Analog-Digital converters, Digital- Analog converters	139	11. Memory circuits	S 55
12.1 Fundamental principles	139	12. Analog-Digital converters, Digital-Analog converters	S 59
12.1.1 General	139		
12.1.2 Analog-Digital converters	139	Appendix	A 1
12.1.3 Digital-Analog converters	140	1. List of symbols used in the formulae	A 1
12.2 Experiments section	141	2. Measuring instruments used	A 1
12.2.1 Analog-Digital conversion	141		
		Overhead foils	F 1
		DIGI BOARD 2	F 1
		DIGI MODULE BOARD	F 2
		Modules	F 3

1. Basic Logic Circuits

1.1 Fundamental Principles

1.1.1 Forms of Representation and Aids

Electronic digital circuits are composed of basic elements in which only two possible input and output signals need to be distinguished. These signals are represented either as a voltage level with the letters **H** (high) or **L** (low) or as a logic value with the values **0** and **1**.

Table 1.1.1.1 summarizes the possible designations of the input and output signals.

Positive logic is used for all the experiments in this manual.

The following forms of representation and aids are used to describe or comprehend the function of a single component or a digital circuit:

Circuit diagram/connection diagram:

Fig. 1.1.1.1 shows a circuit diagram or connection diagram with a NOT circuit with contacts as an example.

Level	Logic value	
	positive logic	negative logic
L	0	1
H	1	0

Tab. 1.1.1.1

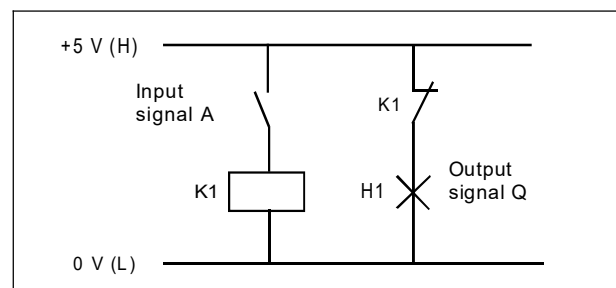


Fig. 1.1.1.1

Working table:

Table 1.1.1.2 shows the dependence of the output signals on the input signals as a level (L or H) with the NOT element as an example.

A	Q
L	H
H	L

Table 1.1.1.2

Value table/truth table:

Table 1.1.1.3 shows the relationship between the input and output signals in the form of logic states (0 or 1) with the NOT element as an example.

A	Q
0	1
1	0

Table 1.1.1.3 Truth table

Function equation:

The function equation shows the relationship between the input and output signal as a formula. Linkage of single or several signals is represented by special symbols.

The form $Q = A \cdot B$ is normally used because, firstly, it is very clear in connection with the symbols for the other switching functions and secondly because the truth table of a logic AND operation corresponds to the laws of arithmetic for multiplication of dual numbers and the form is permissible according to DIN.

NOT element: $Q = \overline{A}$ (read A negated)

AND operation: $Q = A \wedge B$ (read A and B)
or
 $Q = A \cdot B = A B$

OR operation: $Q = A \vee B$ (read A or B)

Logic plan:

A logic plan is the graphic representation of a digital circuit with circuit symbols.

Fig. 1.1.1.2 shows the circuit symbol of the AND element with the two inputs A and B.

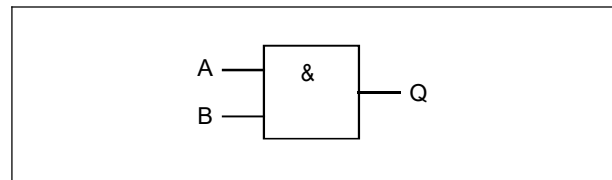


Fig. 1.1.1.2 Circuit symbol of an AND element

Timing diagram/pulse diagram:

A timing diagram or pulse diagram shows input and output signals of individual components or a digital circuit in relation to time.

Fig. 1.1.1.3 shows a timing diagram of a NOT element.

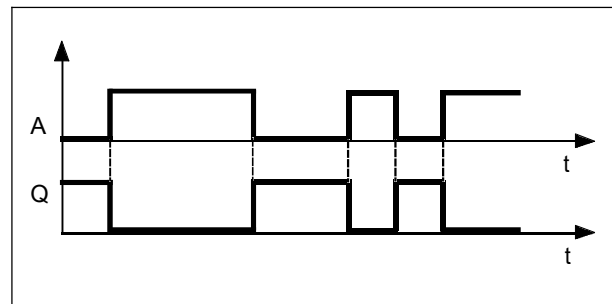


Fig. 1.1.1.3 Timing or pulse diagram

1.1.2 Laws of Switching Algebra

The logic operations AND, OR and NOT represent the three basic elements of digital logic. All simple control circuits can be described and set up with their help.

If the output signal of an AND or OR element is negated this produces a NOT AND or NAND operation or NOT OR or NOR operation.

NAND and NOR elements have their own circuit symbols as they are used very frequently.

The following rules or laws must be observed for working with the basic functions and the switching algebra expressions:

Commutative laws:

(exchangeability of variables)

$$A B = B A \quad \text{and} \quad A \vee B = B \vee A$$

Associative laws:

(allocation of variables)

$$A (B C) = (A B) C = (A C) B = A B C$$

$$A \vee (B \vee C) = (A \vee B) \vee C = (A \vee C) \vee B = A \vee B \vee C$$

Distributive laws:

(distribution of variables)

$$(A B) \vee (A C) = A (B \vee C)$$

$$(A \vee B) (A \vee C) = A \vee (B C)$$

Linkage laws:

These laws concern the linkage of a variable with itself (identities) or with a constant or with its negation (complement).

$A \wedge A = A$	<i>or</i>	$A \vee A = A$
$A \wedge 0 = 0$	<i>or</i>	$0 \vee A = A$
$1 \wedge A = A$	<i>or</i>	$1 \vee A = 1$
$A \wedge \overline{A} = 0$	<i>or</i>	$A \vee \overline{A} = 1$

Absorption laws:

(bonding rules)

$$A \wedge (A \vee B) = A$$

$$A \vee (A \wedge B) = A$$

Double negation:

$$\overline{\overline{A}} = A$$

De Morgan laws:

(laws of reversal)

$$\overline{A \wedge B} = \overline{A} \vee \overline{B} \quad \text{and}$$

$$\overline{A \vee B} = \overline{A} \wedge \overline{B}$$

1.1.3 Complete Disjunctive Normal Form

There are two possible ways of deriving a switching network from a task posed in the form of a truth table, the **complete disjunctive normal form** and the **complete conjunctive normal form**.

The complete disjunctive normal form results when the **minterms** are noted for all combinations for which the output has the „1" state and these are linked by OR functions. A minterm is given by noting the AND operation of all switching variables and negating the variables which are in the "0" state in this combination.

Table 1.1.3.1 shows an example of the complete disjunctive normal form.

A	B	C	Q	Minterms
0	0	0	0	
0	0	1	0	
0	1	0	1	$\bar{A} B \bar{C}$
0	1	1	0	
1	0	0	0	
1	0	1	1	$A \bar{B} C$
1	1	0	0	
1	1	1	1	$A B C$
$Q = \bar{A} B \bar{C} \vee A \bar{B} C \vee A B C$				

Table 1.1.3.1

1.1.4 Complete Conjunctive Normal Form

This results when the **maxterms** are noted for all combinations for which the output has the „0" state and these are linked by AND functions. A maxterm is given by noting the OR operation of all switching variables and negating the variables which are in the "1" state in this combination.

Table 1.1.4.1 shows an example of the complete conjunctive normal form.

The disjunctive or conjunctive normal forms taken from a value table are usually **redundant**, i. e. they contain superfluous components. The switching functions therefore need to be simplified. This minimization can be achieved arithmetically by applying the laws of switching algebra but it needs a lot of practice (see chapter 1.2.3, experiment 1, page 15).

Circuit minimization can be achieved faster and more reliably by a **KV diagram**, named after Karnaugh and Veitch. It is composed of chessboard-like fields, whereby the number of fields is equal to the number of signal combinations of the respective value table.

A	B	C	Z	Maxterms
0	0	0	1	
0	0	1	0	$A \vee B \vee \bar{C}$
0	1	0	1	
0	1	1	0	$A \vee \bar{B} \vee \bar{C}$
1	0	0	1	
1	0	1	1	
1	1	0	0	$\bar{A} \vee \bar{B} \vee C$
1	1	1	1	
$Z = (A \vee B \vee \bar{C}) \wedge (A \vee \bar{B} \vee \bar{C}) \wedge (\bar{A} \vee \bar{B} \vee C)$				

Table 1.1.4.1

1.1.5 KV Diagrams

A KV diagram can be drawn in a number of different ways. To save yourself long derivation work it is advisable to decide on one version. The example in fig. 1.1.5.1 describes the locating of the minimum disjunctive normal form. This is a KV diagram for four input variables with $A = 2^3$ (most significant bit).

The KV diagram contains all the minterms. These can be replaced by the corresponding dual numbers (fig. 1.1.5.2) for further simplification.

For application of the KV diagram the minterms are then read from the value table and transferred to the KV diagram.

The following should apply for the example in fig. 1.1.5.2:

$$Q = \bar{A}\bar{B}\bar{C}\bar{D} \vee \bar{A}\bar{B}C\bar{D} \vee \bar{A}B\bar{C}D \vee A\bar{B}\bar{C}\bar{D} \vee A\bar{B}\bar{C}D \vee A\bar{B}C\bar{D} \vee A\bar{B}CD$$

$$Q = 0 \vee 2 \vee 5 \vee 8 \vee 9 \vee 10 \vee 13$$

Then adjacent fields occupied by „1“ are combined in such a way that the combination has as many ones as possible. 2^n ($n = \text{natural number}$) fields may be combined. The KV diagram in fig. 1.1.5.2 shows the three possible combinations.

The appropriate switching algebra expressions are obtained by combining the variables common to all fields in one logic AND operation.

The expressions for the combination are:

$$(1) \quad B\bar{C}D$$

$$(2) \quad A\bar{C}D$$

$$(3) \quad \bar{B}\bar{D}$$

The minimum switching function is therefore:

$$Q = B\bar{C}D \vee A\bar{C}D \vee \bar{B}\bar{D}$$

It now only has eight variables in comparison with the complete disjunctive normal form.

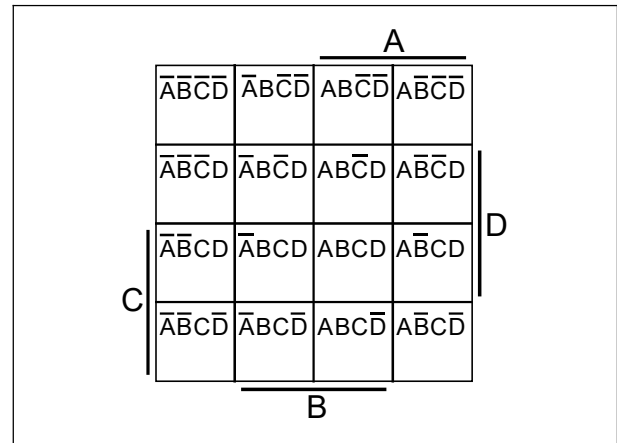


Fig. 1.1.5.1 KV diagram for four input variables

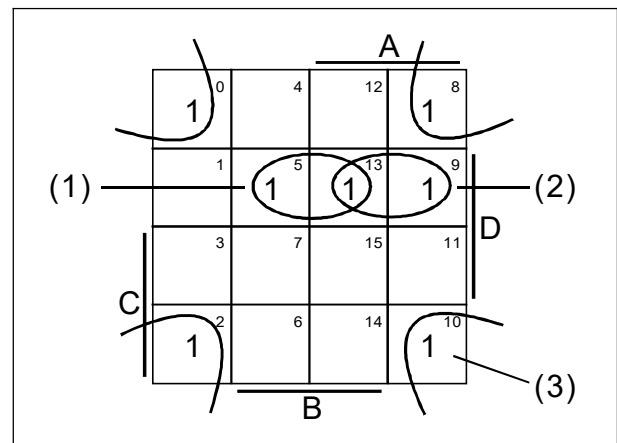


Fig. 1.1.5.2 Example

1.1.6 Logic Functions with NOR and NAND Elements

Since modern integrated circuits can only be manufactured economically in large series, NAND and NOR elements are mainly manufactured in IS technology because they can be used for setting up any switching network. However, the problem of transforming the respective switching function into an expression which contains only NAND or NOR operations is now added to the problem of simplifying the switching functions. The following laws of algebra offer the solution to this problem: De Morgan laws, double negation.

Since the form of writing with the help of NAND and NOR functions is very complicated, a switching network is developed by composing the switching functions from basic functions and minimizing them and only converting them into the desired logic operation in their final state.

Fig. 1.1.6.1 shows a summary of the logic functions with NOR and NAND elements described above.

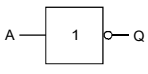
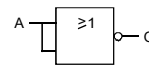
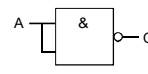
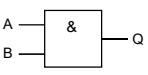
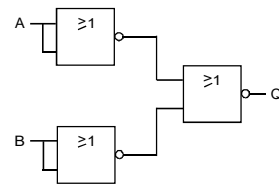
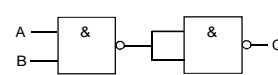
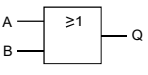
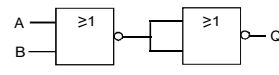
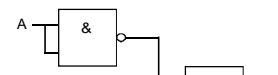
Logic functions with		
Basic elements	NOR elements	NAND elements
 $Q = \bar{A}$	 $Q = \bar{A}$	 $Q = \bar{A}$
 $Q = A \wedge B$	 $Q = A \wedge B$ $\bar{Q} = \overline{A \wedge B} = \bar{A} \vee \bar{B}$ $\bar{\bar{Q}} = Q = \overline{\bar{A} \vee \bar{B}}$	 $Q = A \wedge B$ $\bar{Q} = \overline{A \wedge B}$ $\bar{\bar{Q}} = Q = \overline{\overline{A \wedge B}}$
 $Q = A \vee B$	 $Q = A \vee B$ $\bar{Q} = \overline{A \vee B}$ $\bar{\bar{Q}} = Q = \overline{\overline{A \vee B}}$	 $Q = A \vee B$ $\bar{Q} = \overline{A \vee B} = \bar{A} \wedge \bar{B}$ $\bar{\bar{Q}} = Q = \overline{\bar{A} \wedge \bar{B}}$

Fig. 1.1.6.1

When linking two switching variables A and B, the **equivalence** and **antivalence** are also important in addition to the NOR and NAND operation (fig. 1.1.6.2).

In equivalence, the output always has „1“ state when both inputs have the same level. Equivalence elements are used above all in comparator circuits.

The antivalence always supplies a “1” to the output when both inputs have opposite levels. Antivalence elements are used partly in logic circuits (semi adder, semi subtractor) because their value table corresponds to the rules of arithmetic for two dual numbers (without taking the carry into account).

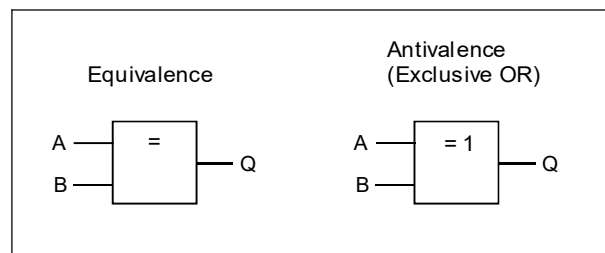


Fig. 1.1.6.2

1.2 Experiments Section

1.2.1 Important Binary Logic Operations

□ Experiment 1: Fundamental principles

Experiment procedure:

- Examine the circuit symbols shown in fig. 1.2.1.1 and fill in the missing data.
- Write the name of the circuit beneath each circuit symbol.

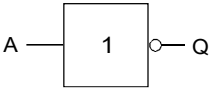

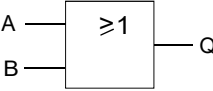

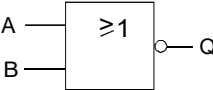
Circuit symbol	Value table	Circuits with contacts	Function equation															
	<table border="1"> <thead> <tr> <th>A</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> </tr> <tr> <td>1</td> <td></td> </tr> </tbody> </table>	A	Q	0		1												
A	Q																	
0																		
1																		
	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td></td> </tr> <tr> <td>0</td> <td>1</td> <td></td> </tr> <tr> <td>1</td> <td>0</td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td></td> </tr> </tbody> </table>	A	B	Q	0	0		0	1		1	0		1	1			
A	B	Q																
0	0																	
0	1																	
1	0																	
1	1																	
	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td></td> </tr> <tr> <td>0</td> <td>1</td> <td></td> </tr> <tr> <td>1</td> <td>0</td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td></td> </tr> </tbody> </table>	A	B	Q	0	0		0	1		1	0		1	1			
A	B	Q																
0	0																	
0	1																	
1	0																	
1	1																	
	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td></td> </tr> <tr> <td>0</td> <td>1</td> <td></td> </tr> <tr> <td>1</td> <td>0</td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td></td> </tr> </tbody> </table>	A	B	Q	0	0		0	1		1	0		1	1			
A	B	Q																
0	0																	
0	1																	
1	0																	
1	1																	
	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td></td> </tr> <tr> <td>0</td> <td>1</td> <td></td> </tr> <tr> <td>1</td> <td>0</td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td></td> </tr> </tbody> </table>	A	B	Q	0	0		0	1		1	0		1	1			
A	B	Q																
0	0																	
0	1																	
1	0																	
1	1																	

Fig. 1.2.1.1

1.2.2 Laws of Switching Algebra

□ Experiment 1: The commutative laws

Make sure that the commutative laws are correct.

Experiment procedure:

- Complete table 1.2.2.1 by entering the output states of the gates specified in fig. 1.2.2.1.

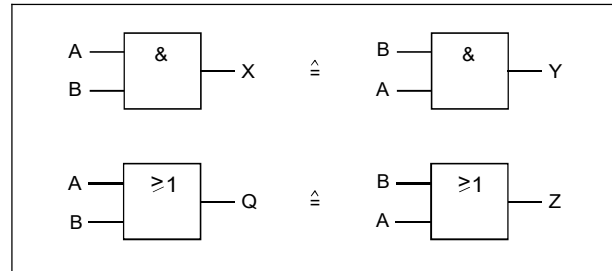


Fig. 1.2.2.1

N. B.:

The order need **not** be observed when assigning AND and OR inputs.

		AND		OR	
A	B	X	Y	Q	Z
0	0				
0	1				
1	0				
1	1				

Table 1.2.2.1 Truth table

□ Experiment 2: The associative laws

Make sure that the associative laws are correct.

Experiment procedure:

- Complete table 1.2.2.2 by entering the output states of the gates specified in fig. 1.2.2.2.

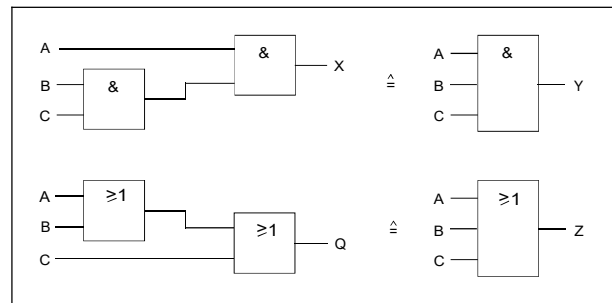


Fig. 1.2.2.2

N. B.:

For AND and OR circuits **one** gate with **many** inputs can be made up of **several** gates with **few** inputs.

			AND		OR	
A	B	C	X	Y	Q	Z
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 1.2.2.2 Truth table

Experiment 3: The distributive laws

Make sure that the distributive laws are correct.

Experiment procedure:

- Complete table 1.2.2.3 by entering the output states of the circuits specified in fig. 1.2.2.3.

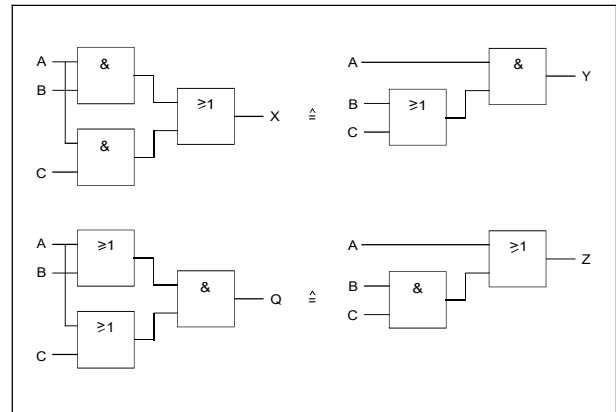


Fig. 1.2.2.3

N. B.:

If AND and OR terms combined by an OR or AND operation contain the **same** variable, this can be **factored out** as a common factor in mathematics. This saves gates.

A	B	C	X	Y	Q	Z
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 1.2.2.3 Truth table

Notes:

□ Experiment 4: Possible combinations of a switching variable

Check the possible combinations of a switching variable for identity, as a complement or as a constant.

Experiment procedure:

- Complete the tables 1.2.2.4 and 1.2.2.5 by entering the respective output state of the gate in fig. 1.2.2.4 and 1.2.2.5 once as a logic value and once as variable A or constant.

N. B.:

If an AND circuit has more inputs than are required, the inputs which are not needed can either be combined with the **required** inputs or applied to the level corresponding to **value 1**.

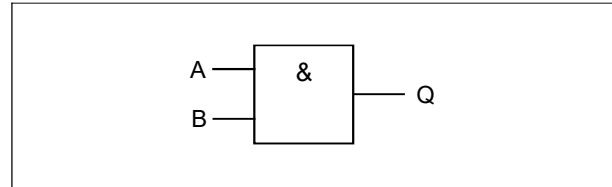


Fig. 1.2.2.4

A	B	Q as logic value	Q as variable A or constant
0	0		
1	1		
0	0		
1	0		
0	1		
1	1		
0	1		
1	0		

Table 1.2.2.4 Truth table

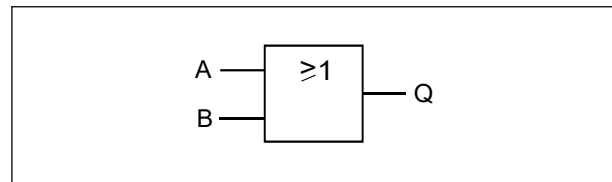


Fig. 1.2.2.5

A	B	Q as logic value	Q as variable A or constant
0	0		
1	1		
0	0		
1	0		
0	1		
1	1		
0	1		
1	0		

Table 1.2.2.5 Truth table

N. B.:

In an OR circuit, extra inputs must be combined with **used** inputs or applied to the level corresponding to **value 0**.

Experiment 5: Absorption laws

Make sure that the absorption laws are correct.

Experiment procedure:

- Complete the table 1.2.2.6 by entering the output states of the circuits specified in fig. 1.2.2.6 once as a logic value and once as a variable.

N. B.:

If OR and AND terms are linked by a common variable AND or OR, the output has the value of the **common** variable. This makes the circuit less complicated.

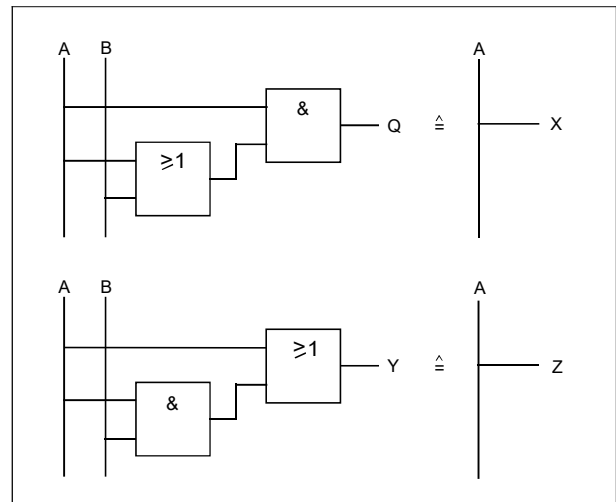


Fig. 1.2.2.6

A	B	Logic value				Output as variable
		Q	X	Y	Z	
0	0					
0	1					
1	0					
1	1					

Table 1.2.2.6 Truth table

Experiment 6: Double negation

Check the law of double negation.

Experiment procedure:

- Complete the table 1.2.2.7 by entering the output states of the circuits specified in fig. 1.2.2.7.

N. B.:

In digital circuits it is sometimes necessary to bridge the signal propagation times of certain gates or to amplify a signal. Inverters are suitable for this because the signal returns to its **original** state after the second inverter.

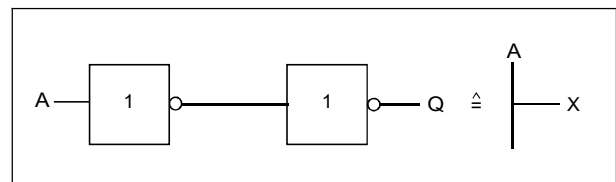


Fig. 1.2.2.7

A	Q	X
0		
1		

Table 1.2.2.7

□ Experiment 7: De Morgan laws

Check the De Morgan laws.

Experiment procedure:

- Complete the tables 1.2.2.8 ... 1.2.2.11 by entering the output states of the circuits specified in figs. 1.2.2.8 ... 1.2.2.11.

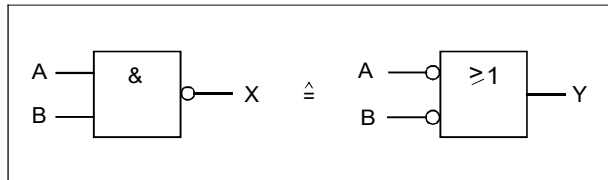


Fig. 1.2.2.8

A	B	X	Y
0	0		
0	1		
1	0		
1	1		

Table 1.2.2.8

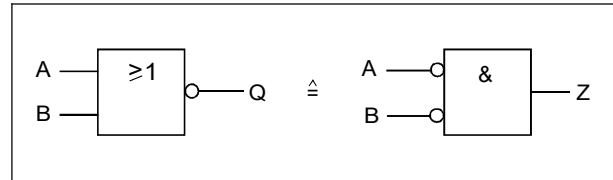


Fig. 1.2.2.9

A	B	Q	Z
0	0		
0	1		
1	0		
1	1		

Table 1.2.2.9

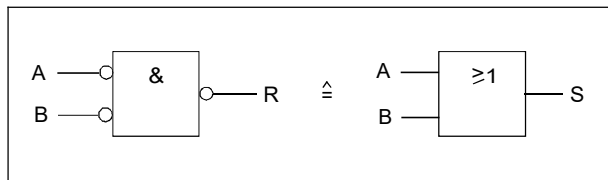


Fig. 1.2.2.10

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

Table 1.2.2.10

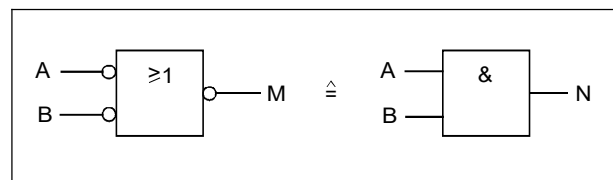


Fig. 1.2.2.11

A	B	M	N
0	0		
0	1		
1	0		
1	1		

Table 1.2.2.11

1.2.3 Disjunctive and Conjunctive Normal Form

□ Experiment 1: Controlling an electric network

A current network must be optically controlled to avoid overloading by built-in heating units. The three sets A, B and C have the rated powers 4 kW, 6 kW and 8 kW. If the power taken from the network exceeds 11 kW a yellow warning lamp should light up, if it exceeds 13 kW a red warning lamp should light up. A green lamp signals safe operation.

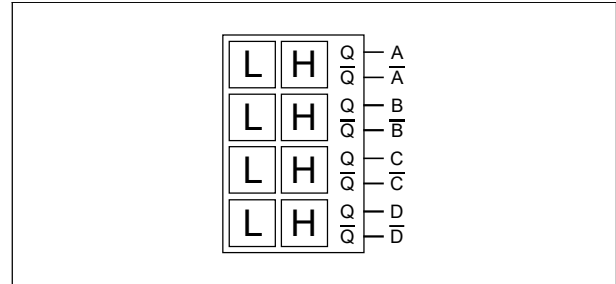


Fig. 1.2.3.1 Input keyboard

Experiment procedure:

- Complete the table 1.2.3.1 with binary numbers.
- Then design the switching functions for driving the lamps with the aid of the truth table and simplify these functions with the laws of the switching algebra.
- Complete the circuit in fig. 1.2.3.2 (page 16) and check its function with the Digital Training System.
- **Note:** Fig. 1.2.3.1 shows how the outputs of the input keyboard can be allocated to the input variables of a circuit.

4 kW A	6 kW B	8 kW C	P [kW]	Warning lamp		
				green	yellow	red
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 1.2.3.1 Value table

Green (conjunctive normal form):

X =

.....

.....

.....

.....

.....

.....

.....

Green (simplified): X =

Yellow (disjunctive normal form):

Y =

.....

.....

Yellow (simplified): Y =

Red (disjunctive normal form):

Z =

.....

Red (simplified): Z =

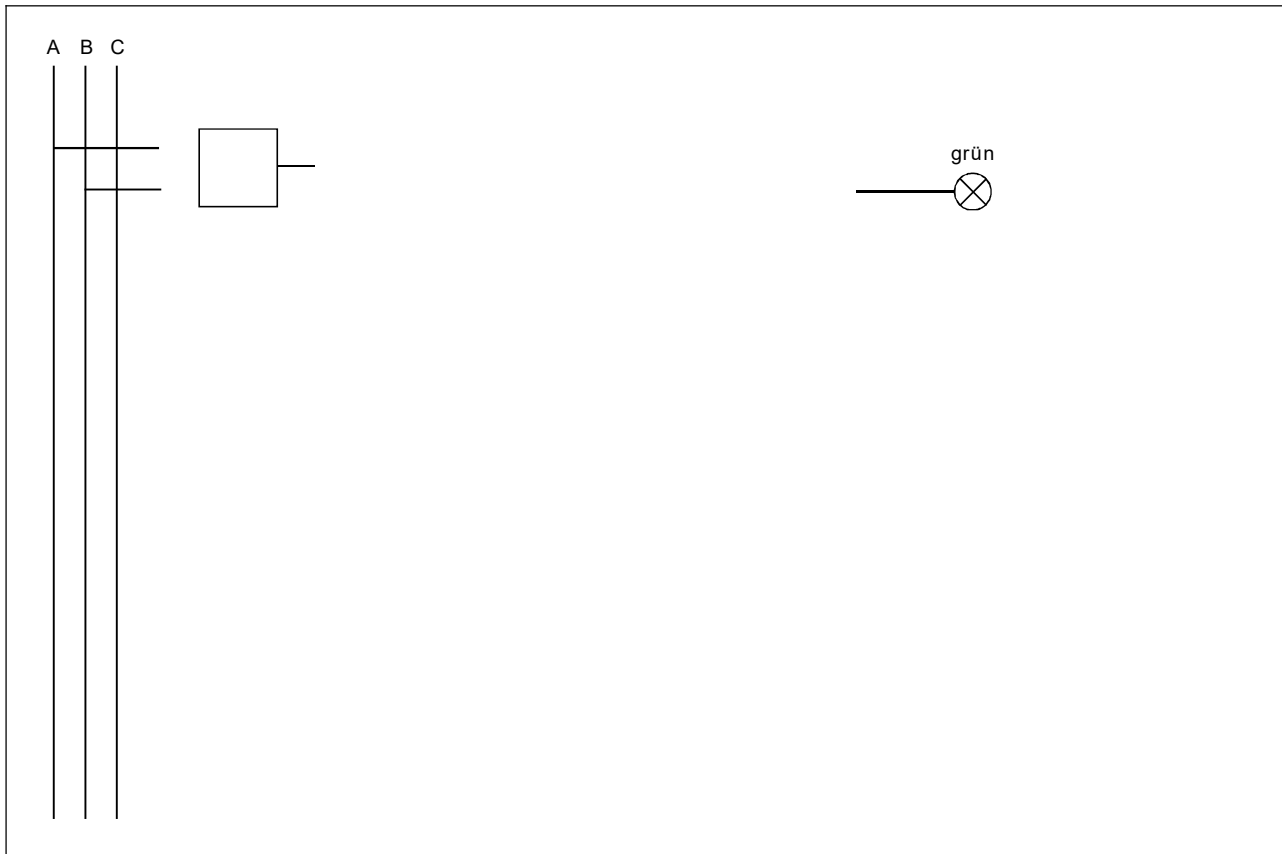


Fig. 1.2.3.2 Circuit

Notes:

1.2.4 Circuit Design with the Aid of KV Diagrams

□ Experiment 1: Number comparator for dual numbers

A digital circuit with which two dual numbers can be checked for equality or inequality is known as a number comparator.

In this experiment, a circuit is to be designed which is able to compare two dual numbers P and Q with two bits each. A greater-smaller comparison should also be carried out in the event of inequality.

Experiment procedure:

- Complete table 1.2.4.1 and design the three function equations Y1, Y2 and Y3 (complete disjunctive normal form) with the aid of the truth table.
- Minimize the three function equations with the aid of the KV diagrams (figs. 1.2.4.1 ... 1.2.4.3).
- Complete the circuit in fig. 1.2.4.4 (page 18) and check its function with the Digital Training System.

Y1 =

Y2 =

Y3 =

Inputs				Outputs		
Number P		Number Q		P > Q	P = Q	P < Q
A	B	C	D	Y1	Y2	Y3
0	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	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			

Table 1.2.4.1 Truth table

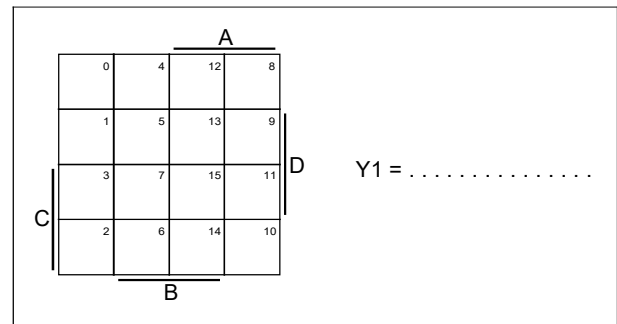


Fig. 1.2.4.1 KV diagram for Y1

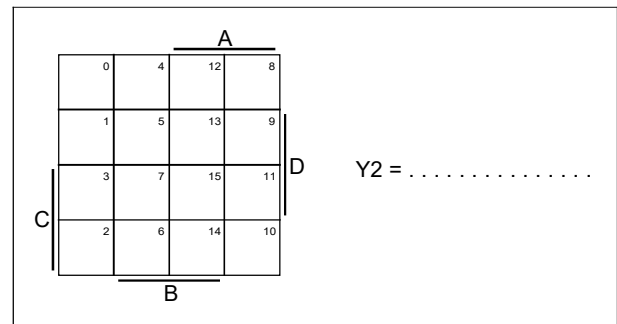


Fig. 1.2.4.2 KV diagram for Y2

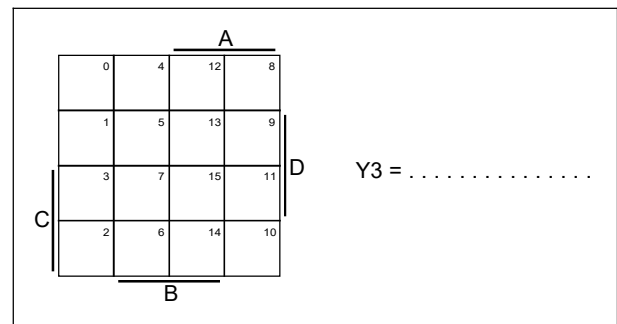


Fig. 1.2.4.3 KV diagram for Y3

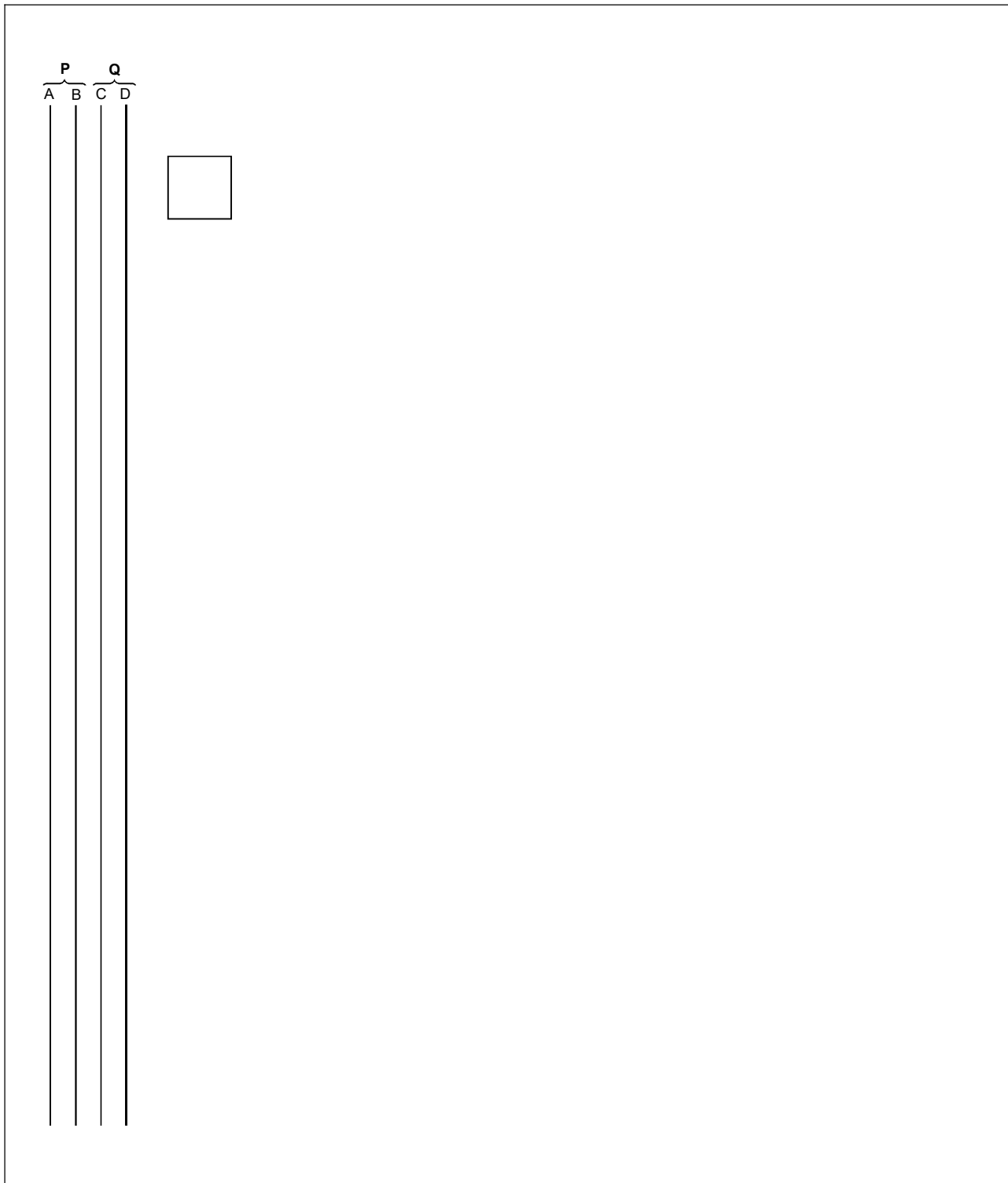


Fig. 1.2.4.4 Circuit of a number comparator for dual numbers

□ Experiment 2: NAND technology

Experiment procedure:

- Convert the circuit in fig. 1.2.5.1 so that it now only consists of NAND gates and complete fig. 1.2.5.2.
- Then do the conversion in fig. 1.2.5.2 by calculation.
- Test the circuit with the Digital Training System.

Q =

\bar{Q} =

$Q||$ =

Question 1: Can NAND gates be saved with the calculation method?

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....



Fig. 1.2.5.2 Circuit in NAND technology

Experiment 3: NOR technology

Experiment procedure:

- Convert the circuit in fig. 1.2.5.1 so that it consists of NOR gates only and complete fig. 1.2.5.3.
- Do the conversion in fig. 1.2.5.3 by calculation.
- Test the circuit with the Digital Training System.

Q =

\bar{Q} =

$\overline{\overline{Q}}$ =

Question 1: Compare experiments 2 and 3. Which method requires the most effort?

Answer:

.....

.....

.....

Question 2: Can NOR gates be saved by the calculation method?

Answer:

.....

.....



Fig. 1.2.5.3 Circuit in NOR technology

1.2.6 Equivalence

□ Experiment 1: Fundamental principles

Examine the circuit for equivalence.

Experiment procedure:

- Draw the symbol for the switching function in fig. 1.2.6.1.
- Complete table 1.2.6.1 with the aid of fig. 1.2.6.1 and specify the disjunctive normal form.

Q =



Fig. 1.2.6.1 Circuit symbol

Table 1.2.6.1 Value table

□ Experiment 2: 1-bit number comparator

Design the circuit of a 1-bit number comparator in which a greater-smaller comparison is also carried out in addition to equality of two 1-digit dual numbers P and Q.

Experiment procedure:

- Complete table 1.2.6.2 and determine the disjunctive normal forms for $P > Q$, $P = Q$ and $P < Q$.
- Draw the circuit in fig. 1.2.6.2 and check its function with the Digital Training System.

$P > Q$:

$P = Q$:

$P < Q$:

P			P = Q	

Table 1.2.6.2 Value table

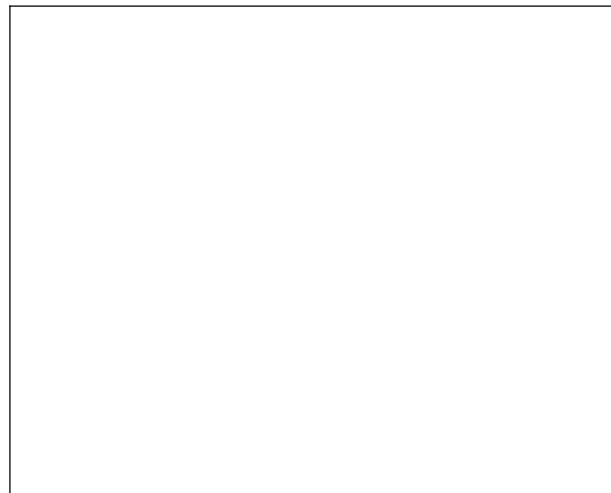


Fig. 1.2.6.2 Circuit

1.2.7 Antivalence

□ Experiment 1: Fundamental principles

Experiment procedure:

- Complete the value table (table 1.2.7.1) for the circuit shown in fig. 1.2.7.1.
- Draw the circuit in fig. 1.2.7.1 as a circuit symbol (fig. 1.2.7.2) and convert it to the disjunctive normal form using the laws of switching algebra.

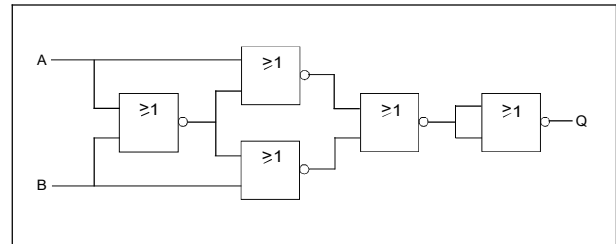


Fig. 1.2.7.1

Q =

 Q =

Table 1.2.7.1 Value table



Fig. 1.2.7.2 Circuit symbol

Question 1: The logic of our circuit corresponds to a lamp circuit familiar from installation technology. Name it and draw the connection diagram as confirmation (fig. 1.2.7.3).

Answer:

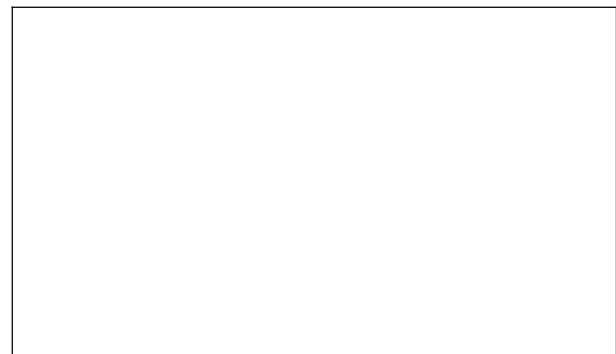


Fig. 1.2.7.3 Connection diagram

1.2.8 Working with TTL Components

□ Experiment 1: AND/NAND gates with pull-up resistor

TTL means Transistor-Transistor Logic. In this circuit family, the logic operations are generated exclusively by bipolar transistor systems.

Fig. 1.2.8.1 shows the principle structure of a NAND gate of the standard TTL series 74.

The circuit consists of a multi-emitter transistor V1 which carries out an AND linkage of the input signals. The series-connected inverter, consisting of the transistors V2, V3 and V4 reverses the input signal.

- **Note:** The NAND or AND gate is already used in chapter 1.2.4.

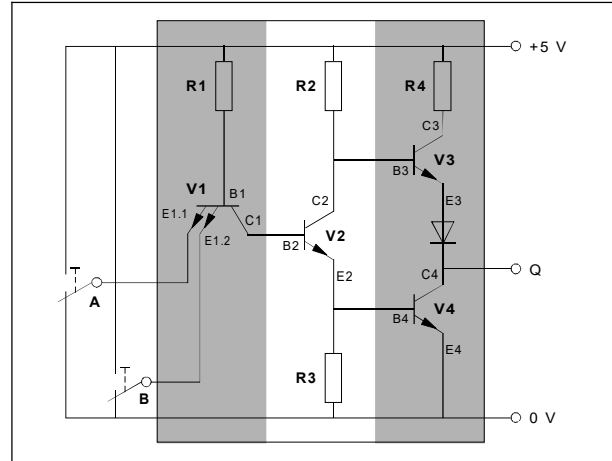


Fig. 1.2.8.1 NAND gate in TTL technology

Experiment procedure:

- Describe the function of the circuit in fig. 1.2.8.1 by completing table 1.2.8.1.
- **Note:** B - E: base-emitter line, B - C: base-collector line
- Then check the AND/NAND gate with the Digital Training System by completing table 1.2.8.2.
- **Note:** Use the connection through the pull-up resistor for the H-level at input C. If the 2 mm connecting plug is not plugged in, the inputs are not connected and faults may occur because there is no defined state.

Inputs		Transistor V1		V2	V3	V4	Outp. Q
A	B	B - E	B - C				
L	L						
L	H						
H	L						
H	H						

Table 1.2.8.1

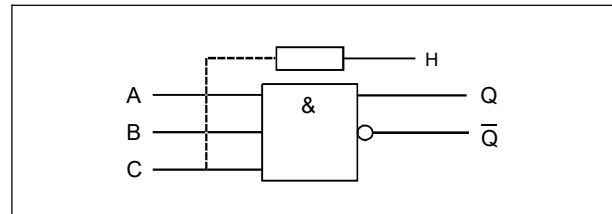


Fig. 1.2.8.2 NAND gate with pull-up resistor

N. B.:

Unconnected inputs of TTL gates behave in the same way as inputs connected to H level.

Inputs			Q for AND	Q-bar for NAND
A	B	C		
L	L	open		
L	L	H		
H	H	open		
H	H	H		

Table 1.2.8.2

□ **Experiment 2: OR/NOR gates with pull-down resistor**

Experiment procedure:

- Check the OR/NOR gate (fig. 1.2.8.3) with the Digital Training System.
- Complete table 1.2.8.3.
- **Note:** Use the connection through the pull-down resistor for the L level at input C.

Inputs			Q for OR	\bar{Q} for NOR
A	B	C		
L	L	open		
L	L	L		
H	H	open		
H	H	L		

Table 1.2.8.3

N. B:

Unused inputs of an AND or NAND gate are effective because they already have H level. In practice it is usual to **apply unused inputs** of a gate to **H level**. This can only be achieved in an OR or NOR gate, however, if the unused inputs are applied to **L level**.

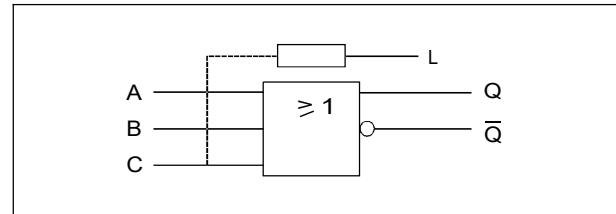


Fig. 1.2.8.3

Notes:

Notes:

2. Schmitt Triggers

2.1 Fundamental Principles

A Schmitt trigger or **threshold switch** is a controllable multivibrator with two stable switching states. The Schmitt trigger forms a squarewave output voltage U_O from a continuously changing input voltage U_I .

Fig. 2.1.1 shows the circuit symbol of a Schmitt trigger. The input of the Schmitt trigger does **not** carry a binary signal.

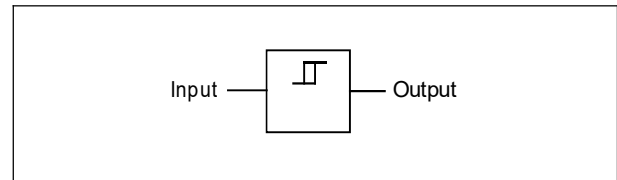


Fig. 2.1.1 Circuit symbol of a Schmitt trigger

The switching state resident at the output of the Schmitt trigger depends on the value of the voltage applied to the control input.

Fig. 2.1.2 shows that the trigger process is activated at the Schmitt trigger input when the **threshold voltage U_{ON}** is exceeded and the output voltage U_{Omax} remains at its maximum value until the input voltage reaches and falls below the value of the **return voltage U_{OFF}** . The difference between these two input voltages is referred to as the **hysteresis U_H** .

$$U_H = U_{ON} - U_{OFF}$$

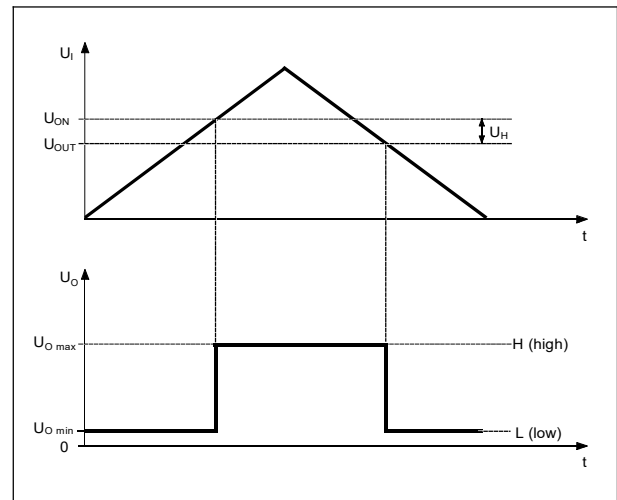


Fig. 2.1.2 Voltages of the Schmitt trigger

2.2 Experiments Section

□ Experiment 1: Inverting Schmitt trigger

Experiment procedure:

- Set up the circuit as shown in fig. 2.2.1.
- Measure the threshold voltage U_{ON} and the return voltage U_{OFF} with an analog voltmeter by turning the potentiometer slowly to the right (U_{ON}) and then slowly to the left (U_{OFF}) and noting the respective last readable voltage value before the jump.
- Determine the hysteresis U_H and then complete the diagram in fig. 2.2.2.

$U_{ON} = \dots\dots\dots$

$U_{OFF} = \dots\dots\dots$

$U_H = \dots\dots\dots$

Question 1: Which logic gate can be used to replace this Schmitt trigger if necessary?

Answer:

Question 2: How does the circuit in fig. 2.2.1 need to be completed in order to produce a Schmitt trigger whose output voltage adopts the upper value on exceeding the on threshold and the lower value on falling below the off threshold?

Answer:

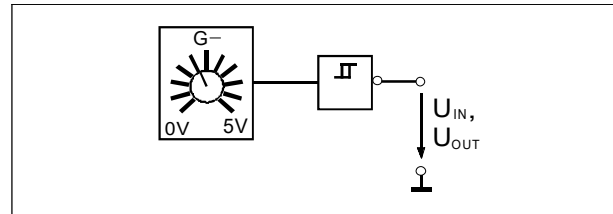


Fig. 2.2.1 Circuit

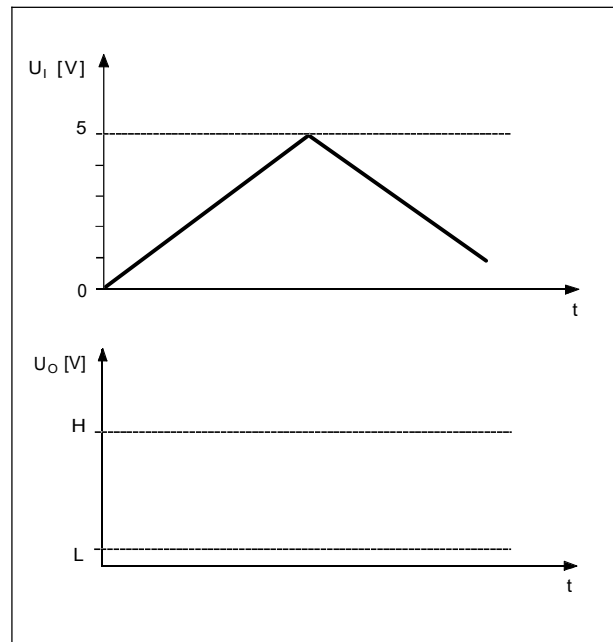


Fig. 2.2.2

3. Bistable Multivibrators

2.1 Fundamental Principles

3.1.1 General

When processing signals, it is frequently necessary to retain the signal states beyond the time that the signals are applied and to put them into effect later. So-called sequential digital circuits have such signal memories in addition to **non-storing** logic elements, e. g. AND, OR, NOT elements.

The value of the output variable of such a circuit therefore depends not only on the values of the input variables but also on the states of the signal memories at the same time.

Signal memories are formed by a bistable multivibrator (flipflop) which can store the information of one bit.

The basic component for all flipflops in digital technology is the **RS flipflop**. The number of inputs may be different but every flipflop has exactly two outputs which carry complementary (opposite) levels. Equal output states may not occur by way of definition even if this is partially possible.

3.1.2 Asynchronous Flipflops

In the circuit symbol the inputs are identified according to their function by **Set** or **S** and **Reset** or **R**. The complementary output levels are represented with the inversion of the Q_2 output.

Fig. 3.1.2.1 shows the circuit symbol of an RS flipflop with non-inverting inputs.

- State „1" at S switches the flipflop to $Q_1 = 1$ (set)
- State „1" at R switches the flipflop to $Q_2 = 1$ (reset)
- The „0" states have no controlling effect here.

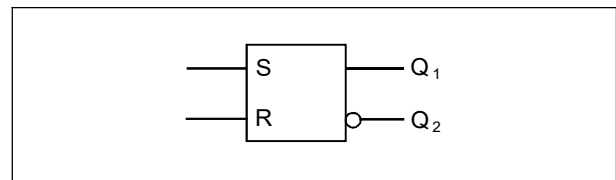


Fig. 3.1.2.1 Circuit symbol of an RS flipflop

RS flipflops which are controlled by 0 states have special inputs characterized by a negation circuit (fig. 3.1.2.2).

- State „0" at S switches the flipflop to $Q_1 = 1$ (set).
- State „0" at R switches the flipflop to $Q_2 = 1$ (reset).
- The „1" states have no controlling effect here.

Flipflops usually have a fixed basic setting. After applying the supply voltage the flipflop is in the so-called **quiescent state** with $Q_1 = 0$ and $Q_2 = 1$.

Flipflops which adopt the operating state $Q_1 = 1$ and $Q_2 = 0$ after switching on the supply voltage are shown in fig. 3.1.2.3.

- The flipflop is set after switching on.

If a flipflop does **not** lose the stored information in the event of a loss of voltage, **NV** (non-volatile) is entered in the circuit symbol (fig. 3.1.2.4).

- The flipflop has the same logic state it had when switching off after switching on the supply voltage.

A considerable disadvantage of the RS flipflop is its irregular state in the case of High level at S and R. In most flipflops, this state combination is prohibited because equal output levels $Q_1 = Q_2 = 0$ occur. By means of additional circuit measures (chapter 3.2.4, experiments 1 and 2, page 34) the flipflop can be set or reset **dominant** (with priority) instead (fig. 3.1.2.5 and 3.1.2.6).

- Dominant S input: The flipflop is set when $S = R = 1$
- Dominant R input: The flipflop is reset when $S = R = 1$

The flipflops dealt with above are referred to as **asynchronous flipflops** because they operate **without a clock signal** and can be set at any time by input signals.

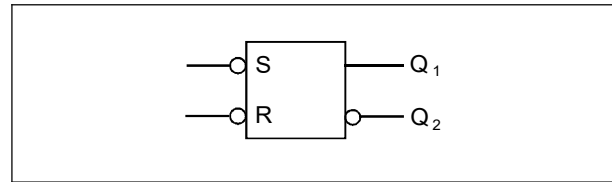


Fig. 3.1.2.2 RS flipflop with inverting inputs

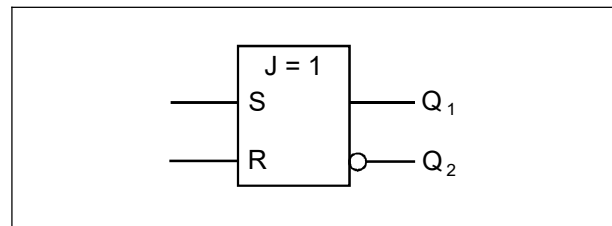


Fig. 3.1.2.3

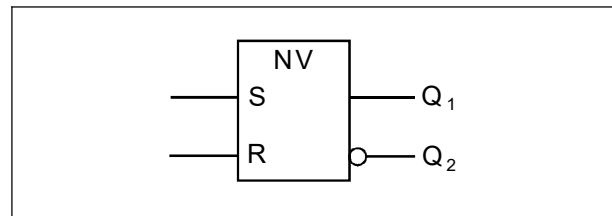


Fig. 3.1.2.4 NV-RS flipflop

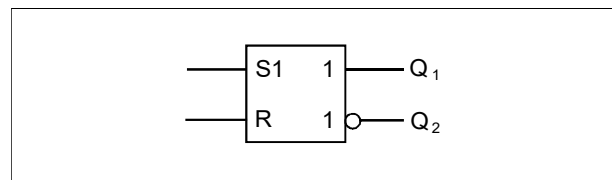


Fig. 3.1.2.5 RS-FF with dominant S-input

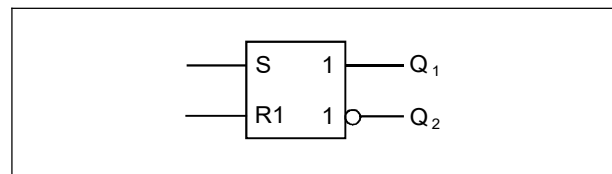


Fig. 3.1.2.6 RS-FF with dominant R input

3.1.3 Synchronous Flipflops

Flipflops which are controlled by a clock are known as synchronous flipflops. They have an input known as the clock input in addition to the set and reset inputs.

Fig. 3.1.3.1 shows the circuit symbol for a **one state controlled RS flipflop**.

- The set and reset inputs are only effective if there is a 1-signal at the clock input **C1**.

Since this flipflop with its static clock input can be set or reset unintentional by an interference pulse throughout the whole clock pulse time, it is better to use a **dynamic** clock input.

A flipflop with a dynamic clock input is known as a **single edge controlled RS flipflop** (fig. 3.1.3.2 and 3.1.3.3).

- The flipflop in fig. 3.1.3.2 can only be set or reset with a rising (positive) clock edge.
- The flipflop in fig. 3.1.3.3 can only be set or reset with a falling (negative) clock edge.

In some application circuits with flipflops such as counters and shift registers, there must be a difference in time between the signal input and the signal output. Flipflops with **signal buffering** are therefore used. The buffer may act **statically** (fig. 3.1.3.4) or **dynamically** (fig. 3.1.3.5).

- The input information is recorded during the one clock state and output until the following clock state (fig. 3.1.3.4).
- The input information is read in during the one clock edge and only passed on with the following edge (fig. 3.1.3.5).

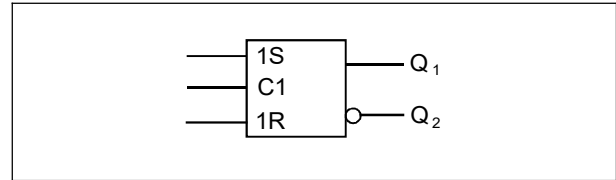


Fig. 3.1.3.1 One state controlled RS flipflop

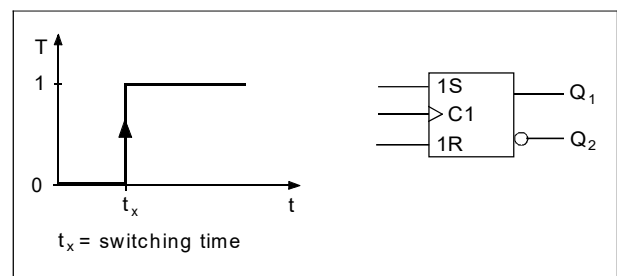


Fig. 3.1.3.2 Single edge controlled RS flipflop

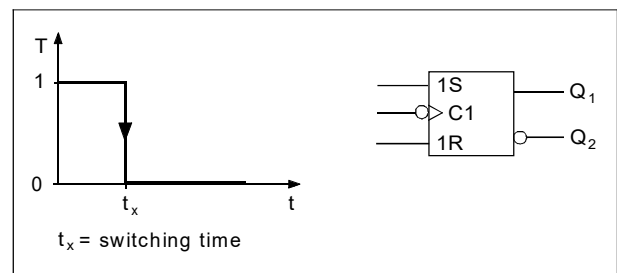


Fig. 3.1.3.3

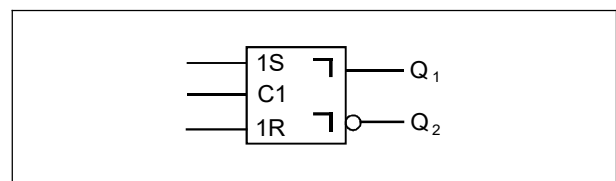


Fig. 3.1.3.4 Two state controlled RS flipflop

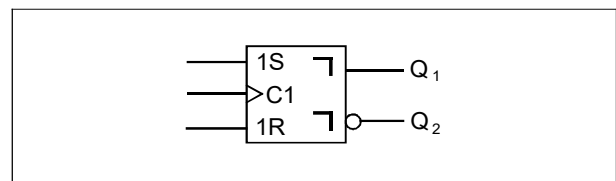



Fig. 3.1.3.5 Two edge controlled RS flipflop

Flipflop outputs at which the input information appears with a delay are referred to as **retarded outputs** and identified in the circuit symbol by .

With a suitable external wiring, the disadvantage common to all RS flipflops of equal output levels at a certain input level combination can be prevented. This results in flipflops as shown in fig. 3.1.3.6 ... 3.1.3.8 independently of the clock control used.

- The T flipflop triggers at every controlling clock edge.
- The D flipflop stores the state at D with the controlling edge and passes it on to Q_1 .
- The JK flipflop records the input signal with rising clock edge and outputs it at Q_1 with the falling clock edge.

The switching behaviour of a flipflop can be described with the aid of a working table.

Since flipflops are very often used in sequential (time-dependent) circuits, however, their behaviour is usually represented in the form of timing diagrams.

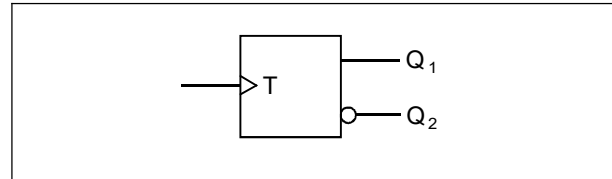


Fig. 3.1.3.6 T flipflop

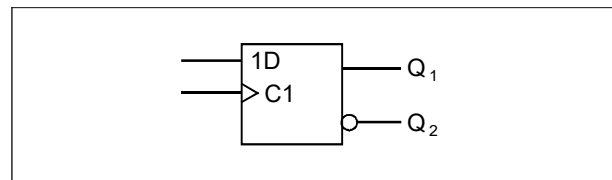


Fig. 3.1.3.7 D flipflop

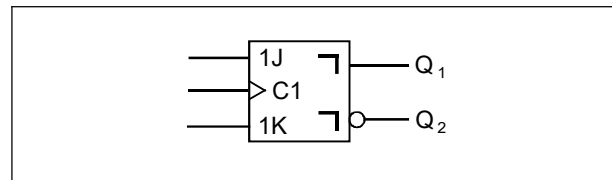


Fig. 3.1.3.8 JK flipflop

3.2 Experiments Section

3.2.1 RS Flipflop consisting of NOR Gates

Experiment 1: Fundamental principles

Examine the RS flipflop in fig. 3.2.1.1.

Experiment procedure:

- Set up the circuit shown in fig. 3.2.1.1.
- Apply the values specified in table 3.2.1.1 one after the other to the inputs S and R of the circuit and fill in the missing output values for Q₁ and Q₂.
- Explain the behaviour of the flipflop in table 3.2.1.1 with the terms **set**, **reset**, **store**, **not defined** (stored output state is random if S and R change from H to L at the same time) and **irregular** (Q₁ and Q₂ have no opposite levels).

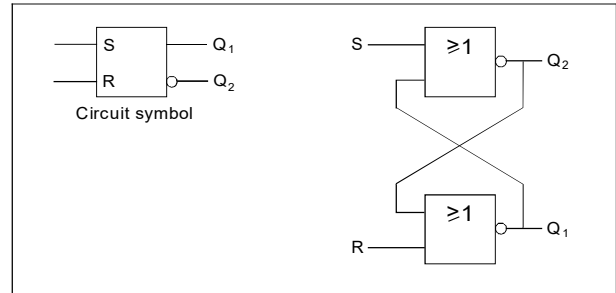


Fig. 3.2.1.1 Circuit

S	R	Q ₁	Q ₂	Explanation
0	0	0	1	Store
1	0	1	0	Set
0	0	1	0	Store
0	1			
0	0			
1	1			
1 → 0	1 → 0	0	1	
		1	0	

Table 3.2.1.1 Value table

Experiment 2: Simplified form

In practice, only a simplified form is normally used instead of the detailed value table.

At the time t_n , the values are applied to the inputs of the binary memory. The time t_{n+1} follows the time t_n . The state of the flipflop at time t_n is therefore described by q_n .

The flipflop may be in the set or reset state at the time t_n . The state of the flipflop complementary to q_n at the time t_n is \bar{q}_n .

t_n		t_{n+1}	
S	R	Q ₁	Q ₂
0	0	q_n	\bar{q}_n
0	1		
1	0		
1	1		
1 → 0	1 → 0		

Table 3.2.1.2

Experiment procedure:

- Complete the table 3.2.1.2.

3.2.2 RS Flipflop consisting of NAND Gates

Experiment 1: Fundamental principles

Examine the RS flipflop in fig. 3.2.2.1.

Experiment procedure:

- Set up the circuit shown in fig. 3.2.2.1.
- Complete the table 3.2.2.1.

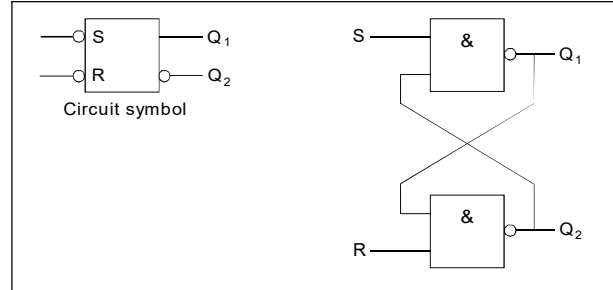


Fig. 3.2.2.1 Circuit

t_n		t_{n+1}		Explanation
S	R	Q ₁	Q ₂	
0	0			
0	1			
1	0			
1	1			
0 → 1	0 → 1			

Table 3.2.2.1

Experiment 2: SR memory flipflop

If a NOT gate is connected before every input of the circuit in fig. 3.2.2.1, an SR memory flipflop is obtained. Characterize the behaviour of the SR memory flipflop.

Experiment procedure:

- Set up the circuit as shown in fig. 3.2.2.1 and insert a NOT gate in front of the inputs S and R.
- Complete the table 3.2.2.2.

N. B:

Since the SR memory flipflop behaves like an **RS flip-flop consisting of NOR gates** when setting and resetting, it also has the same circuit symbol.

				Explanation
→	→			

Table 3.2.2.2

3.2.3 Clock State Controlled RS Flipflops

□ **Experiment 1: One state controlled RS flipflop**

In many cases it is undesirable for the output state of the flipflop to change even only a few nanoseconds after the input state changes. Clock state controlled RS flipflops were developed for this reason.

Experiment procedure:

- Set up the circuit as shown in fig. 3.2.3.1. The clock is treated here as a third binary input.
- Complete the table 3.2.3.1 and the pulse diagram in fig. 3.2.3.2.

N. B.:

The pulse diagram in fig. 3.2.3.2 shows the disadvantage of one state controlled clock inputs. After the clock, the state of the flipflop corresponds to the **last** information at the end of the clock. The probability of errors as a result of this can be kept low if a **small** enough clock duration is selected.

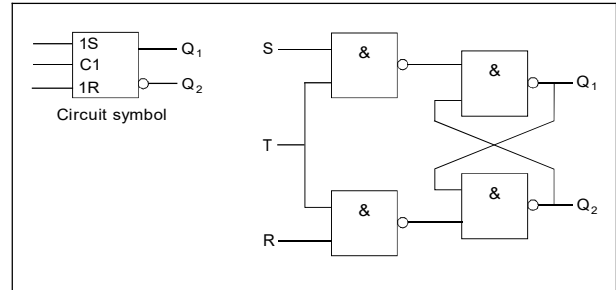


Fig. 3.2.3.1 Circuit

Clock T	S	R	Q ₁	Q ₂	Explanation
0	0	0			
0	1	0			
0	0	1			
0	1	1			
1	0	0			
1	1	0			
1	0	1			
1	1	1			

Table 3.2.3.1

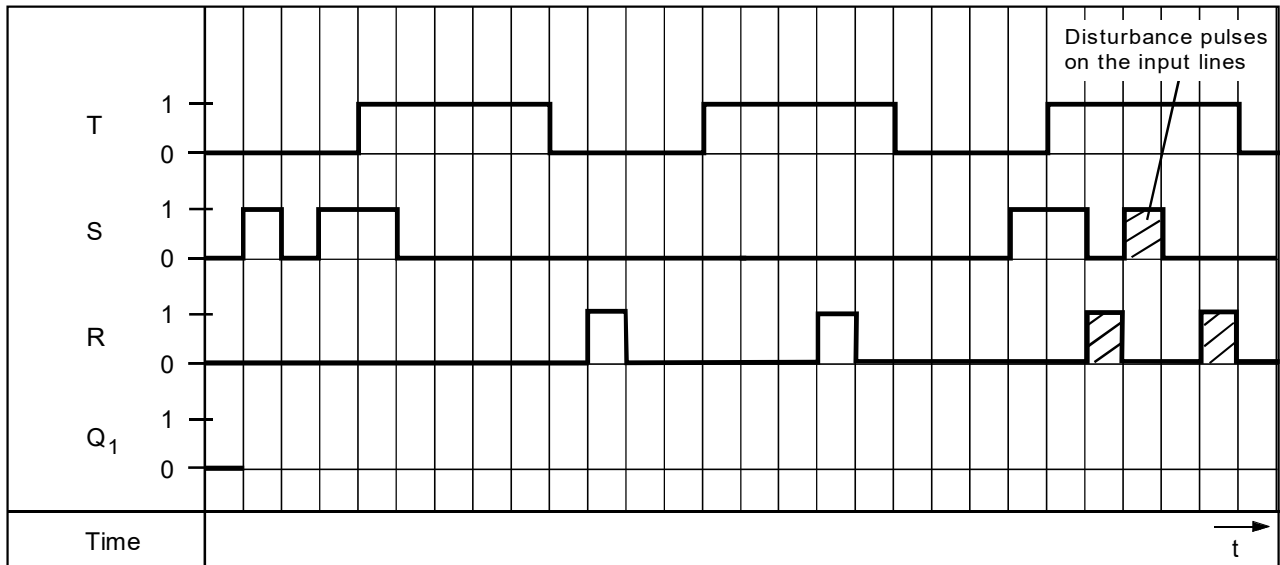


Fig. 3.2.3.2 Pulse diagram of the one state controlled RS flipflop

3.2.4 RS Flipflops with Dominant S or R Input

□ Experiment 1: RS flipflop with dominant S input

The irregular case of the RS flipflop prompts the following consideration: Could you not construct a flipflop which always sets Q_1 to 1 at $S = 1$ and $R = 1$? Design an input wiring which makes this possible.

The circuit symbol (fig. 3.2.4.1) and the value table (table 3.2.4.1) show an RS flipflop which fulfills this condition.

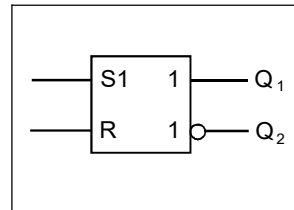


Fig. 3.2.4.1

S	R	Q_1	Q_2
0	0	q_n	\bar{q}_n
0	1	0	1
1	0	1	0
1	1	1	0

Table 3.2.4.1

Experiment procedure:

- Complete fig. 3.2.4.2 with an additional wiring.
- Then check the circuit with the Digital Training System.
- **Note:** Use the outputs \bar{Q} of the input keyboard for inputs R and S because the JK flipflop of the Digital Training System has negated inputs.

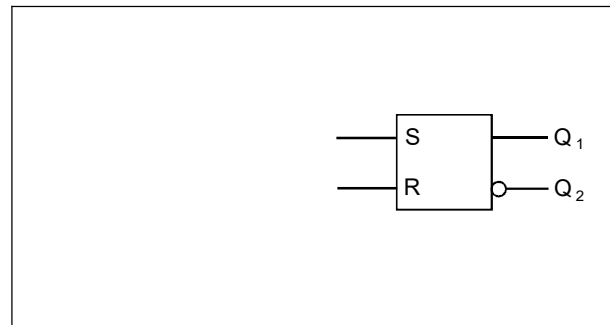


Fig. 3.2.4.2 Circuit

□ Experiment 2: RS flipflop with dominant R input

An RS flipflop which must be reset at $S = 0$ and $R = 0$ ($Q_1 = 0$) would also be conceivable. Design the appropriate input wiring.

The circuit symbol (fig. 3.2.4.3) and the value table (table 3.2.4.2) show an RS flipflop which fulfills this condition.

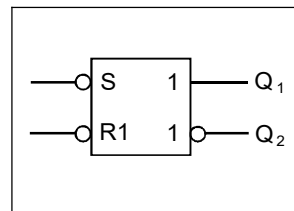


Fig. 3.2.4.3

S	R	Q_1	Q_2
0	0	0	1
0	1	1	0
1	0	0	1
1	1	q_n	\bar{q}_n

Table 3.2.4.2

Experiment procedure:

- Complete fig. 3.2.4.4 with an additional wiring.
- Then check the circuit with the Digital Training System.

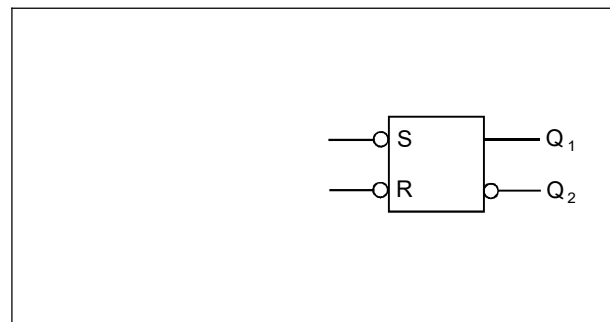


Fig. 3.2.4.4 Circuit

3.2.5 D Flipflops

□ Experiment 1: Fundamental principles

Flipflops with one state controlled clock inputs are often used as binary memory components. They are then usually designed as D flipflops.

Examine the D flipflop in fig. 3.2.5.1.

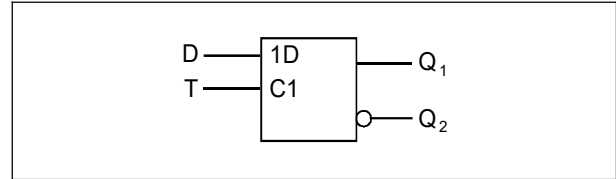


Fig. 3.2.5.1 Circuit symbol

Experiment procedure:

- Set up the circuit as shown in fig. 3.2.5.1.
- Examine the function by completing the value table (table 3.2.5.1).

Clock T	D	Q ₁	Q ₂
1	0		
1	1		

Table 3.2.5.1

□ Experiment 2:

As far as its function is concerned, the D flipflop with state controlled clock input can be considered as a corresponding RS flipflop in which only permissible switching states exist.

Set up this circuit. Only NAND gates are available.

Experiment procedure:

- Complete the circuit in fig. 3.2.5.2 and then enter the levels for transferring a „0" or a "1" into the circuit.
- Check the circuit with the Digital Training System.

N. B.:

This flipflop only has permissible switching states because only a NAND element can have 0 state at the input. This guarantees that the state required for switching can only occur at either D4 or D5.

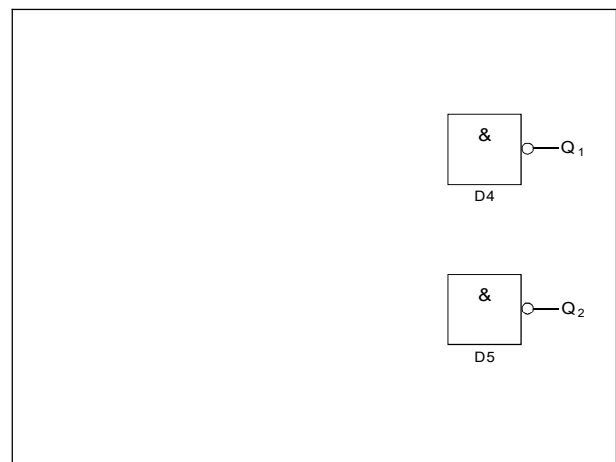


Fig. 3.2.5.2 Circuit

3.2.6 Single Edge Controlled RS Flipflop

□ Experiment 1: Fundamental principles

The susceptibility to disturbance of the clock state controlled RS flipflop (see chapter 3.2.3, page 35) can be further reduced if setting or resetting of a flipflop is only possible during the edge of the clock pulse.

Technically the edge control can be realized with a CR circuit and a diode for inhibiting the negative or positive needle pulses (differentiator).

Fig. 3.2.6.1 shows the circuit symbol and fig. 3.2.6.2 the circuit for the principle of a singled edge controlled RS flipflop without using a differentiator.

Experiment procedure:

- Set up the circuit as shown in fig. 3.2.6.2.
- Check the function of the circuit with the Digital Training System and complete the pulse diagram in fig. 3.2.6.3.

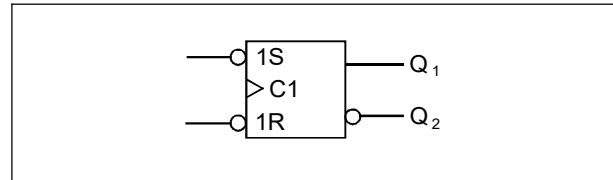


Fig. 3.2.6.1 Circuit symbol

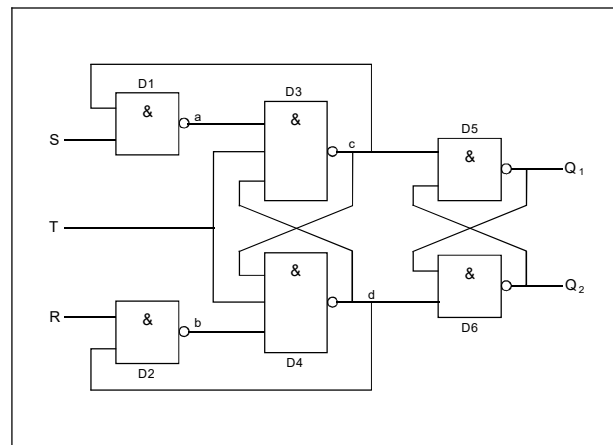


Fig. 3.2.6.2 Circuit

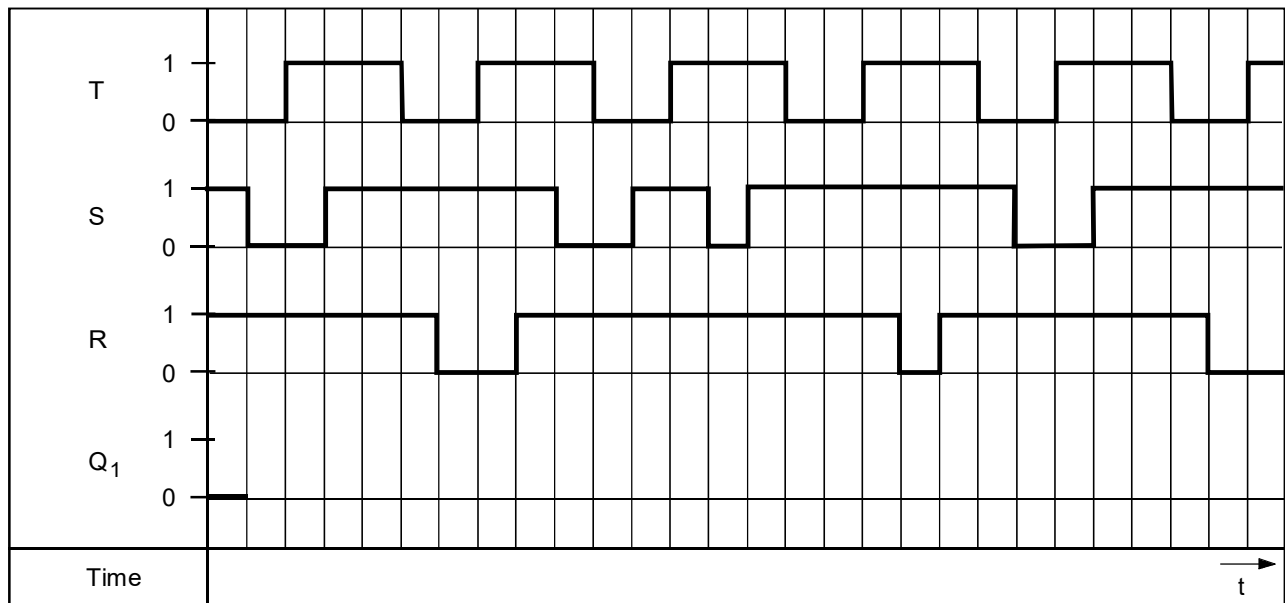


Fig. 3.2.6.3 Pulse diagram of the single edge controlled RS flipflop

Question 1: The pulse diagram in fig. 3.2.6.3 shows that the flipflop can also change its output state during the clock, i. e. no obvious inhibiting effect exists. Test this with the circuit in fig. 3.2.6.2. Connect outputs a, b, c and d to LEDs for this. Describe what happens.

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Question 2: The clock edge control therefore only works if the buffer (D3 and D4) inhibits its own inputs immediately after accepting the signal. Which levels would therefore have to be applied to outputs c and d for the triggering edge?

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Notes:

Experiment 2: Clock edge controlled D flipflop

Now convert your RS multivibrator in fig. 3.2.6.2 into a D multivibrator and examine the inhibiting effect. This produces a clock edge controlled D flipflop (fig. 3.2.6.4).

Experiment procedure:

- Formulate the necessary circuit changes. Try to manage without an extra gate.
- Then check the circuit with the Digital Training System.
- Complete the pulse diagram in fig. 3.2.6.5.

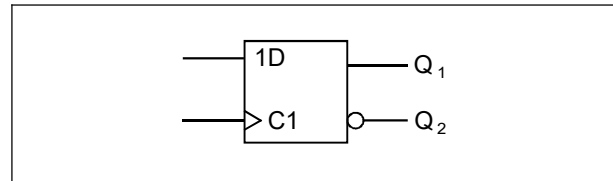


Fig. 3.2.6.4 Circuit symbol

Question 1: Observe the inputs of the buffer (a and b) at T = 0 in the pulse diagram and compare these with your statement in experiment 1, question 1.

Answer:

.....

.....

.....

Circuit change:

.....

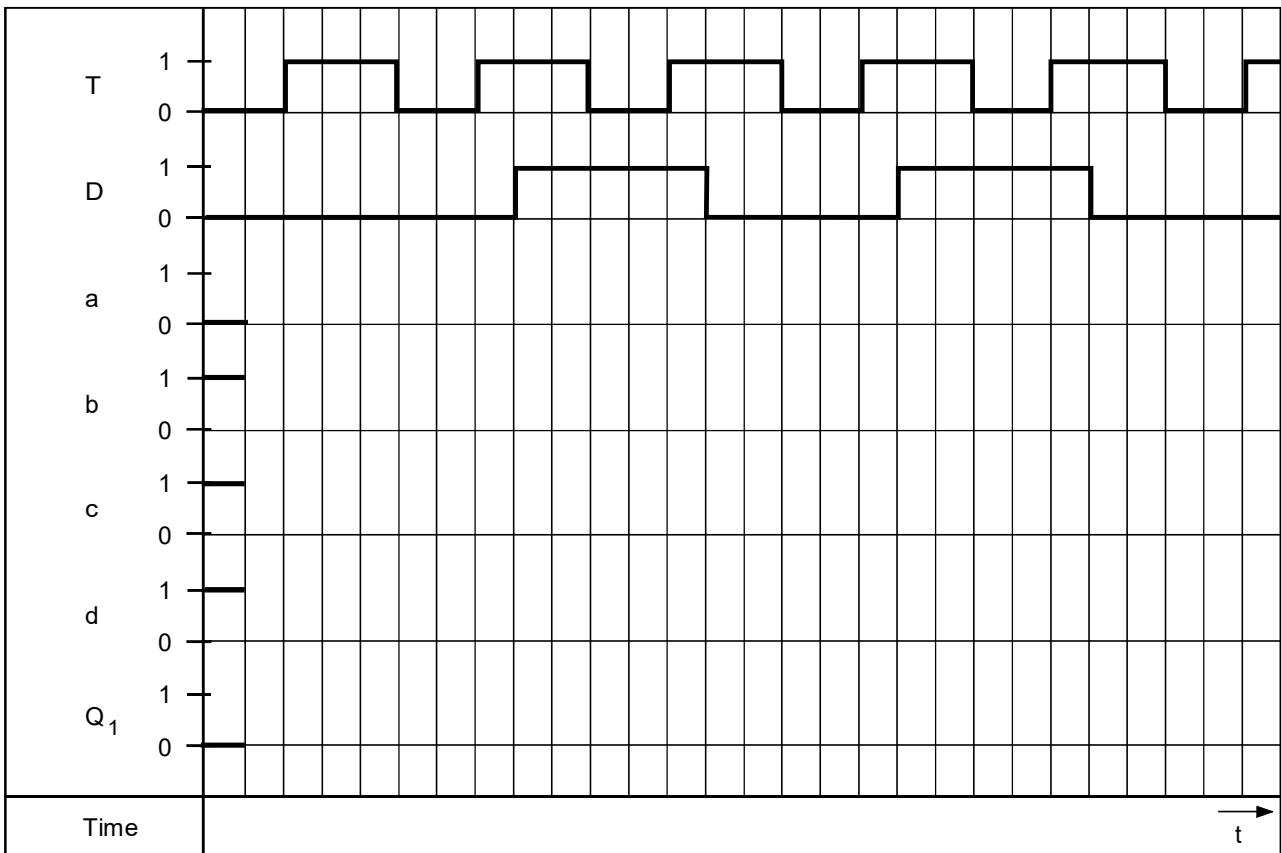


Fig. 3.2.6.5 Pulse diagram of the clock edge controlled D flipflop

3.2.7 Two State Controlled D Flipflop

□ **Experiment 1: Master-slave control**

The D-flipflop is suitable as a basic element for read and write memories (RAM).

If a **simultaneous** evaluation of the output states (read) and new reading in (write) is to be enabled, the flipflop must be equipped with a master-slave control. The circuit symbol in fig. 3.2.7.1 and the reserve circuit diagram in fig. 3.2.7.2 should explain this. Use the inverter on the DIGI BOARD 2 and connect it to high level. If the circuit is in undefined state you have to realize the original state according to Fig. 3.2.7.3.

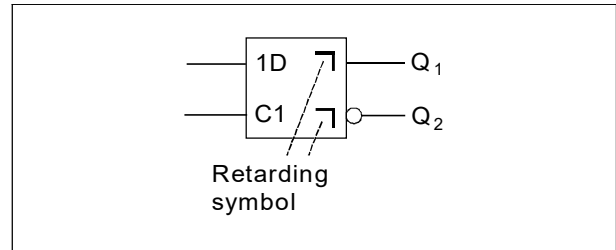


Fig. 3.2.7.1 Circuit symbol

Experiment procedure:

- Set up the circuit as shown in fig. 3.2.7.2 and test it th the Digital Training System.
- Complete the pulse diagram in fig. 3.2.7.3.

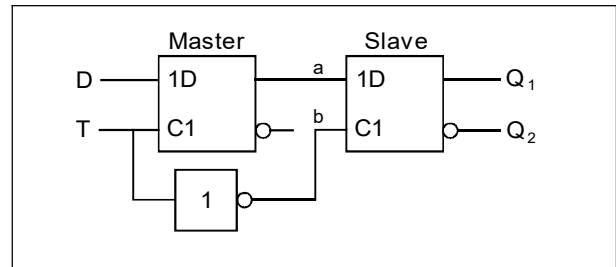


Fig. 3.2.7.2 Reserve circuit diagram

N. B.:

The two-state controlled D flipflop passes on the information at the **end of the clock pulse** from the data input to the output Q_1 **until the clock pause starts**.

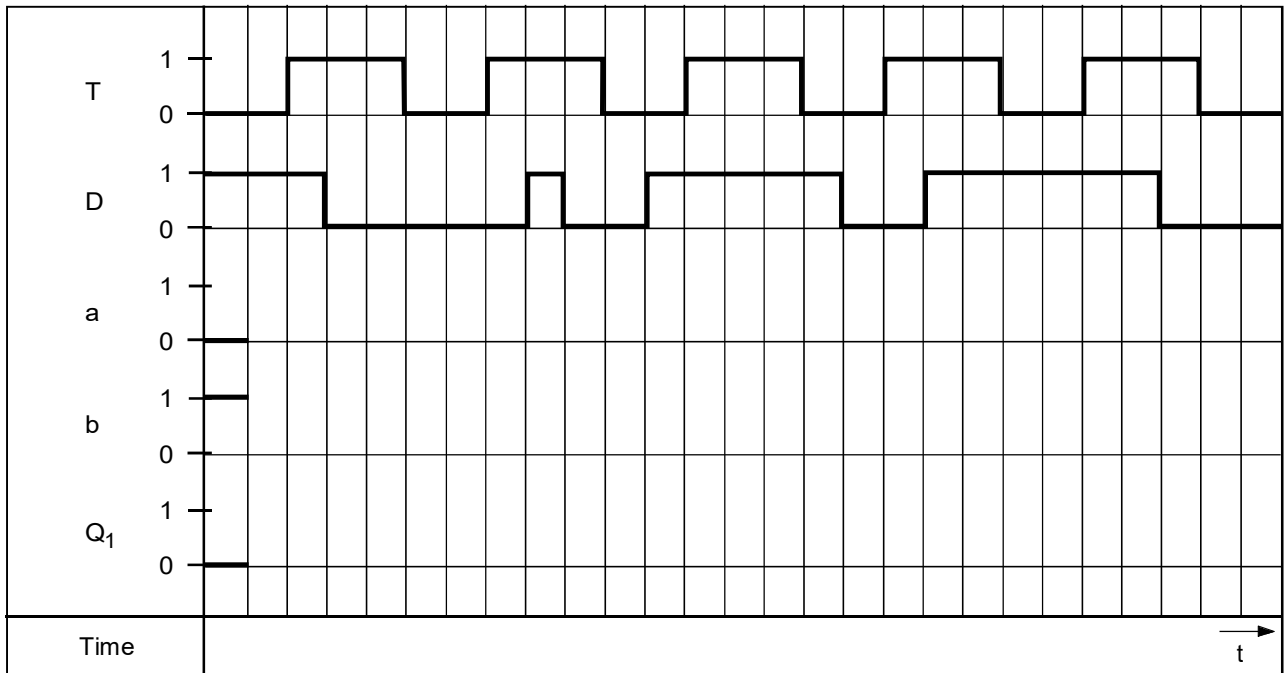


Fig. 3.2.7.3 Pulse diagram of the two state controlled D flipflop

3.2.8 Single Edge Controlled JK Flipflop

The JK flipflop also avoids the **non-defined** output state of the RS flipflop. It can be converted into other flipflops by external wiring measures.

□ Experiment 1: Static set and reset input

Experiment procedure:

- Determine the reserve flipflop for the wiring in fig. 3.2.8.1 by completing the table 3.2.8.1.
- Complete the reserve flipflop in fig. 3.2.8.2.

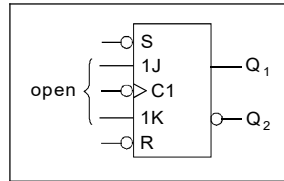


Fig. 3.2.8.1

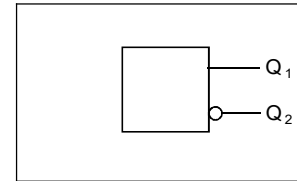


Fig. 3.2.8.2

S	R	Q ₁	Q ₂	Explanation
0	1			
1	1			
1	0			
1	1			
0	0			
0 → 1				

Table 3.2.8.1 Value table

□ Experiment 2: Dynamic J and K input

Experiment procedure:

- Determine the reserve flipflop for the wiring in fig. 3.2.8.3 by completing the table 3.2.8.2.
- Complete the reserve flipflop in fig. 3.2.8.4.

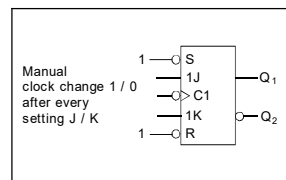


Fig. 3.2.8.3

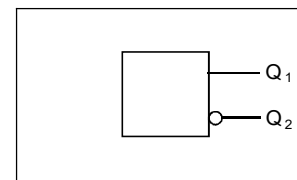


Fig. 3.2.8.4

Clock	J	K	Q ₁	Q ₂	Explanation
1 → 0	1	0			
1 → 0	0	0			
1 → 0	0	1			
1 → 0	1	1			
1 → 0	1	1			
1 → 0	1	0			
1 → 0	0	0			

Table 3.2.8.2 Value table

Experiment 3: Clock input

The JK flipflop exhibits a special behavioural feature caused by parallel circuiting of the JK inputs which are then permanently applied to the operating voltage.

Experiment procedure:

- Determine the reserve flipflop for the wiring in fig. 3.2.8.5 by completing the table 3.2.8.3.
- Complete the reserve flipflop in fig. 3.2.8.6.
- Then complete the pulse diagram in fig. 3.2.8.7, enter the period T_{off} and state the formula relationship.

T_{off} = with $T_{off} = 1/f_{off}$ applies:

f_{off} =

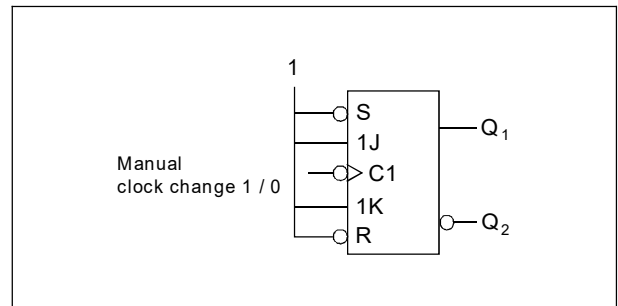


Fig. 3.2.8.5 Wiring

Clock T	Q ₁	Q ₂	Explanation
1 → 0			
1 → 0			
1 → 0			
1 → 0			

Table 3.2.8.3 Value table

Question 1: In which digital circuits could this relationship be significant?

Answer:

.....

.....

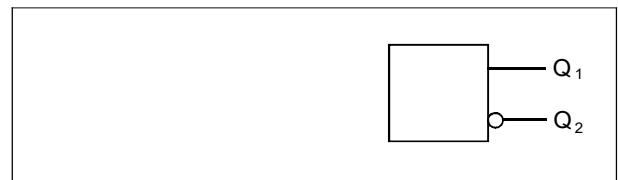


Fig. 3.2.8.6 Reserve flipflop

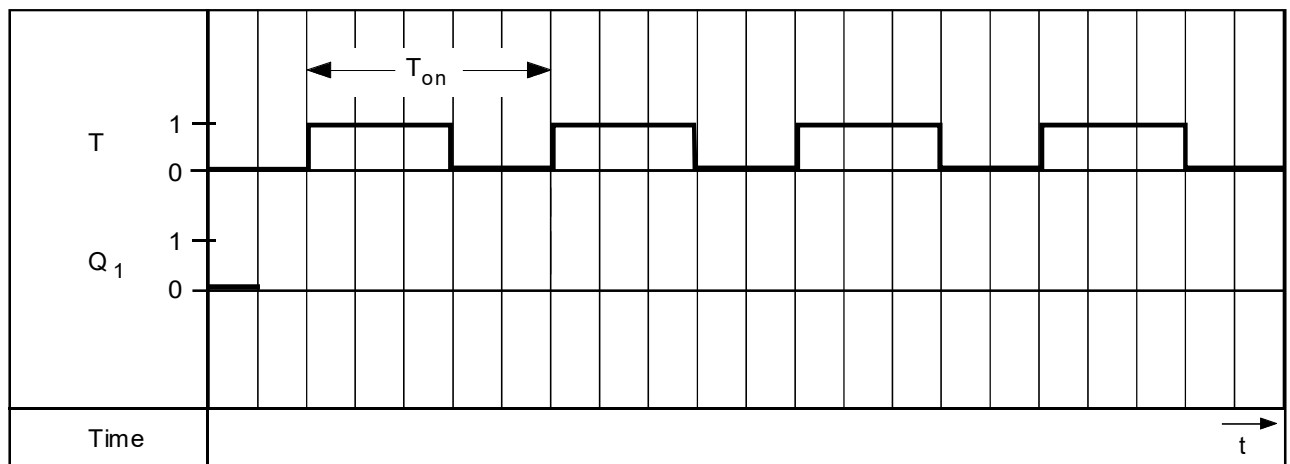


Fig. 3.2.8.7 Pulse diagram

❑ Experiment 4: Wired JK inputs

Experiment procedure:

- Determine the reserve flipflop for the wiring in fig. 3.2.8.8 by completing table 3.2.8.4.
- Complete the reserve flipflop in fig. 3.2.8.9.

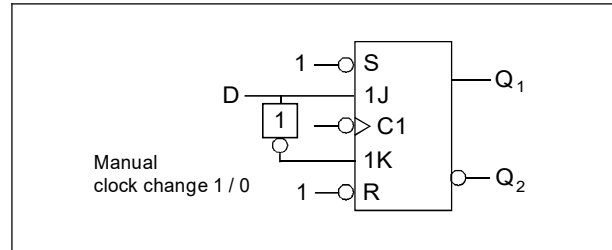


Fig. 3.2.8.8 Wiring

Clock T	D	Q ₁	Q ₂
1 → 0	0		
1 → 0	1		
1 → 0	0		

Table 3.2.8.4 Value table

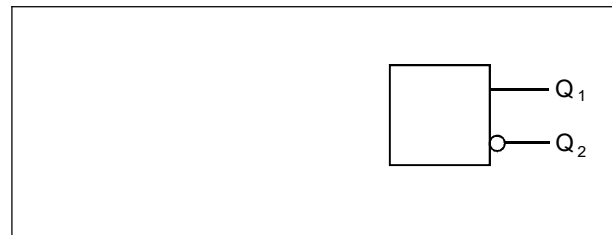


Fig. 3.2.8.9 Reserve flipflop

Notes:

❑ Experiment 5: Two edge controlled JK flipflop

Experiment procedure:

- Complete the circuit in fig. 3.2.8.10 to form a two edge controlled (master-slave principle) JK flipflop.
- Check the circuit in fig. 3.2.8.10 with the Digital Training System and complete the pulse diagram in fig. 3.2.8.11.
- Then complete the reserve flipflop in fig. 3.2.8.12.

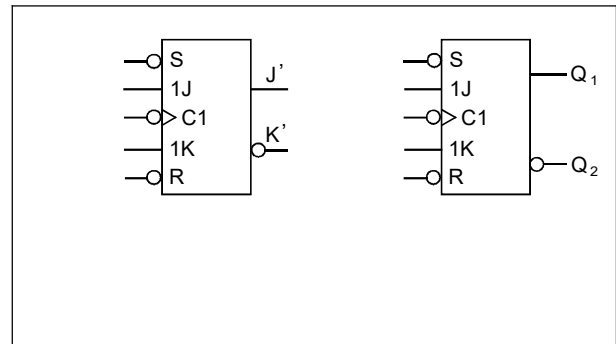


Fig. 3.2.8.10 Two edge controlled JK flipflop



Fig. 3.2.8.12 Reserve flipflop

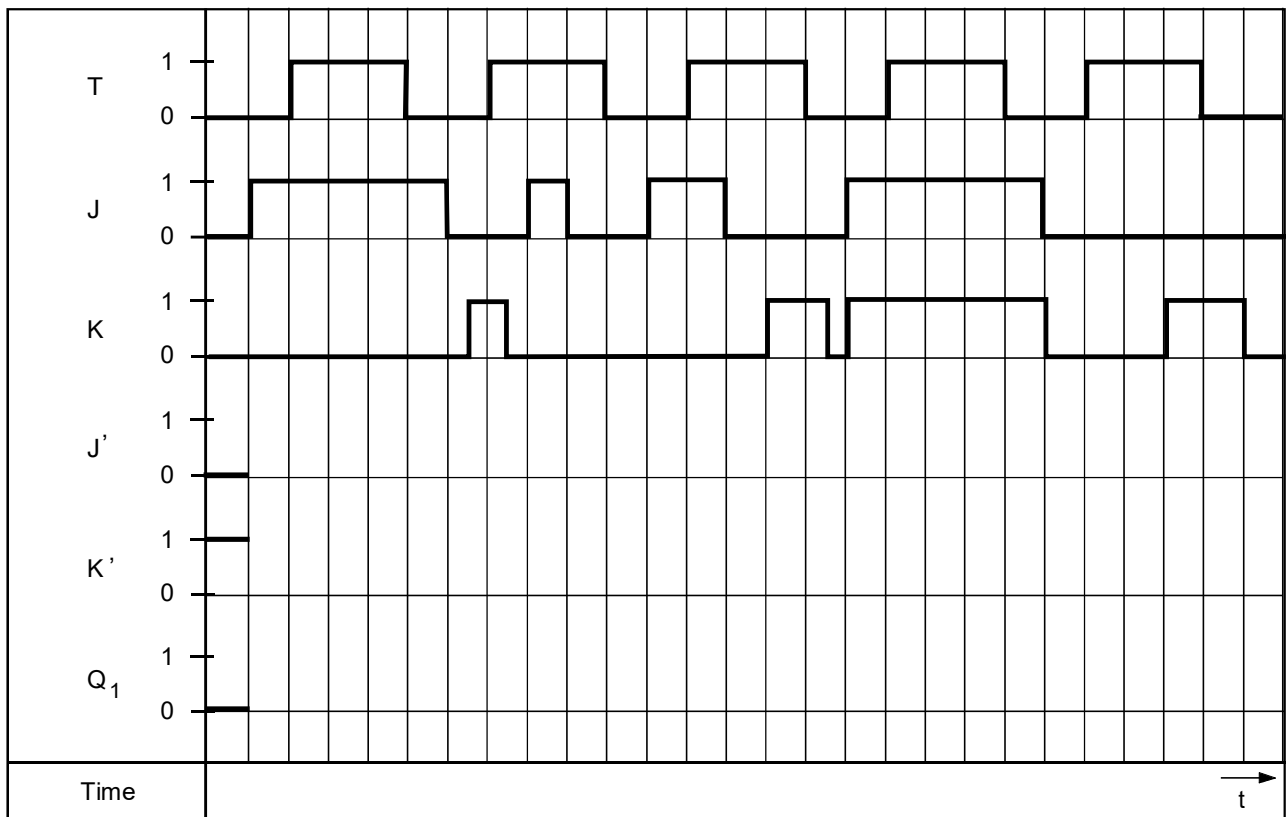


Fig. 3.2.8.11 Pulse diagram of the two edge controlled JK flipflop

Notes:

4. Monostable Multivibrators

4.1 Fundamental Principles

Monostable multivibrators or **monoflops** go into an instable operating state when triggered and return to their stable state automatically after a time which can be determined by the circuit.

A monoflop is in the stable quiescent state when its Q_1 output supplies a signal with the value „0“. If it is switched to its instable state by an H-level at the input - there is also H-level at the Q_1 output in this case - it returns to its stable state after the **dwel time t_Q** .

Monostable multivibrators are used to generate pulses of a certain length. They can store the information content of a bit until returning to the quiescent state.

A distinction is made between **non-retriggerable monoflops** (fig. 4.1.1) and **retriggerable monoflops** (fig. 4.1.2) which are triggered either by state or clock edge control.

The non-retriggerable monoflop is independent of the length and the distance between the trigger pulses for the duration of the operating state.

In the retriggerable monoflop on the other hand every setting pulse edge leads to a new start of the time-defined operating state.

Delay circuits can be realized with monostable elements with which the start (fig. 4.1.3) or end (fig. 4.1.4) of a setting pulse is delayed.

The actual time for t_1 or t_2 can then be used for a concrete delay element.

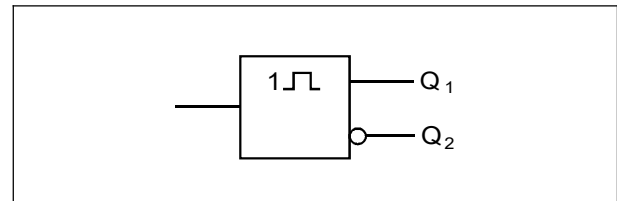


Fig. 4.1.1 Non-retriggerable monoflop

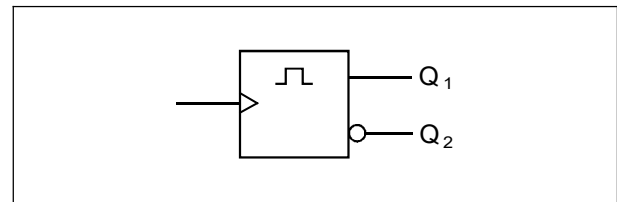


Fig. 4.1.2 Retriggerable monoflop

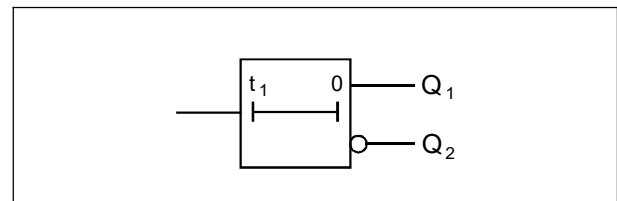


Fig. 4.1.3

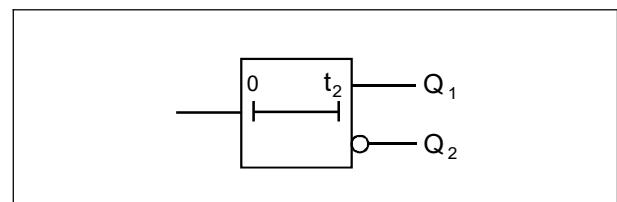


Fig. 4.1.4

4.2 Experiments Section

4.2.1 The Monoflop of the Digital Training System

□ Experiment 1: Fundamental principles

Experiment procedure:

- Examine the switching behaviour of the monoflop by completing the pulse diagram in fig. 4.2.1.1. Select $t_Q = 5$ s.
- Then complete the circuit symbol in fig. 4.2.1.2.

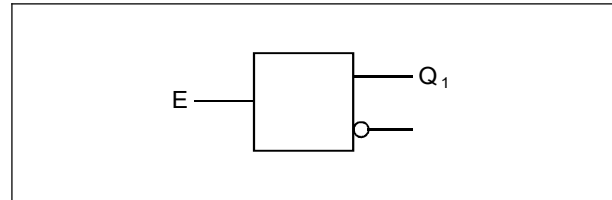


Fig. 4.2.1.2 Circuit symbol

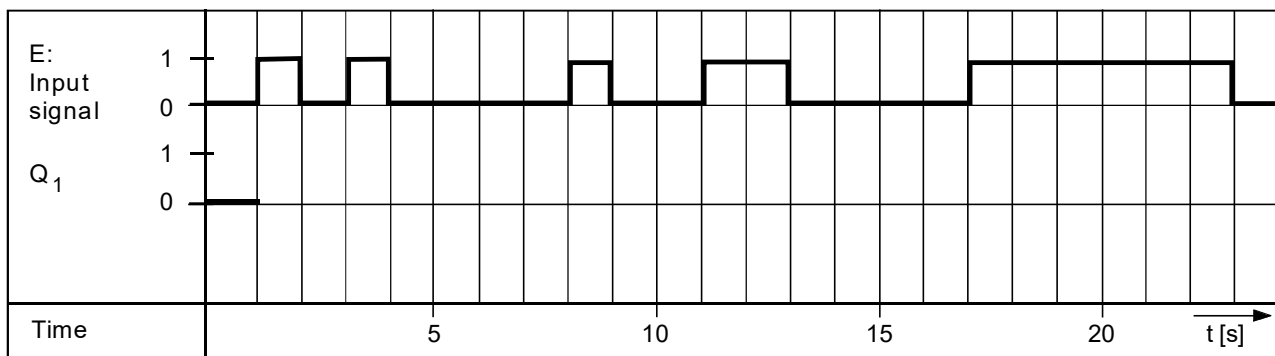


Fig. 4.2.1.1 Pulse diagram

Notes:

4.2.2 Delay Circuits

□ Experiment 1: Switch-on delay

The circuit in fig. 4.2.2.1 shows the principle of a switch-on delay.

Experiment procedure:

- Examine the circuit in fig. 4.2.2.1 with the Digital Training System.
- **Note:** Use the input keyboard to generate the input signal E.
- Complete the pulse diagram in fig. 4.2.2.2 and then complete the circuit symbol in fig. 4.2.2.3.

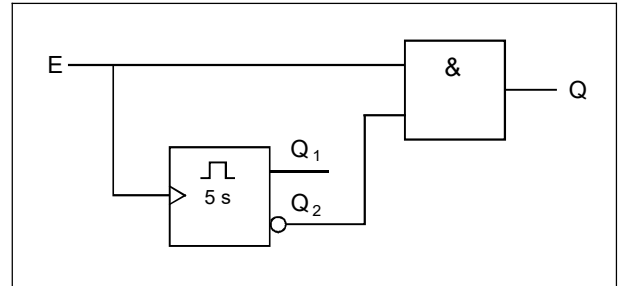


Fig. 4.2.2.1 Circuit

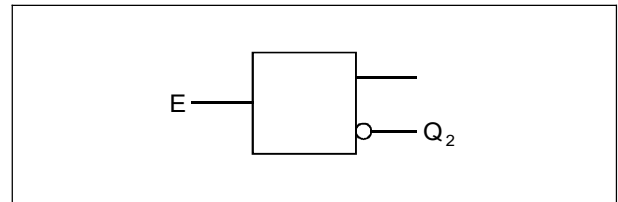


Fig. 4.2.2.3 Circuit symbol

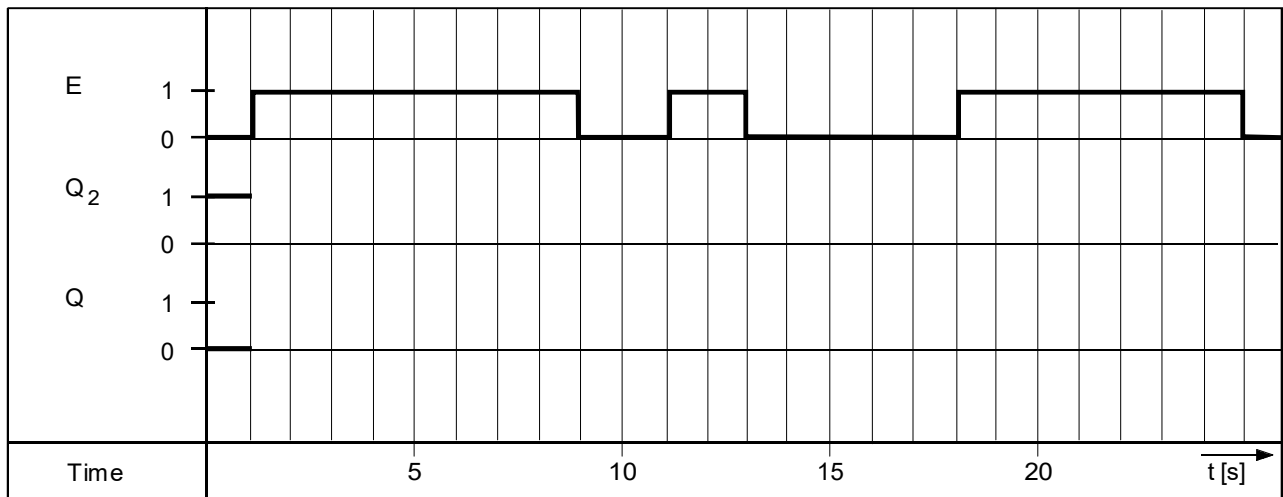


Fig. 4.2.2.2 Pulse diagram of a switch-on delay

□ Experiment 2: Switch-off delay

Fig. 4.2.2.4 shows the pulse diagram of a switch-off delay.

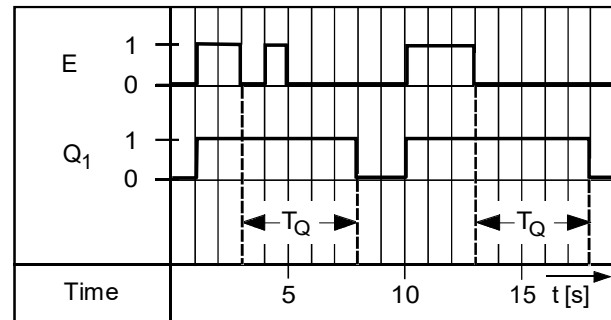


Fig. 4.2.2.4 Pulse diagram

Experiment procedure:

- Complete the circuit in fig. 4.2.2.5 using the pulse diagram in fig. 4.2.2.4.
- Test the circuit with the Digital Training System.
- Then complete the circuit symbol in fig. 4.2.2.6.

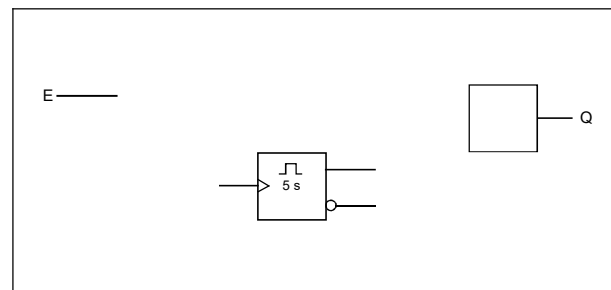


Fig. 4.2.2.5 Circuit for a switch-off delay

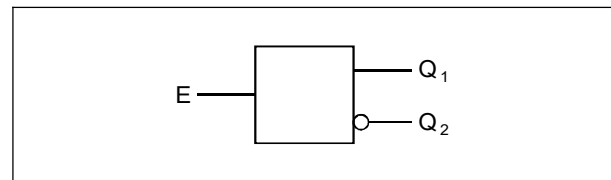


Fig. 4.2.2.6 Circuit symbol

Notes:

□ Experiment 3: Signal propagation time

Both circuits (fig. 4.2.2.1 and 4.2.2.5) explain the **principle** of a delay circuit. In practical application, a signal propagation time of **47.5 ns** must be taken into account. The signal propagation time of a component is the delay between the input and output voltage.

If, for example, the signal propagation time of T to Q₁ of a monoflop is given as 47.5 ns, this means that the output does not switch to H or L level until 47.5 ns after the clock signal appears.

Experiment procedure:

- Examine the effect on the switch-on delay (experiment 1) by completing the pulse diagram in fig. 4.2.2.7.
- **Note:** Scale: 47.5 ns = 1/2 stroke spacing

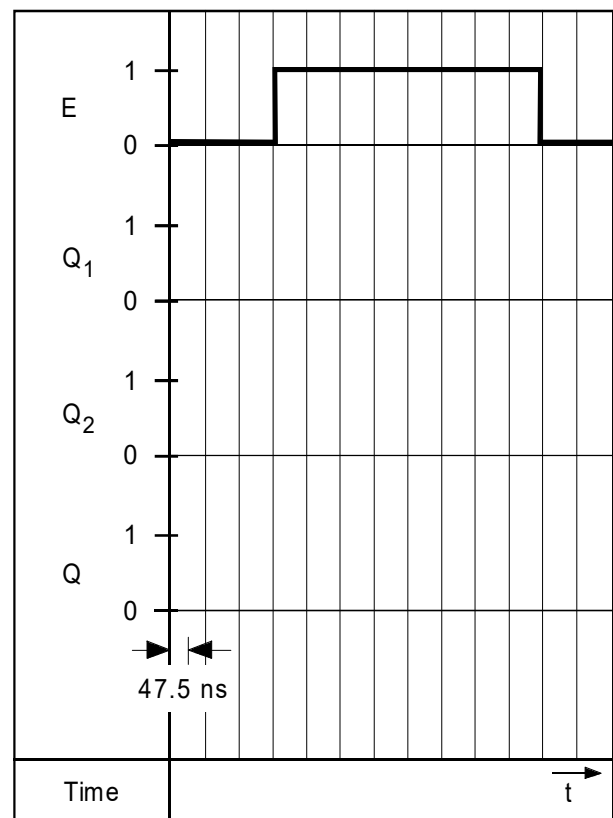


Fig. 4.2.2.7 Pulse diagram

Notes:

□ Experiment 4: Prevention of the signal propagation time pulse

Experiment procedure:

- Design a circuit which prevents the „most undesirable“ signal propagation pulse at the output Q.
- Complete the circuit in fig. 4.2.2.8.
- Test the circuit with the Digital Training System.
- Complete the pulse diagram in fig. 4.2.2.9.

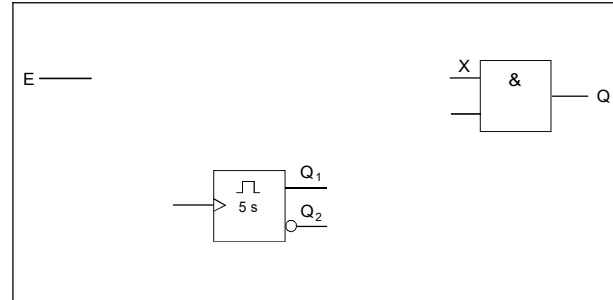


Fig. 4.2.2.8 Circuit

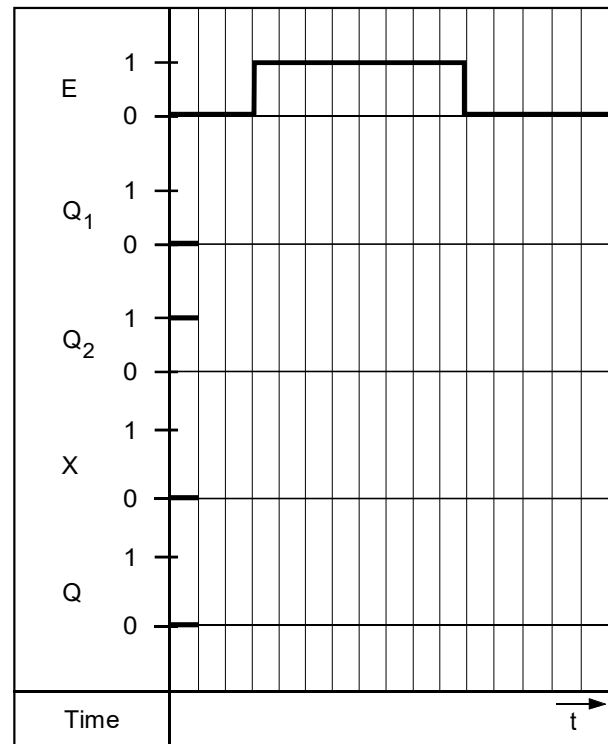


Fig. 4.2.2.9 Pulse diagram

Notes:

5. Code Converters, Coders

5.1 Fundamental Principles

5.1.1 General

Code converters convert a digital code into another code.

The synthesis is produced by placing the function tables (e. g. table 5.1.2.1) of both codes next to each other. The origin code then determines the input variables and the destination code the dependent output variables. The final functions for every output are derived from this by Boolean logic equations and KV diagrams.

The redundant bit combinations of the origin code (e. g. its pseudo-tetrades) then determine the „don't care fields“ marked by an „*“.

5.1.2 8421-BCD/Three-Excess Code Converter

The following example shows the conversion of the 8421 code into three-excess code.

The following OR normal forms apply for the outputs e ... h:

$$e = 0 \vee 2 \vee 4 \vee 6 \vee 8$$

$$f = 0 \vee 3 \vee 4 \vee 7 \vee 8$$

$$g = 1 \vee 2 \vee 3 \vee 4 \vee 9$$

$$h = 5 \vee 6 \vee 7 \vee 8 \vee 9$$

These OR normal forms are simplified with the aid of KV diagrams (figs. 5.1.2.1 ... 5.1.2.4, page 54).

According to the simplified equations, the circuit can be set up as shown in fig. 5.1.2.5 (page 54).

Weighting	Origin code 8421-BCD				Destination code 3-excess			
	d	c	b	a	h	g	f	e
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	Don't care fields			
11	1	0	1	1				
12	1	1	0	0				
13	1	1	0	1				
14	1	1	1	0				
15	1	1	1	1				

Table 5.1.2.1

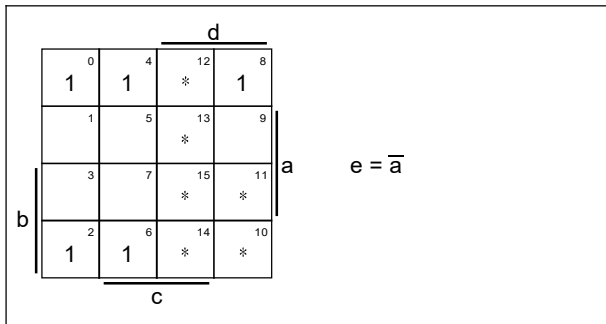


Fig. 5.1.2.1 KV diagram for e

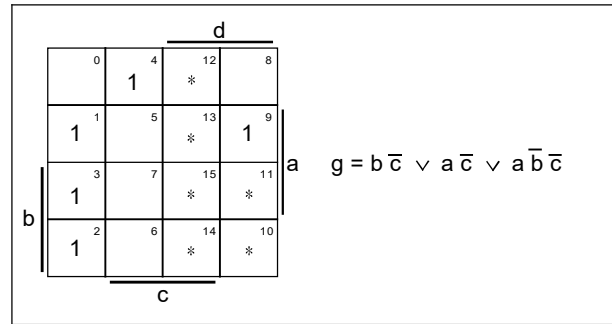


Fig. 5.1.2.3 KV diagram for g

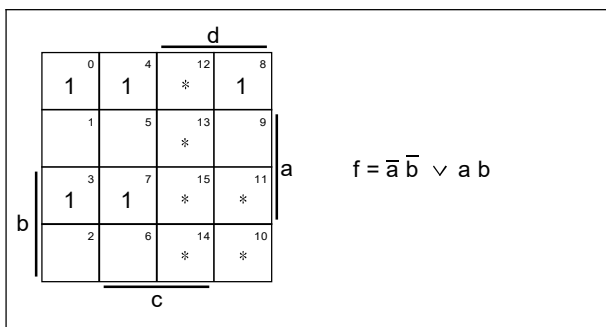


Fig. 5.1.2.2 KV diagram for f

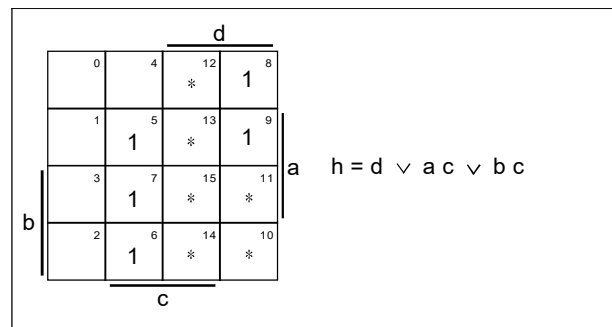


Fig. 5.1.2.4 KV diagram for h

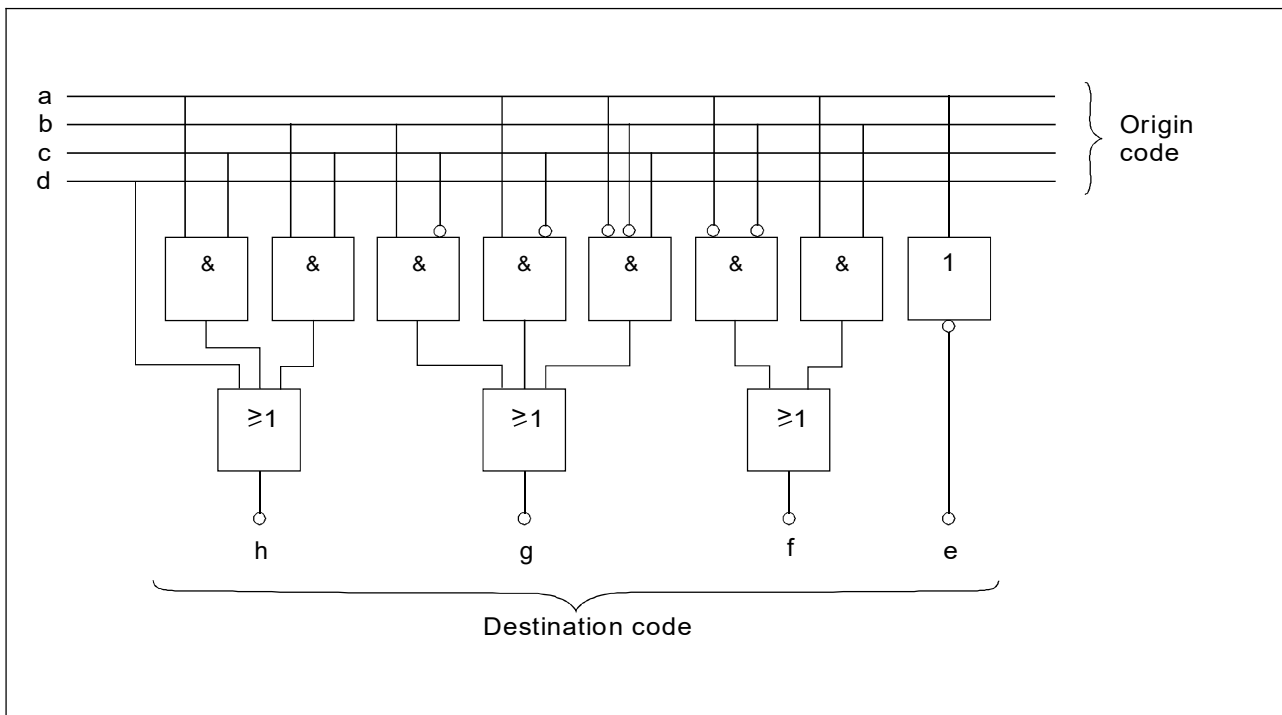


Fig. 5.1.2.5 Circuit for an 8421-BCD / three-excess code converter

If only NAND elements are available for the circuit setup, they must be converted accordingly:

$$e = \bar{a}$$

$$f = \overline{\bar{a}\bar{b} \wedge a b}$$

$$g = \overline{b\bar{c} \wedge a\bar{c} \wedge a\bar{b}\bar{c}}$$

$$h = \overline{\bar{d} \wedge \bar{a}\bar{c} \wedge \bar{b}c}$$

Code converters for any conversion task can be calculated by this method.

Code converters are represented by their own symbol in circuit diagrams. Fig. 5.1.2.6 applies for the example used here.

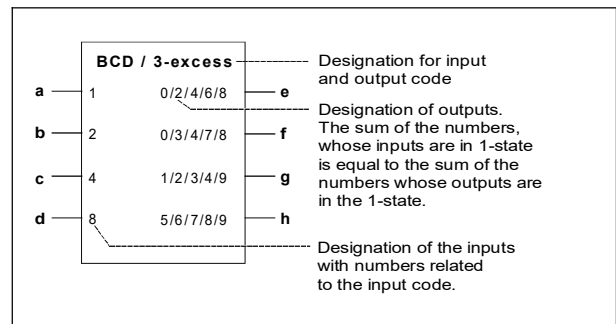


Fig. 5.1.2.6 Circuit symbol

Basically any code (even unweighted) can be used. The code designation in the circuit symbol is then replaced by the general identification X / Y with a reference to the table, e. g. [T 1] (fig. 5.1.2.7). T1 is here table 5.1.2.1 on page 53.

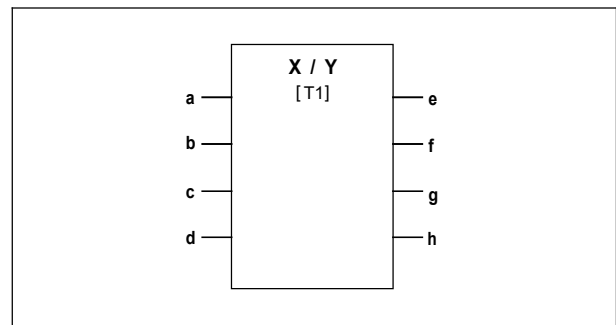


Fig. 5.1.2.7

5.2 Experiments Section

5.2.1 8421-BCD / Decimal Code Converter

Experiment 1:

In many digital circuits - especially in computer circuits - numbers with decimal digits are entered and output. Since the computer components provide the result in BCD code, however, it is necessary to convert into decimal code.

- The circuit cannot be set up with the Digital Training System in this way. Consider minimization possibilities, i. e. how you can reduce the number of your gate inputs. Then complete the circuit in fig. 5.2.1.1 and check the function with the Digital Training System.

Experiment procedure:

- Complete the table 5.2.1.1 and calculate the code converter according to the known algorithm.
- Enter the designation of the code, the inputs and the outputs in the circuit symbol (fig. 5.2.1.2) under observation of the rules.

Dec. value	Inputs: BCD code				Outputs: Decimal code									
	2 ³	2 ²	2 ¹	2 ⁰										
	d	c	b	a	e	f	g	h	i	j	k	l	m	n
0														
1														
2														
3														
4														
5														
6														
7														
8														
9														

Table 5.2.1.1

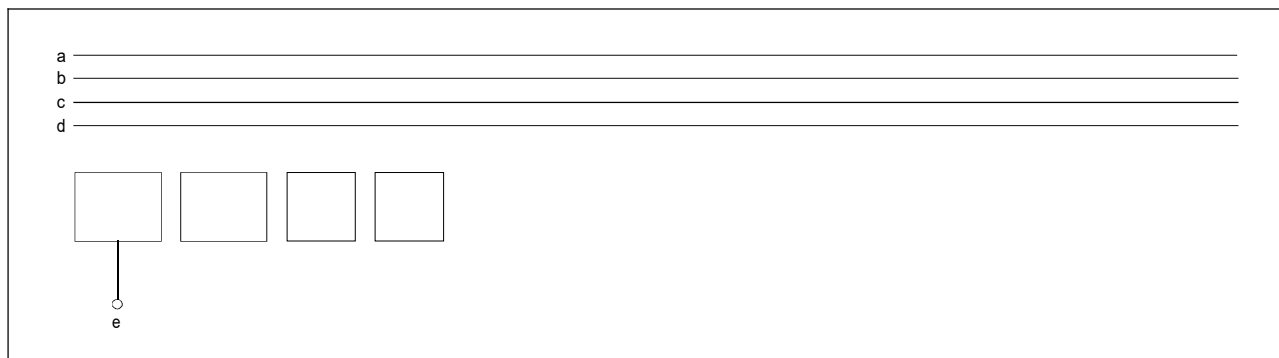


Fig. 5.2.1.1 Circuit for an 8421-BCD / decimal code converter

Function equations:

e =
f =
g =
h =
i =
j =
k =
l =
m =
n =

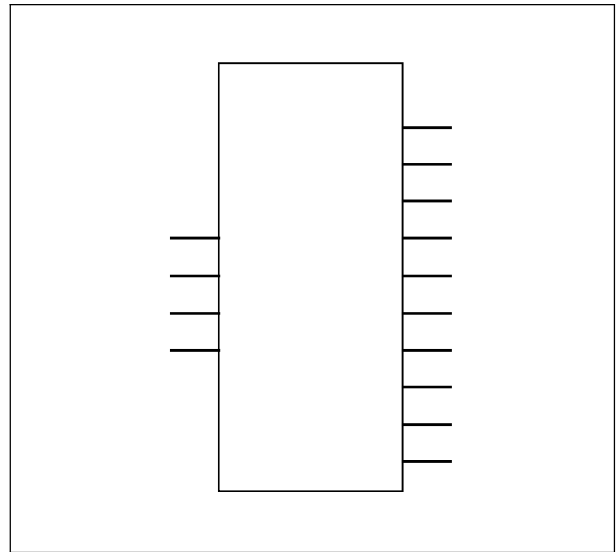


Fig. 5.2.1.2 Circuit symbol

Notes:

5.2.2 8421-BCD / 7-segment Code Converter

The BCD code is used extensively. BCD-coded informations are therefore output frequently through 7-segment display units.

Fig. 5.2.2.1 and 5.2.2.2 show the segment designations and the display table T1.

Experiment 1:

Experiment procedure:

- Enter the output values and the displayed digit in table 5.2.2.1 according to the specifications in the display table T1 (fig. 5.2.2.2).
- A disjunctive normal form can be created for each of the outputs a ... g which you can simplify with the aid of the prepared KV diagrams fig. 5.2.2.3 ... 5.2.2.9.
Note: Use the pseudo-tetrades!
- Complete the circuit in fig. 5.2.2.10 (page 60) and test it with the Digital Training System. Use the 7-segment display for checking. The input x of the display must be applied to L level.

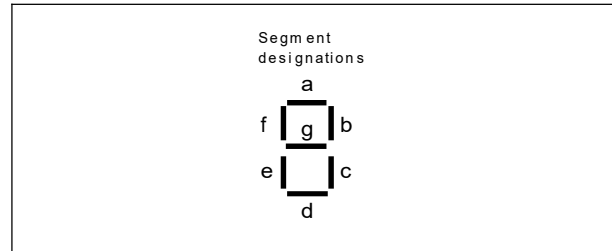


Fig. 5.2.2.1

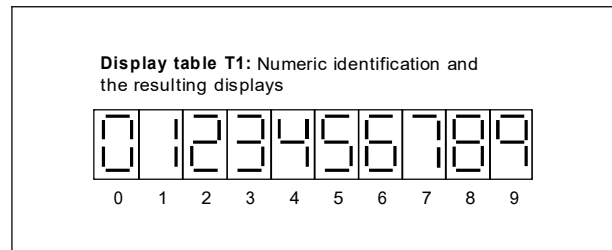


Fig. 5.2.2.2

Inputs: BCD code				Digit	Outputs: 7-segment display						
8	4	2	1		a	b	c	d	e	f	g
D	C	B	A								
0	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	0	0	0								
1	0	0	1								

Table 5.2.2.1

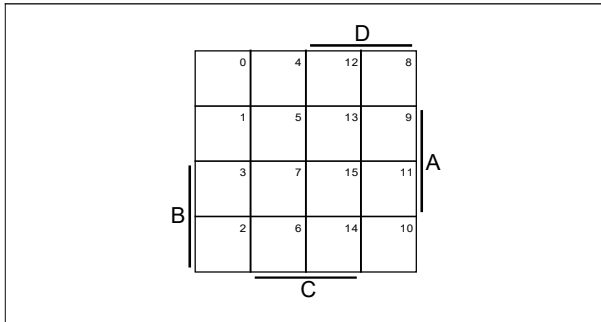


Fig. 5.2.2.3 KV diagram for a

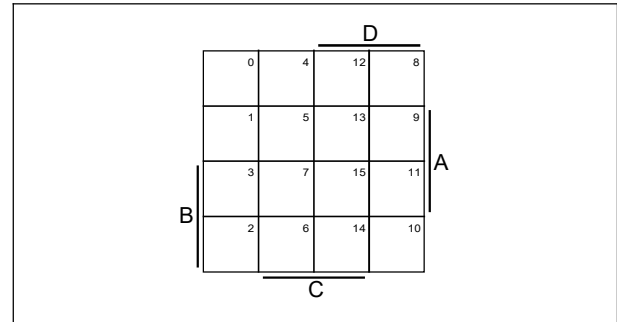


Fig. 5.2.2.4 KV diagram for b

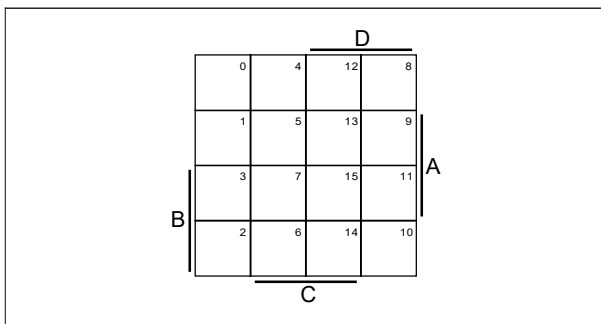


Fig. 5.2.2.5 KV diagram for c

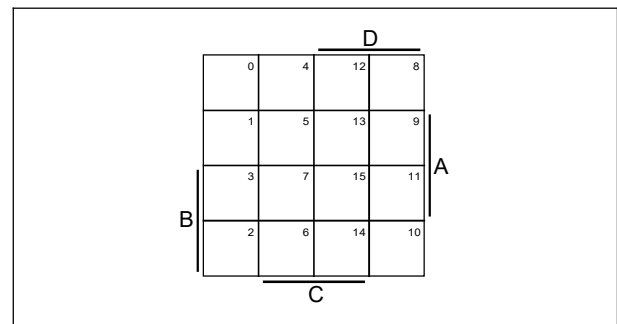


Fig. 5.2.2.6 KV diagram for d

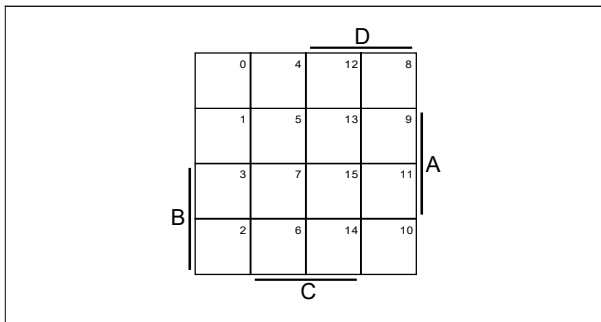


Fig. 5.2.2.7 KV diagram for e

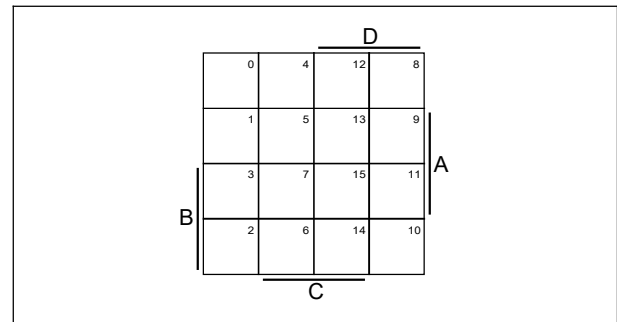


Fig. 5.2.2.8 KV diagram for f

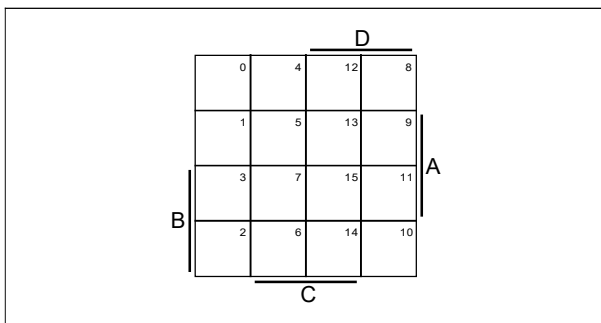


Fig. 5.2.2.9 KV diagram for g

Minimized equations:

- a =
- b =
- c =
- d =
- e =
- f =
- g =

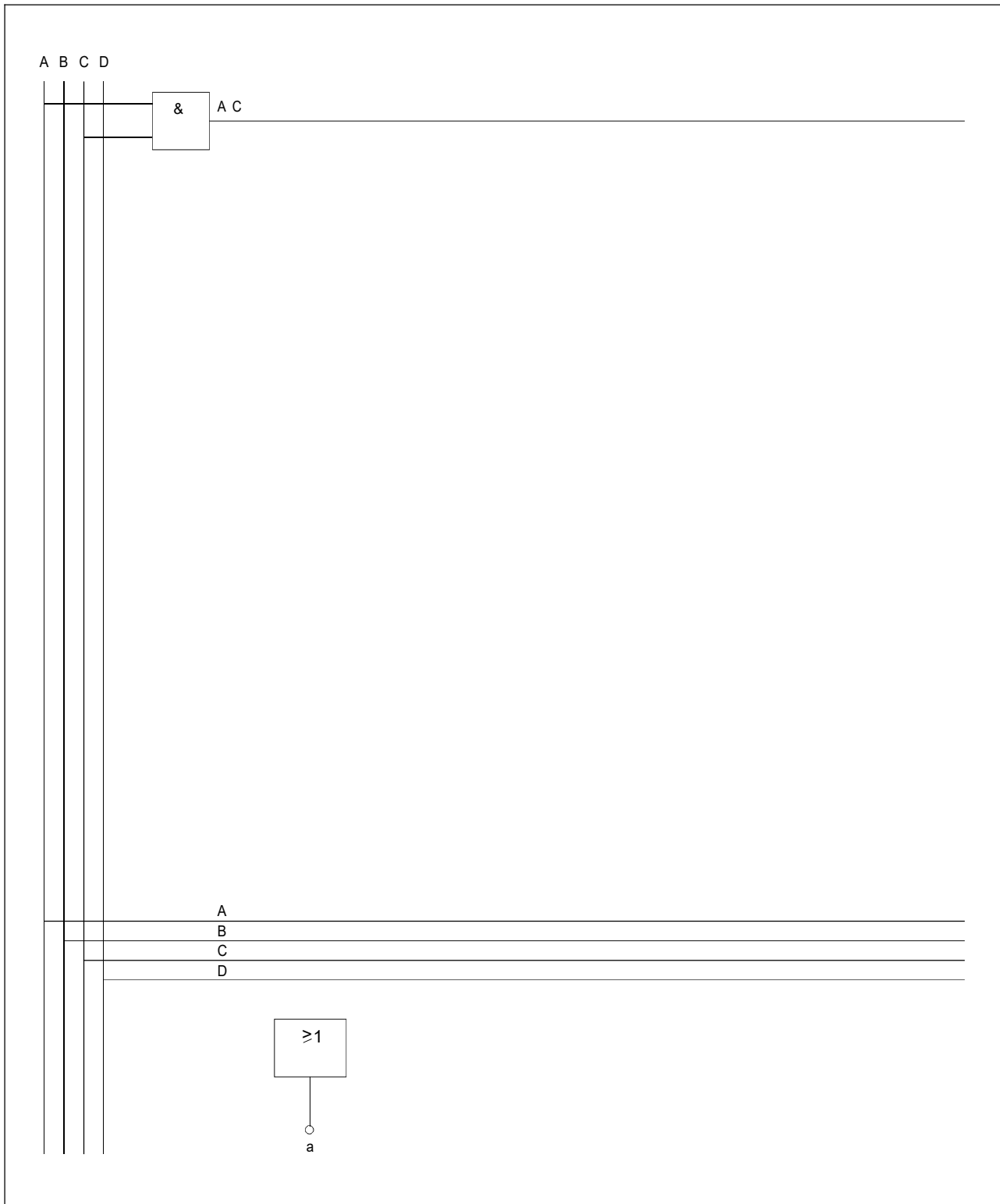


Fig. 5.2.2.10 Circuit for an 8421-BCD / 7-segment code converter

5.2.3 Coding Circuits

The circuits in fig. 5.2.3.1 and 5.2.3.2 represent one code converter each which converts the 8421-BCD code to another code.

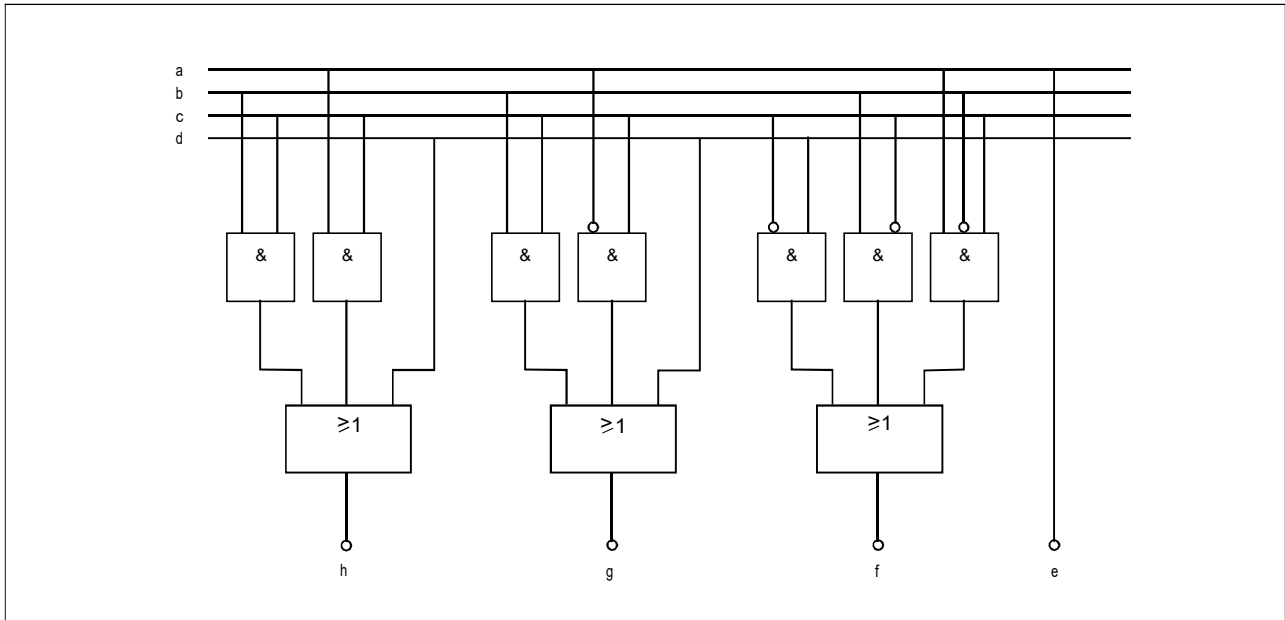


Fig. 5.2.3.1 Circuit 1

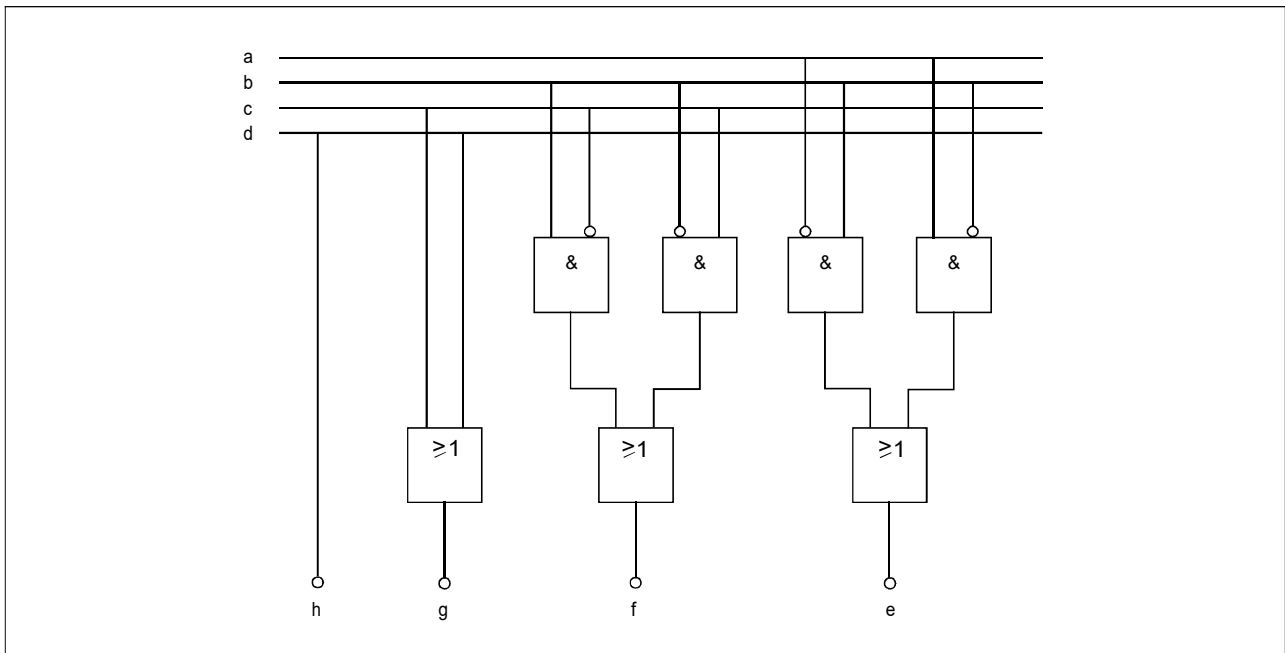


Fig. 5.2.3.2 Circuit 2

□ Experiment 1:

Experiment procedure:

- Complete table 5.2.3.1 for the two code converters (fig. 5.2.3.1 and 5.2.3.2).
- Enter the designation of the code, the inputs and the outputs in the specified circuit symbols (fig. 5.2.3.3 and 5.2.3.4) under observation of the rules.

8421-BCD code				Circuit 1				Circuit 2			
d	c	b	a	h	g	f	e	h	g	f	e

Table 5.2.3.1

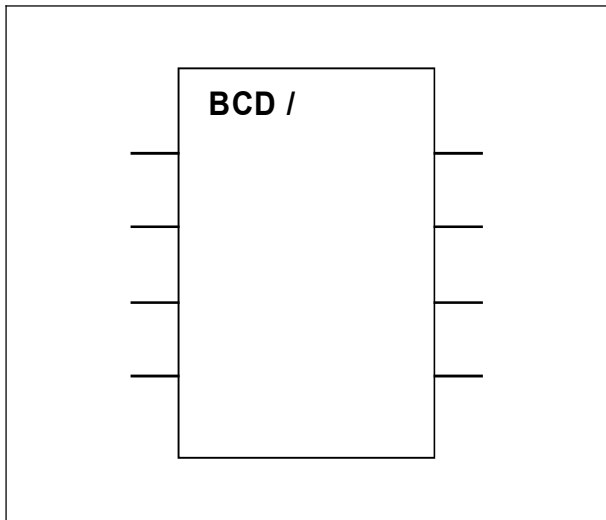


Fig. 5.2.3.3 Circuit symbol for circuit 1

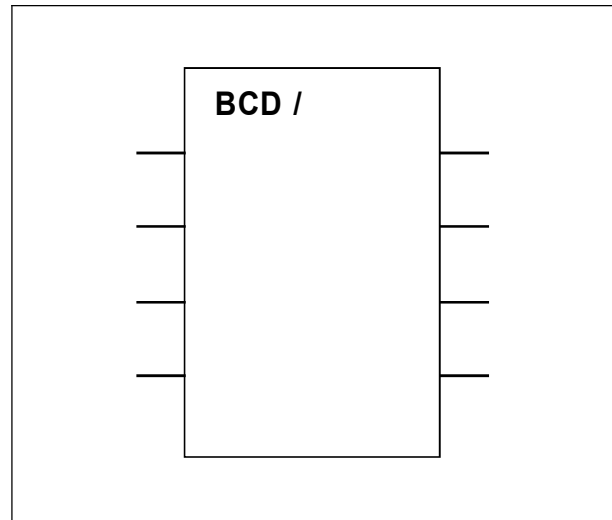


Fig. 5.2.3.4 Circuit symbol for circuit 2

6. Arithmetic Circuits

6.1 Fundamental Principles

6.1.1 General

In computers, all arithmetic functions can be narrowed down to simple additions and subtractions (even the subtraction is carried out by the adding unit). The adding unit is therefore the only part of a processor with which arithmetic operations can be performed.

Such an adding circuit is required for every dual digit which can be derived from the rules of addition for dual numbers.

6.1.2 Semi Adder

Table 6.1.2.1 shows an example for the addition of two 1-digit dual numbers P_0 and Q_0 .

The adding circuit must link the four possible level combinations in a **sum** Σ_0 and a **carry** **CO**.

In the above example, the sum has the same **weight** as the summands (2^0) and the carry the weight of the next highest digit (2^1).

The logic circuit is designed according to the switching algebra equations (disjunctive normal form).

Q_0	P_0	Σ_0	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 6.1.2.1

$$\Sigma_0 = (P_0 \wedge \bar{Q}_0) \vee (\bar{P}_0 \wedge Q_0)$$

Antivalence, Exclusive OR

$$CO = P_0 \wedge Q_0$$

Fig. 6.1.2.1 shows the circuit symbol of a semi adder.

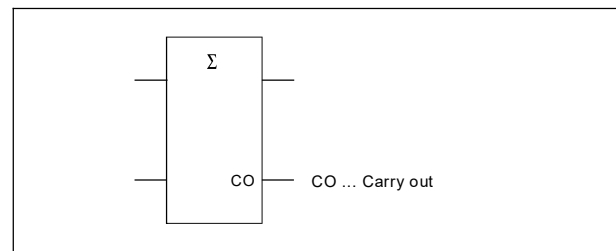


Fig. 6.1.2.1 Circuit symbol of a semi adder

Although the circuit itself supplies a carry CO, it does not take carries from other addition stages into account. It is therefore not a complete adder as is therefore referred to as a **semi adder**.

Fig. 6.1.2.2 shows the circuit of a semi adder.

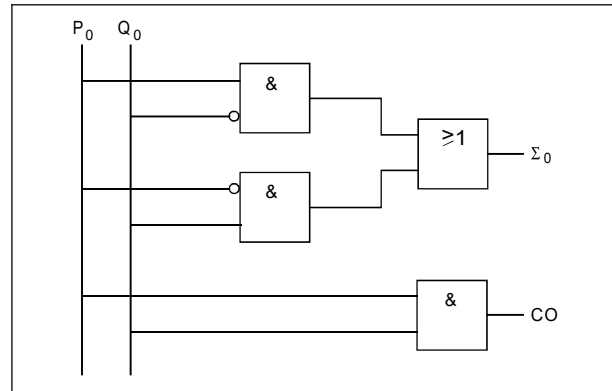


Fig. 6.1.2.2 Circuit of a semi adder

6.1.3 Full Adder

The full adder has an additional third input which takes the carry from the previous, lower value dual digit in the addition of multi-digit dual numbers and adds it to the sum of P_1 and Q_1 .

An example of this would be the addition of the three one digit dual numbers P_1 , Q_1 and CI .

A **1-bit full adder** (table 6.1.3.1) can be realized by two cascade circuited semi adders. Addition then takes place in two stages:

1. Addition of the two numbers P_1 and Q_1 to a subtotal E_1 and the carry C_2 (1st semi adder)
2. Addition of the carry CI and the subtotal E_1 to the final total E and the carry C_3 (2nd semi adder)

The final carry output CO becomes „1“ if the first or the second semi adder supplies a „1“ as a carry. The linkage of the two partial carries is realized by an OR element. Fig. 6.1.3.1 and 6.1.3.2 show the circuit symbol and the circuit of a 1-bit full adder.

To obtain a complete arithmetic unit, a number of full adders corresponding to the used dual digits must be interconnected so that every stage takes over the carry of the respective lower value stage. An example for this would be an adder for decimal numbers.

1st semi adder				2nd smi adder			
Q_1	P_1	Σ_1	C_2	CI	Σ_1	Σ	C_3
0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0
1	0	1	0	1	1	0	1
1	1	0	1	1	0	1	0

Table 6.1.3.1

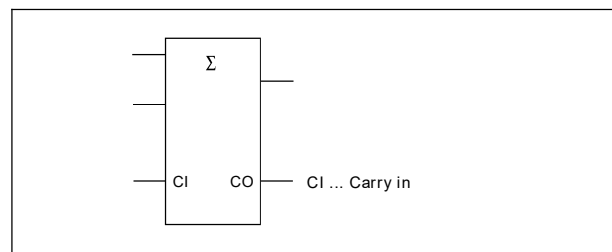


Fig. 6.1.3.1 Circuit symbol of a 1-bit full adder

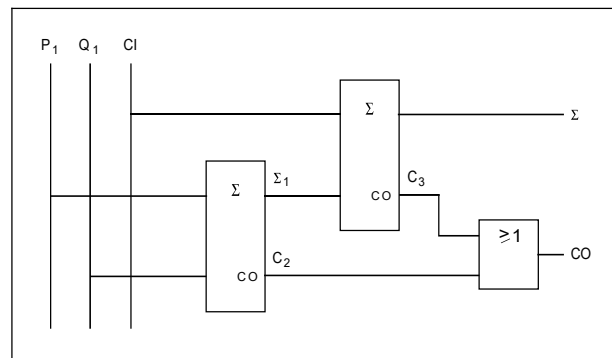


Fig. 6.1.3.2 Circuit of a 1-bit full adder

An adder like this (4-bit full adder, fig. 6.1.3.3) can be considered in principle as a cascade circuit of four 1-bit full adders. However, this type of circuit is rarely used because of the resulting long signal propagation times.

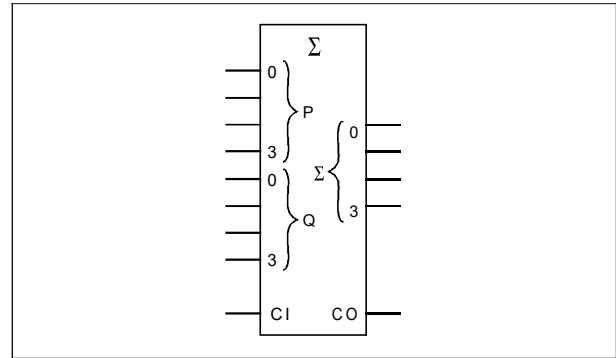


Fig. 6.1.3.3 Circuit symbol of a 4-bit full adder

6.1.4 Correction Addition for Decimal Numbers

Unlike digital computers, man usually works in the decimal system. All numbers or computing results must be converted accordingly by coding or decoding circuits. This coding consists not only of the direct conversion of every single keypress, it is also dependent on the respective previously entered digits. This dependence makes realization of the circuit highly complex. Computers therefore operate not in pure dual code but in the 8421-BCD code. Since, however, the basic circuits of the computer operate in „pure“ dual code, results are produced in „pure“ dual code which have to be corrected to 8421-BCD code.

The following three groups of results are possible:

1. If the result of the addition is less than „10“, it corresponds to the decimal number in the 8421-BCD code (table 6.1.4.1).
2. The result of the addition is a pseudo-tetrad and must be corrected (table 6.1.4.2).
3. The result of the addition gives a carry and must be corrected (table 6.1.4.3).

This necessary correction can be achieved in several possible ways:

- With OR and AND gates only. The circuit is set up on the same principle as a transcoder but the circuit is very time-intensive.

	Decimal number	8421-BCD code
P	4	0 1 0 0
Q	3	0 0 1 1
Σ	7	0 1 1 1

Table 6.1.4.1

	Decimal number	8421-BCD code
P	8	1 0 0 0
Q	4	0 1 0 0
Σ	12	1 1 0 0
Correction tetrad		+ 0 1 1 0
	0 0 0 1	0 0 1 0
	1	2

Table 6.1.4.2

	Decimal number	8421-BCD code
P	9	1 0 0 1
Q	8	1 0 0 0
Σ	17	1 0 0 0 1
Correction tetrad		+ 0 1 1 0
	0 0 0 1	0 1 1 1
	1	7

Table 6.1.4.3

- with AND and OR gates and an additional 4-bit full adder
- with OR gates, a 4-bit number comparator and an additional 4-bit full adder

All three circuit versions are demonstrated in the experiments section.

6.1.5 Subtractor for Dual Numbers

In arithmetic units the subtraction is converted into an addition by adding the complement of the subtrahend Q to the minuend P.

$$\begin{array}{r}
 7 \quad \text{minuend P} \\
 - 5 \quad \text{subtrahend Q} \\
 \hline
 + 2 \quad \text{result } \Sigma \text{ with sign „+“}
 \end{array}$$

$$\begin{array}{r}
 P \quad 0 \ 1 \ 1 \ 1 \\
 Q \quad 0 \ 1 \ 0 \ 1
 \end{array}$$

The 2s complement of Q is formed by inverting every bit (1s complement) and then adding a „1“.

$$\begin{array}{r}
 \bar{Q} \quad 1 \ 0 \ 1 \ 0 \\
 + \quad \quad \quad 1 \\
 \hline
 1 \ 0 \ 1 \ 1 \quad \text{2s complement of Q}
 \end{array}$$

$$\begin{array}{r}
 0 \ 1 \ 1 \ 1 \quad P \\
 + 1 \ 0 \ 1 \ 1 \quad \text{2s complement} \\
 \hline
 \text{+} 0 \ 0 \ 1 \ 0
 \end{array}$$

The carry produced in the complement addition is not part of the subtraction result but can be used for sign evaluation.

Since a **positive sign** is represented by a „0“ in binary coded numbers, the carry must be negated in the circuit.

Fig. 6.1.5.1 shows the subtracting circuit when the minuend is greater than the subtrahend.

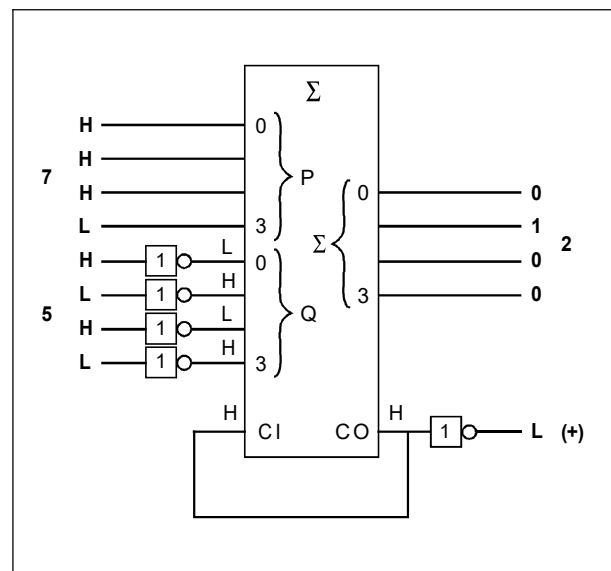


Fig. 6.1.5.1 Minuend > subtrahend

The 2s complement is formed by interconnection of CI and CO.

If the result of a subtraction is negative, 2s complement formation must also take place at the output:

$$\begin{array}{r}
 5 \quad \text{minuend P} \\
 - 7 \quad \text{subtrahend Q} \\
 \hline
 - 2 \quad \text{result } \Sigma \text{ with sign „-“}
 \end{array}$$

$$\begin{array}{r}
 P \quad 0 \ 1 \ 0 \ 1 \\
 Q \quad 0 \ 1 \ 1 \ 1
 \end{array}$$

$$\begin{array}{r}
 \bar{Q} \quad 1 \ 0 \ 0 \ 0 \\
 + \quad \quad \quad 1 \\
 \hline
 1 \ 0 \ 0 \ 1 \quad \text{2s complement of Q}
 \end{array}$$

$$\begin{array}{r}
 0 \ 1 \ 0 \ 1 \quad P \\
 + 1 \ 0 \ 0 \ 1 \quad \text{2s complement} \\
 \hline
 - 1 \ 1 \ 1 \ 0 \quad \text{2s complement of} \\
 \text{result } \Sigma
 \end{array}$$

$$\begin{array}{r}
 0 \ 0 \ 0 \ 1 \quad \bar{\Sigma} \\
 + \quad \quad \quad 1 \\
 \hline
 0 \ 0 \ 1 \ 0
 \end{array}$$

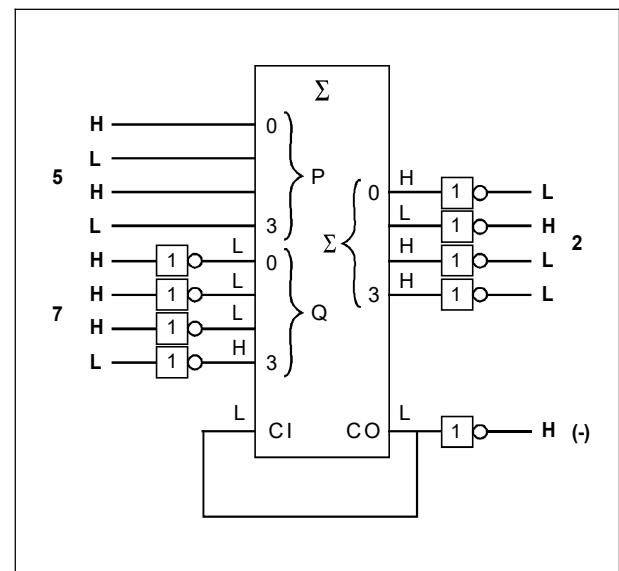


Fig. 6.1.5.2 shows the subtraction circuit when the minuend is smaller than the subtrahend.

In the circuit in fig. 6.1.5.2, addition of the „1” is omitted because it would have to be done at the input and the output.

Fig. 6.1.5.2 Minuend < subtrahend

6.2 Experiments Section

6.2.1 Semi Adder

□ Experiment 1: Semi adder consisting of basic elements

Experiment procedure:

- Design a semi adder consisting exclusively of basic elements and complete the circuit diagram for the adding circuit in fig. 6.2.1.1.
- Enter the output values of all gates for the specified input values in table 6.2.1.1.
- Check that the circuit and table are correct with the Digital Training System.

Summands		Gate outputs				Sum	
P_0	Q_0	D1	D2	D3	D4	Σ_0	C_1
0	0						
0	1						
1	0						
1	1						

Table 6.2.1.1

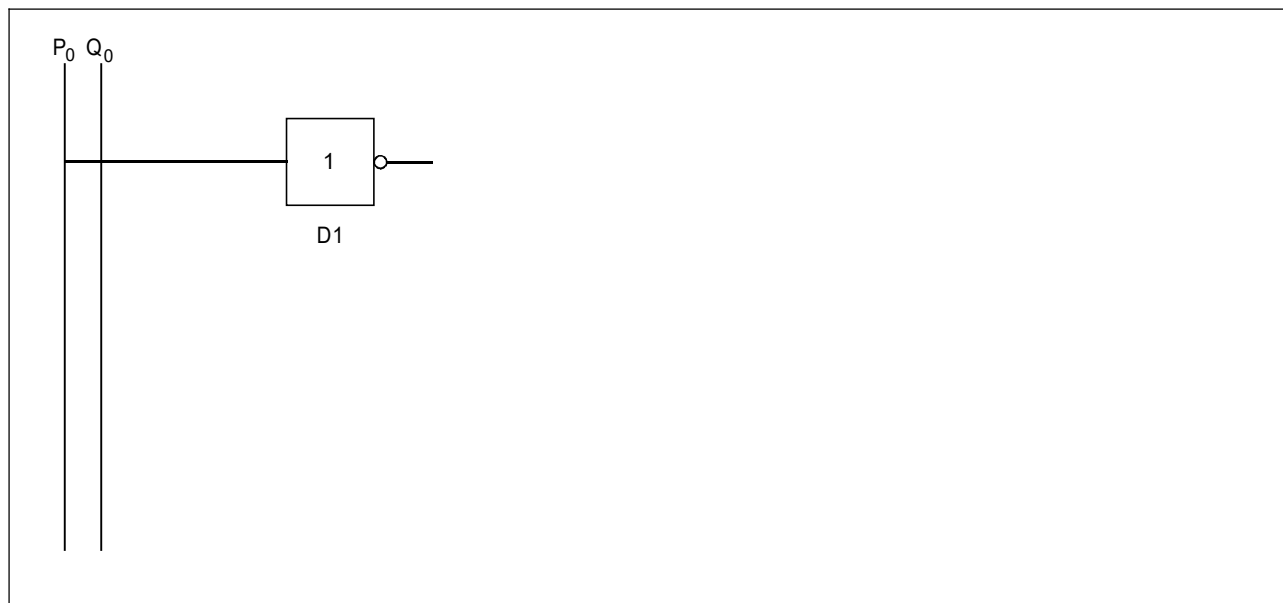


Fig. 6.2.1.1 Circuit for a semi adder consisting of basic elements

❑ Experiment 2: Semi adder consisting exclusively of NAND elements

Experiment procedure:

- Design a semi adder consisting exclusively of NAND elements. Complete the circuit diagram for the adding circuit in fig. 6.2.1.2.
- Enter the output values of all gates for the specified input values in table 6.2.1.2.
- Check that the circuit and table are correct with the Digital Training System.
- The semi adder can also be set up with just two gates. Complete the circuit in fig. 6.2.1.3 and check it with the Digital Training System.

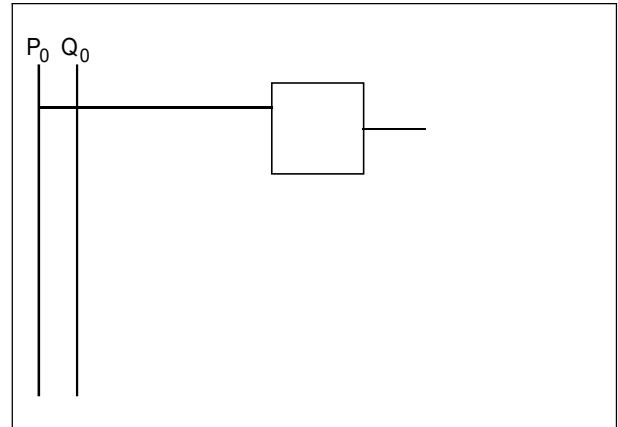


Fig. 6.2.1.3 Circuit

Summands		Gate outputs					Sum	
P ₀	Q ₀	D1	D2	D3	D4	D5	Σ ₀	C ₁
0	0							
0	1							
1	0							
1	1							

Table 6.2.1.2

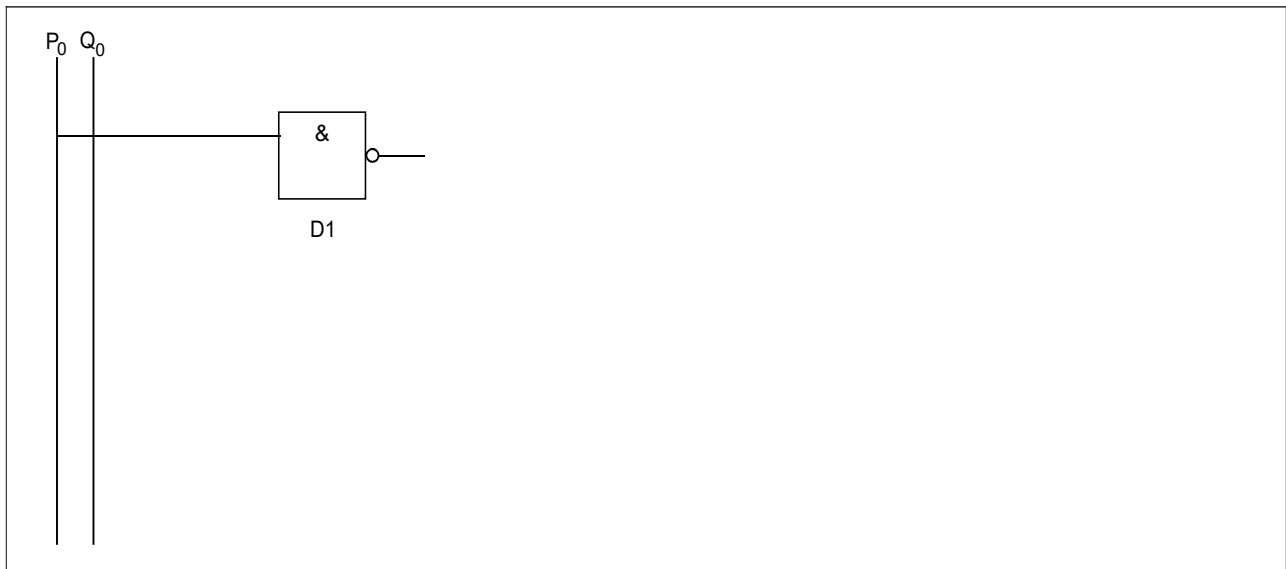


Fig. 6.2.1.2 Circuit for a semi adder consisting exclusively of NAND elements

6.2.2 Full Adder

□ Experiment 1 : 1-bit full adder

A 1-bit full adder should be set up with AND, OR and NOT elements.

Experiment procedure:

- Complete the table 6.2.2.1.
- Compose the minimized formulae for the sum Σ and the carry CO with the help of KV diagrams (fig. 6.2.2.1 and 6.2.2.2).
- Complete the circuit in fig. 6.2.2.3 and check the function with the Digital Training System.

P_1	Q_1	CI	Σ	CO
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
0	1	0		
1	1	1		

Table 6.2.2.1

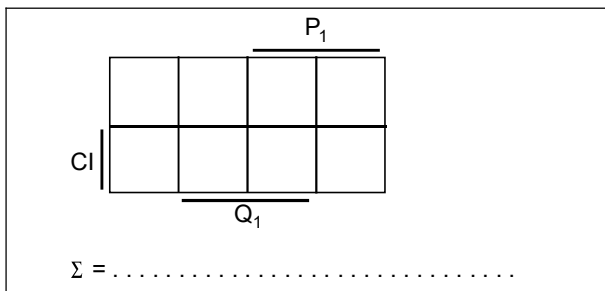


Fig. 6.2.2.1 KV diagram for Σ

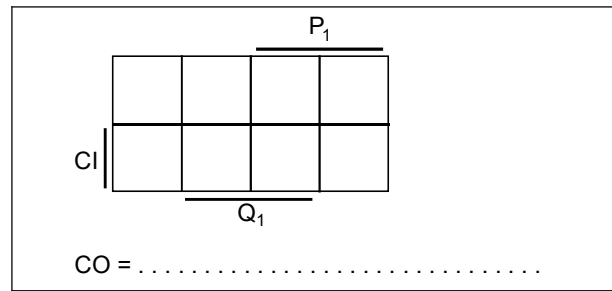


Fig. 6.2.2.2 KV diagram for CO



Fig. 6.2.2.3 Circuit for a 1-bit full adder

□ Experiment 2: 2-bit full adder

A 2-bit full adder should be designed from discrete semi adders.

Experiment procedure:

- Complete table 6.2.2.2.
- Complete the circuit in fig. 6.2.2.4.
- Test the 2-bit full adder with the Digital Training System.
- Enter the values for addition of the dual numbers P = 01 and Q = 11 in the circuit diagram for checking.

P ₁	Q ₁	P ₀	Q ₀	CO	Σ ₁	Σ ₀
0	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	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			

Table 6.2.2.2

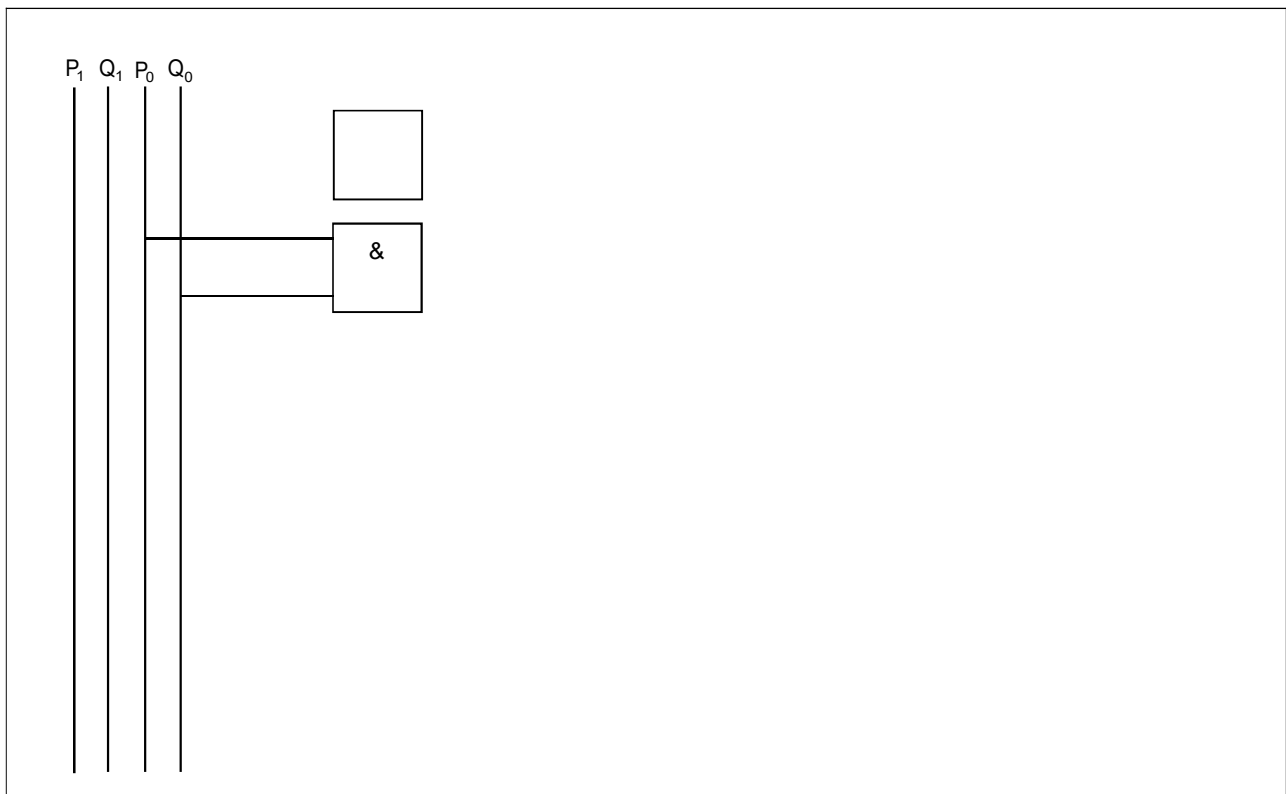


Fig. 6.2.2.4 Circuit for a 2-bit full adder

6.2.3 Adding Circuits for the 8421-BCD Code

□ Experiment 1: Addition of two decimal numbers

Circuits should be designed which enable the addition of decimal numbers. The resulting problem is shown in fig. 6.2.3.1 with the addition term 9 + 9 as an example.

Experiment procedure:

- Set up the correction circuit on the principle of a transcoder. The prepared algorithm in table 6.2.3.1 will help you.
- Complete table 6.2.3.1 and compose the function equations for Σ_0' , Σ_1' , Σ_2' , Σ_3' and CO' .
- With the KV diagrams in fig. 6.2.3.2 ... 6.2.3.5, check whether minimization is possible in each case. Use the don't care fields.
- Complete the circuit in fig. 6.2.3.6 (page 74) and check the function with the Digital Training System.

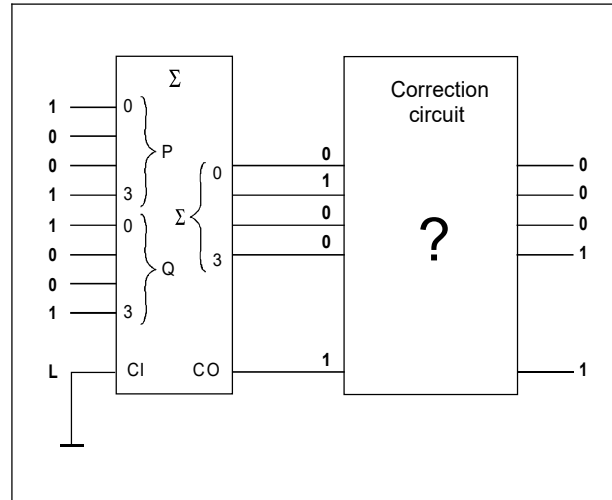


Fig. 6.2.3.1

Decimal number	Adding outputs („pure“ dual)					Corrected result (BCD code)				
	CO	Σ_3	Σ_2	Σ_1	Σ_0	CO'	Σ_3'	Σ_2'	Σ_1'	Σ_0'
0	0	0	0	0	0					
1	0	0	0	0	1					
2	0	0	0	1	0					
3	0	0	0	1	1					
4	0	0	1	0	0					
5	0	0	1	0	1					
6	0	0	1	1	0					
7	0	0	1	1	1					
8	0	1	0	0	0					
9	0	1	0	0	1					
10	0	1	0	1	0					
11	0	1	0	1	1					
12	0	1	1	0	0					
13	0	1	1	0	1					
14	0	1	1	1	0					
15	0	1	1	1	1					
16	1	0	0	0	0					
17	1	0	0	0	1					
18	1	0	0	1	0					
19	1	0	0	1	1					

Table 6.2.3.1

Function equations from table 6.2.3.1:

$$\begin{aligned} \Sigma_0' &= \dots\dots\dots \\ \Sigma_1' &= \dots\dots\dots \\ &= \dots\dots\dots \\ \Sigma_2' &= \dots\dots\dots \\ &= \dots\dots\dots \\ \Sigma_3' &= \dots\dots\dots \\ CO' &= \dots\dots\dots \\ &= \dots\dots\dots \end{aligned}$$

Minimized equations from KV diagrams:

$$\begin{aligned} \Sigma_0' &= \dots\dots\dots \\ \Sigma_1' &= \dots\dots\dots \\ \Sigma_2' &= \dots\dots\dots \\ \Sigma_3' &= \dots\dots\dots \\ CO' &= \dots\dots\dots \end{aligned}$$

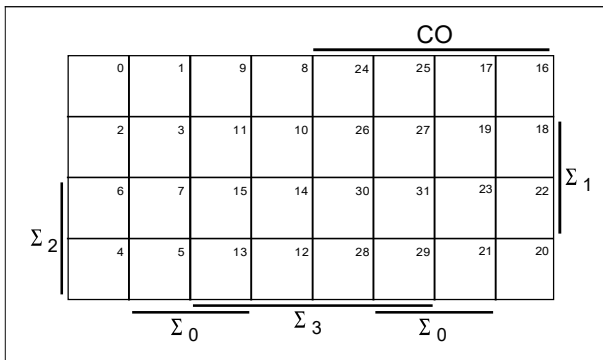


Fig. 6.2.3.2 KV diagram for Σ_1'

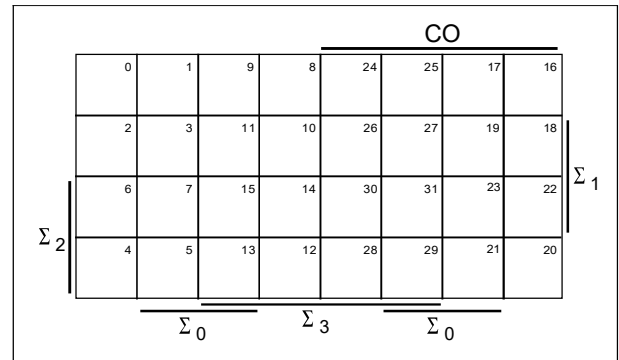


Fig. 6.2.3.3 KV diagram for Σ_2'

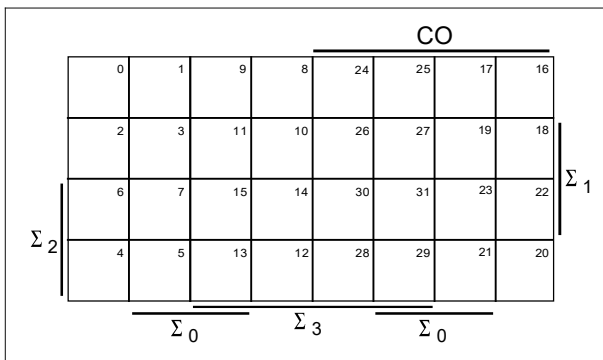


Fig. 6.2.3.4 KV diagram for Σ_3'

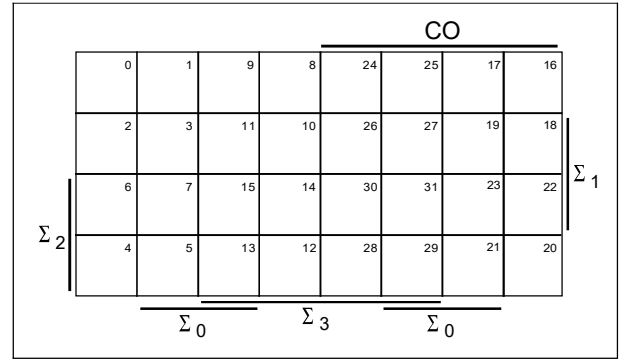


Fig. 6.2.3.5 KV diagram for CO'



Fig. 6.2.3.6 Circuit

Notes:

□ **Experiment 2: Simplified correction circuit**

With a second 4-bit full adder the necessary correction in experiment 1 can be made much simpler.

Experiment procedure:

- Complete the circuit in fig. 6.2.3.7.
- Mark the logic values in the circuit after every circuit part.



Fig. 6.2.3.7 Simplified correction circuit

Question 1: Give reasons for the logic value at the output CO of the second 4-bit full adder.

Answer:

.....

.....

.....

.....

Experiment 3: Addition with the aid of a 4-bit number comparator

An adding circuit should be designed for the addition of two 1-digit decimal numbers using a 4-bit number comparator.

The subtotal of the first adder D1 is compared in the comparator with the permanently applied number „9". In the case of $P < Q$ (Q is a pseudo-tetrad) or a carry to D1, this must be corrected with a second adder D2.

Question 1: When checking the circuit, you may find that the result is too high by „1" or "2". What are your immediate conclusions?

Answer:

.....

.....

.....

.....

.....

.....

Experiment procedure:

- Complete the circuit in fig. 6.2.3.8 and complete the table 6.2.3.2.
- Check the circuit with the Digital Training System.

Addition task	Full adder D1										Comparator D3	Full adder D2											
	P ₃	P ₂	P ₁	P ₀	Q ₃	Q ₂	Q ₁	Q ₀	CO	Σ ₃		Σ ₂	Σ ₁	Σ ₀	P ₃	P ₂	P ₁	P ₀	CO	Σ ₃	Σ ₂	Σ ₁	Σ ₀
4 + 3																							
8 + 4																							
8 + 8																							
9 + 9																							

Table 6.2.3.2

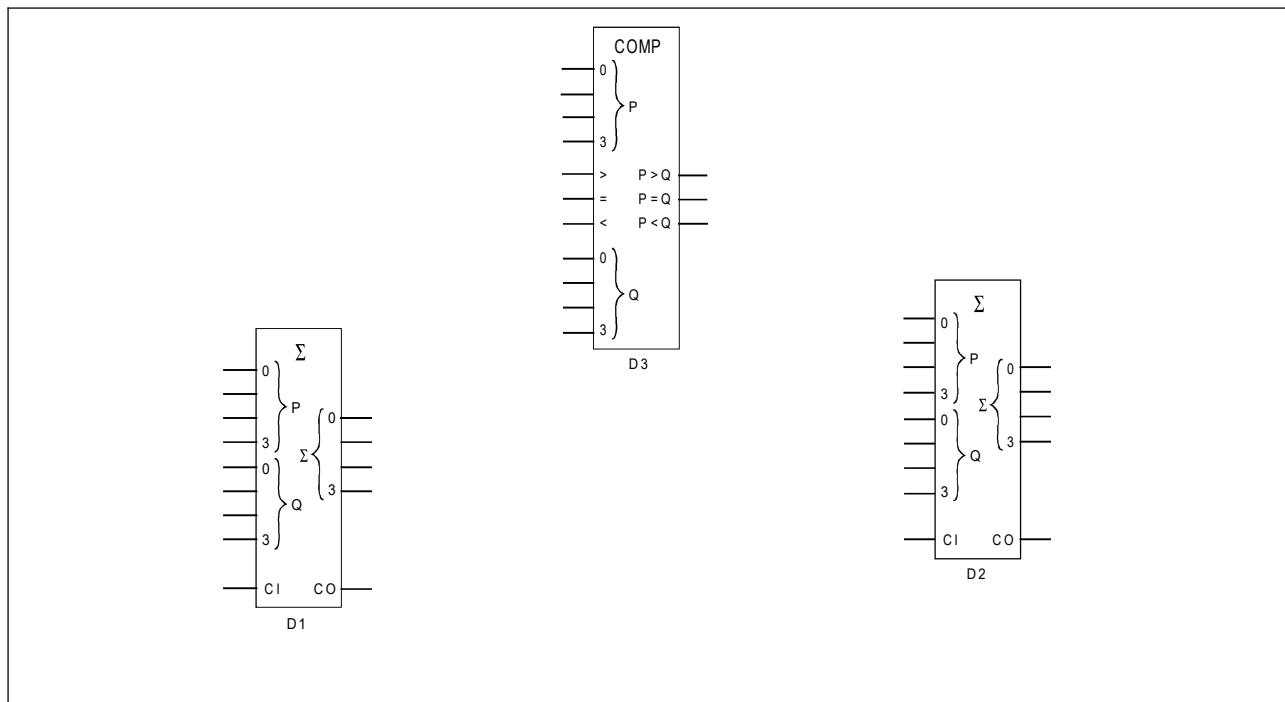


Fig. 6.2.3.8 Circuit

6.2.4 Semi Subtractor

□ Experiment 1: Subtraction of two 1-digit dual numbers

The simplest subtraction circuit is the semi subtractor. It can subtract a dual number (subtrahend **S**) from another dual number (minuend **M**).

Experiment procedure:

- Complete the table 6.2.4.1 under consideration of the rules of arithmetic for subtraction. The abbreviations **D** and **E** stand for difference and a borrow output.
- State the necessary function equations for the circuit.
- Two different semi subtractors can be realized with the Digital Training System. Complete the circuits in fig. 6.2.4.1 and 6.2.4.2.

M	S	D	E
0	0		
0	1		
1	0		
1	1		

Table 6.2.4.1

D =

E =

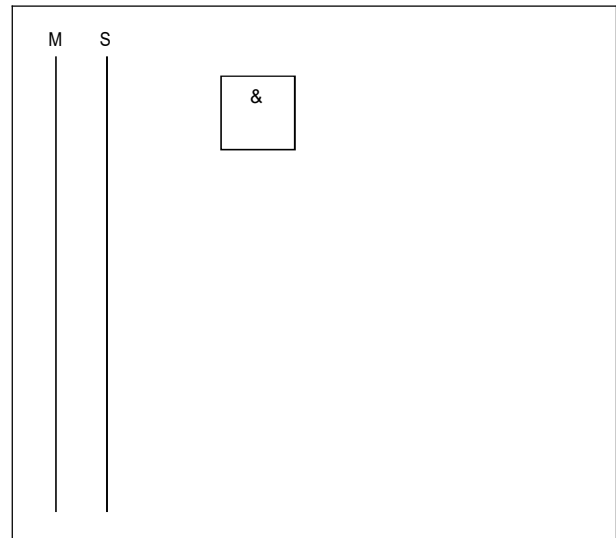


Fig. 6.2.4.1 Circuit 1

Question 1: Semi subtractors can only be used restrictedly. Give reasons.

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

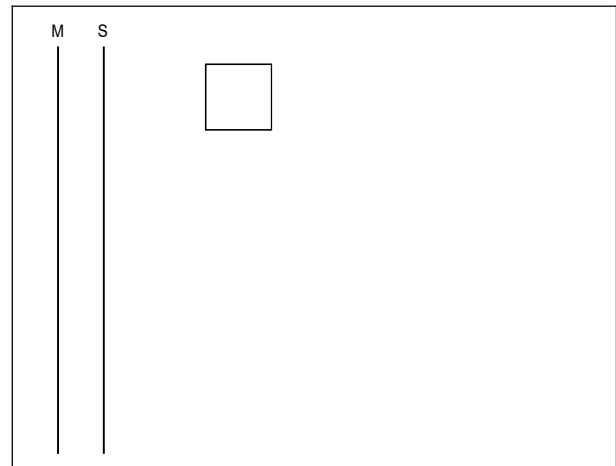


Fig. 6.2.4.2 Circuit 2

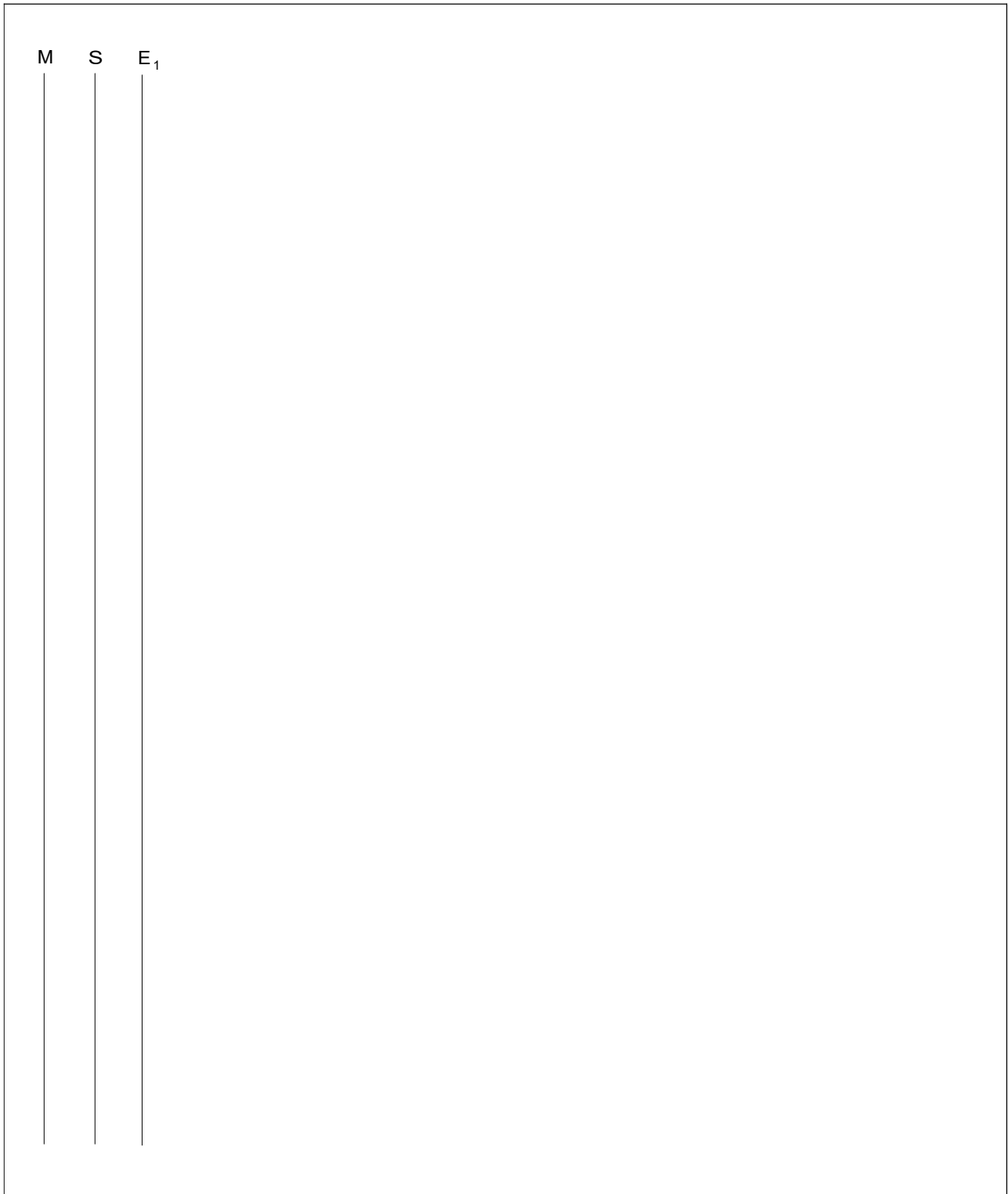


Fig. 6.2.5.3 Circuit of a 1-bit full subtractor

□ Experiment 2: Circuit variations of the 1-bit full subtractor

If the difference formation is considered mathematically, two other circuit variations are possible with the Digital Training System.

N. B.:

The sum of $S + E_1$ is formed by a **semi adder**, the resultant **subtotal** then allows the **difference** to be formed with a **semi subtractor**.

$$\text{Difference} = \text{minuend} - (\text{subtrahend} + \text{borrow})$$

$$D = M - (S + E_1)$$

Experiment procedure:

- Design the first circuit version from the mathematical equation and complete fig. 6.2.5.4.
- A further variation of the 1-bit full subtractor is recognizable from the above equation. Complete fig. 6.2.5.5 and give reasons for your solution.



Fig. 6.2.5.4 Circuit variation 1

Reasons:

.....

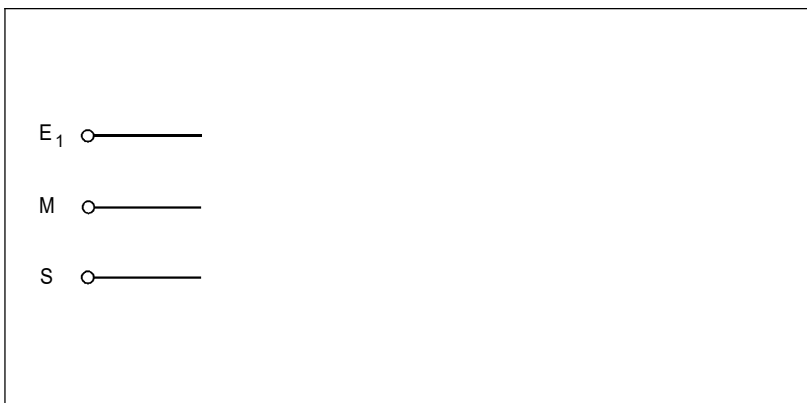


Fig. 6.2.5.5 Circuit variation 2

6.2.7 2-bit Parallel Multiplication Circuit

Experiment 1:

Every AND element can be considered as a 1-bit multiplier because the following applies:

- $0 \cdot 0 = 0$
- $0 \cdot 1 = 0$
- $1 \cdot 0 = 0$
- $1 \cdot 1 = 1$

Two-digit dual numbers can be processed with a 2-bit parallel multiplication circuit.

Experiment procedure:

- Show the multiplication of the digits $A = 2$ and $B = 3$ in binary form. Two summands are produced. Mark these.
- Every summand is given by two 1-bit multipliers, i. e. AND elements. Complete the circuit in fig. 6.2.7.1 and enter the values for your multiplication task.
- Mark the both summands in fig. 6.2.7.1 because these have to be added in correct order.
- Consider which adding components the arithmetic task requires and complete fig. 6.2.7.2.
- Then set up the complete multiplication circuit using a 4-bit full adder.
Note: $P_2 = P_3 = Q_2 = Q_3 = 0$
- Check the function of the circuit with the Digital Training System. Do this using different multiplication tasks.
- Then draw the complete circuit from the Digital Training System (fig. 6.2.7.3, page 84) and enter the values for the maximum task to be multiplied.

$$A_1 A_0 \cdot B_1 B_0$$

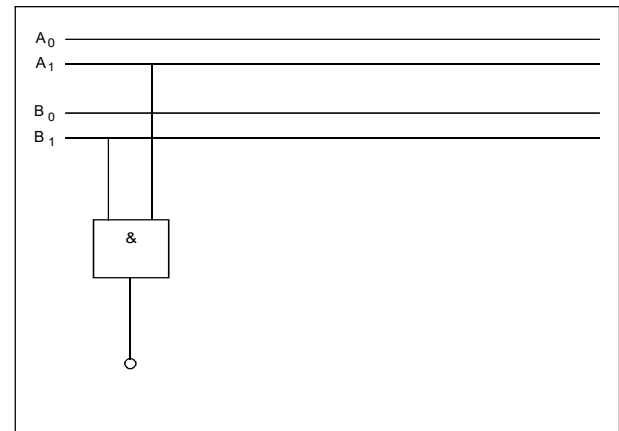


Fig. 6.2.7.1

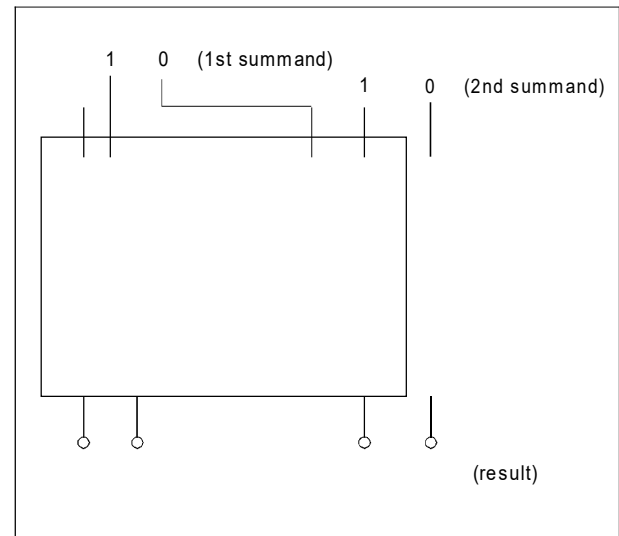


Fig. 6.2.7.2

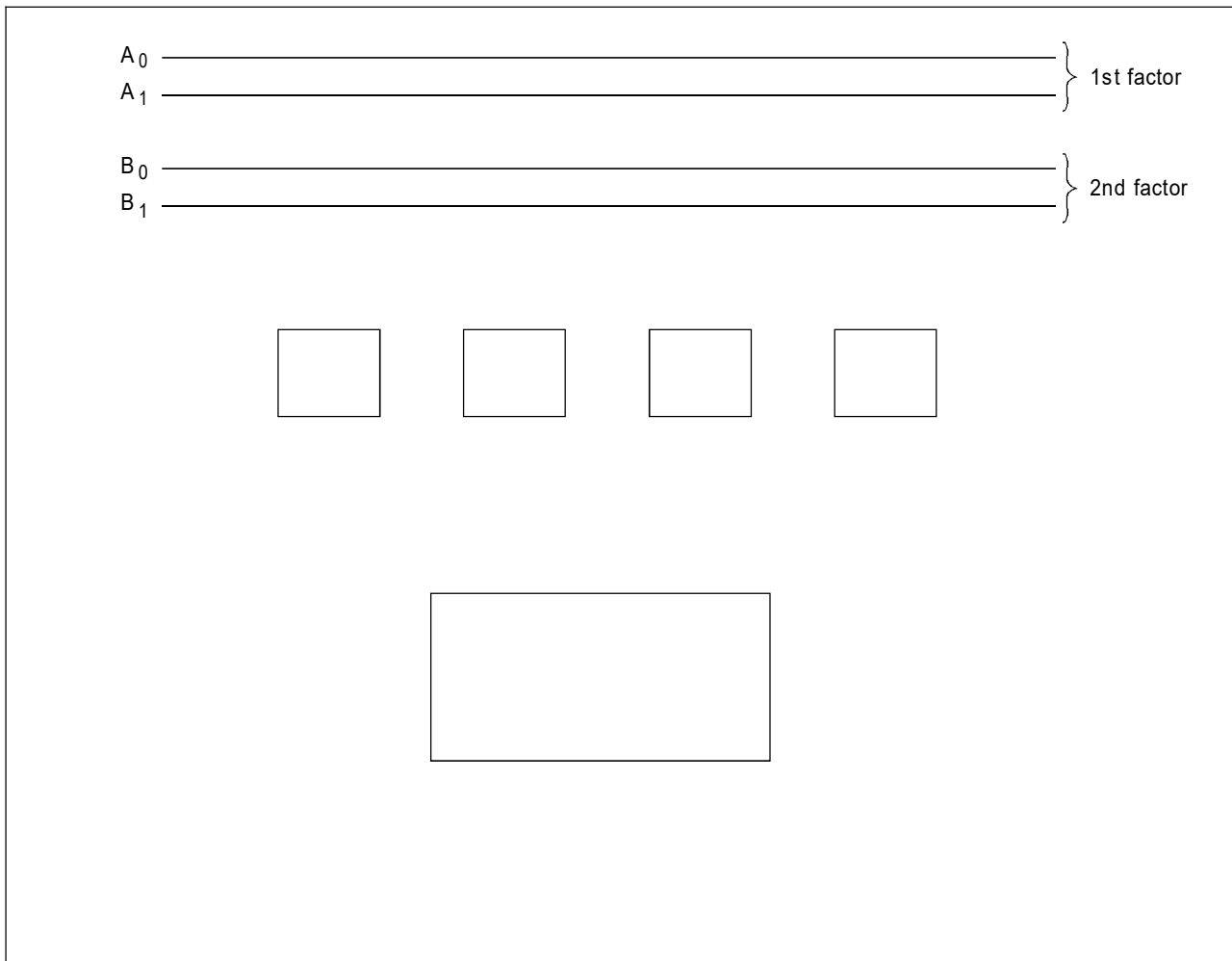


Fig. 6.2.7.3 2-bit parallel multiplication circuit

Notes:

6.2.8 Arithmetic Unit for 4-bit Dual Numbers

□ Experiment 1:

The circuit in fig. 6.2.8.1 shows an arithmetic unit for linking 4-bit dual numbers.

The module D9 contains a 4-bit 1s complement.

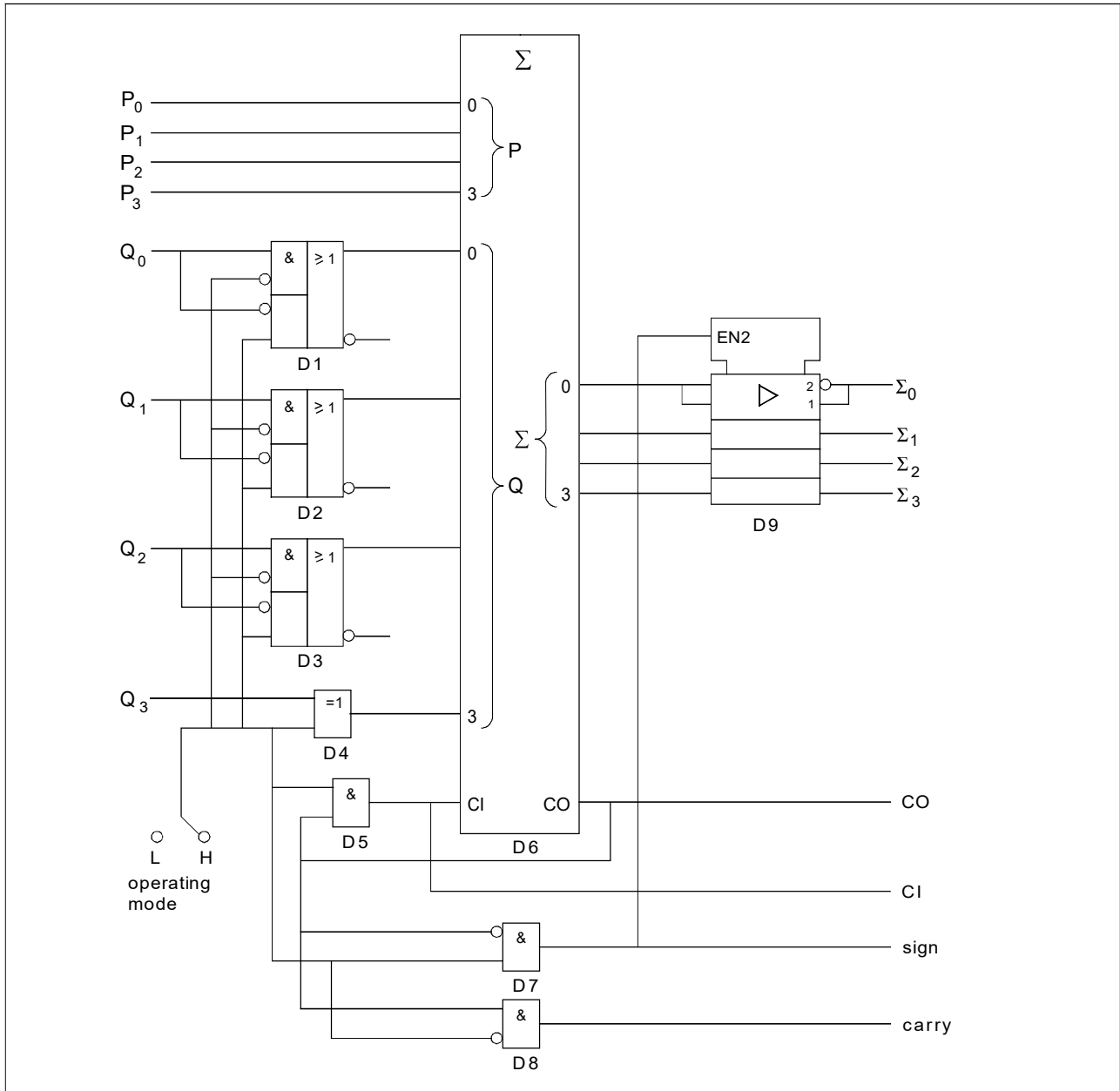


Fig. 6.2.8.1 Circuit of an arithmetic unit for 4-bit dual numbers

Experiment procedure:

- Examine the function of the circuit in fig. 6.2.8.1 by completing table 6.2.8.1 on the basis of the numbers specified for P and Q.
- Then test the circuit with the Digital Training System.

Level for operating mode	Number P				Number Q				CO	Σ_3	Σ_2	Σ_1	Σ_0	CI	Sign	Carry	Executed arithmetic operations
	P3	P2	P1	P0	Q3	Q2	Q1	Q0									
L	0	1	0	1	0	1	0	0									
L	1	0	0	0	0	1	1	0									
L	1	0	0	1	0	1	1	0									
L	1	1	1	1	1	1	1	0									
L	0	0	0	0	0	0	0	0									
H	1	0	0	0	0	0	1	1									
H	1	1	1	1	1	0	0	0									
H	0	0	1	1	1	0	0	0									
H	1	0	0	1	1	1	1	1									
H	0	1	1	1	0	1	1	1									
H	0	0	0	0	0	0	0	0									

Table 6.2.8.1

Question 1: Which arithmetic operation is selected by the H level (operating mode)?

Answer:

.....

Question 2: How is a negative result recognizable?

Answer:

.....

.....

.....

.....

Question 3: What is the difference in addition and subtraction of the numbers P = 0 and Q = 0? Give reasons.

Answer:

.....

.....

.....

.....

.....

7. Counting Circuits

7.1 Fundamental Principles

7.1.1 General

A digital counter is a circuit consisting of a certain linking of flipflops.

Counters have a clock input and an output on every flipflop. If the individual outputs are assigned as a valence e. g. the weights of the dual numbers ($1 = 2^0$, $2 = 2^1$, $4 = 2^2$, $8 = 2^3$, etc.), the counter results are displayed directly in dual code.

Digital counters can be classified as follows:

- according to their **function** as asynchronous and synchronous counters
- according to their **counting direction** as up and down counters
- according to their **coding** of the counter result as dual and BCD counters

7.1.2 Asynchronous Counters

The simplest structure of asynchronous counters is with T flipflops. Asynchronous counters can also be set up with master-slave flipflops although the master-slave function is not necessary for counters.

The flipflops are triggered one after the other when the counter state changes. Every flipflop generates its own clock signal for the next flipflop.

An important precondition for correct pulse counting is the initial position of all flipflops, the reset state (counter state 0). In most counting circuits, clock-independent R inputs are used on the individual flipflops for this.

Fig. 7.1.2.1 shows an example of an asynchronous up counter in dual code.

After every clock pulse, a dual number incremented by „1“ is available at the outputs in the up counter. If the dual number is to be „1“ lower after every clock, a down counter is required.

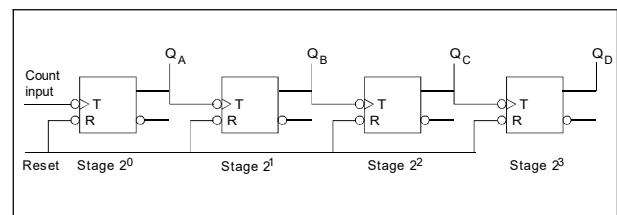


Fig. 7.1.2.1 Asynchronous up counter (dual code)

Whether a counting stage counts up or down depends on the circuiting of the flipflops. The circuit operates as an up counter when the Q outputs are wired to the clock inputs. If, however, the negated flipflop outputs are wired to the following clock inputs, the counting direction is reversed and a down counter is obtained.

Fig. 7.1.2.2 shows the principle of an up / down counter.

In most circuits the counting direction is set by a digital switch (fig. 7.1.2.3 and table 7.1.2.1).

A counter which counts both up or down is known as an up / down counter or a **reversing counter**.

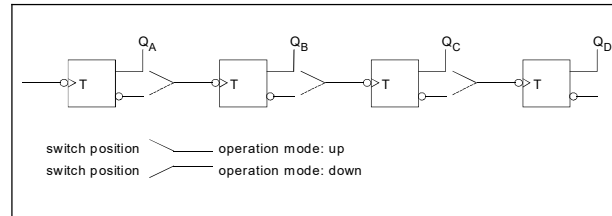


Fig. 7.1.2.2 Up / down counter

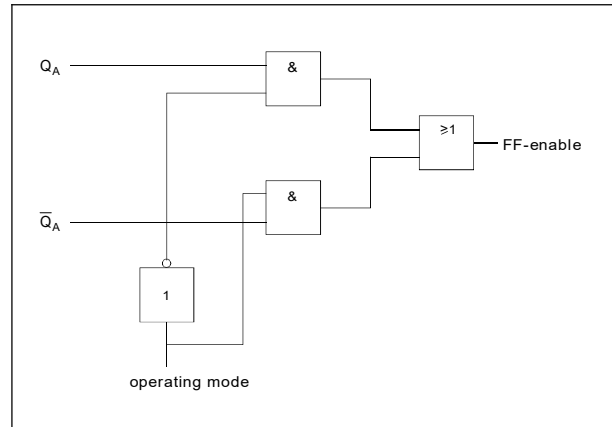


Fig. 7.1.2.3 Digital switch

Operating mode	FF-enable	Counting direction
L	Q_A	up
H	\bar{Q}_A	down

Table 7.1.2.1

7.1.3 Synchronous Counters

In the asynchronous counters, a new number is not available exactly until all flipflops of the counter module have switched one after the other. This time delay increases from flipflop to flipflop in the counting chain with the result that at high counting frequencies the circuit can no longer operate without error. Independence of the highest permissible counting frequency from the number of counter flipflops can only be achieved if all flipflops can switch at the same time, i. e. are clocked **simultaneously** (synchronously). These counters are therefore called synchronous counters. The clock input is fed simultaneously to all flipflops in synchronous counters.

Fig. 7.1.3.1 shows the required logic behaviour of a synchronous 3-bit up counter.

after clock	4	Position 2	1	Number
Start position	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7
8	0	0	0	0
	C	B	A	

Fig. 7.1.3.1 Required logic behaviour

Derived conditions:

- **Position 1** switches after every clock.
- **Position 2** switches when position 1 is logic „1“.
- **Position 4** switches when positions 1 and 2 are logic „1“.

To avoid all flipflops being switched when activated simultaneously, „1“ may only be applied to inputs J and K (T function) if switching is necessary. Locking of the JK inputs can be determined algebraically according to the conditions derived from fig. 7.1.3.1.

Fig. 7.1.3.2 shows the circuit for a synchronous 3-bit up counter.

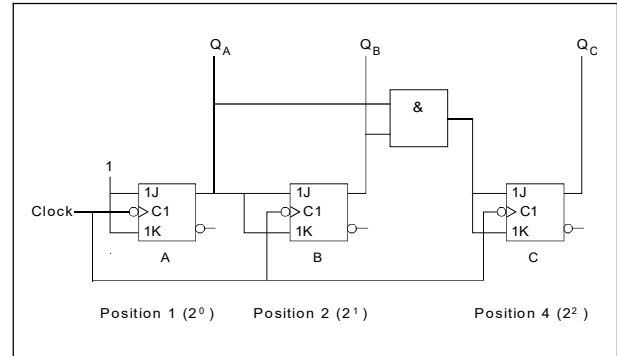


Fig. 7.1.3.2 Synchronous 3-bit up counter

Algebraic calculation:

Flipflop A (position 1):

Flipflop A is set (transition from 0 - 1) for the following number values:

$$\begin{aligned}
 J_A &= 0 \vee 2 \vee 4 \vee 6 \quad \rightarrow \\
 J_A &= \bar{C} \bar{B} \bar{A} \vee \bar{C} B \bar{A} \vee C \bar{B} \bar{A} \vee C B \bar{A} \\
 &= [(\bar{C} \bar{B} \vee C B) \bar{A}] \vee [(C \bar{B} \vee C B) \bar{A}] \\
 &= [1 \wedge \bar{A}] \vee [1 \wedge \bar{A}] \\
 &= \bar{A} \vee \bar{A}
 \end{aligned}$$

$$J_A = \bar{A}$$

Flipflop B is reset (transition from 1 - 0) for the following number values:

$$\begin{aligned}
 K_A &= 1 \vee 3 \vee 5 \vee 7 \quad \rightarrow \\
 K_A &= \bar{C} \bar{B} A \vee \bar{C} B A \vee C \bar{B} A \vee C B A
 \end{aligned}$$

With the same simplification as for J_A it follows that:

$$K_A = A$$

Flipflop B (position 2):

Flipflop B is set (transition from 0 - 1) for the following number values:

$$\begin{aligned}
 J_B &= 1 \vee 5 \quad \rightarrow \\
 J_B &= \bar{C} \bar{B} A \vee C \bar{B} A = (\bar{C} \wedge C) \vee \bar{B} A \\
 &= 0 \vee \bar{B} A
 \end{aligned}$$

$$J_B = \bar{B} A$$

Flipflop B is reset (transition from 1 - 0) at the following number values:

$$\begin{aligned} K_B &= 3 \vee 7 \rightarrow \\ K_B &= \overline{C} B A \vee C B A = (\overline{C} \wedge C) \vee B A \\ &= 0 \vee B A \\ \mathbf{K_B} &= \mathbf{B A} \end{aligned}$$

Flipflop C (position 4):

Flipflop C is set (transition from 0 - 1) at the following number values:

$$\begin{aligned} J_C &= 3 \rightarrow \\ \mathbf{J_C} &= \mathbf{\overline{C} B A} \end{aligned}$$

Flipflop C is reset (transition from 1 - 0) at the following number values:

$$\begin{aligned} K_C &= 7 \rightarrow \\ \mathbf{K_C} &= \mathbf{C B A} \end{aligned}$$

Table 7.1.3.1 shows a summary of the locks of the JK inputs determined up to this point.

The function equations determined up till now in table 7.1.3.1 can be further simplified. This is explained in fig. 7.1.3.3 with flipflop A as an example.

Input J must be in the reset state ($\overline{A} = 1$) if it is to set the flipflop. It then follows that the value „1“ can be entered for all negated variables of the own flipflop at the J inputs.

The same applies accordingly for the reset input K. The value „1“ can be entered for all **non negated** variables of the own flipflop at the K inputs.

Table 7.1.3.2 shows the simplification of the locks in table 7.1.3.1.

7.1.4 Modulo-n Counters

Counters are often required which count up to a desired number value then reset to „0“ and start counting again from the beginning or stop and wait for a new start signal. The modulo-10 counter is used most frequently for displaying BCD numbers in dual code. This is a 4-bit dual counter with only 10 numeric values. It counts up to „9“ and then starts again at „0“. Consequently there

Summary		
$J_C = \overline{C} B A$	$J_B = \overline{B} A$	$J_A = \overline{A}$
$K_C = C B A$	$K_B = B A$	$K_A = A$

Table 7.1.3.1

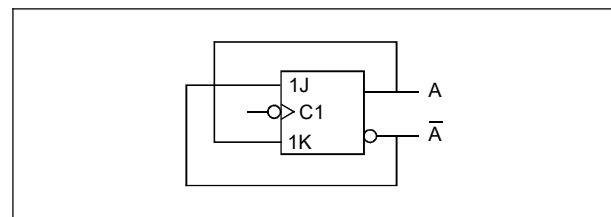


Fig. 7.1.3.3

Simplified summary		
$J_C = B A$	$J_B = A$	$J_A = 1$
$K_C = B A$	$K_B = A$	$K_A = 1$

Table 7.1.3.2

is always a dual number in 8421-BCD code at the outputs. 8421-BCD counters are also known as **decimal counters**. The functional principle of a modulo-10 counter is that a 4-bit dual counter is reset at the instant at which the counter switches to 1010. After switching, the reset signal must become inactive (passive) immediately so that the counting process can be continued with the value 0000.

8241-BCD up counters count from 0 ... 9. Since each counter corresponds to one decade, two counters of this type connected in series can count to „99“, three to “999” and so on. Fig. 7.1.4.1 and table 7.1.4.1 show an asynchronous modulo-6 counter and the corresponding function table.

Since Q_B and Q_C have the same value „1“ for the first time for display 110 (after 6 pulses), it is sufficient to evaluate this event. Evaluation of the output Q_A could be dispensed with here too.

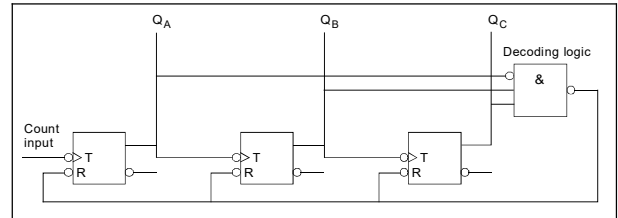


Fig. 7.1.4.1 Asynchronous modulo-6 counter

Pulse	Q_C	Q_B	Q_A	Counter reading
Start position	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
Start position	0	0	0	0

Table 7.1.4.1

7.1.5 Programmable Counters

Programmable counters are counters which accept a binary value at one of the inputs with a control pulse. The counter component in fig. 7.1.5.1 can count up or down in binary code.

Function:

- **Up counter:** clock input 2+ / G1, input G2 positive (positive = unswitched state)
- **Down counter:** clock input 1- / G2, input G1 positive
- **Reset:** input CT=0
- For **presetting**, the appropriate number is applied to the inputs 3D in binary code. If L level is then applied to the load input C3, the counter passes the applied bit combination to the output.
- The **loading process** is independent of clock 2+ / G1 and 1- / G2.
- When the counter reaches the value „15“ when counting up or the value “0” when counting down, the appropriate outputs ($\overline{1}CT=15$ or $\overline{2}CT=0$) are set to L level with negative clock edge.

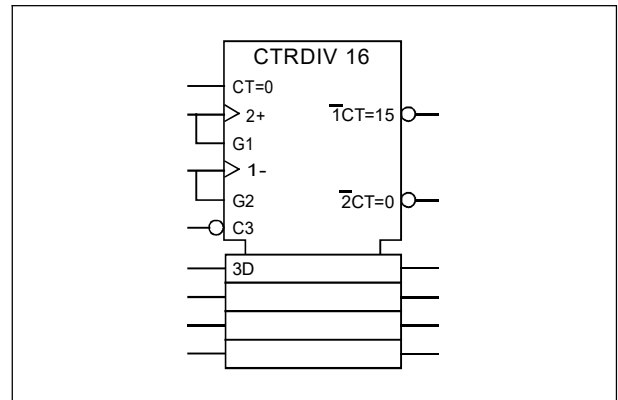


Fig. 7.1.5.1 Binary counter

7.2 Experiments Section

7.2.1 Asynchronous Up Counters

□ Experiment 1: 4-bit dual up counter

Experiment procedure:

- Set up the circuit in fig. 7.2.1.1 with the JK flipflops of the Digital Training System. Connect every Q output to a LED.
- Transmit a clock pulse to the counting input with the bouncefree key.
- **Note:** Connect the reset line to a Q output of the input keyboard. All flipflop outputs can be reset with the reset signal.
- Draw the appropriate levels in the following pulse diagram (fig. 7.2.1.2).
- Then determine the corresponding dual numbers and counter readings.

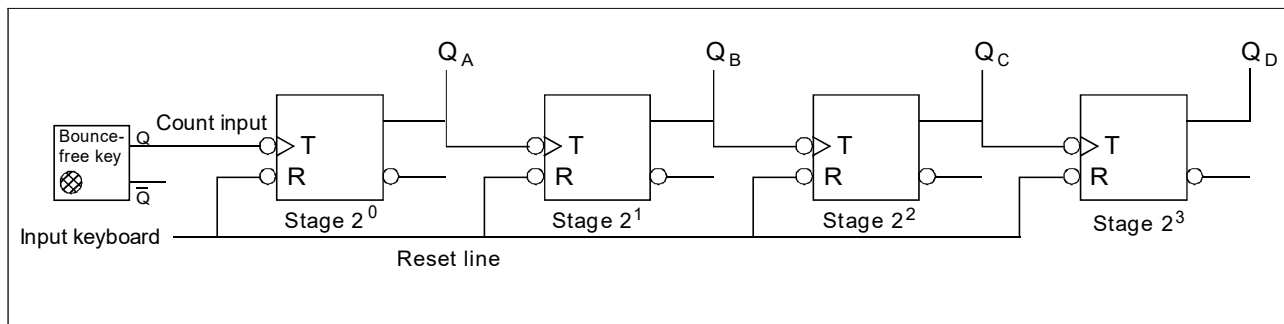


Fig. 7.2.1.1 Circuit of a 4-bit dual up counter

		Time intervals after n pulses																	Time →	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		17
Counting pulses		[Pulse diagram showing a regular square wave sequence]																		
Valence	1	Q _A	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
	2	Q _B	H	H	L	L	H	H	L	L	H	H	L	L	H	H	L	L	H	H
	4	Q _C	H	H	H	L	L	L	H	H	H	L	L	L	H	H	H	L	L	L
	8	Q _D	H	H	H	H	L	L	L	L	L	H	H	H	H	L	L	L	L	L
Dual number		0000																		
Counter reading		0																		

Fig. 7.2.1.2 Pulse diagram of a 4-bit dual up counter

Question 1: How does the 4-bit dual up counter behave after 16 counted pulses?

Answer:

Question 3: Every counter is also a frequency divider. A single flipflop generates a frequency division of 2 : 1. How great is the division ratio in a 4-bit dual counter?

Answer:

Question 2: How many pulses can be counted with an n-bit dual counter?

Answer:

Notes:

7.2.2 Asynchronous Down Counters

□ Experiment 1: 4-bit dual down counter

Experiment procedure:

- Complete the flipflop drive circuit in fig. 7.2.2.1 so that an asynchronous down counter is produced.
- Set up the circuit with the JK flipflops of the Digital Training System. Connect every Q output to a LED.
- Set the Q output of every flipflop to „1“.
- Transmit a clock pulse to the counting input of the circuit and draw the appropriate levels in the following pulse diagram (fig. 7.2.2.2).
- Then determine the corresponding dual numbers and counter readings.

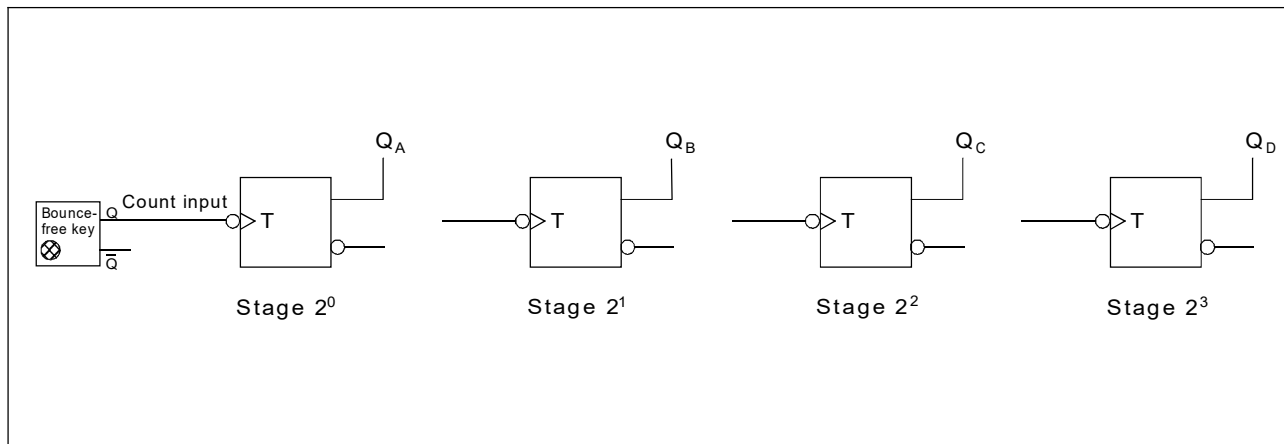


Fig. 7.2.2.1 Circuit of a 4-bit dual down counter

		Time intervals after n pulses																	Time →	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		17
Counting pulses		[Pulse diagram showing a series of 17 square pulses]																		
Valence	1	Q _A	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
	2	Q _B	H	H	L	L	H	H	L	L	H	H	L	L	H	H	L	L	H	H
	4	Q _C	H	H	H	L	L	L	H	H	H	L	L	L	H	H	H	L	L	L
	8	Q _D	H	H	H	H	L	L	L	L	L	H	H	H	H	L	L	L	L	L
Dual number		1111																		
Counter reading		15																		

Fig. 7.2.2.2 Pulse diagram of a 4-bit dual down counter

Question 1: How many pulses has the 4-bit dual down counter received when it reads „0”?

Answer:

.....

.....

.....

.....

Question 2: In TTL circuits the signal propagation time is up to 50 ns. This delay increases from flipflop to flipflop in the counting sequence. Calculate up to what counting frequency a 12-bit asynchronous counter operates without error.

Answer:

.....

.....

.....

.....

Notes:

7.2.3 Asynchronous Reversing Counters

□ Experiment 1: 3-bit reversing counter

Experiment procedure:

- Complete the digital switches for driving the flipflops in fig. 7.2.3.1.
- Set up the circuit in fig. 7.2.3.1 with the Digital Training System. Connect the outputs of the flipflops to the appropriate inputs of the HEX / 7SEG decoder so that the counter reading is in the left-hand display. Set the unused input 8 of the display to L level.
- Then apply a clock pulse of 1 Hz to the counting input of the circuit.
- Complete table 7.2.3.1.

Operating mode	Counting direction
L	
H	

Table 7.2.3.1

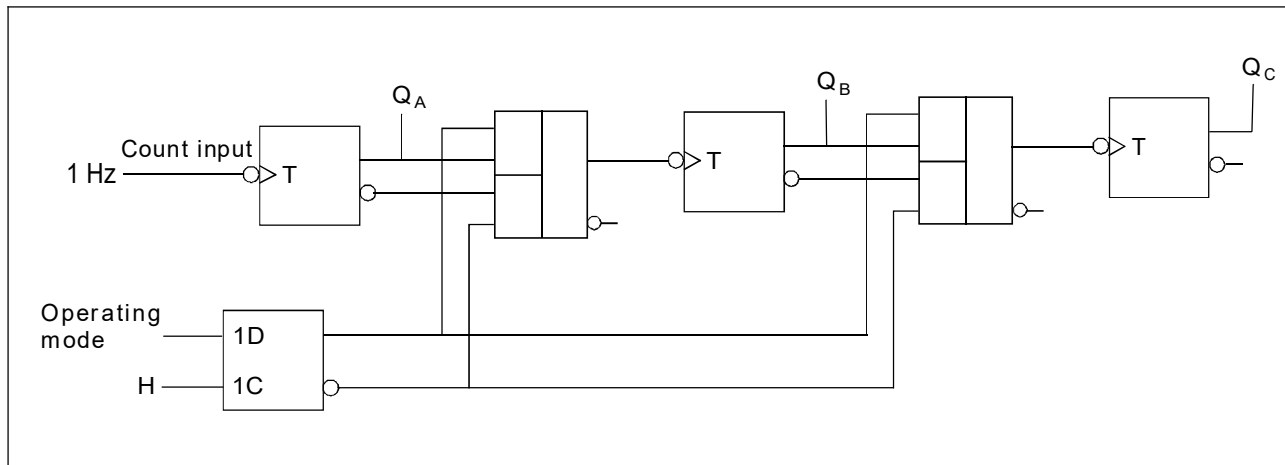


Fig. 7.2.3.1 Circuit of a 3-bit reversing counter

Question 1: How many pulses can the 3-bit reversing counter process before it is back in its initial state?

Answer:

.....

.....

.....

Question 2: Describe the function of the digital switches in fig. 7.2.3.1.

Answer:

.....

.....

.....

7.2.4 Asynchronous Modulo-n Counters

□ Experiment 1: Modulo-10 counter

Experiment procedure:

- Complete the drive for the flipflops in fig. 7.2.4.1 so that an asynchronous 4-bit up counter is produced.
- The counter should be reset on reaching the number „10" (1010_2). Complete the evaluation logic in fig. 7.2.4.1 which determines whether the binary number 1010 is applied to the flipflop outputs and then a reset signal generated.

- Set up the circuit with the Digital Training System. Connect every flipflop output to a LED. Reset all flipflop outputs Q with a reset signal.
- Apply a clock pulse to the counting circuit and complete table 7.2.4.1.

Note: Because of manufacturing problems in propagation time of the JK flip flops, the reset output of the AND gate has to be connected to four inverters for inquiring the meter indication. Don't draw this in the circuit diagram.

Pulse	Q _D	Q _C	Q _B	Q _A	Counter reading	Pulse	Q _D	Q _C	Q _B	Q _A	Counter reading
Start position	0	0	0	0	0	11					
1						12					
2						13					
3						14					
4						15					
5						16					
6						17					
7						18					
8						19					
9						20					
10											

Table 7.2.4.1

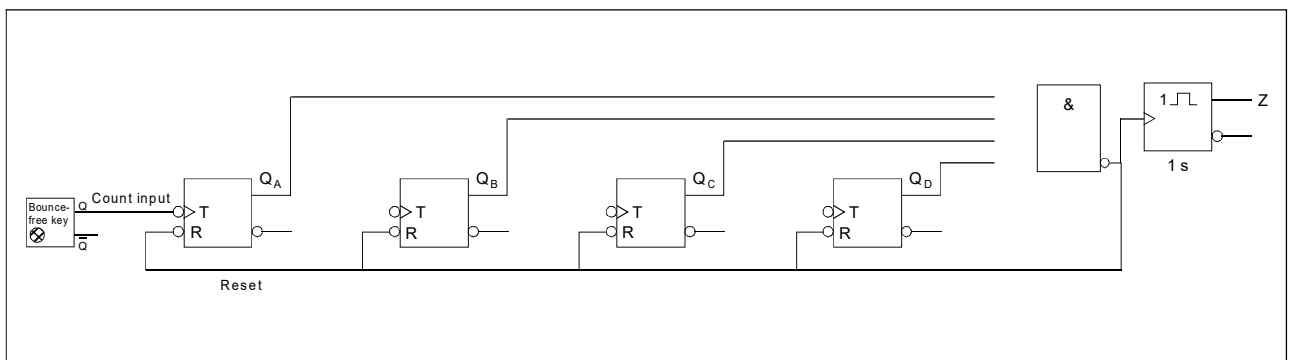


Fig. 7.2.4.1 Modulo-10 counter with practical application

Question 1: Why is counter reading „10“ not displayed?

Answer:
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Question 4: Not all flipflop outputs are always required for the evaluation logic to generate the reset signal. Which flipflop outputs are absolutely necessary for the evaluation logic in fig. 7.2.4.1? Give reasons for your answer.

Answer:
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Question 2: What is the highest counter reading which a modulo-n counter can reach?

Answer:
.....
.....
.....
.....
.....
.....
.....
.....

Question 3: How many counting pulses does it take for a modulo-n counter to reach its initial position?

Answer:
.....
.....
.....
.....

7.2.5 Synchronous Counters

Experiment 1: 4-bit synchronous counter

Experiment procedure:

- First complete the flipflop drive in the circuit of the 4-bit synchronous counter (fig. 7.2.5.1).
- To prevent all flipflops from switching when activated simultaneously, „1" may only be applied to the inputs J and K (T-function) when switching is necessary. A 4-bit synchronous counter must behave as in table 7.2.5.1.
- The following conditions for locking the JK inputs can be read from this table:
 - **Position 1** switches after every clock.
 - **Position 2** may only switch if position 1 is logic „1" ($Q_A = 1$).
 - **Position 4** may only switch if position 1 is logic „1" ($Q_A = 1$) and position 2 is logic „1" ($Q_B = 1$).
 - **Position 8** may only switch if position 1 is logic „1" ($Q_A = 1$) and position 2 is logic „1" ($Q_B = 1$) and position 4 is logic „1" ($Q_C = 1$).
- Complete the circuit in fig. 7.2.5.1 so that the counter behaves according to the conditions for positions 1, 2, 4 and 8.
- Check the circuit with the Digital Training System.

after clock	Position				Numeric value
	8	4	2	1	
Start position	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
etc.					

Table 7.2.5.1

Question 1: What is the typical feature in the synchronous counter with respect to the drive?

Answer:

.....

.....

.....

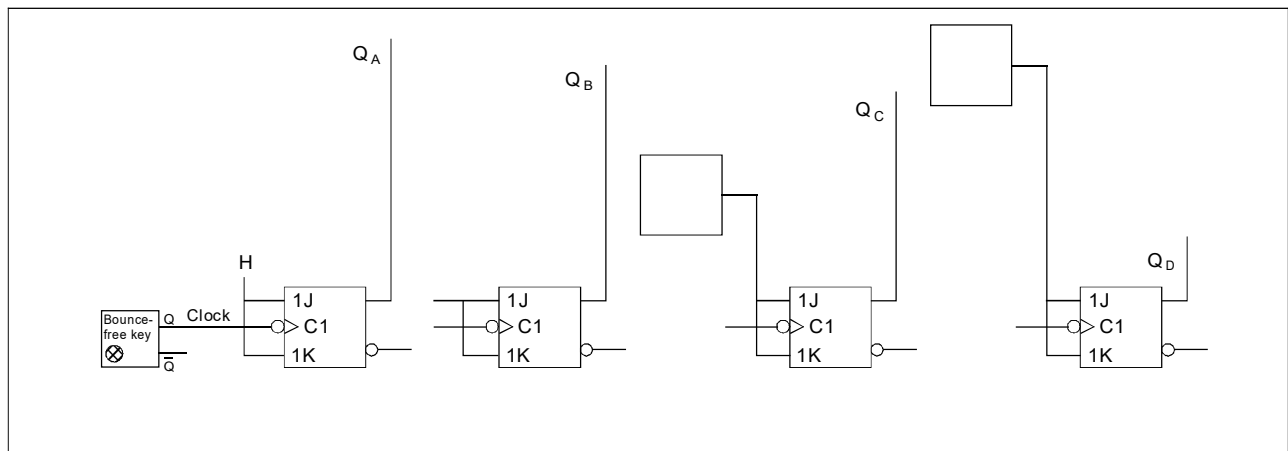


Fig. 7.2.5.1 4-bit synchronous counter

□ Experiment 2: Synchronous up counter in Aiken code

Experiment:

- Design a synchronous up counter for a counting decade (0 ... 9) in Aiken code (table 7.2.5.2).
- **Note:** Consider A here as the most significant position.
- Determine the function equations for inputs J and K of every flipflop (A ... D) with the aid of KV diagrams (fig. 7.2.5.3 ... 7.2.5.10). The bit pattern for the values 5 ... 10 do not appear and can be used as **pseudo-tetrad**s.
The transitions $0 \rightarrow 1$ must be taken into account for the setting inputs J and the transitions $1 \rightarrow 0$ for the resetting inputs K.
- Complete the circuit of the counter in fig. 7.2.5.2. Use the bouncefree key.
- Check the circuit with the Digital Training System.

Decimal number	Aiken code				Actual dual value
	A	B	C	D	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	1	0	1	1	11
6	1	1	0	0	12
7	1	1	0	1	13
8	1	1	1	0	14
9	1	1	1	1	15
0	0	0	0	0	0

Table 7.2.5.2 Aiken code

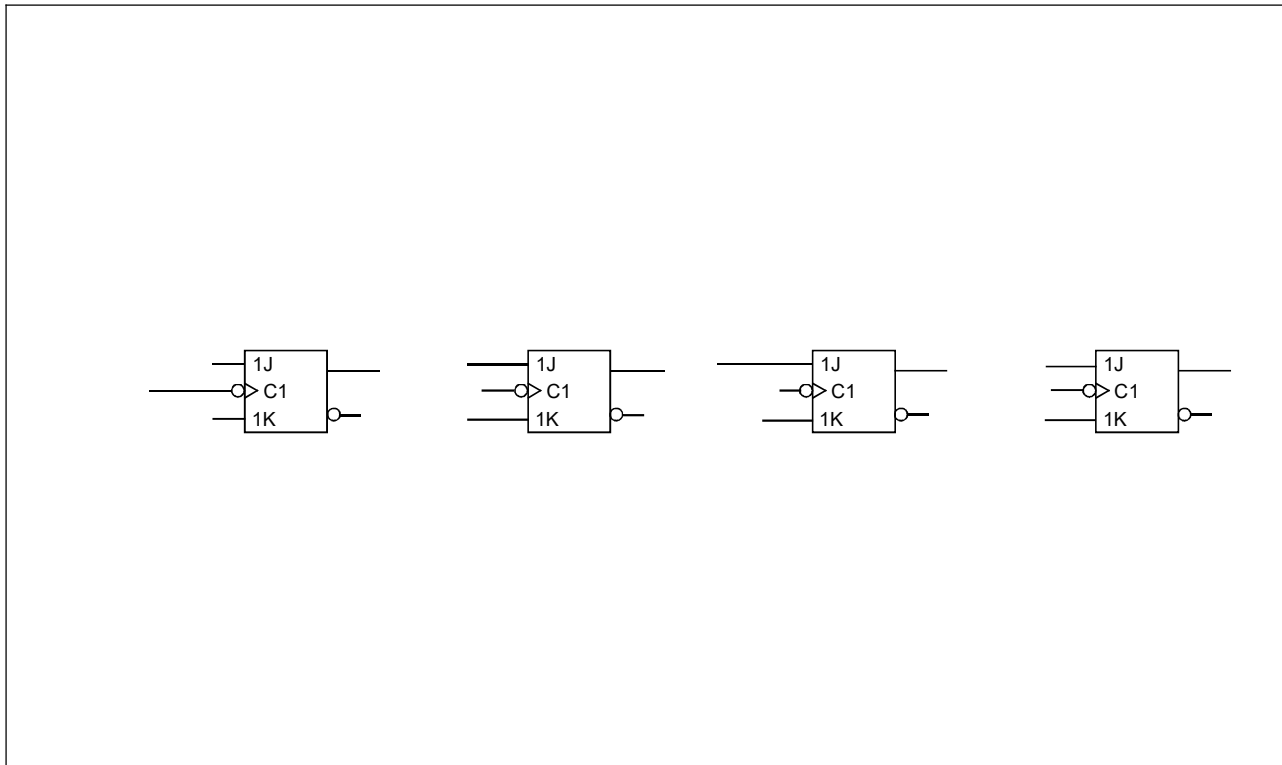


Fig. 7.2.5.2 Circuit of a synchronous 4-bit Aiken up counter

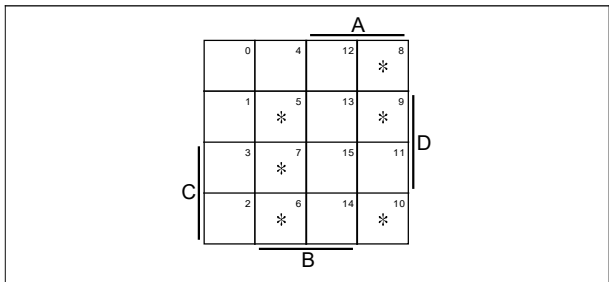


Fig. 7.2.5.3 KV diagram for flipflop J_D

$J_D = \dots\dots\dots$

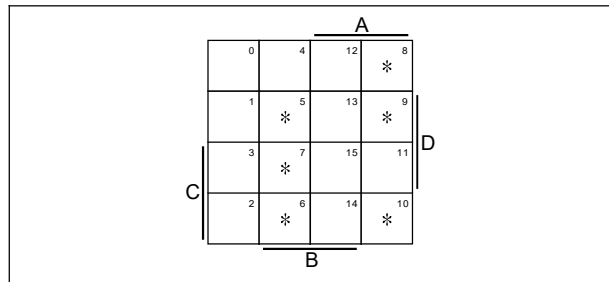


Fig. 7.2.5.7 KV diagram for flipflop K_D

$K_D = \dots\dots\dots$

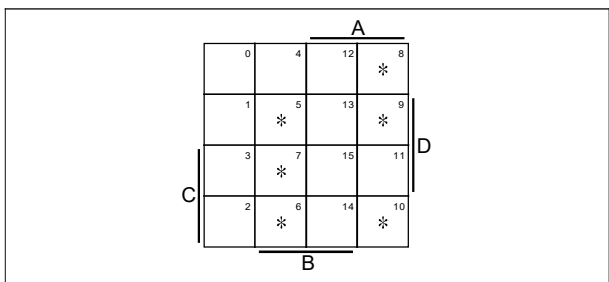


Fig. 7.2.5.4 KV diagram for flipflop J_C

$J_C = \dots\dots\dots$

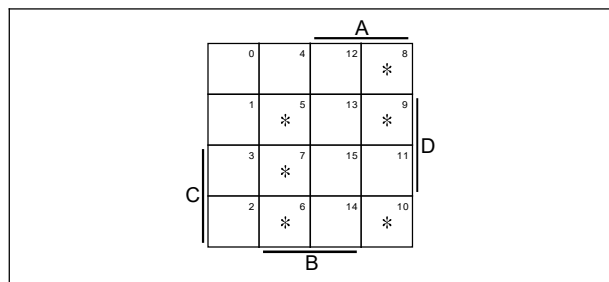


Fig. 7.2.5.8 KV diagram for flipflop K_C

$K_C = \dots\dots\dots$

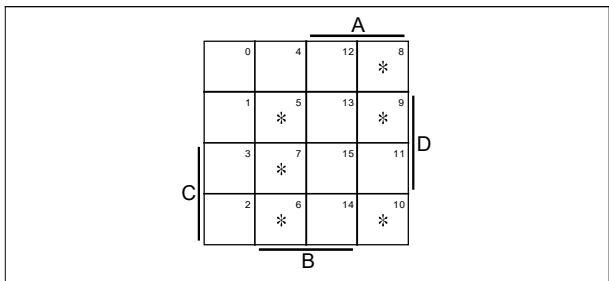


Fig. 7.2.5.5 KV diagram for flipflop J_B

$J_B = \dots\dots\dots$

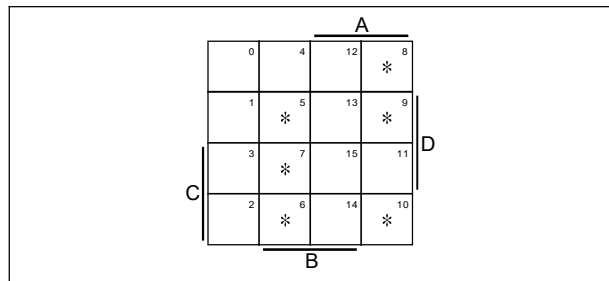


Fig. 7.2.5.9 KV diagram for flipflop K_B

$K_B = \dots\dots\dots$

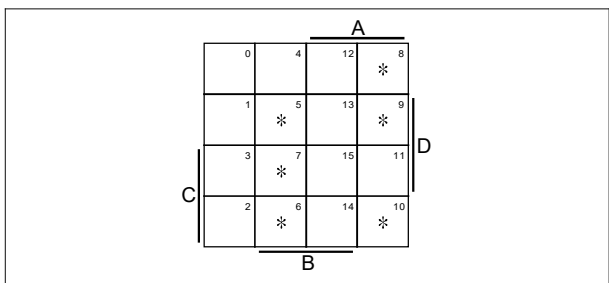


Fig. 7.2.5.6 KV diagram for flipflop J_A

$J_A = \dots\dots\dots$

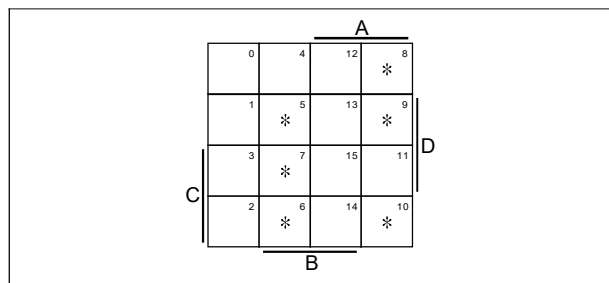


Fig. 7.2.5.10 KV diagram for flipflop K_A

$K_A = \dots\dots\dots$

7.2.6 Programmable Counters

□ Experiment 1: Modulo-n down counter

Experiment procedure:

- Design a modulo-n down counter, using component CTRDIV16 only, which counts down from the settable number n (n = 0 ... 9) to „0" and then starts again at n.
- First complete the circuit in fig. 7.2.6.1.
- Connect the outputs of the counter to the corresponding inputs of the HEX / 7SEG decoder so that the counter reading appears in the left-hand display.
- Apply a counting pulse of 1 Hz to the counting input of the circuit. Connect one side of the coding switch to the counter inputs.
- Test the circuit with the Digital Training System by setting different values and watching the counter.

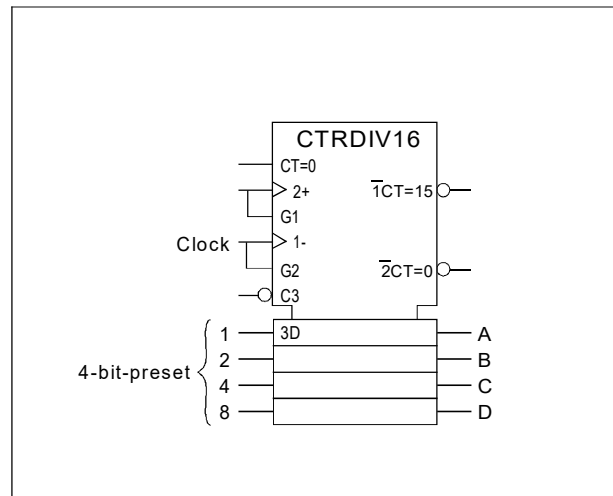


Fig. 7.2.6.1 Programmable counter CTRDIV16

Question 1: Can a modulo-n up counter also be set up with the same circuit (without any extra components)? Give reasons for your answer.

Answer:

.....

.....

.....

Notes:

□ Experiment 2: Digital clock

Experiment procedure:

- Design a circuit for a digital clock which has a counting cycle of 1 minute at a clock frequency of 1 Hz. The seconds should be displayed as a decimal value. Units and tens must be counted separately for this.
- **Note:** A modulo-10 counter is required for the BCD units position and a modulo-6 counter for the BCD tens position.
- Complete the necessary evaluation logic in fig. 7.2.6.2.
- Connect both modulo counters so that the BCD 10s position is clocked by the BCD 1s position.
- Connect the outputs of the counters to the inputs of the HEX / 7SEG display so that the seconds are displayed as a decimal number. Apply the unused input to L level.
- Check the function of the circuit with the Digital Training System.

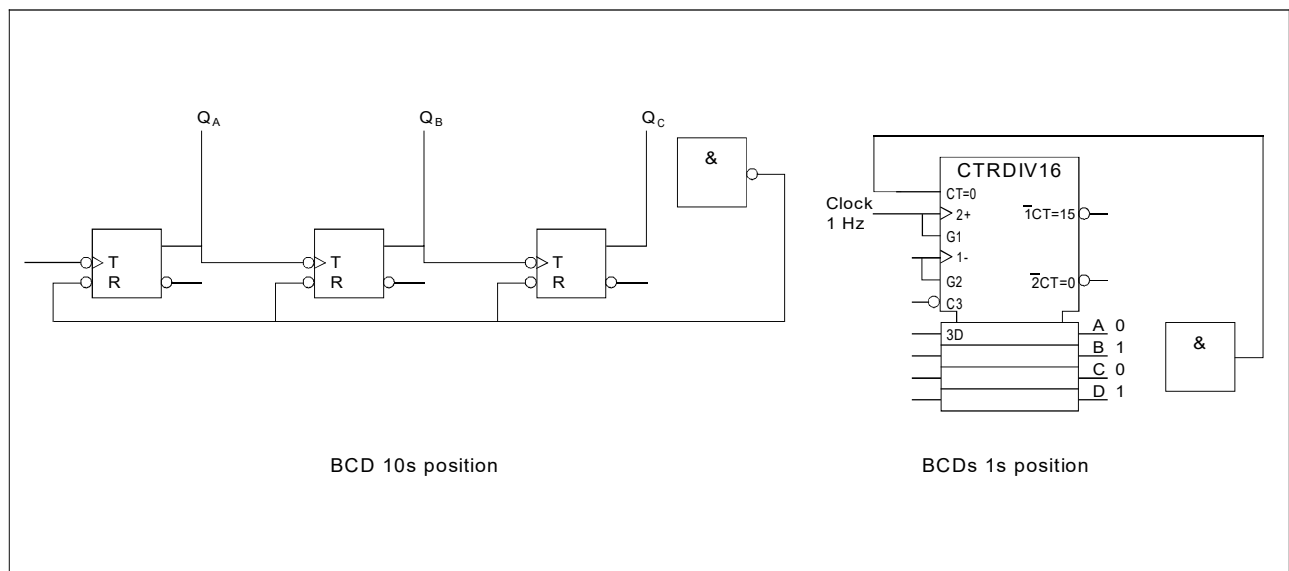


Fig. 7.2.6.2 Circuit for a digital clock

Notes:

8. Register Circuits

8.1 Fundamental Principles

8.1.1 General

In information technology the serial transfer of parallel data (fig. 8.1.1.1) and their re-conversion into a parallel form is encountered frequently. For example, remote data transmission through the network of the German Post Office is handled in serial form.

The converter for parallel data to serial data and vice versa must fulfill two tasks:

1. Buffering of the bits of a data word
2. Shifting of the individual bits through the transmission path

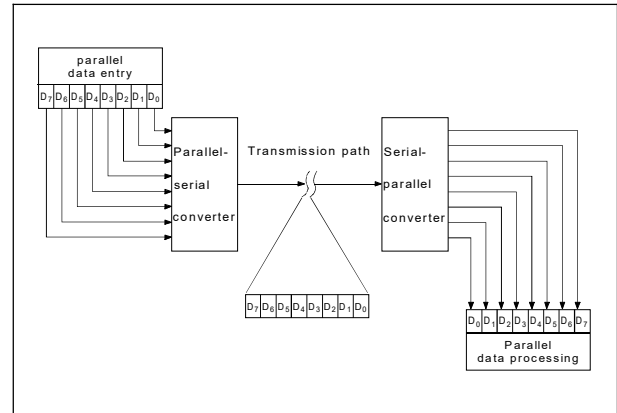


Fig. 8.1.1.1 Serial transfer of parallel information

8.1.2 Shift Registers

A switching unit which is able to store and shift information is known as a shift register.

A JK flipflop is used in each shift register for every bit of the data bit for storing the information.

Fig. 8.1.2.1 shows the basic circuit of a 4-bit shift register.

All flipflops are driven with a common clock pulse. The first flipflop is switched as a D-flipflop. The outputs Q and \bar{Q} of a flipflop are connected to the inputs J and K of the next flipflop.

Four different operating modes are defined in shift registers with respect to data input and data output (fig. 8.1.2.2, page 106).

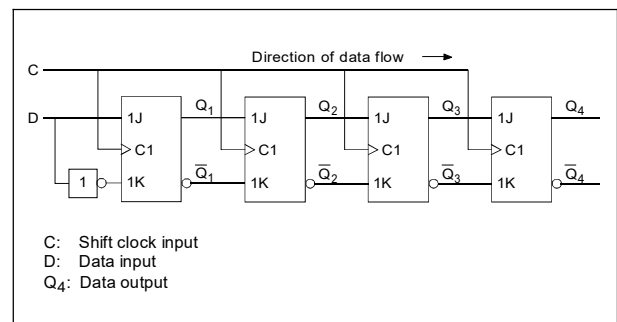


Fig. 8.1.2.1 Basic circuit of a 4-bit shift register

Different tasks can be assigned to the individual operating modes of the shift register:

- **1st operating mode:** serial memory, shift function
- **2nd operating mode:** serial-parallel converter (serial input, parallel output)
- **3rd operating mode:** parallel-serial converter (parallel input, serial output)
- **4th operating mode:** memory with parallel input and output

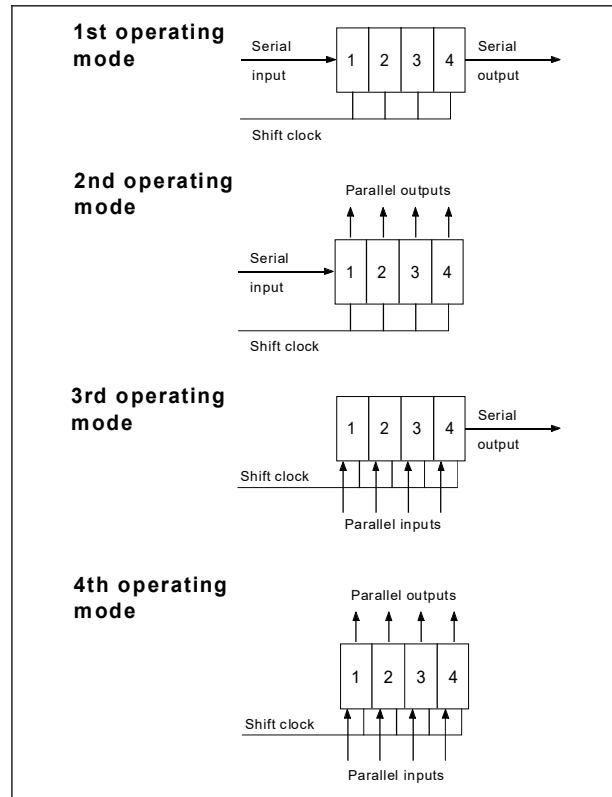


Fig. 8.1.2.2 Operating modes of shift registers

8.1.3 Universal Shift Registers

The universal shift register (fig. 8.1.3.1) can be used for many different applications. Serial input / output can be combined at will with parallel input / output. For this two address lines are required in the universal shift register which determine the operating mode according to table 8.1.3.1.

Function:

- All the flipflops can be reset through the static reset input R.
- **C4/1**→/2←: clock input for shifting (left/right) and parallel data input depending on the selected mode (0 ... 3)
- **1.4D:** serial input for right shift
- **2.4D:** serial input for left shift
- **3.4D:** inputs for parallel shift
- **Note:** The first digit at the inputs indicates the dependence on the operating mode.

Mode	S0	S1	Function
0	0	0	no Function
1	1	0	serial input, shift to right
2	0	1	serial input, shift to left
3	1	1	parallel input

Table 8.1.3.1 Operating modes

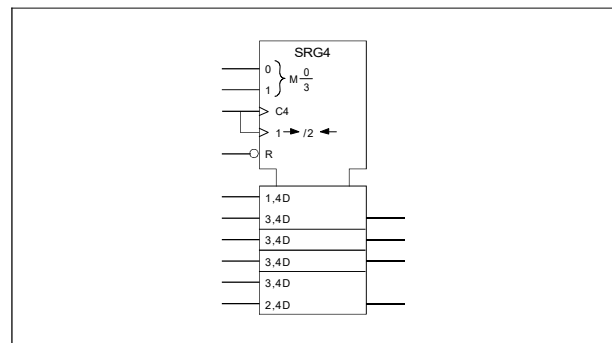


Fig. 8.1.3.1 Circuit symbol

8.2 Experiments Section

8.2.1 JK Shift Registers

□ Experiment 1: Examining the shift effect

Experiment procedure:

- Complete the circuit in fig. 8.2.1.1 so that a shift register is produced.
- Connect the Q outputs of the flipflops to LEDs. Reset all the flipflops if necessary and set the input D to logic „1“.
- Apply five single clocks to the clock input with the bouncefree key and note down your observations in table 8.2.1.1 every time you press the key.
- Set the input D according to the pulse diagram before every clock pulse (fig. 8.2.1.2, page 108). Trigger a clock pulse and enter the signal levels in the pulse diagram. Reset all the flipflops first if necessary.
- Mark the position of the input information 1100111 in the recorded pulse diagram for every output $Q_A \dots Q_D$.

Clock	D	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0	0
1	1				
2	1				
3	1				
4	1				
5	1				

Table 8.2.1.1

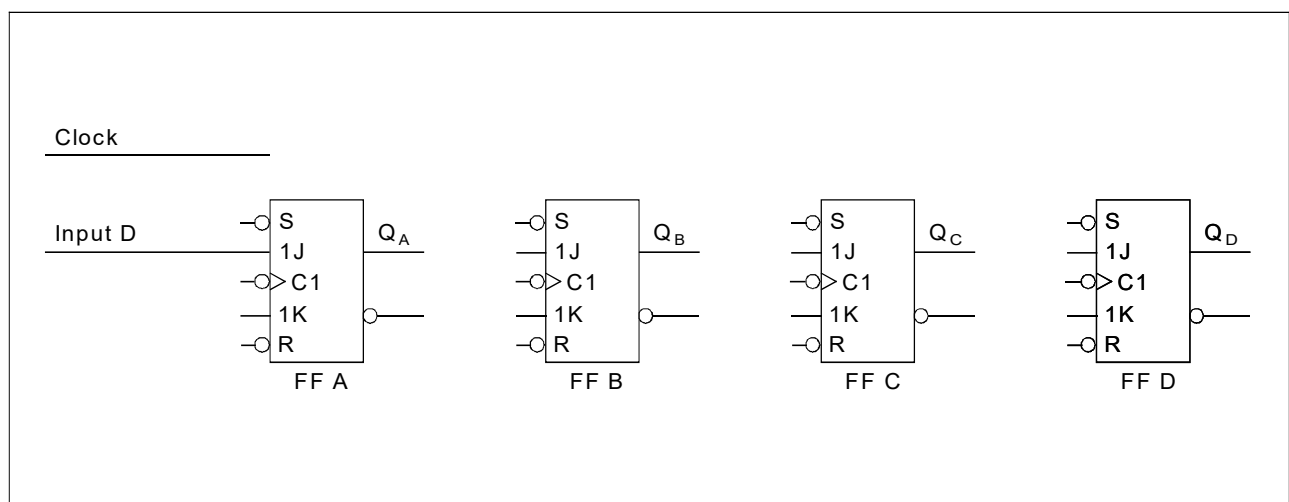


Fig. 8.2.1.1 Circuit of a JK shift register

8.2.2 Shift Registers with Parallel Data Input

Experiment 1: Set and reset inputs of a shift register

Binary data can only be read into a shift register if every flipflop can be set or reset **singly**.

Experiment procedure:

- Determine the function equations for the set input S and the reset input R for the value „0" from table 8.2.2.1.
- Complete the circuit in fig. 8.2.2.1.
- Check the circuit with the Digital Training System.

Read in E	Binary data D	Set input S	Reset input R	Function
0	0	1	1	Input blocked
0	1	1	1	
1	0	1	0	Reset
1	1	0	1	Set

Table 8.2.2.1

\bar{S} =

S =

\bar{R} =

R =

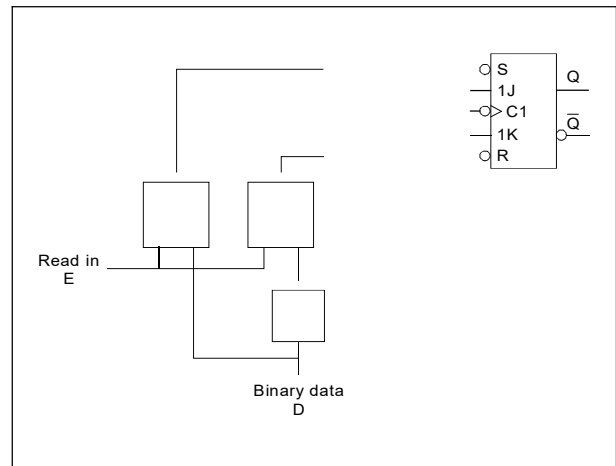


Fig. 8.2.2.1

Notes:

□ Experiment 2: 4-bit shift register with parallel data input**Experiment procedure:**

- Design a 4-bit shift register with parallel data input with the help of experiment 1 (page 109).
- Complete the circuit in fig. 8.2.2.2.
- **Note:** The clock inputs are not inverted here.

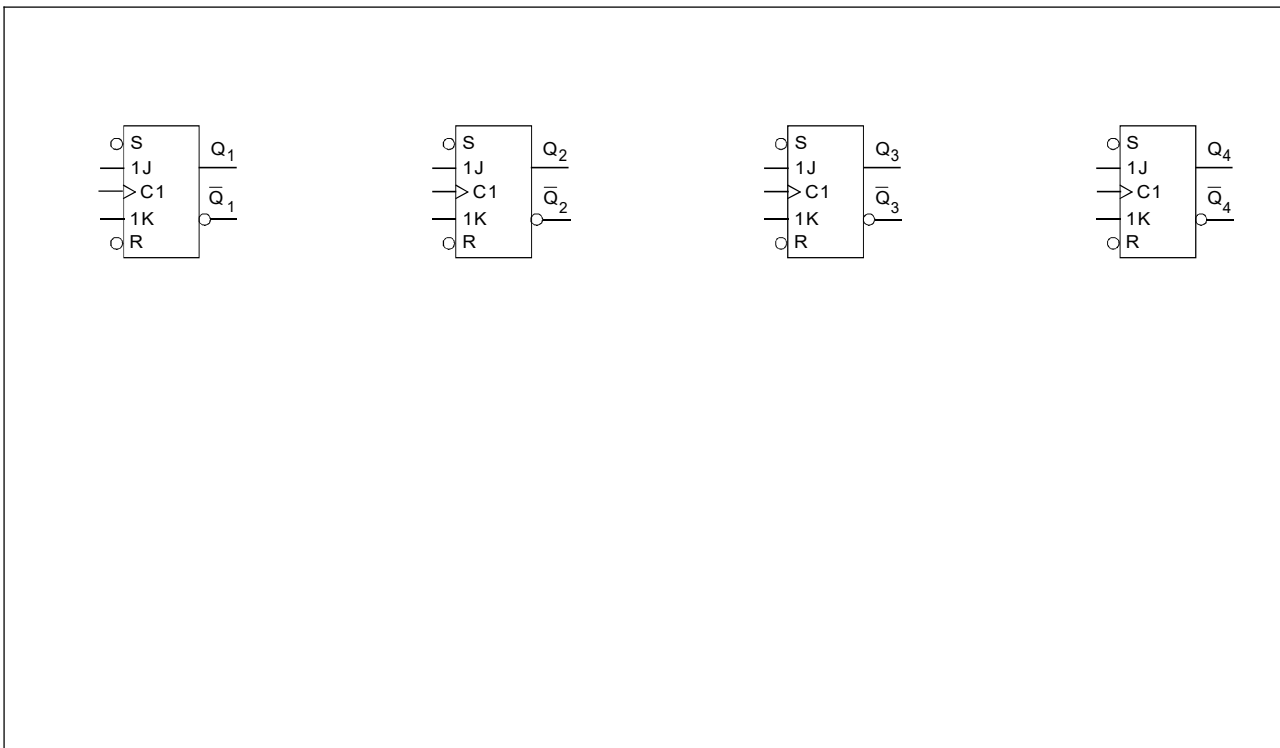


Fig. 8.2.2.2 Circuit of a 4-bit shift register with parallel data input

Notes:

8.2.3 Serial Data Transfer

□ Experiment 1: Serial 4-bit data transfer

In fig. 8.2.3.1 the transmitter and receiver (including buffer) of a modulo-5 counter are clocked. Four counting pulses are used for shifting. The fifth pulse leads to counter resetting and parallel transfer of the input data at the universal shift register.

Experiment procedure:

- Complete the JK flipflops in fig. 8.2.3.2 (page 112) to form a 4-bit shift register and mark the data transmission path.
- Connect the coding switch to the universal shift register for the 4-bit input and connect all outputs of the shift registers additionally to LEDs (directly on JK flipflops) according to the pulse diagram (fig. 8.2.3.3, page 113).
- **Preparation:** Set the value „0" at the coding switch and trigger as many clock pulses as are required before all LEDs are extinguished. Then set the value "13" (D_{16}) on the coding switch and trigger five clock pulses one after another. Enter the states from the fifth pulse onwards in the pulse diagram.
- Proceed as follows to execute serial data transfer and draw the levels of the register outputs in the pulse diagram after every clock pulse.
 - trigger 4 clock pulses
 - set the new input value „9" and trigger 1 clock pulse
 - trigger 4 clock pulses
- Mark the serial output of the transmitter and the serial input of the receiver as well as the 4-bit information which the receiver has ready after the last data transfer in the pulse diagram.

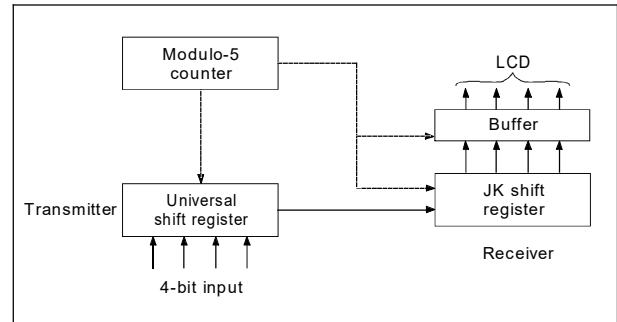


Fig. 8.2.3.1 Serial 4-bit data transfer

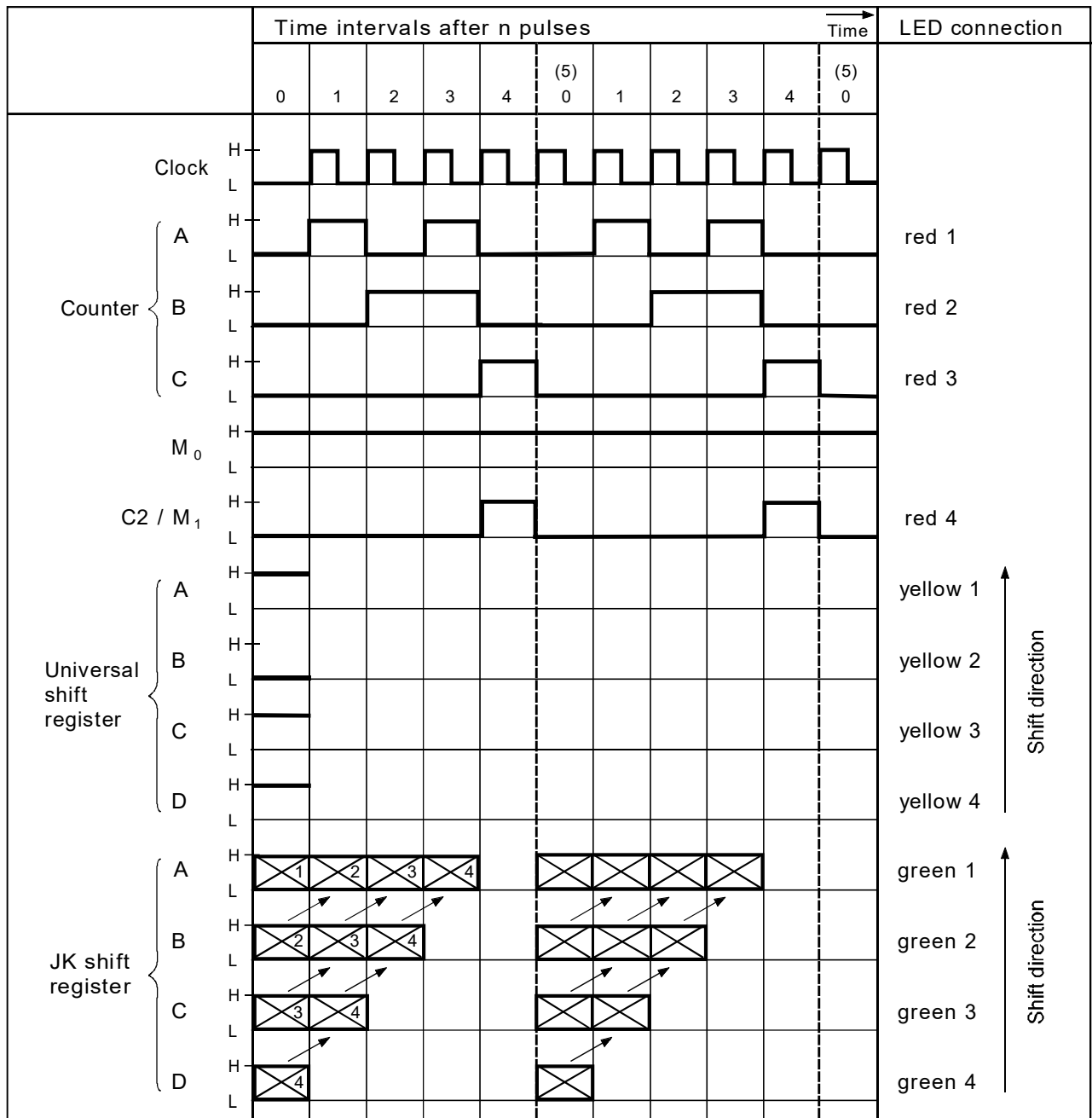


Fig. 8.2.3.3 Pulse diagram for a serial 4-bit data transfer

Notes:

9. Multiplex Mode

9.1 Fundamental Principles

Multiplexers convert parallel data into serial data. Several data inputs are therefore assigned to one data output. The multiplex technique is always used when only **one data line** is used for data transfer.

In the data receiver the serial incoming data are assigned to the individual data outputs according to the set address. This conversion is done by a component known as the **demultiplexer**. The serially transmitted data are therefore available again as parallel data.

Fig. 9.1.1 shows the principle of the multiplex / demultiplex method.

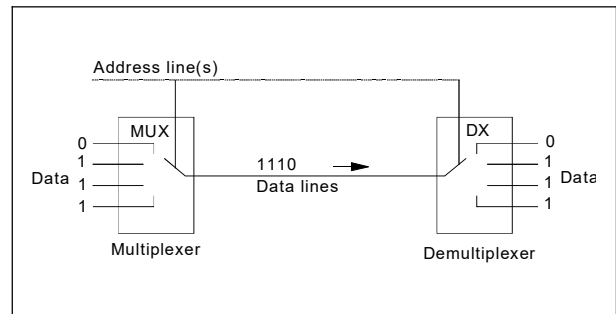


Fig. 9.1.1 Multiplex / Demultiplex method

The data selection in the multiplexer and the subsequent allocation of the data after transmission in the demultiplexer must take place in synchronization. To ensure that the original parallel data are reproduced during the allocation of the serial data, the multiplexer and the demultiplexer must have the same address value for every clock. The number of address lines (select lines) is determined by the number of data inputs in the multiplexer and the number of data outputs in the demultiplexer.

Fig. 9.1.2 and table 9.1.1 show the example of a multiplexer with 4 data input lines and one data output line. This type of multiplexer is also known as a 1 from 4 multiplexer.

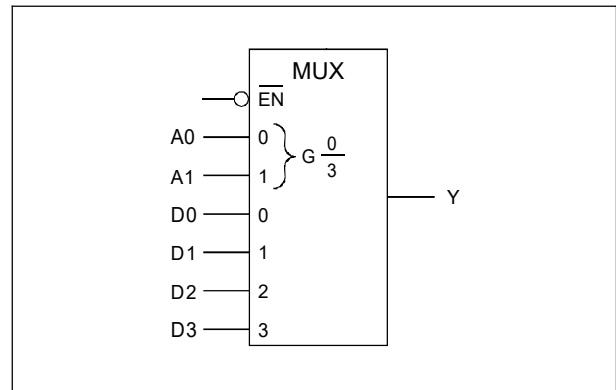


Fig. 9.1.2 1 from 4 multiplexer

- **Note:** The function in table 9.1.1 is only guaranteed if the component in fig. 9.1.2 is activated, i. e. if $\overline{EN} = 0$.

Fig. 9.1.3 (page 116) shows the principle of data transfer in multiplex operation.

Scanning of the data inputs in the multiplexer and switching on the data to the outputs in the demultiplexer is achieved by constant changing of the address values. Two modulo-8 counters which count the same clock pulse can be used for this.

A1	A0	Function
0	0	D0 → Y
0	1	D1 → Y
1	0	D2 → Y
1	1	D3 → Y

Table 9.1.1

In order to generate equal address values at their outputs, the counters must be reset (synchronized) at the start of the data transfer.

The data transfer rate is determined by the clock frequency. Since the single bits of the data are transmitted serially one after the other, we refer to the time division multiplex method.

In addition to this type of multiplex operation, multiplexers are also used in highly integrated components. Since all the connections of a chip can often not be fed out externally, some connections have multiple assignments. For example, in some microprocessors (Intel 8085, 8086, SX types) data lines are also used as address lines. A demultiplexer is therefore required outside the processor to separate the address and data lines.

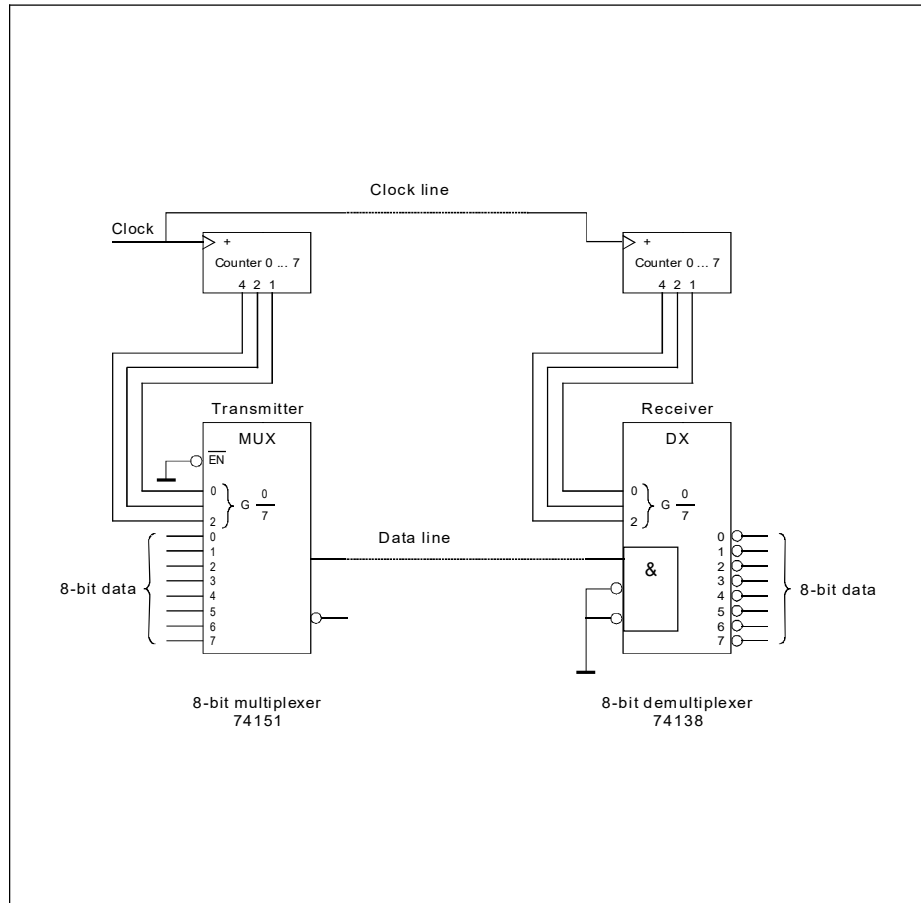


Fig. 9.1.3 Data transfer in multiplex operation

9.2 Experiments Section

□ Experiment 1: Data transfer in multiplex mode

4-bit data are to be transferred in the time division multiplex method. The transmitter and receiver have a common address counter to simplify matters.

Experiment procedure:

- Complete the address control for the multiplexer and the demultiplexer in fig. 9.2.1 (page 118). Also draw the data transfer line between the two components.
- Set up the circuit with the Digital Training System. Note the following instructions:
 - Set up the address counter with two JK flipflops. The inputs J and K remain unconnected for the T function.
 - Connect the clock input of the counter at the bouncefree key.
 - Connect the start signal to an input keyboard.
 - Connect the inputs $E_0 \dots E_3$ of the multiplexer to the other input keyboard with the following setting: $E_0 = H$, $E_1 = L$, $E_2 = H$, $E_3 = L$.
 - Connect the outputs $A_0 \dots A_3$ in the same order as the inputs $E_0 \dots E_3$ to the green LEDs.
 - Connect the data output of the multiplexer additionally to the first yellow LED.
 - Connect the address counter to the left LCD for checking and set input 4 and 8 to „L“. Switch the right LCD off with $x = L$ and connect the counter output Q_A additionally to the first red LED and Q_B to the second red LED.
- Complete the pulse diagram (fig. 9.2.2, page 118). Note the following instructions:
 - Set the start signal according to the pulse diagram.
 - Press the bouncefree key to generate the clock signal according to the pulse diagram. Note that the counter in the active state increments the address value with every negative edge (release key).
 - A uniform representation has been chosen for the time expansion of the clock signal because of the different manual key actuations.
- Mark the position at which the information from input E_0 , E_1 , E_2 and E_3 is located in the pulse diagram for the data line.

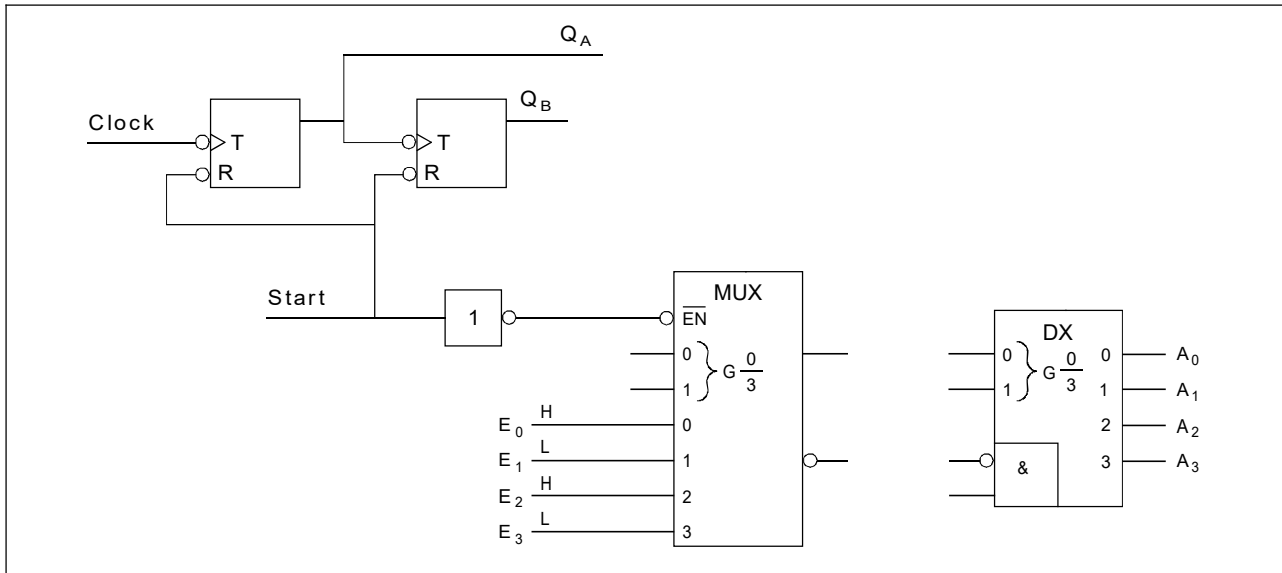


Fig. 9.2.1 Circuit for data transfer in multiplex mode

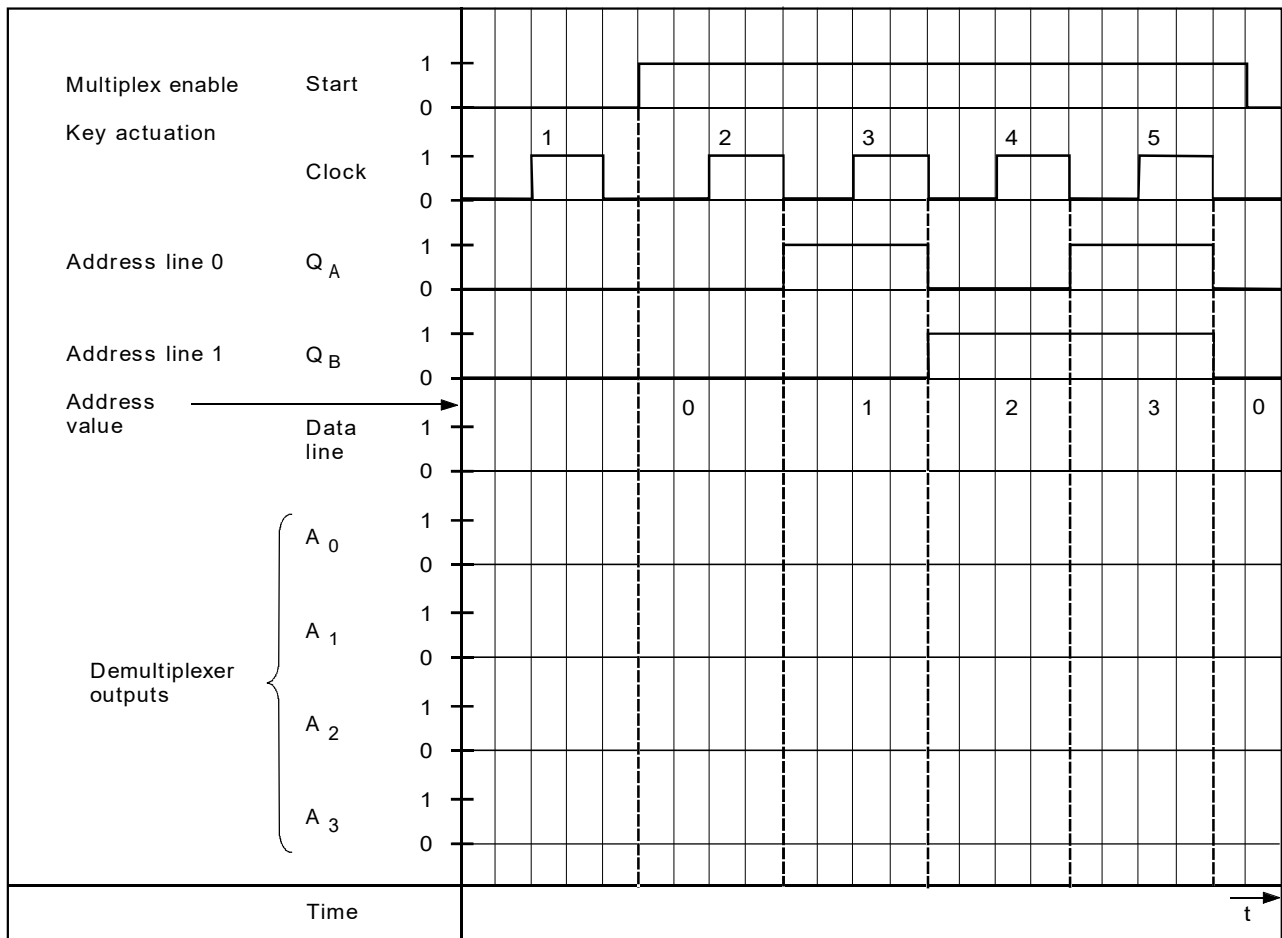


Fig. 9.2.2 Pulse diagram for data transfer in multiplex mode

Notes:

10. Arithmetic Logic Unit

10.1 Fundamental Principles

10.1.1 General

Special arithmetic components have been designed for logic and arithmetic operations which are able to execute these operations dependent on certain control signals. These components are known as **arithmetic logic units** or **ALU** for short.

Fig. 10.1.1.1 shows the block diagram of an Arithmetic Logic Unit.

Arithmetic Logic Units are designed for multi-bit operations. The ALU of the Digital Training System is designed for example for 4-bit operations.

The input informations of the ALU (Operand A and B) are processed in different blocks depending on the type of control signals. In this way either arithmetic or logic operations can be carried out. Only one of the two input informations is transferred in shift operations and each position is shifted to the respective next position.

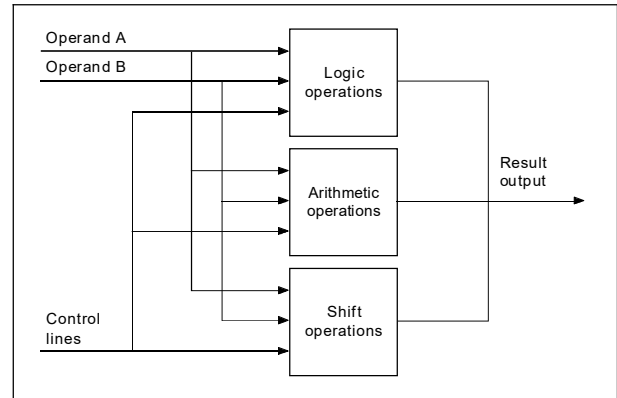


Fig. 10.1.1.1 Block diagram of an ALU

10.1.2 Interaction of ALU and Accumulator

Fig. 10.1.2.1 shows the interaction of ALU and accumulator

In practice the result of the ALU operation is switched to the outputs and transferred to a register there. This register is known as the **accumulator** (accu). At the same time the content of the accumulator is also used as one of the two operands. The operations of the ALU are then always carried out with the content of the accumulator as operand A and the B operands available at the inputs.

By means of special circuits within the ALU component it is possible to transfer an information at the inputs through the ALU to the accumulator without it being changed. With this operation, referred to as **loading** the accumulator, the A operand is available for the ALU. To ensure correct functioning of the circuit, a buffer must be used for the A operand.

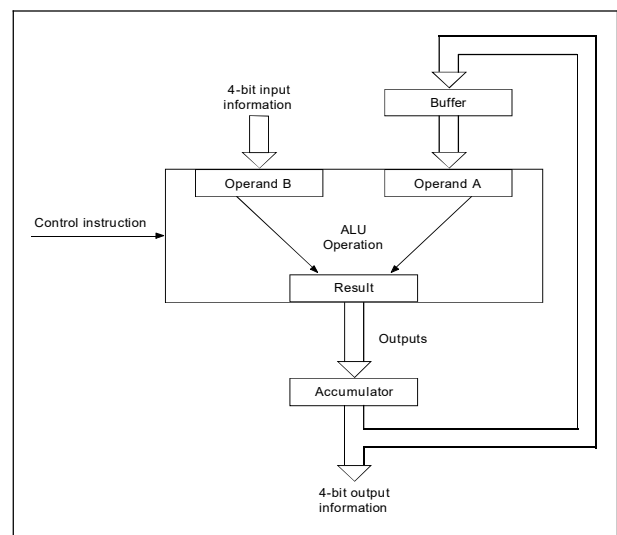


Fig. 10.1.2.1 Interaction of ALU and accu

10.1.3 Arithmetic Logic Unit 74HC / HCT181

Fig. 10.1.3.1 shows the circuit symbol for the arithmetic logic unit 74HC / HCT181.

Pin assignment:

- **P0 ... P3:** 4-bit input data (A operand)
- **Q0 ... Q3:** 4-bit input data (B operand)

Control lines (mode selection):

- **S₀ ... S₃:** function selection with the 4-bit control word at the inputs. The operation is determined with S₀ ... S₃.
- **M:** select line for the operation mode (mode control), switching arithmetic/logic operation
- **CI:** Carry in
- **CO:** Carry out
- **CP, CG:** for special carry formation applications
- **P = Q:** Comparator output. This output carries 1-signal if all result outputs carry 1-signal. This function can be used for certain operations to determine equality of the operands A and B.

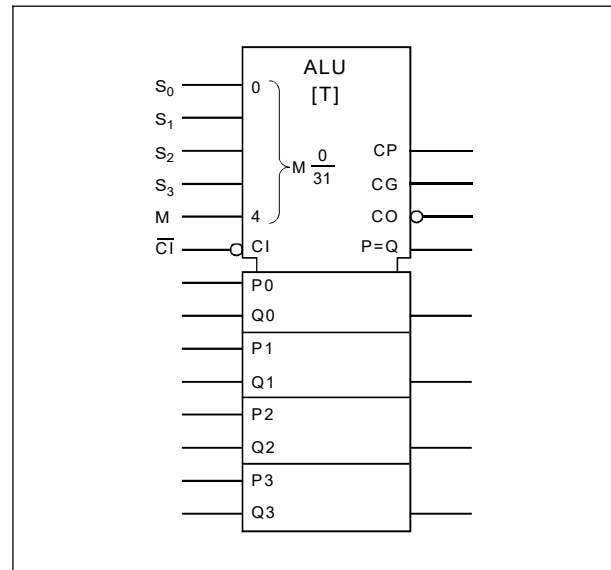


Fig. 10.1.3.1 Arithmetic logic unit 74HC / HCT181

Table 10.1.3.1 contains the most important operations and control signals for the ALU unit 74HC / HCT181.

Notes on pin assignment:

- * : \overline{CI} has no influence on logic operations.
- Operand A corresponds to the information at the inputs P.
- Operand B corresponds to the information at the inputs Q.
- The functions in brackets apply for a circuit set-up of the ALU with accumulator.
- CY has the value „1" and represents the Carry.

Operation	Function	M	S ₃	S ₂	S ₁	S ₀	$\overline{C_i}$	
A + 1	Increment	0	0	0	0	0	0	Arithmetic operations
A - B	Subtraction	0	0	1	1	0	0	
A - B - CY	Subtraction with carry	0	0	1	1	0	1	
A + B	Addition	0	1	0	0	1	1	
A + B + CY	Addition with carry	0	1	0	0	1	0	
A + A	Shift left	0	1	1	0	0	1	
A + A + CY	Rotate left	0	1	1	0	0	0	
A - 1	Decrement	0	1	1	1	1	1	
\overline{A}	NEGATION	1	0	0	0	0	*	Logic operations
\overline{B}	(Load invert)	1	0	1	0	1	*	
$A \oplus B$	EXCLUSIV OR	1	0	1	1	0	*	
$A \equiv B$	EQUIVALENCE	1	1	0	0	1	*	
B	(Load)	1	1	0	1	0	*	
$A \wedge B$	AND	1	1	0	1	1	*	
$A \vee B$	OR	1	1	1	1	0	*	
A	(No effect)	1	1	1	1	1	*	

Table 10.1.3.1 Control signals for the ALU unit 74HC / HCT181

Notes:

10.2 Experiments Section

10.2.1 Arithmetic Operations

□ Experiment 1: Addition and subtraction

Experiment procedure:

- Set up the circuit in fig. 10.2.1.1 with the Digital Training System. Connect the inputs for operand A to the left coding switch and the inputs for operand B to the right coding switch. Connect the outputs of the ALU ($F_0 \dots F_3$) to the red LEDs.
- Determine the control information for addition $A + B$ from the table 10.1.3.1 and enter it in table 10.2.1.1.
- Wire the control inputs of the ALU to the input keyboard.
- Carry out addition with $A = 7$ and $B = 6$ and check the result.
- **Note:** When **adding** 4-bit dual numbers a result greater than 4 bits may occur. In this case the ALU output CO (Carry out) must be taken into account (also known as the Carry flag). Therefore if a carry is produced in an addition, the CO output must be inverted additionally. This additional inversion is not necessary for **subtraction**. Carry out determines the sign.
- Connect the output CO to a LED.
- Carry out the addition in table 10.2.1.2 and check the result.
- Determine the control information for the subtraction $A - B$ from table 10.1.3.1 and enter it in table 10.2.1.3.
- Wire the control inputs of the ALU to the input keyboard.
- Carry out the subtraction in tables 10.2.1.4 and 10.2.1.5 and interpret the results.

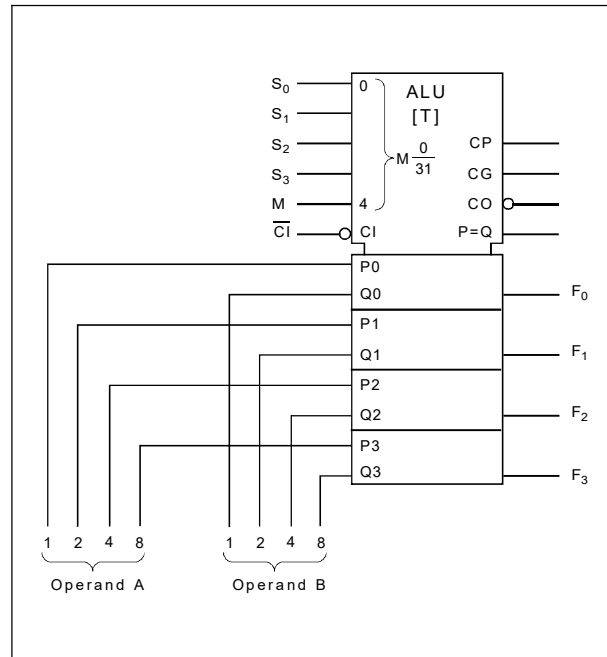


Fig. 10.2.1.1 Circuit

M	S ₃	S ₂	S ₁	S ₀	\overline{CI}

Table 10.2.1.1 Control information for $A + B$

A		1	0	0	1
B	+	1	1	0	0

Table 10.2.1.2 Example addition

M	S ₃	S ₂	S ₁	S ₀	\overline{CI}

Table 10.2.1.3 Control information for $A - B$

A		1	1	0	1
B	-	0	1	1	1

Table 10.2.1.4 Subtraction for $A > B$

A		0	1	1	0
B	-	0	1	1	1

Table 10.2.1.5 Subtraction for $A < B$

Question 1: Realize a **Zero Flag**, i. e. a „1“ is displayed if an operation has the result „0“ (output $P=Q$ of the ALU is not used). Draw the circuit in the box provided (fig. 10.2.1.2).

Answer:

Fig. 10.2.1.2 Zero Flag

Notes:

10.2.2 ALU with Accumulator

□ Experiment 1:

A circuit is to be designed according to the arrangement in fig. 10.2.2.1 which enables operation of an ALU with an accumulator.

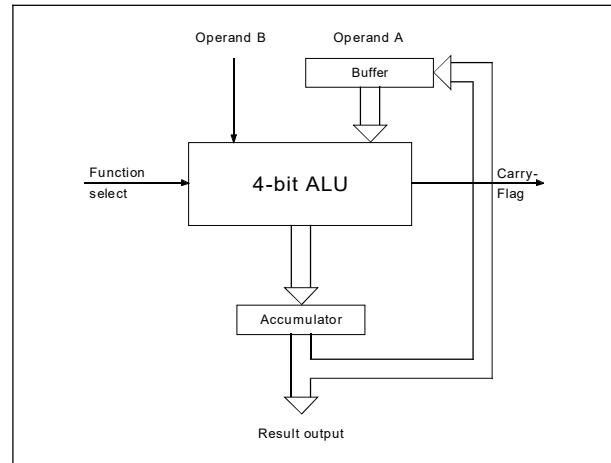


Fig. 10.2.2.1 ALU with accumulator

Experiment procedure:

- First complete all the necessary connections in the circuit diagram (fig. 10.2.2.2).
- Set up the circuit with the Digital Training System. Connect the inputs for operand B to the coding switch.
- Carry out all the operations as specified in table 10.2.2.1 and note down the respective result. Set the necessary control signals for the ALU.

Make sure that the operand A is set on the coding switch before every operation and loaded into the accumulator with the control signals for „Load“ and a clock pulse. After loading, operand A must be reset to „0“ on the coding switch. Then the necessary control signals are applied to the ALU and operand B set on the coding switch. Calculation is carried out in the ALU with another clock pulse.

- **Note:** A = 9 and B = 5 applies for all operations.

Operation	Result
$A + 1$	
$A - B$	
$A + B$	
$A - 1$	
\bar{A}	
$A \oplus B$	
$A \equiv B$	
$A \wedge B$	
$A \vee B$	
A	

Table 10.2.2.1

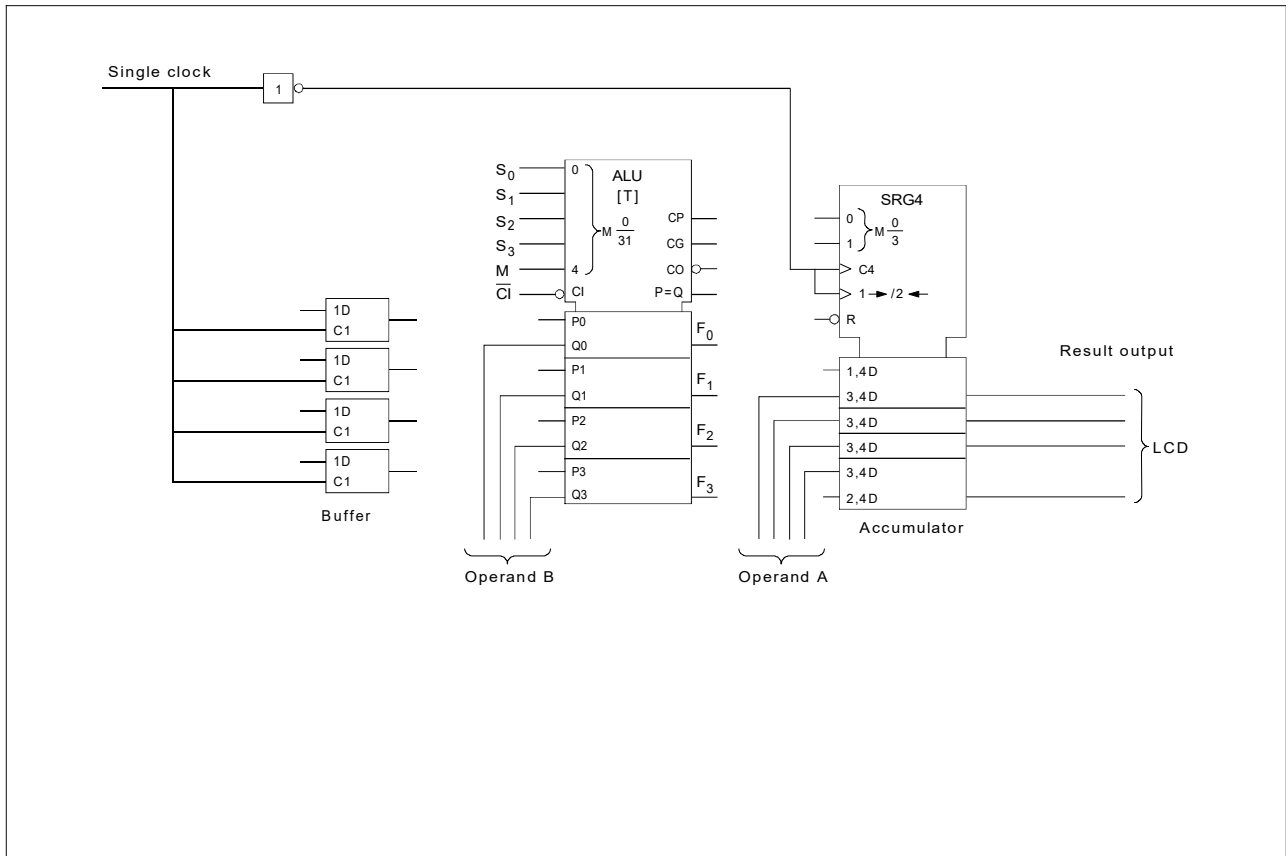


Fig. 10.2.2.2 Circuit diagram

Question 1: What function does the shift register have in the circuit (fig. 10.2.2.2) and what levels must the control inputs M0 and M1 have?

Question 2: What happens to operand A during an arithmetic process?

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

Question 3: Describe the procedure for calculating the expression „4 + 8 - 3 + 2" step by step with the ALU.

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Question 4: What advantage does the use of an accumulator have in connection with the ALU?

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Notes:

11. Memory Circuits

11.1 Fundamental Principles

11.1.1 General

A different memory structure is necessary in processor systems with respect to the number of memory locations (**capacity**) and the number of bits per location (**word width**) depending on the area of application.

However, for reasons of economy, memory components are often used which, viewed individually, do not satisfy the requirements of capacity and word width. These components require special interconnection.

RAMs are used here as memories. These are referred to in semiconductor technology as read and write memories. Every memory location has a fixed storage capacity. It can therefore store an information with a specific bit length. The individual memory locations are identified by addresses. Memory cells can be selected with the aid of these addresses. A RAM therefore operates with random access. The abbreviation RAM stands for Random Access Memory.

11.1.2 Extension of the the Word Width

The word width of the memory components must always be adapted to the processing width of the processor. A 32-bit processor, for example, requires a memory system in which 32 bits can be stored for every address (i. e. for every memory location).

Memory components which do not have the required word width can be operated in parallel.

A memory with an organisation of 1024 memory locations x 8 bits per location can be operated with two RAMs 2114 (1024 x 4 bits) as shown in fig. 11.1.2.1.

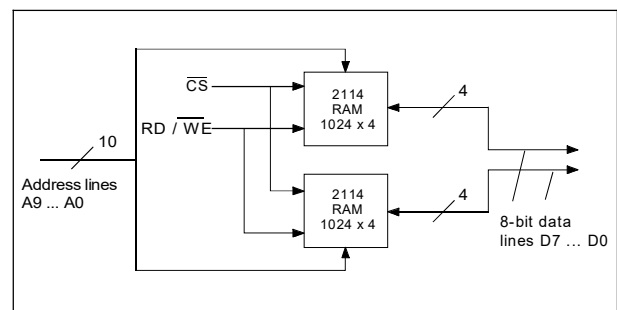


Fig. 11.1.2.1

The address lines and the control lines **CS** (Chip Select) and **WE** (Write Enable) are connected in parallel to the two memory components. An 8-bit memory word is split between the two memories in such a way that each memory stores 4 bits.

The addresses of the memory locations (address bits A9 ... A0) are in the range 0000₍₁₀₎ ... 1023₍₁₀₎. A block diagram (fig. 11.1.2.2) in which the appropriate memory ranges are specially marked is often used to describe memory systems.

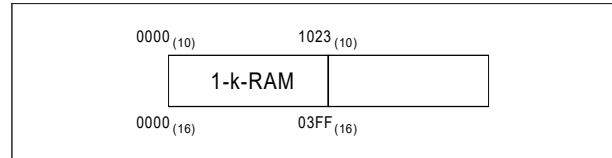


Fig. 11.1.2.2 Block diagram for memory range

11.1.3 Extension of the Number of Memory Locations

If a memory system requires more locations than are contained in the component, the capacity must be increased by connecting the appropriate number of components in series.

For example, a memory with an organisation of 2048 memory locations x 4 bits per location can be operated with two RAMs 2114 (1024 x 4 bits) as shown in fig. 11.1.3.1.

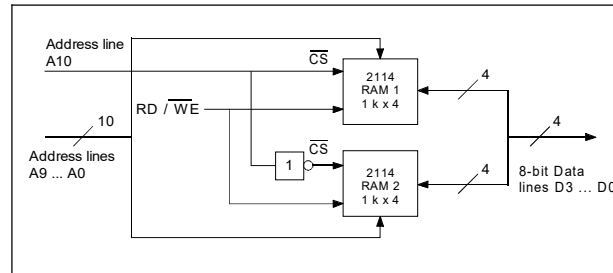


Fig. 11.1.3.1

Fig. 11.1.3.2 shows how the required number of 2048 memory locations is divided in half between the two memory components.

The problem is therefore to design a decoder circuit which only activates the component (CS) which contains the desired memory location. Eleven address lines (A10 ... A0) are required to address the whole system with 2048 memory locations. The address lines A9 ... A0 are fed parallel to the 10 address inputs of the components and address one of 1024 memory locations within a component. The eleventh address line (A10) has - in a binary number consideration of the address lines - a value of $2^{10} = 1024$ and decides whether an address is in the range 0 ... 1023 (A10 = 0) or 1024 ... 2047 (A10 = 1).

In this simple example it can be determined (decoded) using address line A10 whether RAM 1 or RAM 2 is activated.

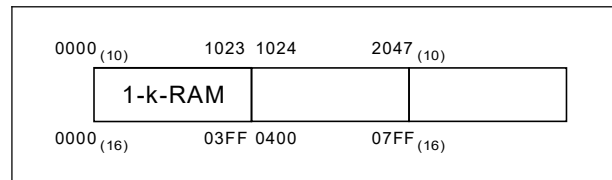


Fig. 11.1.3.2

The decoder circuit consists of only one single inverter. The four I/O lines of the memory components are circuited in parallel with the 4-bit data bus. This can be done without risk because the lines of the respective inactive components of the data bus are switched off (tri-state).

If a data receiver is not switched to reception for example, its data inputs should be high-ohmic. The data receiver may not influence the signals on the data bus lines. Therefore a third high-ohmic state of the inputs must be possible in addition to High and Low. Circuits with inputs and outputs which can be switched high-ohmically are known as **tri-state circuits**.

11.1.4 Memory Components RAM 8x 4 and EEPROM 8 x 4

Fig. 11.1.4.1 shows the circuit symbols of the RAM 8 x 4 and of the EEPROM 8 x 4.

Pin assignment:

- **0 ... 2:** address inputs for selecting the memory locations 0 ... 7
- **WE:** Write Enable
- **OE:** Output Enable
- **CS:** Chip Select

The memory component has four data lines which operate as input or output lines depending on the wiring of the control inputs or which are in the high-ohmic state (tri-state).

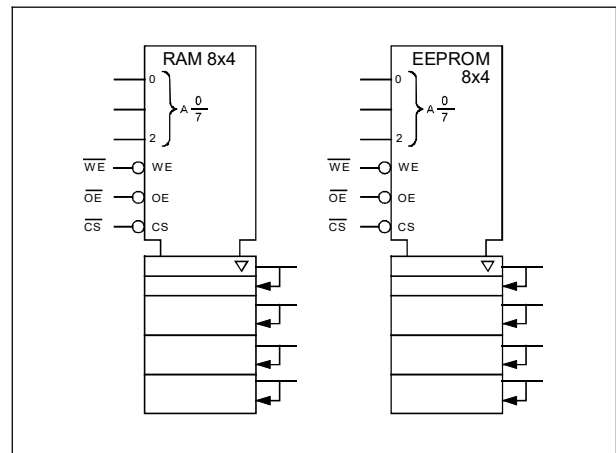


Fig. 11.1.4.1 RAM 8 x 4 and EEPROM 8 x 4

Table 11.1.4.1 gives an overview of the control input wiring, the resultant state of the RAM or EEPROM and the respective function.

- **Note:** The EEPROM used here can keep its data for approx. 1 hour without an operating voltage.

	Control inputs			State of memory	Function
	\overline{WE}	\overline{OE}	\overline{CS}		
Write memory	0	1	0	active	Data lines operate as inputs, bit combination is written in the memory
	0	1	1	not active	Data lines are high-ohmic (switched off)
Read memory	1	0	0	active	Data lines operate as outputs, content of the addressed memory cell is on the data lines
	1	0	1	not active	Data lines are high-ohmic (switched off)

Table 11.1.4.1 Function table

11.2 Experiments Section

11.2.1 Writing and Reading Memory Components

□ Experiment 1: RAM 8 x 4

Experiment procedure:

- First determine the states of the control lines \overline{WE} and \overline{OE} if informations are to be read in (input).
- Set up the circuit (fig. 11.2.1.1) for data storing (input) with the Digital Training System. Wire the control inputs WE and OE accordingly to an input keyboard.
- **Note:** The addresses and the data to be read in must be set with the coding switch.
- Then store the content of table 11.2.1.1 step by step at the respective specified address. Activate the memory component after setting the address and the input information by pressing the bouncefree key \overline{Q} .
- **Attention:** Do not switch off the power supply at the end of the storing procedures otherwise the data in the RAM will be deleted.
- Then determine the states of the control lines \overline{WE} and \overline{OE} if informations are to read out (output).
- Wire the control inputs WE and OE accordingly to the input keyboard and connect the outputs to LEDs.
- Activate the memory component after setting the address and the input information by pressing the bouncefree key \overline{Q} and compare the read out data with the stored data from table 11.2.1.1.
- Extend the circuit in fig. 11.2.1.1 so that the information is retained on the LEDs for the data output even after pressing the key.

Addr.	Memory content					
	Binary				Deci- mal	Hexa- dec.
	D3	D2	D1	D0		
0	0	1	0	0	4	4
1	0	1	1	1	7	7
2	1	0	0	0	8	8
3	1	0	0	1	9	9
4	1	0	1	0	10	A
5	1	0	1	1	11	B
6	1	1	0	1	13	D
7	1	1	1	1	15	F

Table 11.2.1.1

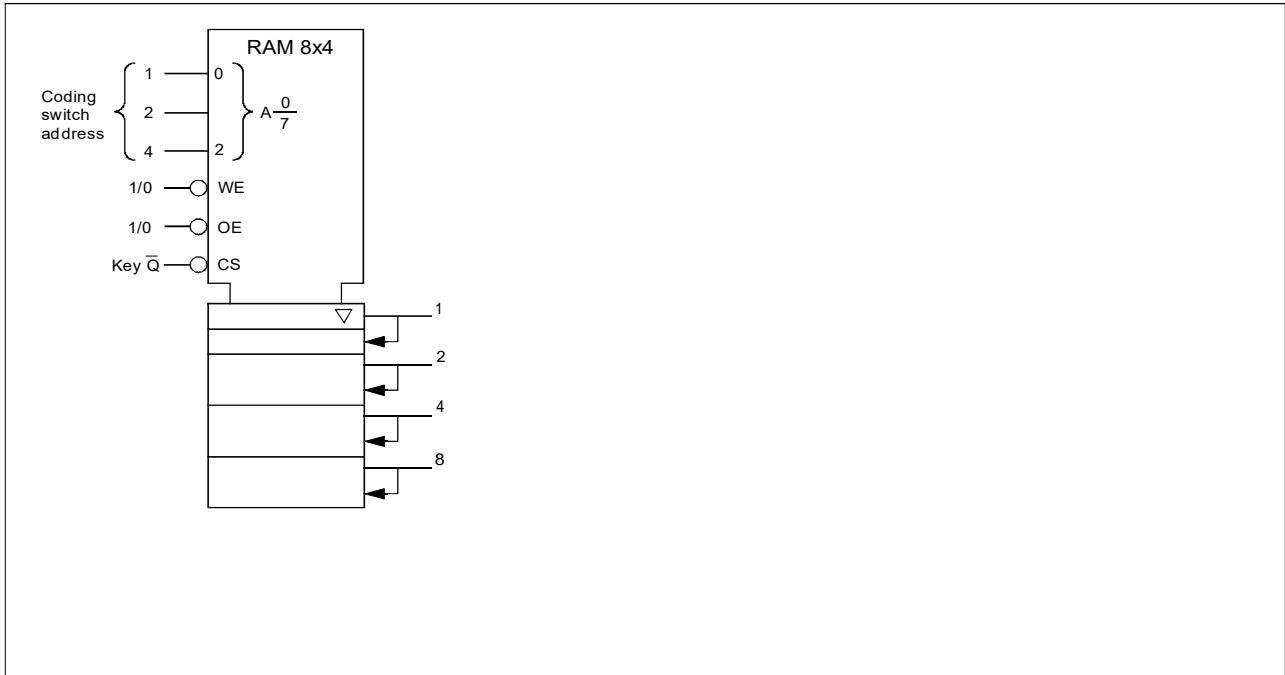


Fig. 11.2.1.1 Circuit

Experiment 2: EEPROM 8 x 4

Experiment procedure:

- Replace the RAM with the EEPROM (simulation) component.
- Disconnect the component briefly from the Digital Training System (the power supply is interrupted).
- Repeat experiment 1.

Question 1: Does an interruption of the power supply lead to loss of data in the EEPROM?

Answer:

.....

.....

.....

.....

.....

.....

.....

11.2.2 Word Width

□ Experiment 1: Extension of the word width

The light pattern illustrated in table 11.2.2.1 is to be generated continuously by programmed driving of 8 LEDs.

Experiment procedure:

- The light pattern of the green LEDs should be realized with the RAM component and the light pattern of the yellow LEDs with the EEPROM (simulation) component.
Determine the appropriate memory contents if step „0“ corresponds to the memory value of address “0” and so on (green 1 / yellow 1 lowest significant bit respectively). Note the values in table 11.2.2.2.
- After setting the address and the respective input data, both memory components should be programmed simultaneously with pressing of the key \bar{Q} .
- Mark the control inputs WE and OE with potential „0“ or “1” required for input (data storage) for every memory component in the following circuit (fig. 11.2.2.1).
- Connect the CS signal to the key so that both components can be activated at the same time and the respective available data can be stored.
- Set up the circuit in fig. 11.2.2.1 with the Digital Training System.
- Input the data according to the determined table 11.2.2.2.
- **Notes:** Do not switch off the power supply within the further course of the experiment otherwise the data will be deleted. Do not disconnect the RAM from the Digital Training System.
- Both memory components are activated permanently ($\bar{CS} = L$) to drive the LEDs (data output). The stored value is displayed immediately by applying an address.
Continuous addressing is taken care of in the following circuit (fig. 11.2.2.2) by a modulo-16 counter which only needs to be clocked accordingly. The highest value output of the counter is not used here.

Light pattern of LEDs								
Step	green	green	green	green	yellow	yellow	yellow	yellow
	4	3	2	1	4	3	2	1
0	1	0	0	0	0	0	0	1
1	1	1	0	0	0	0	1	1
2	1	1	1	0	0	1	1	1
3	1	1	1	1	1	1	1	1
4	1	1	0	1	1	0	1	1
5	1	0	0	1	1	0	0	1
6	0	0	0	1	1	0	0	0
7	0	0	0	0	0	0	0	0

Table 11.2.2.1

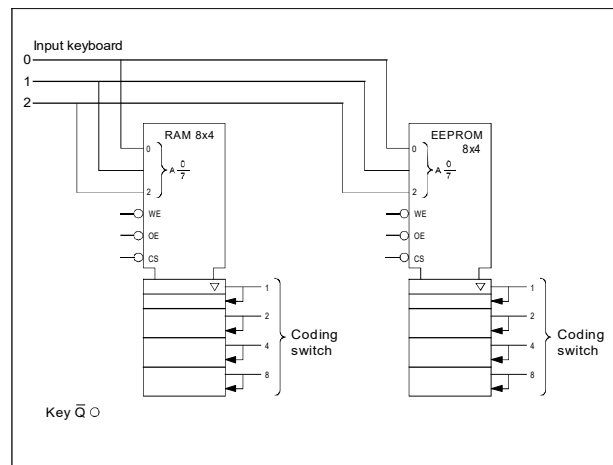


Fig. 11.2.2.1

- Set up the following circuit (fig. 11.2.2.2) with the Digital Training System.
- To check the address values connect the three lowest value outputs of the counter to the three red LEDs.
- Clock the counter by pressing the bouncefree key Q. Observe the address value and the display with the LEDs. Check the memory contents to see if they match table 11.2.2.1. Then replace the single clock with clock frequency of 1 Hz.

Address	RAM		EEPROM	
	Decimal	Hexadec.	Decimal	Hexadec.
0				
1				
2				
3				
4				
5				
6				
7				

Table 11.2.2.2

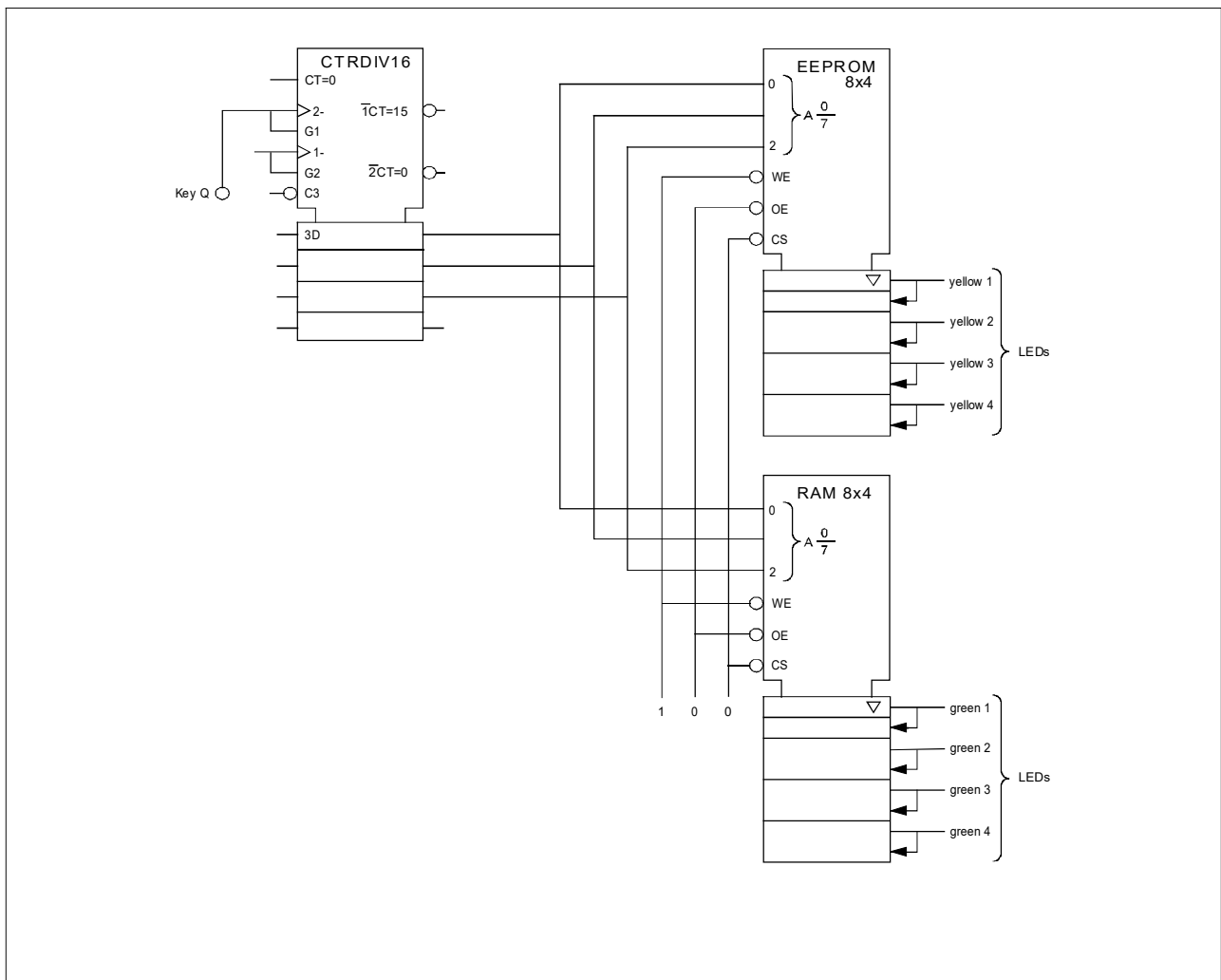


Fig. 11.2.2.2

Question 1: How do the contents of the memory components need to be re-programmed so that a running light (only one diode alight at a time) is produced for example? Note a possible solution in table 11.2.2.3.

Answer:

Step	Light pattern of the LEDs							
	green 4	green 3	green 2	green 1	yellow 4	yellow 3	yellow 2	yellow 1
0								
1								
2								
3								
4								
5								
6								
7								

Table 11.2.2.3

Notes:

Question 2: How do the address and control inputs need to be switched in memory components to achieve a greater word width?

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

11.2.3 Memory Capacity

□ Experiment 1: Increasing of the memory capacity

The light pattern shown in table 11.2.3.1 is to be generated continuously by programmed driving of LEDs.

Experiment procedure:

- The light pattern covers 16 steps which cannot all be stored in one memory component. It is therefore necessary to divide up the memory information: steps 0 ... 7, RAM component; steps 8 ... 15, EEPROM component.
The CS signal must then guarantee that the RAM is activated during steps 0 ... 7 and the EEPROM during steps 8 ... 15.
- Determine the corresponding memory contents when step „0" corresponds to the memory value of address "0" etc. (LED green 1 = least significant bit). Note the values in table 11.2.3.2 (page 138).
- **Note:** The memory components have only three address inputs each. Steps 0 ... 15 are represented, however, by a 4-bit address. The bit A_3 not used for the address inputs of the memory components must be enlisted to decide which memory is to be activated.
- Mark the position of the memory components (RAM, EEPROM) in the following block diagram (fig. 11.2.3.1).
- Set up the circuit for programming the memory components with the Digital Training System according to fig. 11.2.2.1 on page 134. The RAM and the EEPROM must have separate address lines for the programming (2 input keyboards).
- Input the data according to the determined table 11.2.3.2 (store steps 8 ... 15 at addresses 0 .. 7 of the EEPROM).
- **Notes:** Do not switch off the power supply in the further course of the experiment otherwise the data will be deleted. Do not disconnect the RAM from the Digital Training System.
- To drive the LEDs, the memory components are activated alternately depending on which step of the light pattern is to be executed.

Steps	Light pattern of the LEDs			
	green 4	green 3	green 2	green 1
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	0	0	0
6	1	1	1	1
7	0	0	0	0
8	0	0	0	0
9	1	1	1	1
10	0	0	0	0
11	1	1	1	1
12	0	1	1	1
13	0	0	1	1
14	0	0	0	1
15	0	0	0	0

Table 11.2.3.1

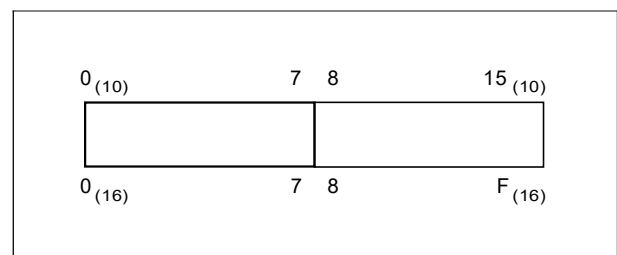


Fig. 11.2.3.1

- Complete the activation of the memory address inputs with a counter in the following circuit (fig. 11.2.3.2).
- Connect the CS control input to the AND element of the fourth counter output so that the memory components are activated at the right time.
- Then set up the circuit in fig. 11.2.3.2 with the Digital Training System.
- Clock the counter by pressing the bounce-free key Q. Check the memory contents to see if they match table 11.2.3.1.
- Then replace the single clock with a clock frequency of 10 Hz.

Step	Address				Value	
	A ₃	A ₂	A ₁	A ₀	Dec.	Hex.
0	0	0	0	0		
1	0	0	0	1		
2	0	0	1	0		
3	0	0	1	1		
4	0	1	0	0		
5	0	1	0	1		
6	0	1	1	0		
7	0	1	1	1		
8	1	0	0	0		
9	1	0	0	1		
10	1	0	1	0		
11	1	0	1	1		
12	1	1	0	0		
13	1	1	0	1		
14	1	1	1	0		
15	1	1	1	1		

Table 11.2.3.2

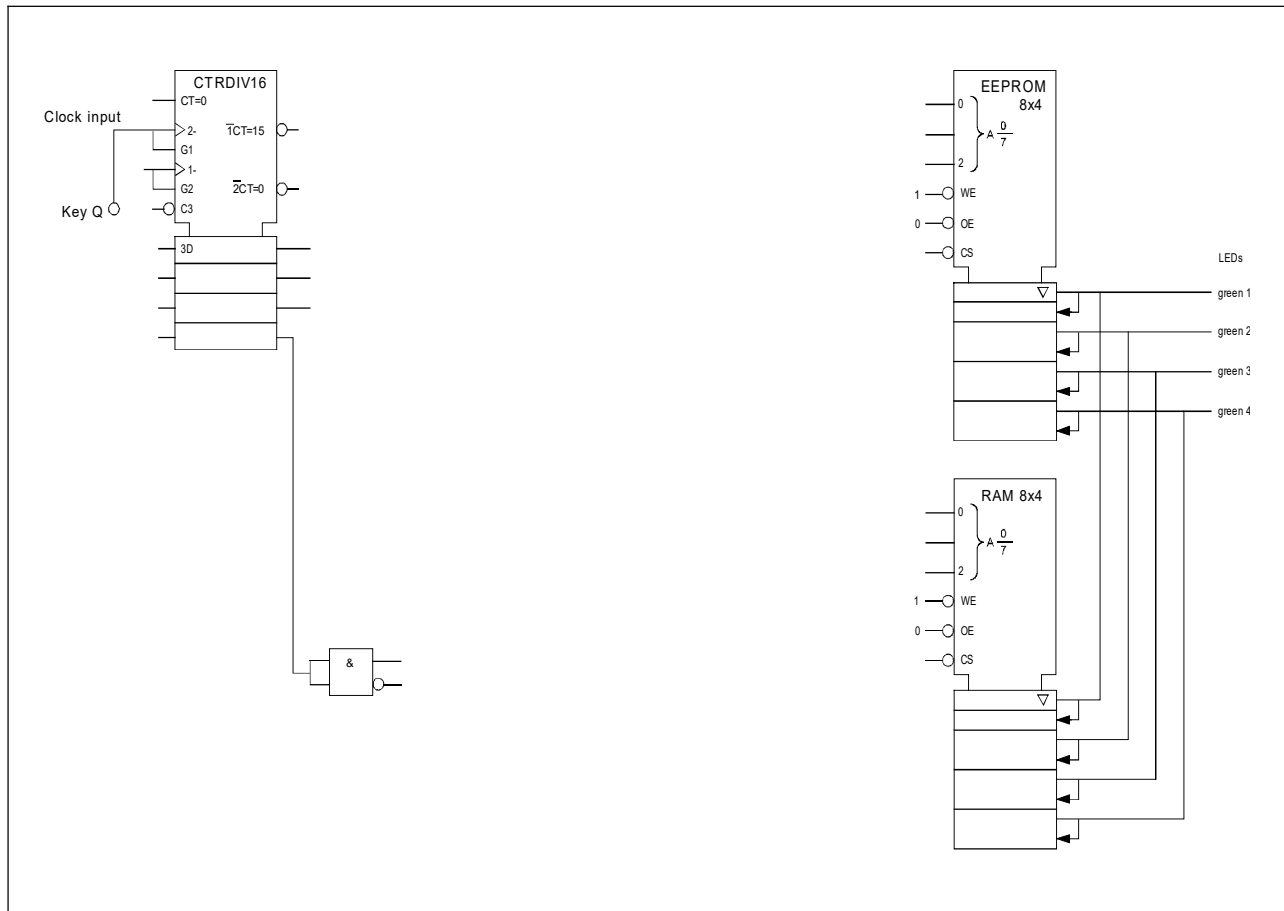


Fig. 11.2.3.2

12. Analog-Digital Converters, Digital-Analog Converters

12.1 Fundamental Principles

12.1.1 General

Digital signals are better suited for long distance transmission than analog signals.

A digital information transfer operates without falsification of the signal value and allows multiple exploitation of the transmission line.

If an analog signal is to be transmitted digitally from point A to point B, an analog-digital converter must be used at A and a digital-analog converter at B.

12.1.2 Analog-Digital Converters

Analog-Digital converters, or **AD converters**, convert analog signals into the appropriate digital signals.

Fig. 12.1.2.1 shows the circuit symbol of an AD converter.

Since every analog signal can be represented by a certain number of amplitude values, the instantaneous value of the analog voltage must be sampled (measured) at constant intervals of time.

To do this, the whole range of analog variables occurring is divided into individual ranges and every range is assigned a binary code word (in dual code or 8421-BCD code).

The size of the voltage ranges is known as resolution and depends on the number of bits available for a code word.

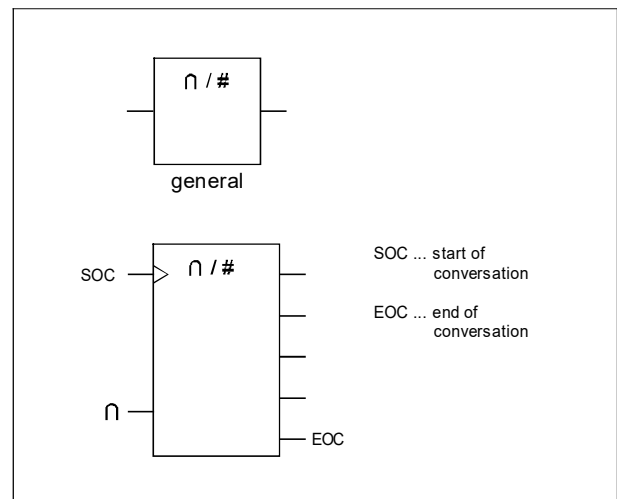


Fig. 12.1.2.1 Circuit symbol of an AD converter

12.1.3 Digital-Analog Converters

A digital-analog converter, or **DA converter**, is required to recover the analog signal from a digital information.

Fig. 12.1.3.1 shows the circuit symbol of a DA converter.

A digital-analog converter assigns the individual positions of a digital signal a certain analog voltage value. Therefore only **weighted codes** are suitable for digital-analog conversion. A code is weighted when its elements are assigned certain numeric values, e. g. in the dual code.

A stepped voltage is obtained as an analog signal, whereby the number of voltage values (2^n) is given by the word length (n = number of bits) of the digital signal's code words.

The **resolution** U_S of a DA or AD converter is the change in voltage per stage in volts (V):

$$U_S = \frac{U_{AN}}{2^n - 1}$$

U_{AN} ... analog voltage range in volts

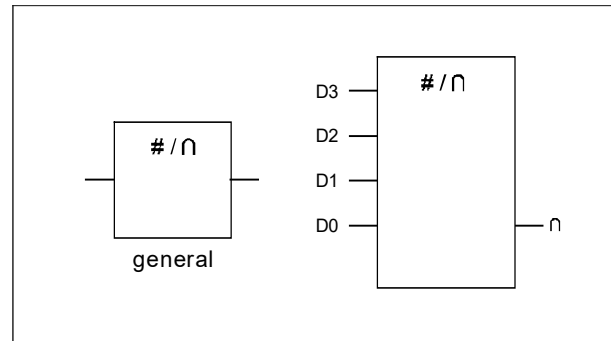


Fig. 12.1.3.1 Circuit symbol of a DA converter

12.2 Experiments Section

12.2.1 Analog-Digital Conversion

Experiment: Principle of analog-digital conversion

Experiment procedure:

- Set up the circuit in fig. 12.2.1.1.
- Measure the analog voltage U_{AN} for the specified code words with an analog or digital voltmeter. Set the value for the respectively required code word with the generator's potentiometer.
- Enter the respective start value in table 12.2.1.1.
- Assign the code word to the respective voltage range in the diagram (fig. 12.2.1.2, page 142).

Code word	U_{AN} [V]
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
b	
C	
d	
E	
F	

Question 1: Specify the voltage range of a code word.

Answer:

Question 2: Calculate the resolution capacity of the AD converter.

Answer:

Table 12.2.1.1

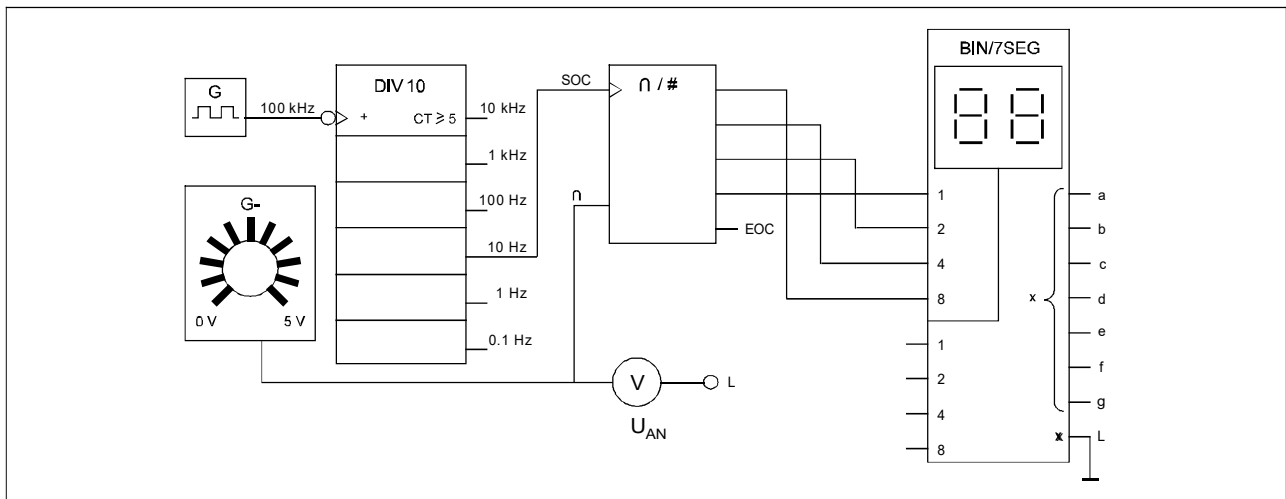


Fig. 12.2.1.1 Circuit for an analog-digital conversion

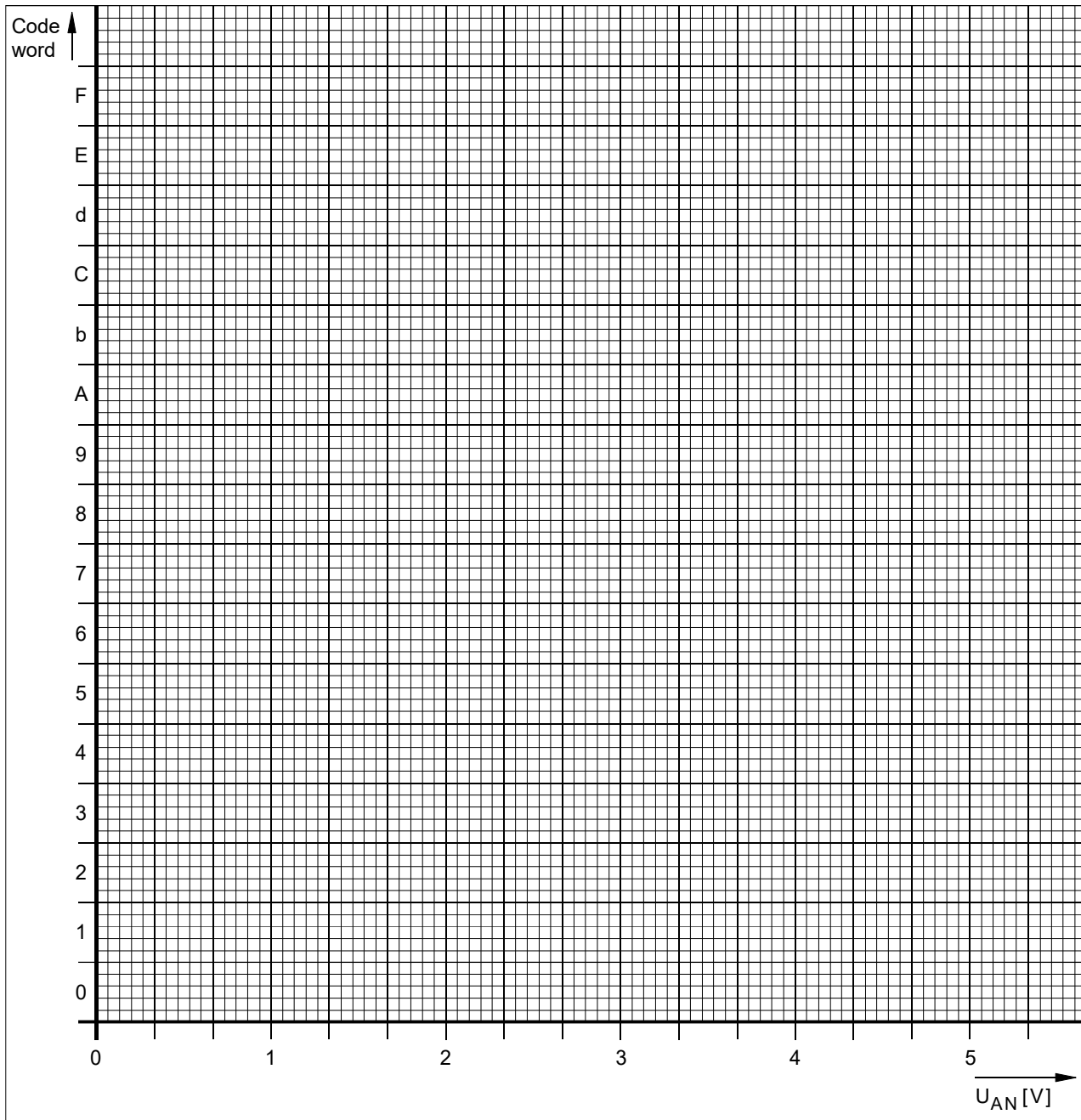


Fig. 12.2.1.2 Diagram

❑ Experiment 2: Examining the EOC output

The circuit must be modified in order to examine the EOC output.

Experiment procedure:

- Replace the clock generator in experiment 1 (fig. 12.2.1.1) with the bouncefree key (Q).
- Connect the EOC output to the monoflop (non inverted output, $t = 1$ s) and display the state of the monoflop with a LED.
- Complete the circuit in fig. 12.2.1.3.

Question 1: Change the voltage U_{AN} constantly from 0 ... 5 V. What do you discover?

Answer:

.....

.....

.....

Question 2: Now set any voltage U_{AN} and press the bouncefree key. Note down your observation.

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

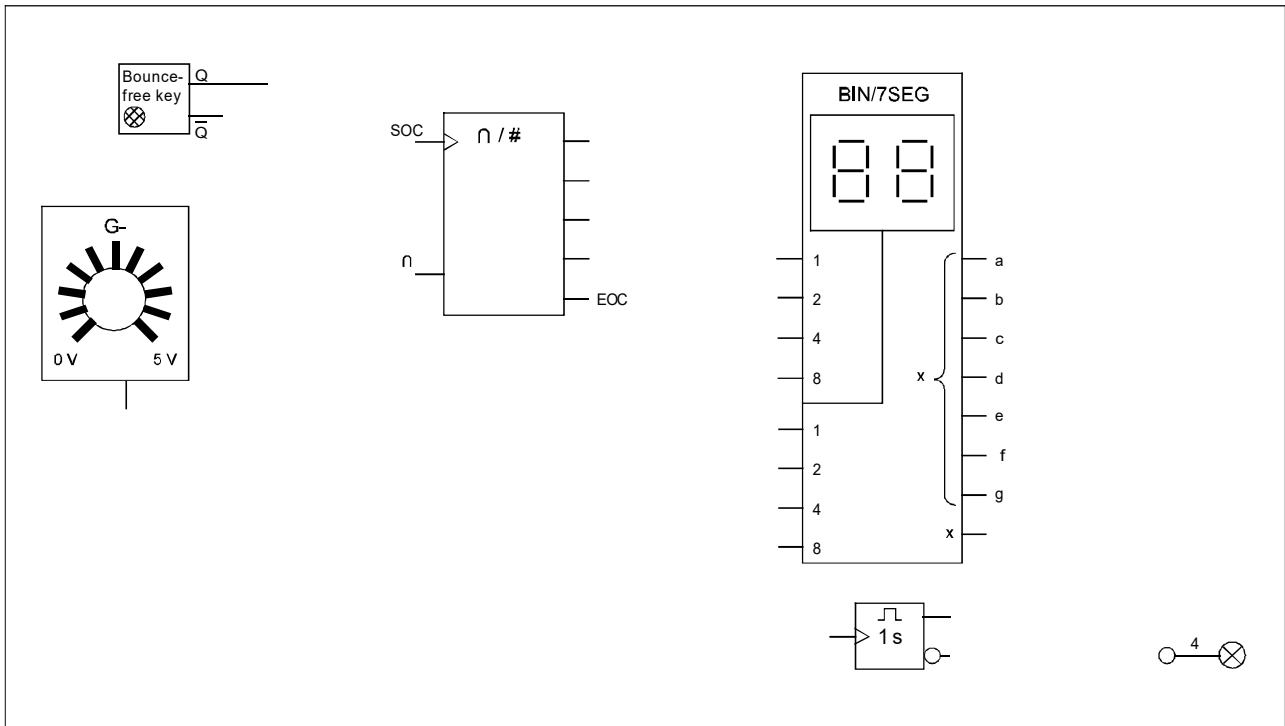


Fig. 12.2.1.3 Circuit for examining the EOC output

12.2.2 Digital-Analog Conversion

Experiment 1: Principle of digital-analog conversion

Experiment procedure:

- Complete the circuit in fig. 12.2.2.1 to record the values required by table 12.2.2.1.
- Draw the analog signal as a staircase voltage in the diagram provided (fig. 12.2.2.2).

Question 1: Mark U_{AN} and U_S in the diagram (fig. 12.2.2.2). What is the value of the voltage change per stage, purely arithmetically?

Answer: $U_S = \dots\dots\dots$

Question 2: Digital signals represented in Gray code are to be converted into analog signals. What do you have to do?

Answer:
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Digital value	Amplitude value [V]	Digital value	Amplitude value [V]
0		8	
1		9	
2		A	
3		B	
4		C	
5		D	
6		E	
7		F	

Table 12.2.2.1

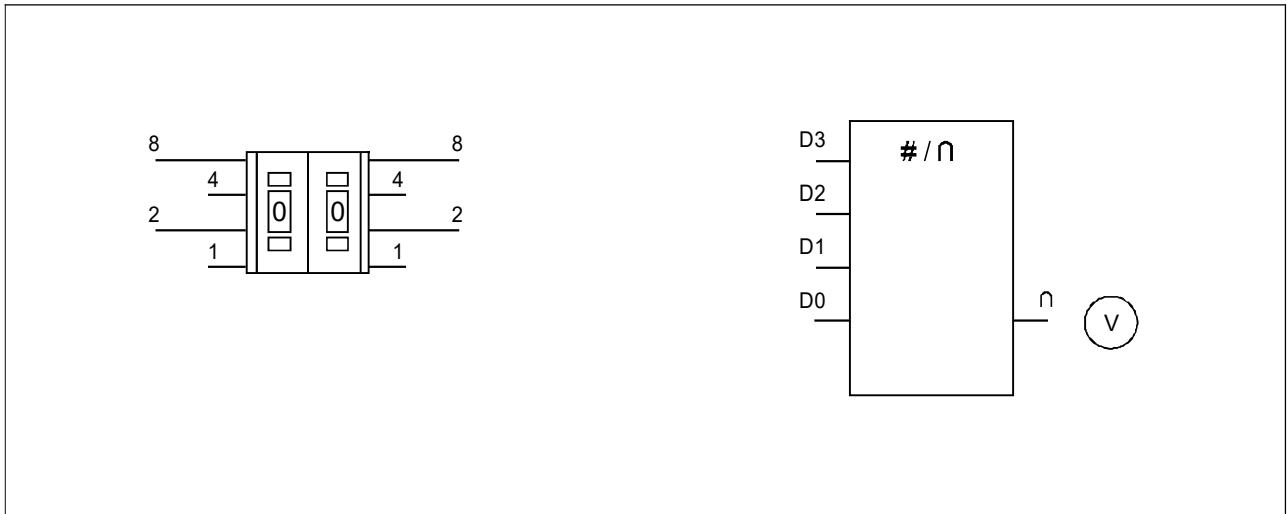


Fig. 12.2.2.1 Circuit for a digital-analog conversion

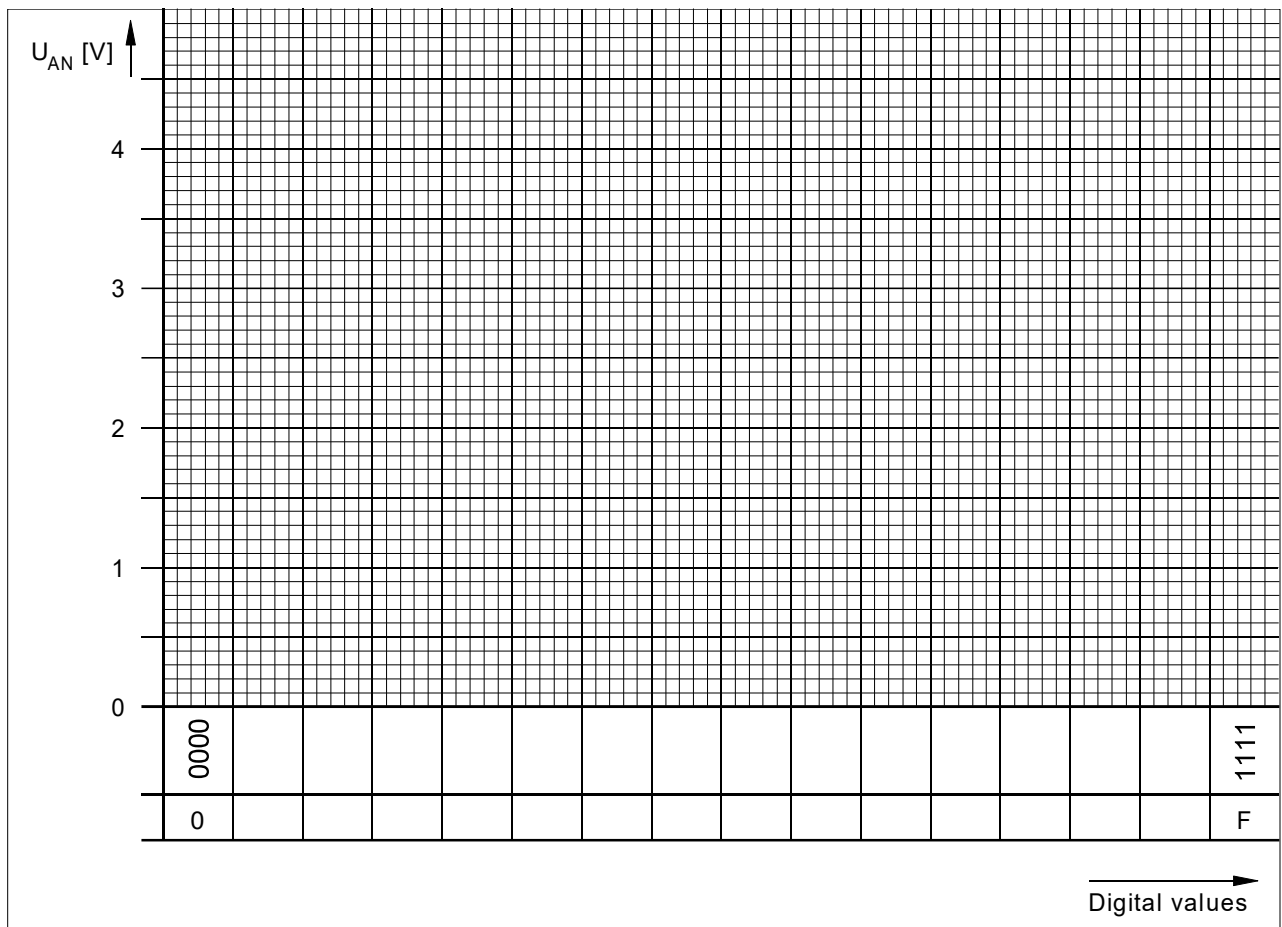


Fig. 12.2.2.2 Diagram

12.2.3 Signal Transfer

□ Experiment 1: Digital transfer of an analog signal

An analog signal is to be transferred digitally over a certain distance and be available at the reception site as an analog signal for processing in the measuring system.

Experiment procedure:

- Complete the circuit in fig. 12.2.3.1.
- Specify the analog voltage values (U_T = voltage at the transmitting place, U_R = voltage at the receiving place) for the individual code words in table 12.2.3.1.
- The accuracy of converters indicates by what fraction of the correct value the result of the conversion may deviate up and down at the maximum. Determine with what accuracy our signal transfer operates by entering the absolute and percentage deviation of the voltage value (ΔU) of the receiver in table 12.2.3.1. Use an analog or digital voltmeter for this.

U_T [V]	Code word	U_R [V]	Deviation	
			ΔU [mV]	ΔU [%]
	0			
	1			
	2			
	3			
	4			
	5			
	6			
	7			
	8			
	9			
	A			
	b			
	C			
	d			
	E			
	F			

Table 12.2.3.1

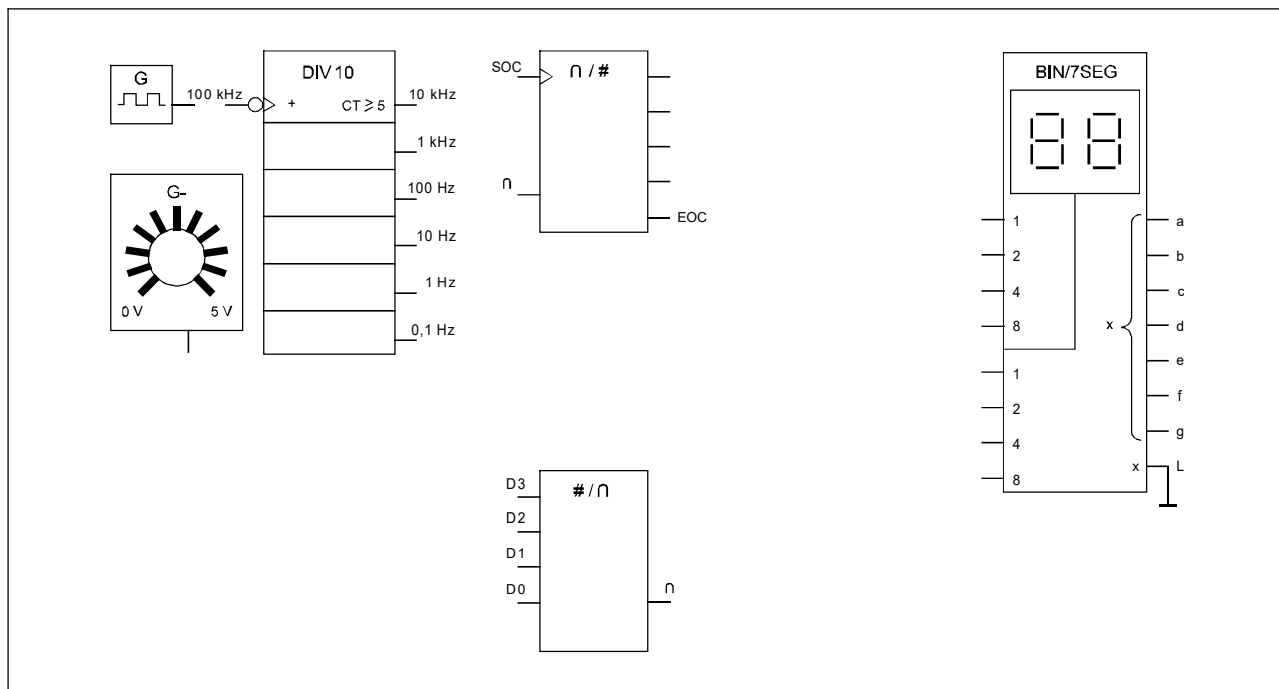


Fig. 12.2.3.1 Circuit for a digital transfer of an analog signal

❑ Experiment 2: Defective transmission line

Use the circuit in fig. 12.2.3.1 to check what effect a defective transmission line can have on the correct result.

Experiment procedure:

- First transfer the values for U_R from table 12.2.3.1 to table 12.2.3.2.
- Then break the connection to D3 (most significant bit) and apply this input to H level.
- Increase the voltage with the generator’s potentiometer until the LCD always shows the next highest value (8 → 9 → A → etc.) and note the read values in table 12.2.3.2.

- Calculate the respective difference:

$\Delta U_R = \text{defective value} - \text{correct value}$

- Then repeat the experiment by breaking the connection to D0 and connect this input the H level (least significant bit).
The connection to D3 must be plugged in.

Correct value U_R [V]	D3 defective		D0 defective	
	U_R [V]	ΔU_R [V]	U_R [V]	ΔU_R [V]

Table 12.2.3.2

Question 1: Give a summary of the knowledge gained from table 12.2.3.2.

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

12.2.4 Digital Function Generator

□ Experiment 1:

A function generator can be used to generate periodic voltages with any timing curve.

Experiment procedure:

- Complete the blocks in fig. 12.2.4.1 to form a function generator.
- Then write the code words stored under every address which are specified in table 12.2.4.1 in the EEPROM. Note the measured voltage values.
- **Note:** Proceed as in chapter 11.2.1, page 132. The signal states of the control inputs \overline{WE} , \overline{OE} and \overline{CS} can be read from table 11.1.4.1 on page 131.
- Draw the staircase curve of the analog voltage in the diagram (fig. 12.2.4.2).

Address	Code word	U [V]
0 0 0	0 0 0 0	
0 0 1	0 0 0 1	
0 1 0	0 0 1 0	
0 1 1	0 0 1 1	
1 0 0	0 1 0 0	
1 0 1	0 0 1 1	
1 1 0	0 0 1 0	
1 1 1	0 0 0 1	

Table 12.2.4.1

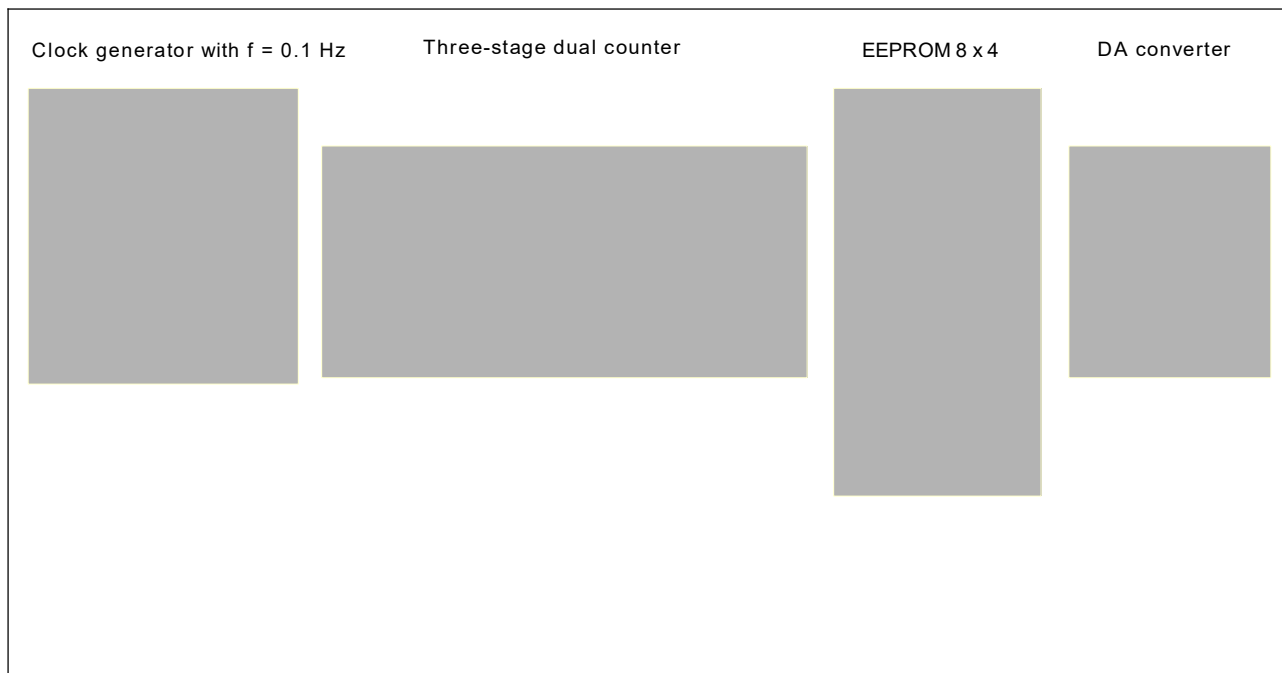


Fig. 12.2.4.1 Circuit of a digital function generator

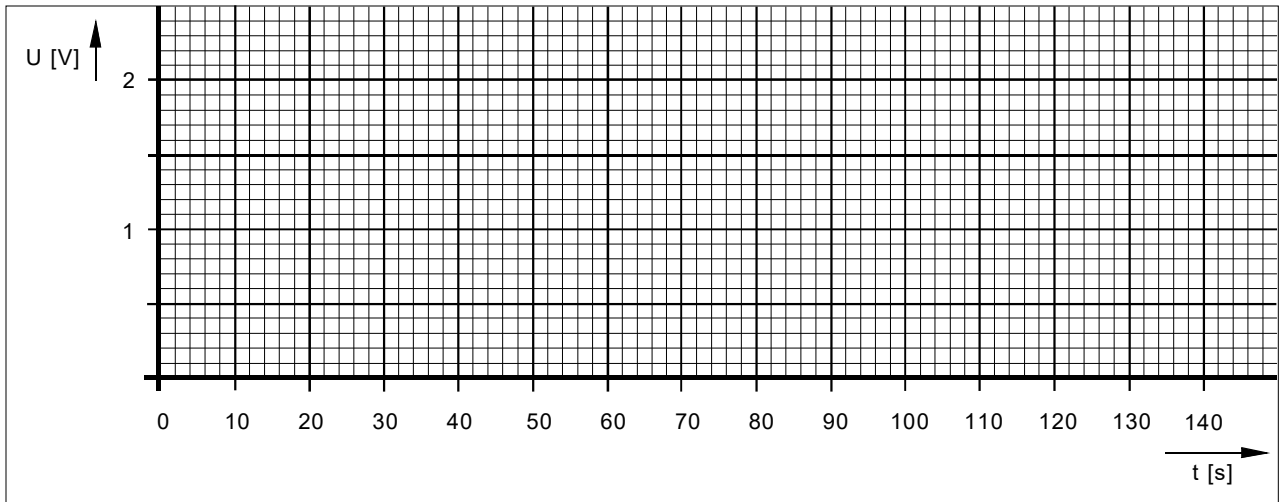


Fig. 12.2.4.2 Diagram

Experiment 2:

Experiment procedure:

- Enter the code words for the eight memory locations in table 12.2.4.2 if the displayed signal curve is to be achieved. Note the measured voltage values.
- Then draw the actual curve of the analog voltage in diagram 12.2.4.3.

Address	Code word	U [V]
0 0 0		
0 0 1		
0 1 0		
0 1 1		
1 0 0		
1 0 1		
1 1 0		
1 1 1		

Table 12.2.4.2

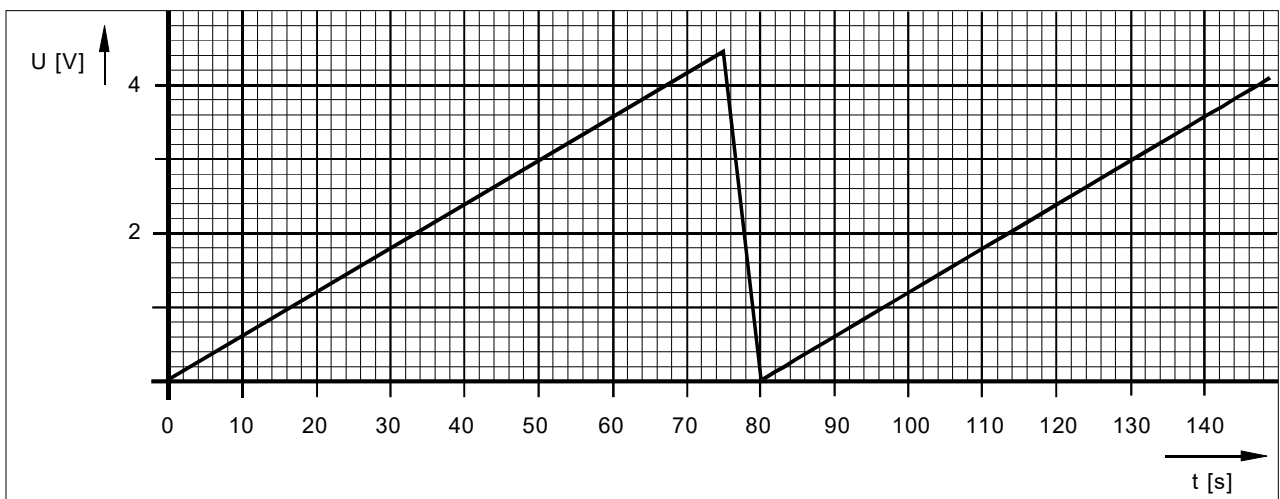


Fig. 12.2.4.3 Diagram

Notes: