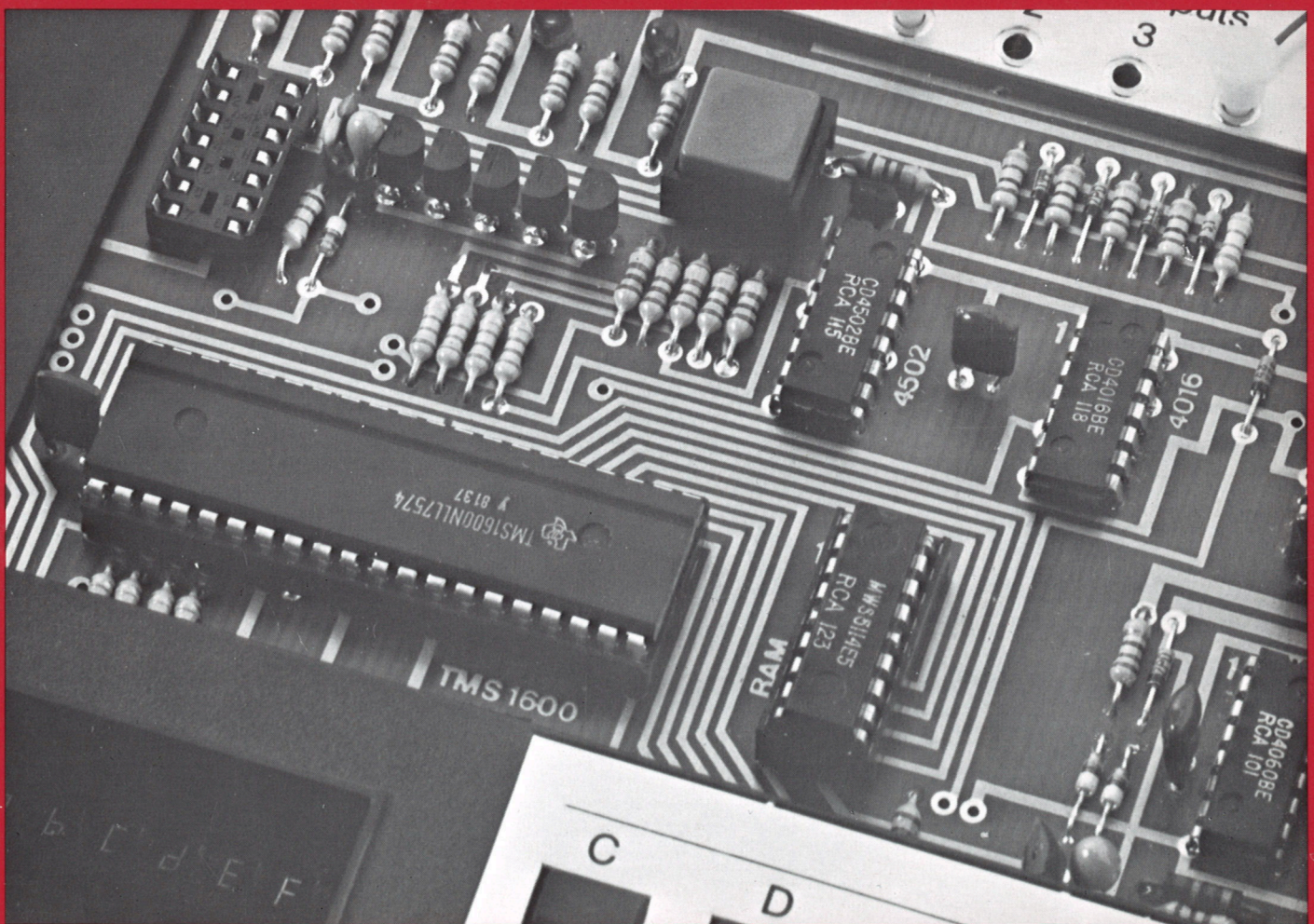


microtronic computer-system

2090

Instruction book Part 2

**Programming – Experimenting – learning playfully
how a computer works.**



In Zusammenarbeit mit
dem Elektronik-Magazin



Errata in the Microtronic Instruction Book, Part 2

On page 44: Timer with two switch-on and
switch-off times.

The input command at address 57 must correctly read: 529
(instead of D29).

Page 62: Computer counts
frequencies

The program printed in the book ends at address 1B. It
must be extended as follows:

1C	0A8	MOV A,8
1D	0B9	MOV B,9
1E	080	MOV 8,0
1F	091	MOV 9,1
20	C01	GOTO 01

Please excuse the oversight.



microtronic computer system

Instruction book Part 2

Introduction to
microprocessor technology.

Programs for
experimenting
with the microcomputer.

By Jörg Vallen

Production and distribution:
BUSCH-Modellspielwaren
P.O. Box 1360
D-6806 Viernheim

In cooperation with the
ELO-Magazin Franzis-
Verlag
P.O. Box 370120
8000 Munich 37



For some experiments,
additionally required:
A 9 V battery (IEC 6 F 22)
or power supply 2059

Copyright 1981 by
BUSCH GmbH. + Co. KG, Viernheim

All rights reserved.

In this part of the instruction book, a small selection of programs is demonstrated from the large palette of application possibilities of the Microtronic computer system. The program explanations assume that Part 1 of the instruction book has already been worked through.

Thanks to the introductory explanations, however, it is also possible — even without detailed computer knowledge — to enter the listed programs in order to get to know interesting computer games and other possibilities.

The following programs are available:

Computer Games

Tic-Tac-Toe	4
Sea Battle	6
Code-Breaker (a thinking and skill game)	9
Nim 2	12

Computer Functions with Decoder Tasks

Morse-Code Decoder	18
Morse Trainer	21
Resistor Colour-Code Calculation	24

Computer Calculations

Pocket-Calculator Program (with all arithmetic operations)	26
Sine Calculation (something for mathematicians)	32
Calculating the days between two dates	34
Calculating a weekday from entered dates	36
Calculating your personal biorhythm	37

Computer Functions with Peripheral Electronics

These experiments require additional BUSCH switching relays or BUSCH Electronic Studios.

Connecting relays	42
Timer with two switch-on and switch-off times	44
Alarm clock with two alarm times	47
Alarm clock with snooze radio	48
Further „timer“ possibilities	48
Computer organ	52
The composing computer	54
Computer controls an electronic music box	54
Acoustic effects for computer games	56
The computer counts people and objects	57
Data transmission with only 3 lines	59
Data transmission using the „handshaking“ method	61
The computer counts frequencies	62
Reaction-time meter	65
Digital voltmeter	68
Computer-controlled model railway	71

These programs serve as inspiration for how the computer can be reprogrammed again and again for the most diverse tasks.

Microtronic computer beginners, please note:

Before programming, always press the keys HALT – NEXT – 00 first. Then enter the command codes according to the program table. After each command-code entry, don't forget to press the NEXT key. After entering the last command code of a program, press the NEXT key once more, then start the program with HALT – NEXT – 00 – RUN. If the program does not work correctly, go back to the start of the program with HALT – NEXT – 00 and, by pressing the NEXT key, compare the entered commands with the program table and correct them if necessary (re-enter). After making the correction, don't forget to press the NEXT key and then restart the program with HALT – NEXT – 00 – RUN.

Tic-Tac-Toe

Tic-Tac-Toe is a well-known, simple board game. It has a playing field of 9 squares, with 3 × 3 squares arranged in a square (see illustration). Each player has 4 stones. Players take turns, and each player may place one stone on a free square. The winner is the player who is first to arrange 3 stones in a horizontal, vertical or diagonal row.

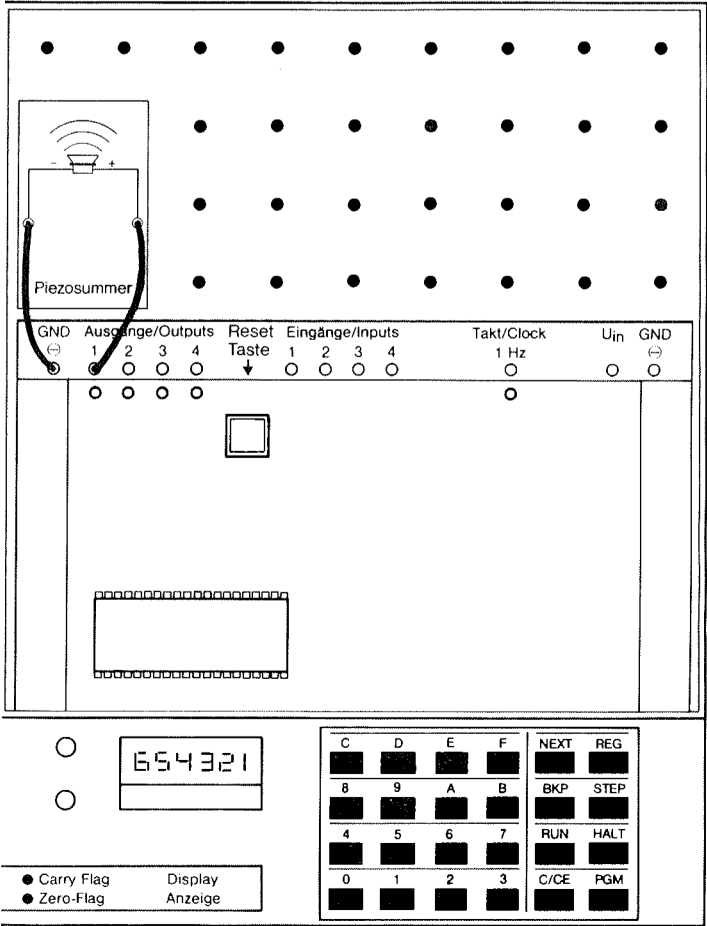
Among the computer accessories we find a Tic-Tac-Toe board with 4 black and 4 red playing pieces each.

First, after HALT – NEXT – 00, we must enter the program according to the table.

Program: Tic – Tac – Toe

Label	Addr.	Code	Mnemonic	Jump to
	00	F08	CLEAR	
	01	F0D	EXRL	
	02	F08	CLEAR	
	03	FE0	DOT 0	
	04	198	MOVI 9,8	
	05	F28	DISP 2,8	
	06	FF9	KIN 9	
	07	51A	ADDI 1,A	
	08	94A	CMPI 4,A	
	09	E21	BRZ 21	„EE“
	0A	09D	MOV 9,D	
	0B	51D	ADDI 1,D	
	0C	B2B	CALL 2B	„BERE“
	0D	0C8	MOV C,8	
	0E	FF9	KIN 9	
	0F	08D	MOV 8,D	
	10	54D	ADDI 4,D	
	11	B2B	CALL 2B	„BERE“
	12	89C	CMP 9,C	
	13	E07	BRZ 07	
	14	0C8	MOV C,8	
	15	1CA	MOVI C,A	
	16	1CB	MOVI C,B	
	17	F48	DISP 4,8	
	18	10F	MOVI 0,F	
	19	FEF	DOT F	
	1A	516	ADDI 1,6	
	1B	D1F	BRC 1F	
	1C	1FF	MOVI F,F	
	1D	FEF	DOT F	
	1E	C18	GOTO 18	
	1F	FF0	KIN 0	
	20	C00	GOTO 00	
„EE“	21	09D	MOV 9,D	
	22	55D	ADDI 5,D	
	23	B2B	CALL 2B	„BERE“
	24	0C8	MOV C,8	
	25	1EA	MOVI E,A	
	26	1EB	MOVI E,B	
	27	1FF	MOVI F,F	
	28	FEF	DOT F	
	29	F48	DISP 4,8	
	2A	C1F	GOTO 1F	
„BERE“	2B	0DC	MOV D,C	
	2C	71D	SUBI 1,D	
	2D	10E	MOVI 0,E	
	2E	F03	HXDZ	
	2F	0D0	MOV D,0	
	30	0E1	MOV E,1	
	31	F0D	EXRL	
	32	180	MOVI 8,0	
	33	F0C	DIV	
	34	F0D	EXRL	
	35	101	MOVI 0,1	
	36	180	MOVI 8,0	
	37	F0B	MULT	
	38	60C	SUB 0,C	
	39	F07	RET	

Connecting the piezo buzzer



Now connect the piezo buzzer (see fig. below). Then start the program: HALT – NEXT – 00 – RUN

The display shows: **09**. This means the computer places its first playing stone on field 9. We help it by placing a red marker on field 9.

1	2	3
8	●	4
7	6	5

We place a white stone on field 3 and enter the digit **3**. The display now shows: **34**. The computer thus confirms it has registered our field 3, and that it wants its own red stone on field 4.

1	2	○
8	●	●
7	6	5

So the computer cannot complete the diagonal row of stones, we place a white stone on field 8 (enter **8**). The display reports: **81** – the computer wants a red stone on field 1.

●	2	○
○	●	●
7	6	5

We place our white stone, say, on field 6 (enter **6**). The display reports: **CC65** and an interrupted buzzing tone sounds.

●	2	○
○	●	●
○	○	●

The message **CC65** means the computer will place its next red stone on field 5, so that 3 red stones now form a diagonal row. The computer has won.

As soon as any number key is pressed, a new game can begin, with the computer again placing its first stone on field 9.

Should we manage to force a „draw“, the display reports **EE** and twice the digit we last entered. This would also have been the digit the computer needed for a win. Since neither player can complete a row, the game ends in a „draw“. A continuous tone sounds until a new game is started by pressing a number key.

We will find it is impossible to win against the computer. A „draw“ is the best result we can achieve.

This is of interest to the programmer:

The computer's actual calculations are carried out in a subprogram (from address 2B to address 39). This subprogram „BERE“ (CALCULATION) is also shown in the program flow chart. The subprogram „BERE“ is called from addresses 0C, 11 or 23.

First, the value of register D is stored into register C. Then the computer calculates the following formula:

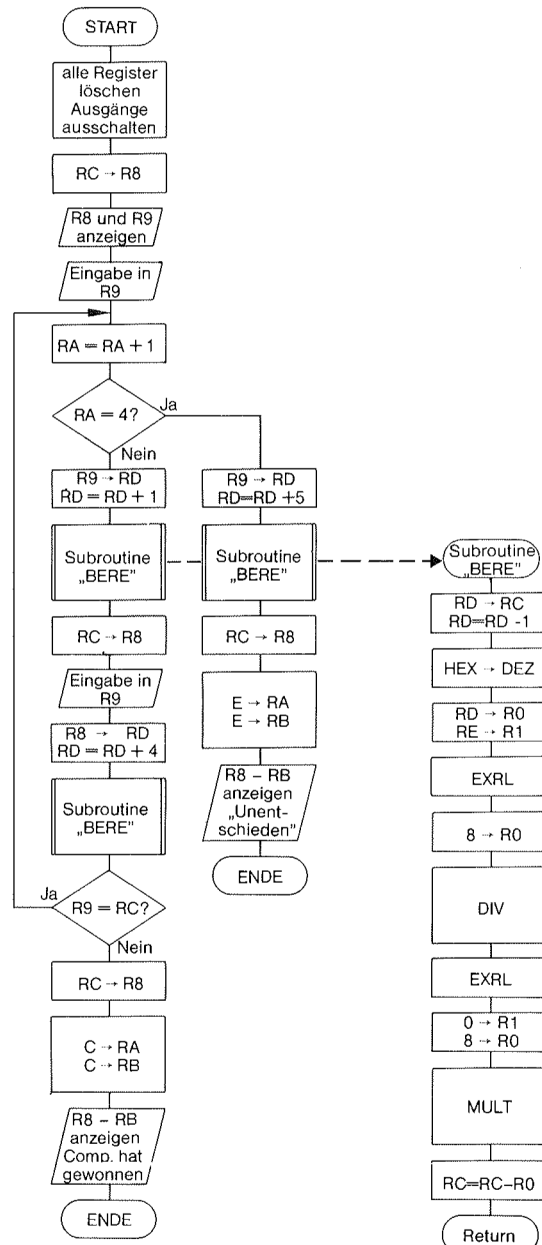
$$RC \text{ (Register C)} = RC - 8 \times ((RD - 1) : 8)$$

The value in register D is reduced by 1 and divided by 8. Suppose register D holds the value 4; reduced by 1, this leaves 3. $3 \div 8$ gives 0 remainder 3. The remainder is not needed in the following calculations, so the next step gives $8 \times 0 = 0$. Since register C held the value 4 and 0 is subtracted, 4 remains. This clever formula ensures that a field between 1 and 9 is always occupied, preferably in a row or a diagonal.

Comparing the program table with the program flow chart shows us the individual functions.

In the program table, the „Label“ column marks the addresses at which the jumped-to program parts or subroutines begin. The „Jump to“ column likewise marks the addresses at which the jumps begin.

Program flow chart = Tic-Tac-Toe



„Sea Battle“ or „Battle in Space“

One could give this game many names. The strategy is as follows: on a playing field of $5 \times 5 = 25$ squares the computer hides 4 enemy ships. By entering the field coordinates, the ships must be found and shot down, i.e. sunk.

First, the game must be programmed. After HALT – NEXT – 00 we carefully make the entry (disconnect the piezo buzzer):

Program: Sea Battle

Label	Addr.	Code	Mnemonic	Jump to
	00	F02	DISOUT	
	01	F08	CLEAR	
	02	FE0	DOT 0	
	03	B53	CALL 53	„RND“
	04	0D0	MOV D, 0	
	05	0E1	MOV E, 1	
	06	B53	CALL 53	„RND“
	07	B6D	CALL 6D	„V1“
	08	D06	BRC 06	
	09	0D2	MOV D, 2	
	0A	0E3	MOV E, 3	
	0B	B53	CALL 53	„RND“
	0C	B68	CALL 68	„V2“
	0D	D0B	BRC 0B	
	0E	0D4	MOV D, 4	
	0F	0E5	MOV E, 5	
	10	B53	CALL 53	„RND“
	11	B63	CALL 63	„V3“
	12	D10	BRC 10	
	13	0D6	MOV D, 6	
	14	0E7	MOV E, 7	
	15	10D	MOVI 0, D	
	16	10E	MOVI 0, E	
	17	10F	MOVI 0, F	
	18	F28	DISP 2, 8	
	19	FF9	KIN 9	
	1A	F28	DISP 2, 8	
	1B	FF8	KIN 8	
	1C	F02	DISOUT	
	1D	51D	ADDI 1, D	
	1E	FBE	ADC E	
	1F	880	CMP 8, 0	
	20	E22	BRZ 22	
	21	C24	GOTO 24	
	22	891	CMP 9, 1	
	23	E38	BRZ 38	„R0“
	24	882	CMP 8, 2	
	25	E27	BRZ 27	
	26	C29	GOTO 29	
	27	893	CMP 9, 3	
	28	E3A	BRZ 3A	
	29	884	CMP 8, 4	
	2A	E2C	BRZ 2C	
	2B	C2E	GOTO 2E	
	2C	895	CMP 9, 5	

Label	Addr.	Code	Mnemonic	Jump to
	2D	E3C	BRZ 3C	„R4“
	2E	886	CMP 8, 6	
	2F	E31	BRZ 31	
	30	C33	GOTO 33	
	31	897	CMP 9, 7	
	32	E3E	BRZ 3E	„R6“
	33	1FC	MOVI F, C	
	34	FEC	DOT C	
	35	10C	MOVI 0, C	
	36	FEC	DOT C	
	37	C18	GOTO 18	
„R0“	38	100	MOVI 0, 0	
	39	C40	GOTO 40	
„R2“	3A	102	MOVI 0, 2	
	3B	C40	GOTO 40	
„R4“	3C	104	MOVI 0, 4	
	3D	C40	GOTO 40	
„R6“	3E	106	MOVI 0, 6	
	3F	1FC	MOVI F, C	
	40	1FA	MOVI F, A	
	41	FEA	DOT A	
	42	10A	MOVI 0, A	
	43	FEA	DOT A	
	44	71C	SUBI 1, C	
	45	E47	BRZ 47	
	46	C40	GOTO 40	
	47	51F	ADDI 1, F	
	48	94F	CMPI 4, F	
	49	E4E	BRZ 4E	
	4A	1AA	MOVI A, A	
	4B	1AB	MOVI A, B	
	4C	F48	DISP 4, 8	
	4D	C19	GOTO 19	
	4E	10F	MOVI 0, F	
	4F	F03	HXDZ	
	50	F3D	DISP 3, D	
	51	FF0	KIN 0	
	52	C00	GOTO 00	
„RND“	53	F05	RND	
	54	90D	CMPI 0, D	
	55	E59	BRZ 59	
	56	95D	CMPI 5, D	
	57	D59	BRC 59	
	58	C5B	GOTO 5B	
	59	59D	ADDI 9, D	
	5A	C54	GOTO 54	
	5B	90E	CMPI 0, E	
	5C	E60	BRZ 60	
	5D	95E	CMPI 5, E	
	5E	D60	BRC 60	
	5F	C62	GOTO 62	
	60	53E	ADDI 3, E	
	61	C5B	GOTO 5B	

Label	Addr.	Code	Mnemonic	Jump to
	62	F07	RET	
„V3“	63	8D4	CMP D, 4	
	64	E66	BRZ 66	
	65	C68	GOTO 68	
	66	8E5	CMP E, 5	
	67	E74	BRZ 74	„STC“
„V2“	68	8D2	CMP D, 2	
	69	E6B	BRZ 6B	
	6A	C6D	GOTO 6D	
	6B	8E3	CMP E, 3	
	6C	E74	BRZ 74	„STC“
„V1“	6D	8D0	CMP D, 0	
	6E	E70	BRZ 70	
	6F	C72	GOTO 72	
	70	8E1	CMP E, 1	
	71	E74	BRZ 74	„STC“
	72	F0A	RSC	
	73	F07	RET	
„STC“	74	F09	STC	
	75	F07	RET	

Reconnect the piezo buzzer, then start the program: HALT – NEXT – 00 – RUN.

The display stays dark for 2 to 3 seconds. The computer is working, as can be seen from the blinking Carry and Zero flags. The display then shows: 00.

Meanwhile the computer has — invisibly to us — distributed the 4 ships across the playing field. The distribution is made using the random-number generator. The ships are therefore hidden in a different arrangement in every game. Let us assume that the ships are located at the positions marked with crosses on the playing plan (see fig.).

	1	2	3	4	5
1					
2		✕			
3					
4				✕	
5	✕			✕	

We can enter two digits. The first digit applies only to the vertical playing-field numbers, the second digit to the horizontal playing-field numbers. As an example we enter 42. The computer responds with a short tone and shows 42. In our example this shot was not a hit.

				1. Ziffer		
		1	2	3	4	5
1						
2. Ziffer	2					
	3					
	4					
	5					

Nicht getroffen

If we were now to enter, for example, 15, the computer produces an interval tone and the display reports: AA15. The first ship has been sunk. If we then entered the same number combination once more, no hit would be reported at this point, because the ship has already been sunk.

		1. Ziffer					
		1	2	3	4	5	
	1						
	2						
	3						
	4						
2. Ziffer	5						Treffer

As soon as the fourth ship has also been hit, the computer reports back with the interval tone and 3 digits are shown on the display. We can read off how many attempts were needed to hit all 4 ships.

After pressing any number key, 4 ships are hidden again, the display shows 00 once more, and a new game can begin.

This is of interest to the programmer:

We compare the description with the program flow chart and the program table.

The game is relatively easy to program, because there is no special winning strategy. First the ships must be distributed. Random numbers between 1 and 5 are stored in registers 0 to 7. The coordinates of the first ship are in registers 0 and 1, those of the second ship in registers 2 and 3, and so on.

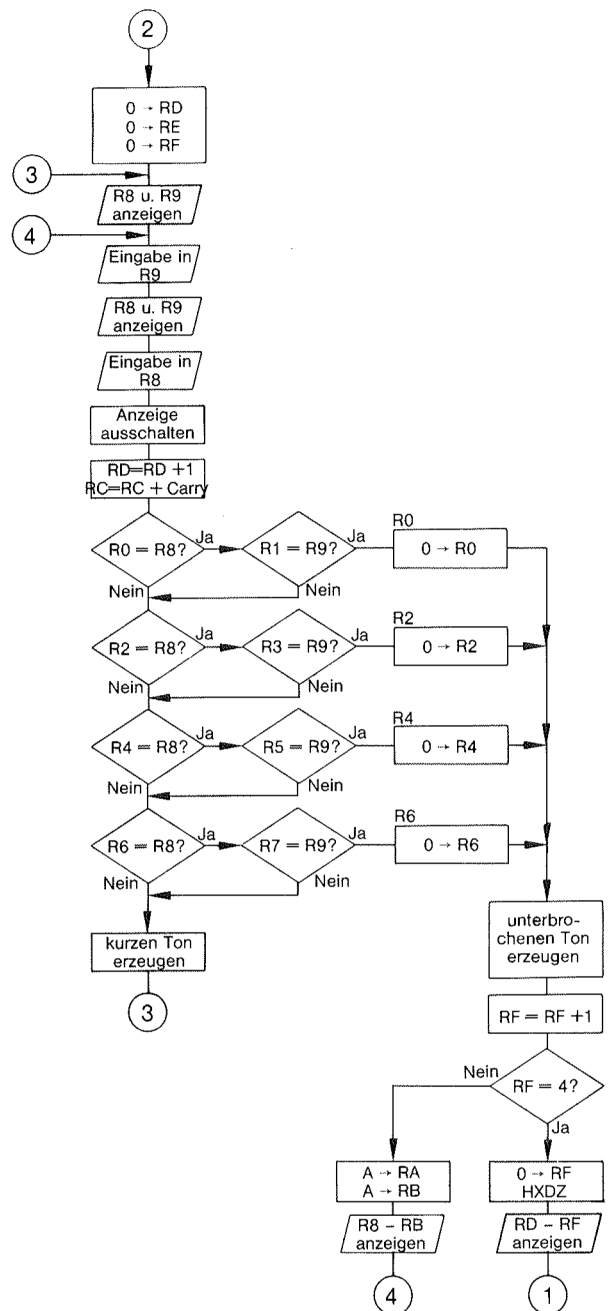
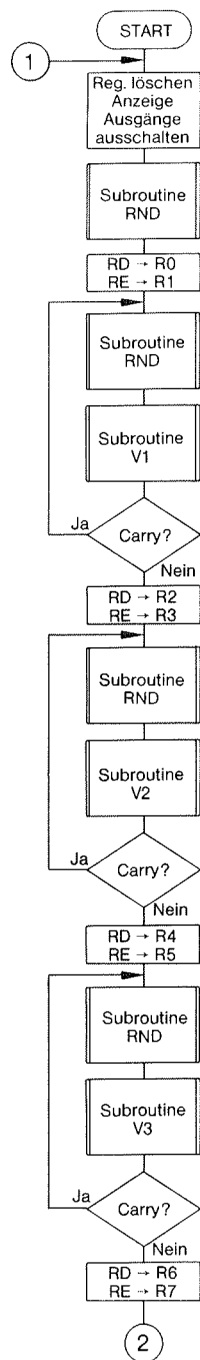
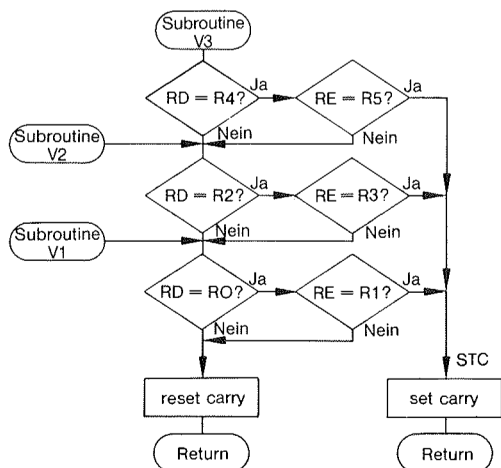
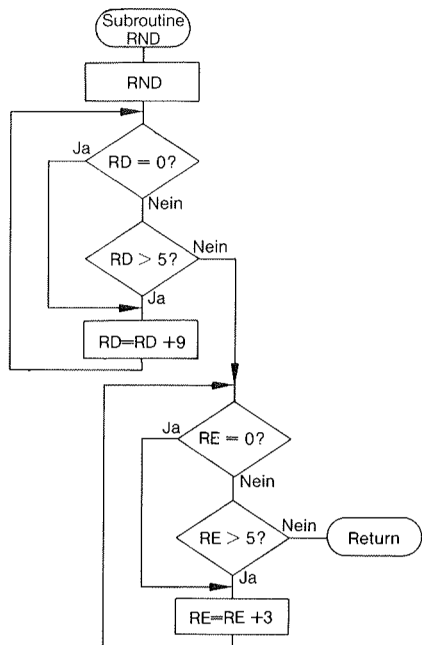
By means of the **subroutines** designated **V1** (V = COMPARE), **V2** and **V3**, it is ensured that no square is occupied twice when the ships are distributed.

The entered coordinates are then stored in registers 8 and 9. Comparison commands are used to check whether the entered coordinates match the coordinates of a hidden ship.

By means of the **subroutine RND** a random number is stored in registers D, E and F. The random numbers in registers D and E yield the coordinates of the hidden ships, which is why only the numbers between 1 and 5 are usable. Comparisons are used to check whether a random number of 0 or greater than 5 appears in these registers. If „yes“, the value 9 is added to register D (or the value 3 to register E) for as long as it takes until both random numbers have values between 1 and 5.

Flow chart Sea Battle (see page 8)

Program flow chart Sea Battle



In the program table, the „Label“ column has been used to indicate the addresses at which the referenced program sections — or subroutines — begin. In the „Jump to“ column it has likewise been indicated at which addresses the jumps begin.

Code Breaker

Code breakers are puzzle games. The best-known kinds of such code games are, for example, Master Mind and Super-Hirn.

In our Code Breaker, a 4-digit number code must be guessed.

Using its built-in random-number generator, the computer determines a 4-digit number combination, which is stored invisibly. It is not only the correct numbers that must be guessed, but the entire number combination in the correct order. The computer provides all information — for example whether a correct number has been found, whether it is in the correct position, and so on — and at the end of the game it announces how many guessing attempts were needed.

To be able to try out this interesting game, after HALT – NEXT – 00 we enter the following program:

Program: Code

Label	Addr.	Code	Mnemonic	Jump to
	00	F08	CLEAR	
	01	F49	DISP 4, 9	
	02	FFC	KIN C	
	03	FFB	KIN B	
	04	FFA	KIN A	
	05	FF9	KIN 9	
	06	F02	DISOUT	
	07	F05	RND	
	08	4D9	ADD D, 9	
	09	4EA	ADD E, A	
	0A	4FB	ADD F, B	
	0B	4DC	ADD D, C	
	0C	579	ADDI 7, 9	
	0D	999	CMPI 9, 9	
	0E	D0C	BRC 0C	
	0F	57A	ADDI 7, A	
	10	99A	CMPI 9, A	
	11	D0F	BRC 0F	
	12	8A9	CMP A, 9	
	13	E0F	BRZ 0F	
	14	57B	ADDI 7, B	
	15	99B	CMPI 9, B	
	16	D14	BRC 14	
	17	8B9	CMP B, 9	
	18	E14	BRZ 14	
	19	8BA	CMP B, A	
	1A	E14	BRZ 14	
	1B	57C	ADDI 7, C	
	1C	99C	CMPI 9, C	
	1D	D1B	BRC 1B	
	1E	8C9	CMP C, 9	
	1F	E1B	BRZ 1B	
	20	8CA	CMP C, A	
	21	E1B	BRZ 1B	
	22	8CB	CMP C, B	
	23	E1B	BRZ 1B	
	24	10D	MOVI 0, D	
	25	10E	MOVI 0, E	
	26	10F	MOVI 0, F	

Label	Addr.	Code	Mnemonic	Jump to
	27	F61	DISP 6, 1	
	28	FF0	KIN 0	
	29	9A0	CMPI A, 0	
	2A	E33	BRZ 33	„Vergl.“
	2B	D31	BRC 31	„Zahl“
	2C	034	MOV 3, 4	
	2D	023	MOV 2, 3	
	2E	012	MOV 1, 2	
	2F	001	MOV 0, 1	
	30	C27	GOTO 27	
„Zahl“	31	F49	DISP 4, 9	
	32	C60	GOTO 60	
„Vergl.“	33	F02	DISOUT	
	34	105	MOVI 0, 5	
	35	106	MOVI 0, 6	
	36	017	MOV 1, 7	
	37	098	MOV 9, 8	
	38	B65	CALL 65	„SR“
	39	0A8	MOV A, 8	
	3A	B6A	CALL 6A	„ZR“
	3B	0B8	MOV B, 8	
	3C	B6A	CALL 6A	„ZR“
	3D	0C8	MOV C, 8	
	3E	B6A	CALL 6A	„ZR“
	3F	027	MOV 2, 7	
	40	B6A	CALL 6A	„ZR“
	41	0B8	MOV B, 8	
	42	B6A	CALL 6A	„ZR“
	43	0A8	MOV A, 8	
	44	B65	CALL 65	„SR“
	45	098	MOV 9, 8	
	46	B6A	CALL 6A	„ZR“
	47	037	MOV 3, 7	
	48	B6A	CALL 6A	„ZR“
	49	0A8	MOV A, 8	
	4A	B6A	CALL 6A	„ZR“
	4B	0B8	MOV B, 8	
	4C	B65	CALL 65	„SR“
	4D	0C8	MOV C, 8	
	4E	B6A	CALL 6A	„ZR“
	4F	047	MOV 4, 7	
	50	B65	CALL 65	„SR“
	51	0B8	MOV B, 8	
	52	B6A	CALL 6A	„ZR“
	53	0A8	MOV A, 8	
	54	B6A	CALL 6A	„ZR“
	55	098	MOV 9, 8	
	56	B6A	CALL 6A	„ZR“
	57	51D	ADDI 1, D	
	58	FBE	ADC E	
	59	FBF	ADC F	
	5A	946	CMPI 4, 6	
	5B	E5D	BRZ 5D	

Label	Addr.	Code	Mnemonic	Jump to
	5C	C27	GOTO 27	
	5D	1E0	MOVI E, 0	
	5E	1E5	MOVI E, 5	
	5F	F60	DISP 6, 0	
	60	FF0	KIN 0	
	61	F03	HXDZ	
	62	F3D	DISP 3, D	
	63	FF0	KIN 0	
	64	C00	GOTO 00	
„SR“	65	878	CMP 7, 8	
	66	E68	BRZ 68	
	67	C69	GOTO 69	
	68	516	ADDI 1, 6	
	69	F07	RET	
„ZR“	6A	878	CMP 7, 8	
	6B	E6D	BRZ 6D	
	6C	C6E	GOTO 6E	
	6D	515	ADDI 1, 5	
	6E	F07	RET	

Program start: HALT – NEXT – 00 – RUN.

The display shows: 0000. Now we have to help the computer find a random number combination. To do this we enter 4 arbitrary numbers or letters (this entry has nothing to do with the number combination to be guessed later). The display switches off for 1 second – then 000000 is shown. The computer has determined the number combination to be guessed and stored it invisibly.

For the guessing process we can always enter 4 numbers between 0 and 9, for example 1 – 2 – 3 – 4. The entered numbers are displayed. By pressing key **A**, the computer compares the entered number values with the invisibly stored number combination. The display shows a 6-digit readout, in which the four rightmost digits continue to show the previously entered number combination (e.g. 1 – 2 – 3 – 4). If, for example, 00 is shown on the two left digits, this means that no number was guessed correctly.

If the display shows, for example, „01“ on the two left digit positions, this would mean:

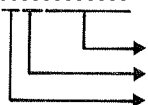
0 = no correctly guessed number in the correct position.

1 = one correct number, but in the wrong position.

A reading of, for example, 21 on the two left display positions would mean that we have guessed 2 correct digits in the correct position, and that we have guessed 1 further correct number which, however, is in the wrong position. After we have entered the number combination 1 – 2 – 3 – 4, it would be established that three of these numbers were guessed correctly, of which two are in the correct position, while one is in the wrong position and one number in the combination invisibly fixed by the computer still has to be guessed.

The following small table once again demonstrates to us how the display shows the values:

X X X X X X



= entered number combination
 = correct number, but in the wrong position
 = correct number, in the correct position

We keep guessing. Enter four new numbers – press key **A**. As soon as the display reports correct numbers on the two left digit positions, we should write down the entered number combinations, so that by shifting and re-entering the individual digits we can guess the correct number combination step by step.

The Code Breaker is an interesting game and it requires a lot of logical thinking to find the correct number combination. With every new entry, do not forget to press key **A**.

If after some time we were to find the correct number combination, for example by entering the numbers 6 – 8 – 5 – 7 (then key **A**), the display reports: **E6857E**. If we then press key **A** once more, the display shows in three digits how many attempts were needed.

If key **A** is pressed again, the display reports, as at the start of the game: 0000, and as soon as 4 arbitrary number keys have been pressed, a new Code Breaker game can begin.

If the course of the game takes too long for us, we can have the computer display the number combination it has fixed by pressing key **B**. However, this aborts the game, and when key **A** is pressed again the number of guessing attempts made so far is shown on the display.

The program must be entered very carefully, because if a command is entered incorrectly the computer's ongoing calculations will not be right and thus incorrect results will be displayed. We can test this very easily if, at the beginning after a few guessing attempts, we have the number combination to be guessed displayed by pressing key **B**. If necessary, check and correct the program with HALT – NEXT – 00 and further presses of the NEXT key.

This is of interest to the programmer:

We again compare the program table and the flow chart. The first part of the program (addresses 00 to 23) provides the four different random numbers that have to be guessed later. Although we determine these random numbers with the help of the random-number generator integrated in the microprocessor, it is difficult to effectively obtain four independent (mutually different) random numbers. Suppose one were simply to store the resulting random numbers in the 4 registers by means of 4 RND commands placed one after another – then numbers „dependent“ on one another would arise. If, for example, the first random number were the digit 1, the next 3 digits would depend on this number; i.e. every time the first random number turns out to be the digit 1, the same number combination would result. This happens because the execution times of the commands are always equal in length, and so the „random counter“ also counts on by the same number of steps within this same time interval.

To obtain four mutually independent random numbers, the following trick was used: 4 arbitrary digits are entered, and to each of these a random number is added. It is then checked whether these numbers are smaller than 9. If this is not the case, the value 7 is added repeatedly until values result that are 9 or smaller than 9. In this way the hexadecimal digits A to F are eliminated.

In addition, the four random numbers are checked to see whether perhaps an identical number is present twice. If so, the value 7 is again added until all 4 random numbers are different. In this way four different and mutually independent random numbers are determined.

At address 28, the „guessing numbers“ are entered.

At address 31 the random numbers can be displayed. At addresses 57 to 59 the number of guessing attempts is counted hexadecimally in registers D to F. The comparison command at address 5A checks whether register 6 contains the value 4. If so, the correct number combination has been guessed.

In the program table, the first column also gives notes on the addresses at which the individual jump targets begin, while the last column contains notes on the addresses at which corresponding jumps – for example to the subroutines – are carried out.

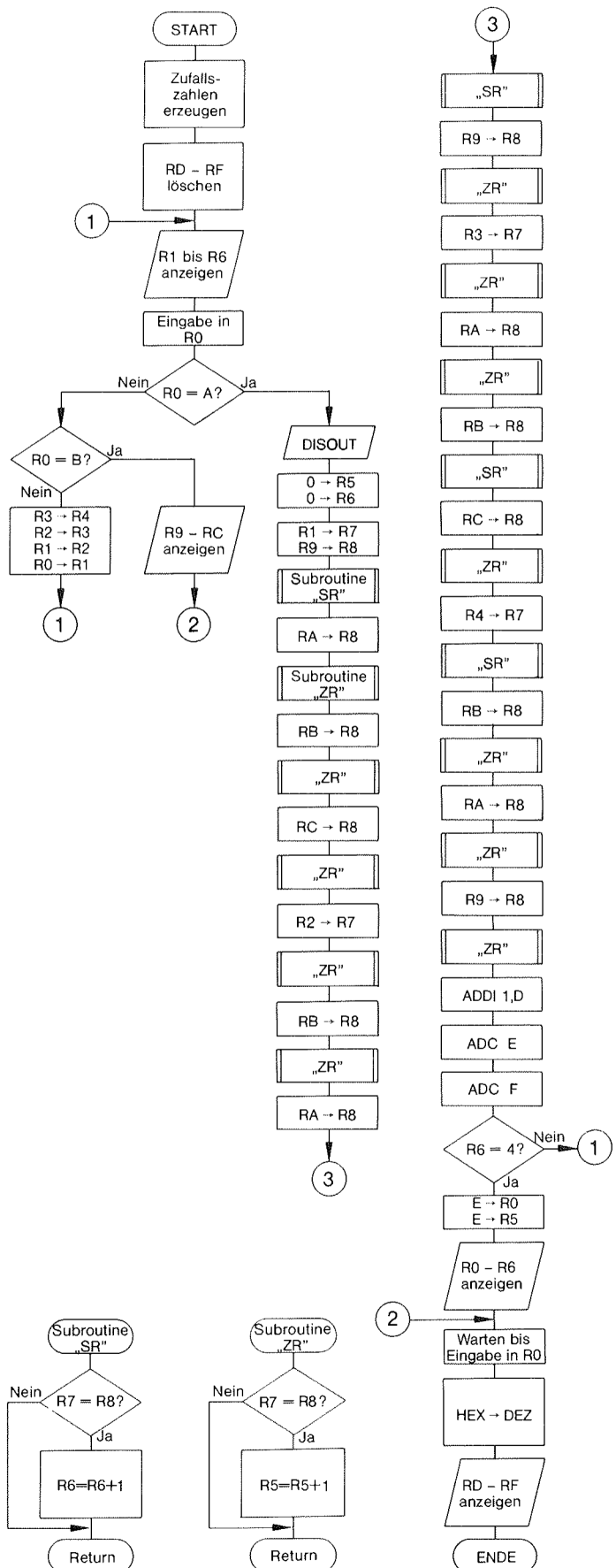
The individual working steps can be seen from the program flow chart. In registers 1 to 4 the „guessing number“ devised by the player is entered, and then each individual entered digit is compared with the computer's random numbers stored in registers 9 to C. This gives rise to the jumps to the subroutine „SR“ (= position correct?), or to the subroutine „ZR“ (= number correct?). The subroutine „SR“ checks whether the correct number is in the correct position. If this is the case, the value 1 is added to register 6.

In the subroutine „ZR“ it is checked whether a correct number has indeed been entered but is in the wrong position. If so, the value 1 is added to register 5.

The number of attempts is counted hexadecimally in registers D, E and F. Only when the number of attempts is to be displayed is the hexadecimal result converted into decimal numbers.

It is impossible, with these longer programs, to explain every single program step (as in the first part of the instruction book). Anyone who takes the trouble to work through the program flow chart consistently, however, will quickly understand the meaning and the operation of the program.

Program flow chart: Code Breaker



„Take 2“

In the first part of the instruction book we got to know a simple version of the Nim game (fixed program PGM 7). The following program is a far more interesting variant of the Nim game. Whereas in the simple version we had only one pile from which the matchsticks were to be removed, there are now three piles, from which the matchsticks can be removed alternately, one pile at a time. At least 1 matchstick must be taken, but a whole pile may also be removed. The winner is the one who can take the last (or the last few) matchsticks. Note the difference from the earlier Nim-game version, in which the player who had to take the last matchstick lost the game.

The new version „Take 2“ can be played at three different skill levels. To say it in advance: at skill level 3 it is almost impossible to win against the computer.

Before the course of the game is explained, first enter the program according to the table after HALT – NEXT – 00.

Program: Take 2

Label	Adr.	Code	Mnemonic	Jump to	Explanation
	00	F08	CLEAR		
	01	FE0	DOT 0		
	02	F10	DISP 1, 0		
	03	FF8	KIN 8		Entering the skill level
	04	F61	DISP 6, 1		
	05	FF3	KIN 3		Entering the pile sizes
	06	FF2	KIN 2		
	07	FF1	KIN 1		
	08	FF6	KIN 6		From which pile?
	09	FF5	KIN 5		„how many?“
	0A	905	CMPI 0, 5		Zero?
	0B	E23	BRZ 23	„SCOMP“	Yes, then the computer's first move
	0C	C12	GOTO 12		
	0D	105	MOVI 0, 5		
	0E	FF6	KIN 6		„which pile?“
	0F	F61	DISP 6, 1		
	10	FF5	KIN 5		„how many?“
	11	E10	BRZ 10		if zero, repeat „how many?“
	12	916	CMPI 1, 6		Pile 1?
	13	E1E	BRZ 1E		
	14	926	CMPI 2, 6		Pile 2?
	15	E1A	BRZ 1A		
	16	815	CMP 1, 5		Is the entry greater than pile 3?
	17	D0E	BRC 0E		Yes, then re-enter
	18	651	SUB 5, 1		
	19	C21	GOTO 21	„SPGEW“	
	1A	825	CMP 2, 5		Entry greater than pile 2?
	1B	D0E	BRC 0E		Yes, then re-enter
	1C	652	SUB 5, 2		
	1D	C21	GOTO 21	„SPGEW“	
	1E	835	CMP 3, 5		Entry greater than pile 1?

Label	Adr.	Code	Mnemonic	Jump to	Explanation
	1F	D0E	BRC 0E		Yes, then re-enter
	20	653	SUB 5, 3		
„SPGEW“	21	B5C	CALL 5C	„GEW“	All three piles zero?
	22	D4E	BRC 4E	„SPGEW“	Yes, then the player has won
„SCOMP“	23	F02	DISOUT		Calculation of the computer's move
	24	01D	MOV 1, D		Pile 3 (register 1) and pile 2 are stored in reg. D and E and added without carry
	25	02E	MOV 2, E		
	26	B66	CALL 66	„XOR“	
	27	8D3	CMP D, 3		Result (reg. D) smaller than pile 1 (reg. 3)?
	28	E35	BRZ 35	„-1“	(if equal, then jump to „-1“)
	29	D42	BRC 42	„E1“	Yes, jump to „E1“
	2A	03D	MOV 3, D		
	2B	01E	MOV 1, E		
	2C	B66	CALL 66	„XOR“	
	2D	8D2	CMP D, 2		
	2E	E35	BRZ 35	„-1“	
	2F	D48	BRC 48	„E2“	
	30	02D	MOV 2, D		
	31	03E	MOV 3, E		
	32	B66	CALL 66	„XOR“	
	33	8D1	CMP D, 1		
	34	D4A	BRC 4A	„E3“	
„-1“	35	901	CMPI 0, 1		„Default move“: take one away from one of the three piles
	36	E39	BRZ 39		
	37	711	SUBI 1, 1		
	38	C3E	GOTO 3E	„CGEW“	
	39	902	CMPI 0, 2		
	3A	E3D	BRZ 3D		
	3B	712	SUBI 1, 2		
	3C	C3E	GOTO 3E	„CGEW“	
	3D	713	SUBI 1, 3		
„CGEW“	3E	B5C	CALL 5C	„GEW“	All three piles zero?
	3F	D59	BRC 59	„CGEW“	Yes, then the computer has won
	40	F31	DISP 3, 1		Otherwise display the piles anew
	41	C0D	GOTO 0D		
„E1“	42	918	CMPI 1, 8		at skill level 1 or 2 (which is in register 8) carry out only one „default move“
	43	E35	BRZ 35	„-1“	
	44	928	CMPI 2, 8		
	45	E35	BRZ 35	„-1“	
	46	0D3	MOV D, 3		Otherwise RD is the result
	47	C3E	GOTO 3E		
„E2“	48	0D2	MOV D, 2		RD is the result (pile 2)
	49	C3E	GOTO 3E		
„E3“	4A	918	CMPI 1, 8		Skill level 1?
	4B	E35	BRZ 35	„-1“	Yes, then default move („-1“)
	4C	0D1	MOV D, 1		Otherwise reg. D is the result (pile 3)
	4D	C3E	GOTO 3E		

Label	Adr.	Code	Mnemonic	Jump to	Explanation
„SPGEW“	4E	1F0	MOVI F, 0		Display: player has won
	4F	FE1	DOT 1		
	50	519	ADDI 1, 9		
	51	E54	BRZ 54		
	52	FE0	DOT 0		
	53	C4F	GOTO 4F		
	54	1E5	MOVI E, 5		
	55	1E6	MOVI E, 6		
„CGEW“	56	F61	DISP 6, 1		
	57	FF0	KIN 0		
	58	C00	GOTO 00		
	59	1F0	MOVI F, 0		Display: computer has won
„GEW“	5A	FE0	DOT 0		
	5B	C54	GOTO 54		
	5C	901	CMPI 0, 1		Subroutine „GEW“ checks whether all three piles are zero
	5D	D64	BRC 64		
	5E	902	CMPI 0, 2		
	5F	D64	BRC 64		
	60	903	CMPI 0, 3		
	61	D64	BRC 64		
	62	F09	STC		all piles zero: set carry flag
	63	F07	RET		
	64	F0A	RSC		otherwise reset carry flag
„XOR“	65	F07	RET		
	66	0EC	MOV E, C		Subroutine „XOR“: registers D and E are added without carry. The result is in reg. D
	67	2DC	AND D, C		
	68	AED	OR E, D		
	69	F8D	INV D		
	6A	ACD	OR C, D		
	6B	F8D	INV D		
	6C	F07	RET		

Now the piezo buzzer should be connected as usual. Then start the program: HALT – NEXT – 00 – RUN.

The display shows 0. The computer wants to know the skill level – we enter, for example, a 1. The display now shows: 000000.

It must be specified how many matchsticks are to lie on the three piles. Example: pile no. 1 has 3 matchsticks (enter 3). Pile no. 2 has 4 matchsticks (enter 4) and pile no. 3 has 5 matchsticks (enter 5). The display now shows 000345.

We want to take 1 matchstick from the second pile (entry 2 1). The display should now show: 335 – but in fact the display shows: 334, because the computer has likewise taken 1 matchstick, namely from the third pile.

We now take the remaining 4 matchsticks from the third pile (entry 3 4). The display should read: 330. But after the computer likewise takes 1 matchstick from the second pile, the display actually shows: 320.

We now take 1 matchstick from the first pile (entry 1 1) and, taking into account that the computer simultaneously takes 1 matchstick from the second pile as well, the display shows: 210. Since the computer too can only ever take from one pile, we arrange things so that, after our removal, 1 matchstick remains on each of the remaining first and second piles.

So we take 1 of the 2 matchsticks of the first pile away (entry 1 1). The computer takes the last remaining matchstick from the second pile, and the display shows: 100. Now that we can take the last matchstick, we are the winner; the display reports: EE0000, and the piezo buzzer sounds an interval tone, which is switched off automatically after a short time.

As soon as any number key is pressed, 0 appears on the display again – a new game can begin.

We can choose a new skill level (entry: 1 or 2 or 3). The pile size to be entered now must be at least 1 or at most 9. On each move, any number of matchsticks may be taken, but always only from one pile. On the next move the removal can be made from a different pile. If the computer is to begin the game (to take the first matchsticks), after entering the pile size we enter: 00. The following 3-digit display then shows the three remaining pile sizes for our entry (as described).

If the computer wins (because we have given it the chance to take the last matchstick(s)), the display likewise shows the message: EE0000. At the same time a continuous tone sounds until the start of the game is reached again by pressing a key. For a new start of the game, always enter the skill level first.

If we make an incorrect entry (e.g. taking more matchsticks from a pile than there are), the computer notices that cheating has occurred – the program stops. We enter the pile number and the number of matchsticks to be removed once more (but without cheating). The program run can then be continued. Or restart the program with HALT – NEXT – 00.

If we enter skill level 2 at the start of the game, we already have to concentrate a little more to win the game. If skill level 3 is entered, we will hardly succeed in beating the computer. Here it becomes very clear that, through intelligent programming, we can bring the computer to the point where it beats us with our own weapons. An interesting prospect – reminiscent, for example, of science-fiction films in which computers and robots programmed by humans defeated intelligent humankind.

This is of interest to the programmer:

In the program table, the first column again marks the addresses at which the jump targets begin. In the second-to-last column (next to Explanations) the addresses can be recognized from which the jumps to the jump targets are made. For this, a few abbreviations are used:

„-1“ = carry out „default move“
 „CGEW?“ = computer wins?
 „SPGEW“ = player wins
 „XOR“ = subroutine „XOR“
 „GEW“ = subroutine win-calculation
 „E1“, „E2“, „E3“ = entry of the skill level (1 or 2 or 3).
 „SCOMP“ = computer's move

For the game „Take 2“, similar to the simple Nim-game version, there is an „optimal winning strategy“ which allows either the player or the computer to win. There is no draw.

As in the Nim game, there are „winning numbers“ or „safe positions“. A typical example of these are the so-called „pairs“, as for instance with the display reading: 033 (33 = pair). In this example the game state would be: first pile 0, second pile 3 and third pile likewise 3. In this situation the opposing player has the following three options:

1. From 033, after removal: 032 or 023.
2. From 033, after removal: 031 or 013
3. From 033, after removal: 030 or 003

From the first and second options a new „pair“ can again be formed. With the third option the player will win, because the opponent leaves only 3 matchsticks remaining on one pile, which remain for the last move (i.e. for the winner).

For the computer there is a relatively simple way to determine whether it has reached a winning position. We recall the chapter „logical operations“, in which it was demonstrated that with the AND and OR commands the individual bits within the binary number system can be changed.

Let us assume that the values 1 – 2 – 3 are still present on the three piles; represented in binary, this looks as follows:

$$\begin{array}{r} 1 = \text{binary } 0001 \\ + 2 = \text{binary } 0010 \\ + 3 = \text{binary } 0011 \\ \hline \text{binary } 0000 \end{array}$$

An AND/OR combination yields

In the preceding example the columns of the individual binary numbers were „added“ (without carry), which yields an overall result of 0000. The overall result 0000 means a „safe position“!

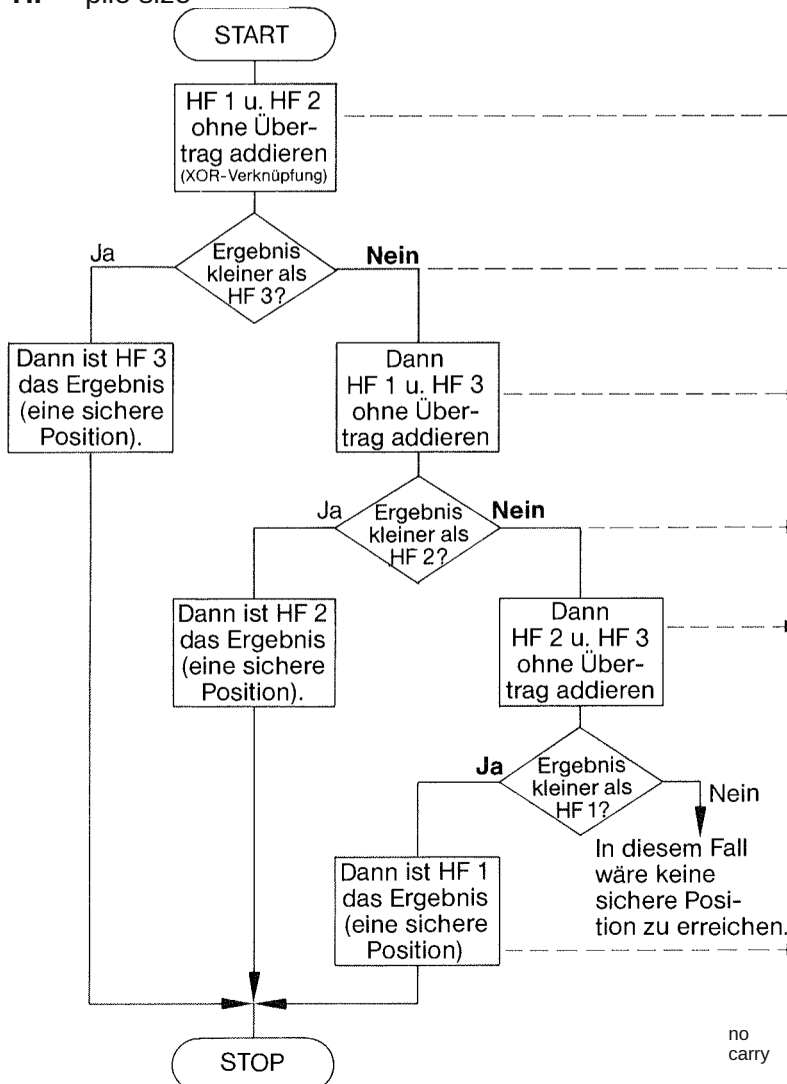
Another example: the following matchsticks still remain on the three piles: 033. Represented in binary, the following situation results:

$$\begin{array}{r} 0 = \text{binary } 0000 \\ + 3 = \text{binary } 0011 \\ + 3 = \text{binary } 0011 \\ \hline \text{binary } 0000 \end{array}$$

An AND/OR combination yields

Simplified program flow chart for calculating a „safe position“

HF = pile size



Calculation examples:
Decimal Binary

$$\begin{array}{r} \text{HF 1} = 2 \text{ } 0010 \\ \text{HF 2} = + 4 \text{ } 0100 \\ \hline \text{Result} = 6 \text{ } 0110 \end{array}$$

No! Since the result 6 is greater than HF 3 (5 matchsticks), continue calculating:

$$\begin{array}{r} \text{HF 1} = 2 \text{ } 0010 \\ \text{HF 3} = + 5 \text{ } 0101 \\ \hline \text{Result} = 7 \text{ } 0111 \end{array}$$

No! Since the result 7 is greater than HF 2 (4 matchsticks), continue calculating:

$$\begin{array}{r} \text{HF 2} = 4 \text{ } 0100 \\ \text{HF 3} = + 5 \text{ } 0101 \\ \hline \text{Result} = 1 \text{ } 0001 \end{array}$$

Yes! The result 1 is greater than HF 1 (2 matchsticks). A safe position results if the computer takes one matchstick from HF 1, because then:

$$\begin{array}{r} \text{HF 1} = 1 \text{ } 0001 \\ \text{HF 2} = 4 \text{ } 0100 \\ \text{HF 3} = 5 \text{ } 0101 \\ \hline \text{Result} = (1)0 \text{ } 0000 \end{array}$$

no carry

With an „unsafe position“ the result would not be 0000. This is to be shown likewise in binary with the pile sizes: 2 – 4 – 5:

$$\begin{array}{r} 2 = \text{binary } 0010 \\ + 4 = \text{binary } 0100 \\ + 5 = \text{binary } 0101 \\ \hline \text{binary } 0011 \end{array}$$

An AND/OR combination yields

From the examples we see that with the binary values, an „addition“ gives the following: $0 + 0 + 0 = 0$. Or: $0 + 1 + 1 = 0$. Or (see the third example with pile sizes 2 – 4 – 5): $1 + 0 + 0 = 1$. Or: $0 + 0 + 1 = 1$. Three times 0 added gives 0. A 0 and the digit 1 twice, added together, likewise gives 0. However, the digit 0 twice plus a 1 added together gives 1.

The binary digit positions were „added without carry“. An addition without carry is a so-called **XOR** (exclusive or) operation, which can be achieved by means of several AND and OR commands.

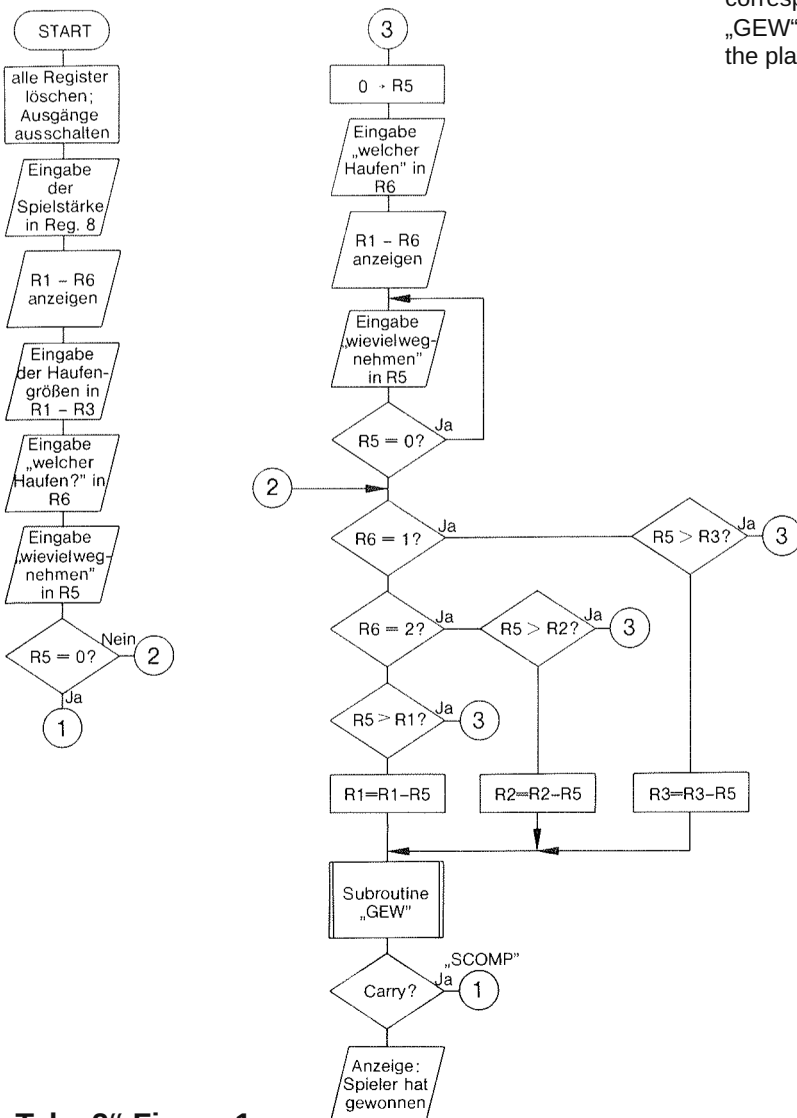
In a simplified program flow we have shown which arithmetic operations the computer must carry out in order to arrive at a „safe position“. To the right of the flow chart, the calculation examples are shown in decimal and binary. We assume that the three pile sizes have the values (number of matchsticks): 2 4 5.

When skill level 1 was entered, it was easily possible to win against the computer. From the preceding presentation we know that the computer always tries to compute a „safe position“ by forming „pairs“. If at skill level 3 we enter, for example, pile sizes for which a pair results (e.g. 0 6 6), it is no longer possible to win against the computer. Let us try it out!

Insight: A „safe position“ for the player against the computer cannot be reached if, already at the start of the game (or during the course of the game), the computer has reached a „safe position“ through „pair formation“. As long as the computer does not reach a „safe position“, it makes a „default move“ by taking only 1 matchstick from one of the three piles, in the hope that the opponent does not pay attention, so as to still reach a „safe position“ during the course of play after all.

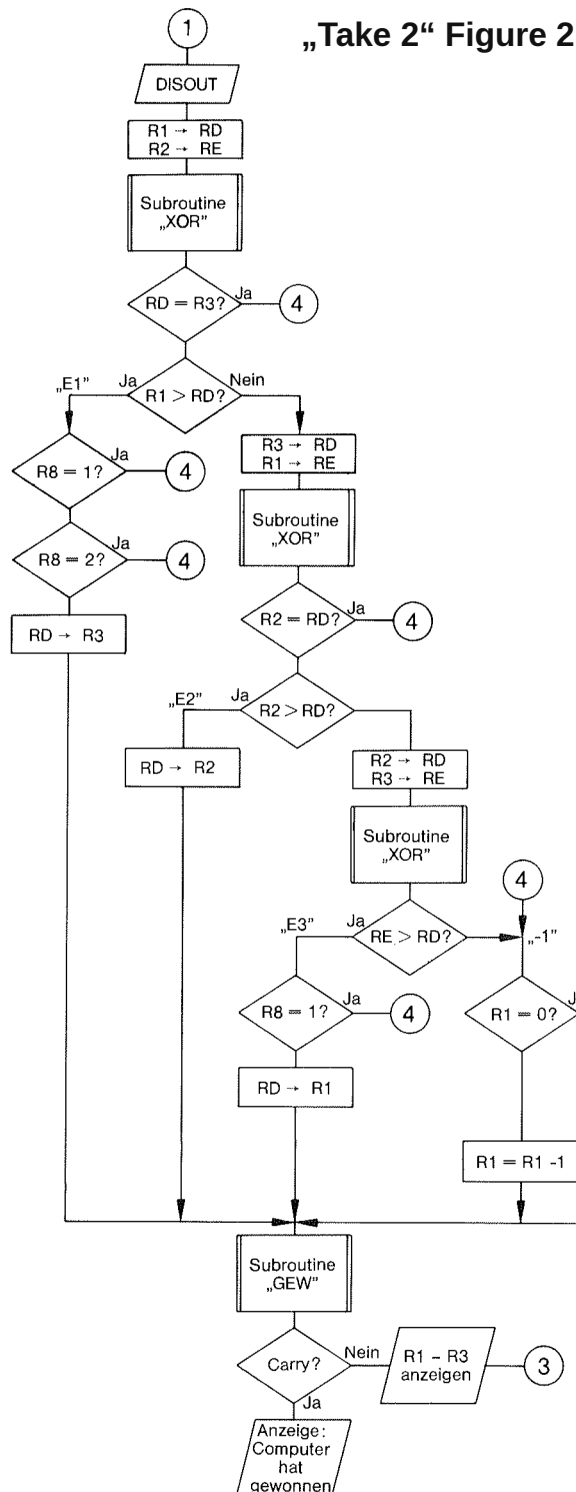
To better understand the entire program flow, let us look at the program flow chart in individual sections.

Figure 1 shows, on the left side, the entry of the skill level and the entry of the pile sizes. On the right side it is demonstrated what happens as soon as a player takes a matchstick from a pile. Above all it is checked whether the removed number of matchsticks is permissible at all (e.g. whether 0 is taken, or whether more matchsticks are taken from a pile than are present). Then the corresponding number of matchsticks is subtracted from the corresponding pile, and a jump to the **subroutine** (subprogram) „GEW“ (= WIN-CHECK) takes place, where it is checked whether the player has won.



„Take 2“ Figure 1

Figure 2 shows the calculation of the computer's moves. The flow chart is constructed similarly to that for the „calculation of a safe position“. Additionally, in this part of the program the queries are built in as to which skill level was chosen (register 8 = 1? or register 8 = 2?). If, for example, skill level 1 or 2 was entered, the computer does not



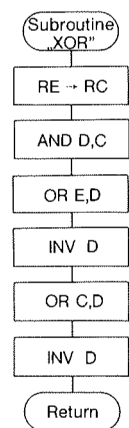
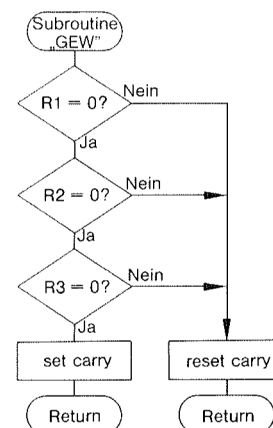
always make the „optimal move“, but rather a „default move“. The default move is calculated in program part „-1“ at the program transition point ⑤, by subtracting the value 1 from a register that does not have the value 0.

In Figure 2 we also recognize the reference to the subroutine „XOR“, in which the binary addition without carry is carried out.

Figure 3 shows the subprograms: subroutine „GEW“ and „XOR“.

Regarding the subprogram „XOR“ the following should also be noted: An „addition without carry“ is a so-called XOR operation. By this it is achieved that the result of a binary digit to be added gives the value 1 only if either the one or the other digit of the binary numbers to be added is 1. If both digits to be added are 1, or both digits to be added are 0, then the result must also be 0. This was already explained at the beginning of the program description. The XOR operation is achieved in the subprogram by the successive logical commands **AND**, **OR** and **INV**.

„Take 2“ Figure 3

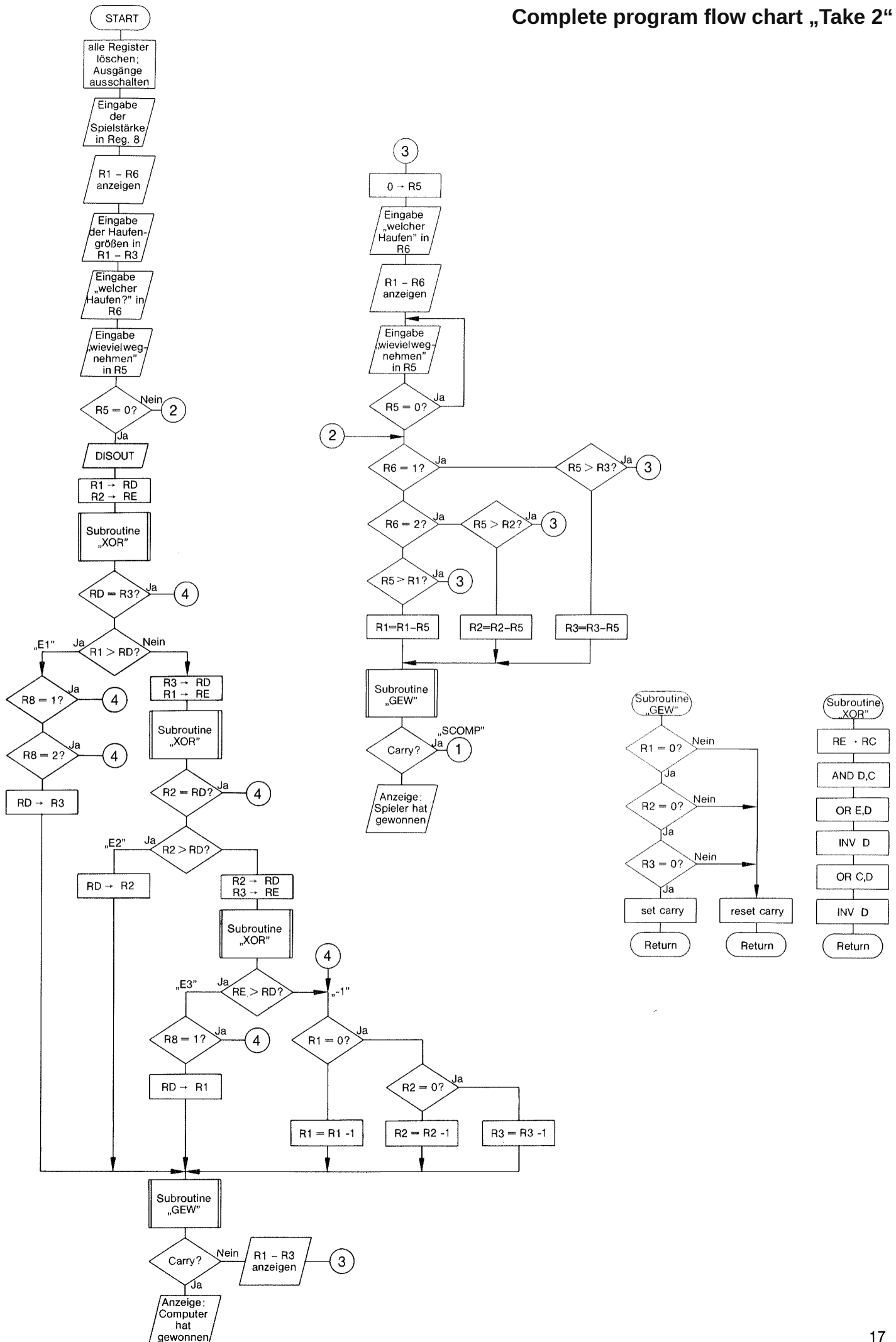


For program specialists, the following table with the register assignment should also not be uninteresting:

Register assignment „Take 2“

HF = pile size

Register assignment „Take 2“																
Register no.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Register content	free	HF 3	HF 2	HF 1	free	how many?	from which pile?	free	skill level	free	free	free	„XOR“	„XOR“	„XOR“	free



Morse Code Decoder

After HALT – NEXT – 00 the large program is entered very carefully. The piezo buzzer still connected from the previous experiment is disconnected during programming and reconnected after the program entry. In addition, a connecting lead is to be made between the TAKT/CLOCK socket and INPUT 4.

During programming we notice that, for example, at addresses 1D and 1E the NOP command is used, and that especially in the second half of the program the same commands have to be entered several times in succession. These repeated, identical commands are necessary. An explanation is found in the following chapter „Morse Trainer“.

Program: Morse Code Decoder / Trainer

Label	Adr.	Code	Mnemonic	Jump to	Explanation
	00	F01	NOP		Input routine
	01	F02	DISOUT		
	02	F08	CLEAR		
	03	FE0	DOT 0		
	04	FF1	KIN 1		
	05	9D1	CMPI D, 1		
	06	D29	BRC 29	„BUCHS“	„BUCHS“ = letters
	07	9A1	CMPI A, 1		
	08	EA2	BRZ A2	„PUNKT“	
	09	9B1	CMPI B, 1		
	0A	EA9	BRZ A9	„KOMMA“	
	0B	9C1	CMPI C, 1		
	0C	EBE	BRZ BE	„AUTO“	„AUTO“ = automatic Morse
	0D	DB0	BRC B0	„SOS“	
	0E	155	MOVI 5, 5		
	0F	951	CMPI 5, 1		Entered number greater or less than 5?
	10	D1F	BRC 1F	„>5“	
	11	901	CMPI 0, 1		
	12	E18	BRZ 18		Generation of the Morse sign for the digits 1 to 5
	13	615	SUB 1, 5		
	14	BD7	CALL D7	„•“	
	15	711	SUBI 1, 1		
	16	E18	BRZ 18		
	17	C14	GOTO 14		
	18	905	CMPI 0, 5		
	19	E00	BRZ 00		
	1A	BE3	CALL E3	„—“	
	1B	715	SUBI 1, 5		
	1C	C18	GOTO 18		Generation of the Morse sign for the digits 6 to 9
	1D	F01	NOP		
	1E	F01	NOP		
„>5“	1F	751	SUBI 5, 1		
	20	615	SUB 1, 5		
	21	BE3	CALL E3	„—“	
	22	711	SUBI 1, 1		
	23	E25	BRZ 25		
	24	C21	GOTO 21		
	25	BD7	CALL D7	„—“	
	26	715	SUBI 1, 5		
	27	E00	BRZ 00		
	28	C25	GOTO 25		

Label	Adr.	Code	Mnemonic	Jump to	Explanation
„BUCHS“	29	FF2	KIN 2		
	2A	9F1	CMPI F, 1		
	2B	E46	BRZ 46	„BU 2“	Key „BU 2“
	2C	902	CMPI 0, 2		Formation of the auxiliary code for the letters
	2D	E60	BRZ 60	„A“	
	2E	912	CMPI 1, 2		
	2F	E64	BRZ 64	„C“	
	30	922	CMPI 2, 2		
	31	E68	BRZ 68	„E“	
	32	932	CMPI 3, 2		
	33	E6C	BRZ 6C	„G“	
	34	942	CMPI 4, 2		
	35	E70	BRZ 70	„I“	
	36	952	CMPI 5, 2		
	37	E74	BRZ 74	„K“	
	38	962	CMPI 6, 2		
	39	E78	BRZ 78	„M“	
	3A	972	CMPI 7, 2		
	3B	E7C	BRZ 7C	„O“	
	3C	982	CMPI 8, 2		
	3D	E80	BRZ 80	„Q“	
	3E	992	CMPI 9, 2		
	3F	E84	BRZ 84	„S“	
	40	9A2	CMPI A, 2		
	41	E88	BRZ 88	„U“	
	42	9B2	CMPI B, 2		
	43	E8C	BRZ 8C	„W“	
„Y“	44	1B2	MOVI B, 2		
	45	C96	GOTO 96	„4“	
„BU 2“	46	902	CMPI 0, 2		
	47	E62	BRZ 62	„B“	
	48	912	CMPI 1, 2		
	49	E66	BRZ 66	„D“	
	4A	922	CMPI 2, 2		
	4B	E6A	BRZ 6A	„F“	
	4C	932	CMPI 3, 2		
	4D	E6E	BRZ 6E	„H“	
	4E	942	CMPI 4, 2		
	4F	E72	BRZ 72	„J“	
	50	952	CMPI 5, 2		
	51	E76	BRZ 76	„L“	
	52	962	CMPI 6, 2		
	53	E7A	BRZ 7A	„N“	
	54	972	CMPI 7, 2		
	55	E7E	BRZ 7E	„P“	
	56	982	CMPI 8, 2		
	57	E82	BRZ 82	„R“	
	58	992	CMPI 9, 2		
	59	E86	BRZ 86	„T“	
	5A	9A2	CMPI A, 2		
	5B	E8A	BRZ 8A	„V“	
	5C	9B2	CMPI B, 2		
	5D	E8E	BRZ 8E	„X“	
„Z“	5E	9C2	MOVI C, 2		
	5F	C96	GOTO 96	„4“	

Label	Adr.	Code	Mnemonic	Jump to	Explanation
"A"	60	142	MOVI 4, 2		„2“ = 2 tone sequence s
	61	C92	GOTO 92	„2“	
"B"	62	182	MOVI 8, 2		
	63	C96	GOTO 96	„4“	
"C"	64	1A2	MOVI A, 2		
	65	C96	GOTO 96	„4“	
"D"	66	182	MOVI 8, 2		
	67	C94	GOTO 94	„3“	
"E"	68	102	MOVI 0, 2		
	69	C90	GOTO 90	„1“	
"F"	6A	122	MOVI 2, 2		
	6B	C96	GOTO 96	„4“	
"G"	6C	1C2	MOVI C, 2		
	6D	C94	GOTO 94	„3“	
"H"	6E	102	MOVI 0, 2		
	6F	C96	GOTO 96	„4“	
"I"	70	102	MOVI 0, 2		
	71	C92	GOTO 92	„2“	
"J"	72	172	MOVI 7, 2		
	73	C96	GOTO 96	„4“	
"K"	74	1A2	MOVI A, 2		
	75	C94	GOTO 94	„3“	
"L"	76	142	MOVI 4, 2		
	77	C96	GOTO 96	„4“	
"M"	78	1C2	MOVI C, 2		
	79	C92	GOTO 92	„2“	
"N"	7A	182	MOVI 8, 2		
	7B	C92	GOTO 92	„2“	
"O"	7C	1E2	MOVI E, 2		
	7D	C94	GOTO 94	„3“	
"P"	7E	162	MOVI 6, 2		
	7F	C96	GOTO 96	„4“	
"Q"	80	1D2	MOVI D, 2		
	81	C96	GOTO 96	„4“	
"R"	82	142	MOVI 4, 2		
	83	C94	GOTO 94	„3“	
"S"	84	102	MOVI 0, 2		
	85	C94	GOTO 94	„3“	
"T"	86	182	MOVI 8, 2		
	87	C90	GOTO 90	„1“	
"U"	88	122	MOVI 2, 2		
	89	C94	GOTO 94	„3“	
"V"	8A	112	MOVI 1, 2		
	8B	C96	GOTO 96	„4“	
"W"	8C	162	MOVI 6, 2		
	8D	C94	GOTO 94	„3“	
"X"	8E	192	MOVI 9, 2		
	8F	C96	GOTO 96	„4“	

Label	Adr.	Code	Mnemonic	Jump to	Explanation
"1"	90	111	MOVI 1, 1		
	91	C97	GOTO 97		
"2"	92	121	MOVI 2, 1		
	93	C97	GOTO 97		
"3"	94	131	MOVI 3, 1		
	95	C97	GOTO 97		
"4"	96	141	MOVI 4, 1		
	97	FA2	SHL 2		
	98	D9D	BRC 9D		
	99	BD7	CALL D7	„“	
	9A	711	SUBI 1, 1		
	9B	E9F	BRZ 9F		
	9C	C97	GOTO 97		Generation of the Morse signs for the letters from the auxiliary code
	9D	BE3	CALL E3	„—“	
	9E	C9A	GOTO 9A		
	9F	906	CMPI 0, 6		
	A0	E00	BRZ 00		
	A1	CC7	GOTO C7	„AUMO“	
"PUNKT"	A2	BD7	CALL D7	„.“	
	A3	BE3	CALL E3	„—“	
	A4	BD7	CALL D7	„.“	
	A5	BE3	CALL E3	„—“	
	A6	BD7	CALL D7	„.“	
	A7	BE3	CALL E3	„—“	
	A8	C9F	GOTO 9F		„AUMO“ = form automatic Morse signs
"KOMMA"	A9	BE3	CALL E3	„—“	
	AA	BE3	CALL E3	„—“	
	AB	BD7	CALL D7	„.“	
	AC	BD7	CALL D7	„.“	
	AD	BE3	CALL E3	„—“	
	AE	BE3	CALL E3	„—“	
	AF	C9F	GOTO 9F		
"SOS"	B0	BD7	CALL D7	„.“	
	B1	BD7	CALL D7	„.“	
	B2	BD7	CALL D7	„.“	
	B3	5FF	ADDI F, F		
	B4	5FF	ADDI F, F		
	B5	BE3	CALL E3	„—“	
	B6	BE3	CALL E3	„—“	
	B7	BE3	CALL E3	„—“	
	B8	5FF	ADDI F, F		
	B9	5FF	ADDI F, F		
	BA	BD7	CALL D7	„.“	
	BB	BD7	CALL D7	„.“	
	BC	BD7	CALL D7	„.“	
	BD	C00	GOTO 00		
"AUTO"	BE	116	MOVI 1, 6		
	BF	F17	DISP 1, 7		

Program continued on page 20

Label	Adr.	Code	Mnemonic	Jump to	Explanation
	C0	FF7	KIN 7		
	C1	F27	DISP 2, 7		
	C2	FF8	KIN 8		
	C3	EC6	BRZ C6		
	C4	F60	DISP 6, 0		
	C5	CC7	GOTO C7		
	C6	F02	DISOUT		
„AUMO“	C7	1FF	MOVI F, F		Generation of the random Morse signs
	C8	71F	SUBI 1, F		
	C9	ECB	BRZ CB		
	CA	CC8	GOTO C8		
	CB	4AD	ADD A, D		
	CC	87D	CMP 7, D		
	CD	DD5	BRC D5		
	CE	0D2	MOV D, 2		
	CF	90E	CMPI 0, E		
	D0	ED3	BRZ D3		
	D1	1E1	MOVI E, 1		
	D2	C2A	GOTO 2A		
	D3	1F1	MOVI F, 1		
	D4	C2A	GOTO 2A		
	D5	57D	ADDI 7, D		
	D6	CCC	GOTO CC		
„*“	D7	1F4	MOVI F, 4		Subroutine for „*“
	D8	FE4	DOT 4		
	D9	5FF	ADDI F, F		
	DA	5FF	ADDI F, F		
	DB	104	MOVI 0, 4		
	DC	FE4	DOT 4		
	DD	FDE	DIN E		
	DE	EE1	BRZ E1		
	DF	53A	ADDI 3, A		Subroutine for „—“
	E0	F07	RET		
	E1	5CA	ADDI C, A		
	E2	F07	RET		
„—“	E3	1F4	MOVI F, 4		
	E4	FE4	DOT 4		
	E5	714	SUBI 1, 4		
	E6	EE8	BRZ E8		
	E7	CE5	GOTO E5		
	E8	FE4	DOT 4		
	E9	FDE	DIN E		
	EA	EEC	BRZ EC		
	EB	F8A	INV A		
	EC	F07	RET		

When key 0 is pressed, a long tone sounds five times. If, for example, key 7 is pressed, the result is twice long and three times short.

If we have not learned the Morse alphabet, we compare with the alphabet table, and with each press of a number key the computer produces the corresponding Morse sign.

Morse Code

a	•—	n	—•	1	•— — —
b	—•••	o	— — —	2	•• — —
c	—• —•	p	• — —•	3	••• — —
d	—••	q	— — —•	3	•••• —
e	•	r	• —•	5	•••••
f	•• —•	s	•••	6	—••••
g	— —•	t	—	7	— —•••
h	••••	u	•• —	8	— — —••
i	••	v	••• —	9	— — — —•
j	• — — —	w	• — —	0	— — — — —
k	—• —	x	—•• —	Punkt	• —• —• —
l	• —••	y	—• — —	Komma	— —•• — —
m	— —	z	— —••	Notruf: SOS	••• — — —•••

Of course we can represent not only numbers but all the letters of the alphabet in Morse code. For this, the available keys must be used multiple times. To make this multiple assignment easier for us, among the accessories of the Microtronic computer we find, among other things, a silver keypad mask with the inscription (bottom left) „Morse Decoder“.

Place the keypad mask on the keypad with the smoke-glass-colored cover hood open, and close the hood again. In the top row (second key from the left) we have **SOS**. When this key is pressed, the computer automatically produces the international distress signal „three times short – three times long – three times short“. The key that was previously used only for the digit 0 is now also responsible for the letters A and B.

If we press this key (in the bottom row on the left), the result is, as before, five times long. If in the top row we press the key labeled **BU 1** and then the key **OAB**, the result is „short-long“, i.e. the Morse sign **A**. If at the top right the key **BU 2** and then again the key **OAB** is pressed, the result is „long – three times short“, i.e. the Morse sign **B**. One quickly gets used to the fact that, for representing the letters, one of the keys **BU 1** or **BU 2** is additionally required in each case.

Even if we cannot do Morse, after a short time we are able to put together complete „radio messages“. Many an amateur radio operator might turn pale with envy.

The special function of the key at the top left with ↑YZ is explained in the following chapter.

The entire experimental setup and programming remains in place for the following chapter.

Program start as usual with HALT – NEXT – 00 – RUN.

Morse Trainer

The preceding program „Morse Code Decoder“ offers us the possibility of learning the Morse alphabet within a short time. Even if a learning effect is not desired, it is interesting to try out this program variation.

We press the key at the top left, ↑YZ. The display shows: 0. If we now press the key **OAB** twice, the computer automatically sends the Morse signs A and B in irregular order.

We abort the program with HALT and start again with NEXT – 00 – RUN. Press the arrow key again, then the key „OAB“ and after that „1CD“. All 6 digits light up on the display and the computer repeats the Morse process of the letters A and B very slowly, which makes it possible to write the signs down. Conclusion: with an entry of 00 (key „OAB“ twice) the signs are sent quickly. With the entry (key „OAB“ and „1CD“) slow Morse signs result.

We abort the program once more and restart with HALT – NEXT – 00 – RUN, press the arrow key again, then „1CD“ and either „1CD“ again (1 = slow) or „OAB“ (O = fast); the computer automatically sends the Morse signs A, B, C and D in irregular order.

If, after HALT – NEXT – 00 – RUN, we again press the arrow key, then „2EF“ (and again 0 for fast or 1 for slow), the computer sends the letters A, B, C, D and F in irregular order.

After HALT – NEXT – 00 – RUN we now press the arrow key twice in succession and additionally 0 for slow or 1 for fast. The computer now sends the entire Morse alphabet with letters, signs and numbers in irregular, random order. A Morse machine for arbitrarily long radio messages, whose content, because of the random sequence, will not always be meaningful. For learning the Morse alphabet, however, an excellent training device.

Should the computer stop the automatic Morse-sign playback by itself after some time, it has ended up in an „endless loop“, i.e. an error has crept in during programming. The entire program would have to be checked carefully once more.

This is of interest to the programmer:

To convert all letters and numbers into Morse signs, various „program tricks“ are necessary.

In the program table we have, at addresses 00 to 0D, the „input routine“. By querying whether, or which, letter key was pressed, a jump is made to the corresponding parts of the program. If no letter key was pressed, the entered value is a number (0 to 9) that is to be converted into Morse signs.

The conversion is relatively easy to solve. For example, all the numbers from 0 to 9 consist of five Morse signs (short or long, i.e. dot or dash). The numbers 1 to 5 each begin with dots and are continued with dashes (e.g. 1 = „• – – – –“, or once short and four times long).

For the numbers 1 to 5, the number of dots (short tones) is stored in register 1 (input register). In register 5 the number of dashes (long tones) is stored. The number of dashes (long tones) is 5 minus the input value. Example: the number 1. Five dashes minus the input value (key value) 1 gives four dashes and one dot. By jumping to the subroutine „•“ a short tone is generated and the value contained in register 1 is decreased by „1“. Then another short tone is generated, whereby again the value „1“ is subtracted from

the content of register 1. This repeats until the content of register 1 has reached the value 0. After that, in the subroutine „–“ a long tone is generated and the value in register 5 is likewise decreased by „1“. This process too repeats until the content of register 5 has reached the value 0. If, for example, the digit 3 was entered to generate the corresponding Morse sign, register 1 is responsible for the three dots (short tones) and register 5 for the two dashes (long tones).

For the digits 6 to 9, at the beginning of the Morse sign there is in each case a dash (long tone) followed by a corresponding number of dots (short tones). In this case the total number of long tones is the entered value minus 5. For the number 7, for example, first two long tones and then three short tones must be generated.

The „decoding“ of letters becomes more difficult. Here too a trick can be applied. The longest Morse sign for letters consists of four „tones“ (i.e. dots or dashes).

The tone sequence short or long (dot or dash) can be achieved by means of a binary number. Example: the Morse sign F = „• • – •“

If we use a binary 0 for dots and a binary 1 for dashes, then for the Morse sign F the binary representation 0010 results, which corresponds to the binary number 2 (see the bit-pattern table).

If the binary arrangement 0010 is shifted to the left by the SHL command, the first shift gives the carry flag 0, and for this a short tone (dot) must be generated. On the second shift to the left the carry flag is again 0. On the third shift a carry flag of 1 results (a long tone must be generated), and on the fourth shift to the left the carry flag becomes 0 again (i.e. a short tone).

The letter A has the Morse sign: „• –“, and for this the binary arrangement 0100 (binary number 4) can be used. However, this binary arrangement may be shifted to the left only twice, because for the representation only one short and one long tone are needed.

So we must convert the letters into a two-digit code. The first digit of the code indicates the number of tones, the second code digit the binary number (which is also called the „bit pattern“). The bit pattern (binary number) is used for the tone sequence (long or short). The letter A thus has the code **24** (2 = 2 tone sequences, 4 = binary number 4 = 0100). The letter F has the code **42** (4 = 4 tone sequences, 2 = binary number 2 = 0010).

The Morse sign F should be self-evident. The binary arrangement 0010 gives the Morse sign: „• • – •“

Less self-evident is the method for the letter A. The binary arrangement 0100 gives a Morse sign „• – • •“. An A, however, has the Morse sign „• –“ (i.e. that we have 2 dots too many).

Here the two-digit code helps, in which the first digit states how many tones (short or long) are to be generated. For the Morse sign A, therefore, only the first two bits (0 and 1) are used.

The conversion of the letters into a corresponding code is handled by the program from address 2B. The binary number (bit pattern), from which the tone sequence long or short results, is stored in register 2. The number of tone sequences (regardless of whether short or long) is stored in register 1. The table „bit pattern“ shows us how the letters of the alphabet are divided into a special code.

Table: Bit Pattern

Buchstabe	Morse- zeichen	Code in Reg. 1	Code in Reg. 2	Morsezeichen Länge (Anzahl Tonfolge)	„Bit-Muster“ (Verwendung der dualen Stellen)			
A	•—	2	4	2	0	1	0	0
B	—•••	4	8	1	1	0	0	0
C	—•—•	4	A	4	1	0	1	0
D	—••	3	8	3	1	0	0	0
E	•	1	0	1	0	0	0	0
F	••—•	4	2	4	0	0	1	0
G	—•—•	3	C	3	1	1	0	0
H	••••	4	0	4	0	0	0	0
I	••	2	0	2	0	0	0	0
J	•—••	4	7	4	0	1	1	1
K	—•—	3	A	3	1	0	1	0
L	•—••	4	4	4	0	1	0	0
M	—•—	2	C	2	1	1	0	0
N	—•	2	8	2	1	0	0	0
O	—•—	3	E	3	1	1	1	0
P	•—••	4	6	4	0	1	1	0
Q	—•—•	4	D	4	1	1	0	1
R	•—•	3	4	3	0	1	0	0
S	•••	3	0	3	0	0	0	0
T	—	1	8	1	1	0	0	0
U	••—	3	2	3	0	0	1	0
V	•••—	4	1	4	0	0	0	1
W	•—•	3	6	3	0	1	1	0
X	—••—	4	9	4	1	0	0	1
Y	—•—•	4	B	4	1	0	1	1
Z	—•••	4	C	4	1	1	0	0

From address 97 onward, by shifting the binary values in register 2 to the left (SHL command), either a long or a short tone is generated. The number of tones is determined by counting down (reducing the values) in register 1.

The actual Morse tone is generated in the subroutines (SUB routine) „•“ and „—“. In addition, in these subroutines a „random number“ arises in register A, depending on whether or not a signal (from TAKT/CLOCK) is present at the INPUT. This controls the irregular repetition of the Morse signs A and B.

The program part „Auto“ (automatic Morse-sign generation) begins at address BE. The keyboard entry normally required is replaced by the generation of random numbers. The random number is generated in register D by adding register A. Since the value in register A is relatively random, a constantly changing random number results in register D, by which the entire Morse alphabet is sent in random order.

In this program the integrated random-number generator is not used. This has the following reason: the generation of the Morse signs requires a certain amount of time, during which the random-number generator counts on by a certain number of steps. This number of steps depends on the Morse-sign length. If one were to use the integrated random-number generator, the constant repetition of a certain letter sequence would result. By using a specially programmed random-number generator, which is controlled by the 1 Hz clock, such a repetition is almost ruled out.

We can try this out by removing the connecting lead TAKT/CLOCK to INPUT 4 and replacing the command ADD A,D (4AD) present at address CB with the RND command (F05). Now the computer will only send a certain letter sequence.

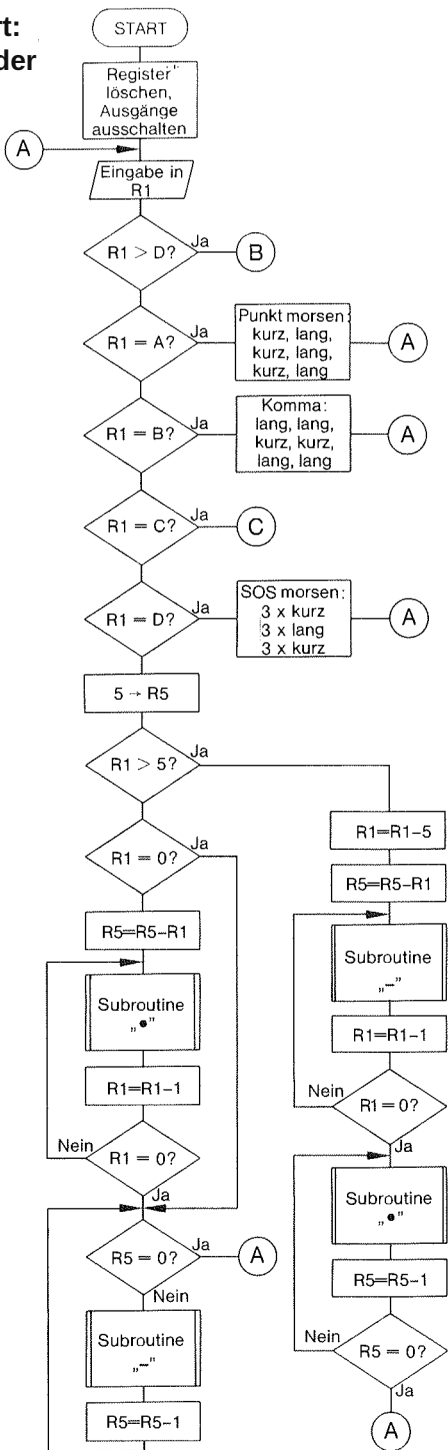
Why was a NOP command inserted at addresses 1D and 1E?

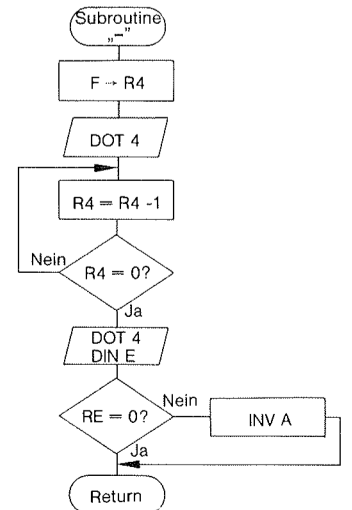
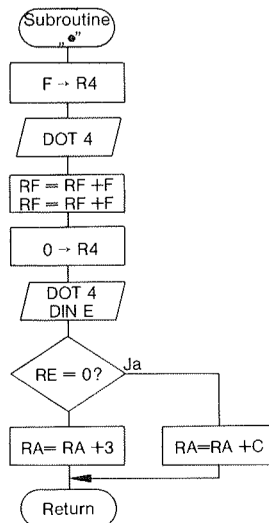
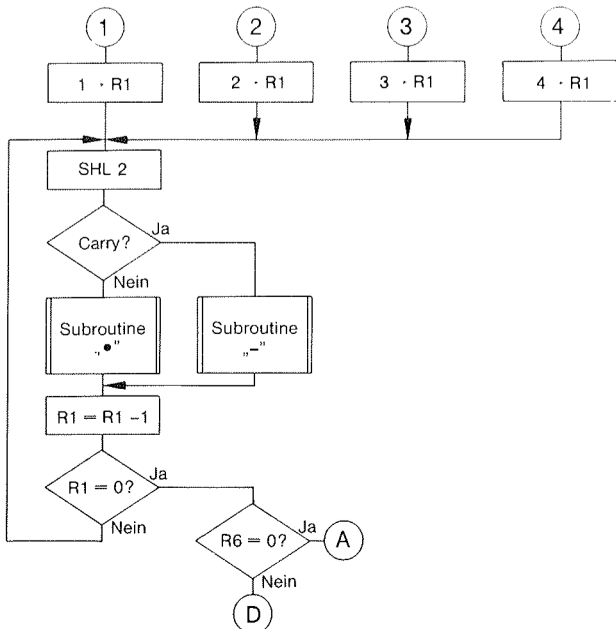
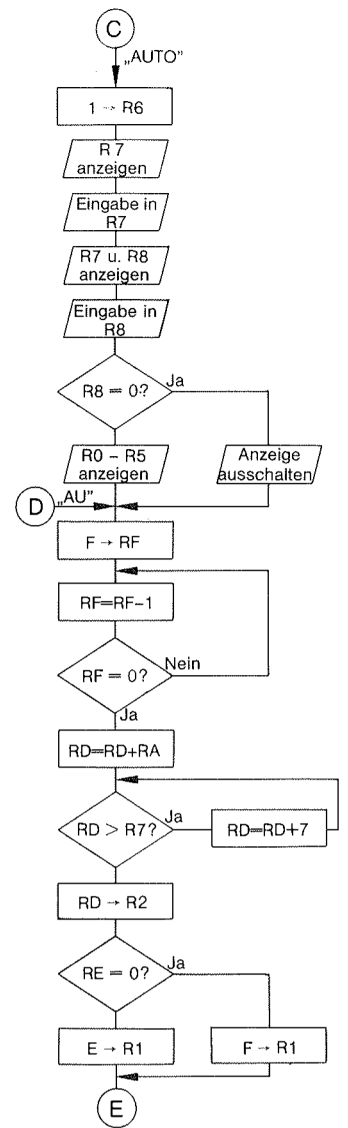
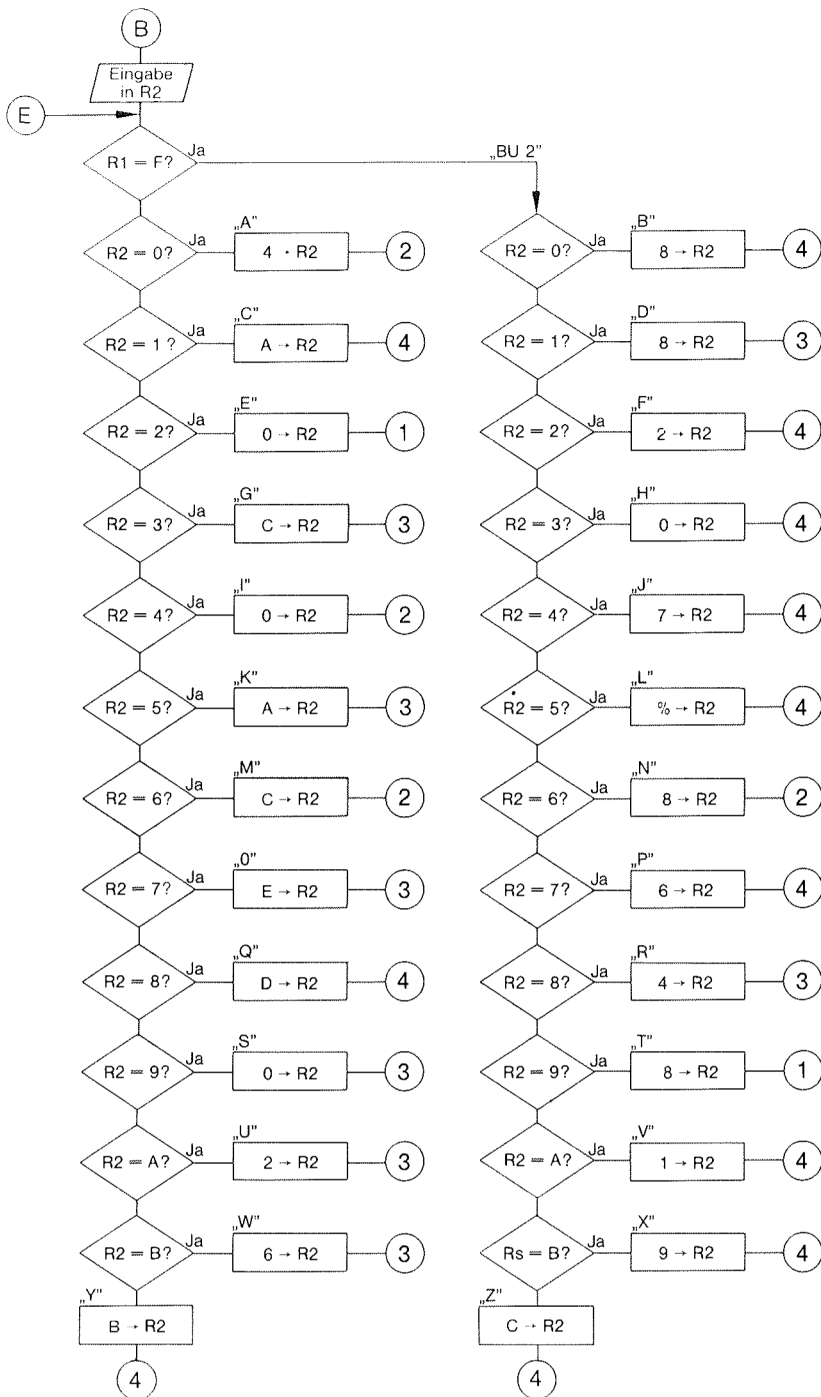
It was merely intended to demonstrate that, with such long programs, it is expedient to enter a few NOP commands now and then, because this gives the possibility of subsequently inserting a command that may have been forgotten during input, without having to re-enter the entire program.

Perhaps we are also surprised by the repetition of identical commands? This arises, for example, at the program locations „•“ and „—“. Here, each time, a jump is made to the corresponding subroutine that is responsible for generating a long or short tone. If, for example, for „SOS“ three short – three long – three short tones result, the subroutine „•“ beginning at address D7 is jumped to three times in succession, then the subroutine „—“ beginning at address E3 three times, and so on.

This program can give us various hints for future program considerations, such as how, by using the binary values and reducing the binary digits, a large number of different signals can be generated. Such signals can be brought to the outputs and used for the most varied possibilities.

**Program flow chart:
Morse Code Decoder
(Morse Trainer)**





For hobby electronics enthusiasts: Resistance Code Calculation

The value of a resistor is indicated by colored rings. For those who do not constantly work with these color markings, it is often difficult to determine the correct resistance values from the color rings.

The following program demonstrates how the computer can calculate the correct resistance values from the entry of various colors.

The piezo buzzer still connected from the last experiment is not needed. Remove the keypad mask that may also still be present, and enter the program according to HALT – NEXT – 00.

Program: Resistance Code Calculation

Label	Adr.	Code	Mnemonic	Jump to	Explanation
	00	F19	DISP 1, 9		Input routine
	01	F08	CLEAR		
	02	FF0	KIN 0		
	03	9B0	CMPI B, 0		
	04	D0D	BRC 0D	START	Key: ↑
	05	519	ADDI 1, 9		
	06	949	CMPI 4, 9		
	07	D00	BRC 00		
	08	034	MOV 3, 4		
	09	023	MOV 2, 3		
	0A	012	MOV 1, 2		
	0B	001	MOV 0, 1		
	0C	C02	GOTO 02		
START	0D	922	CMPI 2, 2		
	0E	D14	BRC 14	KILO	(PRO = percentage tolerance)
	0F	902	CMPI 0, 2		
	10	E1E	BRZ 1E	PRO	
	11	B34	CALL 34	MORI	
	12	712	SUBI 1, 2		Jump to subroutine MORI
	13	C0F	GOTO 0F		
KILO	14	952	CMPI 5, 2		
	15	D1A	BRC 1A	MEGA	
	16	B34	CALL 34	MORI	
	17	1A3	MOVI A, 3		
	18	732	SUBI 3, 2		
	19	C0F	GOTO 0F		
MEGA	1A	B34	CALL 34	MORI	
	1B	1B3	MOVI B, 3		
	1C	762	SUBI 6, 2		
	1D	C0F	GOTO 0F		Jump to subroutine MORI
PRO	1E	911	CMPI 1, 1		
	1F	E2C	BRZ 2C	1 → E	
	20	921	CMPI 2, 1		
	21	E2E	BRZ 2E	2 → E	
	22	9B1	CMPI B, 1		
	23	E30	BRZ 30	5 → E	
	24	9A1	CMPI A, 1		
	25	E32	BRZ 32	1 → F	
	26	12F	MOVI 2, F		
ANZ	27	F63	DISP 6, 3		
	28	FF0	KIN 0		
	29	F2E	DISP 2, E		

Label	Adr.	Code	Mnemonic	Jump to	Explanation
	2A	FF0	KIN 0		ANZ = show display
	2B	C00	GOTO 00		
1 → E	2C	11E	MOVI 1, E		
	2D	C27	GOTO 27	ANZ	
2 → E	2E	12E	MOVI 2, E		
	2F	C27	GOTO 27	ANZ	
5 → E	30	15E	MOVI 5, E		
	31	C27	GOTO 27	ANZ	
1 → E	32	11F	MOVI 1, F		
	33	C27	GOTO 27		
MORI	34	078	MOV 7, 8		„Subroutine“ MORI = move right, i.e. shift the zeros in reg. 3 to 8 to the right, equivalent to shifting the display digits to the left.
	35	067	MOV 6, 7		
	36	056	MOV 5, 6		
	37	045	MOV 4, 5		
	38	034	MOV 3, 4		
	39	993	CMPI 9, 3		
	3A	D3D	BRC 3D		
	3B	103	MOVI 0, 3		
	3C	F07	RET		
	3D	104	MOVI 0, 4		
	3E	F07	RET		

Program start: HALT – NEXT – 00 – RUN. The display shows: 0.

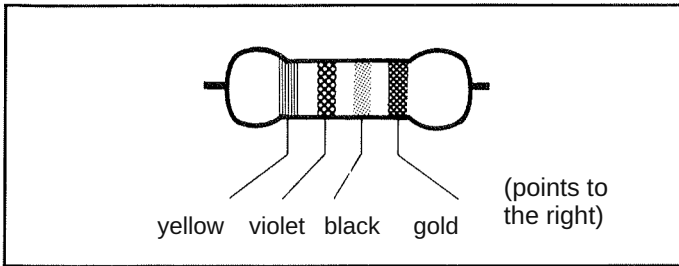
Place the silver-colored keypad mask labeled „Widerstands-Code“ on the keypad with the cover hood open, and close the hood again.

Most resistors have 4 color rings, one of which is silver or gold. For the resistance-value calculation the gold or silver ring must be on the right. The entry therefore begins with the colors (see figure): yellow – violet – black – gold. The display counts the key presses and now shows the digit 4. The color entry is concluded by pressing the key ↑ (top key row).

The display shows 0047. The entered color values correspond to a resistance value of 47 Ω (ohms). Pressing the ↑ key again shows: 05, i.e. that the calculated resistance value has a tolerance of ± 5%.

Pressing the ↑ key again brings 0 back on the display. A new entry can be made.

Color values of a resistor



On the computer board, a whole lot of resistors are arranged whose values can be calculated in this way. It is important that, on principle, 4 colors are always entered. For resistors that have only three color rings, black is entered as the fourth color.

If, for example, the display shows, after entering: yellow – blue – violet – silver: 00460B, then the designation B means that this is a resistor of 460 MΩ (megohms). One megohm is one million ohms.

The entry: red – orange – yellow – gold yields the result 230A. A means kΩ (kilohms), i.e. that the calculated resistance value is 230 kΩ. One kilohm is one thousand ohms.

This is of interest to the programmer:

The function of this program quickly becomes clear to us when we look at the resistor color-code table below. We see that the first and second rings each represent a certain value depending on the color. The third color ring indicates by how many zeros the values are to be supplemented.

Resistor color-code table

Ring	1	2	3	(tolerance)
Gold				5 %
Silver				10 %
Black	0	0	-	
Brown	1	1	0	
Red	2	2	00	
Orange	3	3	000	
Yellow	4	4	0000	
Green	5	5	00000	
Blue	6	6	000000	
Violet	7	7		
Gray	8	8		
White	9	9		

If we were to enter, for example, the color red three times, the result is **2200**. A resistor with the three color rings red thus has 2200 Ω.

If we enter, for example: yellow – violet – red – gold, then we have actually entered the values 4 – 7 – 2 – B. (For this observation, only the attached keypad mask would have to be removed.)

The display thus shows us the first two entered digits directly. For the third entered digit, the number of zeros must be determined, and only for the tolerance specification B must the corresponding values be determined.

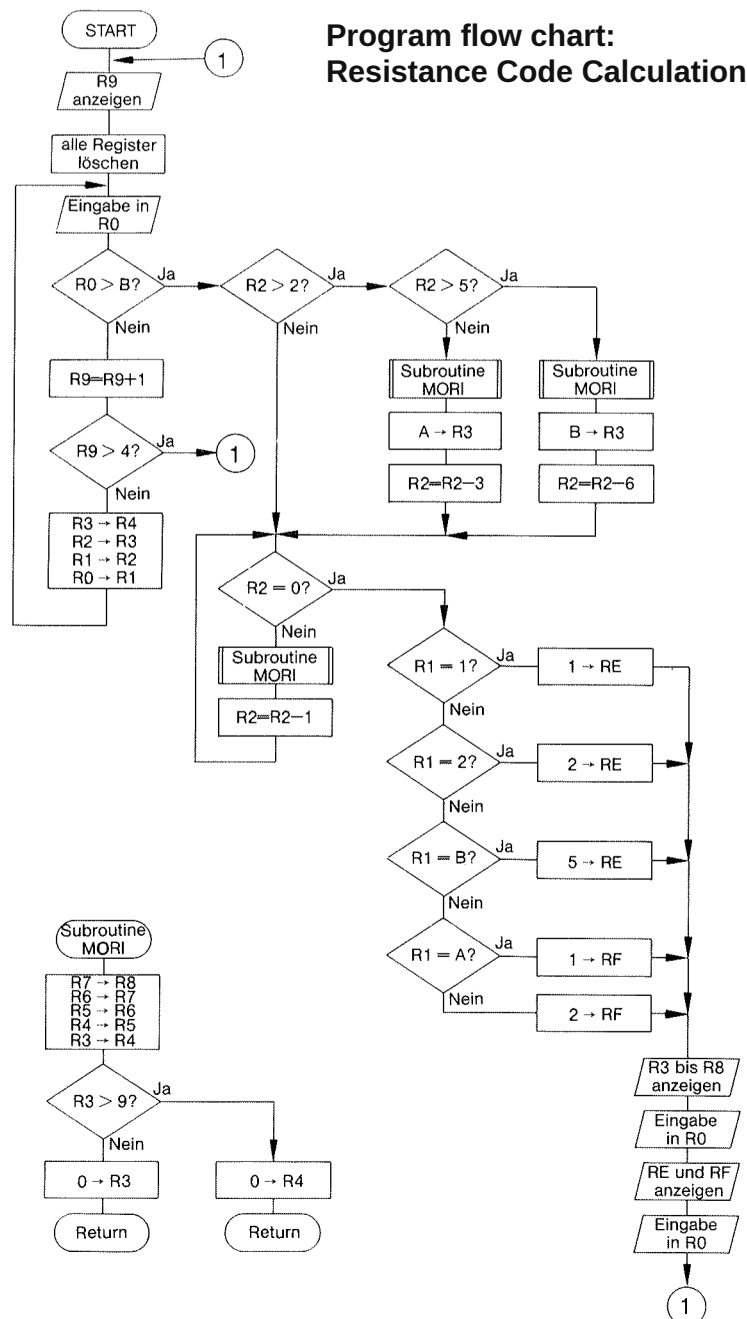
The entry is made into registers 1 to 4. Registers 3 and 4 are displayed directly. Register 2 holds the number of zeros to be shown. The displayed values of registers 3 and 4 must therefore be shifted to the left on the display by the corresponding number of zeros. With the entry yellow – violet – red, the correct result is obtained immediately: 4700 Ω.

So that the 6 display positions available on the display are sufficient, an A (kilo) is used for three zeros, or a B (mega) for six zeros. The displayed result is in registers 3 to 8.

The tolerance is determined by comparison and stored in registers E and F until it is displayed by pressing the arrow key again. See also the program flow chart.

The program example „Resistance Code Calculation“ shows how, through logical reasoning, ideal problem solutions can be achieved with a very simple program.

Program flow chart: Resistance Code Calculation



The „Pocket Calculator“ Program

That our computer can add, subtract, multiply and divide is known to us from the experiments of the instruction book, Part 1.

These individual programs are now combined into a convenient calculator program with which all 4 basic arithmetic operations can be performed.

The rather extensive program should be entered very carefully after HALT – NEXT – 00 and checked.

Program: „Pocket Calculator“

Label	Adr.	Code	Mne- monic	Jump to
	00	F08	CLEAR	
	01	917	CMPI 1,7	
	02	E05	BRZ 05	
	03	F0A	RSC	
	04	C06	GOTO 06	
	05	F09	STC	
	06	F60	DISP 6,0	
	07	FF6	KIN 6	
	08	F02	DISOUT	
	09	996	CMPI 9,6	
	0A	D1D	BRC 10	
	0B	947	CMPI 4,7	
	0C	D15	BRC 15	
	0D	187	MOVI 8,7	
	0E	060	MOV 6,0	
	0F	101	MOVI 0,1	
	10	102	MOVI 0,2	
	11	103	MOVI 0,3	
	12	104	MOVI 0,4	
	13	105	MOVI 0,5	
	14	C03	GOTO 03	
	15	045	MOV 4,5	
	16	034	MOV 3,4	
	17	023	MOV 2,3	
	18	012	MOV 1,2	
	19	001	MOV 0,1	
	1A	060	MOV 6,0	
	1B	387	ANDI 8,7	
	1C	C03	GOTO 03	
	1D	9A6	CMPI A,6	
	1E	E33	BRZ 33 „C/CE“	
	1F	317	ANDI 1,7	
	20	90E	CMPI 0,E	
	21	E2A	BRZ 2A	
	22	9CE	CMPI C,E	
	23	E66	BRZ 66 +	
	24	9DE	CMPI D,E	
	25	E99	BRZ 99 –	
	26	9EE	CMPI E,E	
	27	E39	BRZ 39 X	
	28	9FE	CMPI F,E	
	29	E3F	BRZ 3F :	
	2A	06E	MOV 6,E	
	2B	07F	MOV 7,F	

	2C	008	MOV 0,8	
	2D	019	MOV 1,9	
	2E	02A	MOV 2,A	
	2F	03B	MOV 3,B	
	30	04C	MOV 4,C	
	31	05D	MOV 5,D	
	32	C01	GOTO 01	
„C/CE“	33	347	ANDI 4,7	
	34	947	CMPI 4,7	
	35	E00	BRZ 00	
	36	147	MOVI 4,7	
	37	106	MOVI 0,6	
	38	C0E	GOTO 0E	
X	39	06E	MOV 6,E	
	3A	B49	CALL 49 „+/-“	
	3B	B53	CALL 53 „TAUSCH“	
	3C	F0B	MULT	
	3D	D59	BRC 59 EE	
	3E	C61	GOTO 61 ==	
:	3F	06E	MOV 6,E	
	40	B49	CALL 49 „+/-“	
	41	104	MOVI 0,4	
	42	105	MOVI 0,5	
	43	10C	MOVI 0,C	
	44	10D	MOVI 0,D	
	45	B53	CALL 53 „TAUSCH“	
	46	F0C	OIV	
	47	D59	BRC 59 EE	
	48	C61	GOTO 61 EE	
„+/-“	49	907	CMPI 0,7	
	4A	E4F	BRZ 4F	
	4B	90F	CMPI 0,F	
	4C	D51	BRC 51	
	4D	117	MOVI 1,7	
	4E	C52	GOTO 52	
	4F	90F	CMPI 0,F	
	50	D4D	BRC 4D	
	51	107	MOVI 0,7	
	52	F07	RET	
TAUSCH	53	F0F	EXRA	
	54	06E	MOV 6,E	
	55	F0D	EXRL	
	56	F0F	EXRA	
	57	06E	MOV 6,E	
	58	F07	RET	

EE	59	1E0	MOVI E,0	
	5A	1E1	MOVI E,1	
“	5B	1E2	MOVI E,2	
	5C	1E3	MOVI E,3	
	5D	1E4	MOVI E,4	
	5E	1E5	MOVI E,5	
	5F	107	MOVI 0,7	
	60	C64	GOTO 64	
„==“	61	9BE	CMPI B,E	
	62	E64	BRZ 64	
	63	C2A	GOTO 2A	
	64	10E	MOVI 0,E	
	65	C01	GOTO 01	
+	66	06E	MOV 6,E	
	67	907	CMPI 0,7	
	68	E6D	BRZ 6D	
	69	90F	CMPI 0,F	
	6A	EA2	BRZ A2 „SU“	
	6B	117	MOVI 1,7	
	6C	C71	GOTO 71	
	6D	90F	CMPI 0,F	
	6E	E70	BRZ 70	
	6F	CA2	GOTO A2 „SU“	
	70	107	MOVI 0,7	
„A0“	71	480	ADD 8,0 1. Reg.	
	72	D76	BRC 76	
	73	990	CMPI 9,0	
	74	D76	BRC 76	
	75	C78	GOTO 78	
	76	560	ADDI 6,0	
	77	511	ADDI 1,1	
	78	491	ADD 9,1 2. Reg.	
	79	D7D	BRC 7D	
	7A	991	CMPI 9,1	
	7B	D7D	BRC 7D	
	7C	C7F	GOTO 7F	
	7D	561	ADDI 6,1	
	7E	512	ADDI 1,2	
	7F	4A2	ADD A,2 3. Reg.	
	80	D84	BRC 84	
	81	992	CMPI 9,2	
	82	D84	BRC 84	
	83	C86	GOTO 86	
	84	562	ADDI 6,2	
	85	513	ADDI 1,3	
	86	4B3	ADD B,3 4. Reg.	
	87	D8B	BRC 8B	
	88	993	CMPI 9,3	
	89	D8B	BRC 8B	
	8A	C8D	GOTO 8D	
	8B	563	ADDI 6,3	
	8C	514	ADDI 1,4	
	8D	4C4	ADD C,4 5. Reg.	
	8E	D92	BRC 92	
	8F	994	CMPI 9,4	
	90	D92	BRC 92	

	91	C94	GOTO 94	
	92	564	ADDI 6,4	
	93	515	ADDI 1,5	
	94	4D5	ADD D,5 6. Reg.	
	95	D59	BRC 59 „EE“	
	96	995	CMPI 9,5	
	97	D59	BRC 59 „EE“	
	98	C61	GOTO 61 „==“	
–	99	06E	MOV 6,E	
	9A	907	CMPI 0,7	
	9B	E9F	BRZ 9,F	
	9C	90F	CMPI 0,F	
	9D	EA2	BRZ A2	
	9E	C6B	GOTO 6B	
	9F	90F	CMPI 0,F	
	A0	EA4	BRZ A4	
	A1	C6B	GOTO 6B	
„SU“	A2	F0F	EXRA	
	A3	06E	MOV 6,E	
	A4	608	SUB 0,8 Reg. 1	
	A5	DA7	BRC A7	
	A6	CAA	GOTO AA	
	A7	768	SUBI 6,8	
	A8	719	SUBI 1,9	
	A9	DAD	BRC AD	
	AA	619	SUB 1,9 Reg. 2	
	AB	DAE	BRC AE	
	AC	CB1	GOTO B1	
	AD	619	SUB 1,9	
	AE	769	SUBI 6,9	
	AF	71A	SUBI 1,A	
	B0	DB4	BRC B4	
	B1	62A	SUB 2,A Reg. 3	
	B2	DB5	BRC B5	
	B3	CB8	GOTO B8	
	B4	62A	SUB 2,A	
	B5	76A	SUBI 6,A	
	B6	71B	SUBI 1,B	
	B7	DBB	BRC BB	
	B8	63B	SUB 3,B 4. Reg.	
—	B9	DBC	BRC BC	
	BA	CBF	GOTO BF	
	BB	63B	SUB 3,B	
	BC	76B	SUBI 6,B	
	BD	71C	SUBI 1,C	
	BE	DC2	BRC C2	
	BF	64C	SUB 4,C 5. Reg.	
	C0	DC3	BRC C3	
	C1	CC6	GOTO C6	
	C2	64C	SUB 4,C	
	C3	76C	SUBI 6,C	
	C4	71D	SUBI 1,D	
	C5	DCC	BRC cc	
	C6	65D	SUB 5,D 6. Reg.	
	C7	DCD	BRC CD	
	C8	F0F	EXRA	

	C9	107	MOVI 0,7	
	CA	06E	MOV 6,E	
	CB	C61	GOTO 61	„=“
	CC	65D	SUB 5,0	
	CD	76D	SUBI 6,D	
	CE	117	MOVI 1,7	
	CF	F88	INV 8	
	D0	768	SUBI 6,8	
	D1	F89	INV 9	
	D2	769	SUBI 6,9	
	D3	F8A	INV A	
	D4	76A	SUBI 6,A	
	D5	F8B	INV B	

	D6	76B	SUBI 6,B	
	D7	F8C	INV C	
	D8	76C	SUBI 6,C	
	D9	F8D	INV D	
	DA	76D	SUBI 6,D	
	DB	110	MOVI 1,0	
	DC	101	MOVI 0,1	
	DD	102	MOVI 0,2	
	DE	103	MOVI 0,3	
	DF	104	MOVI 0,4	
	EO	105	MOVI 0,5	
	E1	C71	GOTO 71	„A0“

If a result is larger than the 6-digit display permits, EEEEE appears on the display.

For a division, a maximum of 4 digits may be entered. Division results are therefore always displayed with 4 digits. Since the display of decimal places is not provided, division results are generally rounded down: example: $17 \div 2 = 8$ (instead of the correct result 8.5).

This is of interest to the programmer:

For better understanding, the register assignment will first be presented in the following table:

Register Assignment:

Reg. No.	Register functions
0 1 2 3 4 5	Display register for number entry, for intermediate results and final results
6	Input register for numbers that are shifted onward into reg. 0 to 5
* 7	Plus-/minus index register (storage of whether the number values present in reg. 0 to 5 are positive (+) or negative (-)).
8 9 A B C D	Register for the intermediate storage of the „to be further processed“ number values (value storage until the arithmetic operation with the numbers in reg. 0 to 5 is performed).
E	Memory register for the arithmetic operation to be performed (+ - × :)
* F	Plus-/minus index register (storage of whether the number values present in reg. 8 to D are positive (+) or negative (-)).

The plus/minus index registers (registers 7 and F) have the content 1 when the number values are negative, or 0 when the number values are positive. Index register 7 additionally receives the value 4 as soon as the C/CE key has been pressed once, or it receives the value 8 as soon as a new number is entered.

Program start with HALT – NEXT – 00 – RUN. The display shows: 000000.

With the cover hood open, place the keypad mask labeled „Taschen-Rechner/Calculator“ on the keyboard and close the cover hood again.

The function keys (on the right-hand side) are not used for the arithmetic operations. This concerns above all the C/CE key, which is now present and to be used as a „clear key“ (in the second row from the top).

As with a pocket calculator, the values to be calculated are entered. The entry of the number values must not be made too quickly. The arithmetic operations are triggered by pressing the keys +, -, ÷, × and the result is determined and displayed by the = key. With the C/CE key, a single press clears the most recently entered number; with a double press, all entered numbers are cleared.

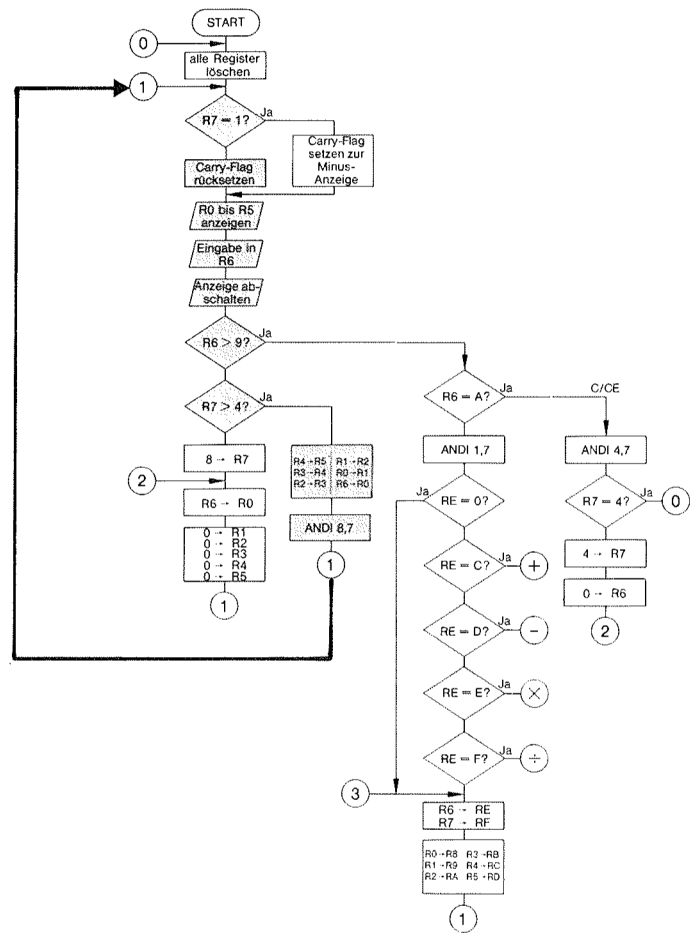
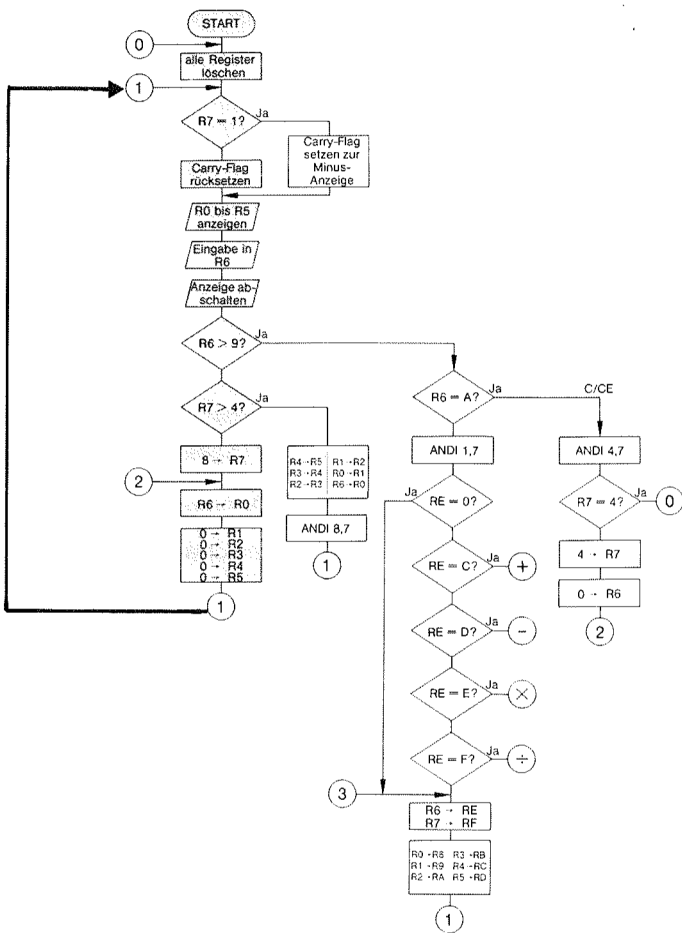
If a negative result arises, the upper carry-flag LED lights up at the same time as the result display.

We first carry out the following calculation: $25 \times 3 + 80 - 300$. The entries, the keys to be pressed, and the display result as follows:

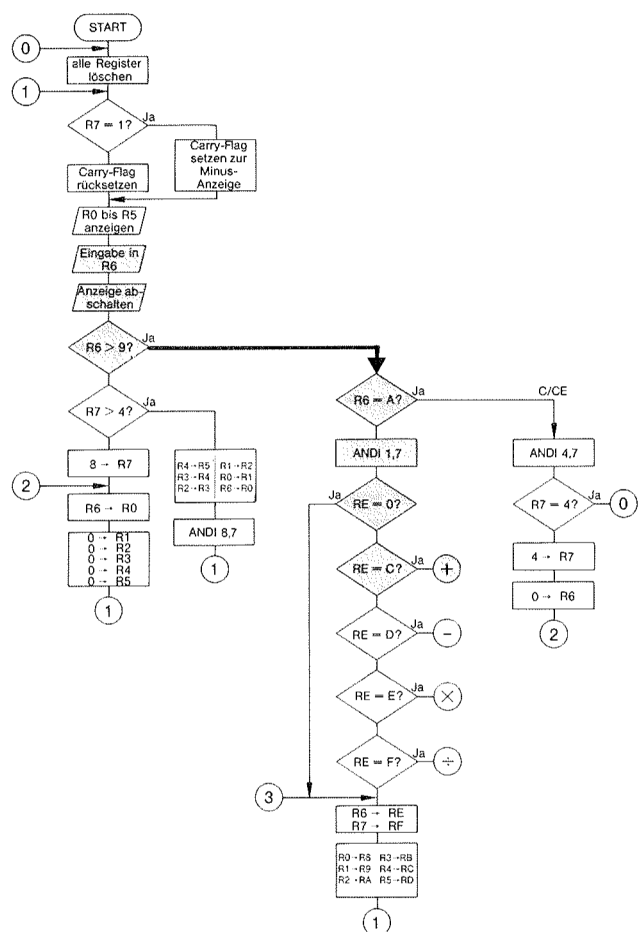
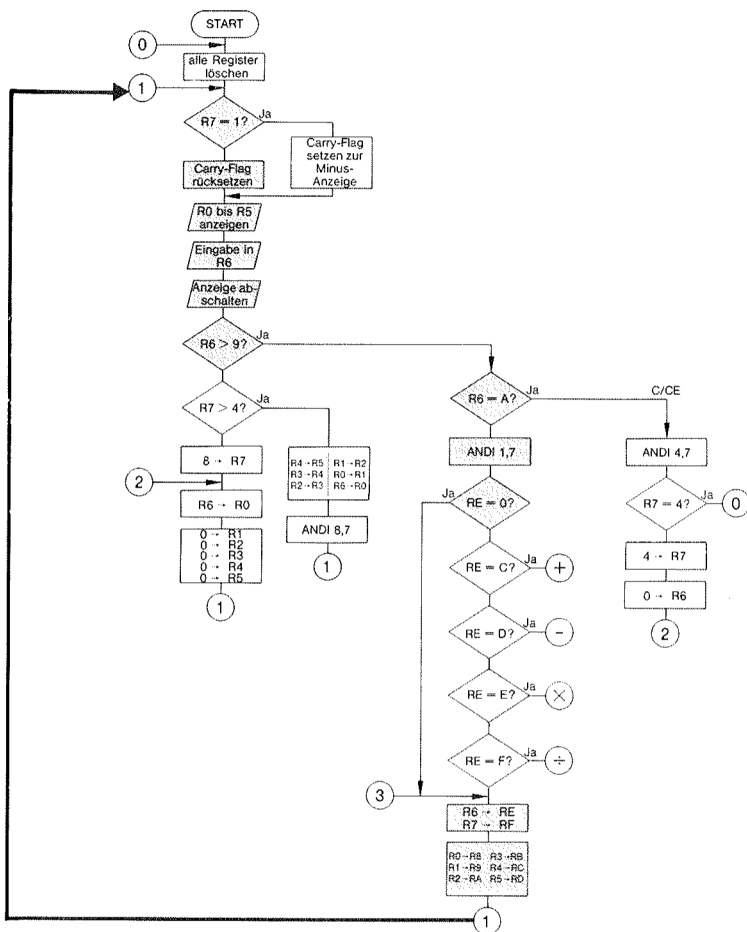
Entry	Display	
25	0 0 0 0 2 5	
	0 0 0 0 2 5	
3	0 0 0 0 0 3	
	0 0 0 0 7 5	1st intermediate result
80	0 0 0 0 8 0	
	0 0 0 1 5 5	2nd intermediate result
300	0 0 0 3 0 0	
	0 0 0 1 4 5	Final result

Glowing carry flag = negative result,
i.e. final result = -145

To demonstrate the program function clearly, the addition of two numbers is shown in a part of the program flow chart. Example: the numbers 24 and 32 are to be added:



1. Entry of the first number, e.g. the number value 2. Since the entered number is not greater than 9, and index register 7 has the value 0, the number value (2) entered in register 6 is stored in register 0. At the same time, registers 1 to 5 are set to zero. Index register 7 is now loaded with the value 8.
2. The second number entry (e.g. 4) is entered. Since index register 7 has meanwhile received the value 8, the query takes place: „Is the content of register 7 greater than 4?“ – „Yes“ – thus a jump occurs and the entered number values are shifted one place to the left on the display.



3. Before the next numbers to be added are entered, the $\oplus$ key is pressed. This corresponds to the hexadecimal value C, so the entry now made in register 6 is greater than 9. Through the comparison command (is the content of register 6 greater than 9?) a jump occurs. Since the content of register 6 does not have the value A (see comparison command), the value 8 in index register no. 7 is cleared by the command ANDI 1,7. Register E still has the value 0 at the moment and a jump occurs. As a result, the content of register 6 is stored in register E. Register E now receives, through the value C, the information that an addition is intended (see also the Register-Assignment table). The contents of registers 0 and 1 (or, if multi-digit numbers are entered, up to register 5) are buffered in registers 8 and 9 (or, for multi-digit numbers, in registers 8 to D).

Now the numbers to be added (in our example 32) are entered, whereby the processes described under 1. and 2. repeat.

4. The addition is to be carried out, the $\ominus$ key is pressed. It has the value B, so a jump occurs as described under 3. After the value C (through the $\oplus$ key entry) has been stored in register E, a jump now results to the program part „AD“ (Addition), which is marked in the overall program flow chart by the circle $\oplus$.

At the beginning of the program part „Addition“ a comparison takes place as to whether one of the two numbers to be added has a negative value – if „yes“, a jump occurs to the program part „SU“ (Subtraction), marked in the program flow chart by the circle $\ominus$. Since this is not the case in our example, the decimal addition of the individual registers takes place (reg. 8 + reg. 0, reg. 9 + reg. 1 etc.). The result is displayed. If the result were larger than the 6-digit display capability on the display, a jump to the program part „EE“ would occur. The result display would then be EEEEE.

The subtraction part likewise begins with a comparison. If positive numbers are to be subtracted (example: $8 - 5 = 3$), the individual registers are subtracted and, if necessary, „decimally corrected“, as described in Part 1 of the instruction book. If a subtraction is carried out with negative numbers (example: $-3 - 5 = -8$), then the overall result is indeed negative, but the two negative number values must be added. With a resulting negative overall result, a carry arises, whereby the individual digits are decimally inverted and then the value 1 is added in the addition program, in order to obtain a correct result (see instruction book Part 1, pages 51 to 54).

For multiplication $\otimes$ and division $\oslash$, the subprogram „+/-“ first determines whether the result is positive or negative, so that the corresponding value can be stored in index register no. 7. For programmers, the following table shows how positive or negative results are calculated:

+	1	x	+	1	=	+	1
+	1	x	-	1	=	-	1
-	1	x	+	1	=	-	1
-	1	x	-	1	=	+	1

For a positive result, the value 0 is stored in index register 7, for a negative result the value 1. Then the values present in registers 8 to D are brought into registers 0 to 5 by the subprogram „Swap“. Now the multiplication or division is carried out.

All 4 calculations are concluded by the program part $\ominus$ or „EE“, and a return to the display routine occurs.

Most program parts correspond to the explanations in Part 1 of the instruction book. New, however, is that we can also continue calculating with negative numbers. For this, the computer must know whether and which numbers are negative.

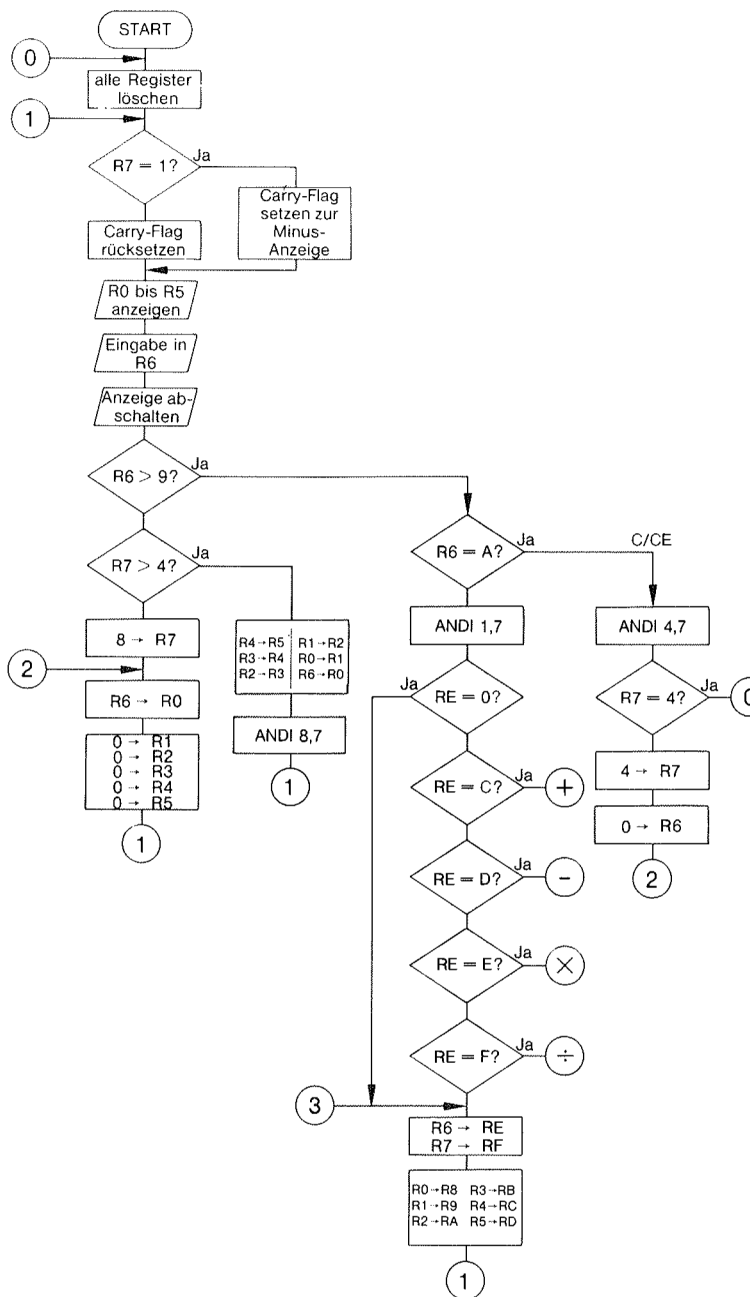
For this, registers 7 and F are available.

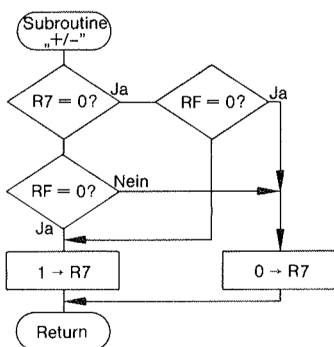
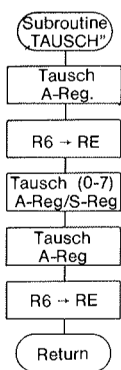
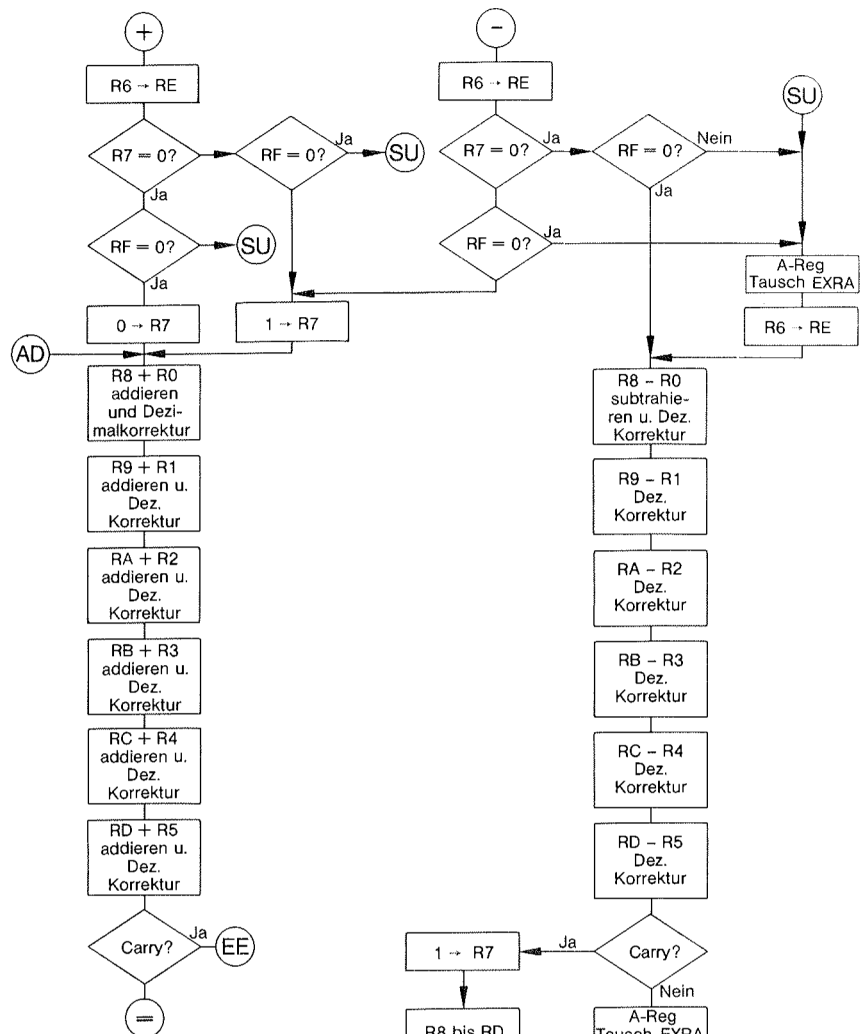
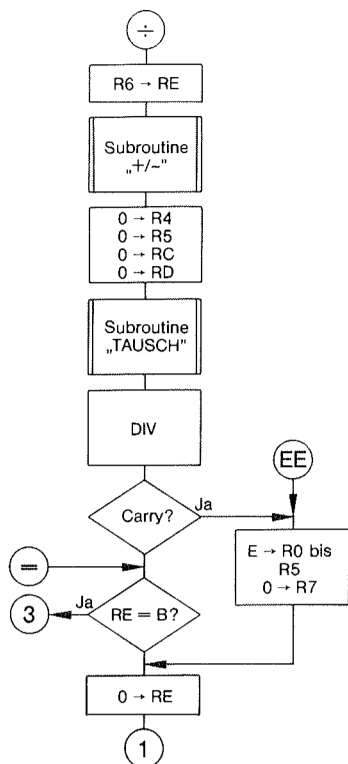
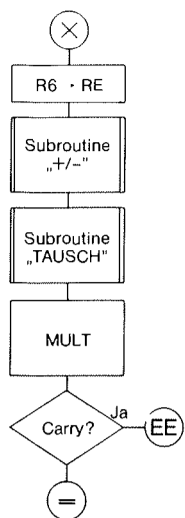
Example: a calculation yields the negative result -7 , and the value $+5$ is to be added to it. This cannot be achieved by the usual addition, since $7 + 5 = 12$ would result. Through the query of registers 7 and F, a jump occurs to the subtraction program „ $\ominus$ “ and the computer then calculates: $5 - 7 = -2$.

A similar case arises when, for example, the last result was -7 and -5 is to be subtracted from it. By comparison, the computer finds out that both numbers are negative, i.e. that the overall result must also be a negative number. It therefore adds both numbers, which yields the result $5 + 7 = 12$. But since it was stored that both numbers are negative, the carry flag is set, whereby the negative result -12 is displayed.

If one wishes to program such a complex program oneself, like this calculator program for example, it is necessary to think through and mentally follow every single program step.

Overall Program Flow Chart: „Pocket Calculator“





A program especially for mathematicians: Sine Calculation

For mathematical calculations, trigonometric functions (angle functions) such as sine, cosine or tangent are often needed.

There are technical/scientific pocket calculators that solve such calculations at the press of a button. These pocket calculators too usually contain a microprocessor that can only add and perhaps just barely multiply. Therefore the sine calculation must be derived from additions, multiplications, etc.

For those interested in mathematics, it is certainly not uninteresting to learn how, for example, a pocket calculator performs a simple sine calculation.

After HALT – NEXT – 00 we enter the following program:

Program: Sine Calculation

Label	Adr.	Code	Mne- monic	Jump to
	00	F08	CLEAR	
	01	F20	DISP 2,0	
	02	FF1	KIN 1	
	03	FF0	KIN 0	
	04	F02	DISOUT	
	05	F0D	EXRL	
	06	F08	CLEAR	
	07	140	MOVI 4,0	
	08	171	MOVI 7,1	
	09	112	MOVI 1,2	
	0A	F0B	MULT	
	0B	00C	MOV 0,C	
	0C	01D	MOV 1,0	
	0D	02E	MOV 2,E	
	0E	03F	MOV 3,F	
	0F	B59	CALL 59	MOVI 0
	10	0F1	MOV F,1	
	11	0E0	MOV E,0	
	12	F0D	EXRL	
	13	0F1	MOV F,1	
	14	0E0	MOV E,0	
	15	F0B	MULT	
	16	F0D	EXRL	
	17	160	MOVI 6,0	
	18	F0C	DIV	
	19	F0D	EXRL	
	1A	B59	CALL 59	MOVI 0
	1B	0F1	MOV F,1	
	1C	0E0	MOV E,0	
	1D	F0B	MULT	
	1E	0C8	MOV C,8	
	1F	0D9	MOV D,9	
	20	0EA	MOV E,A	
	21	0FB	MOV F,B	
	22	628	SUB 2,8	
	23	D25	BRC 25	
	24	C28	GOTO 28	
	25	768	SUBI 6,8	
	26	719	SUBI 1,9	
	27	D2B	BRC 2B	
	28	639	SUB 3,9	

	29	D2B	BRC 2B	
	2A	C2E	GOTO 2E	
	2B	769	SUBI 6,9	
	2C	71A	SUBI 1,A	
	2D	D31	BRC 31	
	2E	64A	SUB 4,A	
	2F	D32	BRC 32	
	30	C34	GOTO 34	
	31	64A	SUB 4,A	
	32	76A	SUBI 6,A	
	33	71B	SUBI 1,B	
	34	030	MOV 3,0	
	35	041	MOV 4,1	
	36	B5B	CALL 5B	MOVI 0
	37	F0D	EXRL	
	38	0E0	MOV E,0	
	39	0F1	MOV F,1	
	3A	F0B	MULT	
	3B	F0D	EXRL	
	3C	121	MOVI 2,1	
	3D	F0C	DIV	
	3E	F0D	EXRL	
	3F	B59	CALL 59	MOVI 0
	40	0E0	MOV E,0	
	41	0F1	MOV F,1	
	42	F0B	MULT	
	43	438	ADD 3,8	
	44	D48	BRC 48	
	45	998	CMPI 9,8	
	46	D48	BRC 48	
	47	C4A	GOTO 4A	
	48	568	ADDI 6,8	
	49	519	ADDI 1,9	
	4A	449	ADD 4,9	
	4B	D4F	BRC 4F	
	4C	999	CMPI 9,9	
	4D	D4F	BRC 4F	
	4E	C56	GOTO 56	
	4F	569	ADDI 6,9	
	50	51A	ADDI 1,A	
	51	99A	CMPI 9,A	

	52	D54	BRC 54	
	53	C56	GOTO 56	
	54	56A	ADDI 6,A	
	55	51D	ADDI 1,D	
	56	F48	DISP 4,8	
	57	FF0	KIN 0	
	58	C00	GOTO 00	

MOVI 0	59	100	MOVI 0,0	
	5A	101	MOVI 0,1	
	5B	102	MOVI 0,2	
	5C	103	MOVI 0,3	
	5D	104	MOVI 0,4	
	5E	105	MOVI 0,5	
	5F	F07	RET	

Program start with HALT – NEXT – 00 – RUN. The display shows: 00.

The sine of an angle of 20° is to be calculated. For this, 20 is entered. After a brief calculation, the display shows: 3415. Since our calculator works without a decimal place, this result means: 0.3415. It is expedient to enter the examples for the program test, so that the correct result can be checked. If the computer does not produce the aforementioned result, the program entry must be checked and, if necessary, corrected.

Before entering a new calculation, press any number key; the display shows 00 again. A new calculation can be carried out. Since a 2-digit entry is required, for a sine calculation of 5° the two-digit entry 05 must be made. In this case the display shows: 0870, i.e. 0.0870.

We should also note that the entered angle must not be greater than 45°, because calculations going beyond 45° are no longer carried out correctly.

This is of interest to the programmer:

Before we deal with the actual program sequence, we must know how a sine function can be calculated through additions, multiplications, etc.

The angle functions can be calculated with the help of so-called „series“. The series for calculating a sine has the following formula:

[Translator's note: in the German original the decimal separator is a comma, e.g. the figures show 0,3415 for 0.3415.]

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \dots\dots\dots$$

A sine to be calculated becomes the more accurate the more terms the series to be calculated has. So that the program does not become too elaborate, the series was terminated after the third term (boxed formula).

For an example, let us calculate the sine of the angle of 20°. However, we cannot insert „20“ directly for „x“ in this formula. First the angle must be converted into radians, by multiplying „20“ by the number „π“ (3.1415) and then dividing by the number 180:

$$20^\circ = \frac{20 \times 3,1415}{180} = 0,349$$

20° corresponds to a radian measure of 0.349

0.349 can now be inserted into the „series“. In order to make it easier for non-mathematicians too to follow these calculations, the calculations are carried out step by step in the following representation:

$$\sin 20^\circ = 0,349 - \frac{0,349^3}{3!} + \frac{0,349^5}{5!}$$

$$\sin 20^\circ = 0,349 - \frac{0,349 \times 0,349 \times 0,349}{1 \times 2 \times 3} + \frac{0,249 \times 0,349 \times 0,349 \times 0,349 \times 0,349}{1 \times 2 \times 3 \times 4 \times 5}$$

$$\sin 20^\circ = 0,349 - \frac{0,0425}{6} + \frac{0,00518}{120}$$

$$\sin 20^\circ = 0,349 - 0,0071 + 0,0000432$$

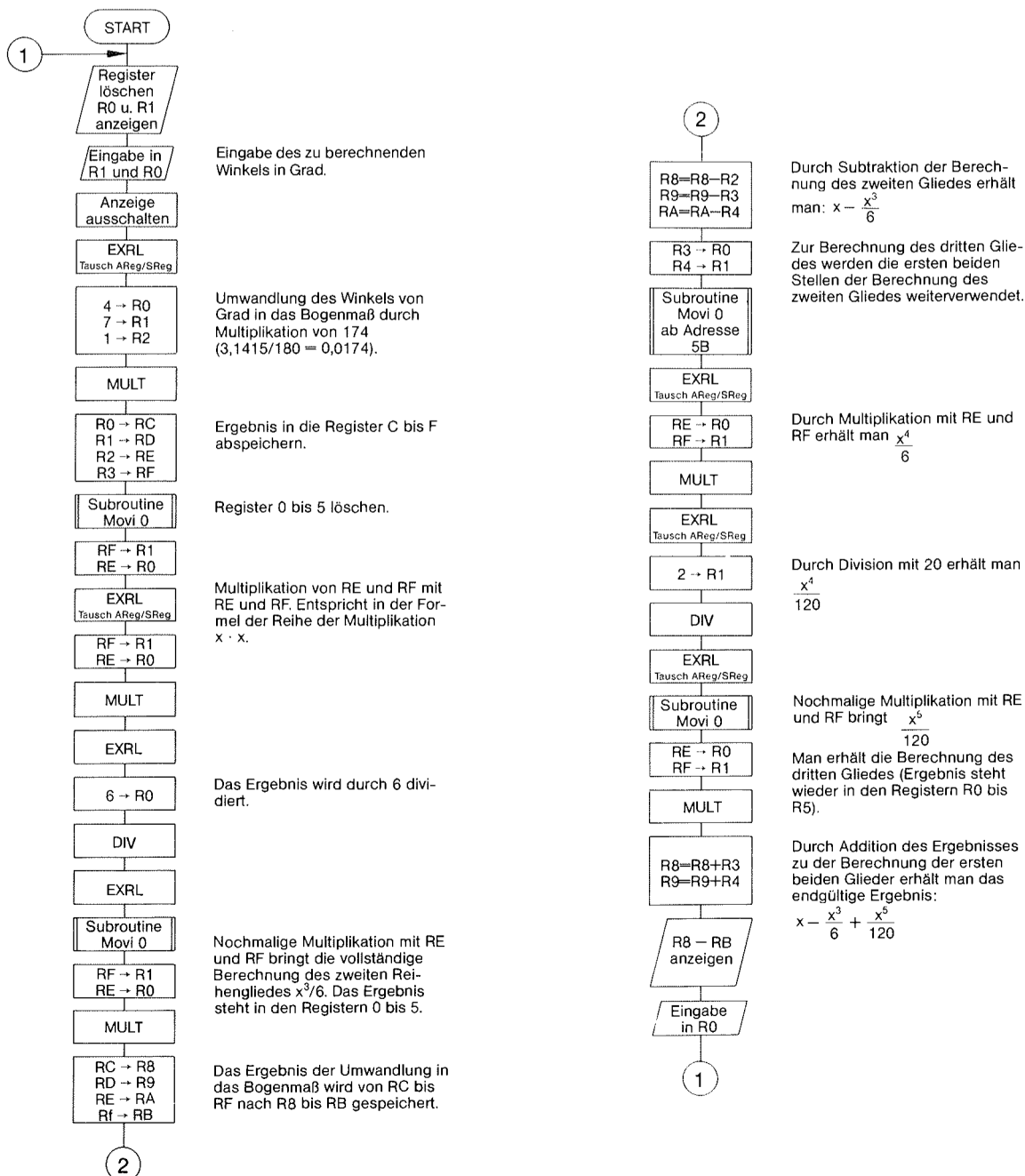
$$\sin 20^\circ = 0,3419$$

Our calculation yields the result 0.3419, while the computer calculated 0.3415. This slight deviation arises from the computer's rounding up and down. Since the maximum deviation is only 0.3%, however, our computer's accuracy is sufficient for most applications.

From the examples above it is evident that many multiplications must be carried out. Since their result is always smaller than 1, decimal places would be needed. The microprocessor, however, is not designed for this. By rounding the multiplications up and down, the computation uses whole numbers, saving much programming work.

The angle values are entered into registers 0 and 1 and then converted to radians by multiplication with 174 (3.1415 : 180 = 0.0174).

Program flow chart: Sine Calculation



Cosine and Tangent Calculations

Our Sine-Calculation program can also be „rebuilt“ for other trigonometric-function calculations. Here are a few suggestions:

The COSINE function can likewise be calculated with the aid of a „series“:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!}$$

Note that „x“ must be the angle converted into radians. Most pocket calculators can internally compute only angles from 0° to 45° of a sine or cosine. To find the sine of 60°, for instance, the calculator computes the cosine of 90° minus 60°, i.e. of 30°. This is possible because the following relationship holds:

$$\sin \delta = \cos (90^\circ - \delta)$$

$$\text{or } \cos \delta = \sin (90^\circ - \delta)$$

Through this simple trick, a good deal of computational work can be saved.

The TANGENT function can be obtained by computing the sine and cosine and then dividing the two results by each other:

$$\tan \delta = \sin \delta : \cos \delta$$

Through such considerations and manipulations, the microprocessor can and must perform enormous computational feats.

A three-part super-program!

Part 1:

Calculating the days between two dates

It is no problem to calculate the days that fall, say, between the 3rd and the 17th of a month. It becomes harder when we want to know how many days lie between 5 March 1981 and 24 December 1985.

For calculating these days, the following program helps us, automatically taking any leap years into account. The calculations are based on the Gregorian calendar.

Before we start programming, the entire program memory should be cleared, because the „day calculation“ is the first part of a super-program that is completed by the two following chapters. Note also that after address 5C we continue the second program part at address 88. The two parts are later completed by the Bio-Rhythm program.

Clearing the program memory: HALT – PGM – 5.

After HALT – NEXT – 00, carry out the programming very carefully:

Program: Calculating the days between 2 dates

Label	Adr.	Code	Mne- monic	Jump to
	00	F08	CLEAR	
	01	F0E	EXRM	
	02	F08	CLEAR	
	03	F6C	DISP 6,C	
	04	FF2	KIN 2	
	05	992	CMPI 9,2	
	06	D0E	BRC 0E	
	07	001	MOV D,1	
	08	0F0	MOV F,0	
	09	0EF	MOV E,F	
	0A	0DE	MOV D,E	
	0B	0CD	MOV C,D	
	0C	02C	MOV 2,C	
	0D	C03	GOTO 03	
	0E	F02	DISDUT	
	0F	017	MOV 1,7	
	10	006	MOV 0,6	
	11	B9E	CALL 9E	MOVI 0
	12	0C0	MOV C,0	
	13	0D1	MOV D,1	
	14	BAE	CALL AE	SUBI 1
	15	F76	MAS 6	
	16	F77	MAS 7	
	17	F0D	EXRL	
	18	B9E	CALL 9E	MOVI 0
	19	110	MOVI 1,0	
	1A	131	MOVI 3,1	
	1B	F0B	MULT	
	1C	BBC	CALL BC	ADD
	1D	90D	CMPI 0,D	
	1E	D2D	BRC 2D	MD
	1F	92C	CMPI 2,C	
	20	D2D	BRC 2D	MD
	21	B99	CALL 99	MOV J
	22	BAE	CALL AE	SUBI 1

	23	F0D	EXRL	
	24	140	MOVI 4,0	
	25	F0C	DIV	
	26	BBC	CALL BC	ADD
	27	B99	CALL 99	MOV J
	28	BAE	CALL AE	SUBI 1
	29	10D	MOVI 0,D	
	2A	C88	GOTO 88	INT
	2B	BDF	CALL DF	SUB
	2C	C43	GOTO 43	BERE
MD	2D	B9E	CALL 9E	MOVI 0
	2E	0D1	MOV D,1	
	2F	0C0	MOV C,0	
	30	F0D	EXRL	
	31	140	MOVI 4,0	
	32	F0B	MULT	
	33	134	MOVI 3,4	
	34	125	MOVI 2,5	
	35	BA5	CALL A5	ADD 2
	36	010	MOV 1,0	
	37	101	MOVI 0,1	
	38	BDF	CALL DF	SUB
	39	B99	CALL 99	MOV J
	3A	F0D	EXRL	
	3B	140	MOVI 4,0	
	3C	F0C	DIV	
	3D	BBC	CALL BC	ADD
	3E	B99	CALL 99	MOV J
	3F	11D	MOVI 1,D	
	40	C88	GOTO 88	INT
	41	10D	MOVI 0,D	
	42	BDF	CALL DF	SUB
BERE	43	B99	CALL 99	MOV J
	44	F0D	EXRL	
	45	B9E	CALL 9E	MOVI 0
	46	150	MOVI 5,0	

	47	161	MOVI 6,1	
	48	132	MOVI 3,2	
	49	F0B	MULT	
	4A	BBC	CALL BC	ADD
	4B	B9E	CALL 9E	MOVI 0
	4C	F20	DISP 2,0	
	4D	FF1	KIN 1	
	4E	FF0	KIN 0	
	4F	F02	DISOUT	
	50	BBC	CALL BC	ADD
	51	F0E	EXRM	
	52	90F	CMPI 0,F	
	53	D57	BRC 57	
	54	1FF	MOVI F,F	
	55	F7F	MAS F	
	56	C02	GOTD 02	
	57	F0F	EXRA	
	58	F0E	EXRM	
	59	BDF	CALL DF	SUB
	5A	F58	DISP 5,8	
	5B	FF0	KIN 0	
	5C	C00	GOTD 00	
„HALT-NEXT-88“.				
Then continue programming =				
INT	88	10C	MOVI 0,C	
	89	F0D	EXRL	
	8A	B9E	CALL 9E	MOVI 0
	8B	112	MOVI 1,2	
	8C	F0C	DIV	
	8D	114	MOVI 1,4	
	8E	BA5	CALL A5	ADD 2
	8F	F0D	EXRL	
	90	B9E	CALL 9E	MOVI 0
	91	130	MOVI 3,0	
	92	F0B	MULT	
	93	F0D	EXRL	
	94	140	MOV 4,0	
	95	F0C	DIV	
	96	90D	CMP 0,D	
	97	E2B	BRZ 2B	
	98	C41	GOTD 41	
MOV J	99	0E0	MOV E,0	
	9A	0F1	MOV F,1	
	9B	062	MOV 6,2	
	9C	073	MOV 7,3	
	9D	CA2	GOTO A2	
MOVI 0	9E	100	MOVI 0,0	
	9F	101	MOVI 0,1	
	A0	102	MOVI 0,2	
	A1	103	MOVI 0,3	
	A2	104	MOVI 0,4	
	A3	105	MOVI 0,5	
	A4	F07	RET	
ADD 2	A5	440	ADD 4,0	
	A6	DAA	BRC AA	
	A7	990	CMPI 9,0	

	A8	DAA	BRC AA	
	A9	CAC	GOTO AC	
	AA	560	ADDI 6,0	
	AB	511	ADDI 1,1	
	AC	451	ADD 5,1	
	AD	CA2	GOTO A2	
SUBI 1	AE	710	SUBI 1,0	
	AF	DB1	BRC B1	
	B0	F07	RET	
	B1	760	SUBI 6,0	
	B2	711	SUBI 1,1	
	B3	DB5	BRC B5	
	B4	F07	RET	
	B5	761	SUBI 6,1	
	B6	712	SUBI 1,2	
	B7	DB9	BRC B9	
	B8	F07	RET	
	B9	762	SUBI 6,2	
	BA	713	SUBI 1,3	
	BB	F07	RET	
ADD	BC	408	ADD 0,8	
	BD	DC1	BRC C1	
	BE	998	CMPI 9,8	
	BF	DC1	BRC C1	
	C0	CC3	GOTO C3	
	C1	568	ADDI 6,8	
	C2	519	ADDI 1,9	
	C3	419	ADD 1,9	
	C4	DC8	BRC C8	
	C5	999	CMPI 9,9	
	C6	DC8	BRC C8	
	C7	CCA	GOTO CA	
	C8	569	ADDI 6,9	
	C9	51A	ADDI 1,A	
	CA	42A	ADD 2,A	
	CB	DCF	BRC CF	
	CC	99A	CMPI 9,A	
	CD	DCF	BRC CF	
	CE	CD1	GOTO D1	
	CF	56A	ADDI 6,A	
	D0	51B	ADDI 1,B	
	D1	43B	ADD 3,B	
	D2	DD6	BRC D6	
	D3	99B	CMPI 9,B	
	D4	DD6	BRC D6	
	D5	CD8	GOTO D8	
	D6	56B	ADDI 6,B	
	D7	51C	ADDI 1,C	
	D8	44C	ADD 4,C	
	D9	DDD	BRC DD	
	DA	99C	CMPI 9,C	
	DB	DDD	BRC DD	
	DC	F07	RET	
	DD	56C	ADDI 6,C	
	DE	F07	RET	

SUB	DF	608	SUB 0,8	
	E0	DE2	BRC E2	
	E1	CE5	GOTO E5	
	E2	768	SUBI 6,8	
	E3	719	SUBI 1,9	
	E4	DE8	BRC E8	
	E5	619	SUB 1,9	
	E6	DE9	BRC E9	
	E7	CEC	GOTO EC	
	E8	619	SUB 1,9	
	E9	769	SUBI 6,9	
	EA	71A	SUBI 1,A	
	EB	DEF	BRC EF	
	EC	62A	SUB 2,A	
	ED	DF0	BRC F0	
	EE	CF3	GOTO F3	
	EF	62A	SUB 2,A	

	F0	76A	SUBI 6,A	
	F1	71B	SUBI 1,B	
	F2	DF6	BRC F6	
	F3	63B	SUB 3,B	
	F4	DF7	BRC F7	
	F5	CFA	GOTO FA	
	F6	63B	SUB 3,B	
	F7	76B	SUBI 6,B	
	F8	71C	SUBI 1,C	
	F9	DFD	BRC FD	
	FA	64C	SUB 4,C	
	FB	DFE	BRC FE	
	FC	F07	RET	
	FD	64C	SUB 4,C	
	FE	76C	SUBI 6,C	
	FF	F07	RET	

Program start with HALT – NEXT – 00 – RUN. The display shows: 000000.

To calculate how many days fall between 5 March 1981 and 24 December 1985, we must first enter the „earlier“ year, i.e.: 1981. Then the month 03. The display shows: 198103.

Press key A – after a few seconds the display shows: 00. Now the day is entered as well: 05. The display now shows 000000 again. The second entry follows: first the year (1985), then the month (12). The display shows: 198512. After pressing key A, enter the day of the month (24).

The computer performs the calculation. If the data and the program were entered correctly, the display shows the number of days lying between the two dates: 1755.

After pressing any number key, the display shows 000000 again, and the days between two other dates can be calculated.

It is important always to enter the earlier (smaller) date first. Months and days always require 2-digit entries, so a 0 must be prefixed to entries smaller than 10 (for July = 7th month – enter 07). The date entry must not be made too quickly, and the correct entry should be checked on the display.

With this program the display can show a maximum of 5 digits, i.e. at most 99,999 days. That is 273 years and 10 months. If the two dates to be calculated lie more than 274 years apart, the missing sixth display digit must be added in.

Example: For an entry between 1 January 1800 (180001 – 01) and 1 January 2200 (220001 – 01), the computer gives the result: 46097. In reality, however, 146097 days lie between the two dates.

The program you have entered is also needed for the two chapters that follow, so do not delete or change it.

Calculating a weekday within any year

With this program we can, by entering a date, calculate which weekday (Sunday, Monday, etc.) it was at that time.

The program already entered must be changed at a few program steps from address 51 onward. We make the entry after HALT – NEXT – 51 by entering the new commands in place of the existing ones (entering a new command code erases the code entered before):

Program change: Calculating the weekday

After we have entered the last command at address 6D, the program is started again: HALT – NEXT – 00 – RUN.

The display shows 000000 again.

First we want to test an easily verifiable date:

On which weekday does Christmas Eve fall in the year 1982?

As with the previous program, the date 24 December 1982 is again entered as follows: year and month (198212) – press key A – then the day (24). The display shows 6. From the weekday table we see that the number 6 is a Friday.

The program is designed so that weekday calculations can be made from 1 January 1918 up to 31 December 2189. If the entered dates lie before or after these years, the displayed results are wrong.

Weekday table

0 = Saturday
1 = Sunday
2 = Monday
3 = Tuesday
4 = Wednesday
5 = Thursday
6 = Friday

The program is still needed for the following chapter, so please do not delete or change it.

Program supplement: Weekday calculation

Label	Adr.	Code	Mnemonic	Jump to
	51	B9E	CALL 9E	MOVI 0
	52	090	MOV 9,0	
	53	0A1	MOV A,1	
	54	0B2	MOV B,2	
	55	0C3	MOV C,3	
	56	F0D	EXRL	
	57	170	MOVI 7,0	
	58	F0C	DIV	
	59	006	MOV 0,6	
	5A	017	MOV 1,7	
	5B	02E	MOV 2,E	
	5C	F0D	EXRL	
	5D	001	MOV 0,1	
	5E	080	MOV 8,0	
	5F	F0D	EXRL	
	60	B9E	CALL 9E	MOVI 0
	61	170	MOVI 7,0	
	62	F0C	DIV	
	63	061	MOV 6,1	
	64	072	MOV 7,2	
	65	0E3	MOV E,3	
	66	F0D	EXRL	
	67	B9E	CALL 9E	MOVI 0
	68	170	MOVI 7,0	
	69	F0B	MULT	
	6A	BDF	CALL DF	
	6B	F18	DISP 1,8	
	6C	FF0	KIN 0	
	6D	C00	GOTO 00	

Calculating the Biorhythm

The theory of biorhythm holds that in every person 3 different, constantly repeating cycles run their course, all beginning on the day of birth.

The first cycle, also called the physical or bodily rhythm, has a length of 23 days. The second cycle, the psychic or emotional rhythm, has a length of 28 days. The third cycle, an intellectual rhythm, has a length of 33 days.

Since the biorhythm assumes that a person's physical, emotional, and intellectual life runs in wave-like highs and lows that can be precisely delimited by the day, the „better“ and „worse“ days can be derived from the biorhythm.

If we are in the first half of the biorhythm, we are in a good phase. In the second half of the rhythm, our intellectual and physical constitution is supposedly somewhat poorer.

For an athlete the physical rhythm might be of primary interest, while for someone who works intellectually the intellectual rhythm can play a more essential role. In general, a person feels better the more rhythms are in the first half of their cycle.

How far the controversial doctrine of biorhythm (by W. Fließ and H. R. Früh) is reliably applicable shall be left open. For the computer, determining the personal biorhythm gives rise to very manifold arithmetic operations.

We can determine our biorhythmic constitution with the following program.

The program already entered, „Calculating the days between two dates“, stays in place. However, from address 51 up to address 87 it is changed or extended.

We begin the modification programming after HALT – NEXT – 51:

Supplementary program: Biorhythm

By entering this part of the program, our program memory is now „filled“ from address 00 up to the last possible address FF.

Label	Adr.	Code	Mnemonic	Jump to
	51	F0E	EXRM	
	52	90F	CMPI 0,F	
	53	D57	BRC 57	
	54	1FF	MOVI F,F	
	55	F7F	MAS F	
	56	C02	GOTO 02	
	57	F0F	EXRA	
	58	F0E	EXRM	
	59	BDF	CALL DF	SUB
	5A	136	MOVI 3,6	
	5B	127	MOVI 2,7	
	5C	11F	MOVI 1,F	
	5D	C65	GOTO 65	
	5E	186	MOVI 8,6	
	5F	127	MOVI 2,7	
	60	12F	MOVI 2,F	
	61	C65	GOTO 65	

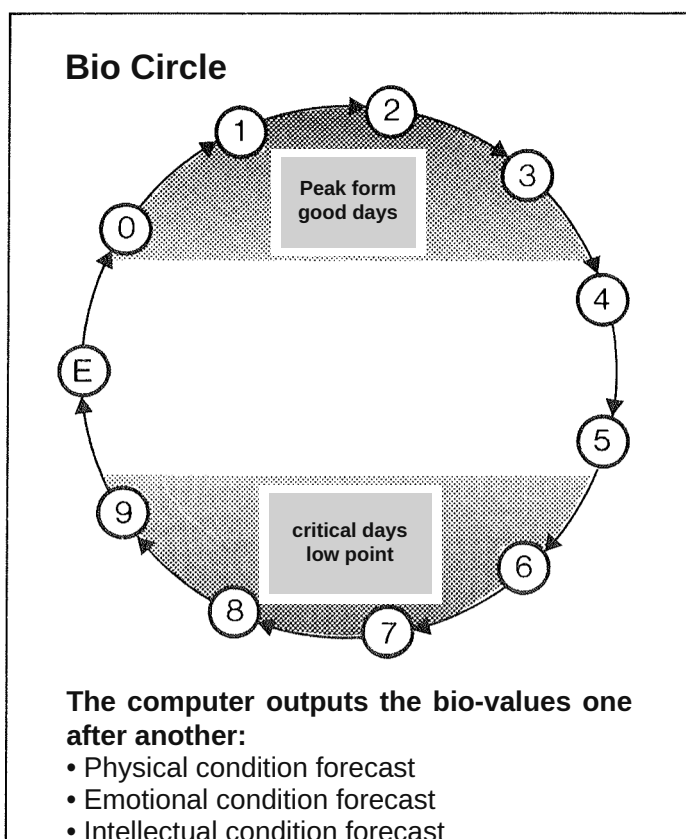
	62	136	MOVI 3,6	
	63	137	MOVI 3,7	
	64	13F	MOVI 3,F	
	65	F76	MAS 6	
	66	F77	MAS 7	
	67	B9E	CALL 9E	MOVI 0
	68	090	MOV 9,0	
	69	0A1	MOV A,1	
	6A	0B2	MOV B,2	
	6B	0C3	MOV C,3	
	6C	F0D	EXRL	
	6D	060	MOV 6,0	
	6E	071	MOV 7,1	
	6F	F0C	DIV	
	70	F0D	EXRL	
	71	012	MOV 1,2	
	72	001	MOV 0,1	
	73	080	MOV 8,0	
	74	F0D	EXRL	
	75	B9E	CALL 9E	MOVI 0
	76	060	MOV 6,0	
	77	071	MOV 7,1	
	78	F0C	DIV	
	79	F0D	EXRL	
	7A	012	MOV 1,2	
	7B	001	MOV 0,1	
	7C	100	MOVI 0,0	
	7D	F0D	EXRL	
	7E	B9E	CALL 9E	MOVI 0
	7F	060	MOV 6,0	
	80	071	MOV 7,1	
	81	F0C	DIV	
	82	F10	DISP 1,0	
	83	FF0	KIN 0	
	84	92F	CMPI 2,F	
	85	E62	BRZ 62	
	86	D00	BRC 00	
	87	C5E	GOTO 5E	

Program start with HALT – NEXT – 00. The display shows: 000000.

As an example, we want to calculate the biorhythm of a person born on 2 July 1952. The calculation should show us whether better or worse cycle-days are to be expected on 24 December 1982.

As in the previous experiments, the birth year and month (195207) are entered first. After pressing key A, wait until the display shows 00, then enter the birthday (02 – the 0 must not be forgotten). The second date of interest now follows in the same way: first the year and month (198212) are entered. Press key A – wait until the display shows 00, then enter the day (24).

Once we have entered the dates mentioned above, the display now shows E, as the digit for the physical rhythm.



When we look at the Bio Circle, we find the E on the left side, i.e. our test subject has physically left the critical days and is approaching the peak-form period.

We press key 0 – the display shows: 5. This is the value for the emotional condition, and we see from the Bio Circle that the emotional state is moving toward the low point.

After pressing key 0 again, the display shows: 3, the value for the intellectual condition. The Bio Circle shows that the test subject is in good shape for intellectual activity. According to the doctrine of biorhythm, our test subject could overall reckon on a very good intellectual, physical, and emotional condition if all three determined values lie between 0 and 4 on the Bio Circle.

After pressing the 0 key once more, the computer is ready for a new biorhythm calculation. The input order is generally: birth year and birth month (don't forget a leading 0 if needed), key A, then the birthday. The second date to be calculated is entered in the same way.

The dates to be calculated must not lie more than 273 years apart.

This is of interest to the programmer:

Register No.	Register functions
0	Computation registers and input registers for the day-date
1	
2	
3	
4	
5	
6	Storage for the first two digits of the year
7	
8	Factor calculation (Reg. 8 to C)
9	
A	Storage for month numbers (Reg. C and D)
B	
C	
D	
E	Storage for the last two digits of the year
F	

In the last three program experiments, time calculations had to be carried out. From the entered dates a factor is calculated. With this factor the final results are determined.

The factor is calculated by the following formula if the entered date falls in the month of January or February:

$$\text{Factor} = 365 \times \text{Year} + \text{Day} + 31 \times (\text{Month} - 1) + \text{INT}((\text{Year} - 1) \div 4) - \text{INT}(3/4 \times \text{INT}(((\text{Year} - 1) \div 1) + 1)).$$

The notation INT (integer) means that, in the result of the following calculation, the digits after the decimal point are to be ignored. This simplifies our programming.

If the entered date falls between March and December, the factor is calculated by the following formula:

$$\text{Factor} = 365 \times \text{Year} + \text{Day} + 31 \times (\text{Month} - 1) - \text{INT}(0.4 \times \text{Month} + 2.3) + \text{INT}(\text{Year} \div 4) - \text{INT}(3/4 \times (\text{INT}(\text{Year} \div 100) + 1)).$$

As an example, the factor for 24 December 1982 is to be calculated. Into the formula we must substitute 1982 for the year, 12 for the month, and 24 for the day:

$$\text{Factor} = 365 \times 1982 + 24 + 31 \times (12 - 1) - \text{INT}(0.4 \times 12 + 2.3) + \text{INT}(1982 \div 4) - \text{INT}(3/4 \times (\text{INT}(1982 \div 100) + 1)).$$

$$\text{Factor} = 723430 + 24 + 341 - 7 + 495 - 15.$$

$$\text{Factor} = 724268$$

In the program „Calculating the days between two dates“, two factors are calculated from the two entered dates. The two factors are subtracted from each other, and the result is the number of days.

The principle is thus relatively simple, but it requires a fairly large amount of computation.

In the program „Biorhythm“, the number of days between the birth date and the date to be calculated is determined by the same principle. This result is divided by the number of days of the corresponding rhythm (the rhythm days are 23, 28, and 33). The remainder of this division is multiplied by 10 and then divided by the number of rhythm days. This result yields a digit between 0 and 9 or E, which is shown on the display.

In the „weekday calculation“, the factor is likewise calculated from the entered date. This factor must then be processed further by the following formula:

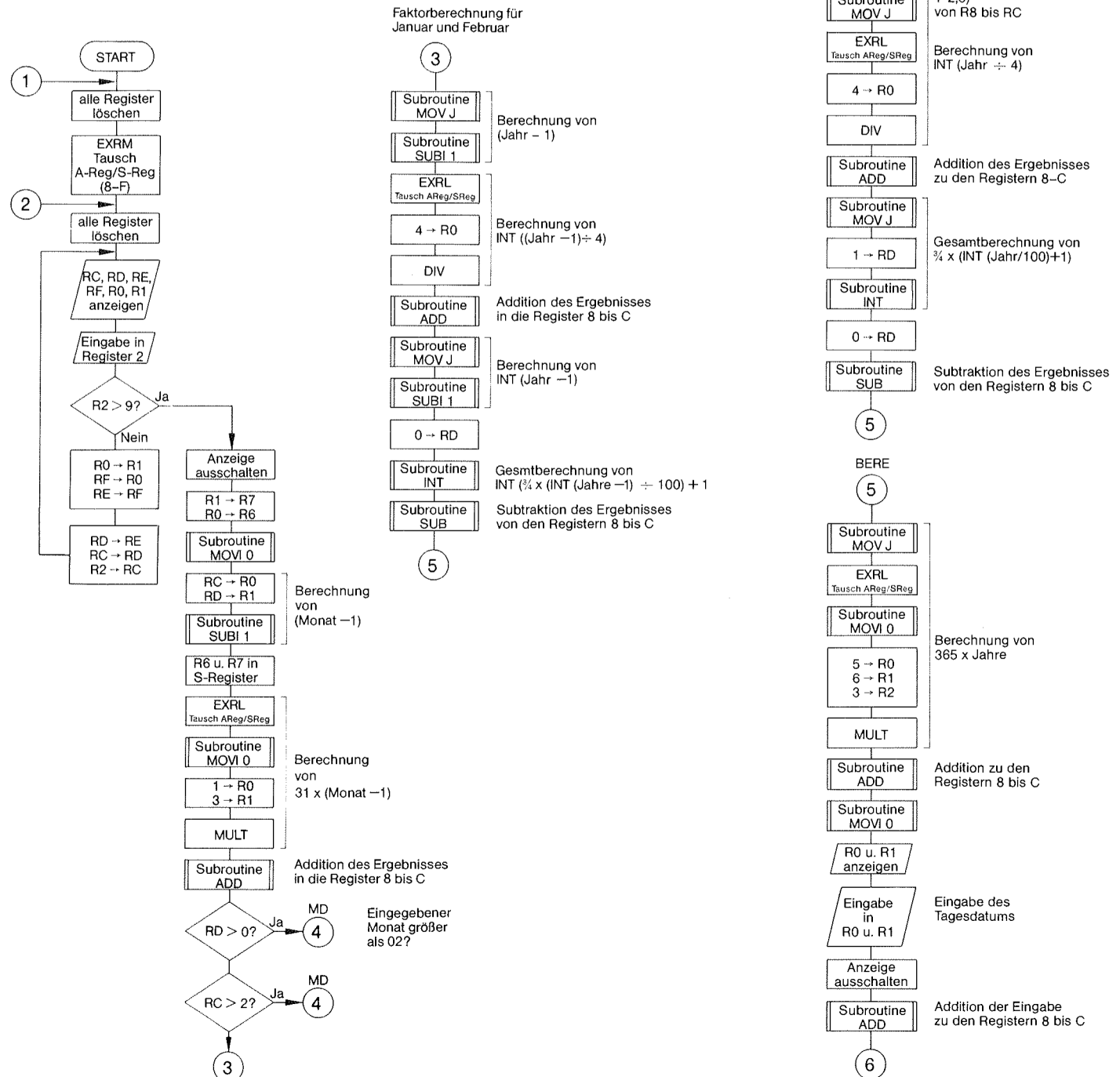
$$\text{Weekday} = \text{Factor} - (\text{INT}(\text{Factor} \div 7) \times 7).$$

If we again insert the factor calculated for the date 24 December 1982 into the formula above, the following calculation results:

Weekday = $724268 - (\text{INT}(724268 \div 7) \times 7)$.
 Weekday = $724268 - (\text{INT}(103466.8571) \times 7)$.
 Weekday = $724268 - (103466 \times 7)$.
 Weekday = $724268 - 724262 = 6$
 Weekday = a Friday (6 – see weekday table).

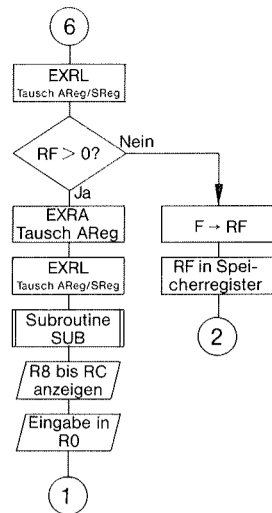
Although this calculation, too, is relatively easy to understand, it results in an elaborate program, because many computation steps have to be carried out. To simplify the program, the computer works with only 5 digits. In the factor 724268, the first digit (7) is not included in the calculation. However, we see from the representations of the weekday calculation that the first digit (7) does not change and therefore has no influence on the result.

Program flow chart: Factor calculation



Program flow charts:

Calculating the days between two dates



The greatest programming effort is required for calculating the „factor“ (see also the program flow chart). To keep the programming effort as low as possible, several subroutines are used that we have already met in other programs. These subroutines are therefore not listed again in the program flow chart. For the sake of completeness, however, they shall be mentioned briefly:

The subroutine **SUB** performs a 5-digit subtraction. Registers 0 to 4 are subtracted from registers 8 to C. The result is left in registers 8 to C. All calculations are carried out in decimal. Decimal subtraction was discussed in the „pocket calculator“ program.

The subroutine **ADD** adds registers 0 to 4 to registers 8 to C. The subroutine ADD performs a 5-digit decimal addition.

The subroutine **SUBI** adds 1 to a number present in registers 0 to 3.

The subroutine **ADD 2** adds registers 4 and 5 in decimal to registers 1 and 2.

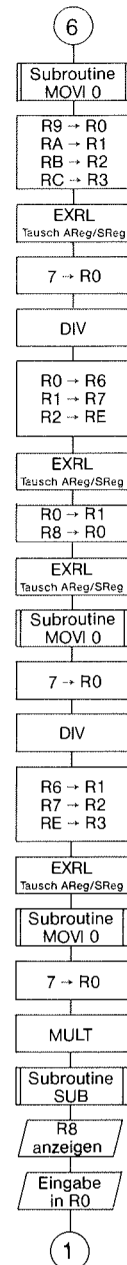
The subroutine **MOVI 0** clears registers 0 to 5.

The subroutine **MOV J** (move-years) brings the year values temporarily stored in registers 6, 7, E, and F into registers 0 to 3.

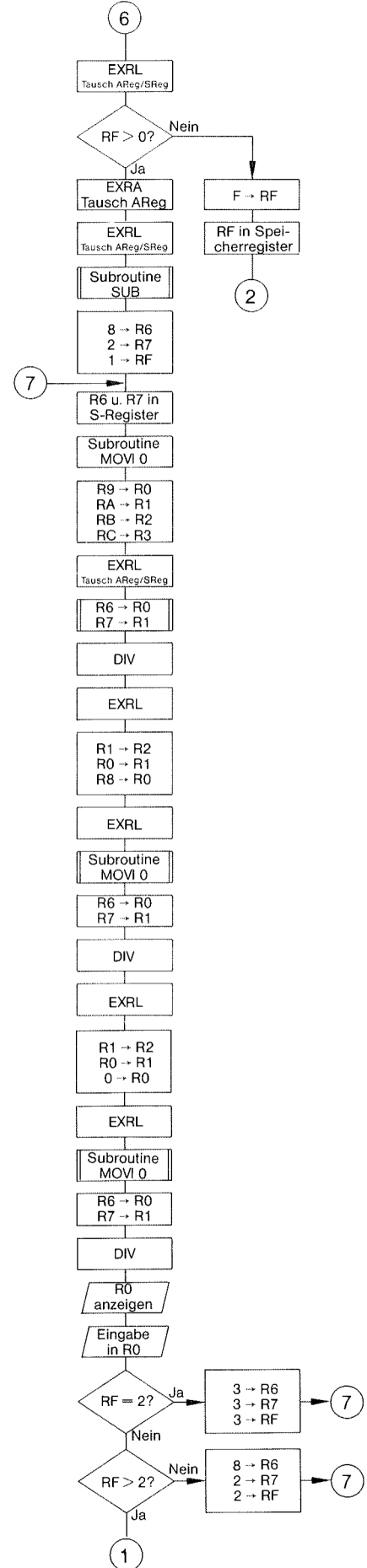
The program section **INT** calculates the formula part $\text{INT}(3/4 \times (\text{INT}(\text{Years} \div 100) + 1))$.

The individual formula segments of the INT program section are calculated one after another and added to or subtracted from registers 8 to C. As the result of these calculations, registers 8 to C then hold the factor, which can be processed further by the following program.

Calculating a weekday



Calculating the biorhythm



An important note for programmers!

The subroutines **ADD** (5-digit decimal addition) and **SUB** (5-digit decimal subtraction) can be used for a wide variety of programs you develop yourself.

We entered the subroutine ADD in the program „Calculating the days between two dates“ from address BC to address DE. The subroutine SUB begins at address DF and ends at FF.

Both subroutines are located at the end of the program memory and can be called at any time (provided the computer is not disconnected from mains power). We recall that with the CALL command we can jump from a program into a subroutine. With the command CALL BC we reach the subroutine ADD; with the command CALL DF, the subroutine SUB.

Computer experiments with additional electronic components

The computer's capabilities are considerably expanded by additionally connected components. Such add-on circuits are called „peripheral electronics“.

Important to observe for circuit experiments:

When experimenting, **under no circumstances may voltages above 9 V DC** be used. Therefore always experiment only with the intended 9-volt battery or with the BUSCH power supply 2059 (which likewise delivers 9 volts DC). Voltages above 9 volts — and AC voltages in particular — can destroy individual computer elements.

The computer outputs must never be connected directly to the plus or minus pole of a 9-volt voltage source (battery or power supply).

At the computer outputs, only small control voltages (maximum 15 mA) are available. Therefore only elements with low power requirements can be driven directly from the computer outputs, such as piezo buzzers, light-emitting diodes, transistors, ICs, and so on. With one transistor and 1 to 2 resistors, however, small amplifier circuits can very easily be built, through which other circuit parts can then also be controlled by the computer outputs. On the following pages, various connection options are demonstrated.

Since the computer inputs, too, need only small voltages for the input signals, using a series resistor is advisable in every case, because this considerably extends the life of the connected battery. On page 54 of the instruction book, Part 1, the arrangement of such series resistors (4.7 k Ω) in connection with the red keys was shown.

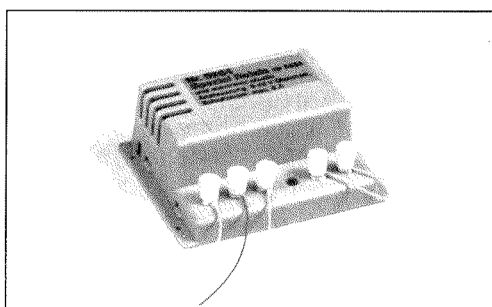
Connecting relays

From various experiments (instruction book, Part 1) we know that the computer can take over switching functions. For this, with the DOT command and appropriate programming, we put „a voltage“ on the computer outputs. With this small control voltage, however, we can drive only, say, the piezo buzzer or a light-emitting diode.

Larger current consumers get their power supply via a switching relay. The switching relay is driven from the computer output. Since the relay function too needs more current or voltage than is available at the computer output, the output voltage must be amplified by a transistor circuit.

Two relay options are available:

BUSCH low-current switching relay no. 5964



This relay is also offered, among other places, in the BUSCH model-railway accessory range. It can be used for a wide variety of switching and control functions.

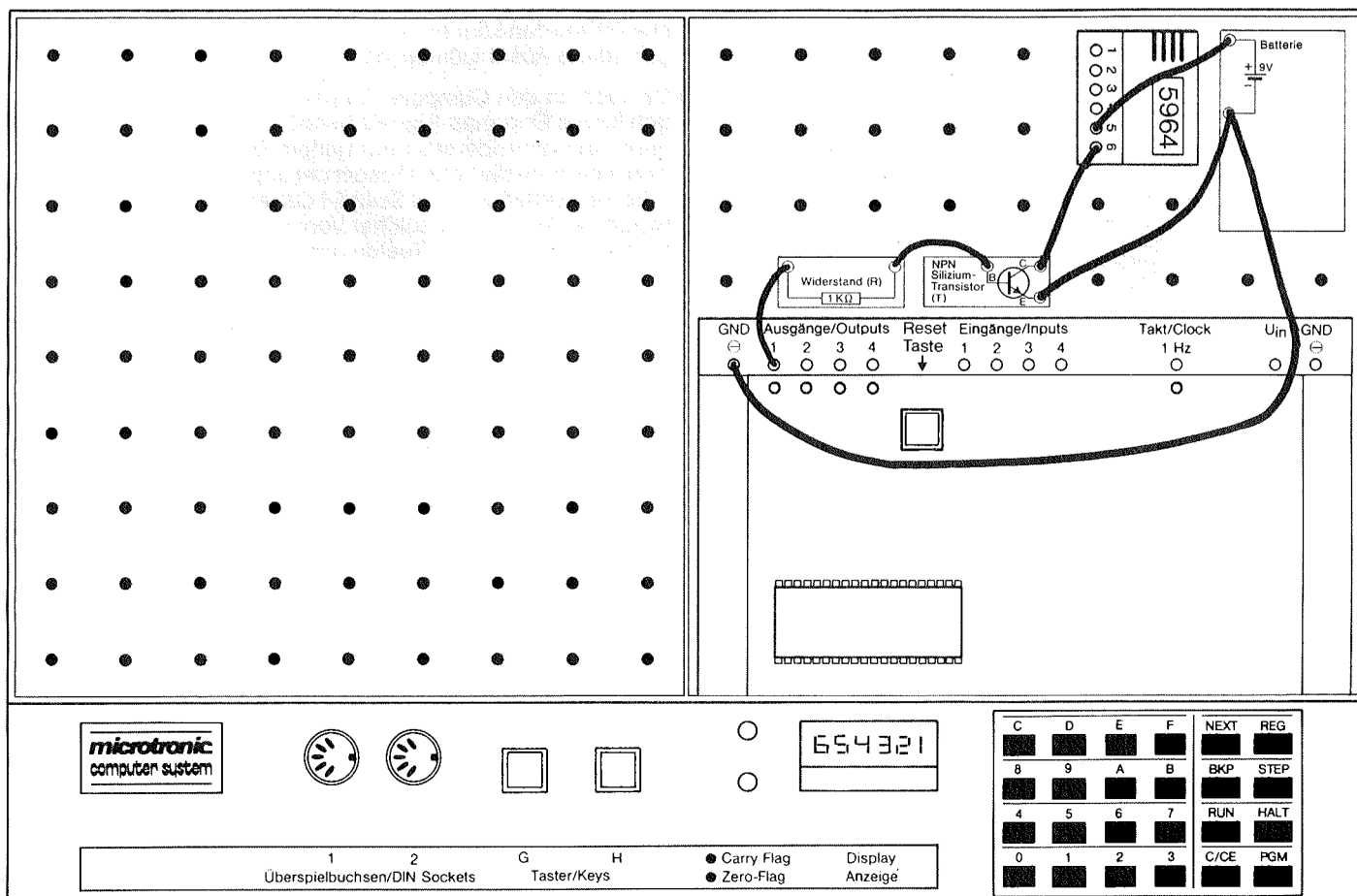
All kinds of devices capable of low-current operation (up to a maximum of 24 volts, 8 A) can be switched on or off — on a model-railway layout, for example, lighting, cranes for automatic loading and unloading, bells, horns, sirens, and so on. Setting or switching points is also possible if points with automatic end shut-off are used.

The 5964 relay works as a „changeover switch“. Two separate circuits can be switched.

The figure shows how the relay is driven, e.g. from computer output no. 1 with a 1 kΩ resistor and a transistor. As soon as the DOT command puts „a voltage“ on the output, the relay's switching action is triggered: circuit 1 off — circuit 2 on. This state is held until „no voltage“ is present at the computer output any longer. The relay springs back: circuit 1 on — circuit 2 off. The corresponding circuits can be connected at the relay's contacts. A detailed description is supplied with the relay.

For experiments — e.g. with a model railway — it is essential that live parts never come into contact with a computer connection. The model-railway transformer voltage of 16 V AC will inevitably destroy individual components.

Connecting the 5964 relay to a computer output



Since each computer output can be driven individually with appropriate programming, up to 4 relays can be connected. Each relay requires one 1 k Ω resistor and one transistor.

BUSCH mains-power switching unit 2087



Besides many other uses, the computer — together with BUSCH Electronic Studios and the 2087 switching unit — can, for example, simulate the presence of a house's occupants during a holiday absence. A photoresistor switches a light on at nightfall, which switches off again at dawn. A noise-sensitive circuit provides temporary extra lighting whenever a certain noise level is exceeded. Programmed as a timer, the computer can also handle two further on/off actions, with separate times settable. A secured door triggers an alarm tone as soon as the door is opened.

The following program examples are meant only as suggestions. Listing and explaining every possible application would go beyond the scope of this instruction book. For owners of additional BUSCH Electronic Studios, countless possibilities open up.

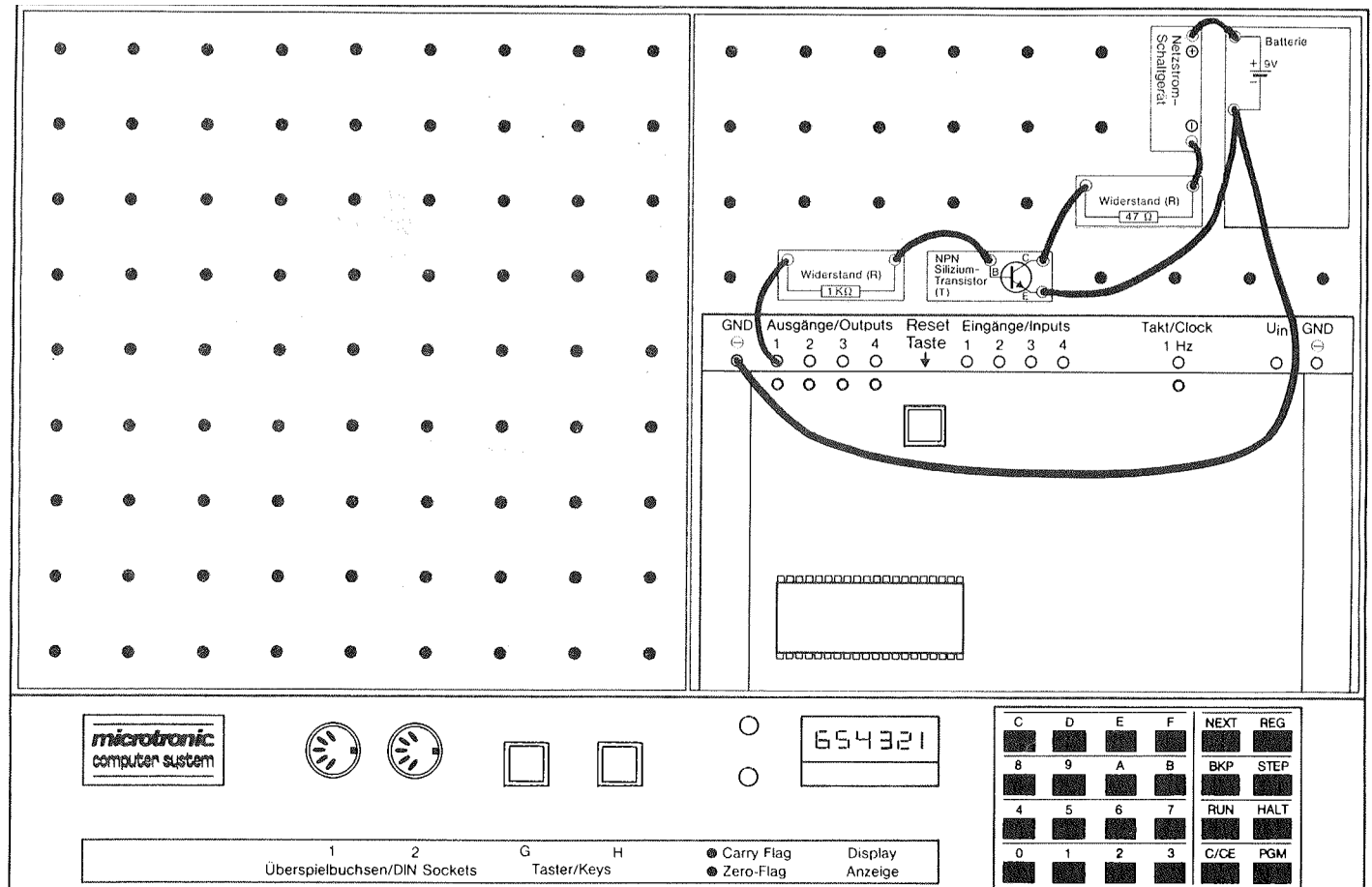
This switching unit, too, can be driven with low current (6–9 volts). Via a relay it triggers switching operations in the 220 V mains lighting circuit (220 V, max. 1000 watts). With two resistors and a transistor (see figure), the unit is connected to a computer output.

The 2087 switching unit complies with the VDE regulations. It is completely safe to use, because the 220 V mains section is fully isolated from the low-current side.

The 2087 switching unit is an „on/off switch“. As soon as a small control voltage is present, the relay closes a 220 V circuit; this circuit is switched off again when the control voltage is interrupted. The relay is designed for 20 million switching operations and can run continuously. It switches radios, tape recorders, warning systems, fan heaters and all electrical devices rated up to 1000 watts on and off. If devices with a higher rating are switched, a considerable drop in switching capacity must be expected, so when connecting fan heaters and the like it is essential to ensure that the 1000-watt limit is not exceeded.

Up to 4 mains-power switching units 2087 can be connected to the computer outputs. Each unit requires two resistors and one transistor.

Connecting the 2087 mains-power switching unit to a computer output



Timer clock with two on- and off-switching times

With the following program, two separate on- and off-switching times can be achieved. In this way, two different functions at different times are possible.

After HALT – NEXT – 00, enter the program:

Program: Timer clock for two on- and off-times

Label	Adr.	Code	Mnemonic	Jump to	Explanations
	00	F08	CLEAR		Clear working registers
	01	F0D	EXRL		Swap working/memory registers
	02	F08	CLEAR		Clear working registers
	03	FE8	DOT 8		Switch off outputs
„START“	04	108	MOVI 0,8		
	05	F18	DISP 1,8		Display R8
	06	FF8	KIN 8		Input into R8
	07	9A8	CMPI A,8		
	08	E3A	BRZ 3A	„S UHR“	
	09	9B8	CMPI B,8		
	0A	E32	BRZ 32	„UHR“	
	0B	9C8	CMPI C,8		
	0C	E13	BRZ 13	„EIN 1“	
	0D	9D8	CMPI D,8		
	0E	E17	BRZ 17	„AUS 1“	
	0F	9E8	CMPI E,8		
	10	E1B	BRZ 1B	„EIN 2“	
	11	D1D	BRC 1D	„AUS 2“	
	12	C04	GOTO 04	„START“	
„EIN 1“	13	F0D	EXRL		Swap memory/working registers
	14	B1F	CALL 1F	„S EIN“	
	15	F0D	EXRL		Swap working/memory registers
	16	C04	GOTO 04	„START“	
„AUS 1“	17	F0D	EXRL		Swap memory/working registers
	18	B29	CALL 29	„S AUS“	
	19	F0D	EXRL		Swap working/memory registers
	1A	C04	GOTO 04	„START“	
„EIN 2“	1B	B1F	CALL 1F	„S EIN“	
	1C	C04	GOTO 04	„START“	
„AUS 2“	1D	B29	CALL 29	„S AUS“	
	1E	C04	GOTO 04	„START“	
„S EIN“	1F	F40	DISP 4,0		Subroutine „S EIN“
	20	FF8	KIN 8		Entry of the switch-on time
	21	998	CMPI 9,8		
	22	D28	BRC 28		
	23	023	MOV 2,3		
	24	012	MOV 1,2		
	25	001	MOV 0,1		
	26	080	MOV 8,0		
	27	C1F	GOTO 1F		
	28	F07	RET		
„S AUS“	29	F44	DISP 4,4		Subroutine „S AUS“
	2A	FF8	KIN 8		Entry of the switch-off time
	2B	998	SMPI 9,8		
	2C	D28	BRC 28		
	2D	067	MOV 6,7		
	2E	056	MOV 5,6		
	2F	045	MOV 4,5		
	30	084	MOV 8,4		
	31	C29	GOTO 29		

„UHR“	32	108	MOVI 0,8		Clock mode without switching function
	33	FE8	DOT 8		Outputs switched off
	34	F6A	DISP 6,A		
	35	F06	TIME		
	36	FD8	DIN 8		
	37	FA8	SHL 8		
	38	E34	BRZ 34		
	39	C04	GOTO 04	„START“	
„S UHR“	3A	F4C	DISP 4,C		Timer-clock mode
	3B	F06	TIME		
	3C	B65	CALL 65	„V EIN“	
	3D	D40	BRC 40		
	3E	12A	MOVI 2,A		
	3F	AA9	OR A,9		
	40	B74	CALL 74	„V AUS“	
	41	D43	BRC 43		
	42	3D9	ANDI D,9		
	43	F0D	EXRL		
	44	B65	CALL 65	„V EIN“	
	45	D48	BRC 48		
	46	11A	MOVI 1,A		
	47	AA9	OR A,9		
	48	B74	CALL 74	„V AUS“	
	49	D4B	BRC 4B		
	4A	3E9	ANDI E,9		
	4B	F0D	EXRL		
	4C	FD8	DIN 8		Entry of buttons G and H into R8
	4D	338	ANDI 3,8		Clear clock pulse from R8
	4E	E60	BRZ 60	„OR“	
	4F	938	CMPI 3,8		
	50	E04	BRZ 04	„START“	
	51	09A	MOV 9,A		
	52	918	CMPI 1,8		
	53	E5B	BRZ 5B	„TA-G“	
„TA-H“	54	F9A	SHR A		Evaluation of button H
	55	F9A	SHR A		
	56	D59	BRC 59		
	57	D29	ADDI 2,9		
	58	C60	GOTO 60	„OR“	
	59	3D9	ANDI D,9		
	5A	C60	GOTO 60	„OR“	
„TA-G“	5B	F9A	SHR A		Evaluation of button G
	5C	D5F	BRC 5F		
	5D	519	ADDI 1,9		
	5E	C60	GOTO 60	„OR“	
	5F	3E9	ANDI E,9		
„OR“	60	339	ANDI 3,9		If outputs 1 and 2 are „low“ (0), the zero flag is set, so output 3 also becomes „low“.
	61	E63	BRZ 63		
	62	549	ADDI 4,9		
	63	FE9	DOT 9		
	64	C3A	GOTO 3A	„S UHR“	
„V EIN“	65	83F	CMP 3,F		10-hour comparison
	66	E68	BRZ 68		
	67	C70	GOTO 70		

	68	82E	CMP 2,E		Hour comparison
	69	E6B	BRZ 6B		
	6A	C70	GOTO 70		
	6B	81D	CMP 1,D		10-minute comparison
	6C	E6E	BRZ 6E		
	6D	C70	GOTO 70		
	6E	80C	CMP 0,C		Minute comparison
	6F	E72	BRZ 72		
	70	F09	STC		Set carry flag (1)
	71	F07	RET		
	72	F0A	RSC		Reset carry flag (0)
	73	F07	RET		
„V AUS“	74	87F	CMP 7,F		10-hour comparison
	75	E77	BRZ 77		
	76	C70	GOTO 70		
	77	86E	CMP 6,E		Hour comparison
	78	E7A	BRZ 7A		
	79	C70	GOTO 70		
	7A	85D	CMP 5,D		10-minute comparison
	7B	E7D	BRZ 7D		
	7C	C70	GOTO 70		
	7D	84C	CMP 4,C		Minute comparison
	7E	E72	BRZ 72		
	7F	C70	GOTO 70		

After the program has been entered, the two red buttons G and H must be connected to INPUTS 1 and 2, and in addition the TAKT/CLOCK connecting lead to INPUT 4. The piezo buzzer is connected to OUTPUT 3 and GND. The connecting leads of the piezo buzzer must not be interchanged. (See figure.)

After the keys HALT – PGM – 3, enter the current time of day. Then start the program with HALT – NEXT – 00 – RUN. The display shows 0. Now the following options arise:

Button B: The clock is shown on the display. The timer clock is still working without switching functions. If one of the two red buttons G or H is pressed, the display shows 0 again and the computer waits for the next entry.

Button C: The first switch-on time can be entered (hours and minutes). If a wrong time was entered by mistake, repeat the correct entry until the desired hours and minutes are shown on the display. Finish the entry by pressing button C again. The display shows 0 again – the computer waits for a new entry.

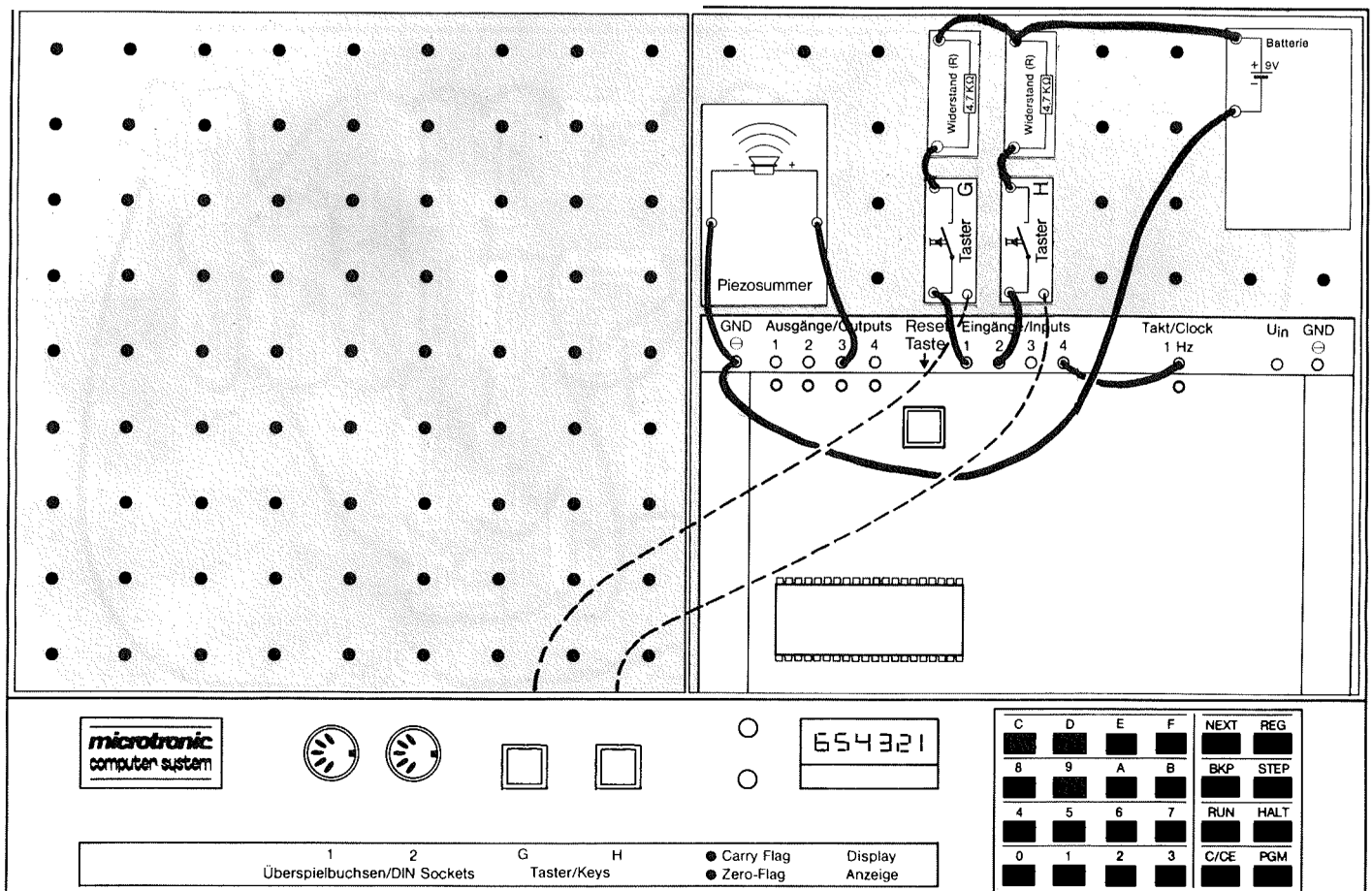
Button D: Entry of the first switch-off time (hours and minutes). Finish the entry by pressing button D again. The display shows 0 again.

Button E: Entry of the second switch-on time. End the entry by pressing button E once more.

Button F: Entry of the second switch-off time. End the entry by pressing button F once more.

Button A: The timer-clock functions are put into operation. The display shows the current time of day in hours and

Connecting the buttons & piezo buzzer for the timer clock



At the first switch-on time, the LEDs at output 1 and 3 light up. At the second switch-on time, the LEDs at outputs 2 and 3 light up. With the red button G, outputs 1 and 3 can thus be switched on or off, and with button H outputs 2 and 3.

To change the switching times, both buttons G and H must be pressed at the same time. The display shows 0 again, and we can now use buttons C, D, E and F to display the set times, or change them by entering new ones. After finishing the display or the change, the corresponding letter button is pressed once more in each case. With button A, the timer clock is put back into operation.

Computer-controlled timer clock with many variation options

For the following experiments, the „Schaltuhr“ program from the previous chapter continues to be used. By connecting additional components, the following variations arise, for example:

Radio alarm clock

(BUSCH Electronic Studio 2070 required).
With the BUSCH Studio Center 2070, a radio receiver is built. (See figure.) In radio circuits, the connecting cable normally runs from the minus pole of the battery to the slide switch of the Electronic Studio. Since the radio is to be switched on by the computer, the „Überspielkabel“ (patch-cable) module is connected in between, as can be seen from the figure. We can now connect the Electronic Center 2070 with the assembled radio to the computer by joining the two devices with a standard patch cable (available at any radio shop). This patch cable runs from the patch socket of the Studio Center to patch socket no. 1 of the computer.

In the computer housing, a simple amplifier circuit is built with a transistor and the 1 kΩ resistor, connected to the „Überspielkabel 1“ module. Since we cannot operate our radio set with the low voltage of the computer output, building the small transistor amplifier is necessary. The transistor amplifier circuit is connected to OUTPUT 2 of the computer. The piezo buzzer is connected to OUTPUT 1 of the computer. The arrangement of the buttons stays the same as in the previous experiment. The figure shows all the required circuit assemblies and changes.

Entering the switch-on and switch-off times, as well as putting it into operation, still follows the instructions in the „Schaltuhr“ (timer clock) chapter.

It makes sense to check the assembled radio set for perfect operation before connecting the „Überspielkabel“ module in between.

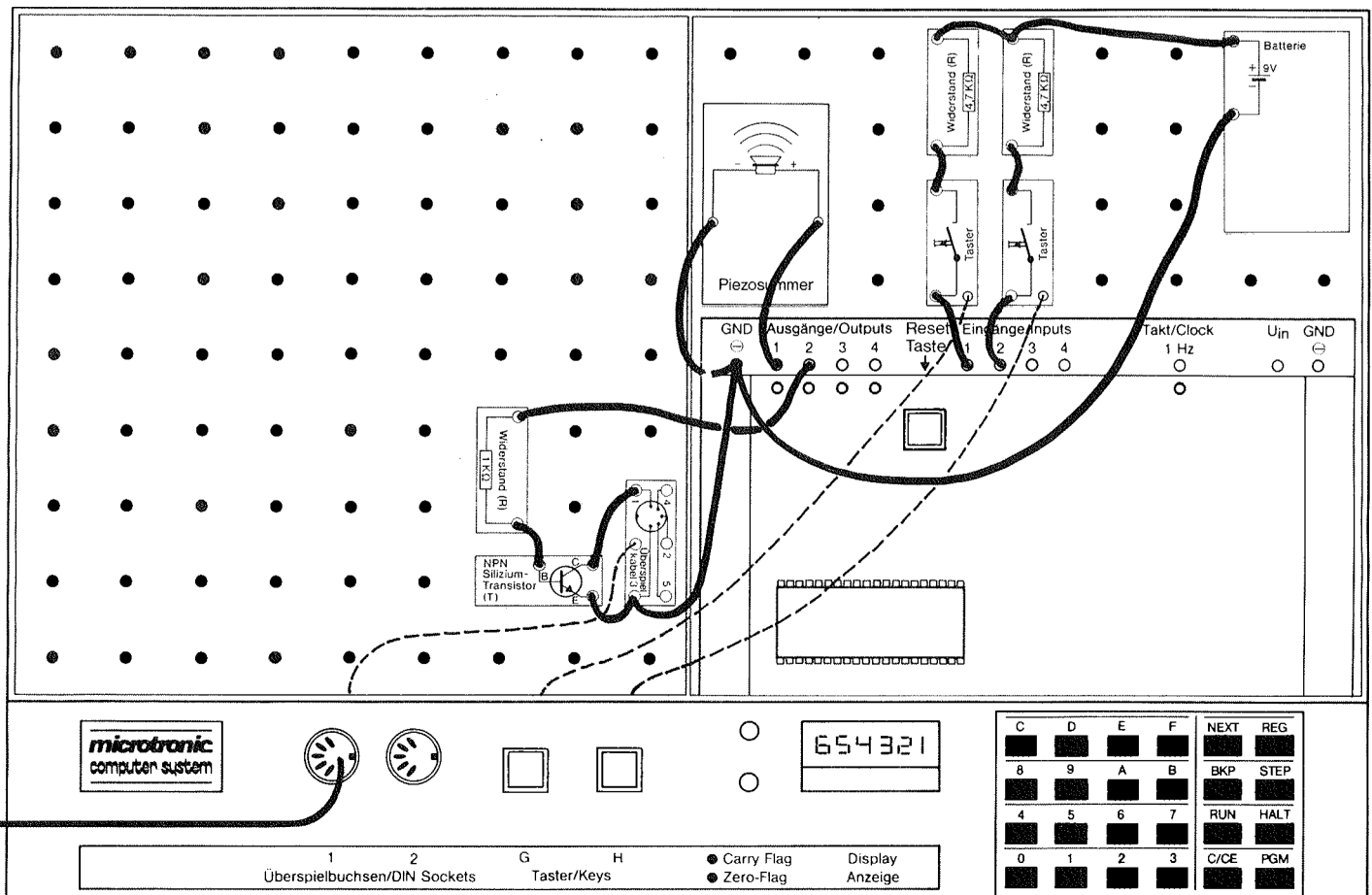
Entering the first alarm time will now trigger the piezo buzzer, while the second alarm time switches on the radio receiver.

We can now operate the two interconnected devices (computer with radio receiver) as follows:

The first alarm time is again entered as 06:30, the first switch-off time as 06:31. As the second switch-on time we again enter 06:45, and as the second switch-off time 07:15.

The computer signals at 6:30 in the morning with a buzzing tone lasting one minute. After a 14-minute pause, the radio set is switched on at 6:45. If the radio receiver is not switched off by pressing a red button, it switches off automatically at 7:15.

Timer clock for MW radio connection



Alarm clock with snooze radio

If we like to fall asleep to quiet music in the evening but wake up again a few hours later because the radio is still playing, we can turn our radio alarm clock into a snooze radio by changing the time entry. For this, the entire arrangement of the radio alarm clock described above stays in place. We merely make a change to the times.

Example: we go to bed at 22:00 and want the radio to switch off at 23:00. Only the second switch-off time (button F) is changed, by entering 23:00. As a result, our radio receiver does not switch off automatically at 7:15, as with the radio alarm clock described earlier, but only at 23:00.

If we switch on at 22:00 by pressing the red button H, automatic switch-off occurs at 23:00. Since we have not changed the other switch-on times, however, our „Schlummer-Radio-Wecker“ (snooze-radio alarm clock) will pre-wake us at 6:30 with a buzzing tone lasting one minute, and at 6:45 the radio switches on. Once we have got up, a single press of the red button H is enough to switch off the receiver. In the evening it is switched on again with button H, and switches off automatically at 23:00 (or at any other entered time).

If the radio is not needed during the day, we can additionally switch it off at the slide switch of the Studio Center, because our transistor amplifier circuit draws a very small operating voltage from the battery, which we can save by switching off.

Incidentally, we can achieve the same effect with a mains-powered radio receiver or a stereo system. Instead of the home-built radio and the transistor circuit, we use the BUSCH mains-power switching unit 2087. In this case the piezo buzzer stays on output 1 of the computer, while the mains-power switching unit is connected to output 2 (as was shown in the figure on page 43). The radio receiver is connected to the mains-power switching unit.

More „Schaltuhr“ possibilities

The entered „Schaltuhr“ program stays unchanged.

The figure shows that we have connection options at three computer outputs for the „Schaltuhr“ program.

At output no. 1, the simple transistor amplifier circuit, with which, for example, a home-built radio or other circuits can be driven — circuits that are possible with the BUSCH Electronic Studios 2060, 2065, 2070 or 2075.

At output 2, the low-current relay 5964 is connected, for time-dependent control tasks such as model-railway lighting, etc.

At output 3, the BUSCH mains-power switching unit 2087 is connected. Output 3 is switched on whenever output 1 **or** output 2 **or** both outputs are switched on. Of course, instead of the variations shown with three different connection options, we could also connect, for example, 3 relays 5964 or 3 mains-power switching units 2087 to the outputs.

Application example

During our absence, a 90-minute radio broadcast is to be recorded. The maximum playing time of an ordinary cassette is 45 minutes (if it is not turned over). If we own two cassette recorders, we connect three mains-power switching units 2087 to outputs 1 to 3. The first cassette recorder, with 45 minutes of playing time, is connected to the mains-power switching unit on output 1; the second cassette recorder to the mains-power switching unit on output 2. The radio set needed for the recording is connected to the mains-power switching unit on output 3.

The radio broadcast is to last, for example, from 15:00 to 16:30. The first switch-on time is 15:00, the first switch-off time 15:45. The second switch-on time is 15:45, the second switch-off time 16:30. The radio receiver is in operation the whole time, because it is connected to the mains switching unit on output 3.

„Schaltuhr“: of interest to the programmer

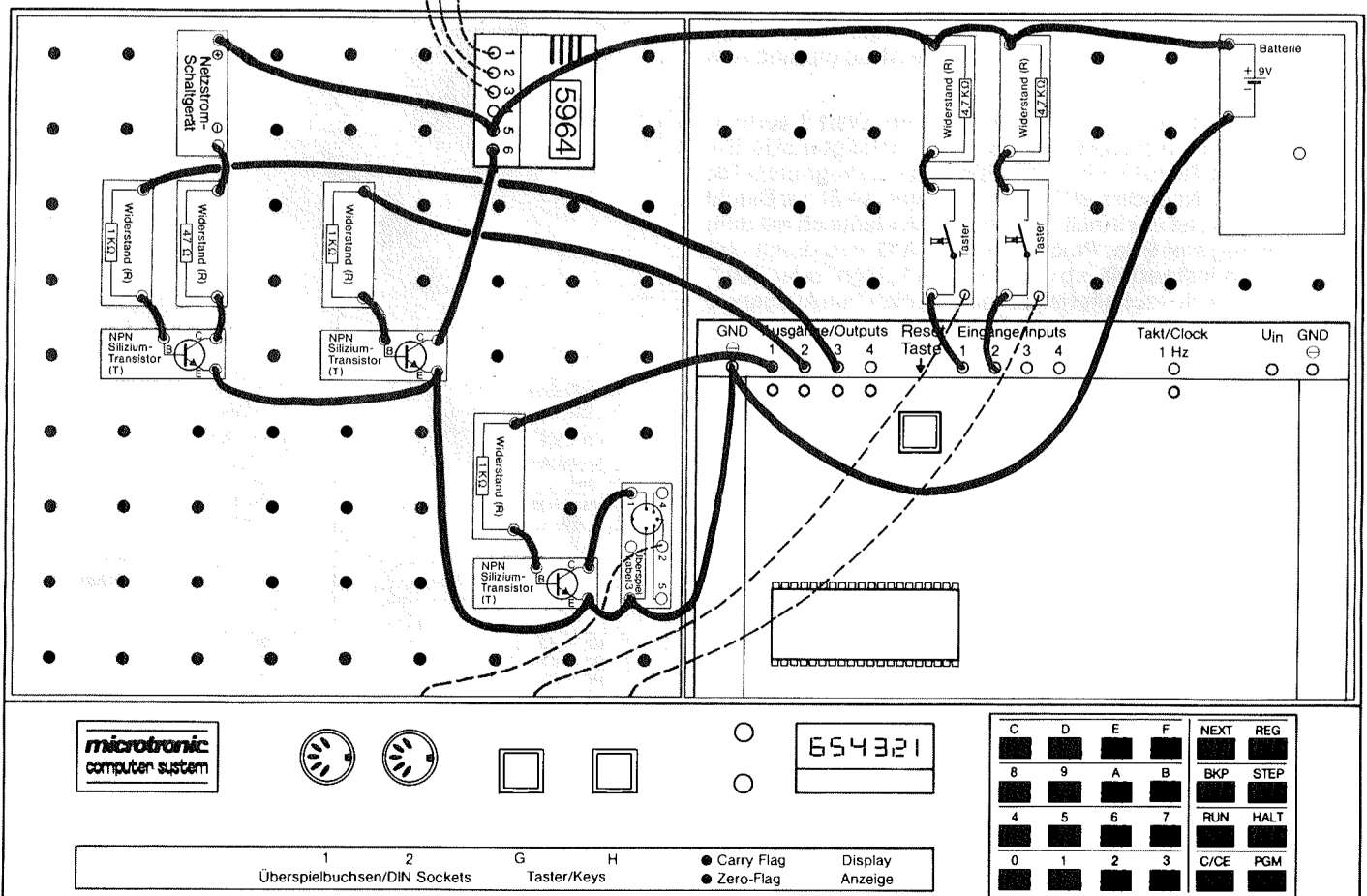
For the „Schaltuhr“ program, all 16 available working registers and the 8 memory registers 0 to 7 are used. The memory registers 0 to 7 hold the switching times for output 1; the working registers 0 to 7 hold the switching times for output 2.

This gives the following register allocation:

Register no.	Register functions	
0	Minutes (ones)	Switch-on time (working or memory register)
1	Minutes (tens)	
2	Hours (ones)	
3	Hours (tens)	
4	Minutes (ones)	Switch-off time (working or memory register)
5	Minutes (tens)	
6	Hours (ones)	
7	Hours (tens)	
8	Input register (for keypad and inputs)	
9	Output register (bit pattern for outputs)	
A	Seconds (ones)	Time of day (TIME command)
B	Seconds (tens)	
C	Minutes (ones)	
D	Minutes (tens)	
E	Hours (ones)	
F	Hours (tens)	

Since the seconds are not shown in timer-clock operation, register A can also be used for further calculations.

Connecting relays etc. to the „Schaltuhr“



The functions follow from the flow chart. At **START** (first column) we find the usual input routine. The abbreviations used are:

EIN 1 = first switch-on time
AUS 1 = first switch-off time
EIN 2 = second switch-on time
AUS 2 = second switch-off time
S UHR = timer clock
UHR = clock operation without switching

The switching times are entered by means of the two subroutines:

SUB-routine **S EIN** = setting the switch-on time
 SUB-routine **S AUS** = setting the switch-off time

In program part 3, the clock runs without switching capability. The SHL command (shift left) removes the clock pulse present at input 4 from register 8, so that no jump back to **START** occurs.

The timer-clock function is in program part 2, **S UHR**. The current time is compared with the entered switching times in the subroutines **V EIN** (compare switch-on time) and **V AUS** (compare switch-off time). There the carry flag is set (value 1) while a switching time does not match the current time, and reset (value 0) when it matches. Querying the carry flag, the main program switches the individual outputs on or off via the AND/OR commands.

In program part 2, **S UHR**, it is checked whether one of the two red buttons was pressed, and the outputs are switched accordingly. As long as neither red button G nor H is pressed, register 8 holds 0 and the following program part is skipped.

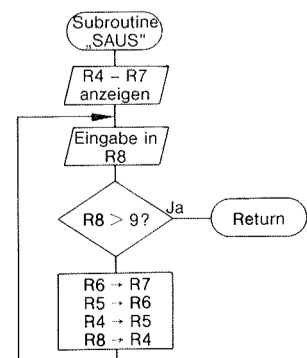
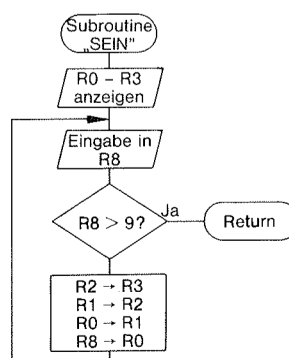
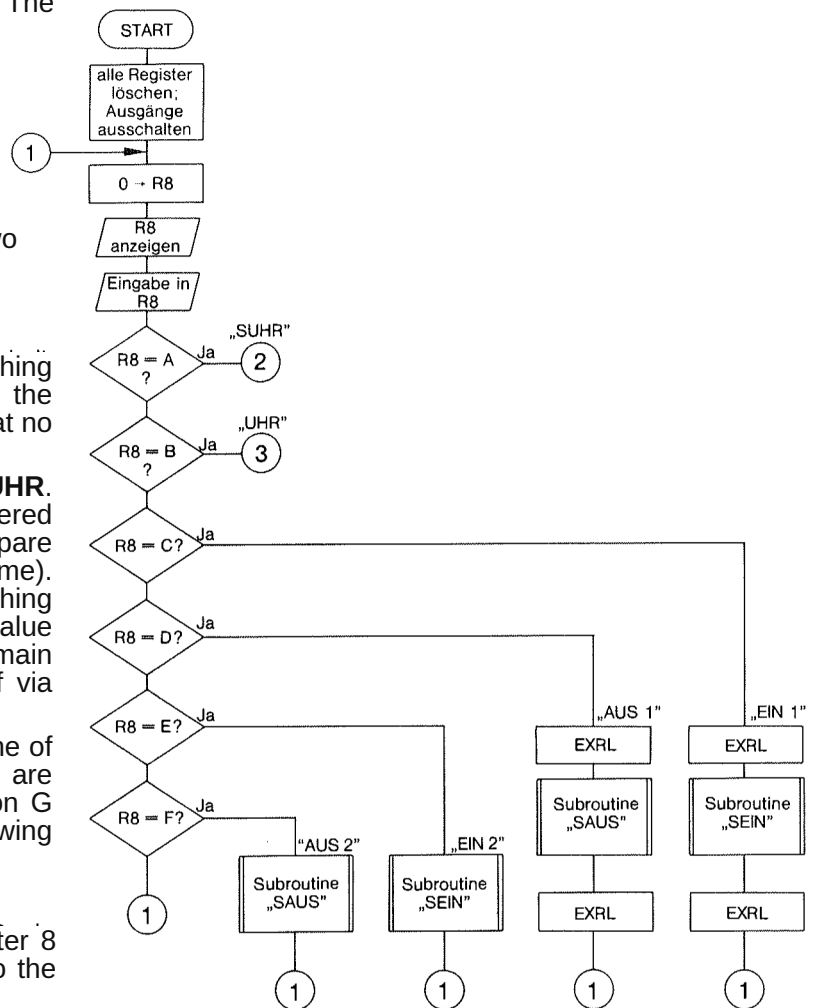
If, however, both red buttons were pressed, register 8 has the value 3 (binary: 0011) and a jump back to the program part **START** occurs.

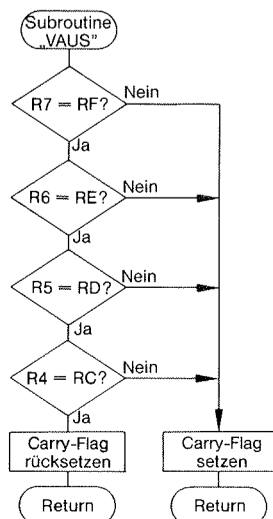
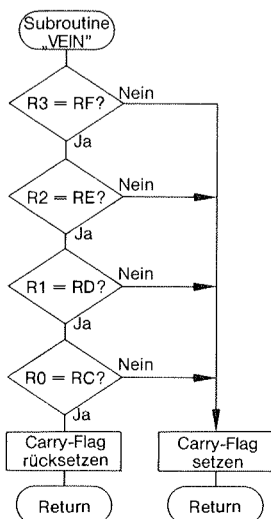
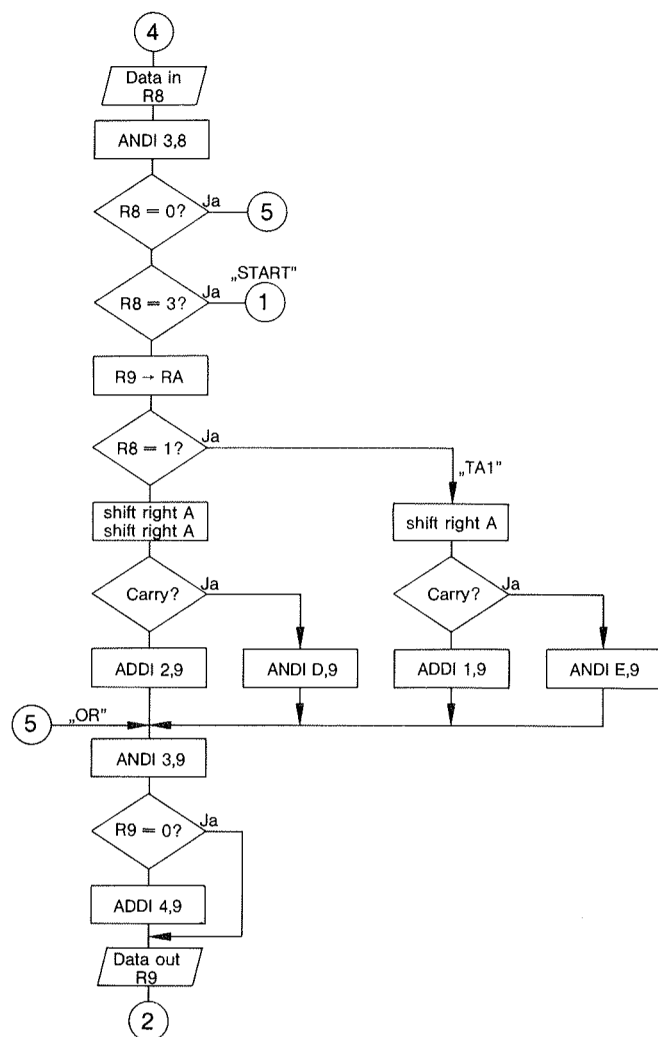
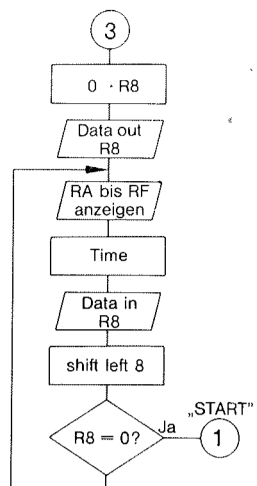
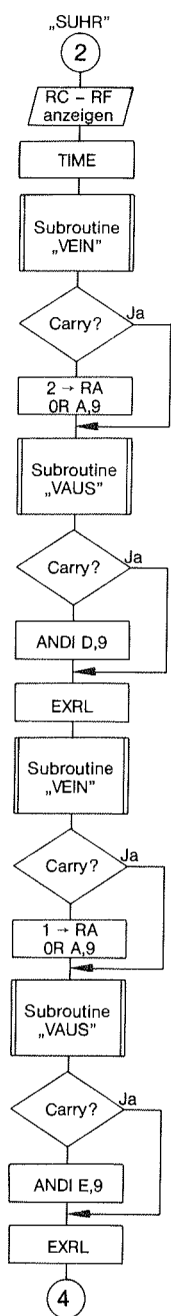
The next comparison, „Is the value 1 present in register 8?“, determines whether the red button G or H was pressed. When button G is pressed, a jump to program part **TA G** (button G) occurs. Since the command MOV 9,A occurs earlier at address 51, the content of register A is identical to the content of register 9. In program part **TA G**, the command SHR A determines whether output 1 is „high“ or „low“, i.e. switched on or off. With a „high“ at output 1 (binary: 0001), the SHR command (shift right) sets the carry flag and the command ANDI E,9 (binary: 1110) switches the output off. If, however, the output was „low“, it is switched on by the command ADDI 1,9 (binary: 0001).

The same applies in principle to program part **TA H** (button H).

In program part **OR**, it is determined whether one of the two outputs no. 1 or 2 is „high“, i.e. switched on. If outputs no. 1 or 2 are switched on, output 3 is also switched on.

Program flow chart: Timer clock





The computer makes music

(BUSCH Electronic Studio no. 2060 or 2065 or 2070 required).

A mini-organ

After HALT – NEXT – 00 we enter the short program:

Mini-organ: long-tone version

Address	Code	Mnemonic
00	F10	DISP 10
01	FF0	KIN 0
02	FE0	DOT 0
03	C00	GOTO 00

Program start: HALT – NEXT – 00 – RUN. The display shows 0. Now briefly press the number key 0. Then build the small additional electronic circuit as shown in the figure.

Owners of an Electronic Studio 2060 or 2065 build the circuit directly in the computer housing. Owners of the larger Electronic Studio 2070 build the circuit in the studio housing and run the corresponding connecting cables to computer outputs no. 1 to 4. Since both the Studio Center 2070 and the computer have dubbing sockets, the lines intended for the computer outputs can first also be routed to the „Überspielkabel“ (patch-cable) module. In this case the „Überspielkabel“ module is also used on the computer, and from there the connecting lines to the computer outputs are made. Care must then be taken to use the same numbering for the connections on both „Überspielkabel“ modules. The two devices can now be connected to each other with a standard patch cable.

If we now press number key no. 4 on the computer, a tone sounds. Entering the number 5 gives a higher tone. If we press the number and letter keys one after another, 12 tones are available. When key 0 is pressed, tone generation is interrupted until a number or letter key is pressed again.

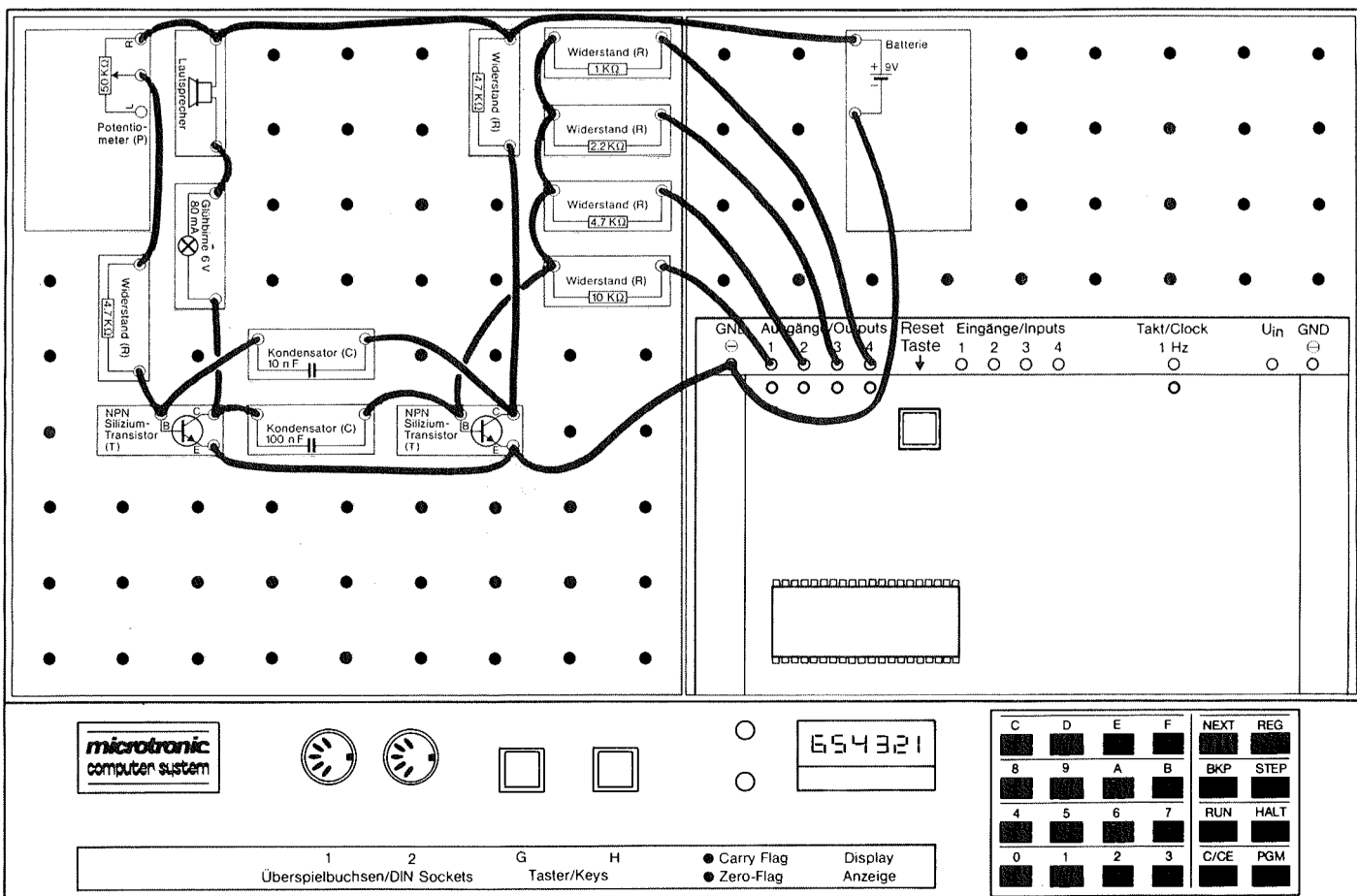
The pitch can be adjusted with the rotary knob of the potentiometer. This is necessary in order to obtain a scale-like arrangement. With keys 1 to 3, no „usable“ tones result.

Our mini-organ has the disadvantage that, in the version now programmed, continuous tones are produced. We can change the programming so that a key press generates only single tones. After HALT – NEXT – 00 we enter the following program:

Mini-organ: short-tone version

Address	Code	Mnemonic
00	F10	DISP 1,0
01	FF0	KIN 0
02	FE0	DOT 0
03	511	ADDI 1,1
04	E06	BRZ 06
05	C03	GOTO 03
06	FE1	DOT 1
07	C00	GOTO 00

Mini-organ from components of the Electronic Studios 2060 or 2065



Program start: HALT – NEXT – 00 – RUN. The display shows 0, and the computer produces a continuous tone until key 0 is pressed. If the various number and letter keys are now pressed, tones result that switch off automatically after a short time. We can shorten the tone sequence further by changing the ADDI command at address 03 from 511 to 521 (or 541, or 581).

This is of interest to the programmer

We already got to know the first program entered on page 16 in Part 1 of the instruction book (binary system). Depending on the key pressed, a binary value is placed on the outputs. The corresponding LEDs at the outputs light up. Outputs with lit LEDs are „high“, i.e. a small voltage of about 3 to 5 volts is present there.

In the additional electronic circuit, transistors T1 and T2 are connected as an astable multivibrator. A base voltage reaches the base of transistor T2 via computer outputs no. 1 to 4, and is varied by the various resistors. This gives a variable tone frequency.

If all outputs are „low“ (LEDs off, display shows 0), transistor T2 lacks the base voltage, so no oscillations can be generated. If key 1 is pressed (LED at output 1 lights up), the 10 kΩ resistor (22 kΩ in the Studio 2070) produces a very small base voltage, not yet sufficient to generate tone oscillations. By pressing the various number and letter keys, however, different resistor combinations arise, through which oscillations are produced in the various tone-frequency ranges.

The sound volume can be improved with the amplifier module of the inexpensive Electronic Studio add-on set „IC-Verstärker-Technik 2072“ (IC amplifier technology 2072). The 2072 instruction book shows how the amplifier module is arranged in such circuits.

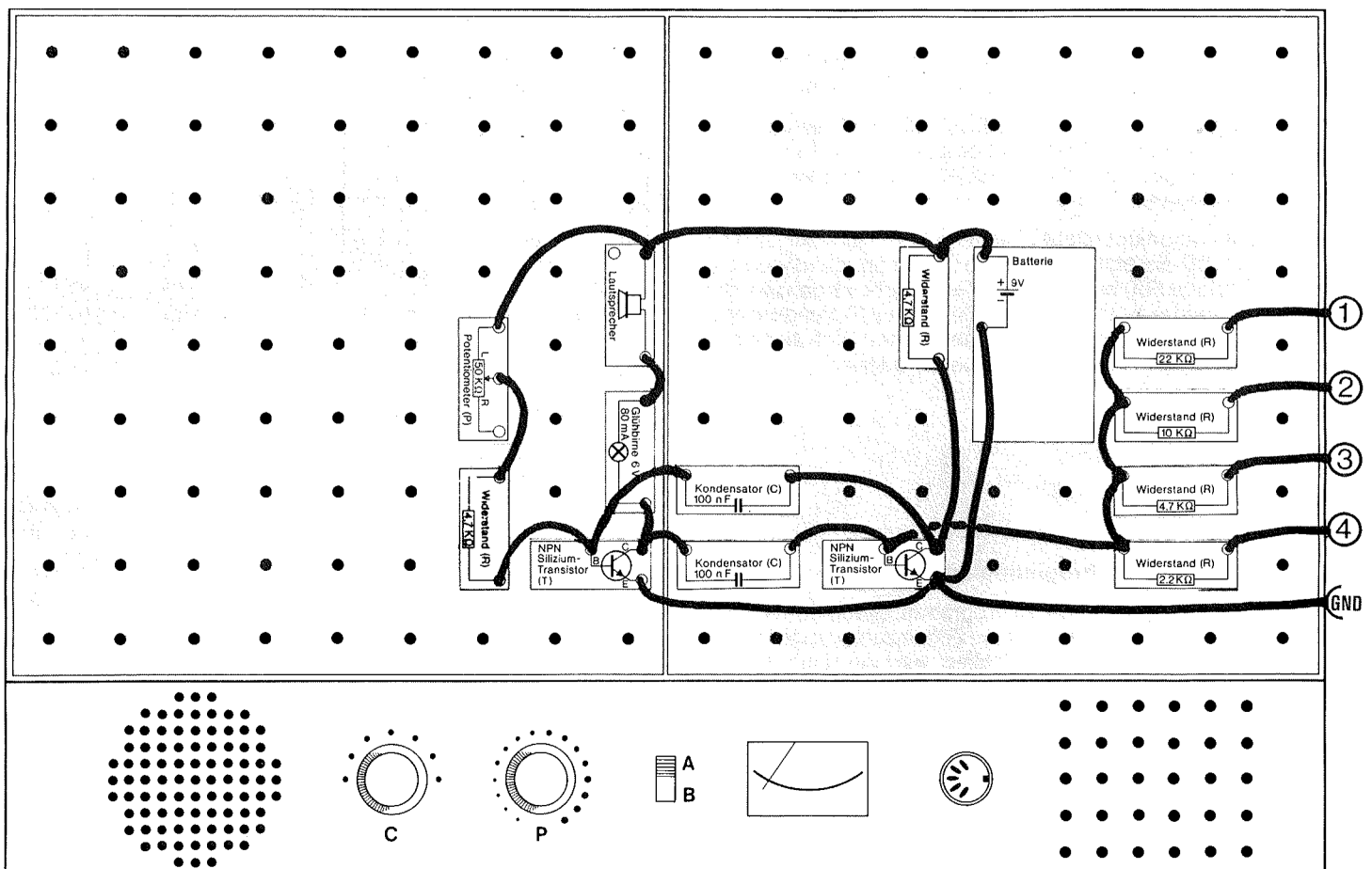
It has already been explained that, for additional electronic circuits, the voltages available at the computer outputs must be amplified — i.e. that an additional voltage source (battery or BUSCH power supply 2059) is required. In all such circuits, the minus pole of the battery must also be connected to the minus pole of the computer. The minus pole of the computer is called „ground“ (GND). On the computer board we find a GND socket both on the left and on the right. For electronic circuits, the left GND socket should always be used. The right GND socket is only for the battery connection, to power the RAM during a mains-power interruption.

The computer has a socket labeled U_{in} . This must never be connected to the minus pole of the battery (danger of short circuit!).

In the second program version (mini-organ with short-tone generation), the ADDI command at address 03 is responsible for the tone length. The constant value 1 is continuously added to register 1 (an automatic counter) until a carry results, which triggers the zero flag. If the ADDI command is changed so that, for example, the constant value 2 is added, the counting process is accelerated and tone generation is shortened.

The experimental setup also remains in place for the following experiment.

Mini-organ built with the Electronic Studio 2070. The four connecting leads go to the computer outputs.



The composing computer (melody generator)

The computer is to compose melodies on its own, with long and short, high and low tones. An interesting experiment, for which the same circuit setup as for the preceding mini-organ is used.

Only a new programming is required. So that the computer's tone generation does not disturb us while programming, one of the connecting leads to the battery is disconnected. After HALT – NEXT – 00, enter the following program:

Program: The composing computer

Address	Code	Mnemonic
00	F08	CLEAR
01	FE0	DOT 0
02	F60	DISP 6,0
03	FF0	KIN 0
04	F05	RND
05	531	ADDI 3,1
06	40D	ADD 0,D
07	E0D	BRZ 0D
08	41D	ADD 1,D
09	E04	BRZ 04
0A	93D	CMPI 3,D
0D	D0D	BRC 0D
0C	57D	ADDI 7,D
0D	FED	DOT 0
0E	71E	SUBI 1,E
0F	E04	BRZ 04
10	C0E	GOTO 0E

Program start with HALT – NEXT – 00 – RUN. The display shows: 000000. Reconnect the connecting lead to the battery. As soon as a number or letter key is pressed, the computer composes random melody sequences.

The program is stopped with HALT – NEXT – 00 – RUN. The melody generator carries on working as soon as a number or a letter is entered. Ever-new melody variations result.

Further variation possibilities arise by changing the DISP command at address 02. We change the entry F60 there to F10. Don't forget the NEXT key, then a new program start. Now completely different melody sequences result. We can change the DISP command several times, and again and again new variations result.

This is of interest to the programmer:

A very easily surveyable program. By means of the RND command, a random number is loaded into registers D to F, and to help chance along a little, the numeric value entered after program start is added into register D.

The value in register D is responsible for the pitch. By counting the random number down in register E, the tone length is produced.

The experimental setup also remains in place for the following experiment.

Computer controls electronic music box

In the preceding experiments we produced tones by pressing keys, or got the computer, with the help of the random generator, to compose melodies on its own.

With this program we want to teach the computer to play „proper melodies“ automatically. So that flawless tone sequences result, the circuit setup with the Electronic Studio 2070 (see page 53) is required. Anyone who owns the smaller Electronic Studio 2060 can obtain the two still-missing components via the BUSCH spare-parts service (see page 53 for this):

Order no. 20669
Order no. 20689

1 disc capacitor 100 nF
1 resistor 22 kΩ

(Please transfer DM 3.50 incl. postage and packing to postal-cheque account Karlsruhe no. 57 614-751, Busch GmbH & Co. KG, and state the order numbers on the payee section).

With the following program the computer, together with the connected electronic circuit, plays the three well-known melodies one after another: „Sentimental Journey – Hänschen klein – Du, Du liegst mir am Herzen“.

Before we begin the programming after HALT – NEXT – 00, the electronic circuit is put out of operation by disconnecting the battery cable.

Program: Music box

Addr.	Code	Mnemonic
00	F02	DISOUT
01	C0B	GOTO 0B

(Subroutine)

02	FE0	DOT 0
03	511	ADDI 1,1
04	E06	BRZ 06
05	C03	GOTO 03
06	902	CMPI 0,2
07	E09	BRZ 09
08	F07	RET
09	FE1	DOT 1
0A	F07	RET

(„Sentimental Journey“)

0B	112	MOVI 1,2
0C	180	MOVI 8,0
0D	B02	CALL 02
0E	140	MOVI 4,0
0F	B02	CALL 02
10	180	MOVI 8,0
11	B02	CALL 02
12	140	MOVI 4,0
13	B02	CALL 02
14	180	MOVI 8,0
15	B02	CALL 02
16	140	MOVI 4,0
17	B02	CALL 02
18	100	MOVI 0,0
19	B02	CALL 02
1A	180	MOVI 8,0
1B	B02	CALL 02

1C	140	MOVI 4,0
1D	B02	CALL 02
1E	180	MOVI 8,0
1F	B02	CALL 02
20	190	MOVI 9,0
21	B02	CALL 02
22	180	MOVI 8,0
23	B02	CALL 02
24	170	MOVI 7,0
25	B02	CALL 02
26	160	MOVI 6,0
27	B02	CALL 02
28	100	MOVI 0,0
29	B02	CALL 02
2A	180	MOVI 8,0
2B	B02	CALL 02
2C	140	MOVI 4,0
2D	B02	CALL 02
2E	180	MOVI 8,0
2F	B02	CALL 02
30	140	MOVI 4,0
31	B02	CALL 02
32	170	MOVI 7,0
33	B02	CALL 02
34	140	MOVI 4,0
35	B02	CALL 02
36	170	MOVI 7,0
37	B02	CALL 02
38	140	MOVI 4,0
39	B02	CALL 02
3A	180	MOVI 8,0
3B	B02	CALL 02

3C	140	MOVI 4,0
3D	B02	CALL 02
3E	100	MOVI 0,0
3F	B02	CALL 02
40	180	MOVI 8,0
41	B02	CALL 02
42	190	MOVI 9,0
43	B02	CALL 02
44	180	MOVI 8,0
45	B02	CALL 02
46	170	MOVI 7,0
47	B02	CALL 02
48	160	MOVI 6,0
49	B02	CALL 02
4A	140	MOVI 4,0
4B	B02	CALL 02
4C	100	MOVI 0,0
4D	B02	CALL 02

71	1A0	MOVI A,0
72	B02	CALL 02
73	160	MOVI 6,0
74	B02	CALL 02
75	160	MOVI 6,0
76	B02	CALL 02
77	140	MOVI 4,0
78	B02	CALL 02
79	180	MOVI 8,0
7A	B02	CALL 02
7B	1C0	MOVI C,0
7C	B02	CALL 02
7D	1C0	MOVI C,0
7E	B02	CALL 02
7F	140	MOVI 4,0
80	B02	CALL 02
81	100	MOVI 0,0
82	B02	CALL 02

A6	102	MOVI 0,2
A7	B02	CALL 02
A8	112	MOVI 1,2
A9	B02	CALL 02
AA	102	MOVI 0,2
AB	B02	CALL 02
AC	112	MOVI 1,2
AD	B02	CALL 02
AE	102	MOVI 0,2
AF	B02	CALL 02
B0	B02	CALL 02
B1	140	MOVI 4,0
B2	B02	CALL 02
B3	160	MOVI 6,0
B4	112	MOVI 1,2
B5	B02	CALL 02
B6	190	MOVI 9,0
B7	B02	CALL 02
B8	B02	CALL 02
B9	170	MOVI 7,0
BA	B02	CALL 02
BB	102	MOVI 0,2

BC	B02	CALL 02
BD	140	MOVI 4,0
BE	B02	CALL 02
BF	160	MOVI 6,0
C0	B02	CALL 02
C1	170	MOVI 7,0
C2	B02	CALL 02
C3	1C0	MOVI C,0
C4	B02	CALL 02
C5	190	MOVI 9,0
C6	B02	CALL 02
C7	170	MOVI 7,0
C8	B02	CALL 02
C9	160	MOVI 6,0
CA	B02	CALL 02
CB	112	MOVI 1,2
CC	140	MOVI 4,0
CD	B02	CALL 02
CE	B02	CALL 02
CF	B02	CALL 02
D0	100	MOVI 0,0
D1	B02	CALL 02

(„Hänschen Klein“)

4E	102	MOVI 0,2
4F	1C0	MOVI C,0
50	B02	CALL 02
51	180	MOVI 8,0
52	B02	CALL 02
53	180	MOVI 8,0
54	B02	CALL 02
55	1A0	MOVI A,0
56	B02	CALL 02
57	160	MOVI 6,0
58	B02	CALL 02
59	160	MOVI 6,0
5A	B02	CALL 02
5B	140	MOVI 4,0
5C	B02	CALL 02
5D	160	MOVI 6,0
5E	B02	CALL 02
5F	180	MOVI 8,0
60	B02	CALL 02
61	1A0	MOVI A,0
62	B02	CALL 02
63	1C0	MOVI C,0
64	B02	CALL 02
65	1C0	MOVI C,0
66	B02	CALL 02
67	1C0	MOVI C,0
68	B02	CALL 02
69	100	MOVI 0,0
6A	B02	CALL 02
6B	1C0	MOVI C,0
6C	B02	CALL 02
6D	180	MOVI 8,0
6E	B02	CALL 02
6F	180	MOVI 8,0
70	B02	CALL 02

(„Du liegst mir am Herzen“)

83	112	MOVI 1,2
84	160	MOVI 6,0
85	B02	CALL 02
86	102	MOVI 0,2
87	B02	CALL 02
88	B02	CALL 02
89	102	MOVI 0,2
8A	B02	CALL 02
8B	B02	CALL 02
8C	140	MOVI 4,0
8D	B02	CALL 02
8E	160	MOVI 6,0
8F	B02	CALL 02
90	112	MOVI 1,2
91	190	MOVI 9,0
92	B02	CALL 02
93	B02	CALL 02
94	170	MOVI 7,0
95	B02	CALL 02
96	140	MOVI 4,0
97	B02	CALL 02
98	102	MOVI 0,2
99	B02	CALL 02
9A	112	MOVI 1,2
9B	B02	CALL 02
9C	102	MOVI 0,2
9D	B02	CALL 02
9E	B02	CALL 02
9F	190	MOVI 9,0
A0	B02	CALL 02
A1	170	MOVI 7,0
A2	B02	CALL 02
A3	112	MOVI 1,2
A4	160	MOVI 6,0
A5	B02	CALL 02

Reconnect the battery cable to the electronic circuit. Program start with HALT – NEXT – 00 – RUN.

The computer immediately begins to play the 3 melodies one after another, repeating them until the green RESET button on the computer board is pressed. A new program start is made as usual with HALT – NEXT – 00 – RUN.

With the rotary knob of the potentiometer, the pitch can be trimmed a little if the melodies are not played in the correct key.

If the owners of the Electronic Studios 2060 or 2065 do not have the two aforementioned components sent to them, this experiment can still be carried out; however, wrong tone sequences must then be accepted in some places.

This is of interest to the programmer:

Through hexadecimal entries, binary values are placed on the outputs, whereby the desired „high“ and „low“ states arise there. Depending on which outputs — and how many of the outputs — are „high“, the various pitches are played.

To get the computer to play particular melodies, the required tones must be produced in the correct order by the corresponding binary values at the outputs. We must know which resistor combination of our circuit produces the desired tone. The programming of the tone sequences follows accordingly.

Since the 3 melodies consist of a whole host of individual tones, a longer program is required, because every single tone has to be programmed.

Nevertheless, even without a program flow chart, a simple and clear presentation results, because the entire „tone program“ is carried out with pitch in register 0 and tone length in register 2.

Acoustic effects for computer games:

The MOVI commands bring the hexadecimal values for the binary representation at the outputs into register 0. The CALL commands cause a jump into the subroutine (from address 02 to 0A), whereby the corresponding outputs are activated for a short time.

The MOVI commands also take care of the tone lengths, by bringing either the value 0 (short tones) or the value 1 (long tones) into register 2. If register 2 has the content 1, a continuous tone results, which changes only with the next following tone. If register 2 holds the value 0, a short tone arises — i.e. a short pause results between two tones. The melody „Sentimental Journey“ begins at address 0B. The command MOVI 1,2 brings the value 1 into register 2, i.e. all following tone commands produce continuous tones. With the following command MOVI 8,0, the value 8 is brought into register 0 (binary: 1000), i.e. output 4 becomes „high“ and the corresponding tone is produced by the electronic circuit. Each MOVI command is followed by a CALL command. Through this constant jumping back and forth between the „melody program“ and the „subroutine“, the melody sequence results.

The melody „Hänschen klein“ results from address 4E onward, where the command MOVI 0,2 brings the value 0 into register 2, so that all following tones are „played short“.

In the melody „Du, Du liegst mir am Herzen“, registers 0 and 2 are addressed alternately, because both long and short tones are required for this melody.

It should not be difficult for us to program the computer ourselves with a melody that we particularly like.

The „computer games“ offered in shops are often bought because of their acoustic effects. Many a game can be made more interesting through an acoustic representation of the score reached.

The game „The Moon Landing“ (page 23, instruction book Part 1) becomes considerably more effective through acoustic signals at the end of the game.

Let's try it out!

We enter the Moon Landing program once again. Apart from a few commands, the program can be taken over from page 24:

At address 56 (FE6), the change to CA5 is required. At address 6D (1FF) it is changed to C99. At address 7F (C6D) it is replaced by C90, and address 84 (C6D) by C9F.

Then the program is extended from address 90 onward. We carry out the additional reprogramming after HALT – NEXT – 90:

Program addition:

Acoustic effects for „Moon Landing“

Address	Code	Mnemonic	Explanations
90	11D	MOVI 1,D	Tone generation for „lunar module crashes“
91	51E	ADDI 1,E	
92	FBD	ADC D	
93	E73	BRZ 73	
94	FEE	DOT E	
95	0EF	MOV E,F	
96	F8F	INV F	
97	FEF	DOT F	
98	C91	GOTO 91	Tone generation for „fuel used up“
99	11D	MOVI 1,D	
9A	71E	SUBI 1,E	
9B	FBD	ADC D	
9C	E73	BRZ 73	
9D	FEE	DOT E	
9E	C9A	GOTO 9A	
9F	11D	MOVI 1,D	Tone generation for „lunar module drifts off“
A0	51E	ADDI 1,E	
A1	FBD	ADC D	
A2	E73	BRZ 73	
A3	FEE	DOT E	
A4	CA0	GOTO A0	
A5	10D	MOVI 0,D	Tone generation for „soft moon landing“
A6	14E	MOVI 4,E	
A7	FEE	DOT E	
A8	1AE	MOVI A,E	
A9	FEE	DOT E	
AA	54D	ADDI 4,D	
AB	FBC	ADC D	
AC	DAE	BRC AE	
AD	CA6	GOTO A6	
AE	10E	MOVI 0,E	
AF	FEE	DOT E	
B0	C57	GOTO 57	

The computer counts people and objects

(BUSCH Electronic Studio 2065 or 2070 required)

The tone-generating circuit from the preceding experiments is also used unchanged for this experiment.

After HALT – NEXT – 00 – RUN, the program is put into operation in the same way as described on page 23 of instruction book Part 1. At the end of the game, additional acoustic effects now result, which are matched to the game result by a corresponding tone sequence:

Soft landing of the lunar module = melodic tone sequence of high and low tones.

Lunar module has smashed on the lunar surface = simultaneously rising and falling melody sequence.

Fuel supply used up = repeatedly falling tone sequence.

Lunar module vanishes into space = rapidly rising tone sequence.

The game „Moon Landing“ is now perfect. The short additional program shows how such games can be made even more interesting through acoustic effects. The individual tone-program parts can also be adopted afterward for other game programs. (Note the command changes for the jumps to the subroutine!)

Where there is light, there is also shadow! Through this effect, counting effects can be achieved with light barriers, in that an illuminated photoresistor passes on the change between light and dark to the computer input by changing the voltage.

We build the electronic circuit according to the figure. The photoresistor should be illuminated by daylight or by the light of a lamp. By slowly turning the potentiometer knob, the circuit is adjusted so that the light bulb just goes out. If the photoresistor is then darkened somewhat — e.g. by the shadow of a hand held over it — the little bulb must light up.

After the electronic circuit has been connected to the computer, we carry out the programming after HALT – NEXT – 00:

Program: Light-barrier controlled counter

Address	Code	Mnemonic	Explanations
00	F08	CLEAR	Clear all registers
01	F21	DISP 2,1	Show registers 1 and 2
02	FD0	DIN 0	Store input value in Reg. 0
03	330	ANDI 3,0	Clear value at inputs 3 and 4
04	803	CMP 0,3	Reg. 0 = Reg. 3?
05	E02	BRZ 02	Yes, then jump to 02
06	003	MOV 0,3	Reg 0 → Reg. 3
07	910	CMPI 1,0	Reg 0 = 1?
08	E02	BRZ 02	Yes, jump to 02
09	D00	BRC 00	Greater than 1, jump to 00
0A	511	ADDI 1,1	Decimal counting program in registers 1 and 2
0B	991	CMPI 9,1	similar to page 25 (described from address 08 on)
0C	D0E	BRC 0E	
0D	C02	GOTO 02	
0E	561	ADDI 6,1	
0F	512	ADDI 1,2	
10	992	CMPI 9,2	
11	D00	BRC 00	
12	C02	GOTO 02	

Program start: HALT – NEXT – 00 – RUN. The display shows 00.

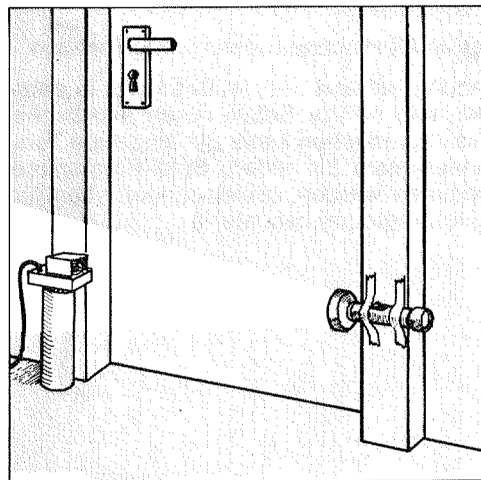
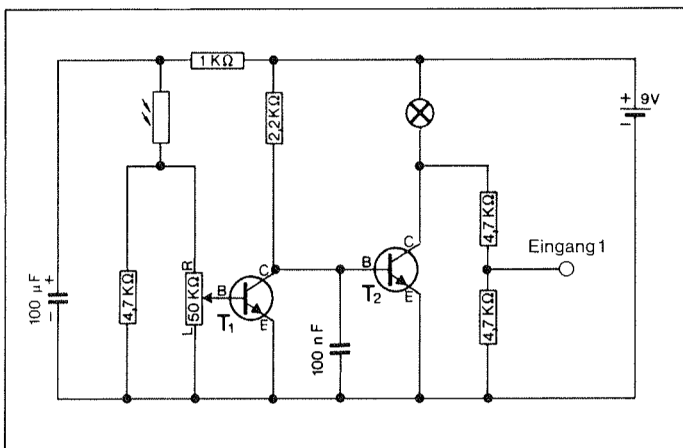
As soon as we bring a shadow onto the photoresistor (e.g. with the hand), this light–dark change is counted. The display shows 01 (i.e. how often a shadow was registered by the photoresistor). With the red button H, the counter can be reset to 0 again.

If the connecting cable is extended, the photoresistor can be mounted, for example, on a door. The computer then counts how many people have entered a room through this door.

With a focused beam of light, objects can be counted, e.g. on a conveyor belt. The circuit can also be used as a lap counter for model railways or electric slot-car tracks, and so on, and so on.

If two light barriers are mounted next to each other — e.g. at the entrance of a building — it could be determined in the evening how many people have entered and left the building again. A compare command could check on which of the two light barriers a shadow was registered first, whereby the first shadow triggers an up-counting effect and the second shadow a down-counting effect. If a difference appears on the display, it would be possible to determine, for example, that a certain number of people are still in the building.

With a slot-car racing track, for instance, the possibility arises — by incorporating the BUSCH relay 5964 — to automatically switch off the driving current for the cars at a certain preset number of laps, etc.



Principle of data transmission

(BUSCH Digital Studio 2075 required)

This is of interest to the programmer:

The circuit setup of the light barrier is familiar to us from instruction book 2065 or 2070. In addition, only the two 4.7 kΩ resistors have been inserted, in order to achieve the lowest possible current consumption of the battery.

By means of the command DIN 0, the input value is stored in register 0. With ANDI 3,0, the third and fourth binary digit (i.e. at inputs 3 and 4) are set to 0, so that, for example, a clock connected to input 4 (if we happen to be running the clock program alongside) does not destroy the subsequent comparisons.

Then a comparison is made as to whether the same value is present in registers 0 and 3. If „yes“, the input value has not changed and a jump back to address 02 occurs. If „no“, the input value is stored in registers 0 and 3. By means of the compare command CMPI it is determined: the value 1 is in register 0, i.e. the light barrier was not interrupted, and a jump to address 02 occurs. If the register content is greater than 1, the red button H was pressed, and a jump to address 00 occurs, to clear all registers. Otherwise, register 0 gives the value 0, i.e. the light barrier was interrupted. In the following decimal counter program, the interruption is counted as value 1.

Why does an interrupted light barrier give the value 0 in register 0?

The electronic circuit is designed so that, with an uninterrupted light barrier, a small voltage is present at input 1 of the computer — the input is „high“. With an interrupted light barrier (when the lamp on the circuit lights up), no voltage is present at the computer input. This gives a „low“, i.e. the value 0, which is stored in register 0.

For the many interesting experiments with electronic peripheral circuits, appropriate basic knowledge is a prerequisite. The BUSCH Electronic Studios explain a broad palette of electronic possibilities which, in conjunction with the computer, lead to ever-new applications.

In practice, it is often necessary to transmit data from one data-processing system (EDV) to another.

With the IC counter module from the Digital Studio 2075, we can make a register content of our computer visible on the large 7-segment display of the counter module. There are two possibilities for data transmission:

Data transmission with only three lines

We enter the program as usual:

Address	Code	Mnemonic
00	F08	CLEAR
01	132	MOVI 3,2
02	123	MOVI 2,3
03	114	MOVI 1,4
04	F10	DISP 1,0
05	FF0	KIN 0
06	101	MOVI 0,1
07	FE4	DOT 4
08	FE2	DOT 2
09	801	CMP 0,1
0A	E04	BRZ 04
0B	511	ADDI 1,1
0C	FE3	DOT 3
0D	C08	GOTO 08

The IC counter module is connected to the computer as shown in the figure.

After the program start, the display shows 0. If a number is entered, it is shown both on the computer display and on the IC counter module.

As long as we enter only numbers, these numbers are also visible on the large 7-segment display. If we enter, for example, C, the number 2 appears on the large 7-segment display.

The IC counter module works only in the decimal range. The hexadecimal C corresponds to the decimal number 12. Since the IC counter module displays only single-digit results, only the last digit — namely the digit 2 — is shown.

If we had several IC counter modules, it would be readily possible to transmit multi-digit results as well. How several IC counter modules are connected can be seen in instruction book 2075.

This is of interest to the programmer:

From instruction book 2075 we know that the IC counter module can „only“ count. The counter module always counts up by one as soon as the voltage at its clock input changes from „high“ to „low“. The counter module can be reset to zero as soon as its RESET input becomes „low“.

So that the IC counter module displays the same number that is entered into the computer, a corresponding number of „high“ – „low“ – „high“ pulses are passed one after another to the IC counter module, which are then

counted by it until the result is displayed. The number of pulses is also counted internally by the computer in register no. 1, so that the computer knows what count the IC counter module has.

By means of the entered program, the value 3 (binary: 0011) is first stored in register 2. In register 3 the value 2 (binary: 0010) and in register 4 the value 1 (binary: 0001) are stored. If a numeric value is now entered into register 0 via the keyboard, register 1 is set to the value 0, and by means of the commands DOT 4 (0001) and DOT 2 (0011) the IC counter module is also set to 0.

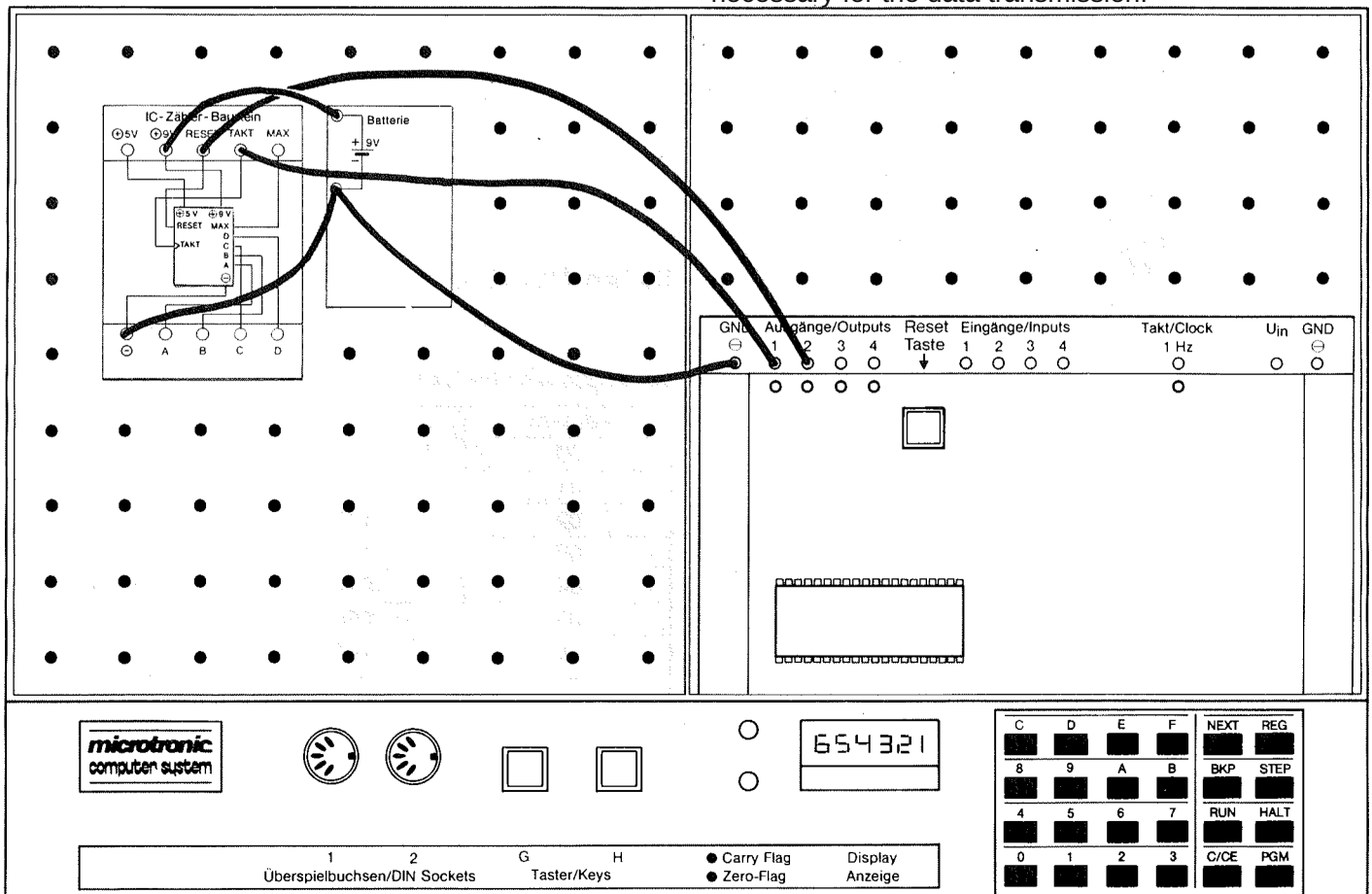
Now the comparison follows as to whether the content of register 0 is identical to the content of register 1. If „yes“, then the value shown by the counter module is identical to the entered value. If „no“, the value 1 is added to register 1, and by means of the command DOT 3 (content of register 3 = binary 0010) output 1 is set to zero. By means of the change of the „high“ – „low“ state, the counter module counts up by one, and by means of the command DOT 2 (0011) output 1 becomes „high“ again. This process is continued until the contents of registers 0 and 1 are equal.

This method has two disadvantages.

If, for example, the line connection to the IC module is interrupted by a fault, a wrong result display arises there.

Or if, for example, the value 9 is entered at the computer, the IC counter module begins to count up from 0 to the entered value, which requires a certain amount of time. The advantage of the method, however, lies in the fact that only three lines are necessary for the data transmission.

Data transmission with 3 lines



Data transmission by the „hand-shaking“ method

The following program serves to demonstrate this method; it is entered as usual:

Address	Code	Mnemonic
00	F08	CLEAR
01	132	MOVI 3,2
02	123	MOVI 2,3
03	F10	DISP 1,0
04	FF0	KIN 0
05	FD1	DIN 1
06	801	CMP 0,1
07	E04	BRZ 04
08	FE3	DOT 3
09	FE2	DOT 2
0A	C05	GOTO 05

The IC counter module is connected to the computer as shown in the figure. We now need twice the number of transmission lines compared to the previous experiment.

Start the program and enter a number. Compared to the previous experiment, it is displayed considerably faster on the IC counter module. If we now enter a letter, the counter module is „at a loss“ — it counts on infinitely, because it cannot display letters.

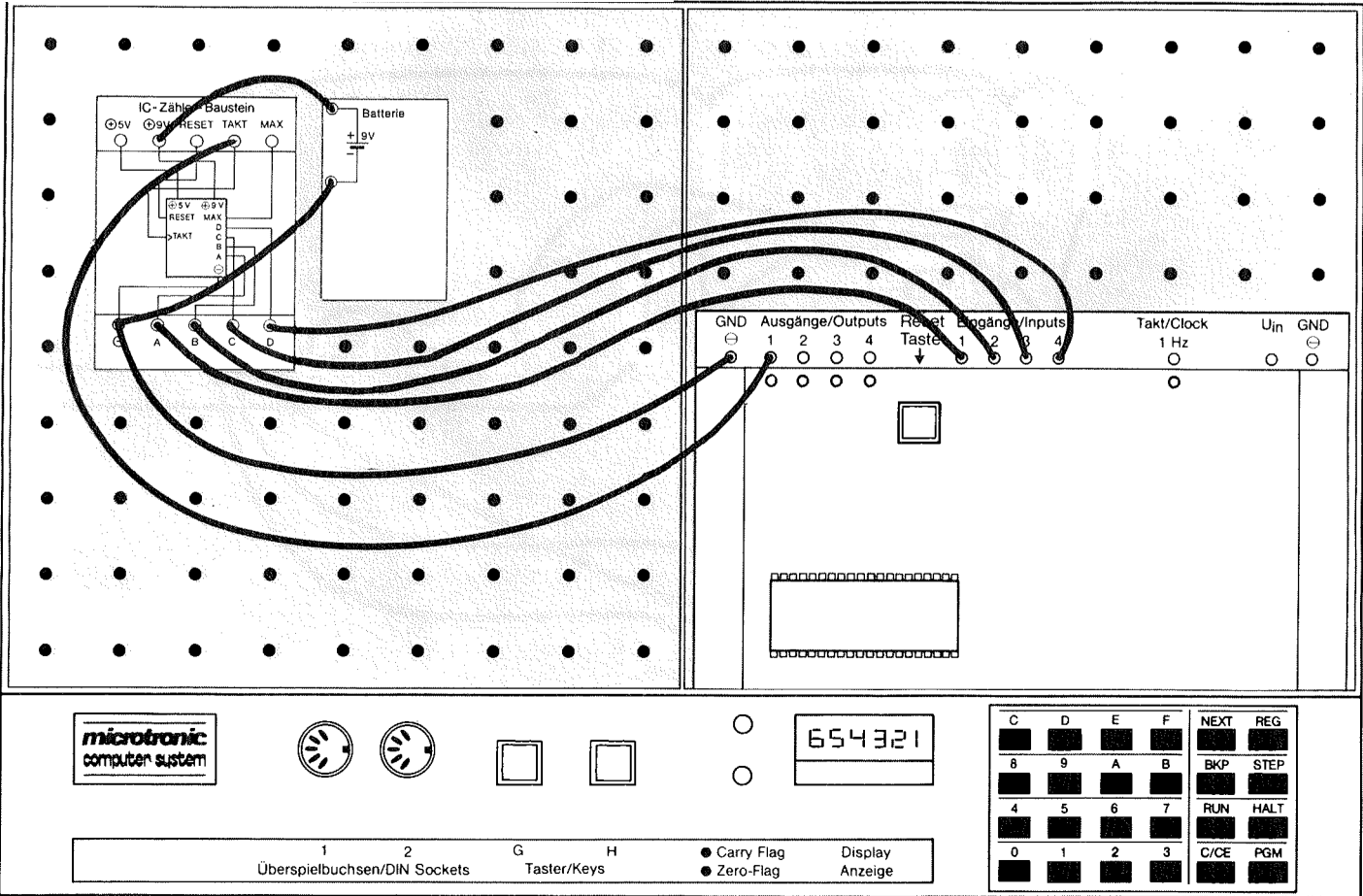
This is of interest to the programmer:

We see that a line runs from computer output no. 1 to the clock input of the counter module. The four outputs of the counter module (A, B, C, D) are connected to the four computer inputs. The four outputs of the counter module pass on the displayed number as binary values. These are taken into register 1 of the computer by means of the command DIN 1. The entered program compares this number with the number entered via the keyboard (in register 0), and if both are equal, the result has been reached. Otherwise, by means of the commands DOT 3 and DOT 2, the IC counter is incremented by one and the comparison is repeated until the result is reached. Thus a feedback from the IC counter to the computer results, by which a wrong data transmission is ruled out.

The experiments show that, for example, ICs can be connected directly to the computer inputs and outputs. Since the ICs work with low voltages, no additional amplifier circuits are required for the control.

Incidentally, our IC counter module too has a carry signal (a kind of carry flag) for controlling a second counter module. The output MAX is constantly „high“ and only becomes „low“ when the value 9 is reached, in order to become „high“ again on the transition to 0. With this carry signal, a second counter module — or several — can be controlled.

Data transmission by „hand-shaking“



Computer counts frequencies

(BUSCH Electronic Studio 2065 or 2070 required).

Frequency measuring instruments are needed in electronics for checking oscillation-generating circuits. With them, one can determine, for example, the transmitting frequency of a radio transmitter, or the audio frequency of an astable multivibrator.

The measuring principle is the same for all frequency counters. During a defined time (e.g. one second), the number of frequency pulses is counted and evaluated. Our experiment works on this principle too.

Program: Frequency counter

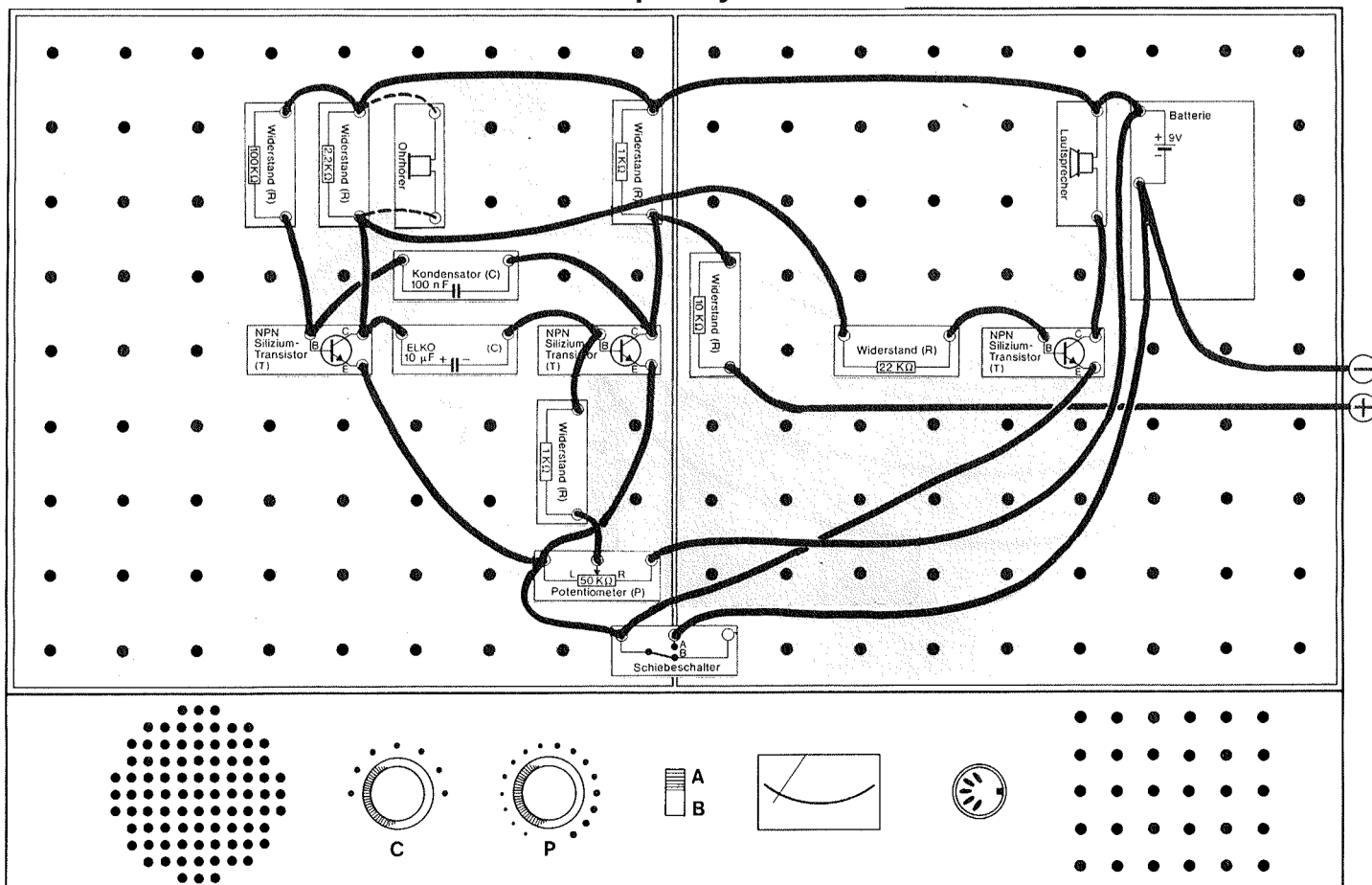
00	F08	CLEAR
01	F28	DISP 2,8
02	FDF	DIN F
03	FAF	SHL F
04	E02	BRZ 02
05	FDF	DIN F
06	FAF	SHL F
07	E09	BRZ 09
08	C05	GOTO 05
09	F06	TIME
0A	F0F	EXTRA
0B	FDF	DIN F
0C	FAF	SHL F

0D	E0B	BRZ 0B
0E	FDF	DIN F
0F	FAF	SHL F
10	E12	BRZ 12
11	C0E	GOTO 0E
12	F06	TIME
13	62A	SUB 2,A
14	D16	BRC 16
15	C18	GOTO 18
16	76A	SUBI 6,A
17	513	ADDI 1,3
18	63B	SUB 3,B
19	D1B	BRC 1B
1A	C1C	GOTO 1C
1B	7AB	SUBI A,B

Before we start the program, the circuit is to be built up as shown in the figure. The connecting lead marked with $\oplus$ is connected to INPUT 4 of the computer. The lead marked with $\ominus$ leads to the GND socket of the computer.

On the computer board, TAKT/CLOCK is to be connected to INPUT 1.

Circuit of the astable multivibrator for frequency measurement



The assembly plan (figure) is authoritative when the large Electronic Studio 2070 is available. If only the Electronic Studio 2065 is present, the 3 components shown directly to the left of the battery are omitted (loudspeaker, transistor and 22 k Ω resistor). In this case, the earphone is to be used (drawn in the plan with a dotted line). The connecting lead between the two 22 k Ω resistors is omitted, and the connecting lead from the plus pole of the battery leads directly to the 1 k Ω resistor (top center).

Start the program and put the electronic circuit into operation (slide switch in position A). As long as the potentiometer knob is at the far left stop, no tone is produced in the loudspeaker (or earphone). The computer display shows 00.

Now we turn the potentiometer knob a little to the right until a slow crackling becomes audible in the loudspeaker (earphone). The computer display shows the frequency of the crackling tone in hertz (oscillations per second). If the potentiometer is turned further to the right, the tone sequence becomes faster — the frequency increases, which is also shown on the display by a higher hertz value.

The potentiometer should always be turned only a small step further, because the computer needs a measuring time of about 2 seconds until the result is displayed.

The further the potentiometer knob is turned to the right, the faster the tone sequence and the higher the frequency display. We will reach a point at which the computer no longer counts along (or shows fluctuating results), although, due to the faster oscillations, the pitch rises and therefore the frequency display ought to be higher.

The frequency generated by our circuit has now become too large (on the display this will be between 30 and 50).

For a following experiment, we should note up to which frequency reading the experiment works. The accuracy of the experiment that follows later depends on as precise a value as possible.

The limit frequency is determined by turning the potentiometer only a very small step further at the critical point, while observing the display. At a certain setting, the displayed value will fluctuate strongly. The potentiometer setting is now reduced just enough that the displayed value no longer changes by more than 3 Hz. We record this value in the following table.

The limit frequency was determined as: Hz.

This is of interest to the programmer:

It is not possible to determine the frequencies by means of a simple counting program in which the pulses at an input are counted. Our computer works too slowly for high frequencies.

Already at 3 to 4 pulses (oscillations) per second, the computer can no longer count along. This is because our microprocessor is programmed for very versatile applications. It does not only have to count; it monitors the keyboard, it handles the clock program running on in the background, and, and, and...

By means of a simple trick, we can outwit the computer: in the operating system, the time of day runs continuously. This clock program too is a „counter“, which counts the clock pulses at INPUT 4 and evaluates them for the time of day.

This „clock counter“ in the operating system works relatively fast. Therefore, the pulses to be measured are connected to INPUT 4. There, the pulses are counted along and further processed into a kind of „time of day“.

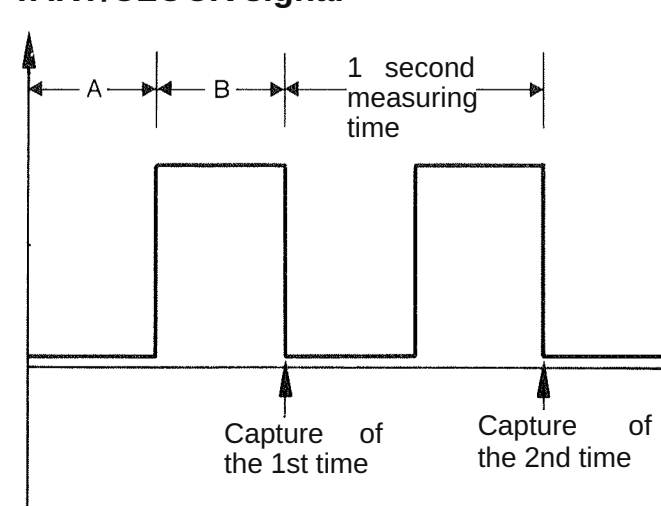
This process can easily be checked by calling the clock program with HALT – PGM – 4 and, at the same time, generating a tone with the connected circuit. We can observe that, depending on the clock frequency of the circuit, the seconds display on the display is shown at considerable speed.

The computer program brings the „time of day“ into registers A to F by means of the TIME command, whereby the values are stored in registers 2 to 7 by means of the EXTRA command. Exactly one second later, the „time of day“ is taken over again and stored. If one subtracts the two „times“ from each other, the frequency in hertz (oscillations per second) results.

It must also be ensured that the measuring time between the two TIME commands is exactly one second. For this, the pulses of the 1 Hz TAKT/CLOCK output are used. As soon as the program is started, the computer waits in a wait loop until the clock signal is „high“ (1) present (see figure, section A).

In a second wait loop, the computer waits until the clock signal becomes „low“, i.e. 0, again (in the figure, section B).

TAKT/CLOCK signal



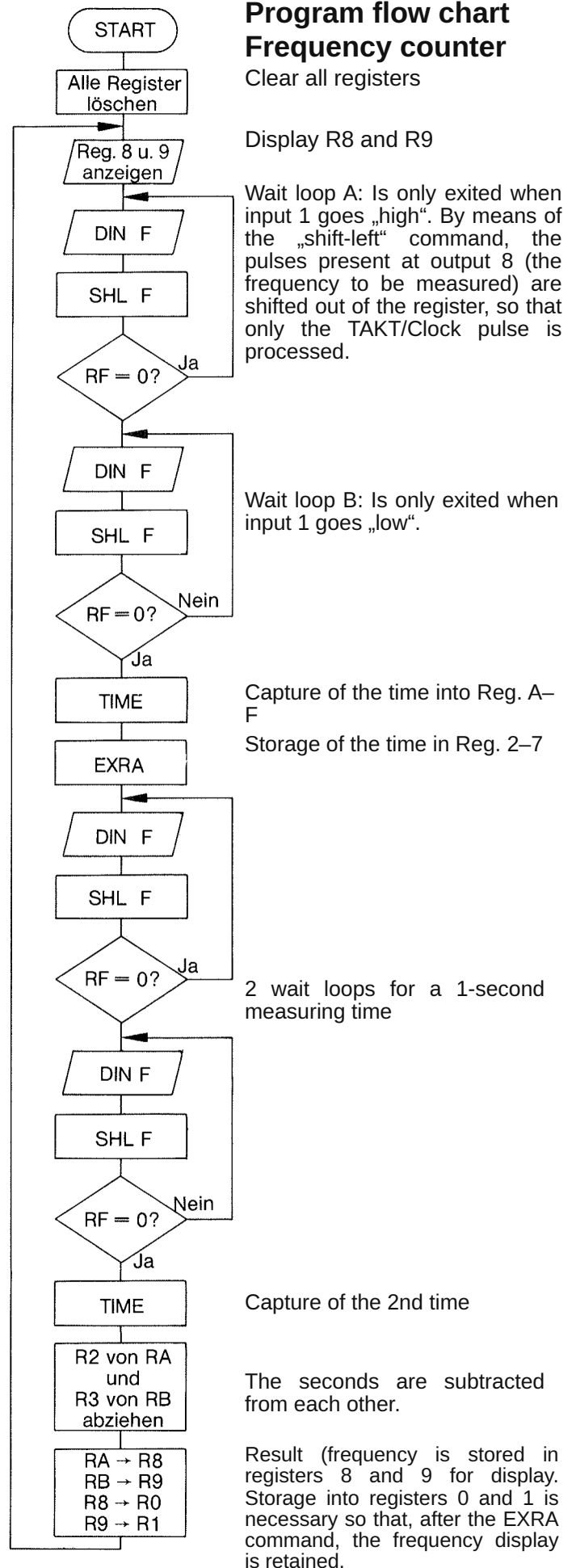
At the same moment when the clock signal becomes „low“ (0), the computer takes over the first time reading. With the second wait loop, it is achieved that the signal change to „high“ and back again to „low“ is waited for (exactly 1 second), in order to then take over and evaluate the second time reading.

The two wait loops (sections A and B in the figure) are necessary in order to achieve a defined start of the measuring time. The individual program functions can be seen from the program flow chart. It is not the complete „times of day“ that are subtracted from each other, but only the seconds. When subtracting the tens-of-seconds digit, the decimal correction, on the occurrence of a carry, is carried out by subtracting the hexadecimal value A (decimal 10), because the tens-of-seconds only go up to 5.

The electronic construction of the multivibrator (also called an astable multivibrator) should be familiar from the experiments with the Electronic Studios 2065 or 2070. In our frequency-measuring experiment, the loudspeaker is operated with an additional transistor. This is necessary because the computer needs clean square-wave pulses for the counting process. If the loudspeaker were operated without the second transistor, the coil windings of the loudspeaker could severely distort the square-wave pulses.

The arrangement of the electronic multivibrator circuit also remains in place for the following experiment.

Program flow chart Frequency counter



A highly interesting experiment

Reaction-time meter

The multivibrator (astable multivibrator) of the previous experiment remains in place unchanged. On the computer board, make the connections as shown in the figure. The plus and minus leads of the multivibrator to the computer board are also shown in the figure. The piezo buzzer is connected only after programming. The program is entered as usual.

Program: Reaction-time meter

Address	Code	Mne- monic
00	F08	CLEAR
01	F0E	EXRM
02	F08	CLEAR
03	FE0	DOT 0
04	F28	DISP 2,8
05	FDF	DIN F
06	31F	ANDI 1,F
07	E05	BRZ 05
08	FDF	DIN F
09	31F	ANDI 1,F
0A	E0C	BRZ 0C
0B	C08	GOTO 08
0C	F06	TIME
0D	F0F	EXRA
0E	FDF	DIN F
0F	31F	ANDI 1,F
10	E0E	BRZ 0E

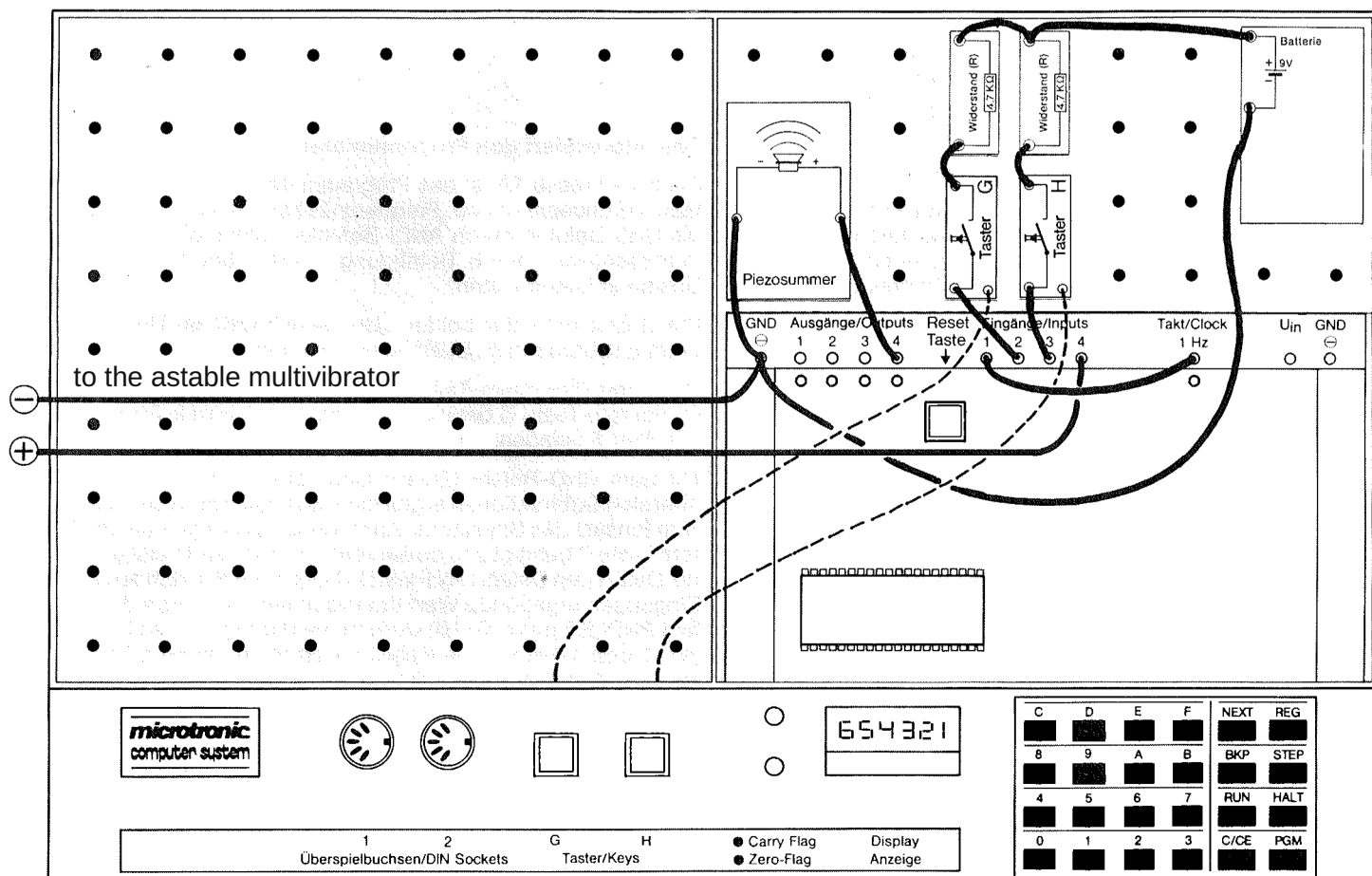
11	FDF	DIN F
12	31F	ANDI 1,F
13	E15	BRZ 15
14	C11	GOTO 11
15	F06	TIME
16	B78	CALL 78
17	0A8	MOV A,8
18	0B9	MOV B,9
19	080	MOV 8,0
1A	091	MOV 9,1
1B	FDF	DIN F
1C	32F	ANDI 2,F
1D	E04	BRZ 04
1E	158	MOVI 5,8
1F	F02	DISOUT
20	F05	RND
21	71D	SUBI 1,D
22	FCE	SUBC E

23	D25	BRC 25
24	C21	GOTO 21
25	FDF	DIN F
26	36F	ANDI 6,F
27	92F	CMPI 2,F
28	E31	BRZ 31
29	1F0	MOVI F,0
2A	1F1	MOVI F,1
2B	FE1	DOT 1
2C	101	MOVI 0,1
2D	FE1	DOT 1
2E	710	SUBI 1,0
2F	E20	BRZ 20
30	C2A	GOTO 2A
31	1FF	MOVI F,F
32	FEF	DOT F
33	F06	TIME
34	F0F	EXRA
35	FDF	DIN F
36	34F	ANDI 4,F
37	E35	BRZ 35
38	F06	TIME
39	10F	MOVI 0,F
3A	FEF	DOT F
3B	B78	CALL 78
3C	008	MOV 0,8

3D	100	MOVI 0,0
3E	0A1	MOV A,1
3F	0B2	MOV B,2
40	0C3	MOV C,3
41	104	MOVI 0,4
42	105	MOVI 0,5
43	F0D	EXRL
44	120	MOVI 2,0
45	101	MOVI 0,1
46	102	MOVI 0,2
47	103	MOVI 0,3
48	104	MOVI 0,4
49	105	MOVI 0,5
4A	F0C	DIV
4B	F0E	EXRM
4C	408	ADD 0,8
4D	D51	BRC 51
4E	998	CMPI 9,8
4F	D51	BRC 51
50	C53	GOTO 53
51	568	ADDI 6,8
52	519	ADDI 1,9
53	419	ADD 1,9
54	D58	BRC 58
55	999	CMPI 9,9

Program continues —
please see next page!

Connection of the astable multivibrator (from page 62) to the computer



Address	Code	Mne- monic
56	D58	BRC 58
57	C5A	GOTO 5A
58	569	ADDI 6,9
59	51A	ADDI 1,A
5A	42A	ADD 2,A
5B	D5F	BRC 5F
5C	99A	CMPI 9,A
5D	D5F	BRC 5F
5E	C61	GOTO 61
5F	56A	ADDI 6,A
60	51B	ADDI 1,B
61	F0E	EXRM
62	F30	DISP 3,0
63	FDF	DIN F
64	32F	ANDI 2,F
65	E63	BRZ 63
66	718	SUBI 1,8
67	E69	BRZ 69
68	C1F	GOTO 1F
69	F02	DISDUT
6A	F0E	EXRM
6B	F0F	EXRA
6C	F0D	EXRL
6D	150	MDVI 5,0
6E	101	MOVI 0,1
6F	102	MOVI 0,2
70	103	MOVI 0,3
71	104	MOVI 0,4
72	F0C	DIV
73	1E3	MOVI E,3

74	1E4	MOVI E,4
75	F50	DISP 5,0
76	FF0	KIN 0
77	C00	GOTO 00
78	62A	SUB 2,A
79	D7B	BRC 7B
7A	C7D	GOTO 7D
7B	76A	SUBI 6,A
7C	513	ADDI 1,3
7D	63B	SUB 3,B
7E	D80	BRC 80
7F	C82	GOTO 82
80	7AB	SUBI A,B
81	514	ADDI 1,4
82	10F	MDVI 0,F
83	64C	SUB 4,C
84	D86	BRC 86
85	C87	GOTO 87
86	76C	SUBI 6,C
87	90C	CMPI 0,C
88	E91	BRZ 91
89	71C	SUBI 1,C
8A	56B	ADDI 6,B
8B	99B	CMPI 9,B
8C	D8E	BRC 8E
8D	C87	GOTO 87
8E	56B	ADDI 6,B
8F	51F	ADDI 1,F
90	C87	GOTO 87
91	0FC	MOV F,C
92	F07	RET

If any number key is then pressed, the frequency measurement reappears on the display, and we can check whether the set frequency number „20“ is still correct. As soon as the red button G is pressed again, the next reaction-time measurement begins.

If the red button G is not pressed before the signal tone sounds, or button H is pressed prematurely, an interrupted signal tone sounds to demonstrate that the measurement was not carried out correctly. Incorrect measurements are not counted and are also not included in the calculation of the average time.

False readings (wrong times) can occur if the reaction time of a test subject amounts to several seconds (which hopefully does not apply in our case).

In the previous experiment, „Frequency meter“, we determined the limit frequency of our computer. This can be around 30 Hz (sometimes even 40 or 50 Hz).

At the moment, our reaction-time meter is set to 20 Hz. The accuracy is thus 1/20 second. If our computer has a limit frequency of 30 or 40 or perhaps even 50 Hz, an even more precise reaction-time measurement results, because we can then operate with 1/30, 1/40 or 1/50 seconds.

At a limit value of 30 Hz, a program change at address 44 is required. The existing command MOVI 2,0 (entry 120) is to be changed to MOVI 3,0 (entry 130). The multivibrator is to be adjusted so that the display shows exactly „30“.

If our limit frequency was at or above 40 Hz, the MOVI command is changed to the entry 140. The multivibrator is to be adjusted so that exactly „40“ is shown on the display. At 50 Hz, the MOVI entry is 150 and the display must show „50“.

This is of interest to the programmer:

Up to address 1A, the program is structured similarly to the previous „frequency counter“. However, the SHL commands were replaced by ANDI commands, so that no nonsense can arise in the wait loops when a red button is pressed.

The subtraction of the two „times of day“ is carried out in the subroutine „SUB“.

After the „frequency counter“ program part follows the query as to whether the red button G is pressed. If „yes“, the value 5 is loaded into register 8.

With the RND command (random generator) and the corresponding subtractions, a random waiting time until the start of the signal tone results. Then it is checked whether the red button G was pressed and whether button H was not yet pressed. By means of the command DIN F, the value resulting at the 4 inputs is taken into register F, and by means of the command ANDI 6,F (binary: 0110), the signals at inputs 1 and 4 are cleared. The value 2 (binary: 0010) is then still present in register F if only button G was pressed. If this is not the case, the outputs are switched on and off several times (a rapidly interrupted signal tone as a demonstration that the button operation was not carried out correctly). Otherwise, the continuous tone is switched on by means of the command MOVI F,F and DOT F, and the first „time of day“ is taken over. In a wait loop, the pressing of button H is awaited. Then the second „time of day“ is likewise taken over.

Connect the piezo buzzer. Start the program.

The computer first works as a frequency counter. With the potentiometer knob, the frequency is set until the display shows exactly 20 Hz. In case of any fluctuations in the reading, it must be readjusted until exactly 20 is shown.

With this program, we are able to measure our reaction time very precisely. Perhaps it is interesting to repeat a measurement under unfavorable conditions (lack of sleep, alcohol consumption) as well.

We press the red button G and hold it down until the computer signals with the randomly starting signal tone. Now the red button H must be pressed as quickly as possible. The display shows the reaction time with 3 digits. The left display digit shows the seconds, the middle one the 1/10 seconds, and the right display digit the 1/100 seconds. „040“ therefore means 0.4 seconds.

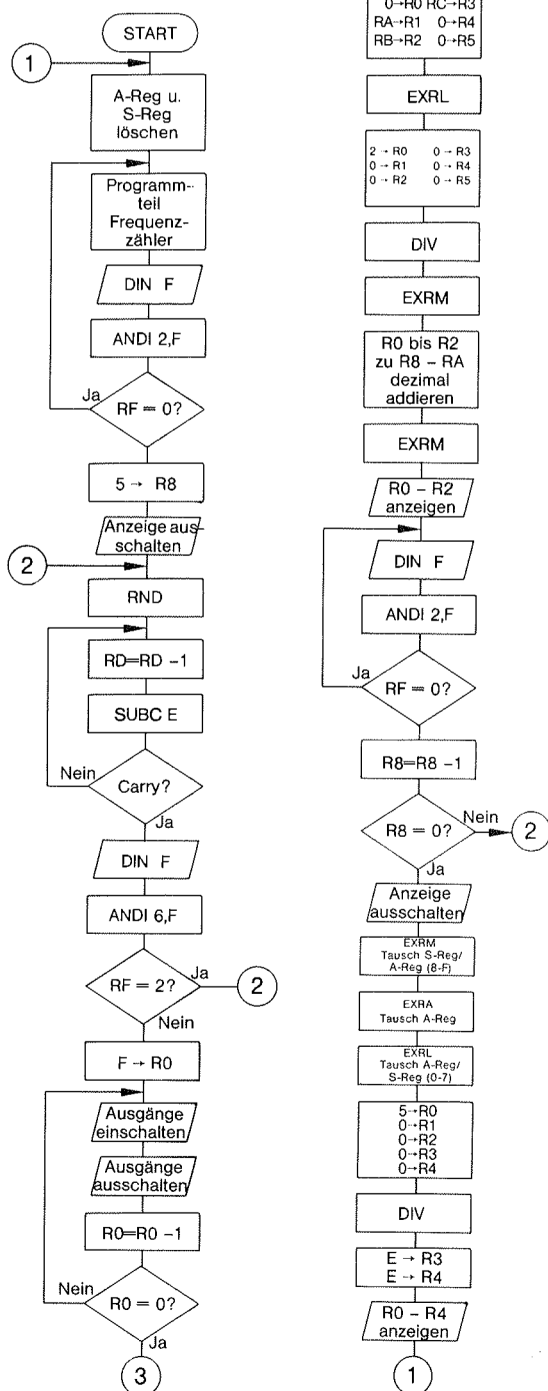
5 reaction-time measurements in succession are provided for, which is why the red button G is pressed again until the start of the signal tone. The switch to the red button H brings the second measurement.

After the fifth measurement, the display shows: EE XXX. For „X“, 3 numbers are displayed – an average value of the five reaction-time measurements.

In the subroutine „SUB“, the number of counted pulses (frequency measurement) is determined by subtracting the two times of day. At a frequency setting of 20 Hz, the measured pulse count is divided by 2, whereby the reaction time is determined. (At a frequency of 40 Hz it is divided by 4, etc. – command change at address 44).

The determined reaction time is added into the memory registers 8 to A, i.e. after five reaction-time measurements the total time is stored there.

The program waits until button G is pressed. Then the value 1 is subtracted from register 8 (in which the value 5 was present at the start) each time. The measurements are ended as soon as the value 0 is reached in register 8. The sum of the reaction times (in the memory registers 8 to A) is divided by 5, and the average reaction time is displayed.



Digital voltmeter

(BUSCH Electronic Studios 2065 or 2070 required)

A digital voltmeter is a voltage-measuring instrument in which the measurement results are displayed digitally, i.e. in directly readable numerical values. Compared with an analog measuring instrument, in which the measurement result is shown by the deflection of a pointer, digital displays are easier to read. In addition, digital measuring instruments work more accurately. However, a greater circuit complexity is also required.

In a digital voltmeter, the voltage (an analog quantity) must be converted into a digital quantity. For this there are special integrated circuits, so-called A/D converters (analog-to-digital converters).

The display accuracy of a digital measuring instrument depends on how many parts the quantity to be measured can be divided into. If, for example, a 10 V voltage is divided into 20 parts, the smallest displayed value (or the smallest voltage difference) works out to a range of 0.5 V. If 10 V is divided into 100 parts, an accuracy of 0.1 V already results.

For the analog-to-digital conversion we again use an „astable multivibrator“, whose construction, however, changes compared with the multivibrator used up to now. We rebuild the circuit according to the figure. TAKT/CLOCK must again be connected to INPUT 1.

The construction plan is authoritative when the large Electronic Studio 2070 is available. When using the Electronic Studio 2065, the 3 components shown directly to the left of the battery are omitted (loudspeaker, transistor and 22 K Ohm resistor). In this case the earphone is to be used (drawn in the plan with a dotted line). The connecting leads to the unused components

are omitted, and the plus pole of the battery is connected directly to the 1 K Ohm resistor (top center).

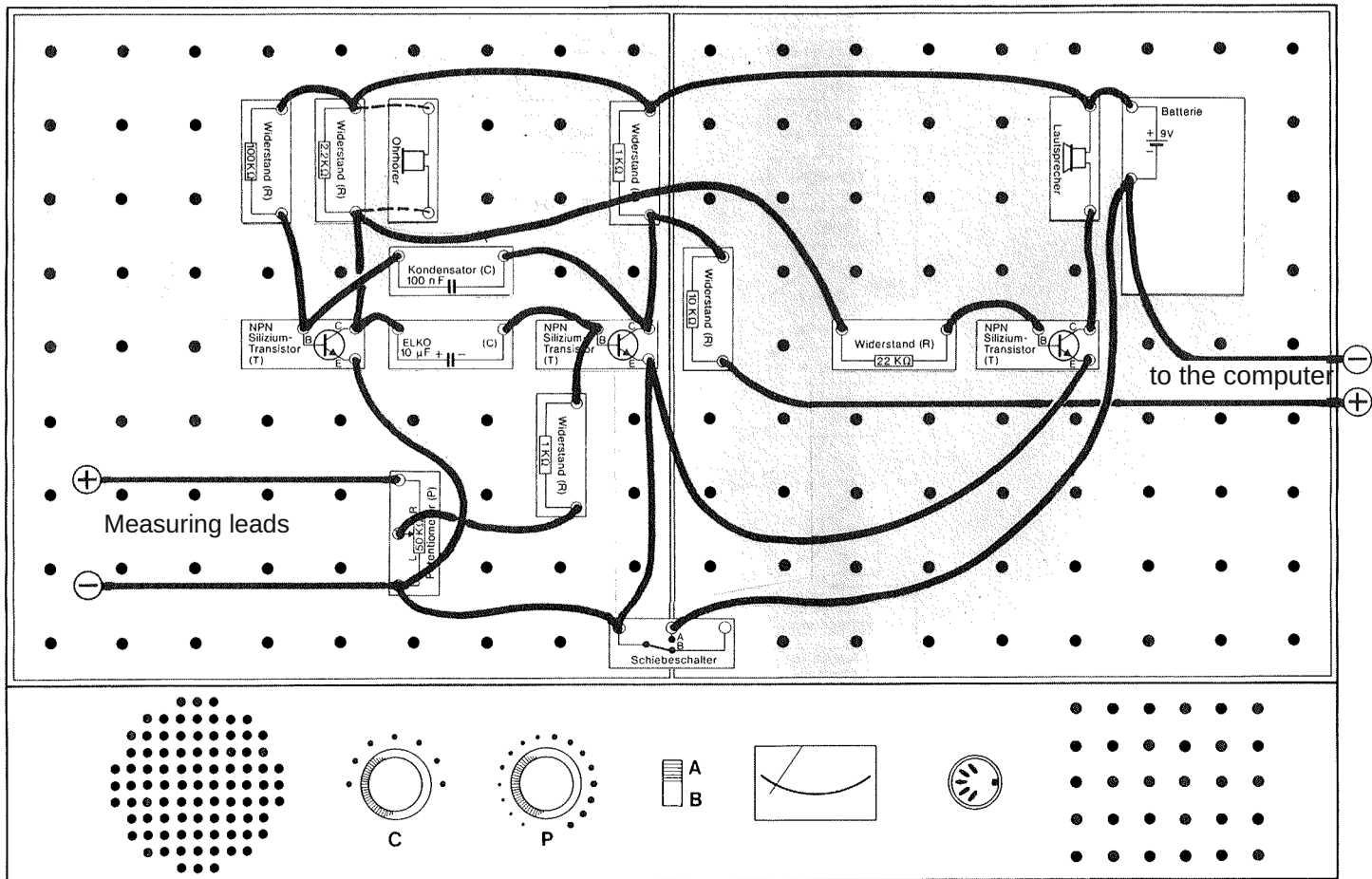
The leads leading to the computer are again marked with $\oplus$ and $\ominus$. As in the previous experiment, the $\oplus$ lead is again connected to INPUT 4 of the computer board. The $\ominus$ lead leads to the GND socket. The red buttons G and H that may still be connected are not required.

So that we can „calibrate“ our digital voltmeter, the program must first be entered according to the table.

Program: Digital voltmeter

Addr. No.	Code	Mnemonic	Explanations
00	F08	CLEAR	
01	F28	DISP 2,8	
02	FDF	DIN F	
03	FAF	SHL F	
04	E02	BRZ 02	
05	FDF	DIN F	
06	FAF	SHL F	
07	E09	BRZ 09	
08	C05	GOTO 05	
09	F06	TIME	
0A	F0F	EXRA	
0B	FDF	DIN F	
0C	FAF	SHL F	
0D	E0B	BRZ 0B	

Astable multivibrator for the digital voltmeter



0E	FDF	DIN F		
0F	FAF	SHL F		
10	E12	BRZ 12		
11	C0E	GOTO 0E		
12	F06	TIME		
13	62A	SUB 2,A		
14	D16	BRC 16		
15	C18	GOTO 18		
16	76A	SUBI 6,A		
17	513	ADDI 1,3		
18	63B	SUB 3,B		
19	D1B	BRC 1B		
1A	C1C	GOTO 1C		
1B	7AB	SUBI A,B		
1C	B4C	CALL 4C		
1D	0A0	MOV A,0	Jump to „MOVI 0“	
1E	0B1	MOV B,1	Jump to „MOVI 0“	
1F	F0D	EXRL		
20	B4C	CALL 4C		
21	0A0	MOV A,0		
22	0B1	MOV B,1		
23	F0B	MULT		
24	F0D	EXRL		
*25	170	MOVI 7,0	(MOVI 5,0 150)	Command change at 40 Hz frequency
*26	161	MOVI 6,1	(MOVI 2,1 121)	
27	F0B	MULT	Jump to „MOVI 0“	
28	04F	MOV 4,F		
29	03E	MOV 3,E		
2A	02D	MOV 2,D		
2B	B4C	CALL 4C		
2C	0A0	MOV A,0		
2D	0B1	MOV B,1		
2E	F0D	EXRL		
*2F	161	MOVI 6,1	(MOVI 5,1 151)	Command change at 40 Hz frequency
*30	112	MOVI 1,2	(MOVI 2,2 122)	
*31	133	MOVI 3,3	(MOVI 1,3 113)	
32	F0B	MULT		
33	42D	ADD 2,D		
34	D38	BRC 38		
35	99D	CMPI 9,D		
36	D38	BRC 38		
37	C3A	GOTO 3A		
38	56D	ADDI 6,D		
39	51E	ADDI 1,E		
3A	43E	ADD 3,E		
3B	D3F	BRC 3F		
3C	99E	CMPI 9,E		
3D	D3F	BRC 3F		
3E	C41	GOTO 41		
3F	56E	ADDI 6,E		
40	51F	ADDI 1,F		
41	44F	ADD 4,F		
42	D46	BRC 46		
43	99F	CMPI 9,F		

44	D46	BRC 46	Subroutine „MOVI 0“
45	C47	GOTO 47	
46	56F	ADDI 6,F	
47	0F9	MOV F,9	
48	0E8	MOV E,8	
49	091	MOV 9,1	
4A	080	MOV 8,0	
4B	C01	GOTO 01	
4C	100	MOVI 0,0	
4D	101	MOVI 0,1	
4E	102	MOVI 0,2	
4F	103	MOVI 0,3	
50	104	MOVI 0,4	
51	105	MOVI 0,5	
52	F07	RET	

* At these addresses, note the command changes in the 2nd part of the experiment.

Start the program. The display shows 00. To calibrate the voltmeter we need a 9 V battery that has already been in operation for some time. We deliberately do not use a factory-fresh battery, because such new batteries (despite the corresponding label) usually have 9.6–9.8 V, whereas batteries that have already been in use show quite exactly 9 V when no current is being drawn. The battery is connected to the **measuring leads** marked with $\oplus$ and $\ominus$ (see figure). We turn the potentiometer knob back to the left stop. Nothing can be heard in the earphone or loudspeaker. Now the potentiometer knob is slowly turned to the right in small steps. The loudspeaker (earphone) again produces the familiar crackling sound. We keep turning to the right until the display shows, as exactly as possible, 90 (9.0 V). The computer needs about 2–3 seconds for the measuring process, which is why after each potentiometer setting one must wait for the changed display value.

We will probably find that we will not succeed in calibrating precisely at 9 V. The display may show fluctuating values between 8.4 and 9.5 V. This should not bother us for now.

With our digital voltmeter we can now measure DC voltages **up to 9 V**. We surely have 1.5 V batteries available in another electrical device (e.g. in a portable radio), so that when using 2 batteries we can measure 3 V, or with 3 batteries 4.5 V, and so on. If we own the Electronic Studio „Digital-Technik 2075“, the very accurately working 5 V socket of the IC counter module can also be used for calibrating and measuring.

In its current form, our digital voltmeter has a measuring accuracy of 0.4 to 0.5 V. With voltages that have intermediate values, the mentioned display fluctuations result.

If, however, in the „Frequency counter“ chapter we found that the limit frequency of our microcomputer is above 40 Hz, the measuring accuracy of our digital voltmeter can be substantially improved. For this, the commands marked with a * in the program table must be exchanged for the commands given in parentheses. So, for example, at address 25 we change the command entered as „170“ to „150“, and so on.

After this program change, the device must be recalibrated as described above. The measuring accuracy of our digital voltmeter is now already about 0.2 V. However, at voltages above 6 V there will still be certain display fluctuations.

Attention! The digital voltmeter can only be used for direct-current measurements up to 9 V. Alternating-current voltages are not displayed. Connecting alternating-current voltages of over 9 V to the measuring leads can lead to the destruction of components.

How does the digital voltmeter work?

Owners of the Electronic Studio „Digital-Technik 2075“ are familiar with the principle of converting an analog voltage into a digital measurement result.

In our experiment, the astable multivibrator generates a frequency which is low at low voltage (slow crackling) and becomes higher as the voltage rises (fast crackling). The computer counts the frequencies and reports them on the display as the voltage measurement result.

Example: The frequency amounts to 20 Hz at a connected voltage of 9 V. The voltage at a frequency of only 10 Hz results from the following formula:

$$U = \frac{9 \text{ V}}{20 \text{ Hz}} \times 10 \text{ Hz} = 4.5 \text{ V}$$

or at a frequency of 11 Hz:

$$U = \frac{9 \text{ V}}{20 \text{ Hz}} \times 11 \text{ Hz} = 4.95 \text{ V (rounded up to 5 V)}.$$

In our currently constructed experiment, the calculation is unfortunately not quite so simple, because the higher or lower frequency is not generated linearly in relation to the connected voltage. If, at a voltage of 9 V, the 20 Hz mentioned in the example are generated, then at a 5 V connection 11 Hz would have to be generated in order to arrive at a correct result. In reality, however, our astable multivibrator does not generate 11 Hz at a connected voltage of 5 V, but 12.5 Hz. If we were to apply the same formula again, we would arrive at the following (incorrect) result:

$$U = \frac{9 \text{ V}}{20 \text{ Hz}} \times 12.5 \text{ Hz} = 5.6 \text{ V (5 V would be correct)}$$

For the conversion of the frequency into a voltage, we must find a different formula for the astable multivibrator that is available. Taking into account the non-linear frequency generation, the following formula was found through experiments:

$$U = 0.0067 \times f^2 + 0.3160 \times f$$

If the corresponding frequency, e.g. 20 Hz, is substituted for f, we arrive at the following result:

$$U = 0.0067 \times 20 \times 20 + 0.3160 \times 20 = 9 \text{ V}.$$

At 12.5 Hz the result is:

$$U = 0.0067 \times 12.5 \times 12.5 + 0.3160 \times 12.5 = 5 \text{ V}.$$

Since our computer cannot measure 12.5 Hz (but only either 12 or 13 Hz), a connected 5 V measuring voltage produces fluctuating display values between 4.7 and 5.2 V.

This is of interest to the programmer:

Up to address 1B, the same program section results as in the „Frequency counter“. The generated frequency is taken into registers A and B. These values move into registers 0 and 1 and are stored in the memory registers 0 and 1.

Then registers 0 and 1 are once again loaded with the contents of registers A and B, and the command MULT is executed. Registers 0 to 5 now contain the square of the frequency. This result is multiplied by 67 (instead of 0.0067 in our previously mentioned formula). The first three digits of this result (in registers 2, 3 and 4) are stored in registers D, E and F.

Registers A and B still contain the measured frequency values, which are now multiplied by 3160 (instead of 0.3160 in our formula).

The result is added into registers D, E and F.

The final result in registers E and F is transferred to registers 8 and 9 (or registers 0 and 1) for the result display.

The program sequence is to be presented clearly once more:

1. Measure the frequency (e.g. 14 Hz)
2. Square the frequency $14 \times 14 = 196$
3. Multiply the result by 67: $196 \times 67 = 13132$
4. Store the first three digits: $= 131$
5. Multiply the measured frequency
14 Hz by 3160: $14 \times 3160 = 44240$
6. Add the first three digits to the
result of 4: $131 + 442 = 573$
7. Display the first two digits

We compare with the formula found earlier:

$$U = 0.0067 \times 14^2 + 0.3160 \times 14 = 5.737$$

The computer's calculations agree with this formula. It would be pointless to display the result with 3 digits, because the frequency generated by the astable multivibrator is not accurate enough.

If the frequency is increased to 40 Hz (at 9 V), the formula changes as follows:

$$U = 0.0025 \times f^2 + 0.125 \times f$$

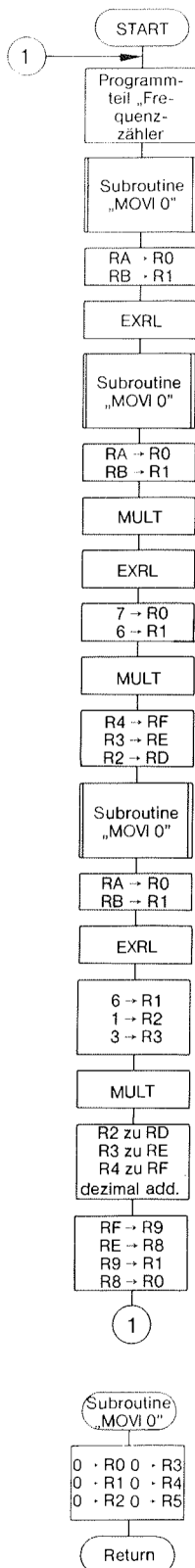
This formula change was taken into account in the command changes given in parentheses.

A supplementary pack for exact measurement results is in preparation. This pack contains, among other things, a special A/D converter, temperature sensor, etc., in order to be able to carry out very accurate measurements of voltages, temperatures, light, and so on. By incorporating these measured values – which, for example, are reported by the computer to the outputs when values fall below or rise above set limits – many further application possibilities of the Microtronic computer system arise.

Computer-controlled model railway

First expansion stage

Program flow chart: Digital voltmeter.



The monitoring and control of a model-railway layout is an interesting application for our computer.

Computer control offers great advantages over the relay circuits customary up to now: if, for example, the control sequence is to be changed, only a change to the computer program is necessary. A correctly executed „hardware wiring“ remains in place unchanged. The computer takes over the fully automatic model-railway operation, and by means of a small additional program this automatic mode can be switched over to semi-automatic or manual operation. One can also intervene in automatic operation, for example when a point setting controlled by the computer is to be changed manually. This point changeover is registered by a computer input. The computer checks whether an „accident“ could occur and diverts the possibly „threatened“ train onto another track – the point in question can be set. If this is not possible, the planned point setting is held back until the endangered train has passed the possibly threatened spot. Only then is the manual point setting released.

The control of so-called „point routes“ can also be a task for the computer.

Some model-railway enthusiasts are of the opinion that only a few connecting cables need be attached to the computer in order to achieve a fully automatically controlled model-railway operation. Now that we have in the meantime occupied ourselves intensively with the possibilities of a microcomputer, it is clear to us that various preparations must be made for such a conversion.

It is essential to note:

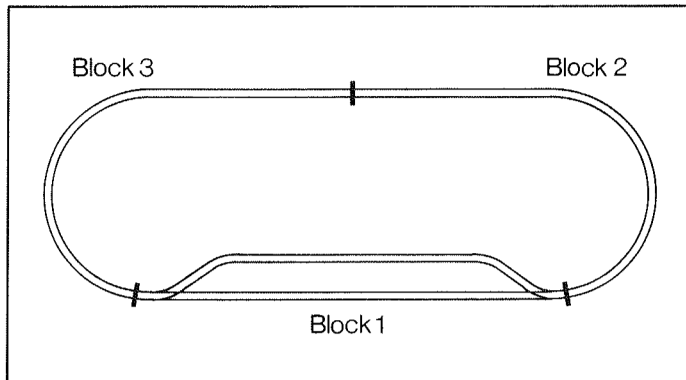
Since the computer works only with direct-current voltages and very small currents, in the following experiments no currents (voltages) from the railway transformer may reach the inputs or outputs of the computer. Points, track sections or other electrical items supplied by the railway transformer may therefore never be connected directly to the computer board.

The danger is great that a careless connection will lead to the destruction of our valuable microcomputer. Thus the computer must be told, via its inputs, which tracks are occupied, in which directions the points are set, and so on. For this, no track contacts or the like may be connected to the computer. By „interposing“ the special BUSCH relay 5964 and the BUSCH light barrier 5962, suitable aids are available to us for the computer-controlled model railway.

First, some consideration is necessary as to how a model-railway layout is to be prepared for computer control. This is to be demonstrated on a simple track oval with a pair of points.

On this circuit with a passing siding (station), three trains can run fully automatically and without accidents. While one train arrives at the station and stops there, the train waiting on the second track is released. The third train is on the „line“.

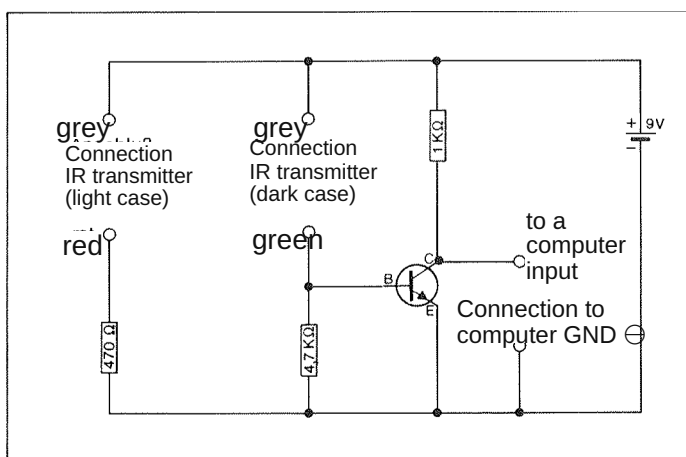
The circuit is divided, as shown in the figure, into three „blocks“. The computer must then check and calculate which track block is to be blocked or released and how the points are to be set.



All manufacturers of model railways offer easily realizable ways of dividing the track routes into individual block sections by means of appropriate insulating pieces.

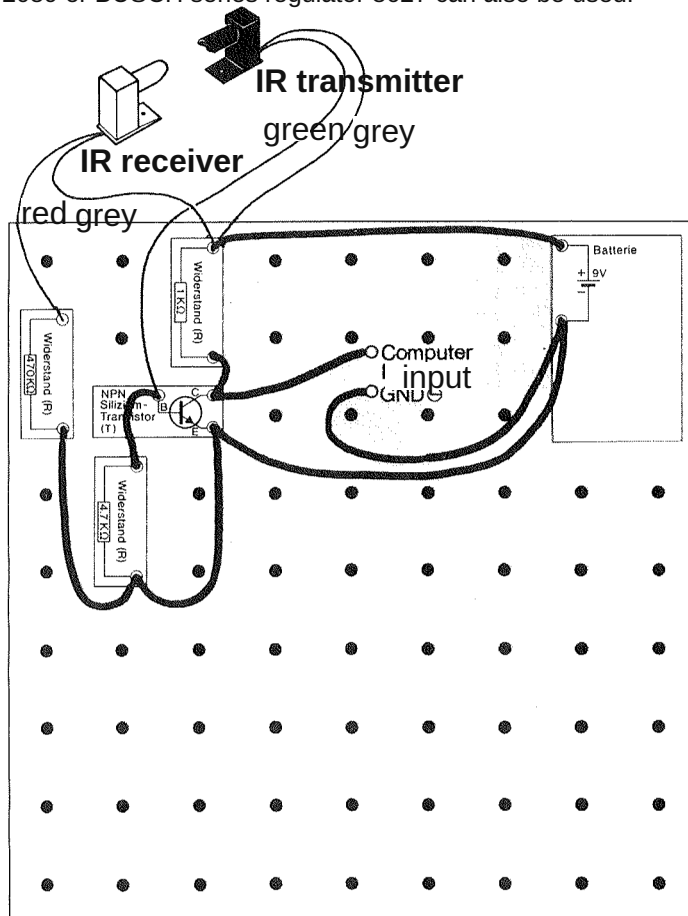
The BUSCH light barriers No. 5962 are ideally suited for the „monitoring“ of the individual track blocks. They are not only relatively inexpensive, but also provide the contactless data transmission required for the computer. A light barrier consists of a small infrared transmitter and receiver, which are mounted opposite each other on a track section to be monitored. When a train passes through, a signal is triggered which is evaluated by the computer. For this, a simple amplifier circuit consisting of one transistor and three resistors is required. Owners of a BUSCH Electronic Studio can easily build the circuit by using the corresponding BUSCH plug-in modules. Since all components are also available individually, they can be obtained from the BUSCH customer service. You will find a parts list at the end of this chapter.

Circuit diagram: IR light-barrier amplifier



Assembly plan: IR light-barrier amplifier

Instead of the battery module, the BUSCH power-supply unit 2059 or BUSCH series regulator 8627 can also be used.



Assembly plan of the amplifier circuit for light barriers. Several amplifier circuits can be built on one plug-in board and supplied from the same voltage source.

The small amplifier circuit (as per the figures) supplies the computer with a „1“ signal (voltage present) when the light barrier is interrupted (train passing through), or a „0“ signal when the light barrier is not interrupted. To simplify matters in the overall circuit diagram of the model layout, the small amplifier circuit is represented by the following symbol:

Light-barrier amplifier symbol



Running currents and points are switched with the BUSCH relay 5964. In the chapter „Connection of relays“ it was shown how the relay 5964 can be driven by the computer by means of a simple transistor stage (1 resistor, 1 transistor).

For the power supply of the light-barrier amplifier and for the simple transistor stage of the relay 5964, a 9 V battery block has been provided in each of the assembly plans. Instead of the battery module, the BUSCH power-supply unit 2059 can also be used, which likewise delivers a 9 V DC voltage (250 mA). For the relay, a DC control current of 9 V is required in every case. Under no circumstances may 9 V AC be connected. The light barriers work in a range of 5 to 9 V DC. The BUSCH series regulator No. 8627 is also suitable for this. When connected to an AC transformer of 12 to 16 V, this series regulator delivers a 5 V DC voltage.

We need electromagnetic points with a stop function and **end cut-off**. The end cut-off is necessary because the points receive a longer switching pulse via the relays, which – in the case of points without end cut-off – can lead to overheating and, with prolonged use, to the destruction of the electromagnetic drive.

Now we can carry out the wiring of our – for the time being still simple – model layout. We need four light barriers and three relays, which are to be wired according to the overall circuit diagram. We note that the cables responsible for the straight-ahead setting of the points lead to contact point 3 of relay No. 1, while the cables for the branch setting are to be connected to contact point 1. This is important because both points must be operated in unison, i.e. both points are set either to straight-ahead or to branch. Now we still need a computer program that takes over the monitoring and automatic control. The following prerequisites result from this:

A train passes light barrier 4 (L4). The track block No. 3 lying behind it must be switched off (de-energized), so that a possible second following train does not come too close to a train travelling ahead.

A train passes light barrier 1 (L1). Track block 2 must be switched off. At the same time, track block 3 is released again. Depending on the point setting, light barriers 2 and 3 (L2 and L3) must be monitored until a train arrives. Light barriers 2 and 3 have the following task:

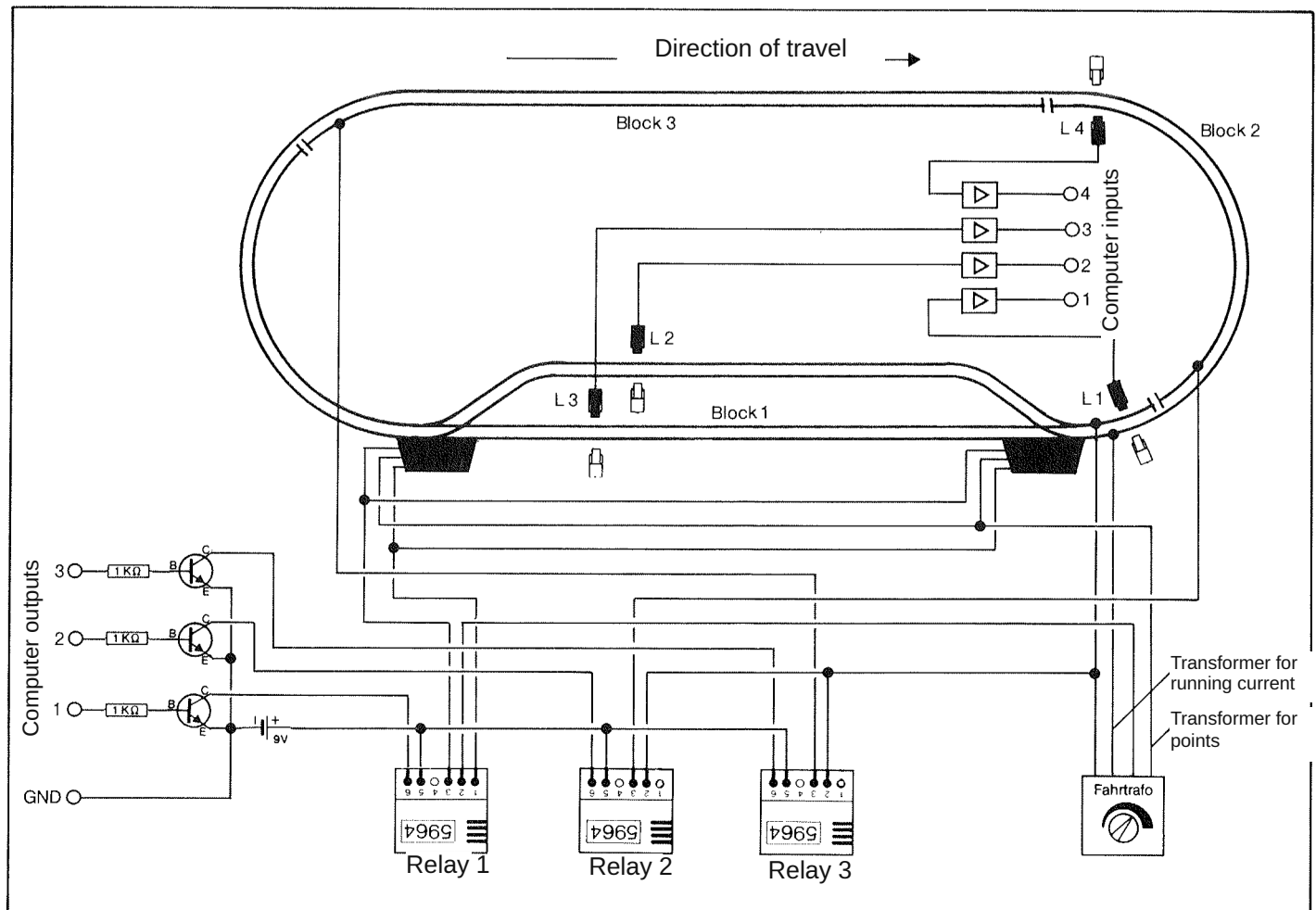
When a train arrives at the station, the two points must be switched over. When stop points („thinking“ points) are used, this switches off the running current of the incoming train and releases the opposite track. In addition, track block 2 must be switched on again.

In order to be able to incorporate these considerations into a computer program, a whole series of so-called „logical operations“ (ANDI or OR commands) are necessary, in order to address, of the 4 computer inputs or outputs, only the one that is required for the control of the respective situation.

If, for example, the state of light barrier 4 (light barrier interrupted or light barrier free, i.e. „1“ or „0“) is to be taken over, then all values of the 4 connected light barriers must be stored in a register by means of the DIN command. By means of the command ANDI 8, the current state of light barrier 4 can then be processed. The value of the 3 remaining inputs is briefly set to 0.

The explanations in the program table should be sufficient to understand the program. It works through the conditions formulated above one after the other. In register 0, the value of the light barriers is taken over. In register 1, it is stored which outputs are switched on or off. In register 2, an intermediate storage takes place if light barrier L1 was interrupted, until light barriers L2 or L3 are interrupted. It must be pointed out that the connecting lead – possibly present on the computer board from earlier experiments – between

Overall circuit diagram, first expansion stage



TAKT/CLOCK and an INPUT must be removed without fail, because otherwise program malfunctions arise. We enter the program according to the table:

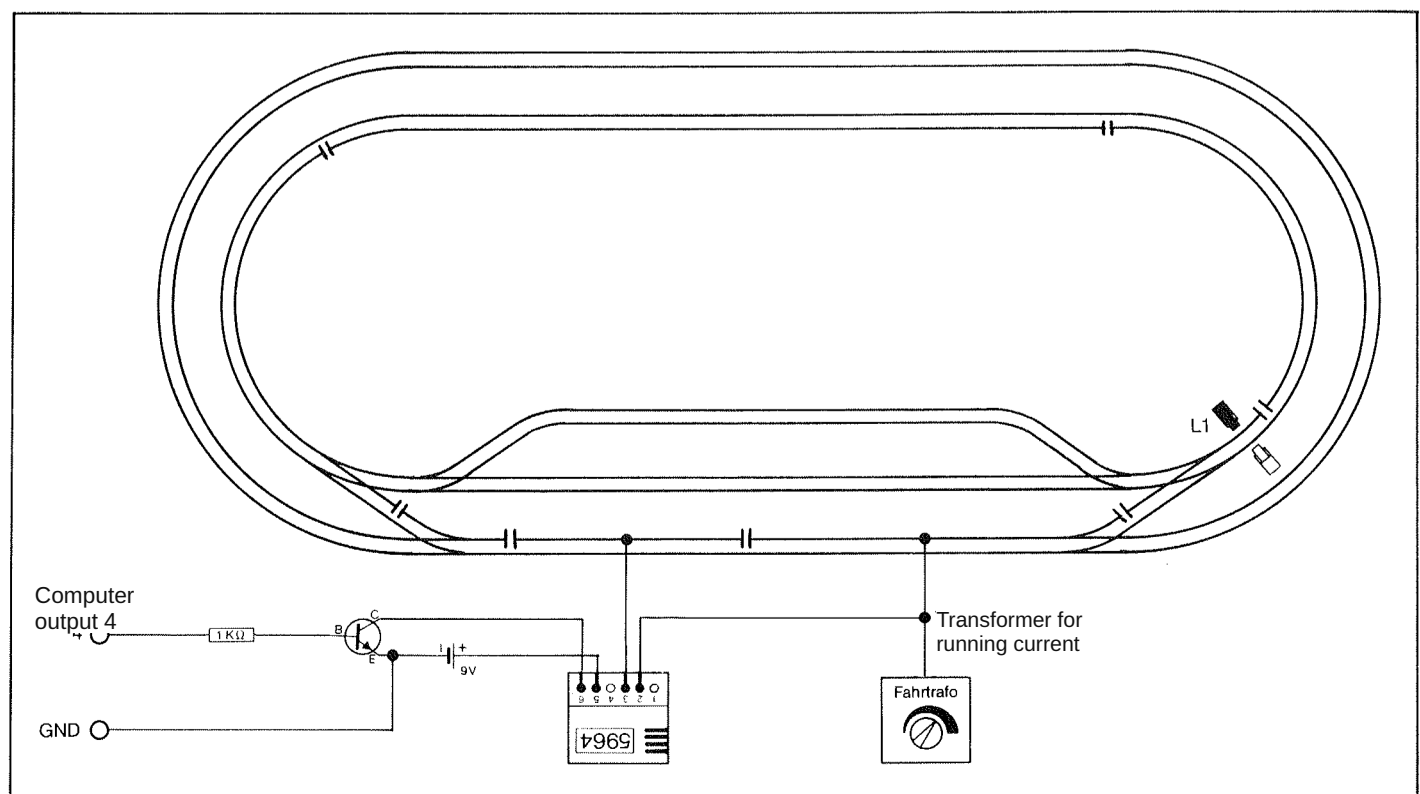
Program: Model-railway control, first expansion stage

Adr. No.	Code	Mnemonic	Explanations
00	F08	CLEAR	
01	F02	DISOUT	
02	161	MOVI 6,1	
03	FE1	DOT 1	Value of R1 to outputs
04	FD0	DIN 0	Take over value of light barriers
05	380	ANDI 8,0	By ANDI 8,0 take over only value of L4
06	E08	BRZ 08	If L4 equals 0, then jump to address 08
07	3B1	ANDI B,1	By ANDI B (1011) switch off block 3
08	FD0	DIN 0	Take over value of light barriers
09	310	ANDI 1,0	By ANDI 1,0 take over only value of L1
0A	E0F	BRZ 0F	If L1 equals 0, then jump to address 0F
0B	1F2	MOVI F,2	Store that L1 was interrupted
0C	3D1	ANDI D,1	By ANDI D (1101) switch off block 2
0D	14F	MOVI 4,F	By OR operation of 4 (0100) release block 3
0E	AF1	OR F,1	
0F	9F2	CMPI F,2	Was L1 interrupted?
10	E12	BRZ 12	Yes, then jump to 12
11	C03	GOTO 03	No, then jump to 03
12	013	MOV 1,3	Determine whether the points (output 1) are set to straight (0) or branch (1).
13	313	ANDI 1,3	
14	E1D	BRZ 1D	
15	FD0	DIN 0	Take over value of light barrier
16	320	ANDI 2,0	By ANDI 2,0 take over only value of L2
17	E03	BRZ 03	If L2 equals 0, then jump to 03

18	3E1	ANDI E,1	By ANDI E (1110) set points to „straight“
19	12F	MOVI 2,F	By OR operation of 2 (0010) release block 2
1A	AF1	OR F,1	
1B	102	MOVI 0,2	Clear storage of L1
1C	C03	GOTO 03	Jump to 03
1D	FD0	DIN 0	Take over value of light barriers
1E	340	ANDI 4,0	By ANDI 4,0 take over only value of L3
1F	E03	BRZ 03	If L3 equals 0, then jump to address 03
20	11F	MOVI 1,F	By OR operation of 1 (0001) set points to branch
21	AF1	OR F,1	
22	C19	GOTO 19	Jump to 19
23	F01	NOP	
24	F01	NOP	
25	161	MOVI 6,1	Additional program for manual operation
26	FE1	DOT 1	
27	FF0	KIN 0	
28	171	MOVI 7,1	
29	FE1	DOT 1	
2A	FF0	KIN 0	
2B	C25	GOTO 25	

For fully automatic model-railway operation, the program is started with HALT – NEXT – 00 – RUN. Two trains expediently stand on the two tracks at the station (block 1); a third train stands opposite in block 3. Logically, no locomotives or wagons may stand on the points. After switching on the running current, fully automatic model-railway operation sets in. The display is switched off, because the computer must partly react (compute) very quickly.

Circuit diagram, second expansion stage, with an additional outer loop



Second expansion stage – model-railway control

Without changing the entire electrical wiring, our small layout can also be run in „manual operation“ at any time. In this case we stop the program at address 25 with HALT – NEXT – 25 – RUN. Track blocks 2 and 3 are now no longer monitored, but are permanently switched on. For the point setting, any number key can be pressed.

If the green RESET key on the computer board is pressed, all outputs are switched off immediately. This blocks track blocks 2 and 3. The RESET key can be used as an „emergency-stop key“.

Even with this small and simple layout, the manifold possibilities of a computer control can be recognized, which with the means customary up to now (block-post control, etc.) are not achievable with this precision. As with all automated function sequences, care must be taken that, for example, all wagon couplings work perfectly, so that no unforeseen breakdowns occur due to wagons that have come to a standstill.

Our small model-railway layout becomes even more interesting when it is completed by an additional outer loop.

With additional track material and two further pairs of points, the layout is built up as shown in the figure. The inner loop corresponds unchanged to the previous setup. All light-barrier and relay connections remain in place. It is only necessary to ensure that the light barrier L1 is arranged to the right of the newly inserted point.

Since we still have one computer output free, we can set up a „stopping point“ on the new outer loop in the station area, which is driven by a further relay via the 4th computer output. This outer loop is intended for long-distance trains, which only occasionally stop at the station for a certain time and then automatically continue. To achieve this, the already entered program is completed from address 2C. Additional program entry after HALT – NEXT – 2C:

Additional program: model-railway control, 2nd stage

Address	Code	Mnemonic	Explanations
2C	516	ADDI 1,6	Stop time at station, outer loop
2D	FB7	ADC 7	
2E	977	CMPI 7,7	
2F	D32	BRC 32	
30	371	ANDI 7,1	
31	C08	GOTO 08	
32	188	MOVI 8,8	
33	A81	OR 8,1	
34	C08	GOTO 08	
35	FD0	DIN 0	Stop before station, inner loop
36	549	ADDI 4,9	
37	FBA	ADC A	
38	E3A	BRZ 3A	
39	F07	RET	
3A	12F	MOVI 2,F	
3B	AF1	OR F,1	
3C	C03	GOTO 03	

Before we start the program, the following changes must be made to the first program part: with HALT – NEXT – 06 make the first command change, and then move on with the NEXT key to the next address to be changed, and after the last command change do not forget to press the NEXT key once more:

At address 06, change the command entry to **E2C** (BRZ 2C)

At address 15, change the command entry to **B35** (CALL 35)

At address 1D, change the command entry to **B35** (CALL 35)

All other functions remain unchanged. Program start with HALT – NEXT – 00 – RUN.

On the outer loop, a fourth train can be operated either with the same running-current transformer with which the three other trains already run, or with an additional running-current transformer. On our layout, „a busy hustle and bustle“ will now develop. Since the train on the outer loop only stops occasionally, the four trains running simultaneously create a fascinating impression for a layperson who does not know that it is a computer-controlled layout.

The stopping time of the express train on the outer loop is variable. It can be shortened if the command at address 2C is changed by entering 526 or 536 or 546, etc.

When all four trains are travelling in the same direction, a train can be switched from the inner loop to the outer loop by manual operation or by the usual remote-control point switches. As soon as a train leaves the inner loop, a train immediately following it stops in track block 2 for about 5 seconds, which gives enough time for the points to reset. This time can also be extended if the command at address 36 is changed by entering 539 or 529 or 519. Since there is no light-barrier monitoring on the outer loop, only one train should run there at a time, i.e. when switching from the inner loop to the outer loop, one of the two trains temporarily running there should soon switch back to the inner loop.

The additional program contains, from address 2C to 34, a two-digit counter by which computer output 4 is switched on or off depending on the counter reading. Since this is a „random counter“, the train running on the outer loop stops at irregular intervals. From address 35, there likewise follows a counter which releases block 2 (computer output 2) after about 5 seconds, and not, as normally, after a train has passed through light barrier L2 or L3. Thus 2 programmed timers have been integrated into the program sequence.

Of course, we can expand our layout further with sidings not monitored by the computer, etc. On relays No. 2 and 3, one connection socket (No. 1) each is still free, to which, for example, further relays can be connected, in order to switch a warning-flasher unit at the level crossing or lighting within the model town on and off. Additional signals (coupled with the point setting) could be connected to relay No. 1. On the computer itself, we have used up all inputs and outputs, which is why additional supplementary elements are necessary for further computer functions.

BUSCH will develop additional memory modules especially for model-railway enthusiasts, so that for a further expansion the four inputs and outputs now available can be multiplied. Interested model-railway enthusiasts, please note the announcements in the new BUSCH model-railway catalogue, which will be available from late summer 1982. You will be amazed at what can still be „got out“ of the Microtronic computer system.

Parts list: Additional components for model-railway control

Taking into account the electronic modules contained in the 2090 pack, the following individual parts are also needed for building the model-railway control:

4	modules, resistor 470 Ohm	No. 20684
7	modules, resistor 1 k Ohm	No. 20685
2	modules, resistor 4.7 K Ohm	No. 20687
7	modules, NPN transistors	No. 20611
1	plug-in board (for modules)	No. 20782
1	bag of cable pieces in various lengths	No. 20785
1	bag of 48 yellow plastic plugs	No. 20795

You can order these parts from your Microtronic specialist dealer or directly from us (against prepayment of the amount to our postal giro account). Please use the enclosed order form, in which you will find all the necessary details. If necessary, request a new order form from us free of charge.

The remaining necessary parts:

4	IR light barriers	No. 5962
4	special relays	No. 5964

should be in stock in the model-railway departments of good specialist toy shops.

Can the Microtronic computer system do even more?

It is impossible to give an example for every application in this instruction book. The book would never come to an end, because the possible uses of such a microcomputer cannot be surveyed.

Therefore the most exciting part of working with the Microtronic computer system now begins:

We consider which tasks the computer can take over. We put our considerations into the order that can be carried out by the computer, and we develop the programs required for this. For many applications, modified program parts from the instruction books can be used. With intensive work, however, we will discover yet infinite programming possibilities. It is up to us to make use of these possibilities. We will find that our microcomputer not only provides a serious and satisfying leisure activity, but that it can also be used for many practical applications. For this, BUSCH will over time bring further supplements onto the market, whereby the Microtronic system is adapted to the current state of the art.

Microtronic Programming Contest

The purpose of the Microtronic computer system is to master the possibilities of microcomputer technology and to arrive at ever new ideas and programs.

In order to coordinate the exchange of experience, it is planned to hold a contest under the motto:

„The most interesting program suggestions for the Microtronic computer system“

Submissions are possible at any time. The program suggestions awarded a prize by BUSCH will be published in future Microtronic program books. Every program that comes to publication is rewarded by payment of a prize of DM 100.–.

Conditions of participation for the BUSCH Microtronic Program Contest

Eligible to participate are all users of a Microtronic computer system, with the exception of employees of the company BUSCH GmbH.

The program suggestions to be submitted must be new, i.e. they should not represent modified variations of already published programs, and they should fall within the scope of the following 3 categories:

1. Program suggestions which can be carried out with the Microtronic computer system without additional accessories.

For the functioning of these programs, only the Microtronic computer and the supplementary parts contained in the 2090 box may be used.

2. Program suggestions using additional BUSCH Electronic components.

For these program functions, all the individual parts of the BUSCH Electronic Studios, BUSCH switching relays and the supplementary elements coming further onto the market from BUSCH may be used.

3. Program suggestions using generally purchasable components.

For these program functions, general, standard components available everywhere may be included. The purchase price (general retail price) of the additionally necessary parts should not exceed the amount of DM 80.–.

The programs submitted for a contest entry should be laid out in the manner of the Microtronic program listings.

A short functional description is required, necessary program explanations, and for larger programs a program flow chart.

If additional electronic components are needed, the inclusion of a circuit and construction plan is necessary.

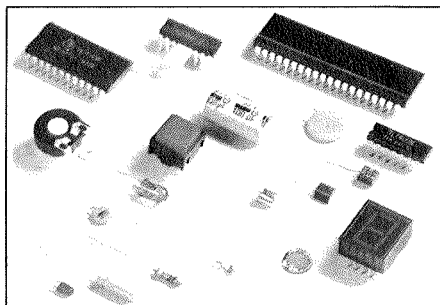
Please address your program suggestions to:

BUSCH GmbH & Co. KG
Microtronic Contest Department
P.O. Box 1360
6806 Viernheim/W-Germany

Concept and logic: The electronic experimentation system from

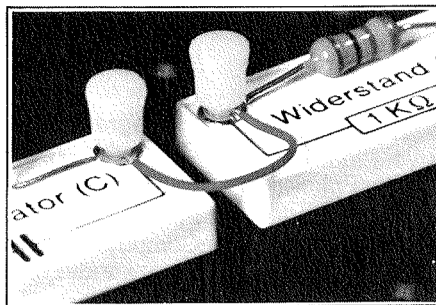


In collaboration with the electronics magazine



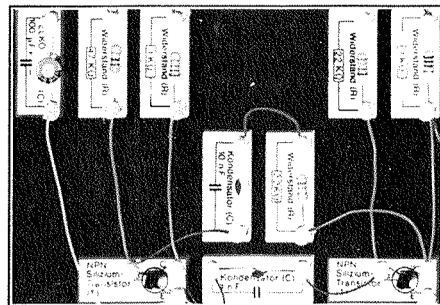
Modern industrial components...

From the simple resistor through capacitors, diodes, photocells, transistors, loudspeakers, etc. up to the IC and microprocessor.



... mounted on BUSCH electronic modules

Ready to install and function-tested. Understandable for everyone thanks to component labelling and circuit diagram.



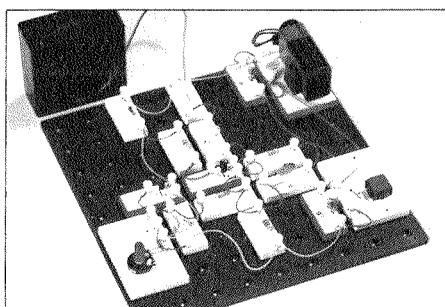
Perfect experimentation technique...

For building the circuit, place the modules on the plug-in board and wire them with absolutely reliable contact using the protected BUSCH lead-clamping technique.

Compact-Studio 2060

Basic set
from 10 – 12 years

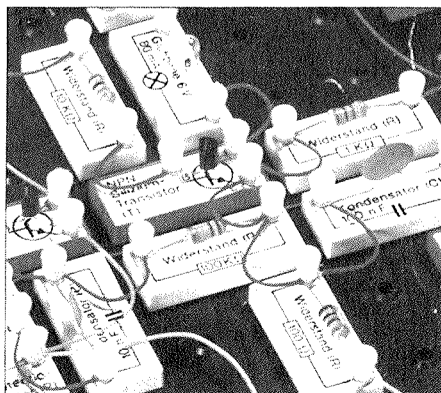
Expandable with the Electronic Studios
2061, 2070, 2090.



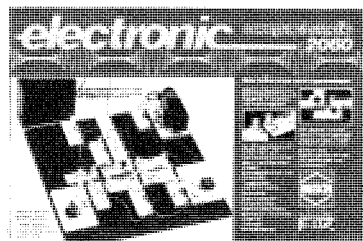
Start experimenting immediately without prior knowledge! The inexpensive starter pack for those who want to try it out first and expand it further later.

A detailed instruction book with over 120 illustrations and circuit diagrams shows, understandably for everyone, the construction and function of 40 interesting experiments and device circuits.

Device circuits: From the simple electric circuit to the basic light-organ circuit. Among others: rain-warning system, voltage tester, alarm systems, dimmer circuit, automatic time and delay circuits, flashing-light circuits, electronic two-tone horn, siren wailing, tone generators, acoustic step detector, capacitive proximity detector, remote-controlled electronic relay, AF sound amplifier, Morse-code generator, electronic mini-organ.



Contents: The pack contains everything necessary to begin, such as: mini loudspeaker box, electronic modules with transistors, potentiometer, capacitors, resistors, push-buttons, light bulb, battery holder, plug-in board, cable, stripped cable pieces in various lengths, clamp plugs, etc. (approx. 100 individual parts).



Supplementary pack Power supply 2059



Suitable for all Electronic Studios.



For everyone who experiments for longer or uses assembled device circuits (e.g. clock radios) in continuous operation, the purchase of the special power supply 2059 is recommended. The plug-in module of the power supply is used within the circuits in the same way as the otherwise usual battery module. The device saves batteries and always ensures the correctly dosed voltage.

The power supply 2059 is a compact, double-insulated safety transformer. The integrated electronic stabilization ensures uniform voltage output with hum filtering (important for radio experiments).

Input (primary) 220 V AC.
Output (secondary) 9 V DC, maximum 200 mA.
Short-circuit-proof, tested according to VDE guidelines.

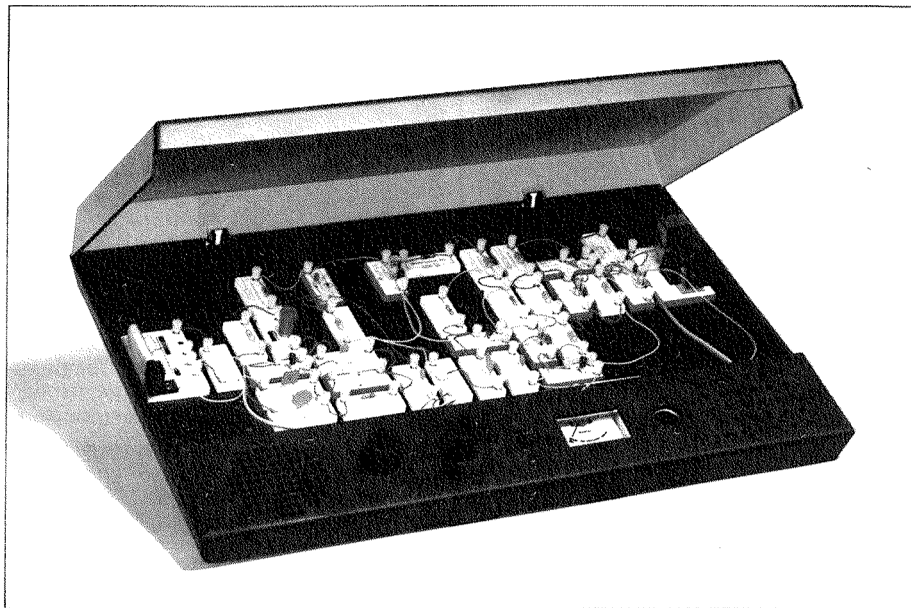
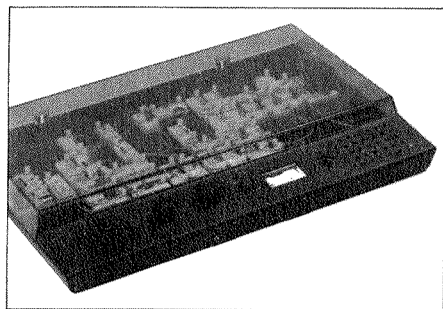
Studio-Center 2070

Basic set

from 12 – 14 years

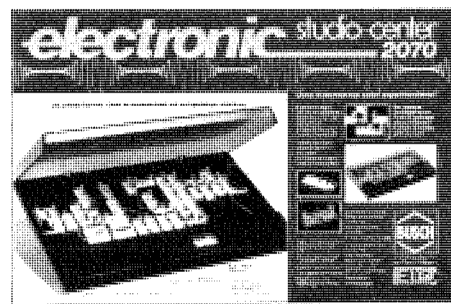
Expandable with the Electronic Studios 2060, 2072, 2075 and 2090.

The large Studio-Center 2070 is the centrepiece of the BUSCH Electronic Studio series. It offers all-round experimentation possibilities in all areas of modern electronics. By building circuits in the studio housing, over 130 realistically functioning devices are created. A comprehensive instruction book with approx. 370 illustrations, construction plans understandable for everyone, and professional circuit-diagram representations. An excellent thing, also for those who do not only want to experiment.



Device circuits e.g.: medium-wave radios with push-pull output stage, long-wave (LW) receiver, short-wave (SW) radio, VHF/FM broadcast receiver, 1-channel light organ, acceleration and phon measuring devices, cable and magnetic-field detectors, highly sensitive telephone monitoring circuit and bugging device, transmitter experiments, electronic piano and Hawaiian guitar, automatic telephone dial, rhythm device, car alarm system, reaction-time meter, digital counter, electronic dice, hearing-ability tester, acoustically remote-controlled switch, shot detector, reaction-time test device, electronic roulette, metal detector, automatic warning-flasher unit, light barrier with acoustic continuous alarm, electronic „nerve saw“ and much more. In addition, all the device circuits listed for 2060 and 2065 are possible.

Contents: Electronic Studio-Center with smoked-glass-coloured cover hood and instrument board with built-in loudspeaker, potentiometer, variable capacitor, slide switch, universal measuring instrument and standard phono socket. In addition, a rich selection of electronic modules with transistors, radio IC (integrated circuit), diode and light-emitting diode, electrolytic and disc capacitors, photoresistor, ferrite antennas, choke coil, potentiometer, resistors, push-buttons, earphone, light bulb, wires, cable, cable pieces and clamp plugs.

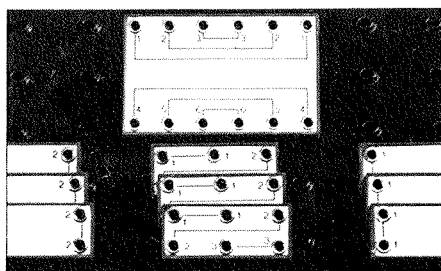


Supplementary pack 2079

Plug-in modules for self-assembly.

Possibilities of endless further experimentation: attach any electronic components to the modules without soldering.

Contents: 10 modules and 45 clamp plugs. For the easy self-assembly of 2-, 3- or 6-pin components such as resistors, capacitors, transistors, thyristors, coils, transformers, switching relays, etc.

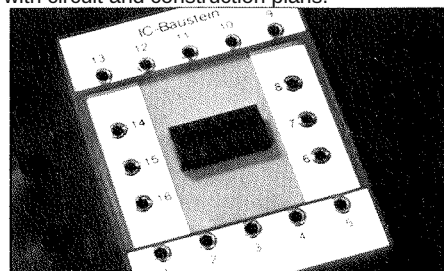


Supplementary pack 2089

IC sockets for self-assembly.

Once initial IC experience has been gained through the experiments with the Electronic Studios 2072 and 2075 and above all with the microcomputer 2090, these self-assemblable IC modules open up endless expansion possibilities.

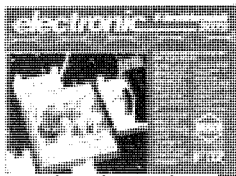
Contents: 2 modules for plugging in any ICs up to 16 connection pins, plus 40 clamp plugs. Detailed instructions with references to the most common IC types, application examples with circuit and construction plans.



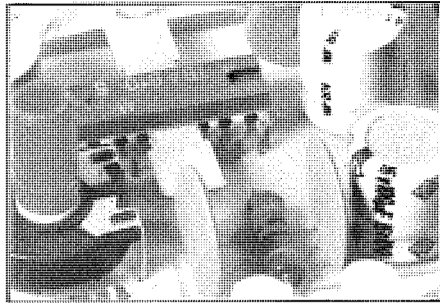
Supplementary pack IC amplifier technology

2072

from 12 years



Ideal as a supplement to the Electronic Studios 2065, 2070 and 2090.



This add-on box „IC amplifier technology“ brings power and the hi-fi super sound!

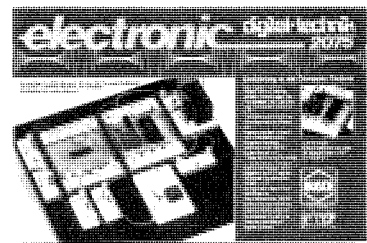
Device circuits: The detailed instruction book with over 50 illustrations and construction plans shows 30 sound-generating extension circuits for broadcast receivers with hi-fi sound, phono amplifier with power output stage, electronic organ, spinet and Hawaiian guitar, rhythm device, intercom systems, hi-fi mono amplifier with treble and bass control, hi-fi stereo amplifier, telephone amplifier, vibrato organ with light control, reverb effects, electronic interval-signal generator, sound transmission by light, etc. A genuine 2-channel hi-fi stereo amplifier is possible by using 2 x 2072.

Contents: On a special module with an integrated board, a high-quality AF power amplifier is built, consisting of an amplifier IC, several electrolytic capacitors, disc capacitors, resistors and a trimmer potentiometer for tuning. With the connection socket also included, playback to large external loudspeakers (hi-fi boxes) is also possible.

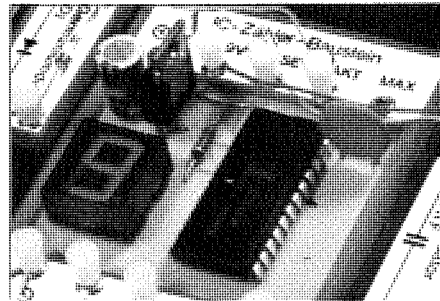
Digital-Technik 2075

Basic set

from 14 years



Expandable with the Electronic Studios 2070 and 2090.



The „Digital-Technik“ studio explains the fundamentals of calculator and computer technology. An independent experimentation kit and at the same time a supplementary kit.

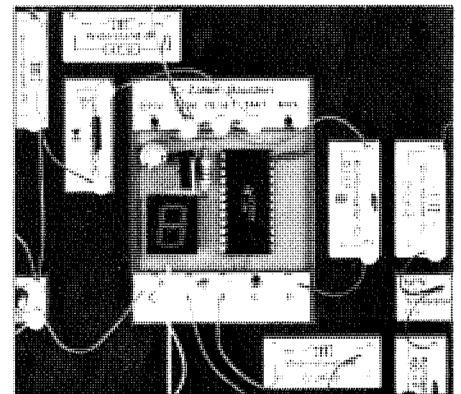
The core is an IC counter module with a 7-segment illuminated-digit display and an integrated circuit with over 350 transistor functions. This IC (a so-called 4-bit counter-latch-decoder) contains, besides a counter, a memory and a decoder for driving the illuminated display.

Digital electronics is conquering ever more fields of application. The best-known examples of this new technology are computers and pocket calculators. However, digital technology is used just as much in consumer electronics – in televisions and large receivers – as in digital clocks or for the program control of washing machines. The latest technical marvel, the microprocessor (see 2090), likewise works according to the basic rules of digital technology.

Device circuits: The detailed instruction book with over 100 illustrations and circuit diagrams brings 30 highly interesting experiments, such as the principle of the digital clock, programmable counter, electronic roulette with illuminated display, memory circuit, digital memory, electronic calculator, digital stopwatch, opto-electronic adder, frequency meter, principle of the synthesizer, opto-acoustic counter, digital-to-analog converter, digital voltmeter, electronic music box, etc.

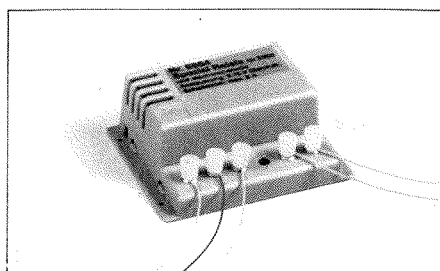
The Digital-Technik studio is at the same time a recommendable preliminary stage to the Microtronic computer system 2090.

Contents: Over 60 individual parts, including a high-quality IC counter module with illuminated display, IC gate module, as well as resistors, electrolytic capacitors, capacitors, light-emitting diode, push-buttons, battery holder, cable and clamp plugs.



Low-current special relay 5964

This relay can be driven by a minimal control current (8–16 V, 50 mA) of the Electronic Studio circuits or by the computer 2090, and switches (in conjunction with, for example, a railway transformer) up to 8 A at 16 V. Ideal for experiments with light-, sound- and time-dependent circuits with the model railway. Also usable in conjunction with the BUSCH IR light barrier / timer 5961 (see the BUSCH model-railway accessories catalogue).



Mains-power switching unit 2087

for all Electronic Studios and the computer 2090.

The switching unit is controlled by the circuits built with the Electronic Studios using the experimentation low voltage (6–9 V), and triggers, via a relay, switching operations in the lighting mains (220 V, max. 1000 watts). For example, switching radios, tape recorders, warning systems, fan heaters, etc. on and off. The high-current side is completely encapsulated and separated from the low-current side, and can therefore be operated absolutely safely even by laypeople and children. Tested according to VDE regulations. Detailed instructions with wiring examples.



2090



In collaboration with
the electronics magazine



Perfect for further experimenting . . .

BUSCH Electronic Studios are the ideal supplement to the Microtronic computer system. For example the Studio-Center 2070 with approx. 130 interesting experiments and device circuits across the entire field of modern electronics.

