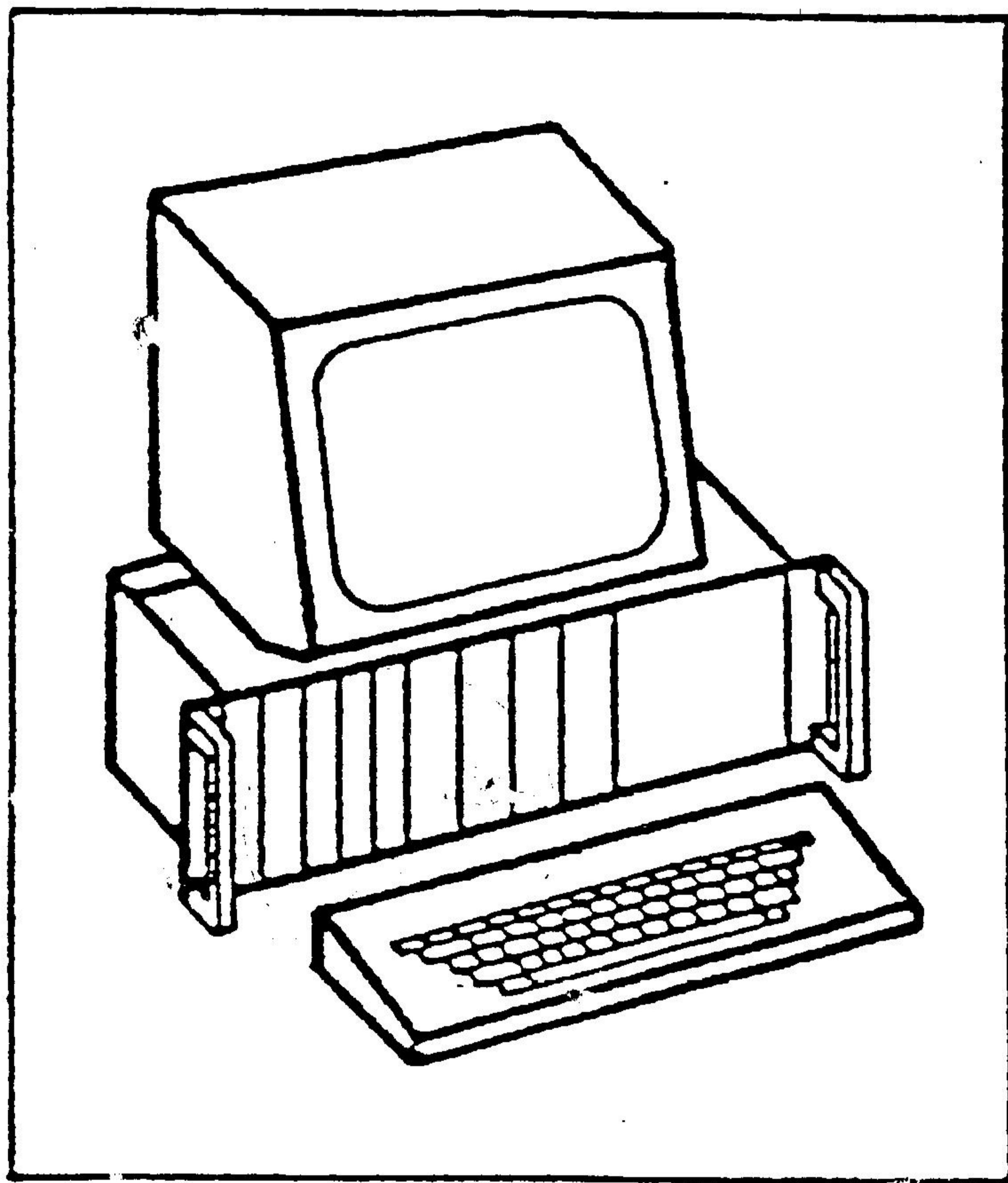


# MIKROCOMPUTER

## TECHNIK



## INHALTSVERZEICHNIS

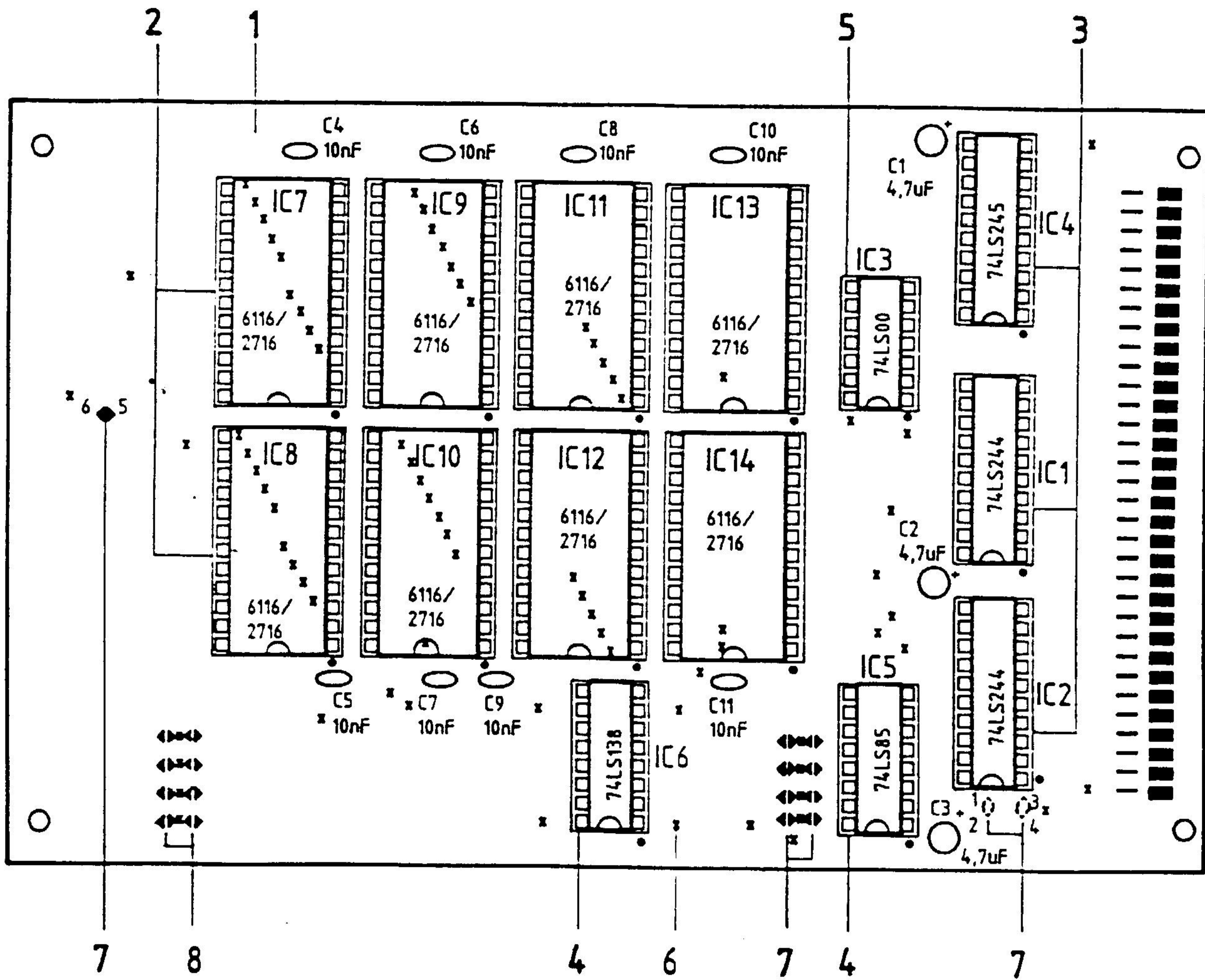
Teil 1	Beschreibung der Kommandos	Seite 1...14
Teil 2	Einige Befehlsbeschreibungen	Seite 15...20
Teil 3	MAT 85 Eigenheiten, Tabellen, Symbole	Seite 21...30
Teil 4	Befehlsliste, verschiedene Ausführungen	Seite 31...42
Teil 5	Schaltungen und Bestückungspläne	Seite 43...96

Teil 1 Beschreibung der Monitor-Kommandos

-HELP	S. 1
-MEMORY	S. 2
-PRINT	S. 3
-GO	S. 4
-DISASSEMBLER	S. 5
-NEXT-INSTRUCTION	S. 6
-ASSEMBLER	S. 7
-REGISTER	S. 8
-BREAKPCINT	S.10
-TRACE-INTERVALL	S.11
-IN	S.12
-OUT	S.13
-SAVE	S.14

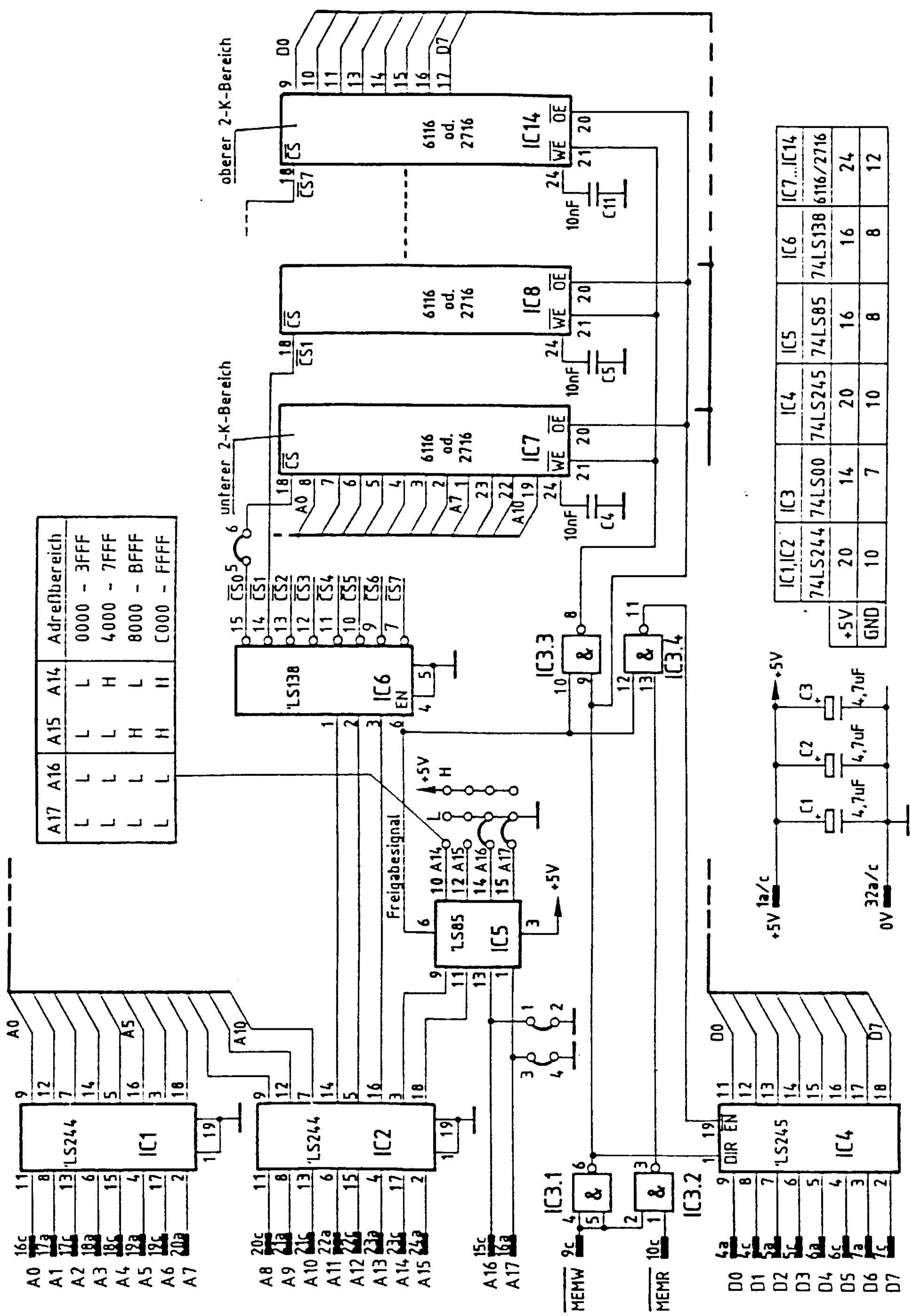


# Bestückungsplan

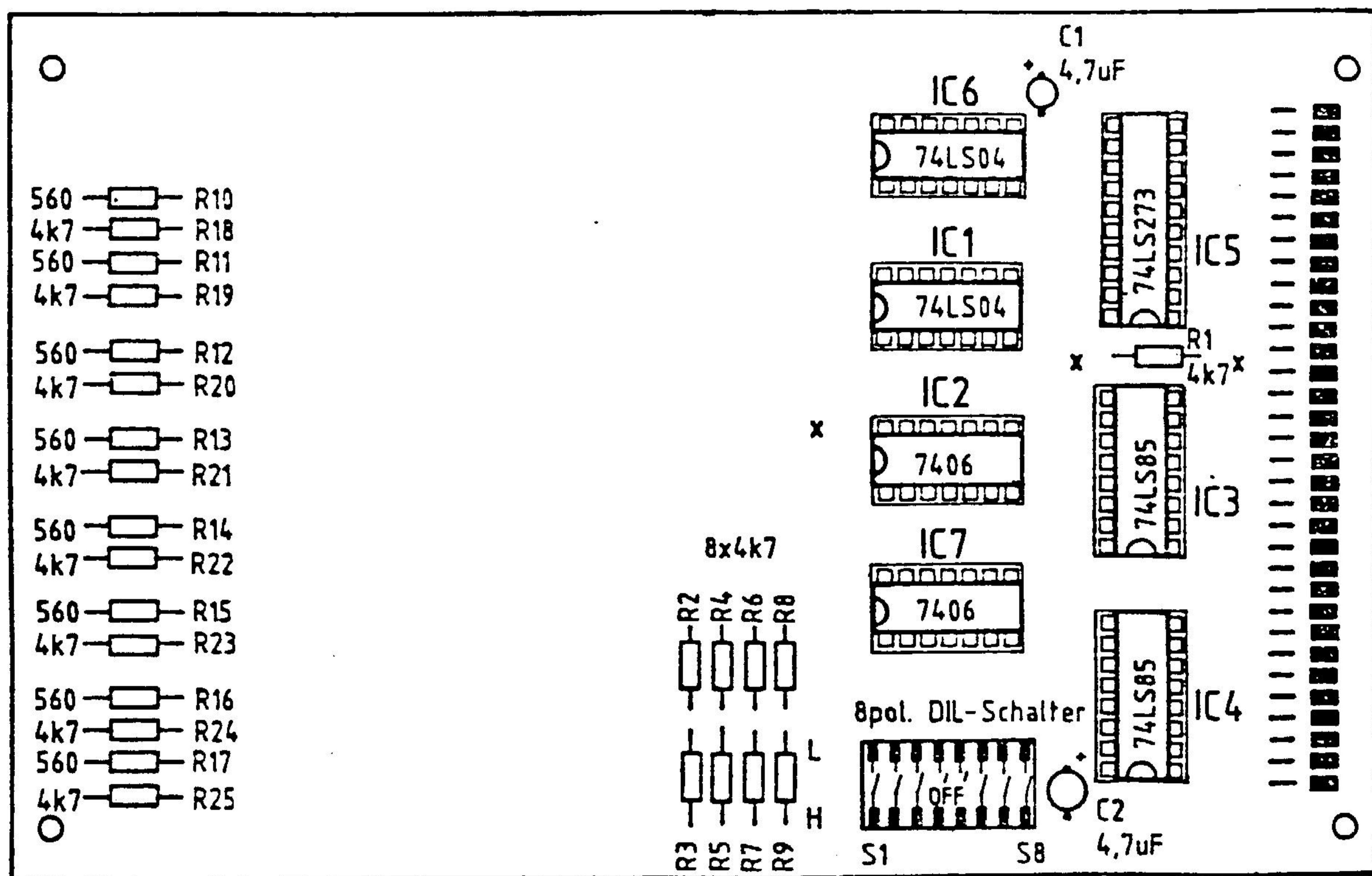


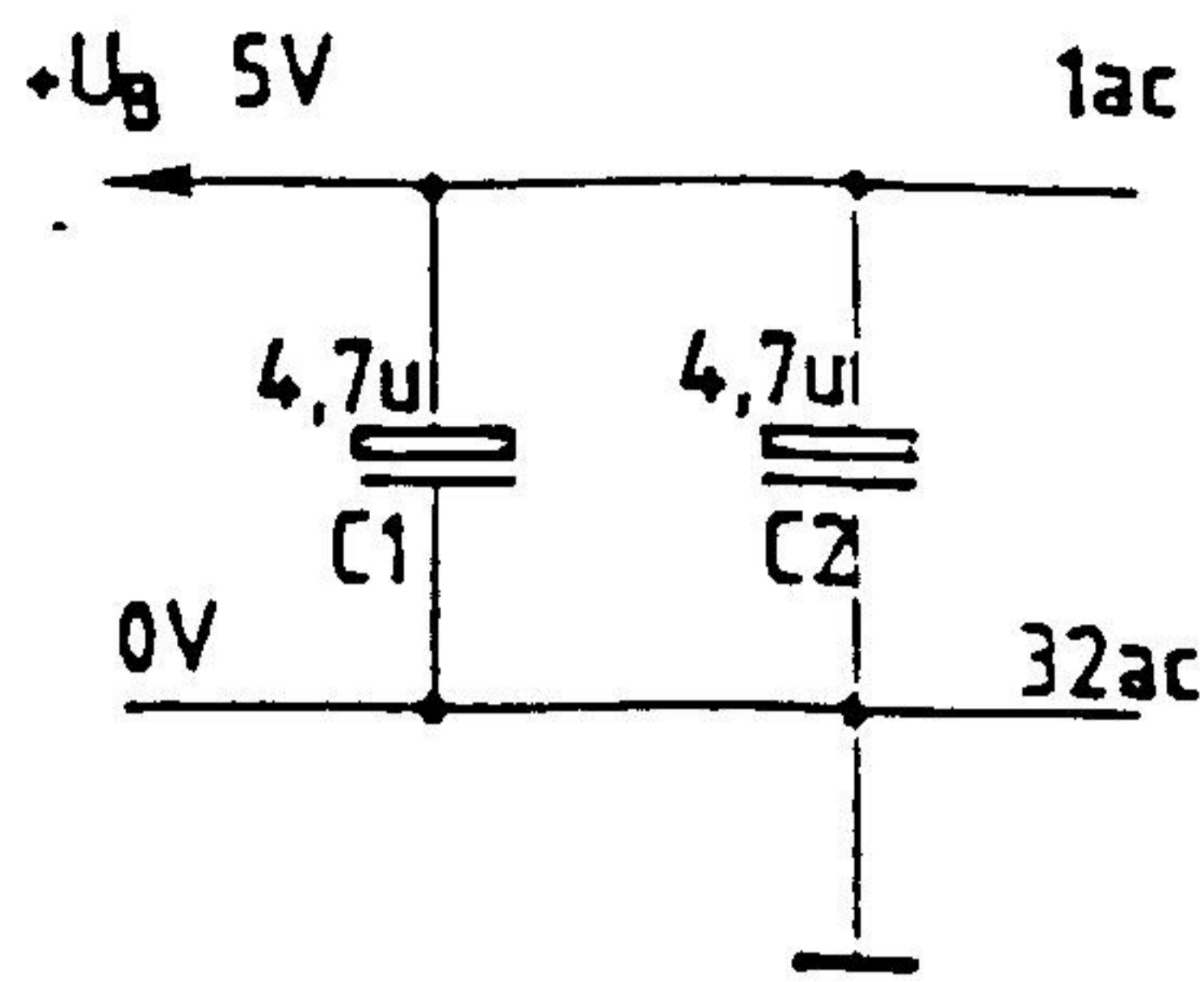
Die Lötbrücken 1-2, 3-4 und 5-6 (Pos. 7) sind zu schließen!





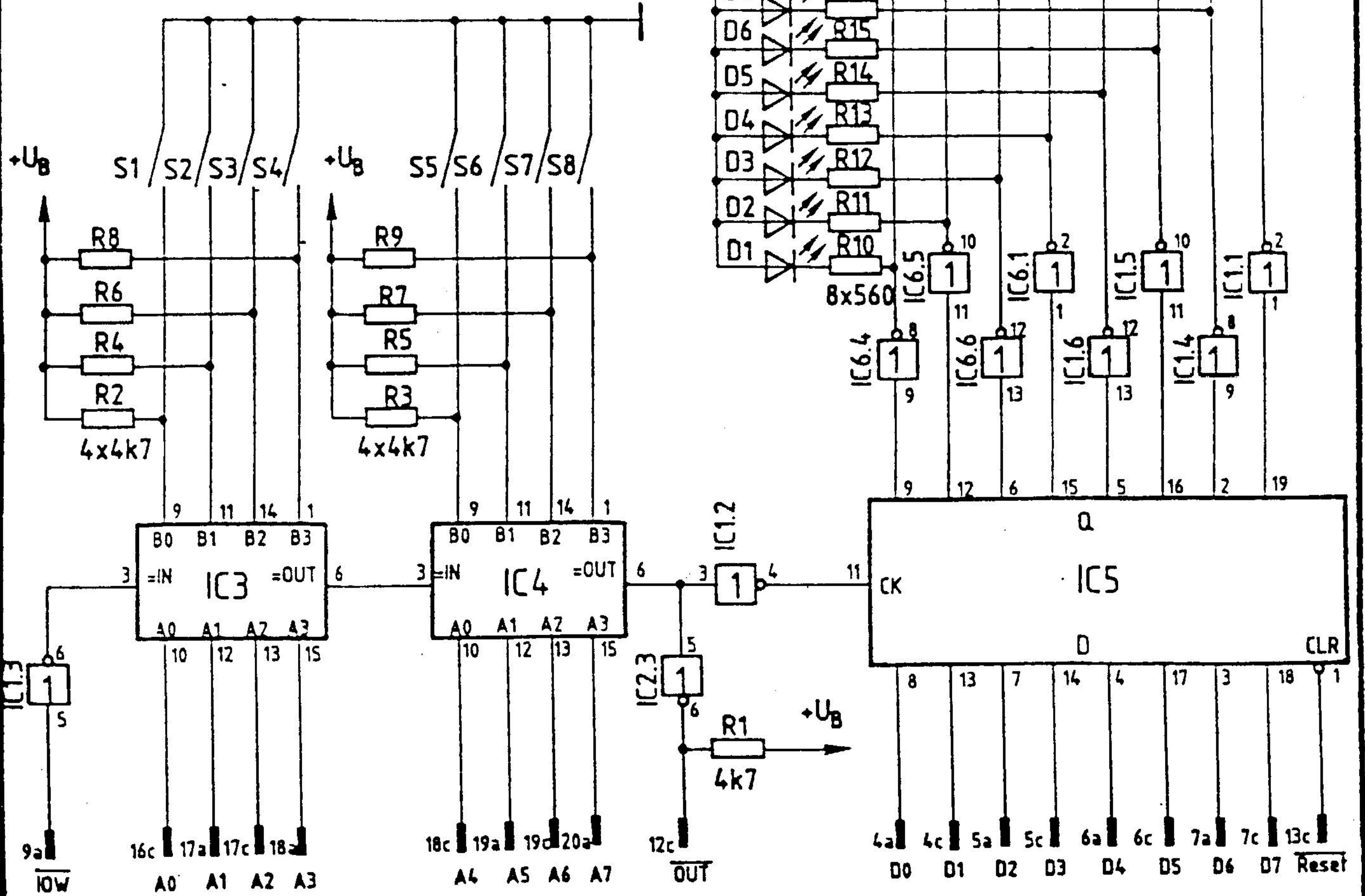
# Bestückungsplan



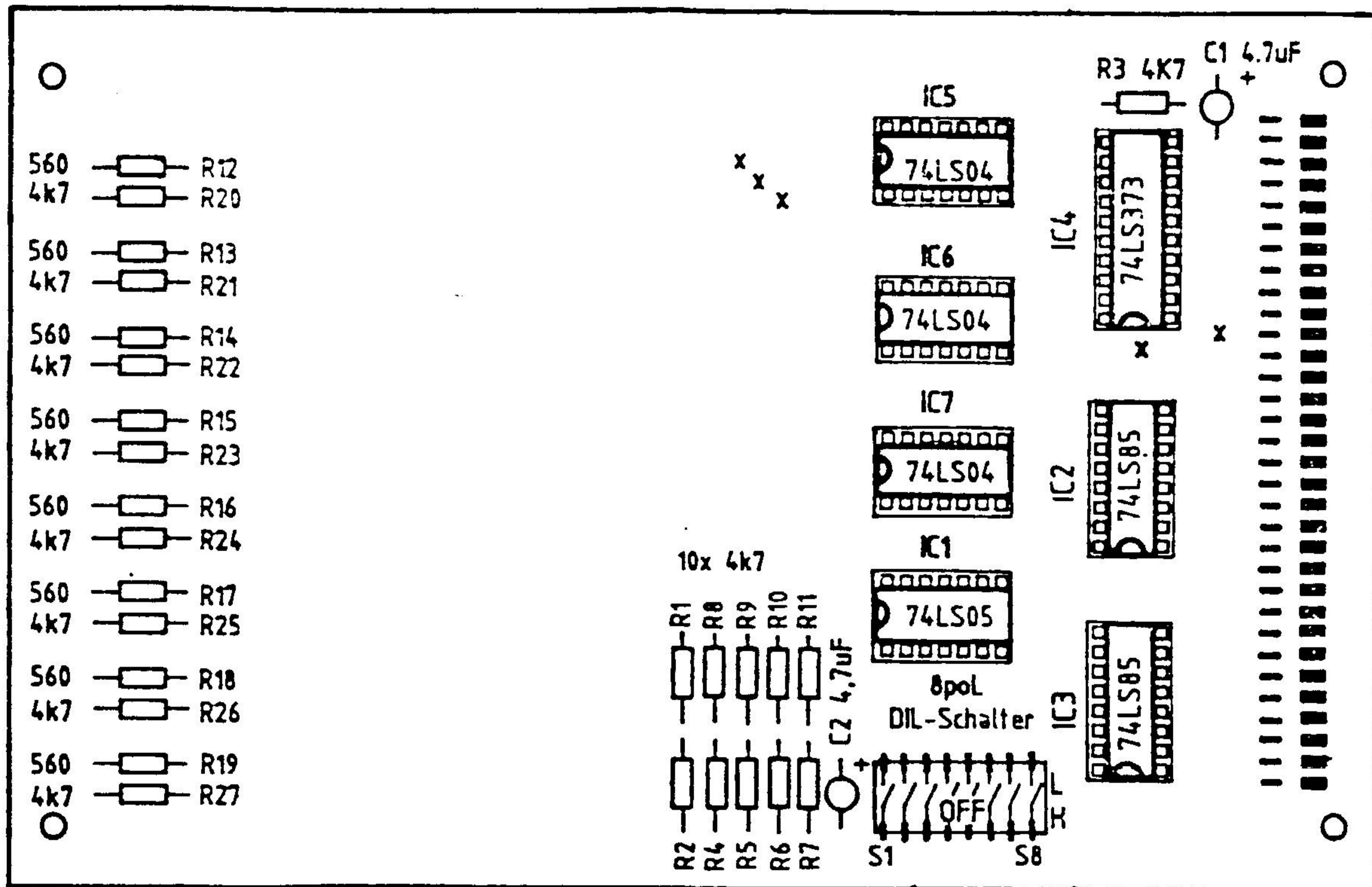


	IC3,4	IC5	IC1,6	IC2,7
	74LS85	74LS273	74LS04	7406
+U <sub>B</sub>	16	20	14	14
0V	8	10	7	7

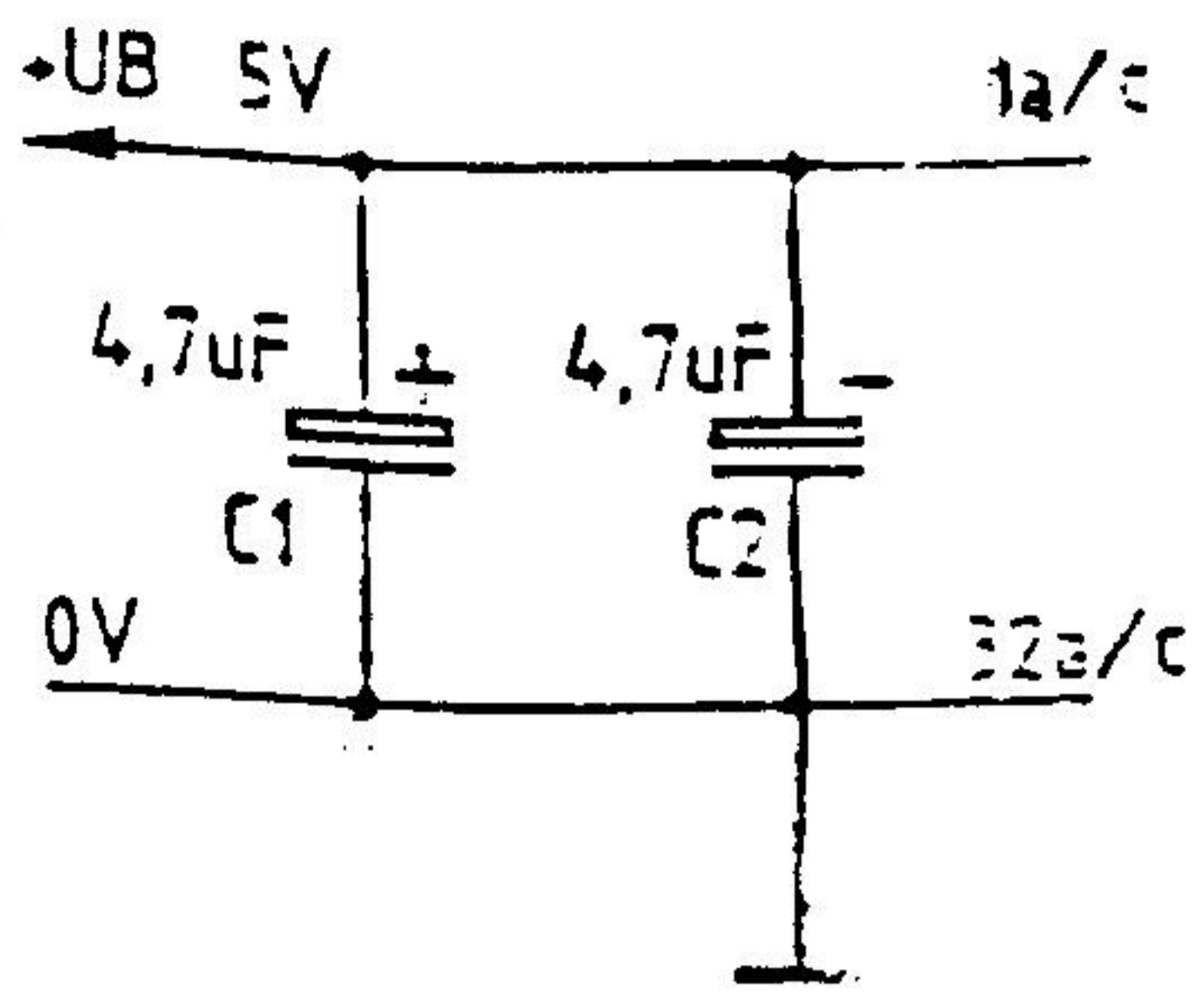
Offen: H-Signal  
Geschl.: L-Signal



# Bestückungsplan

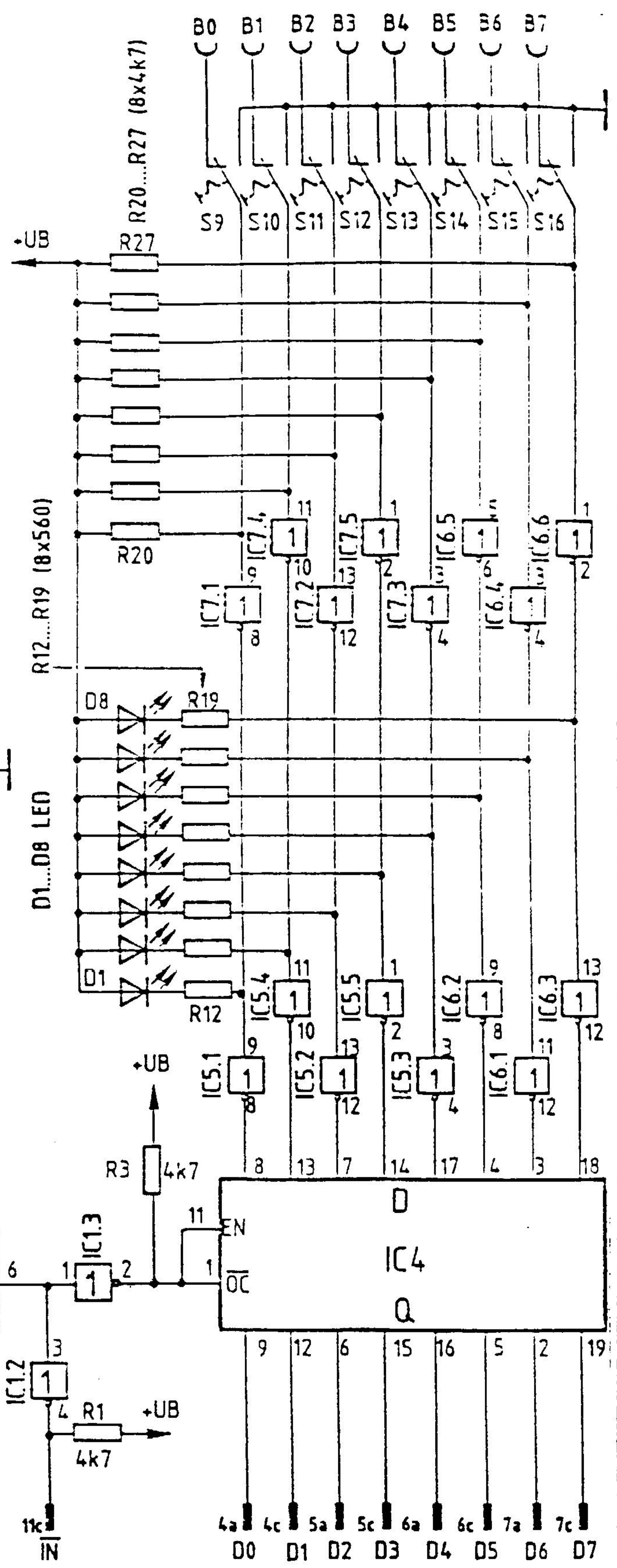
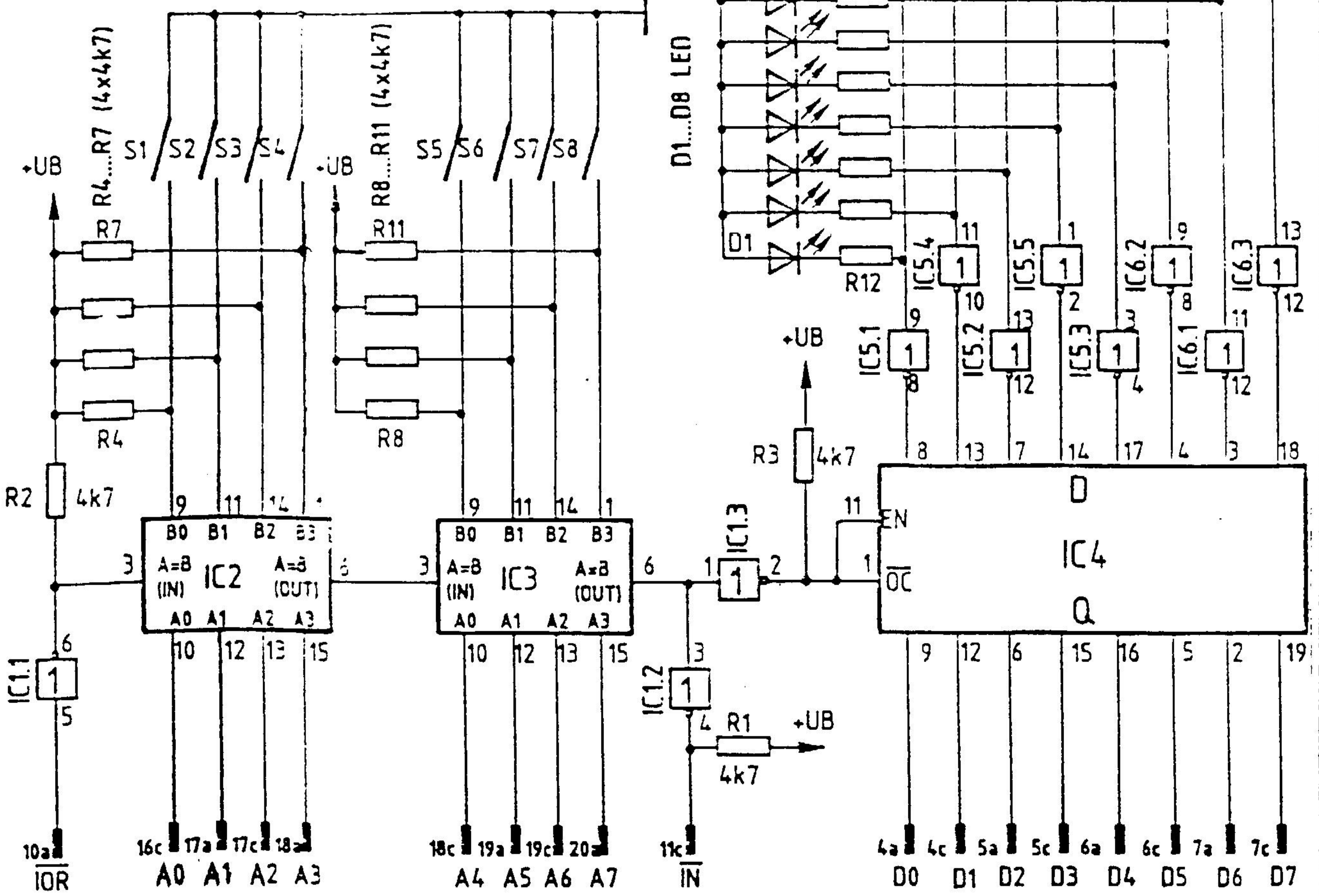




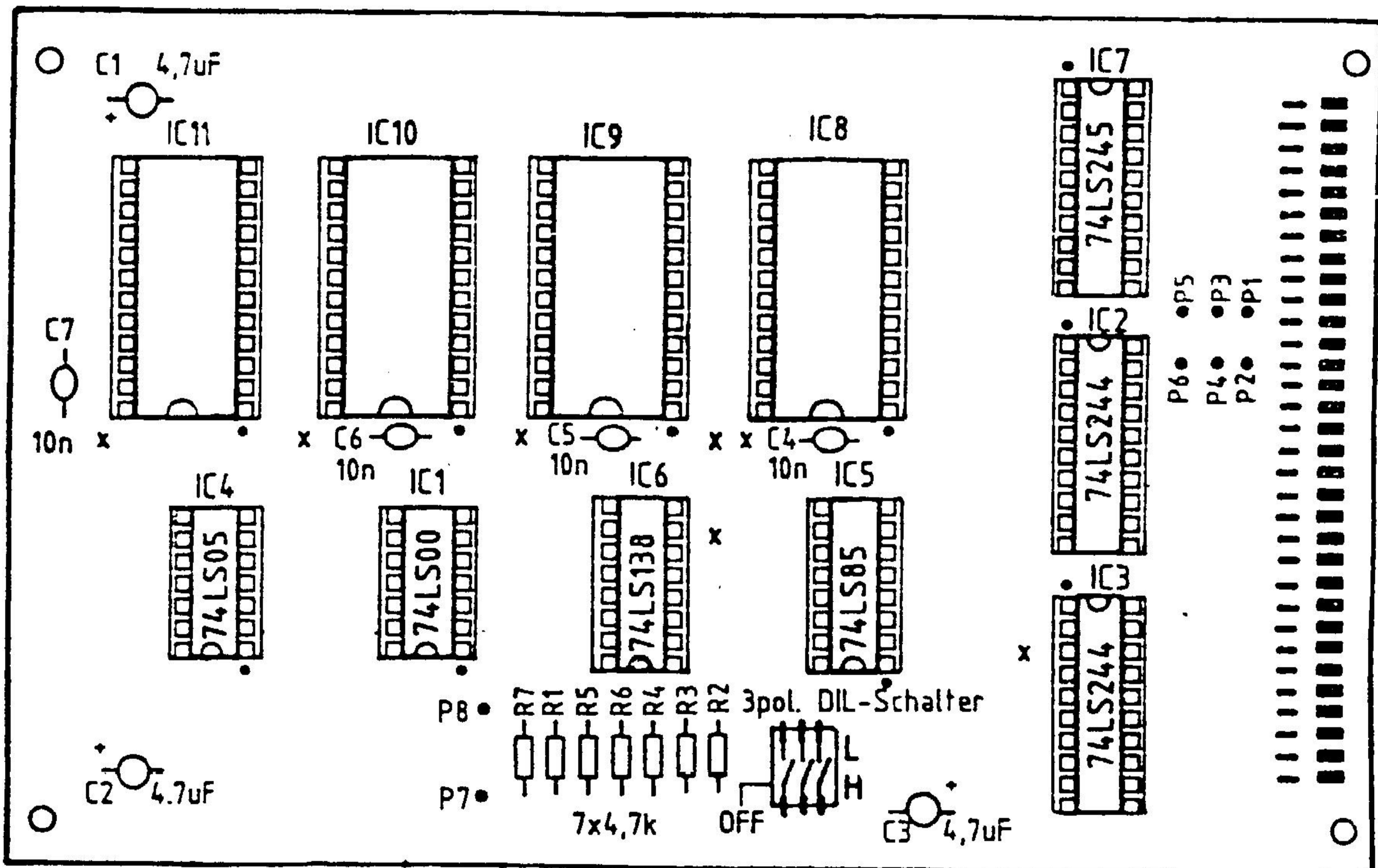


	IC1	IC2,3	IC4	IC5,6,7
	74LS05	74LS05	74LS373	74LS04
+UB	14	16	20	14
0V	7	8	10	7

offen: H-Signal  
geschl.: L-Signal

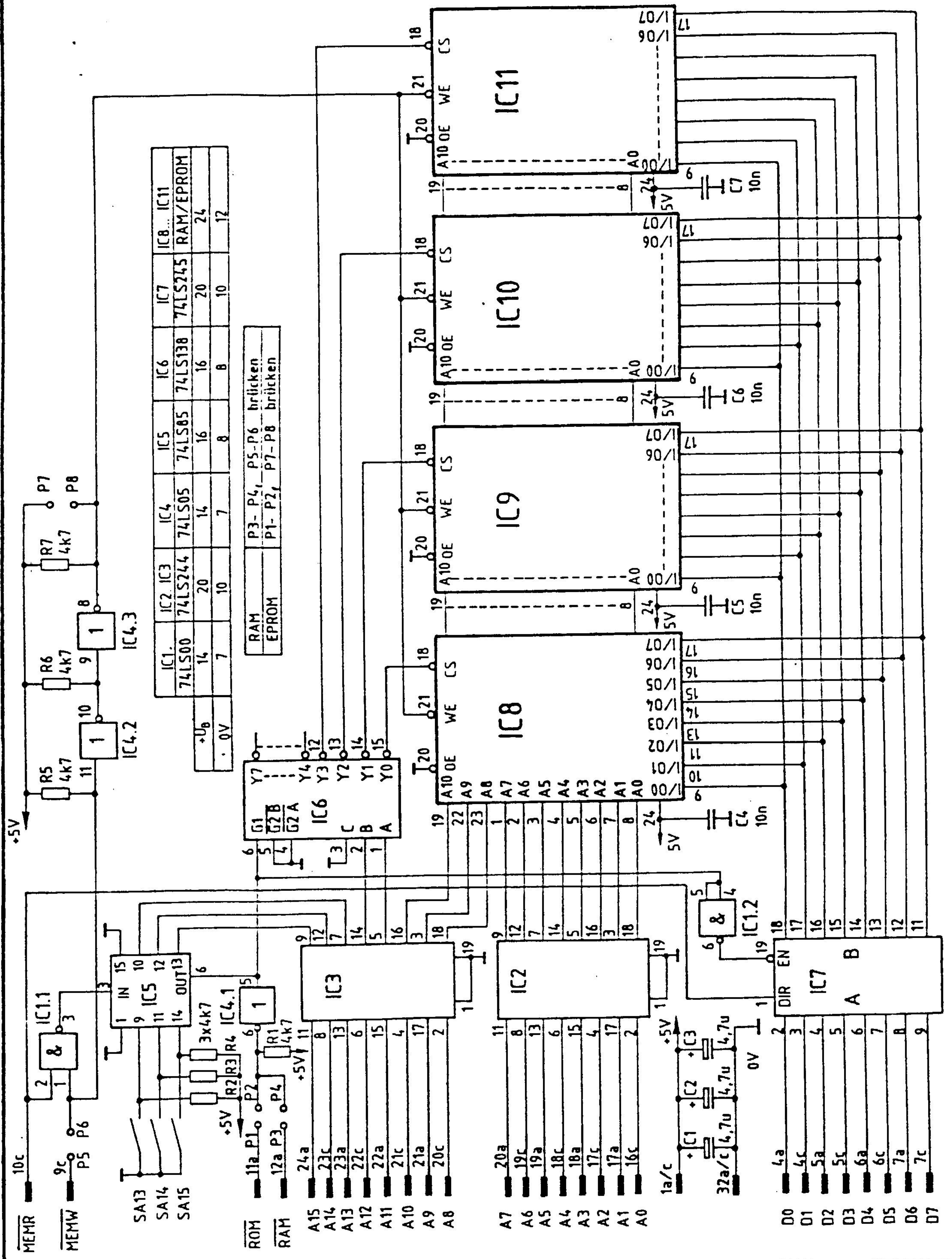


# Bestückungsplan





Stromlaufplan 8-K-RAM/EPROM-Baugruppe

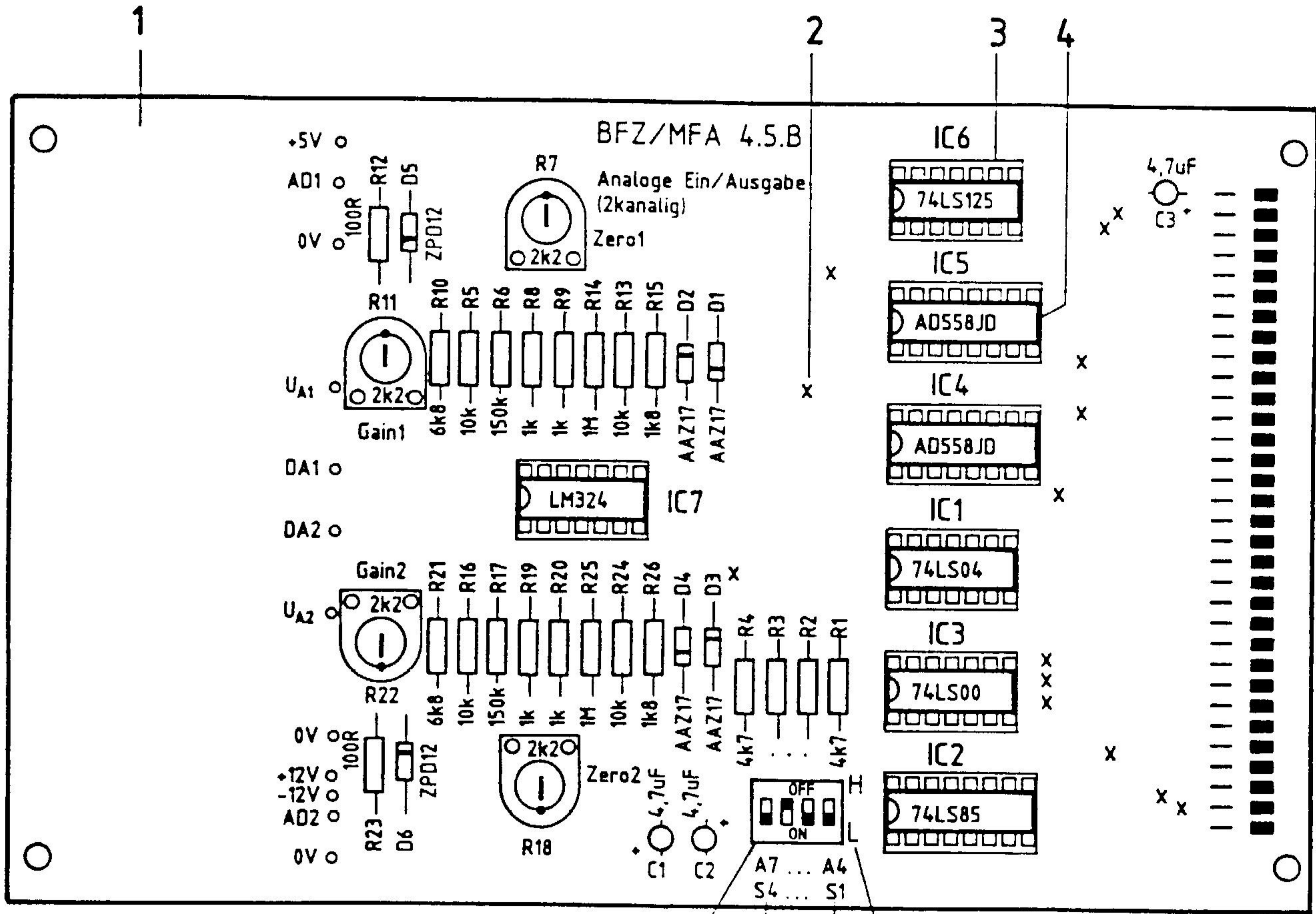


IC1	IC2	IC3	IC4	IC5	IC6	IC7	IC8... IC11
74LS00	74LS244	74LS05	74LS05	74LS05	74LS138	74LS245	RAM/EPROM
14	20	14	14	16	16	20	24
7	10	7	7	8	8	10	12

RAM	EPROM
P3- P4, P5- P6	brücken
P1- P2, P7- P8	brücken

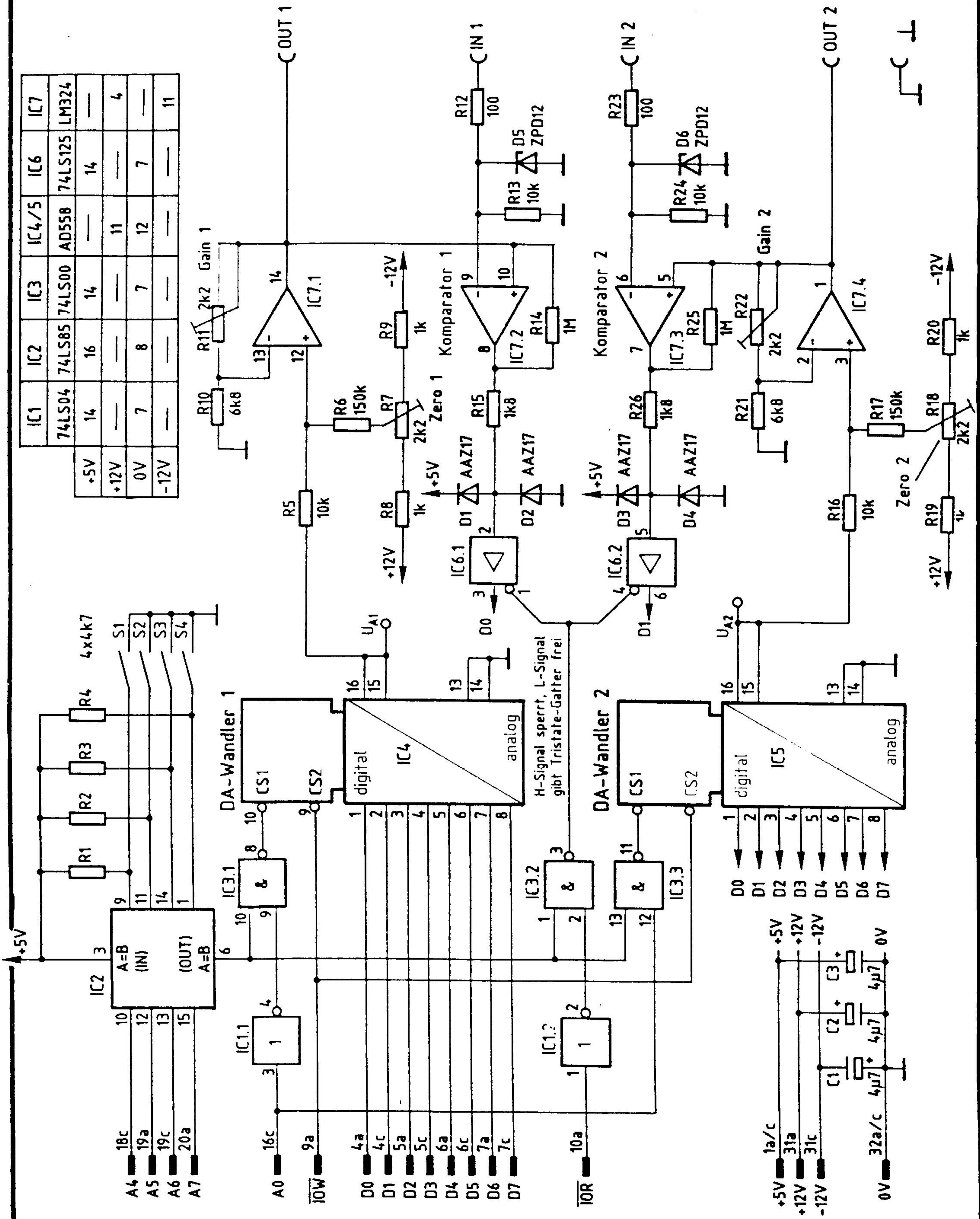
IC6
Y7
G2B
G2A
C
B
A
Y0
Y1
Y2
Y3
Y4
Y12
Y13
Y14
Y15

# Bestückungsplan



5 Beschriften Sie die Karte mit einem wasserfesten Stift





IC1	IC2	IC3	IC4/5	IC6	IC7
74LS04	74LS85	74LS00	AD558	74LS125	LM324
+5V	14	14	—	14	—
+12V	—	—	11	—	4
0V	7	7	12	7	—
-12V	—	—	—	—	11

Teil 2 Einige Befehlsbeschreibungen

-Transport-Befehle	S.16
-Verarbeitungs-Befehle	S.17
-Programmsteuerbefehle	S.18
-Befehle im MNEMO-Code	S.19

T R A N S P O R T B E F E H L E

1. Befehlsbyte		Befehls- länge	Wirkung/Beispiel
binär	hex.		
00111010	3A	3	Der Inhalt der Speicherstelle, deren Adresse auf den dem Befehl folgenden Speicherstellen steht, wird in den ACCU geladen (LDA). Beispiel: <b>3A</b> bewirkt, daß der Inhalt <b>04</b> der Speicherstelle 1304E <b>13</b> in den ACCU geladen wird.
00110010	32	3	Der ACCU-Inhalt wird im Speicher abgelegt. Die Adresse der Speicherstelle steht in den dem Befehl folgenden Speicherstellen (STA). Beispiel: <b>32</b> bewirkt, daß der ACCU-Inhalt <b>FE</b> unter der Adresse 70FEH <b>70</b> gespeichert wird.
00111110	3E	2	Der ACCU wird mit dem Inhalt der dem Befehl folgenden Speicherstelle geladen (MVI). Beispiel: <b>3E</b> bewirkt, daß der ACCU mit <b>F9</b> den Wert F9H geladen wird.
11011011	DB	2	Der ACCU wird mit dem Signalzustand der Eingabe-Baugruppe geladen, deren Nummer auf dem folgenden Speicherplatz steht. (IN). Beispiel: <b>DB</b> bewirkt, daß der Signalzustand an den Eingängen der Eingabe-Baugruppe Nr. 45E <b>45</b> in den ACCU geladen wird.
11010011	D3	2	Der ACCU-Inhalt wird an eine Ausgabe-Baugruppe übergeben. Die Baugruppen-Nummer steht auf der dem Befehl folgenden Speicherstelle. (OUT). Beispiel: <b>D3</b> bewirkt, daß der ACCU-Inhalt <b>AC</b> an die Ausgabe-Baugruppe Nr. ACH übergeben wird.



V E R A R B E I T U N G S B E F E H L E

1. Befehlsbyte		Befehls- länge	Wirkung/Beispiel
binär	hex.		
00101111	2F	1	Der ACCU-Inhalt wird invertiert (komplementiert, CMA).
00111100	3C	1	Der ACCU-Inhalt wird um 1 erhöht, d.h. inkrementiert (INR).
00111101	3D	1	Der ACCU-Inhalt wird um 1 erniedrigt, d.h. dekrementiert (DCR).
11000110	C6	2	Das dem Befehl folgende Daten-Byte wird zum ACCU hinzuaddiert (ADI). Beispiel: <b>C6</b> bewirkt, daß zum ACCU- <b>5B</b> Inhalt 5BH hinzuaddiert wird.
11010110	D6	2	Das dem Befehl folgende Daten-Byte wird vom ACCU-Inhalt abgezogen (SUI). Beispiel: <b>D6</b> bewirkt, daß der ACCU- <b>3D</b> Inhalt um 3DH vermindert wird.
11100110	E6	2	Der ACCU-Inhalt wird Bit-für-Bit mit dem Daten-Byte UND-verknüpft, das in der dem Befehl folgenden Speicherstelle steht (ANI). Beispiel: <b>E6</b> wenn vor der Befehlsausführung im ACCU 83H steht, so bewirkt der Befehl, daß im ACCU der Wert 02E steht.
11110110	F6	2	Der ACCU-Inhalt wird Bit-für-Bit mit dem Daten-Byte ODER-verknüpft, das in der dem Befehl folgenden Speicherstelle steht (ORI). Beispiel: <b>F6</b> wenn vor der Befehlsausführung im ACCU 83H steht, so bewirkt der Befehl, daß im ACCU der Wert 8FH steht.





P R O G R A M M S T E U E R B E F E H L E

1. Befehlsbyte		Befehls- länge	Wirkung/Beispiel
binär	hex.		
11000011	C3	3	Die Programmabarbeitung wird an der Speicherstelle fortgesetzt, deren Adresse in den dem Befehl folgenden Speicherstellen steht (JMP). Beispiel: <span style="border: 1px solid black; padding: 2px;">C3</span> bewirkt, daß der nächste <span style="border: 1px solid black; padding: 2px;">12</span> Befehl von der Speicherstelle <span style="border: 1px solid black; padding: 2px;">F7</span> gelesen wird.
11001010	CA	3	Dieser Sprungbefehl zu der Speicheradresse, die in den folgenden Speicherstellen steht, wird nur ausgeführt, wenn die letzte Rechenoperation <u>Null</u> ergab (JZ). Beispiel: <span style="border: 1px solid black; padding: 2px;">CA</span> bewirkt, daß der nächste <span style="border: 1px solid black; padding: 2px;">12</span> Befehl nur dann von der Speicherstelle <span style="border: 1px solid black; padding: 2px;">F7</span> gelesen wird, wenn das Zero-Bit 1 ist. Sonst wird der Sprungbefehl ignoriert.
11000010	C2	3	Dieser Sprungbefehl zu der Speicheradresse, die in den folgenden Speicherstellen steht, wird nur dann ausgeführt, wenn die letzte Rechenoperation <u>nicht Null</u> (Zero-Bit=0) ergab (JNZ).
01110110	76	1	Dieser Befehl bewirkt, daß die Befehlsabarbeitung gestoppt wird. Nur ein RESET (oder Interrupt) kann die Befehlsabarbeitung wieder einleiten (HLT).



### Transportbefehle

- LDA (3A) = Load ACCU direct,  
lade den ACCU direkt
- STA (32) = Store ACCU direct,  
speicher den ACCU direkt
- MVI A (3E) = Move immediate,  
bewege unmittelbar in den ACCU
- IN (DB) = Input,  
Eingabe
- OUT (D3) = Output,  
Ausgabe

### Verarbeitungsbefehle

- CMA (2F) = Complement ACCU,  
komplementiere (invertiere) ACCU
- INR A (3C) = Increment Register,  
inkrementiere Register A (ACCU)
- DCR A (3D) = Decrement Register,  
dekrementiere Register A (ACCU)
- ADI (C6) = Add immediate to ACCU,  
addiere unmittelbar zum ACCU
- SUI (D6) = Subtract immediate from ACCU,  
subtrahiere unmittelbar vom ACCU
- ANI (E6) = And immediate with ACCU,  
UND unmittelbar mit ACCU
- ORI (F6) = Or immediate with ACCU,  
ODER unmittelbar mit ACCU

### Programmsteuerbefehle

- JMP (C3) = Jump unconditional,  
springe unbedingt (immer)
- JZ (CA) = Jump on Zero,  
springe wenn Null (bedingt)
- JNZ (C2) = Jump on no Zero,  
springe wenn nicht Null (bedingt)
- HLT (76) = Halt,  
anhalten



Teil 4 Befehlsliste, verschiedene Ausführungen

-Befehlsliste Hexadezimal	S.32
-Befehlsliste 8080/8085	S.34
Transfer-Befehle	S.34
Arithmetische Befehle	S.35
Logische Befehle	S.36
Sprung-Befehle	S.37
Unterprogramm Befehle	S.38
Sonstige Befehle	S.39
-Befehlsliste 8080/8085, Kurzfassung	S.40

Hex	Mnem	Hex	Mnemo	Hex	Mnemo	Hex	Mnemo
00	NOP	42	MOV B,D	84	ADD H	C6	ADI d8
01	LXI B,d16	43	MOV B,E	85	ADD L	C7	RST0
02	STAX B	44	MOV B,H	86	ADD M	C8	RZ
03	INX B	45	MOV B,L	87	ADD A	C9	RET
04	INR B	46	MOV B,M	88	ADC B	CA	JZ a16
05	DCR B	47	MOV B,A	89	ADC C	CB	----
06	MVI B,d8	48	MOV C,B	8A	ADC D	CC	CZ a16
07	RLC	49	MOV C,C	8B	ADC E	CD	CALL a16
08	----	4A	MOV C,D	8C	ADC H	CE	ACI d8
09	DAD B	4B	MOV C,E	8D	ADC L	CF	RST1
0A	LDAX B	4C	MOV C,H	8E	ADC M	D0	RNC
0B	DCX B	4D	MOV C,L	8F	ADC A	D1	POP D
0C	INR C	4E	MOV C,M	90	SUB B	D2	JNC a16
0D	DCR C	4F	MOV C,A	91	SUB C	D3	OUT a8
0E	MVI C,d8	50	MOV D,B	92	SUB D	D4	CNC a16
0F	RRC	51	MOV D,C	93	SUB E	D5	PUSH D
10	----	52	MOV D,D	94	SUB H	D6	SUI d8
11	LXI D,d16	53	MOV D,E	95	SUB L	D7	RST2
12	STAX D	54	MOV D,H	96	SUB M	D8	RC
13	INX D	55	MOV D,L	97	SUB A	D9	----
14	INR D	56	MOV D,M	98	SBB B	DA	JC a16
15	DCR D	57	MOV D,A	99	SBB C	DB	IN a8
16	MVI D,d8	58	MOV E,B	9A	SBB D	DC	CC a16
17	RAL	59	MOV E,C	9B	SBB E	DD	----
18	----	5A	MOV E,D	9C	SBB H	DE	SBI d8
19	DAD D	5B	MOV E,E	9D	SBB L	DF	RST3
1A	LDAX D	5C	MOV E,H	9E	SBB M	E0	RPO
1B	DCX D	5D	MOV E,L	9F	SBB A	E1	POP H
1C	INR E	5E	MOV E,M	A0	ANA B	E2	JPO a16
1D	DCR E	5F	MOV E,A	A1	ANA C	E3	XTHL
1E	MVI E,d8	60	MOV H,B	A2	ANA D	E4	CPO a16
1F	RAR	61	MOV H,C	A3	ANA E	E5	PUSH H
20	RIM	62	MOV H,D	A4	ANA H	E6	ANI d8
21	LXI H,d16	63	MOV H,E	A5	ANA L	E7	RST4
22	SHLD a16	64	MOV H,H	A6	ANA M	E8	RPE
23	INX H	65	MOV H,L	A7	ANA A	E9	PCHL
24	INR H	66	MOV H,M	A8	XRA B	EA	JPE a16
25	DCR H	67	MOV H,A	A9	XRA C	EB	XCHG
26	MVI H,d8	68	MOV L,B	AA	XRA D	EC	CPE a16
27	DAA	69	MOV L,C	AB	XRA E	ED	----
28	----	6A	MOV L,D	AC	XRA H	EE	XRI d8
29	DAD H	6B	MOV L,E	AD	XRA L	EF	RST5
2A	LHLD a16	6C	MOV L,H	AE	XRA M	F0	RP
2B	DCX H	6D	MOV L,L	AF	XRA A	F1	POP PSW
2C	INR L	6E	MOV L,M	B0	ORA B	F2	JP a16
2D	DCR L	6F	MOV L,A	B1	ORA C	F3	DI <sup>Disable</sup> <sub>Interrupt</sub> a16
2E	MVI L,d8	70	MOV M,B	B2	ORA D	F4	CP a16
2F	CMA	71	MOV M,C	B3	ORA E	F5	PUSH PSW
30	SIM	72	MOV M,D	B4	ORA H	F6	ORI d8
31	LXI SP,d16	73	MOV M,E	B5	ORA L	F7	RST6
32	STA a16	74	MOV M,H	B6	ORA M	F8	RM
33	INX SP	75	MOV M,L	B7	ORA A	F9	SPHL
34	INR M	76	HLT	B8	CMP B	FA	JM a16
35	DCR M	77	MOV M,A	B9	CMP C	FB	EI <sup>enable</sup> <sub>Interrupt</sub> a16
36	MVI M,d8	78	MOV A,B	BA	CMP D	FC	CM a16
37	STC	79	MOV A,C	BB	CMP E	FD	----
38	---	7A	MOV A,D	BC	CMP H	FE	CPI d8
39	DAD SP	7B	MOV A,E	BD	CMP L	FF	RST7
3A	LDA a16	7C	MOV A,H	BE	CMP M		
3B	DCX SP	7D	MOV A,L	BF	CMP A		
3C	INR A	7E	MOV A,M	C0	RNZ		
3D	DCR A	7F	MOV A,A	C1	POP B		
3E	MVI A,d8	80	ADD B	C2	JNZ a16		
3F	CMC	81	ADD C	C3	JMP a16		
40	MOV B,B	82	ADD D	C4	CNZ a16		
41	MOV B,C	83	ADD E	C5	PUSH B		

d8/d16 Eine Konstante oder ein log./arithm. Ausdruck, der eine 8-/16-Bit Datengröße darstellt.  
a8/a16 Eine 8-/16-Bit Adresse



# Einführung in die Programmierung

## Befehlsliste Hexadezimal

Befehlsliste geordnet in Befehlsgruppen  
Kurzfassung

Transfer 8 Bit		INPUT/OUTPUT	
MOVE	DB IN   Nr	DB OUT	Nr
06 MVI B.	50 MOV D.B	60 MOV H.B	70 MOV M.B
0E MVI C.	51 MOV D.C	61 MOV H.C	71 MOV M.C
16 MVI D.	52 MOV D.D	62 MOV H.D	72 MOV M.D
1E MVI E.	53 MOV D.E	63 MOV H.E	73 MOV M.E
26 MVI H.	54 MOV D.H	64 MOV H.H	74 MOV M.H
2E MVI L.	55 MOV D.L	65 MOV H.L	75 MOV M.L
36 MVI M.	56 MOV D.M	66 MOV H.M	
3E MVI A.	57 MOV D.A	67 MOV H.A	77 MOV M.A
LOAD STORE	58 MOV E.B	68 MOV L.B	78 MOV A.B
3A LDA   Adr	59 MOV E.C	69 MOV L.C	79 MOV A.C
32 STA   Adr	5A MOV E.D	6A MOV L.D	7A MOV A.D
0A LDAX B	5B MOV E.E	6B MOV L.E	7B MOV A.E
1A LDAX D	5C MOV E.H	6C MOV L.H	7C MOV A.H
02 STAX B	5D MOV E.L	6D MOV L.L	7D MOV A.L
12 STAX D	5E MOV E.M	6E MOV L.M	7E MOV A.M
	5F MOV E.A	6F MOV L.A	7F MOV A.A
Transfer 16 Bit		LOAD/STORE	
LOAD	LOAD/STORE	STACK	
01 LXI B.	2A LHLD   Adr	C3 PUSH B	C1 POP B
11 LXI D.	22 SHLD   Adr	D3 PUSH D	D1 POP D
21 LXI H.		E3 PUSH H	E1 POP H
31 LXI SP.	EB XCHG	F3 PUSH PSW	F1 POP PSW
			EJ XTHL
			F9 SPILL
Arithmetische Operationen 8 Bit		SUBTRACTION*	
ADDITION*	IN- CREMENT**	DE- CREMENT**	DE- CREMENT**
80 ADD B	88 ADC B	90 SUB B	98 SBB B
81 ADD C	89 ADC C	91 SUB C	99 SBB C
82 ADD D	8A ADC D	92 SUB D	9A SBB D
83 ADD E	8B ADC E	93 SUB E	9B SBB E
84 ADD H	8C ADC H	94 SUB H	9C SBB H
85 ADD L	8D ADC L	95 SUB L	9D SBB L
86 ADD M	8E ADC M	96 SUB M	9E SBB M
87 ADD A	8F ADC A	97 SUB A	9F SBB A
C6 ADI D8	CE ACI D8	D6 SUI D8	DESBI D8
			ADJUST*
			7F DAA
Arithmetische Operationen 16 Bit		INCREMENT	
ADDITION*	INCREMENT	DECREMENT	DECREMENT
09 DAD B	03 INX B	0B DCX B	
19 DAD D	13 INX D	1B DCX D	
29 DAD H	23 INX H	2B DCX H	
39 DAD SP	33 INX SP	3B DCX SP	

Logische Operationen 8 Bit		COMPARE*		COMPLEMENT	
AND*	XOR*	OR*			
A0 ANA B	A8 XRA B	B0 ORA B	B8 CMP B		2F CMA
A1 ANA C	A9 XRA C	B1 ORA C	B9 CMP C		
A2 ANA D	AA XRA D	B2 ORA D	BA CMP D		
A3 ANA E	AB XRA E	B3 ORA E	BB CNPE		
A4 ANA H	AC XRA H	B4 ORA H	BC CNPH		
A5 ANA L	AD XRA L	B5 ORA L	BD CNPL		
A6 ANA M	AE XRA M	B6 ORA M	BE CNPM		
A7 ANA A	AF XRA A	B7 ORA A	BF CNPA		
E6 ANI D8	EE XRI D8	F6 ORI D8	FE CPI D8		
Sprung		RETURN		RESTART	
JUMP	CALL				
C3 JMP	CD CALL	C9 RET		C7 RST 0	
C2 JNZ	C4 CNZ	C0 RNZ		CF RST 1	
CA JZ	CC CZ	C8 RZ		D7 RST 2	
D2 JNC	D4 CNC	D0 RNC		DF RST 3	
DA JC	DC CC	D8 RC		E7 RST 4	
E2 JPO	E4 CPO	E0 RPO		EF RST 5	
EA JPE	EC CPE	E8 RPE		F7 RST 6	
F2 JP	F4 CP	F0 RP		FF RST 7	
FA JM	FC CM	F8 RM			
E9 PCHL					
Registeranwahlungs- und sonstige Befehle		CARRY*		zusätzliche Befehle des MIP 8085	
ROTATE*	CONTROL				
07 RLC	00 NOP	37 STC		20 RIM	
0F RRC	76 HLT	3F CMC		30 SIM	
17 RAL	F3 DI				
1F RAR	FB EI				

Erläuterungen zur Tabelle

- D 16 = 16-Bit-Konstante
- D 8 = 8-Bit-Konstante
- Adr = Speicherplatzadresse
- Nr = E/A-Kanalnummer
- \* = beeinflusst alle Zustandsbits
- \*\* = beeinflusst alle Zustandsbits außer Carry
- + = beeinflusst nur Carry

# Befehlsliste des MP 8080/8085 geordnet in Befehlsgruppen

(nach Siemens)

Mnemonic	Binär-Code	Beeinflusste Zustands-Bits	Bytes	Taktzyklen 8080 / 8085	Englische Befehlsbeschreibung	Funktion des Befehls
----------	------------	----------------------------	-------	------------------------	-------------------------------	----------------------

## TRANSFERBEFEHLE

### a) Register → Register

MOV	r <sub>1</sub> , r <sub>2</sub>	0 1 c c c s s s	- - - - -	1	5	4	Move register to register	r <sub>1</sub> , r <sub>2</sub> = A, B, C, D, E, H oder L. Lade Register r <sub>1</sub> mit dem Inhalt von Register r <sub>2</sub> .
XCHG		1 1 1 0 1 0 1 1	- - - - -	1	4	4	Exchange D & E, H & L	Vertausche Inhalte der Registerpaare (D, E) und (H, L)
XTHL		1 1 1 c 0 0 1 1	- - - - -	1	18	16	Exchange top of stack H & L	Vertausche Inhalt des Registerpaars (H, L) und den Inhalt des Wortes, das durch den Stackpointer adressiert ist
SPHL		1 1 1 1 1 0 0 1	- - - - -	1	5	6	H & L to stackpointer	Lade Stackpointer mit dem Inhalt des Registerpaars (H, L)

### b) Speicher, Peripherie → Register

MOV	r <sub>1</sub> , M	0 1 d c d 1 1 0	- - - - -	1	7	7	Move memory to register	r <sub>1</sub> = A, B, C, D, E, H oder L. Lade Register r <sub>1</sub> mit dem Inhalt des Speicherbytes, das durch den Inhalt des Registerpaars (H, L) adressiert ist
LDA	adr	0 0 1 1 1 0 1 0	- - - - -	3	13	13	Load accu direct	Akkumulator laden mit dem Inhalt der Adresse adr
LDAX	rp	0 0 r r 1 0 1 0	- - - - -	1	7	7	Load accu indirect	rp = B, D: Akkumulator laden mit dem Inhalt des Speicherplatzes, der durch den Inhalt des Registerpaars rp adressiert ist
LHLD	adr	0 0 1 0 1 0 1 0	- - - - -	3	16	16	Load H & L direct	Lade Registerpaar (H, L) mit dem Inhalt der Adressen adr und (adr + 1)
POP	rp PSW	1 1 r r 0 0 0 1 1 1 1 1 0 0 0 1	Z. S. P, CY, AC	1	10	10	Pop register pair off stack	rp = B, D, H, PSW: Registerpaar rp wird mit dem Wort geladen, das durch den Stackpointer adressiert ist
IN	nr	1 1 0 1 1 0 1 1	- - - - -	2	10	10	Input	Akkumulator wird mit dem Inhalt des Eingabekanals (Nummer nr ≤ 255) geladen

### c) Konstante → Registerpaar

LXI	rp, adr	0 0 r r 0 0 0 1	- - - - -	3	10	10	Load register pair immediate	rp = B, D, H, SP: Lade Registerpaar rp mit Wert adr
-----	---------	-----------------	-----------	---	----	----	------------------------------	---

### d) Register → Speicher, Peripherie

MOV	M, r <sub>1</sub>	0 1 1 1 0 s s s	- - - - -	1	7	7	Move register to memory	r <sub>1</sub> = A, B, C, D, E, H oder L. Inhalt von Register r <sub>1</sub> auf den Speicherplatz abspeichern, der durch den Inhalt des Registerpaars (H, L) adressiert ist
STA	adr	0 0 1 1 0 0 1 0	- - - - -	3	13	13	Store accu direct	Akkumulator-Inhalt unter Adresse adr abspeichern
STAX	rp	0 0 r r 0 0 1 0	- - - - -	1	7	7	Store accu indirect	rp = B, D: Akkumulator in dem Byte abspeichern, das durch den Inhalt des Registerpaars rp adressiert ist
SHLD	adr	0 0 1 0 0 0 1 0	- - - - -	3	16	16	Store H & L direct	Registerpaar (H, L) unter Adresse adr u (adr + 1) abspeichern
PUSH	rp	1 1 r r 0 1 0 1	- - - - -	1	11	12	Push register pair rp on stack	rp = B, D, H, PSW: Inhalt des Registerpaars rp wird in das Wort übertragen, das durch den Stackpointer adressiert ist
OUT	nr	1 1 0 1 0 0 1 1	- - - - -	2	10	10	Output	Akkumulator-Inhalt wird auf Ausgabekanal (Nummer nr ≤ 255) ausgegeben

### e) Konstante → Register-Speicher

MVI	M, konst	0 0 1 1 0 1 1 0	- - - - -	2	10	10	Move to memory immediate	Lade den Speicherplatz, der durch den Inhalt des Registerpaars (H, L) adressiert ist, mit Konstante (konst ≤ 255)
MVI	r <sub>1</sub> , konst	0 0 d d d 1 1 0	- - - - -	2	7	7	Move immediate Register	r <sub>1</sub> = A, B, C, D, E, H oder L. Lade Register r <sub>1</sub> mit Konstante (konst ≤ 255)



BEFEHLSLISTE 8080/8085

TRANSFER-BEFEHLE

Mnemonic	Binär-Code	Beeinflusste Zustands-Bits	Bytes	Taktzyklen 8080 / 8085	Englische Befehlsbeschreibung	Funktion des Befehls
<b>ARITHMETISCHE OPERATIONEN</b>						
INR $r_i$	00ddd100	Z, S, P, - AC	1	5 4	Increment register	$r_i = A, B, C, D, E, H$ oder L: Zum Inhalt des Registers $r_i$ wird 1 addiert
INR M	00110100	Z, S, P, - AC	1	10 10	Increment memory	Zum Inhalt des durch Registerpaar (H, L) adressierten Bytes wird 1 addiert
DCR $r_i$	00ddd101	Z, S, P, - AC	1	5 4	Decrement register	$r_i = A, B, C, D, E, H$ oder L: Vom Inhalt des Registers $r_i$ wird 1 subtrahiert
DCR M	00110101	Z, S, P, - AC	1	10 10	Decrement memory	Vom Inhalt des durch Registerpaar (H, L) adressierten Bytes wird 1 subtrahiert
INX rp	00rr0011	- - - - -	1	5 6	Increment register pair	rp = B, D, H, SP: Der Inhalt des Registerpaares rp wird um 1 erhöht
DCX rp	00rr1011	- - - - -	1	5 6	Decrement register pair	rp = B, D, H, SP: Der Inhalt des Registerpaares rp wird um 1 erniedrigt
ADD $r_i$	10000sss	Z, S, P, CY, AC	1	4 4	Add register to accu	$r_i = A, B, C, D, E, H$ oder L: Inhalt von Register $r_i$ wird zum Inhalt des Akkumulators addiert
ADD M	10000110	Z, S, P, CY, AC	1	7 7	Add memory to accu	Inhalt des Speicherbytes, das durch den Inhalt des Registerpaares (H, L) adressiert ist, wird zum Inhalt des Akkumulators addiert
ADC $r_i$	10001sss	Z, S, P, CY, AC	1	4 4	Add register to accu with carry	$r_i = A, B, C, D, E, H$ oder L: Inhalt von Register $r_i$ und Inhalt des Carry-Bits werden zum Inhalt des Akkumulators addiert
ADC M	10001110	Z, S, P, CY, AC	1	7 7	Add memory to accu with carry	Inhalt des Speicherbytes, das durch den Inhalt des Registerpaares (H, L) adressiert ist und der Inhalt des Carry-Bits werden zum Inhalt des Akkumulators addiert
DAD rp	00rr1001	- - - CY -	1	10 10	Add register pair to H and L	rp = B, D, H, SP: Inhalt des Registerpaares rp und der Inhalt des Registerpaares (H, L) werden addiert. Ergebnis in (H, L)
SUB $r_i$	10010sss	Z, S, P, CY, AC	1	4 4	Subtract register from accu	$r_i = A, B, C, D, E, H$ oder L: Inhalt des Registers $r_i$ wird vom Akkumulator-Inhalt subtrahiert
SUB M	10010110	Z, S, P, CY, AC	1	7 7	Subtract memory from accu	Inhalt des Speicherbytes, das durch den Inhalt des Registerpaares (H, L) adressiert ist, wird vom Akkumulator subtrahiert
SBB $r_i$	10011sss	Z, S, P, CY, AC	1	4 4	Subtract register from accu with borrow	$r_i = A, B, C, D, E, H$ oder L: Inhalt von Register $r_i$ und Inhalt des Carry-Bits werden vom Akkumulator-Inhalt subtrahiert
SBB M	10011110	Z, S, P, CY, AC	1	7 7	Subtract memory from accu with borrow	Inhalt des Speicherbytes, das durch das Registerpaar (H, L) adressiert ist und Inhalt des Carry-Bits werden vom Akkumulator subtrahiert
ADI konst	11000110	Z, S, P, CY, AC	2	7 7	Add immediate to accu	Konstante (konst $\leq 255$ ) wird zum Inhalt des Akkumulators addiert
ACI konst	11001110	Z, S, P, CY, AC	2	7 7	Add immediate to accu with carry	Zum Akkumulator-Inhalt werden konst $\leq 255$ und Carry-Bit addiert
SUI konst	11010110	Z, S, P, CY, AC	2	7 7	Subtract immediate from accu	konst $\leq 255$ wird vom Akkumulator-Inhalt subtrahiert
SBI konst	11011110	Z, S, P, CY, AC	2	7 7	Subtract immediate from accu with borrow	konst $\leq 255$ und das Carry-Bit werden vom Akkumulator-Inhalt subtrahiert
DAA	00100111	Z, S, P, CY, AC	1	4 4	Decimal adjust accu	Akkumulator-Inhalt wird in eine 2stellige BCD-Zahl umgewandelt

Bei AC = 1 wird zu D<sub>0</sub> - D<sub>3</sub> Binär 6 addiert.

bei C = 1 " " D<sub>4</sub> - D<sub>7</sub> " 6 "

AC: Auxiliary carry (Half carry) (H)



BEFEHLSLISTE 8080/8085

ARITHMETISCHE BEFEHLE

Mnemonic	Binary-Code	Beeinflusste Zustands-Bits	Bytes	Taktzyklen	Englische Befehlsbeschreibung	Funktion des Befehls
----------	-------------	----------------------------	-------	------------	-------------------------------	----------------------

## LOGISCHE OPERATIONEN

CMA	00101111	- - - - -	1	4	4	Complement accu	Akkumulator-Inhalt wird negiert
ANA $r_1$ für 8085	10100sss	Z. S. P. 0, AC Z. S. P. 0, 1	1	4	4	And register with accu	$r_1 = A, B, C, D, E, H$ oder L: Akkumulator-Inhalt und der Inhalt des Registers $r_1$ werden UND-verknüpft
ANA M für 8085	10100110	Z. S. P. 0, AC Z. S. P. 0, 1	1	7	7	And memory with accu	Der Inhalt des durch Registerpaar (H, L) adressierten Bytes wird mit dem Akkumulator-Inhalt UND-verknüpft
ANI konst für 8085	11100110	Z. S. P. 0, AC Z. S. P. 0, 1	2	7	7	And immediate with accu	Akkumulator-Inhalt wird mit der konst $\leq 255$ UND-verknüpft
ORA $r_1$	10110sss	Z. S. P. 0, 0	1	4	4	Or register with accu	$r_1 = A, B, C, D, E, H$ oder L: Akkumulator-Inhalt wird mit dem Inhalt des Registers $r_1$ ODER-verknüpft
ORA M	10110110	Z. S. P. 0, 0	1	7	7	Or memory with accu	Inhalt des über Registerpaar (H, L) adressierten Bytes wird mit dem Akkumulator-Inhalt ODER-verknüpft
ORI konst	11110110	Z. S. P. 0, 0	2	7	7	Or immediate with accu	Akkumulator-Inhalt wird mit konst $\leq 255$ ODER-verknüpft
XRA $r_1$	10101sss	Z. S. P. 0, 0	1	4	4	Exclusive Or register with accu	$r_1 = A, B, C, D, E, H$ oder L: Akkumulator-Inhalt wird mit dem Inhalt des Registers $r_1$ Exklusiv-ODER-verknüpft
XRA M	10101110	Z. S. P. 0, 0	1	7	7	Exclusive Or memory with accu	Das über Registerpaar (H, L) adressierte Byte wird Exklusiv-ODER mit dem Akkumulator-Inhalt verknüpft
XRI konst	11101110	Z. S. P. 0, 0	2	7	7	Exclusive Or immediate with accu	Der Akkumulator-Inhalt wird mit dem Wert konst $\leq 255$ Exklusiv-ODER verknüpft
CMP $r_1$	10111sss	Z. S. P. CY, AC	1	4	4	Compare register with accu	$r_1 = A, B, C, D, E, H$ oder L: Akkumulator-Inhalt wird mit dem Inhalt des Registers $r_1$ verglichen
CMP M	10111110	Z. S. P. CY, AC	1	7	7	Compare memory with accu	Akkumulator-Inhalt wird mit dem Inhalt des durch Registerpaar (H, L) adressierten Bytes verglichen
CPI konst	11111110	Z. S. P. CY, AC	2	7	7	Compare immediate with accu	Akkumulator-Inhalt wird mit konst $\leq 255$ verglichen

## REGISTERANWEISUNGEN

### a) Akkumulator rotieren

RLC	00000111	- - - CY -	1	4	4	Rotate accu left	Akkumulatorinhalt wird zyklisch um 1 Bit nach links verschoben. Bit $2^7$ wird in das Carry-Bit geschrieben. Bit $2^0 =$ Bit $2^7$
RRC	00001111	- - - Cy -	1	4	4	Rotate accu right	Akkumulator-Inhalt wird zyklisch um 1 Bit nach rechts verschoben. Bit $2^0$ wird in das Carry-Bit geschrieben. Bit $2^7 =$ Bit $2^0$
RAL	00010111	- - - CY -	1	4	4	Rotate accu left through carry	Akkumulatorinhalt wird um 1 Bit nach links geschoben. Bit $2^7$ wird in das Carry-Bit und das Carry-Bit in das Bit $2^0$ geschrieben
RAR	00011111	- - - CY -	1	4	4	Rotate accu right through carry	Akkumulator-Inhalt wird um 1 Bit nach rechts geschoben. Bit $2^0$ wird in das Carry-Bit und das Carry-Bit in das Bit $2^7$ geschrieben

### b) Übertragsbit-Anweisungen

CMC	00111111	- - - CY -	1	4	4	Complement carry	Carry-Bit wird negiert
STC	00110111	- - - 1 -	1	4	4	Set carry	Carry-Bit wird gesetzt

CMP / CPI : AKKU < Konstante : Carry = 1 Zero = 0  
 AKKU = " : Carry = 0 Zero = 1  
 AKKU > " : Carry = 0 Zero = 0



BEFEHLSLISTE 8080/8085

LOGISCHE BEFEHLE



Mnemonic	Binär-Code	Beeinflusste Zustands-Bits	Bytes	Taktrzyklen 8080 / 8085	Englische Befehlsbeschreibung	Funktion des Befehls
----------	------------	----------------------------	-------	-------------------------	-------------------------------	----------------------

### SPRUNGBEFEHLE

#### a) Unbedingte Sprünge

PCHL	11101001	- - - - -	1	5	6	H & L to program counter	Programm wird an der Adresse fortgesetzt, die im Registerpaar (H, L) steht
JMP adr	11000011	- - - - -	3	10	10	Jump unconditional	Programm wird an der Adresse adr fortgesetzt

#### b) Bedingte Sprünge

JC adr	11011010	- - - - -	3	10	7/10	Jump on carry	Bei Carry-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
JNC adr	11010010	- - - - -	3	10	7/10	Jump on no carry	Bei Carry-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt
JZ adr	11001010	- - - - -	3	10	7/10	Jump on zero	Bei Zero-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
JNZ adr	11000010	- - - - -	3	10	7/10	Jump on no zero	Bei Zero-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt
JM adr	11111010	- - - - -	3	10	7/10	Jump on minus	Bei Sign-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
JP adr	11110010	- - - - -	3	10	7/10	Jump on positiv	Bei Sign-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt
JPE adr	11101010	- - - - -	3	10	7/10	Jump on parity even	Bei Parity-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
JPO adr	11100010	- - - - -	3	10	7/10	Jump on parity odd	Bei Parity-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt



Mnemonic	Binary-Code	Beeinflusste Zustands-Bits	Bytes	Taktzyklen 8080 / 8085	Englische Befehlsbeschreibung	Funktion des Befehls
----------	-------------	----------------------------	-------	------------------------	-------------------------------	----------------------

## UNTERPROGRAMMBEHANDLUNG

### a) Programmaufrufe

Bei allen Aufrufbefehlen wird die Rückkehradresse in dem Wort, das durch den Stackpointer adressiert ist, abgelegt

CALL	adr	11001101	- - - - -	3	17 18	Call unconditional	Programm wird bei der Adresse fortgesetzt
CC	adr	11011100	- - - - -	3	11/17 9/18	Call on carry	Bei Carry-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
CNC	adr	11010100	- - - - -	3	11/17 9/18	Call on no carry	Bei Carry-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt
CZ	adr	11001100	- - - - -	3	11/17 9/18	Call on zero	Bei Zero-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
CNZ	adr	11000100	- - - - -	3	11/17 9/18	Call on no zero	Bei Zero-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt
CM	adr	11111100	- - - - -	3	11/17 9/18	Call on minus	Bei Sign-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
CP	adr	11110100	- - - - -	3	11/17 9/18	Call on positiv	Bei Sign-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt
CPE	adr	11101100	- - - - -	3	11/17 9/18	Call on parity even	Bei Parity-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt
CPO	adr	11100100	- - - - -	3	11/17 9/18	Call on parity odd	Bei Parity-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt
RST	konst	11nnn111	- - - - -	1	11 12	Restart	Programm wird auf der Adresse $8 \times konst$ fortgesetzt ( $0 \leq konst \leq 7$ )

### b) Rücksprungbefehle

RET		11001001	- - - - -	1	10 10	Return	Programm wird an der Adresse fortgesetzt, die in dem Wort steht, das über dem Stackpointer adressiert ist
RC		11011000	- - - - -	1	5/11 6/12	Return on carry	Bei Carry-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht
RNC		11010000	- - - - -	1	5/11 6/12	Return on no carry	Bei Carry-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht
RZ		11001000	- - - - -	1	5/11 6/12	Return on zero	Bei Zero-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht
RNZ		11000000	- - - - -	1	5/11 6/12	Return on no zero	Bei Zero-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht
RM		11111000	- - - - -	1	5/11 6/12	Return on minus	Bei Sign-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht
RP		11110000	- - - - -	1	5/11 6/12	Return on positiv	Bei Sign-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht
RPE		11101000	- - - - -	1	5/11 6/12	Return on parity even	Bei Parity-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht
RPO		11100000	- - - - -	1	5/11 6/12	Return on parity odd	Bei Parity-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stackpointer adressierten Wort steht



Alte mnemonic	Binär-Code	Beeinflusste Zustands-Bits	Bytes	Taktzyklen 8080 / 8085	Englische Befehlsbeschreibung	Funktion des Befehls
---------------	------------	-------------------------------	-------	---------------------------	----------------------------------	----------------------

### PROGRAMMUNTERBRECHUNG

EI	11111011	- - - - -	1	4	4	Enable interrupts	INTE-Flipflop wird gesetzt, der Mikroprozessor kann eine Unterbrechungsanforderung annehmen
DI	11111001	- - - - -	1	4	4	Disable interrupts	INTE-Flipflop wird rückgesetzt; der Mikroprozessor ignoriert Unterbrechungsanforderungen

### SONSTIGE BEFEHLE

HLT	01110110	- - - - -	1	7	5	Halt	Programm hält an, bis eine Unterbrechungsanforderung eintritt
NOP	00000000	- - - - -	1	4	4	No operation	Leerbefehl

### SAB 8085 BEFEHLE

RIM	00100000	- - - - -	1	-	4	Read interrupt Mask	Lesen Unterbrechungsmaske und seriellen Eingang in Akkumulator
SIM	00110000	- - - - -	1	-	4	Set interrupt Mask	Setzen Unterbrechungsmaske und seriellen Ausgang

#### Legende zu „Binär-Code“

ddd = Zielregister	ddd/sss	rp
rp = Registerpaar	000 ≙ Reg. B	00 ≙ B/C
sss = Quellregister	001 ≙ Reg. C	01 ≙ D/E
	010 ≙ Reg. D	10 ≙ H/L
	011 ≙ Reg. E	11 ≙ Stack Ptr. (SP)
	100 ≙ Reg. H	
	101 ≙ Reg. L	
	110 ≙ Speicher (M)	
	111 ≙ Akku	

Verschiedenes: |- ACCU löschen mit XRA A (gleichzeitig wird das Carry-Bit gelöscht )  
|- FLAGS setzen mit ORA A



Bedeutung der Spalten Befehlsgruppe	Mnemonic	Maschinen Code	Taktzyklen	Bytes	Masch Zyklus	Funktion des Befehls	Veränderte Flags				
							N	Z	P	C	H
Daten-Transport	MOV r1, r2		4	1	1	(r1) ← (r2)	X	X	X	X	X
	MOV M, r		7	1	2	(M) ← (r)	X	X	X	X	X
	MOV r, M		7	1	2	(r) ← (M)	X	X	X	X	X
	MVI r, n		7	2	2	(r) ← n	X	X	X	X	X
	MVI M, n	3 6	10	2	3	(M) ← n	X	X	X	X	X
	LXI B, m	0 1	10	3	3	(C) ← <B2> (B) ← <B3>    m = <B3><B2>	X	X	X	X	X
	LXI D, m	1 1	10	3	3	(E) ← <B2> (D) ← <B3>    m = <B3><B2>	X	X	X	X	X
	LXI H, m	2 1	10	3	3	(L) ← <B2> (H) ← <B3>    m = <B3><B2>	X	X	X	X	X
	LXI SP, m	3 1	10	3	3	(SP) ← m	X	X	X	X	X
	SPHL	F 9	6	1	1	(SP) ← (H) (L)	X	X	X	X	X
	STAX B	0 2	7	1	2	((B) (C)) ← (A)	X	X	X	X	X
	STAX D	1 2	7	1	2	((D) (E)) ← (A)	X	X	X	X	X
	LDAX B	0 A	7	1	2	(A) ← ((B) (C))	X	X	X	X	X
	LDAX D	1 A	7	1	2	(A) ← ((D) (E))	X	X	X	X	X
	STA m	3 2	13	3	4	(m) ← (A)	X	X	X	X	X
LDA m	3 A	13	3	4	(A) ← (m)	X	X	X	X	X	
SHLD m	2 2	16	3	5	(m) ← (L) (m+1) ← (H)	X	X	X	X	X	
SHLD m	2 A	16	3	5	(L) ← (m) (H) ← (m+1)	X	X	X	X	X	
XCHG	E B	4	1	1	(H) (L) ↔ (D) (E)	X	X	X	X	X	
XTHL	E 3	16	1	5	(H) (L) ↔ ((SP)-1) ((SP))	X	X	X	X	X	
Arithmetik, Logik, Vergleich	ADD r		4	1	1	(A) ← (A)+(r)	0	0	0	0	0
	ADD M	8 6	7	1	2	(A) ← (A)+(M)	0	0	0	0	0
	ADI n	C 6	7	2	2	(A) ← (A)+n	0	0	0	0	0
	ADC r		4	1	1	(A) ← (A)+(r)+(C)	0	0	0	0	0
	ADC M	B E	7	1	2	(A) ← (A)+(M)+(C)    C = Carry	0	0	0	0	0
	ACI n	C E	7	2	2	(A) ← (A)+n+(C)	0	0	0	0	0
	DAD B	0 9	10	1	3	(H) (L) ← ((H) (L))+(B) (C)	X	X	X	0	X
	DAD D	1 9	10	1	3	(H) (L) ← ((H) (L))+(D) (E)	X	X	X	0	X
	DAD H	2 9	10	1	3	(H) (L) ← ((H) (L))+(H) (L)	X	X	X	0	X
	DAD SP	3 9	10	1	3	(H) (L) ← ((H) (L))+(SP)	X	X	X	0	X
	SUB r		4	1	1	(A) ← (A)-(r)	0	0	0	0	0
	SUB M	9 6	7	1	2	(A) ← (A)-(M)	0	0	0	0	0
	SUI n	D 6	7	2	2	(A) ← (A)-n	0	0	0	0	0
	SBB r		4	1	1	(A) ← (A)-(r)-(C)	0	0	0	0	0
	SBB M	9 E	7	1	2	(A) ← (A)-(M)-(C)    C = Carry	0	0	0	0	0
	SBI n	D E	7	2	2	(A) ← (A)-n-(C)	0	0	0	0	0
	ANA r		4	1	1	(A) ← (A)∧(r)	0	0	0	0	1
	ANA M	A 6	7	1	2	(A) ← (A)∧(M)	0	0	0	0	1
ANI n	E 6	7	2	2	(A) ← (A)∧n	0	0	0	0	1	
XRA r		4	1	1	(A) ← (A)⊕(r)	0	0	0	0	0	
XRA M	A E	7	1	2	(A) ← (A)⊕(M)	0	0	0	0	0	
XRI n	E E	7	2	2	(A) ← (A)⊕n	0	0	0	0	0	
ORA r		4	1	1	(A) ← (A)∨(r)	0	0	0	0	0	
ORA M	B 6	7	1	2	(A) ← (A)∨(M)	0	0	0	0	0	
ORI n	F 6	7	2	2	(A) ← (A)∨n	0	0	0	0	0	
CMP r		4	1	1	(A) - (r)	0	0	0	0	0	
CMP M	B E	7	1	2	(A) - (M)	0	0	0	0	0	
CPI n	F E	7	2	2	(A) - n	0	0	0	0	0	
Register Inkrement Dekrement	INR r		4	1	1	(r) ← (r)+1	0	0	0	X	0
	INR M	3 4	10	1	3	(M) ← (M)+1	0	0	0	X	0
	DCR r		4	1	1	(r) ← (r)-1	0	0	0	X	0
	DCR M	3 5	10	1	3	(M) ← (M)-1	0	0	0	X	0
	INC B	0 3	6	1	1	(B) (C) ← ((B) (C))+1	X	X	X	X	X
	INC D	1 3	6	1	1	(D) (E) ← ((D) (E))+1	X	X	X	X	X
	INC H	2 3	6	1	1	(H) (L) ← ((H) (L))+1	X	X	X	X	X
	INC SP	3 3	6	1	1	(SP) ← (SP)+1	X	X	X	X	X
DCX B	0 B	6	1	1	(B) (C) ← ((B) (C))-1	X	X	X	X	X	
DCX D	1 B	6	1	1	(D) (E) ← ((D) (E))-1	X	X	X	X	X	
DCX H	2 B	6	1	1	(H) (L) ← ((H) (L))-1	X	X	X	X	X	
DCX SP	3 B	6	1	1	(SP) ← (SP)-1	X	X	X	X	X	
Akku rotieren	RLC	0 7	4	1	1	Links schieben C ← [A7 A6 ..... A1 A0]	X	X	X	0	X
	RRC	0 F	4	1	1	Rechts schieben C ← [A7 A6 ..... A1 A0]	X	X	X	0	X
	RAL	1 7	4	1	1	Links rotieren C ← [A7 A6 ..... A1 A0]	X	X	X	0	X
	RAR	1 F	4	1	1	Rechts rotieren C ← [A7 A6 ..... A1 A0]	X	X	X	0	X
Akku beeinfl.	CMA	2 F	4	1	1	(A) ← ~ (A)	X	X	X	0	0
	DAA	2 7	4	1	1	(A) in BCD wandeln	0	0	0	0	0
Carry setzen	STC	3 7	4	1	1	(C) ← 1	X	X	X	1	X
	CMC	3 F	4	1	1	(C) ← ~ (C)	X	X	X	0	X



Bedeutung der Spalten Befehls- gruppe	Mnemonic	Maschinen Code	Taktzyklen	Bytes	Masch. Zyklus	Funktion des Befehls	Veränderte Flags				
							N	Z	P	C	H
Sprünge	JMP m	C 3	10	3	3	(PC) ← m	X	X	X	X	X
	PCHL	E 9	6	1	1	(PC) ← (H) (L)	X	X	X	X	X
	JC m	D A	10/7	3	3/2	(C)=1	Wenn Bedingung erfüllt (PC) ← m	X	X	X	X
	JNC m	D 2	10/7	3	3/2	(C)=0		X	X	X	X
	JZ m	C A	10/7	3	3/2	(Z)=1		X	X	X	X
	JNZ m	C 2	10/7	3	3/2	(Z)=0		X	X	X	X
	JF m	F 2	10/7	3	3/2	(N)=0	Wenn Bedingung nicht erfüllt (PC) ← (PC) + 3	X	X	X	X
	JM m	F A	10/7	3	3/2	(N)=1		X	X	X	X
JPE m	E A	10/7	3	3/2	(P)=1	X		X	X	X	
JPO m	E 2	10/7	3	3/2	(P)=0	X		X	X	X	
Unter- programm- aufrufe	CALL n	C 0	18	3	5	((SP)-1) ((SP)-2) ← (PC)+3, (PC) ← m (SP) ← (SP)-2	X	X	X	X	
	RST n		12	1	3	((SP)-1) ((SP)-2) ← (PC)+L, (PC) ← n x 8 (SP) ← (SP)-2 wobei 0 ≤ n ≤ 7	X	X	X	X	
	CC m	D C	18/9	3	5/2	(C)=1	Wenn Bedingung erfüllt ((SP)-1) ((SP)-2) ← (PC)+3 (PC) ← m, (SP) ← (SP)-2	X	X	X	X
	CNC m	D 4	18/9	3	5/2	(C)=0		X	X	X	X
	CZ m	C C	18/9	3	5/2	(Z)=1		X	X	X	X
	CNZ m	C 4	18/9	3	5/2	(Z)=0		X	X	X	X
	CP m	F 4	18/9	3	5/2	(N)=0	Wenn Bedingung nicht erfüllt (PC) ← (PC)+3	X	X	X	X
	CM m	F C	18/9	3	5/2	(N)=1		X	X	X	X
CPE m	E C	18/9	3	5/2	(P)=1	X		X	X	X	
CPO m	E 4	18/9	3	5/2	(P)=0	X		X	X	X	
Unter- programm- Rück- sprünge	RET	C 9	10	1	3	(PC) ← ((SP)-1) ((SP)-2) ← (SP)-2	X	X	X	X	
	RC	D 8	12/6	1	3/1	(C)=1	Wenn Bedingung erfüllt (PC) ← ((SP)-1) ((SP)-2) ← (SP)+2	X	X	X	X
	RNC	D 0	12/6	1	3/1	(C)=0		X	X	X	X
	RZ	C 8	12/6	1	3/1	(Z)=1		X	X	X	X
	RNZ	C 0	12/6	1	3/1	(Z)=0		X	X	X	X
	RP	F 0	12/6	1	3/1	(N)=0	Wenn Bedingung nicht erfüllt (PC) ← (PC)+1	X	X	X	X
	RM	F 8	12/6	1	3/1	(N)=1		X	X	X	X
	RPE	E 8	12/6	1	3/1	(P)=1		X	X	X	X
RPO	E 0	12/6	1	3/1	(P)=0	X		X	X	X	
Ein- Ausgabe	IN n	D B	10	2	3	(A) ← (Eing. Puffer) ← (Eingang Daten von Gerät n)	X	X	X	X	
	OUT n	D 3	10	2	3	(Ausgabe Gerät n) ← (A)	X	X	X	X	
Unterbrech. Steuerung	EI	F B	4	1	1	(INTE) ← 1	X	X	X	X	
	DI	F 3	4	1	1	(INTE) ← 0	X	X	X	X	
Stapel- steuerung	PUSH PSW	F 5	12	1	3	((SP)-1) ← (A), ((SP)-2) ← (F), (SP) ← (SP)-2	X	X	X	X	
	PUSH B	C 5	12	1	3	((SP)-1) ← (B), ((SP)-2) ← (C), (SP) ← (SP)-2	X	X	X	X	
	PUSH D	D 5	12	1	3	((SP)-1) ← (D), ((SP)-2) ← (E), (SP) ← (SP)-2	X	X	X	X	
	PUSH H	E 5	12	1	3	((SP)-1) ← (H), ((SP)-2) ← (L), (SP) ← (SP)-2	X	X	X	X	
	POP PSW	F 1	10	1	3	(F) ← ((SP)-1), (A) ← ((SP)-2), (SP) ← (SP)+2	•	•	•	•	
	POP B	C 1	10	1	3	(C) ← ((SP)-1), (B) ← ((SP)-2), (SP) ← (SP)+2	X	X	X	X	
	POP D	D 1	10	1	3	(E) ← ((SP)-1), (D) ← ((SP)-2), (SP) ← (SP)+2	X	X	X	X	
	POP H	E 1	10	1	3	(L) ← ((SP)-1), (H) ← ((SP)-2), (SP) ← (SP)+2	X	X	X	X	
Sonstige	HLT	7 6	5	1	1	(PC) ← (PC)+1	X	X	X	X	
	NOP	0 0	4	1	1	(PC) ← (PC)+1	X	X	X	X	
8085 Befehle	RM	2 0	4	1	1	Liest Unterbrechungsmaske und seriellen Eingang in Akku	X	X	X	X	
	SM	3 0	4	1	1	Setzt Unterbrechungsmaske und seriellen Ausgang	X	X	X	X	

Symbol	Erklärung
r	Register (A, B, C, D, E, H, L, M)
m	Zwei-Byte Daten
n	Ein-Byte Daten
<B7>, <B3>	2. bzw. 3. Byte des Befehls
F	8-Bit-Daten-Wort aus M, Z, X, H, X, P, X, C
PC	Programm-Zähler
SP	Stapel-Zeiger
←	Datum in gezeichneter Richtung transportiert

Symbol	Erklärung
( )	Inhalt eines Registers oder Speichers
∨, ∇	Inklusiv ODER, Exklusiv ODER
∧	Log. UND
—	1er Komplement
X	Flaginhalt bleibt unverändert
•	Flaginhalt wird gesetzt oder rückgesetzt
M	Inhalt der durch M und L adressierten Speicherzeile



## Einführung in die Programmierung

### Befehlsliste 8085

## Das HELP-Kommando

Mit dem HELP-Kommando lassen sich die Namen aller zulässigen Kommandos des Betriebssystems MAT 85 in alphabetischer Reihenfolge ausdrucken.

Aufruf:    KMD>HELP CR  
                  (Kommandoausführung)  
                  KMD>\_

Nach dem Ausdruck aller Kommandonamen erfolgt ein Rücksprung in die Kommando-Routine ( KMD>\_ )  
Die obere Zeile "KMD>HELP" wird überschrieben und ist auf dem Monitor nicht mehr sichtbar.

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen-  
alle weiteren Ausdrücke werden vom Betriebsprogramm ausgeführt.

CR : CR-Taste (CR = carriage return, Wagenrücklauf)



Einführung in die Programmierung

Arbeitsblatt: Betriebsprogramm-Kommandos

## Das MEMORY-Kommando (memory, Speicher)

Mit dem MEMORY-Kommando ist es möglich, sich die Inhalte einzelner Speicherstellen anzuschauen (ROM und RAM) und bei Bedarf zu ändern (nur beim RAM).

Aufruf:            KMD > MEMORY CR  
                      START-ADR = XXXX E000 CR  
                      FORMAT    = H CR  
                      E000 XX 41 ┘  
                      E001 XX

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen—  
alle weiteren Ausdrücke werden vom Betriebsprogramm  
ausgeführt.

- CR : carriage-return, Wagenrücklauf
- XXXX : Vorschlagsadresse durch das Betriebsprogramm
- XX : Speicherinhalt (alt)
- 41 : neuer Speicherinhalt
- ┘ : Leertaste



Einführung in die Programmierung

Arbeitsblatt: Betriebsprogramm-Kommandos

## Das PRINT-Kommando

Mit dem PRINT-Kommando ist es möglich, die Inhalte eines Speicherbereichs im gewünschten Format auszudrucken. Die möglichen Formate entsprechen denen des M-Kommandos. Im Ausdruck werden je nach gewähltem Format bis zu acht Speicherinhalte in einer Zeile ausgedruckt. Jede Zeile beginnt mit der Adresse des ersten in der Zeile ausgedruckten Speicherinhaltes.

```
Aufruf:      KMD>PRINT CR
                START-ADR = XXXX E000 CR
                STOP  -ADR = XXXX E014 CR
                FORMAT   = H CR
E000  DB 01 2F D3 02 C3 00 E0
E008  3E 03 3D C2 00 E0 CF 79
E010  49 20 4F CF D4
KMD>_
```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen-  
alle weiteren Ausdrücke werden vom Betriebsprogramm  
ausgeführt.

CR : carriage-return, Wagenrücklauf  
XXXX : Vorschlagsadresse durch das Betriebsprogramm





## Das GO-Kommando

Mit dem GO-Kommando wird der Prozessor veranlaßt, Anwender-Programme von einer bestimmten Startadresse an zu bearbeiten. Bei Ausführung zyklischer Programme (Schleifen) kann nur durch Betätigen der RESET-Taste eine Rückkehr zum Betriebsprogramm erfolgen.

Aufruf: KMD>GO CR  
START-ADR = XXXX E000 CR oder SP  
\*\*\* RESET \*\*\* (nach Betätigen der Taste)  
KMD \_

Nicht zyklische Programme müssen mit einem Rücksprungbefehl (RST1, RESTART, CFhex) abgeschlossen sein. Wenn dieser Befehl ausgeführt wurde meldet sich das Betriebsprogramm mit dem Ausdruck \*\*\* USER \*\*\*

Aufruf: KMD>GO CR  
START-ADR = XXXX E000 CR oder SP  
(Kommandoausführung)  
\*\*\* USER \*\*\* (nach Ausführung des Rücksprung-  
befehls )  
KMD> \_

Wird das Programm kontinuierlich abgearbeitet, erscheint folgender Ausdruck auf dem Monitor:

KMD>GO CR  
START-ADR = XXXX E000 CR oder SP  
\_ (blinkender Cursor)



## Das DISASSEMBLER-Kommando

Dieses Kommando ermöglicht im Speicher stehende Maschinenprogramme in den Mnemo-Code zu übersetzen. Der Vorteil dieser Darstellung eines Maschinenprogramms ist die bessere Lesbarkeit.

```

                                     [CR]
Aufruf:   KMD> DISASSEMBLER
              START-ADR = XXXX E000 [CR]
              STOP -ADR = XXXX E006 [CR]
              E000 DB 01           IN 01
              E002 D3 02          OUT 02
              E004 C3 00 E0       JMP E000
                                   END
Speicher-  Maschinen-  Mnemo-Code
adresse   code
      KMD> _
```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen - alle weiteren Ausdrücke werden vom Betriebsprogramm ausgeführt.

CR : carriage-return, Wagenrücklauf  
XXXX : Vorschlagsadresse durch das Betriebsprogramm



Einführung in die Programmierung

Arbeitsblatt: Betriebsprogramm-Kommandos

## Das NEXT-INSTRUCTION-Kommando

Über dieses Kommando ist eine schrittweise Abarbeitung (Trace-Lauf, trace=Spur, verfolgen ...) eines Anwenderprogramms möglich. Nach jedem ausgeführten Befehl werden die momentanen Registerinhalte der CPU auf dem Bildschirm ausgedruckt. Den Ausdruck veranlaßt das Betriebsprogramm über entsprechende Befehle an die CPU. Dadurch erhöht sich die Programmausführungszeit erheblich.

### Aufruf:

```
      CR
      |
KMD> NEXT INSTRUCTION
START-ADR = XXXX F800 CR
STEPS     = X 5 CR
```

PC	LABEL:	OP	ADR.FELD	A	NZHPC	B	C	D	E	H	L	I	SP
FB00	START:	MVI	A,03	03	00000	00	00	00	00	00	00	80	FC32
FB02	SCHL1:	DCR	A	02	00100	00	00	00	00	00	00	80	FC32
FB03		JNZ	SCHL1	02	00100	00	00	00	00	00	00	80	FC32
FB02	SCHL1:	DCR	A	01	00100	00	00	00	00	00	00	80	FC32
FB03		JNZ	SCHL1	01	00100	00	00	00	00	00	00	80	FC32
FB02	SCHL1:	DCR	A										

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen-  
alle weiteren Ausdrücke werden vom Betriebsprogramm  
ausgeführt

CR : carriage-return, Wagenrücklauf

XXXX : Vorschlagsadresse durch das Betriebsprogramm

STEPS : Dezimale Eingabe der zu protokollierenden  
Befehle



## Das ASSEMBLER-Kommando

Der Assembler gestattet die Programmeingabe im Mnemo-Code. Jede im Mnemo-Code eingegebene Programmzeile wird nach Betätigung der Taste "CR" auf Richtigkeit überprüft und in den Maschinen-Code übersetzt. Werden bestimmte Regeln\* nicht eingehalten oder ist der eingegebene Mnemo-Code unbekannt, so markiert der Assembler die betreffende Stelle mit einem Fragezeichen.

\*Regeln zur richtigen Eingabe im Mnemo-Code sind auf einem gesonderten Arbeitsblatt.

```
Aufruf:  KMD>ASSEMBLER CR
          START-ADR=XXXX E000 CR
          E000
          -
          EXXX          END
*** RESTART ? (JA/NEIN) N
```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen-  
alle weiteren Ausdrücke werden vom Betriebsprogramm  
ausgeführt.

CR :carriage-return, Wagenrücklauf

XXXX :Vorschlagadresse durch das Betriebsprogramm

END :Beendigung der Eingabe



## REGISTER-KOMMANDO

Mit dem Register-Kommando können die Inhalte der CPU-Register angezeigt und verändert werden. Dazu werden zunächst die alten Register-Inhalte ausgedruckt und durch eine Kopfzeile kommentiert. Die Kopfzeile hat die Form:

```
PC LABEL: OP ADR.FELD A NZHPC B C D E H L I SP
```

Bedeutung der verwendeten Abkürzungen:

PC - PROGRAM COUNTER (PROGRAMMZAehler, 16 BIT)

LABEL:  
OP  
ADR.FELD } SIEHE ASSEMBLER-/DISASSEMBLER-KOMMANDOS

A - REGISTER A (ACCU, 8 BIT)

N - NEGATIV -FLAG (NEGATIV-BIT)

Z - ZERO -FLAG (NULL-BIT)

H - HALF CARRY-FLAG (ZWISCHENUEBERTRAGS-BIT)

P - PARITY -FLAG (PARITAETS-BIT)

C - CARRY -FLAG (UEBERTRAGS-BIT)

} FLAG-BITS DES  
PROZESSOR-  
STATUS-REGISTERS  
(STATUS-WORT BZW.  
BEDINGUNGSREGISTER)

B - REGISTER B (8 BIT)

C - REGISTER C (8 BIT)

D - REGISTER D (8 BIT)

E - REGISTER E (8 BIT)

H - REGISTER H (8 BIT)

L - REGISTER L (8 BIT)

I - INTERRUPT CONTROL REGISTER (INTERRUPT-MASKEN-REGISTER, 8 BIT)

SP - STACK POINTER (STAPEL-ZEIGER, 16 BIT)

Die Schreibmarke (CURSOR) steht nach dem Ausdruck der alten Register-Inhalte unterhalb des Programmzähler-Inhalts. Soll dieser verändert werden, so kann der neue Inhalt über die Tastatur eingegeben werden. Will man einen Register-Inhalt nicht verändern, so kann die Schreibmarke durch die Leertaste (SPACE) zum nächsten Register weiterbewegt werden. Das Kommando wird beendet, wenn die Taste CR oder, beim letzten Register, die Leertaste betätigt wird. Die neuen Register-Inhalte werden erst beim Start eines Programms mit dem GO-Kommando in die CPU-Register übernommen.



Einführung in die Programmierung

Arbeitsblatt: Betriebsprogramm-Kommandos

8

BEISPIEL:

① Aufruf des R-Kommandos und Vorbelegung der Register.

```

      ↙ CR
KMD > REGISTER
PC LABEL: OP ADR.FELD A NZHPC B C D E H L I SP
1E09 00 00000 00 00 00 00 00 00 00 FC32
F800 41 01001 55 21 FB 20

```

② Erneuter Aufruf des R-Kommandos und Registervorbelegung anschauen.

```

      ↙ CR
KMD > REGISTER
PC LABEL: OP ADR.FELD A NZHPC B C D E H L I SP
F800 41 01001 55 21 00 00 FB 20 00 FC32
CR
KMD >

```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen.  
Alle weiteren Ausdrücke werden vom Betriebsprogramm ausgeführt.

CR : CR-Taste (CR = carriage return, Wagenrücklauf)



## Das BREAKPOINT-Kommando (breakpoint = Haltepunkt)

Mit dem BREAKPOINT-Kommando wird das Einsetzen von Haltepunkten in Programme ermöglicht. Dadurch können bestimmte Programmteile in Echtzeit durchlaufen werden und die am Haltepunkt aktuellen Registerinhalte ausgedruckt werden. Von dem Haltepunkt aus kann das Programm erneut gestartet werden (GO) oder mit dem N-Kommando schrittweise beobachtet werden. Die Eingabe der Breakpoint-Adressen erfolgt innerhalb der Ausführung des G-Kommandos nach der Eingabe der Programmstartadresse.

Aufruf:           KMD>BREAKPOINT CR  
                   BREAK-ADR1 = XXXX  
                   BREAK-ADR2 = XXXX  
                   BREAK-ADR3 = XXXX           Änderungen nur unter dem  
                   BREAK-ADR4 = XXXX           Go-Kommando möglich

EIN/AUS       = A E CR oder SP

                  KMD>GO CR  
                   START-ADR = XXXX E000 CR  
                   BREAK-ADR1= XXXX E030 SP  
                   BREAK-ADR2= XXXX E065 CR  
                   BREAK-ADR3= XXXX           CR  
                   BREAK-ADR4= XXXX           CR

Wenn alle vier  
Break-Adr. gesetzt  
werden sollen, bei A1-A3  
mit SP abschließen!

### Ausführung:

\*\*\* BREAKPOINT \*\*\*

PC	LABEL	OP	ADR.FELD	A	NZHPC	B	C	D	E	H	L	I	SP
E030		OUT	02		87	00000	00	00000	00	E0	2A	80	FC32

Der Befehl, auf den die Break-Adresse zeigt, ist noch nicht ausgeführt worden.



Einführung in die Programmierung

Arbeitsblatt: Betriebsprogramm-Kommandos

## Das TRACE-INTERVAL-Kommando

Mit dem TRACE-INTERVAL-Kommando ist es möglich, die Protokollierung der Registerinhalte für einen Programmabschnitt (Fenster) eines Programms ein- oder auszuschalten. Start- und Stopadresse des Programmabschnittes müssen angegeben werden - siehe Beispiel.

### Beispiel:

```
KMD > DISASSEMBLER
  START-ADR =0000 E000
  STOP -ADR =0000 E008
E000 3E 00          MVI  A,00
E002 3D          NEXT: DCR  A
E003 02 02E0      JNZ  NEXT
E005 03 4000      JMF  0040
                        END
```

### Aufruf:

```
KMD > TRACE INTERVAL  CR
  EIN/AUS  =A E
  START-ADR =0000 E003  CR
  STOP -ADR =0000 E005  CR

KMD > GO
  START-ADR =1E09 E000

PC LABEL: OP ADR.FELD A NZHPC B C D E H L I SF
E003          JNZ NEXT FF 10010 05 07 FF FA 00 5D 80 FC0E
E002 NEXT:   DCR A
PC LABEL: OP ADR.FELD A NZHPC B C D E H L I SF
E003          JNZ NEXT FE 10100 05 07 FF FA 00 5D 80 FC0E
E002 NEXT:   DCR A
```

Nach dem Einschalten des `Trace-Interval` und Aufruf des G-Kommandos wird das Programm Befehl für Befehl durchlaufen und vom Tracer überprüft, ob der bearbeitete Programmteil im darzustellenden Adressbereich liegt. Ist dies der Fall, so werden, wie beim N-Kommando die Registerinhalte und die Befehle auf dem Monitor dargestellt. In beiden Fällen ergibt sich eine verlängerte Bearbeitungszeit.





## Das IN-Kommando

Mit dem IN-Kommando können von Eingabe-Baugruppen direkt Daten eingelesen werden.

Anwendung: Test und Inbetriebnahme von Eingabe-Baugruppen und von evtl. angeschlossenen Peripherie-Geräten.

Aufruf:

```
      CR
      |
      v
KMD> IN
      PORT-NR=XX 01 CR
      DATEN  =XX  
      DATEN  =XX  
```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen.

- CR: carriage return, Wagenrücklauf  
XX: 1. durch das Betriebsprogramm vorgeschlagene  
      PORT-NR  
      2. Daten vom Eingabe-Port  
  : space, Leertaste



## Das OUT-Kommando

Das OUT-Kommando erlaubt eine direkte Datenausgabe an Ausgabe-Baugruppen.

Anwendung: Test und Inbetriebnahme von Ausgabe-Baugruppen und von evtl. angeschlossenen Peripherie-Geräten.

Aufruf:

```
      ↙ CR
KMD > OUT
      PORT-NR=XX 02 CR
      DATEN  =XX 41  
      DATEN  =41 42  
```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen.

CR: carriage return, Wagenrücklauf

XX: 1. durch das Betriebsprogramm vorgeschlagene  
PORT-NR

2. durch das Betriebsprogramm vorgeschlagene  
Daten

: space, Leertaste



## Das SAVE-Kommando

Mit dem SAVE-Kommando können Programme und Daten über einen Kassetten-Recorder auf Magnetband gespeichert werden. Es ist zusätzlich die Baugruppe 'Kassetten-Interface' erforderlich.

Aufruf:    KMD SAVE CR  
              START-ADR = XXXX E000 CR  
              STOP -ADR = XXXX E120 CR  
              BAND EINSCHALTEN, DANN SPACE SP  
              (Kommandoausführung)  
              KMD \_

## Das LOAD-Kommando

Mit dem LOAD-Kommando werden Daten vom Kassetten-Recorder in den Speicher zurückgelesen. Sollen die Daten nicht in den beim SAVE-Kommando angegebenen Speicherbereich übertragen werden, so kann eine neue Startadresse angegeben werden.

Aufruf:    KMD LOAD TAPE CR  
              START-ADR = XXXX E800 CR  
              SPACE, DANN BAND EINSCHALTEN SP  
              (Kommandoausführung)  
              READY  
              KMD \_

Mögliche Fehlermeldungen:    \*\*\* NICHT HEX = 7F  
                                  \*\*\* CHECKSUM-ERROR

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen-  
alle weiteren Ausdrücke werden vom Betriebsprogramm  
ausgeführt

CR: carriage return, Wagenrücklauf

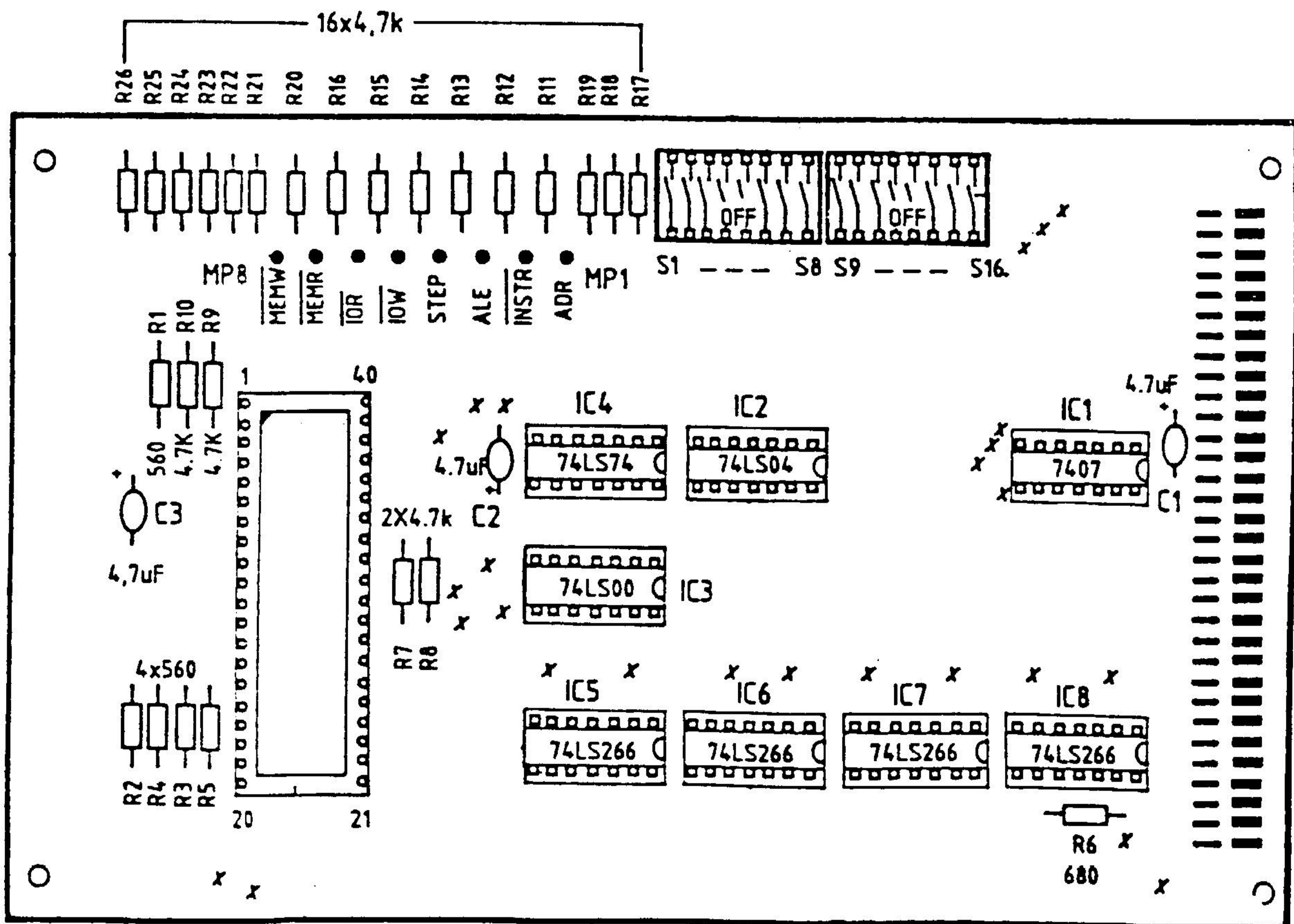
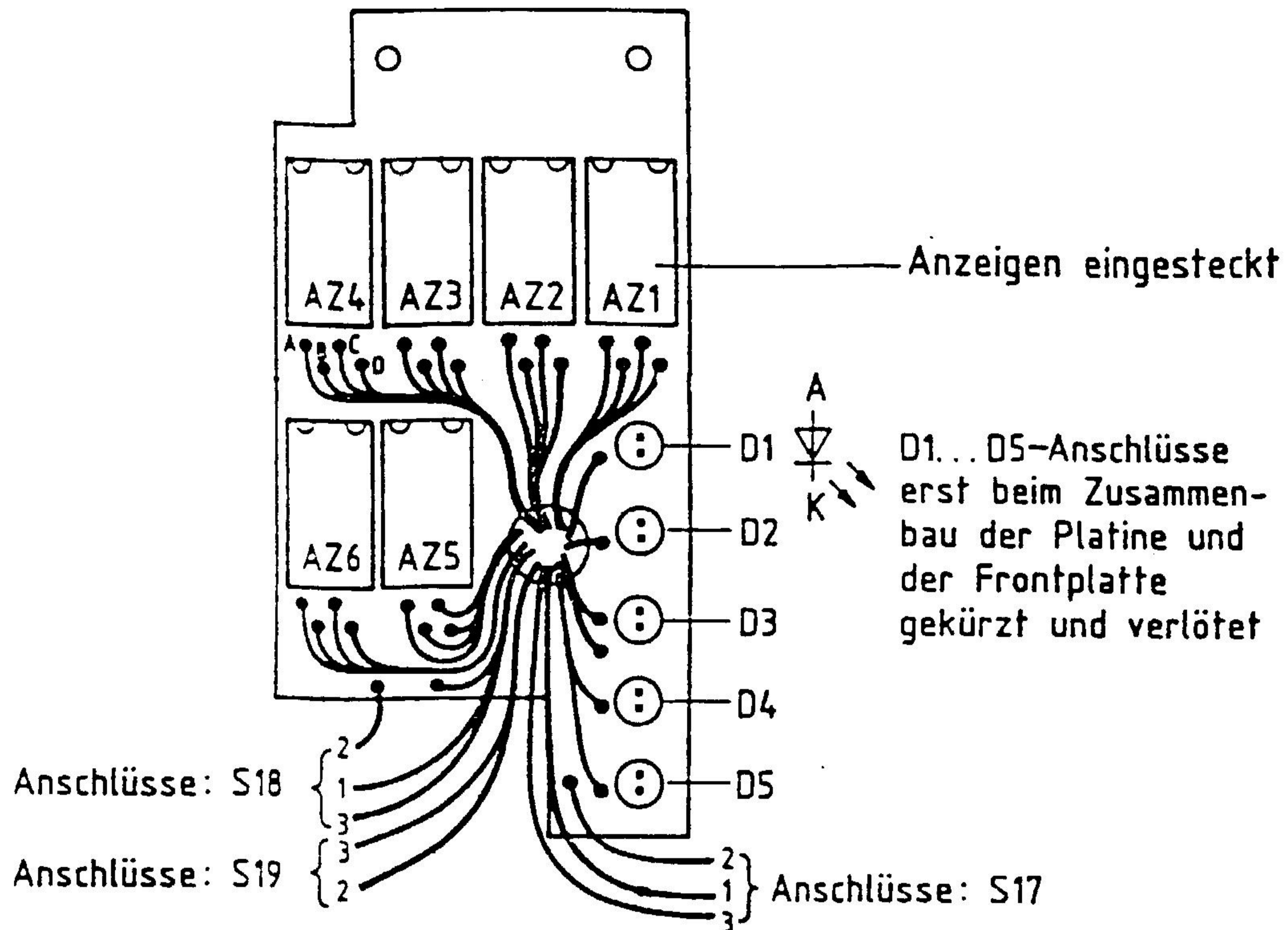
XXXX: Vorschlagsadresse durch das Betriebsprogramm

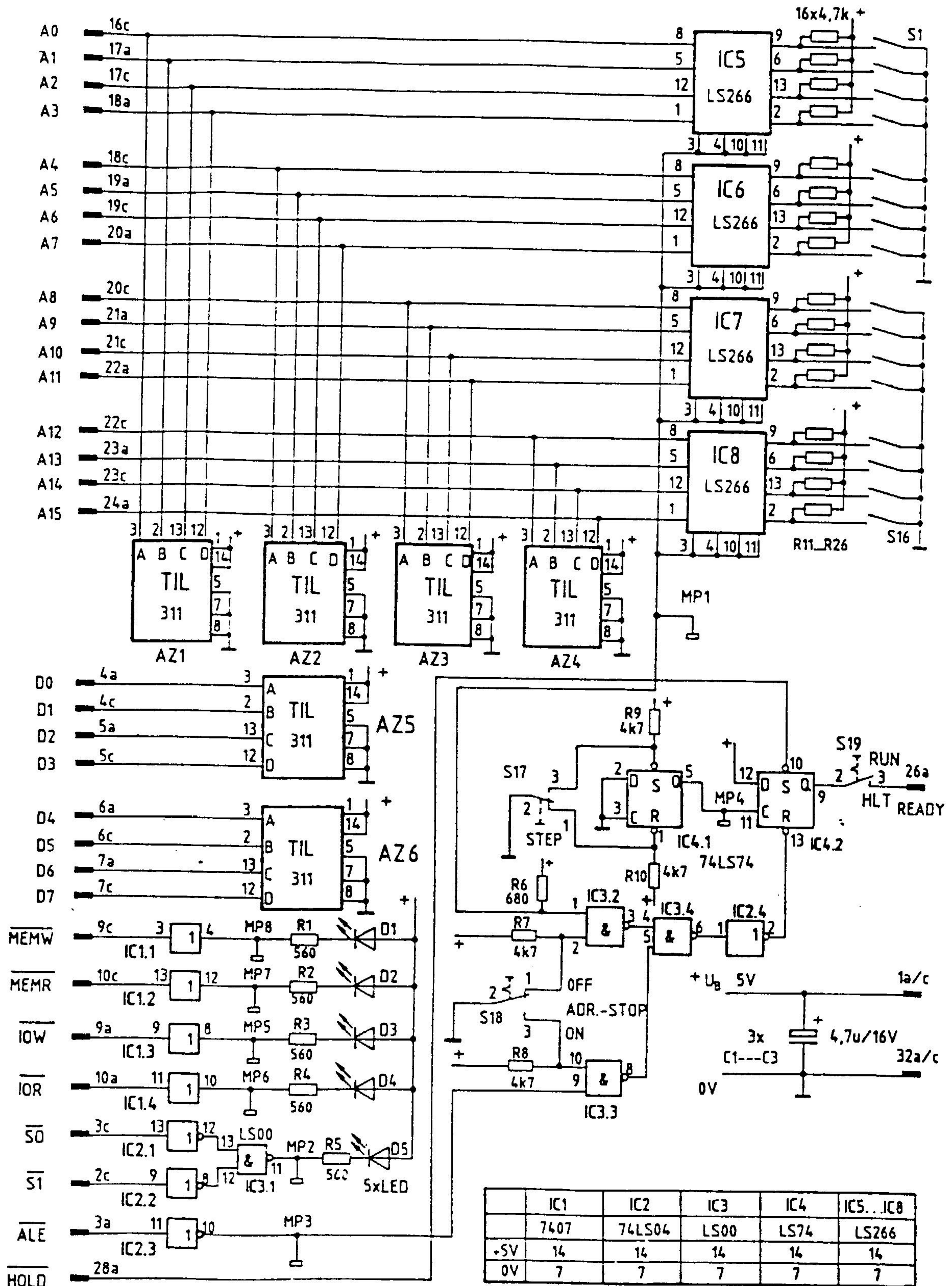
SP: space, Leertaste

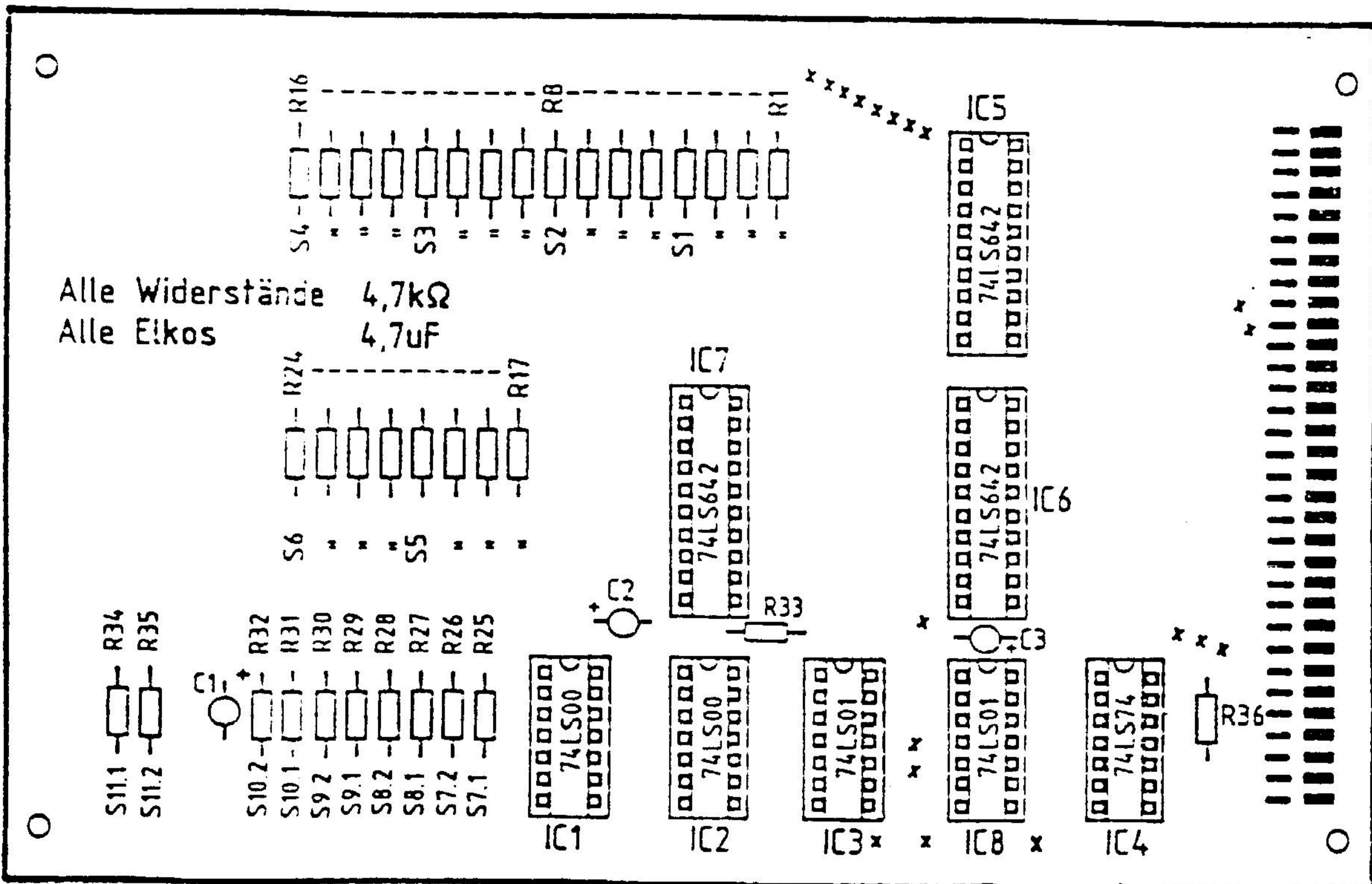


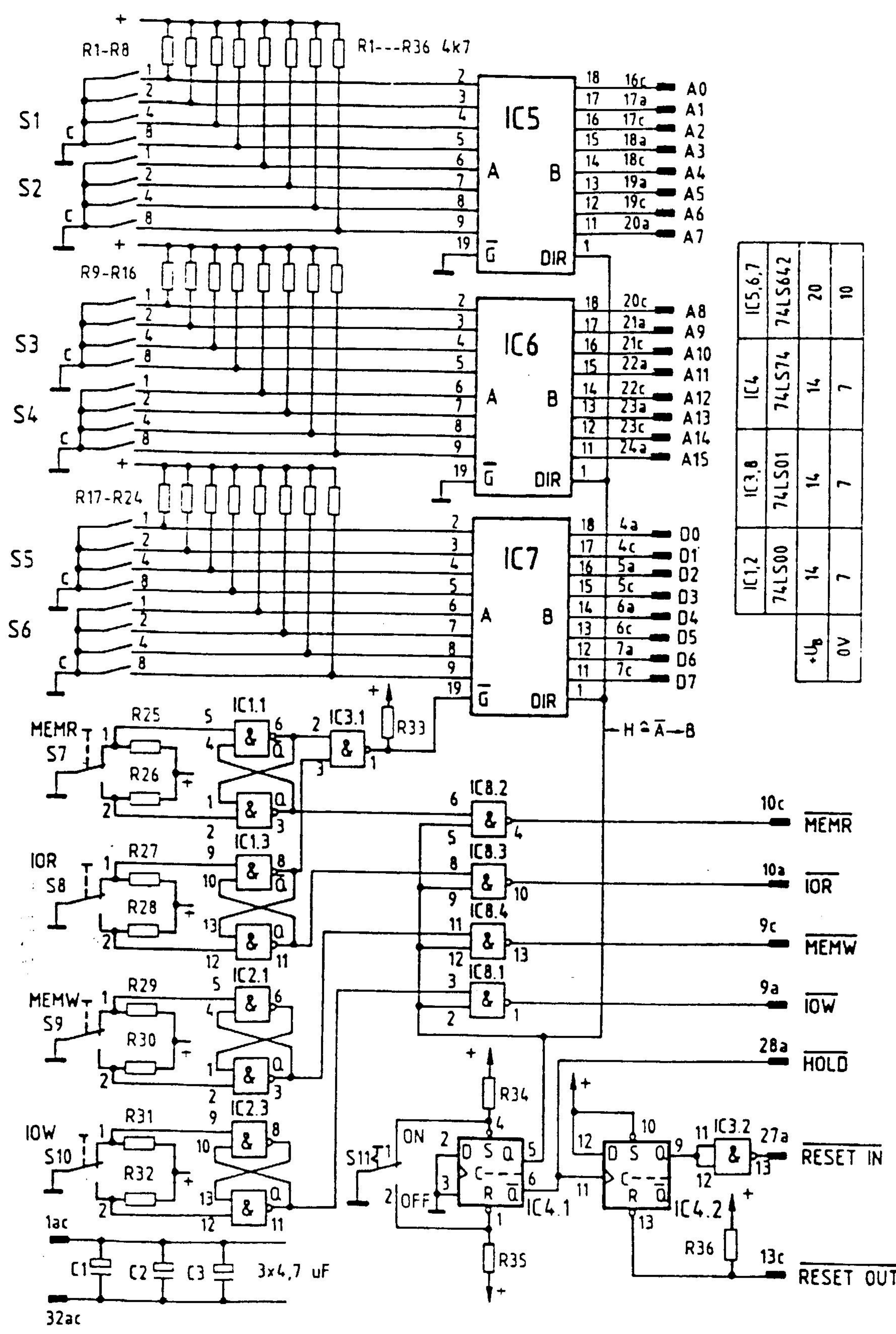
Einführung in die Programmierung

Arbeitsblatt: Betriebsprogramm-Kommandos





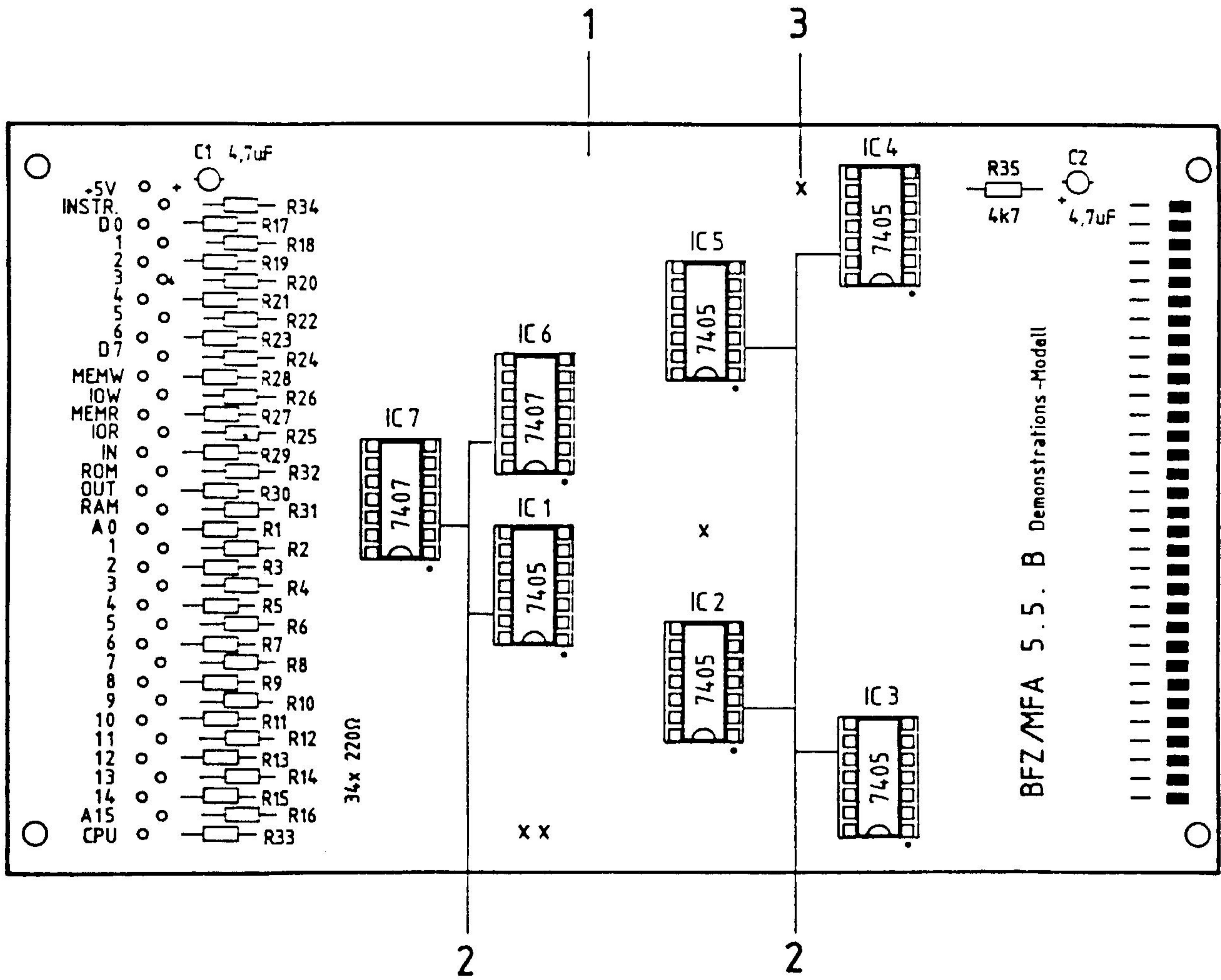




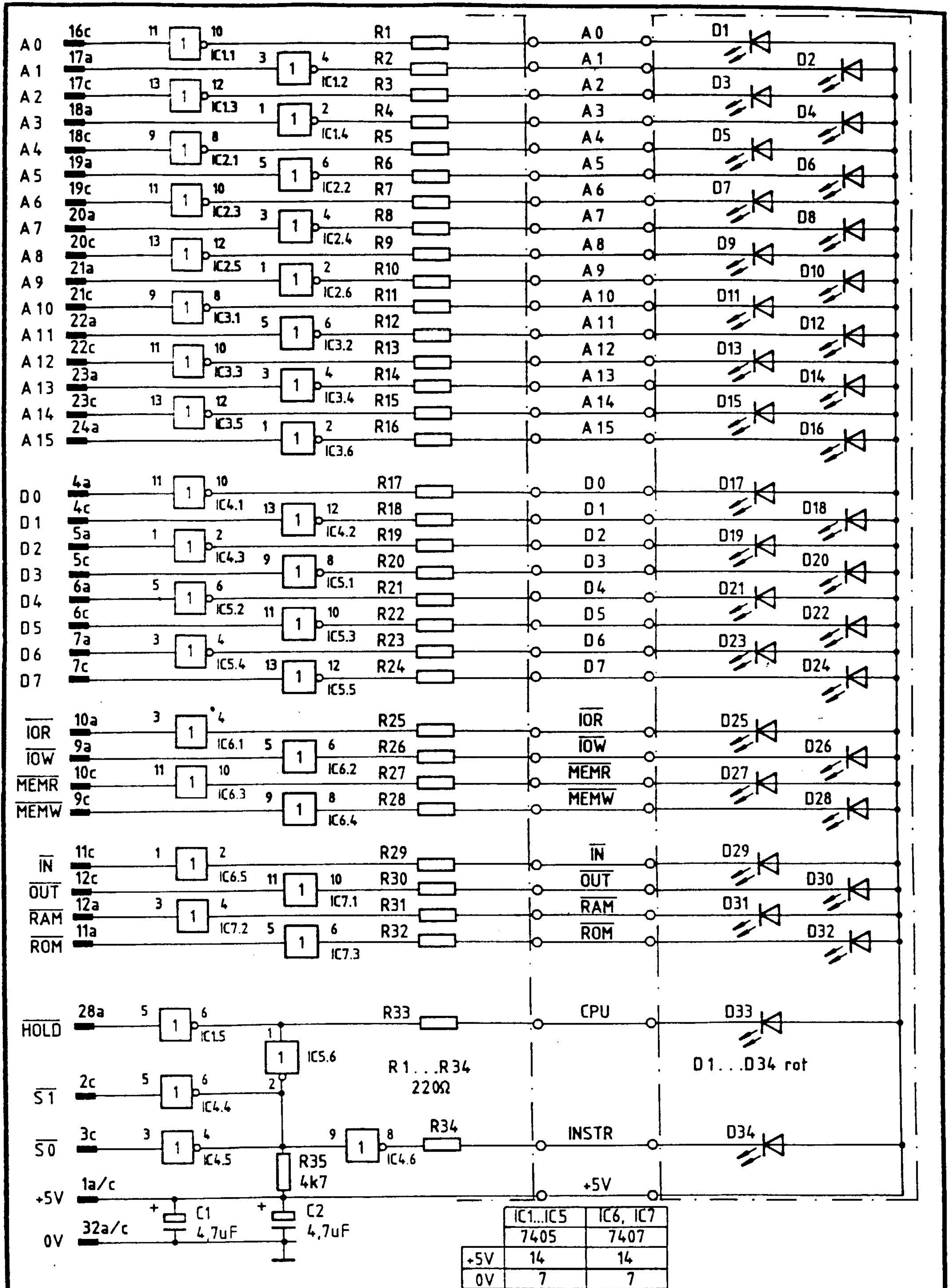
IC5,6,7	74LS642	20	10
IC4	74LS74	14	7
IC3,8	74LS01	14	7
IC1,2	74LS00	14	7
+Ub			0V



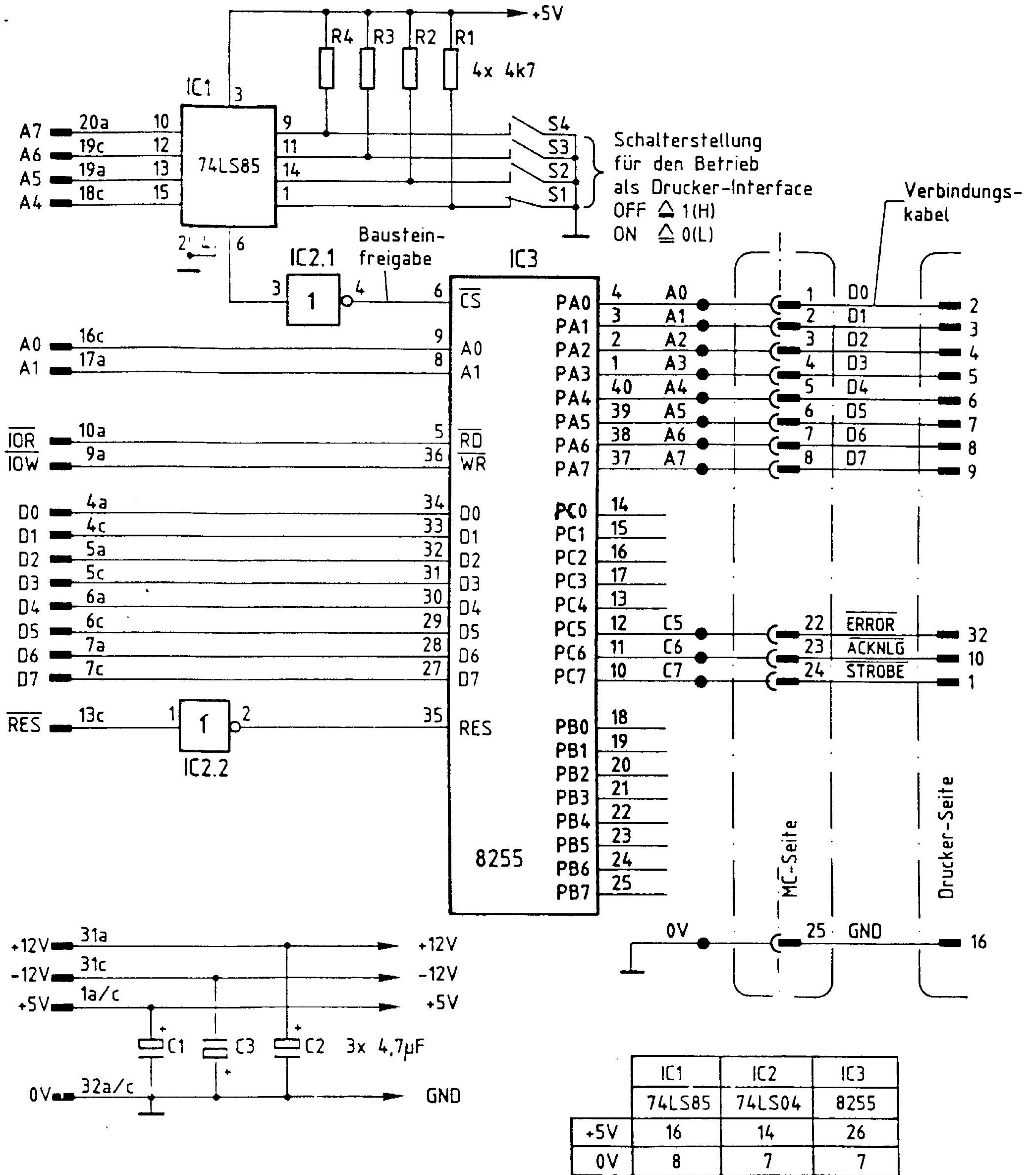
# Bestückungsplan











Teil 3 MAT 85 Eigenheiten, Tabellen, Symbole

-Kurzbeschreibung der Kommandos	S.22
-Restart-Adressen	S.23
-Unterprogramm-Adressen	S.24
-Aufruf und Bedienung der Drucker- und der Eprom-Programmier-Routine	S.27
-ASCII-Code-Tabelle und hexadezimale Verschlüsselung der Tastenfunktionen	S.28
-Bedeutung der Steuerzeichen	S.29
-Symbole für Flußdiagramme	S.30

- HELP \_\_\_\_\_: Dient dazu, alle verfügbaren Kommandos des Betriebsprogramms anzuzeigen.
- MEMORY \_\_\_\_\_: Mit diesem Kommando lassen sich die Inhalte von Speicherzeilen in verschiedenen Formaten ausdrucken und ändern.
- PRINT \_\_\_\_\_: Mit diesem Kommando können die Inhalte von Speicherzeilen in verschiedenen Formaten (Binär, Hexadezimal, Dezimal, ASCII) formatiert (pro Zeile max. 8 Inhalte) ausgedruckt werden.
- GO \_\_\_\_\_: Mit diesem Kommando können eingegebene Programme gestartet werden.
- DISASSEMBLER \_\_\_\_\_: Mit diesem Kommando können Programme, die im Maschinen-Code gespeichert sind, in den Assembler-Code übersetzt werden.
- NEXT INSTRUCTION\_: Mit diesem Kommando wird ein Tracer (Verfolger) aktiviert, der es ermöglicht, die Ausführung und Wirkungsweise einer vorgegebenen Anzahl von Programmbefehlen zu verfolgen. Dazu wird nach jedem Befehl (engl. Instruction) die Programmbe-  
arbeitung kurz unterbrochen und die Inhalte aller CPU-Register werden protokolliert.
- ASSEMBLER \_\_\_\_\_: Mit diesem Kommando wird ein Programm aufgerufen, das es ermöglicht, Anwendungsprogramme im Mnemo-Code (8085-Intel-Format) einzugeben. Der eingegebene Code wird Zeile für Zeile in den zugehörigen Maschinen-Code übersetzt und im RAM-Speicher abgelegt.
- REGISTER \_\_\_\_\_: Mit diesem Kommando können die Anfangswerte der CPU-Register, z.B. vor einem Testlauf des Anwenderprogramms, vorgegeben werden.
- BREAKPOINT \_\_\_\_\_: Dieses Kommando ermöglicht es, mit dem GO-Kommando Unterbrechungspunkte einzugeben. Unterbrechungspunkte (engl. Breakpoints) sind Adressen aus dem Speicherbereich des Anwenderprogramms, an denen die Programmabarbeitung unterbrochen werden soll.  
Nach der Unterbrechung werden die Inhalte der CPU-Register angezeigt.
- TRACE INTERVAL \_\_\_\_: Dieses Kommando bewirkt eine Protokollierung der Registerinhalte immer dann, wenn diejenigen Programmbefehle abgearbeitet werden, die in einem vorher zu bestimmenden Speicherbereich liegen.
- IN \_\_\_\_\_: Dieses Kommando dient dazu, Daten von Eingabe-Ports zu lesen und anzuzeigen.
- OUT \_\_\_\_\_: Dient dazu, Daten an Ausgabe-Ports zu senden.
- SAVE \_\_\_\_\_: Dient dazu, Daten auf einem Kassetten-Recorder zu speichern. Hierzu wird das Kassetten-Interface BFZ/MFA 4.4.a benötigt.
- LOAD TAPE \_\_\_\_\_: Lädt Daten von einer Magnetband-Kassette in den Speicher des Mikrocomputers. Hierzu wird das Kassetten-Interface BFZ/MFA 4.4.a benötigt.



Adresse	Funktion	Ausdruck
0000	RST 0, Monitor-RESET (warmer RESET) im Gegensatz zum "kalten RESET" erfolgt keine Erfassung der Baudrate und kein Ausdruck der Monitorkommandos.	***RESET***
0008	RST 1, Rücksprung aus einem Anwenderprogramm in den Monitor.	***USER***
0010	RST 2, Sprung in den RAM-Bereich nach Adresse FC8CH. (siehe Anmerkung)	
0018	RST 3, Sprung in den RAM-Bereich nach Adresse FC8FH. (siehe Anmerkung)	
0020	RST 4, Sprung zur BREAKPOINT-Routine (02DFH). Ausführung der BREAKPOINT-Routine, wenn <ul style="list-style-type: none"> <li>- Breakpoint's eingeschaltet sind und</li> <li>- die Breakpoint-Adressen nicht alle 0</li> <li>- das Anwenderprogramm eine der eingestellten Breakpoint-Adressen erreicht.</li> </ul> <p><u>Hinweis:</u> Der Registerausdruck gibt den Inhalt der Register nach der Ausführung des Befehls vor dem Breakpoint wieder.</p> <p>Steht ein RST 4 im Programm, ohne das die Breakpoint's eingeschaltet und eine Adresse eingestellt ist, so erfolgt ein Sprung in die Breakpoint-Fehleroutine.</p>	***BREAK-POINT*** und Registerausdruck
0024	TRAP, Neustart des Monitorprogramms mit der erneuten Erfassung der Baudrate. <i>Adresse 0860</i>	***MONITOR RESTART***
0028	RST 5, Sprung in den RAM-Bereich nach Adresse FC92H. (siehe Anmerkung)	
002C	RST 5.5, Sprung in den RAM-Bereich nach Adresse FC95H. (siehe Anmerkung)	
0030	RST 6, Sprung in den RAM-Bereich nach Adresse FC98H. (siehe Anmerkung)	
0034	RST 6.5, Sprung in den RAM-Bereich nach Adresse FC9BH. (siehe Anmerkung)	
0038	RST 7, Sprung in eine Monitorroutine, die einen Programmabbruch signalisiert. Diese Routine wird erreicht, wenn der Prozessor in einem Anwenderprogramm den Befehlscode FFH liest.	***PROGRAMM-ABORT***
003C	RST 7.5, Sprung in den RAM-Bereich nach Adresse FC9EH.	

Adresse: FCDF/FCED

Anmerkung: Übertragungsraten

Diese RESTART-Adressen können vom Anwender benutzt werden. Von den angegebenen RAM-Speicheradressen (angegebene Adresse mitgezählt) stehen drei Speicherplätze für einen Sprungbefehl in ein Programm zur Verfügung, das nach der Unterbrechung abgearbeitet werden soll.

0135: 7200  
004C: 2400  
004P: 4800



Einführung in die Programmierung

RESTART-Adressen

## Unterprogramme des Betriebsprogramms

Unterpr. Name	Eingangs-Adresse	Veränd. Register	Funktion des Unterprogramms
KMD	0040		Rücksprung in die Kommandoroutine des Monitorprogramms und Ausdruck von KMD>.
RCHAR	0043	A	Liest ein Zeichen von der Tastatur ein. Der ASCII-Code des Zeichens steht im Akku. Bei <b>[ESC]</b> Rückkehr in die Kommandoroutine und Klingeln.
	0052	A	Ein Zeichen auf dem Bildschirm bzw. Drucker ausgeben. Das auszugebende Zeichen muß im Akkumulator (A) stehen.
WCHARI	0055		Gibt 1 Zeichen, das nach dem CALL-Befehl im Speicher steht, auf Bildschirm und Drucker (wenn Ein) aus. Das auszugebende Zeichen muß mit der DB-Anweisung in den Speicher geschrieben werden. Beispiel: CALL 0055 DB 'A' ; A wird ausgegeben
WAHEX	0058		Gibt den Akku-Inhalt (8Bit) als zwei Hexadezimalziffern auf dem Bildschirm/Drucker aus.
WHLHEX	005B	H,L	Gibt den HL-Registerinhalt (16Bit) als vier Hexadezimalziffern auf dem Bildschirm/Drucker aus.
WABIN	005E	A	Gibt den Akku-Inhalt (8Bit) als Binärzahl am Bildschirm/Drucker aus. Beispiel: MVI A,23 ; 23 Hexadezimal CALL 005E ; wird als 00100011 ausgegeben
WADEZ	0061	A	Gibt den Akku-Inhalt (8Bit) als Dezimalzahl auf dem Bildschirm/Drucker aus.  MVI A,23 ;23 Hexadezimal wird CALL 0061 ;als 35 ausgegeben



Einführung in die Programmierung

Unterprogramm-Adressen

Unterpr. Name	Eingangs-Adresse	Veränd. Register	Funktion des Unterprogramms
WAFOR	0064	A,C	Gibt den Akku-Inhalt (8Bit) in einem der zu wählenden Formate ASCII-Binär-Dezimal-Hex auf dem Bildschirm/Drucker aus. Das Format wird durch den Inhalt des Registers C wie folgt gewählt: 0 → ASCII-Zeichen 1 → Binärzahl 2 → Dezimalzahl 3 → Hexadezimalzahl  Beispiel: <pre>MVI A,23      ; 23 Hexadezimal wird MVI C,1      ; als 00100011 ausge- CALL 0064    ; geben</pre>
WBLANK	0067		Gibt ein Leerzeichen (Blank) auf dem Bildschirm/Drucker aus.
WBUFI	006D		Gibt den hinter dem CALL-Befehl stehenden Text auf dem Bildschirm aus. Der Text muß mit der DB-Anweisung in den Speicher (Textpuffer) geladen werden. Am Ende des Textes muß als Enderkennung eine 0 stehen. Beispiel: <pre>CALL 006D DB ' DIES IST EINE UEBUNG', 0</pre> Gibt den Text DIES IST EINE UEBUNG aus.
WCRLF	0073		Gibt einen Wagenrücklauf (CR), eine Neue Zeile (LF, Line Feed) und Text in diese neue Zeile aus. Der Text muß wie bei "WBUFI" vorher eingegeben werden.
HADR	08DF		Liest eine 16-Bit-Adresse (4 Hex-Stellen) von der Tastatur ein und speichert sie im Doppelregister H ab. Dabei gelangt der höherwertige Teil der Adresse ins H-Register und der niederwertige Teil ins L-Register. Die Eingabe der Adresse muß mit <span style="border: 1px solid black; padding: 0 2px;">CR</span> oder <span style="border: 1px solid black; padding: 0 2px;">SP</span> abgeschlossen werden.
BSTIME	0895	A,D,E	Zeitverzögerung von 0,24 s. Damit die Inhalte der Register A,D und E vor Aufruf des Unterprogramms gerettet und nachher





Unterpr. Name	Eingangs-Adresse	Veränd. Register	Funktion des Unterprogramms
BSTIME	0895	A,D,E	wiederhergestellt werden, muß die folgende Befehlsfolge eingehalten werden:  PUSH PSW ;Registerinhalte A,D,E PUSH D ;retten CALL 0895 ;Zeitverzögerung POP D ;Registerinhalte wiederherstellen POP PSW ;stellen
CMP2	0EA8	A	Vergleicht die Inhalte der Register DE mit denen der Register HL. Wenn (HL) > (DE) ist, wird das Carry-Flag auf 1 gesetzt, sonst auf 0. Die zu vergleichenden Inhalte müssen vor Aufruf des Unterprogramms in die Doppelregister D und H geladen werden. Beispiel: LXI D, Zahl 1 LXI H, Zahl 2 CALL CMP2
SUB2	1039	A,HL, DE	Subtrahiert die 16-Bit-Zahl im Doppelregister D von der 16-Bit-Zahl im Doppelregister H. Das Ergebnis steht dann im Doppelregister H. [(HL) = (HL) - (DE)]
WBUF	OBA1		Gibt Text aus einem Textpuffer aus, dessen Anfangsadresse durch den Inhalt des HL-Registers adressiert ist. Der Text wird mit der DB-Anweisung ab dieser Adresse geladen, das Textende muß mit 0 gekennzeichnet sein. Nach der Ausgabe des Textes zeigt das HL-Register auf die Adresse nach dem Endezeichen.



Druckerroutine:

Aufruf: KMD > GO  
START-ADR = 1E00 ;Drucker "EIN"

Ausdruck: \*\*\* PRINTER ON \*\*\* ;Drucker betriebsbereit  
oder  
\*\*\* PRINTER NOT READY \*\*\* ;Drucker nicht betriebs-  
bereit

Aufruf: KMD > GO  
START-ADR = 1E03 ;Drucker "AUS"

Ausdruck: \*\*\* PRINTER OFF \*\*\*

EPROM-Programmerroutine:

Aufruf: KMD > GO  
START-ADR = 1E06

Ausdruck: \*\*\* EPROM-PROGRAMMER \*\*\*

mögliche Kommandos { TEST  
READ  
PROG  
COMP  
QUIT

Bedeutung der Kommandos:

TEST: Test ob EPROM gelöscht ist.

READ: EPROM-Inhalt in anzugebenden RAM-Speicherbereich lesen.

PROG: Programmieren von der anzugebenden Speicheradresse aus.

COMP: EPROM-Inhalt mit dem Inhalt des anzugebenden Speicher-  
bereiches vergleichen.

QUIT: Rückkehr in das Monitorprogramm.

KMD > GO  
START-ADR = 1E09 1E06

\*\*\* EPROM-PROGRAMMER \*\*\*

TEST  
READ  
PROG  
COMP  
QUIT

\* TEST  
READY

\* READ  
START-ADR = E000  
READY

\* PROG  
START-ADR = E000  
READY

\* COMP  
START-ADR = E000  
READY

\* QUIT

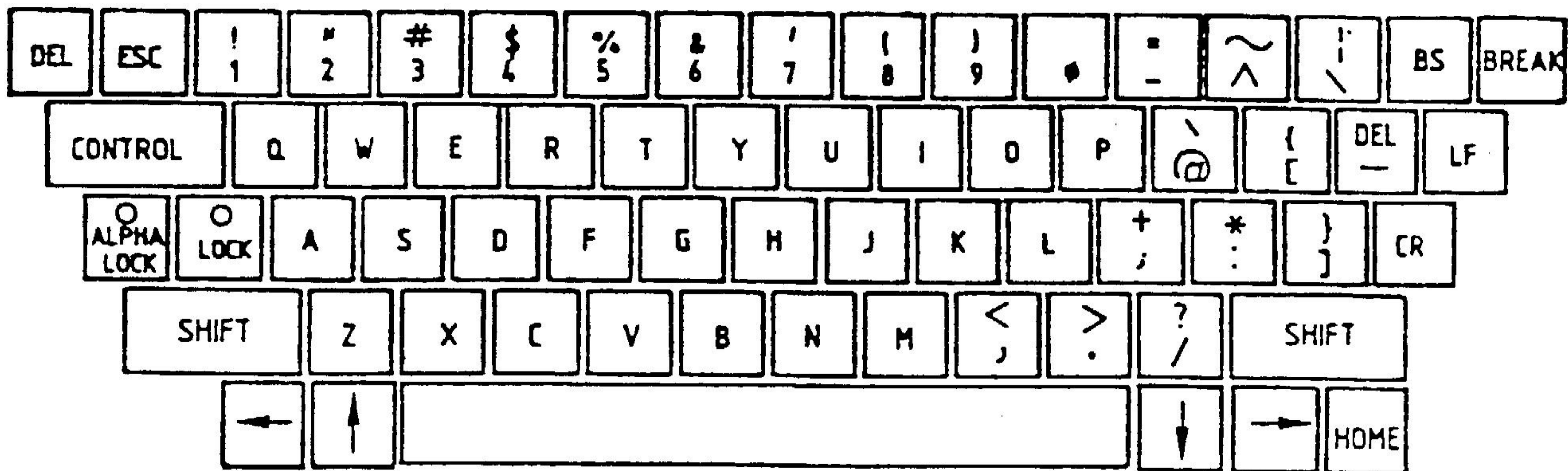
KMD >



# ASCII-Code-Tabelle (American Standard Code for Information Interchange)

	B3 0 B0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
B6 0 B4 000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1 001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	VS
2 010	SPC	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3 011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4 100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5 101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6 110	\	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7 111	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Beschriftung der Tasten und hexadezimale Verschlüsselung der Tastenfunktion



7F	1B	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	BREAK
CONTROL	11	17	05	12	14	19	15	09	0F	10	00	1B	1F	0A	51	57	45	52	54	59	55	49	4F	50	60	7B	7F	0A	71	77	65	72	74	79	75	69	6F	70	40	5B	5F	0A																																																																																																																																																													
ALPHA LOCK	01	13	04	06	07	08	0A	0B	0C	3B	3A	1D	0D	41	53	44	46	47	48	4A	4B	4C	3B	3A	7D	0D	61	73	64	66	67	68	6A	6B	6C	3B	3A	5D	0D																																																																																																																																																																
SHIFT	1A	1B	03	16	02	0E	0D	2C	2E	2F	SHIFT	5A	5B	63	56	42	4E	4D	3C	3E	3F	SHIFT	7A	7B	63	76	62	6E	6D	2C	2E	2F	SHIFT																																																																																																																																																																						
	08	0B	20											08	0B	20											08	0B	20																																																																																																																																																																										



Einführung in die Programmierung


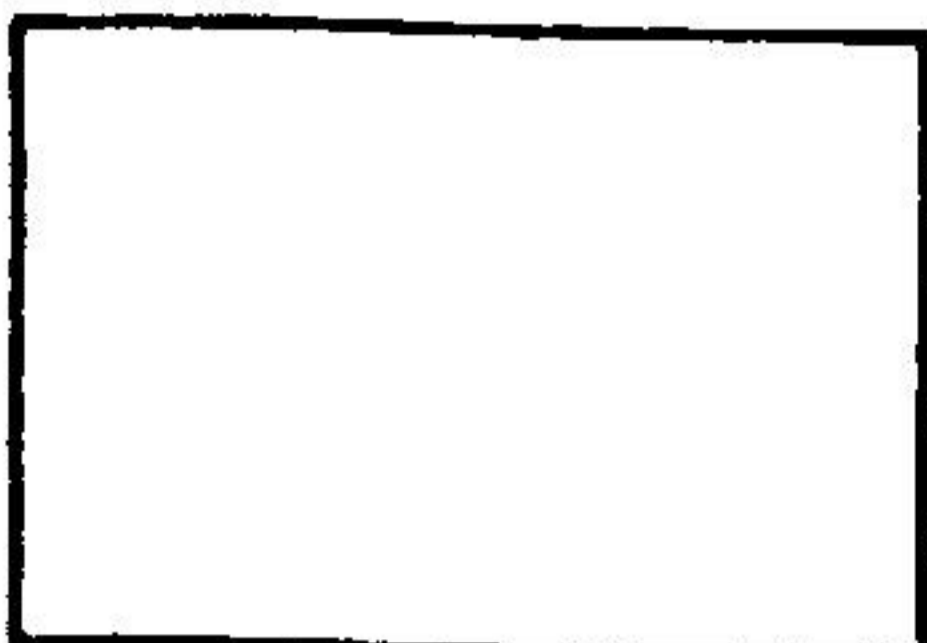
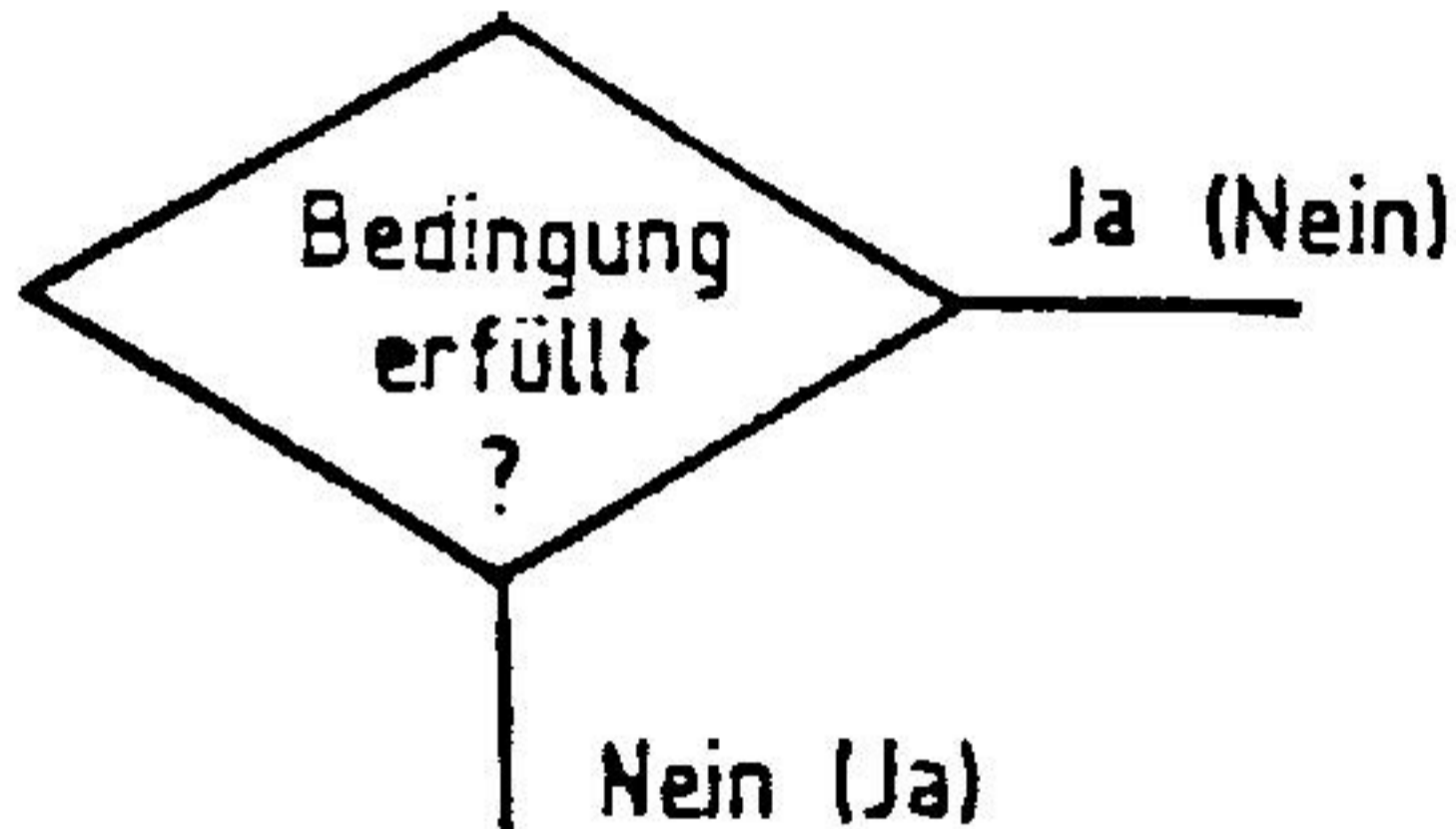
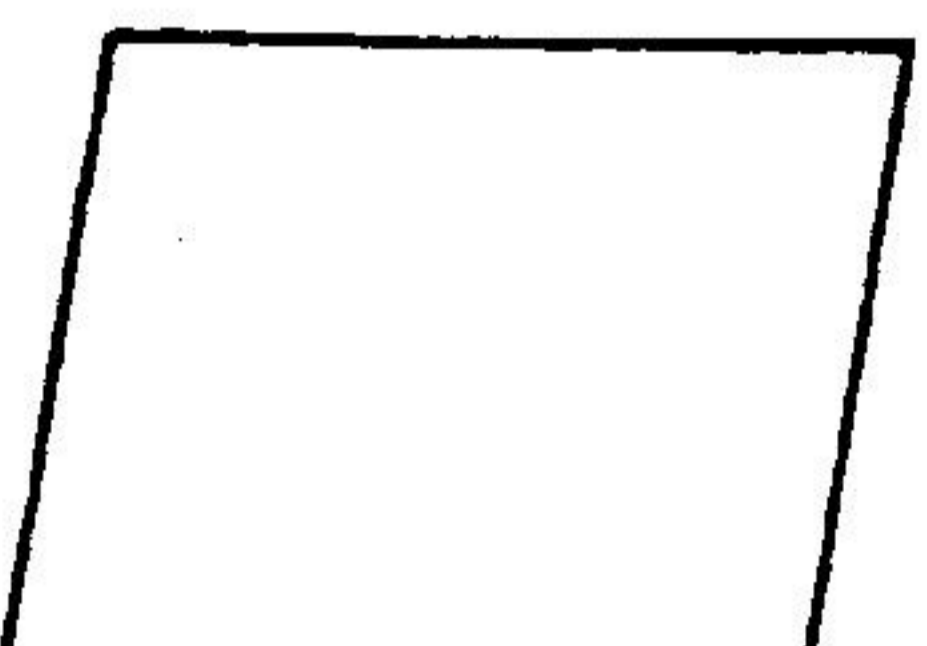



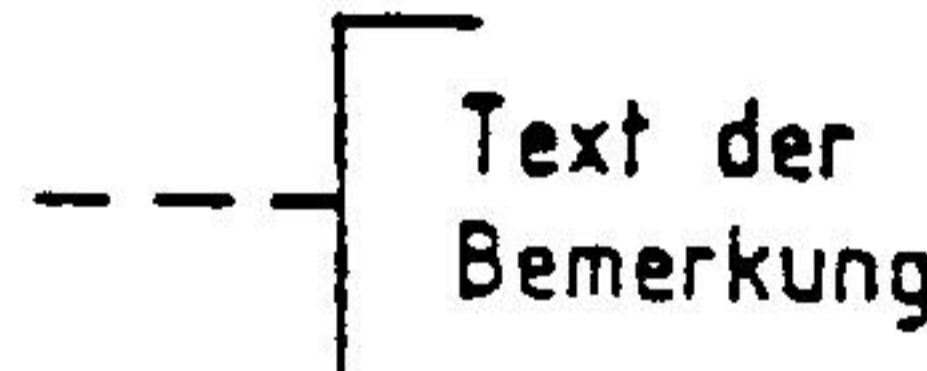

ASCII-Code-Tabelle

Hexadezimale Verschlüsselung der Tasten

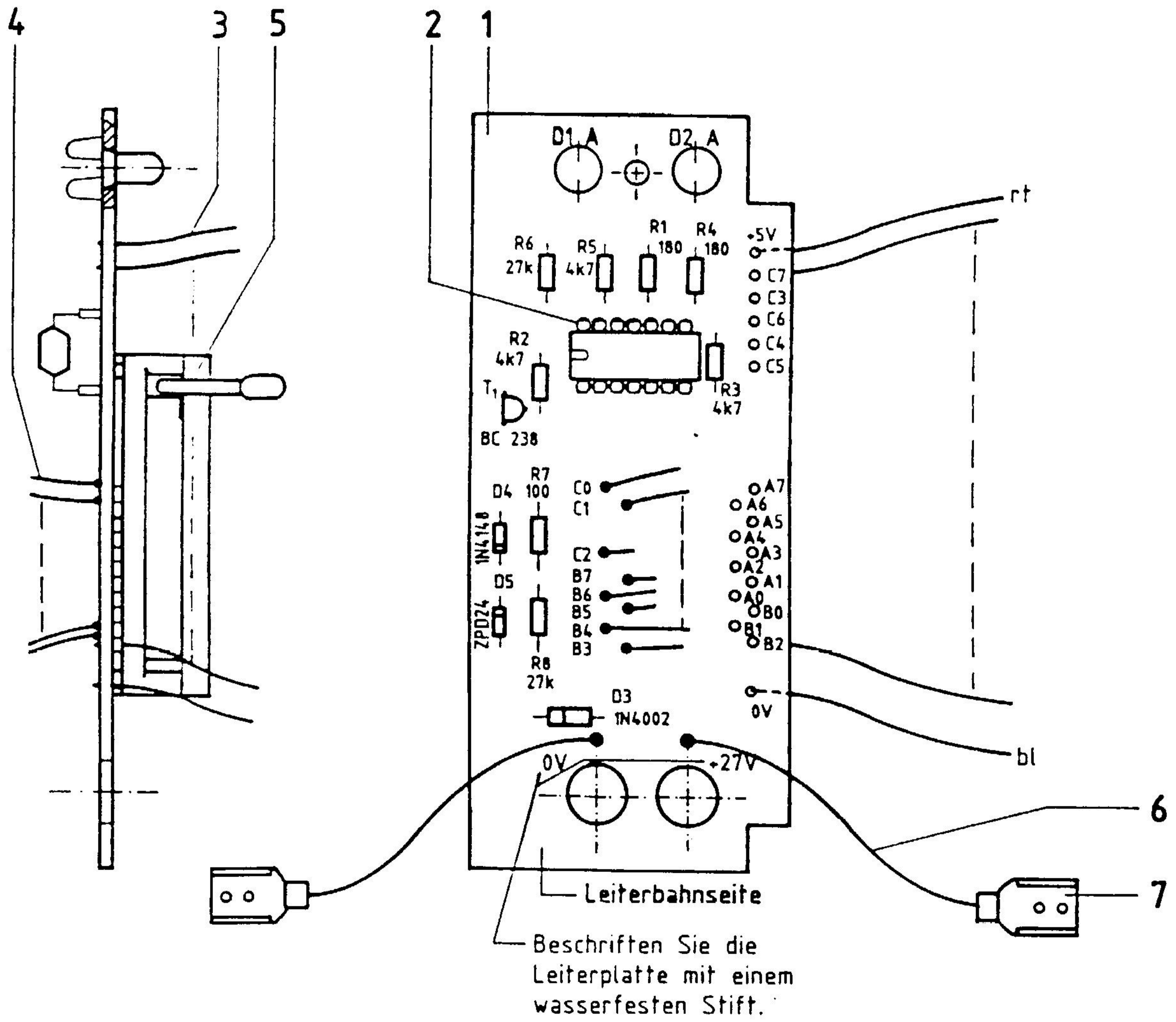
Die Bedeutung der Steuerzeichen (00H-20H und 7FH)

Code		Bedeutung	Steuerung des Cursors im Video-Interface BFZ/MFA 8.2.
Hex	ASCII		
00	NUL	Null, Nichts	
01	SOH	Kopfzellenbeginn	
02	STX	Textanfangszeichen	
03	ETX	Textendezeichen	
04	EOT	Ende der Übertragung	
05	ENQ	Aufforderung zur Datenübertragung	
06	ACK	Positive Rückmeldung	
07	BEL	Klingelzeichen	
08	BS	Rückwärtsschritt, (←)	um 1 Stelle nach links
09	HT	Horizontal Tabulator, (→)	um 1 Stelle nach rechts
0A	LF	Zeilenvorschub, (↓)	um 1 Stelle nach unten
0B	VT	Vertikal Tabulator, (↑)	um 1 Stelle nach oben
0C	FF	Seitenvorschub	Bildschirm löschen und zurück nach links oben (Zeitdauer dazu ca. 150ms)
0D	CR	Wagenrücklauf	zurück zum Zeilenanfang und Löschen des Zeilenendes.
0E	SO	Dauerumschaltzeichen	
0F	SI	Rückschaltzeichen	
10	DLE	Datenübertragungsumschaltung	
11	DC1	Gerätesteuerzeichen 1	
12	DC2	Gerätesteuerzeichen 2	
13	DC3	Gerätesteuerzeichen 3	
14	DC4	Gerätesteuerzeichen 4	
15	NAK	Negative Rückmeldung	
16	SYN	Synchronisierung	
17	ETB	Ende des Datenübertragungsblocks	
18	CAN	Ungültig	
19	EM	Ende der Aufzeichnung	
1A	SUB	Substitution	
1B	ESC	Umschaltung	um 1 Zeile nach unten ohne Löschen der letzten Zeile
1C	FS	Hauptgruppentrennzeichen	zurück nach links oben
1D	GS	Gruppentrennzeichen	zurück zum Zeilenanfang
1E	RS	Untergruppentrennzeichen	
1F	US	Teilgruppentrennzeichen	
20	SP	Leerzeichen	
7F	DEL	Löschen des vorhergehenden Zeichens	



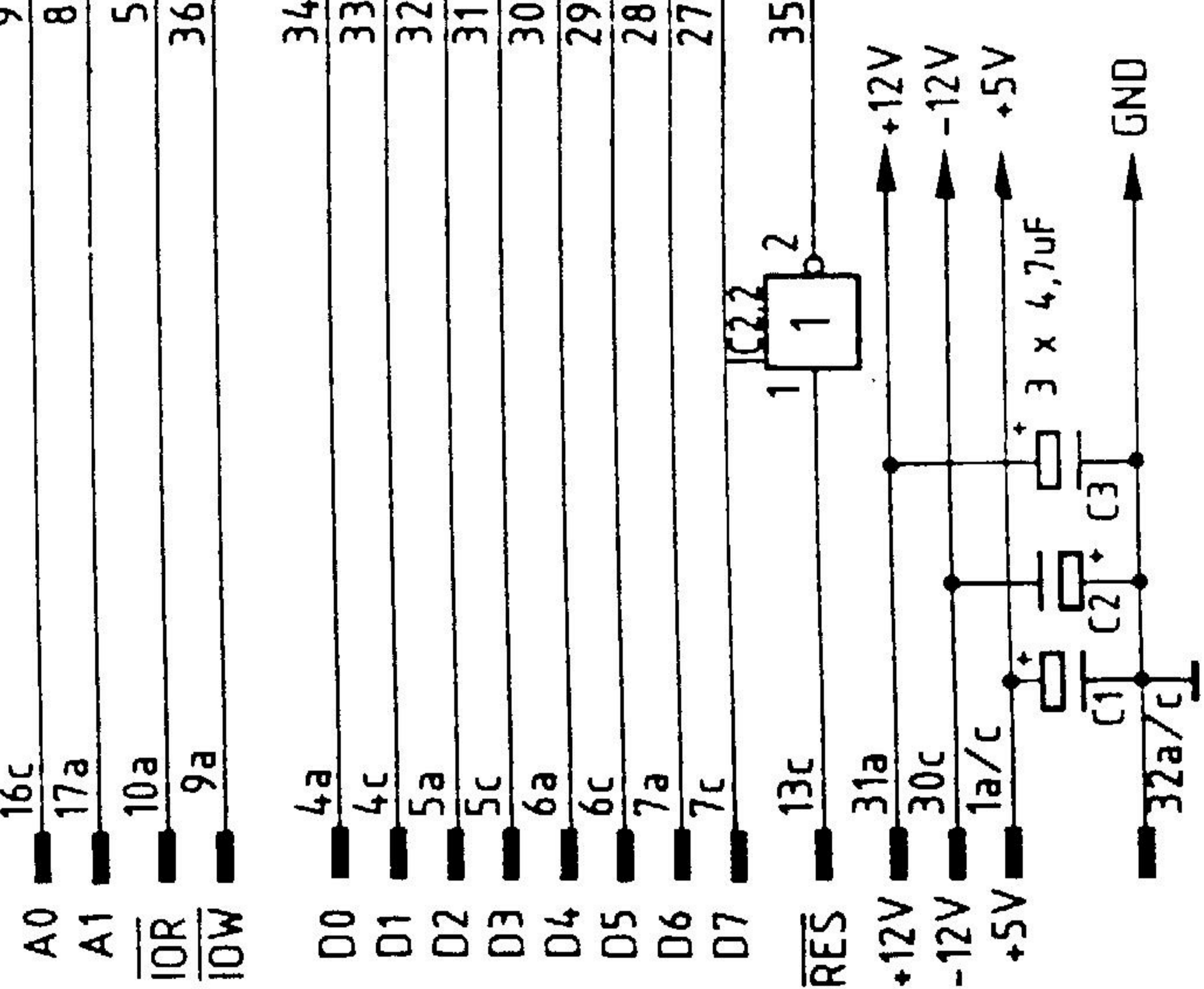
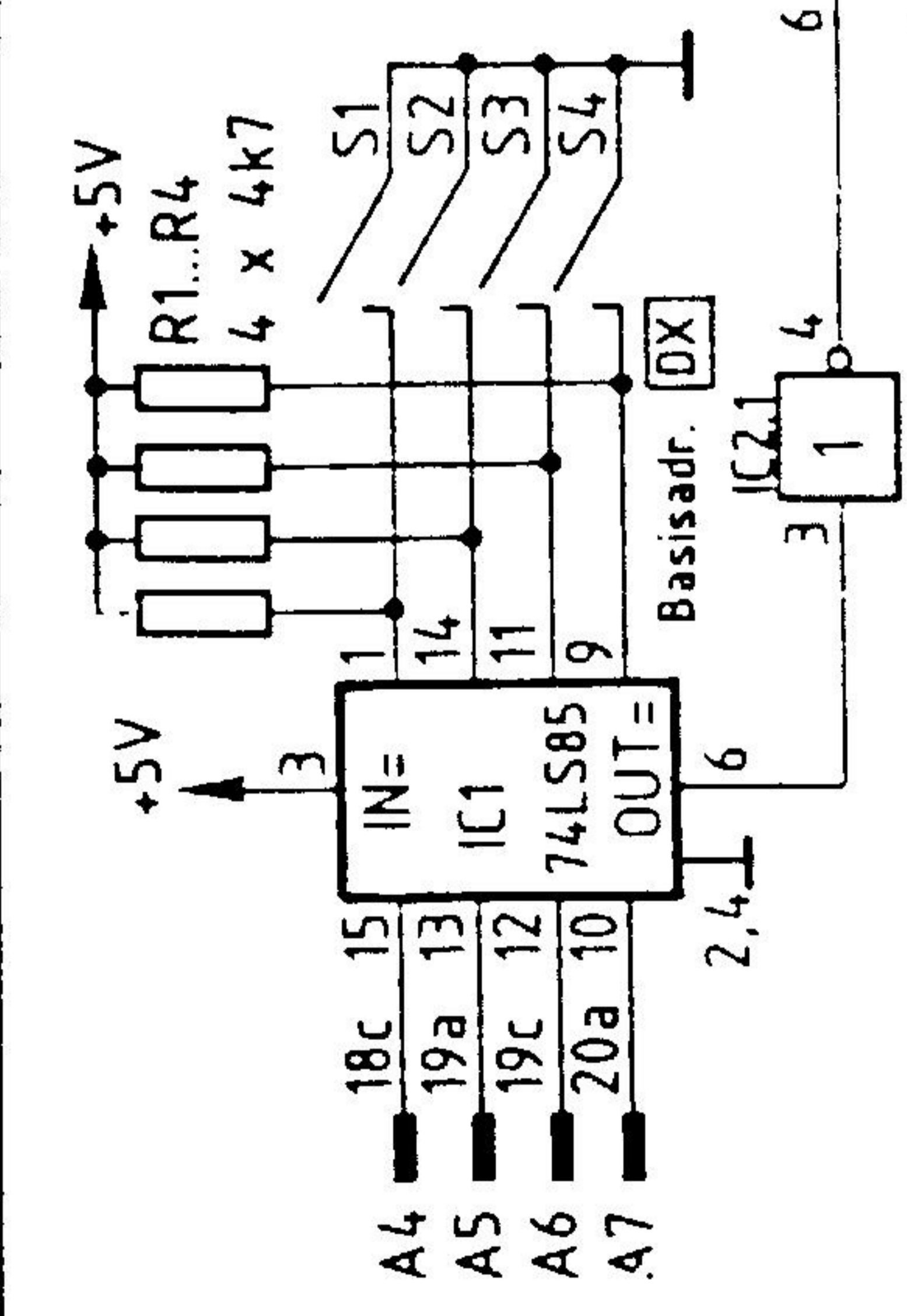
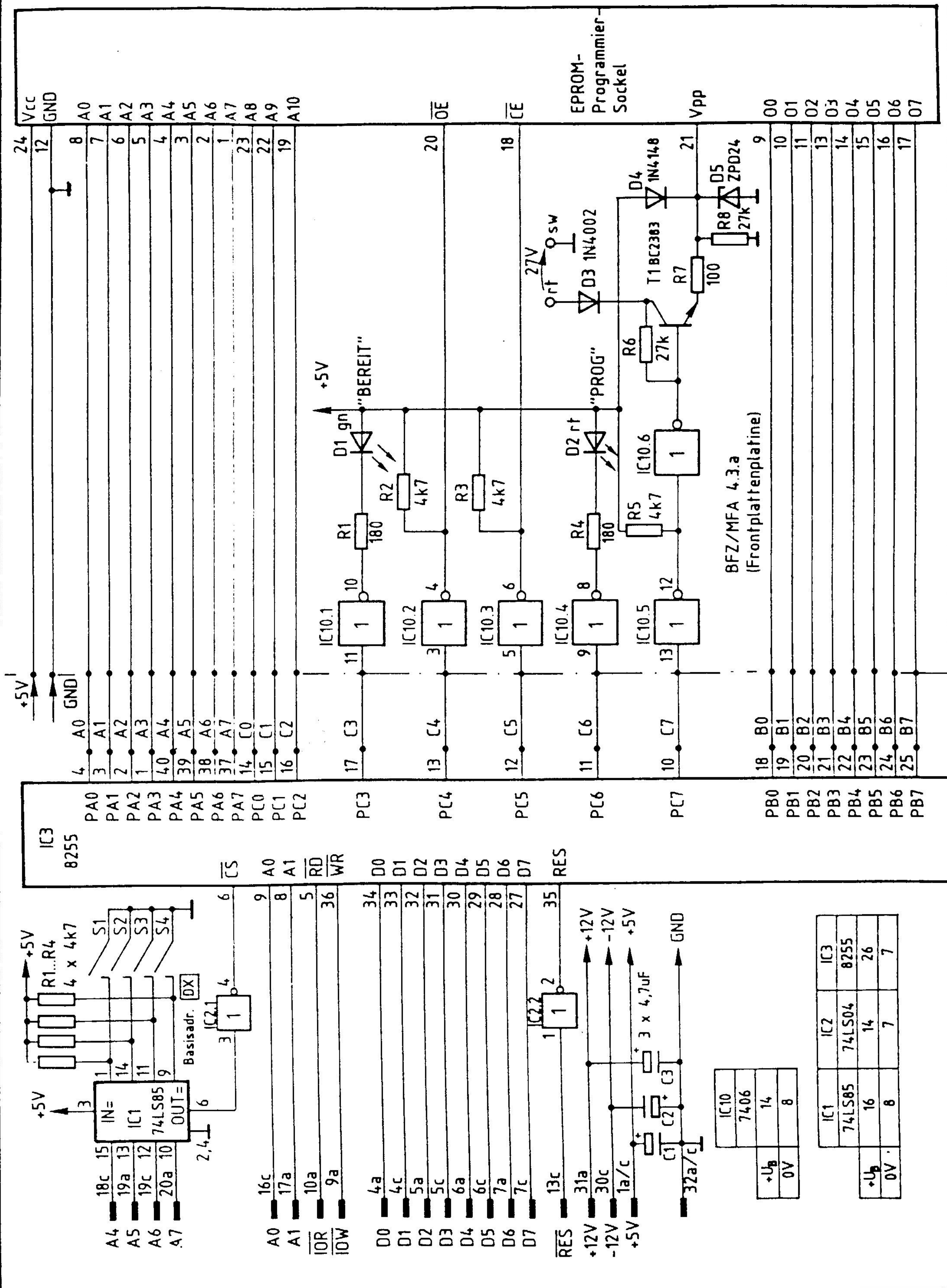
Symbole	Bedeutung
	Grenzstelle (Beginn, Ende ...)
	allgemeine Operation
	Verzweigung
	Ein-/Ausgabe- Operation
	Unterprogramm
	Flußlinie (Richtung der Abarbeitung)
	Zusammenführung
	Bemerkung
	Nahtstelle

Bestückungsplan





Stromlaufplan EPROM-Programmierer



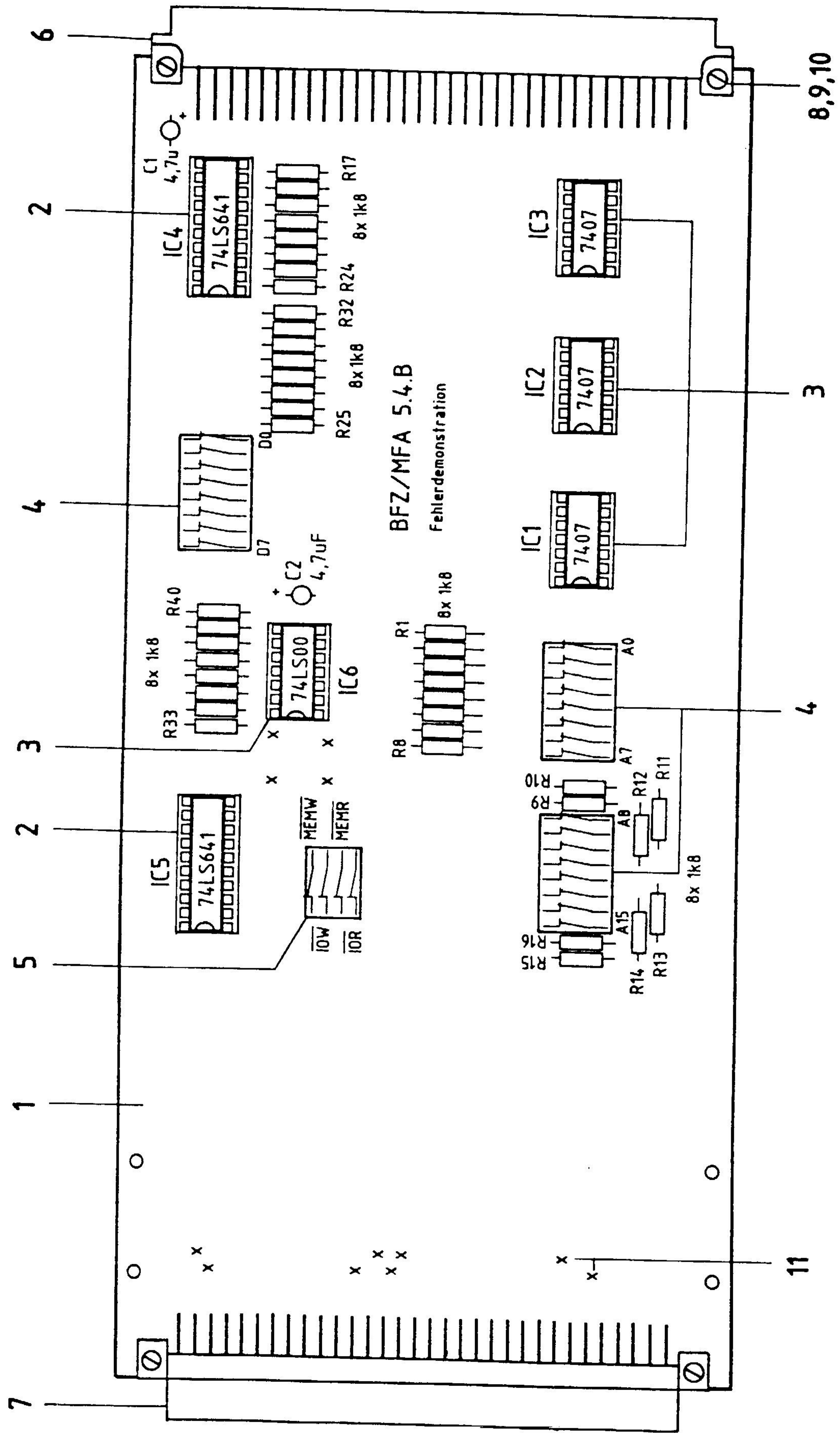
IC10	IC2	IC3
7406	74LS04	8255
14	14	26
8	7	7

IC1	IC2	IC3
74LS85	74LS04	8255
16	14	26
8	7	7

IC3	8255
PA0	PC3
PA1	PC4
PA2	PC5
PA3	PC6
PA4	PC7
PA5	PB0
PA6	PB1
PA7	PB2
PC0	PB3
PC1	PB4
PC2	PB5
	PB6
	PB7

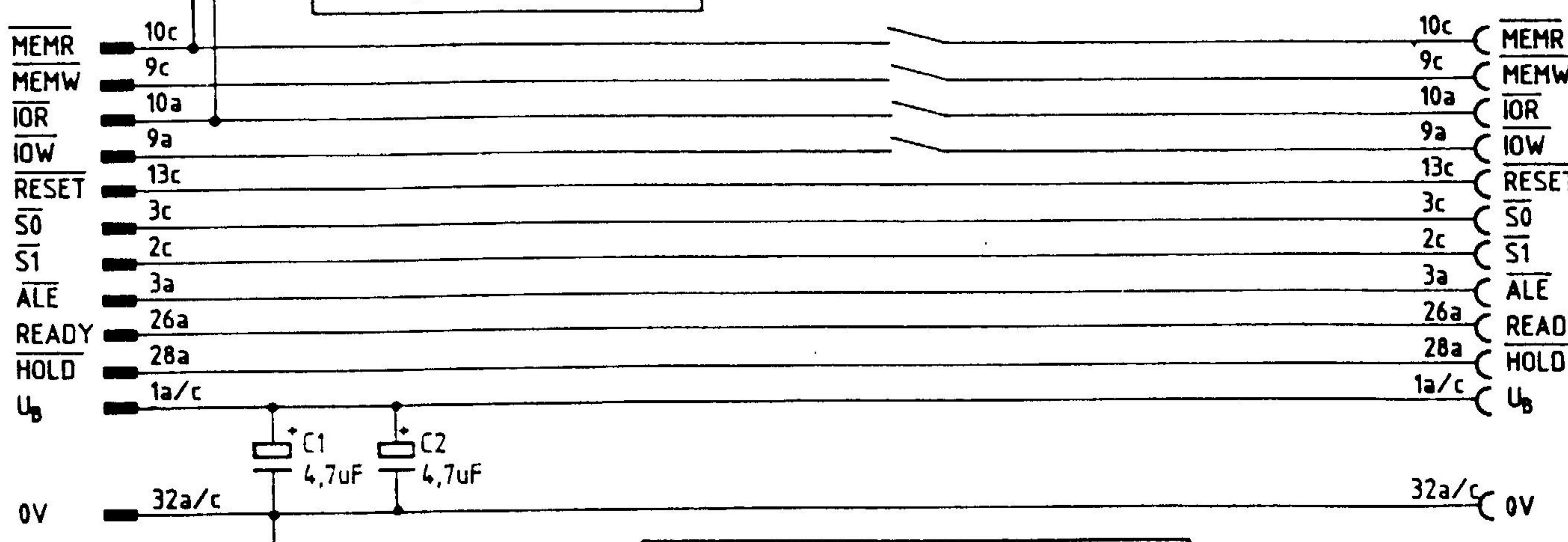
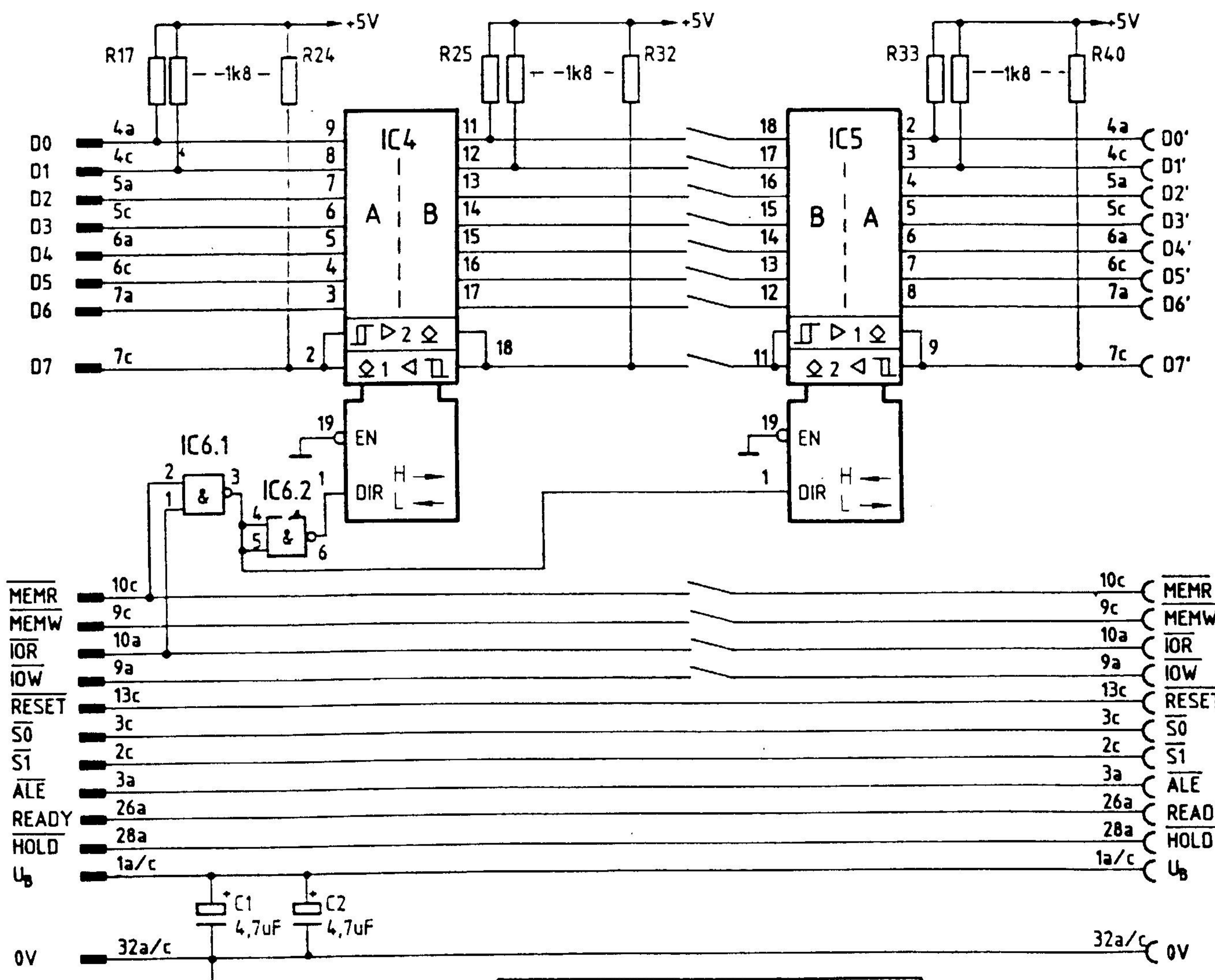
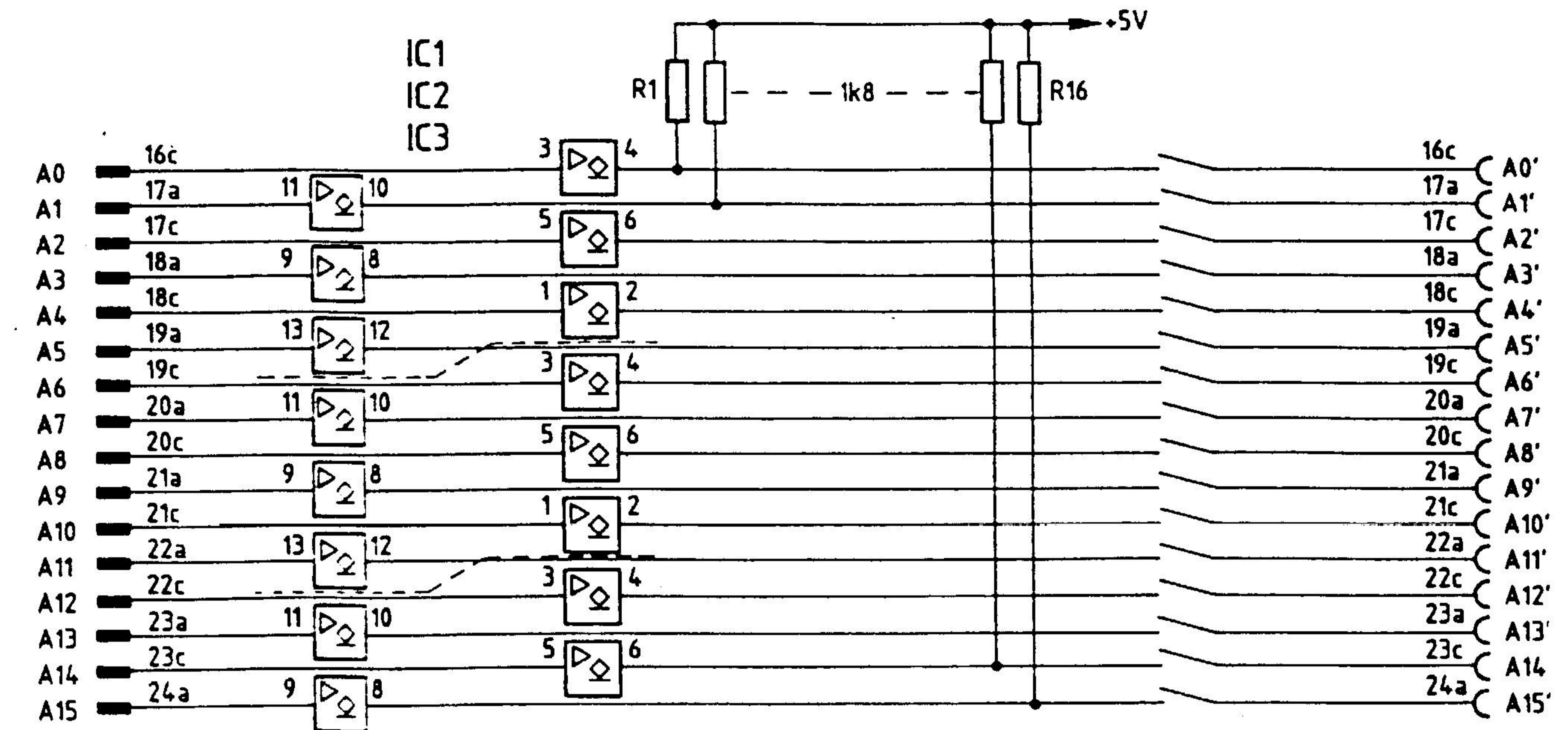
BFZ/MFA 4.3.a  
(Frontplattenplatine)

EPROM-  
Programmier-  
Socket



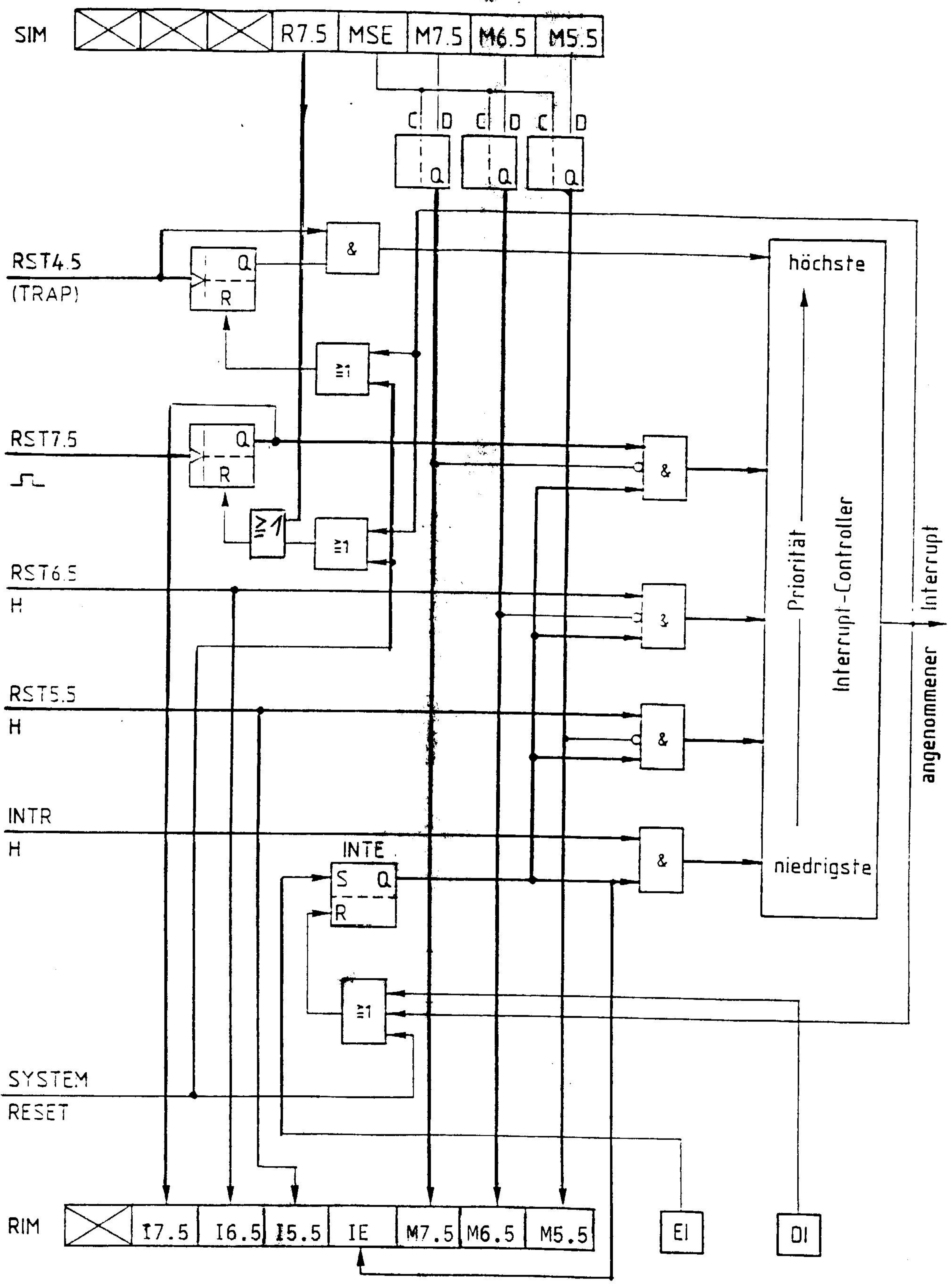
Bestückungsplan

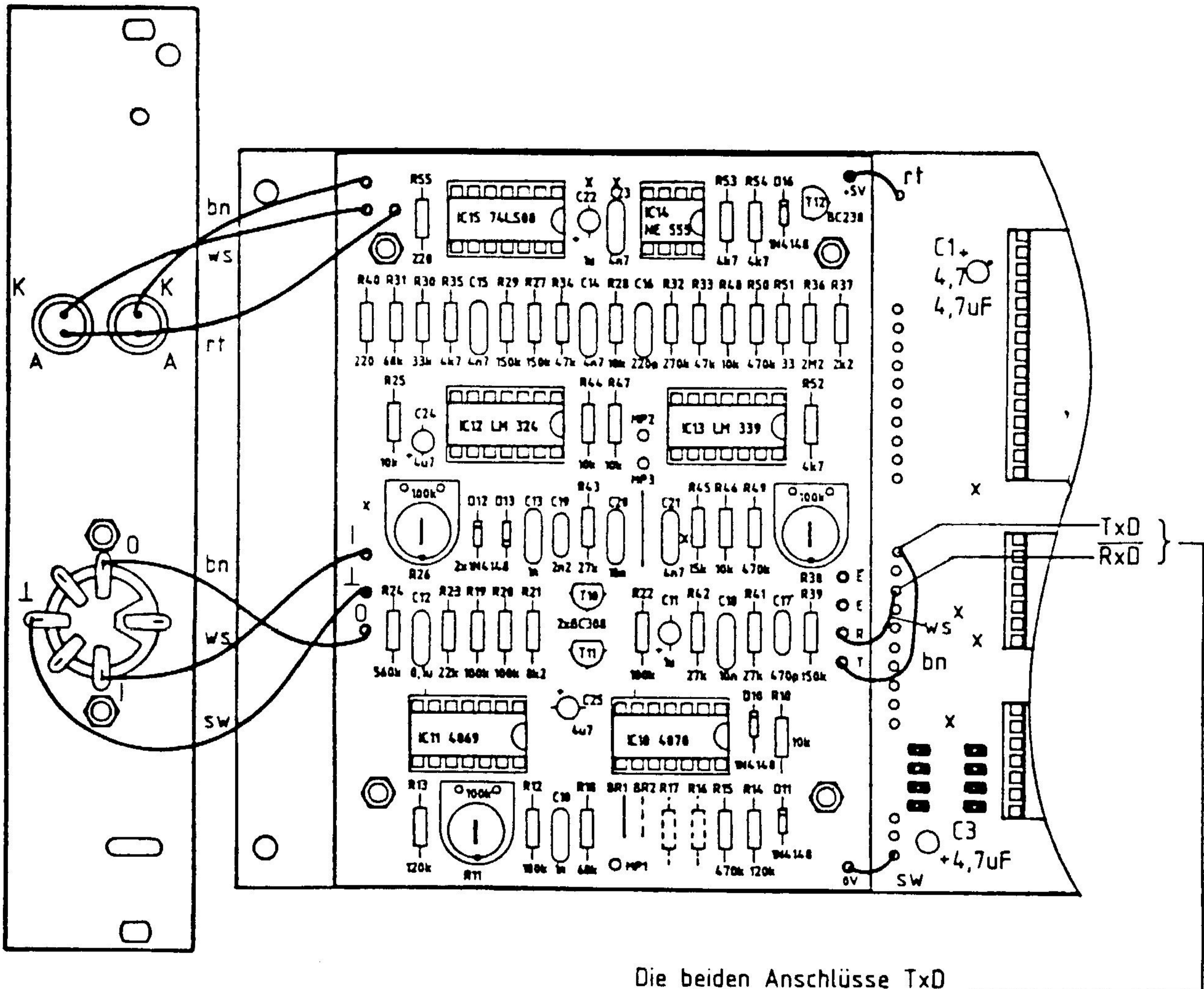




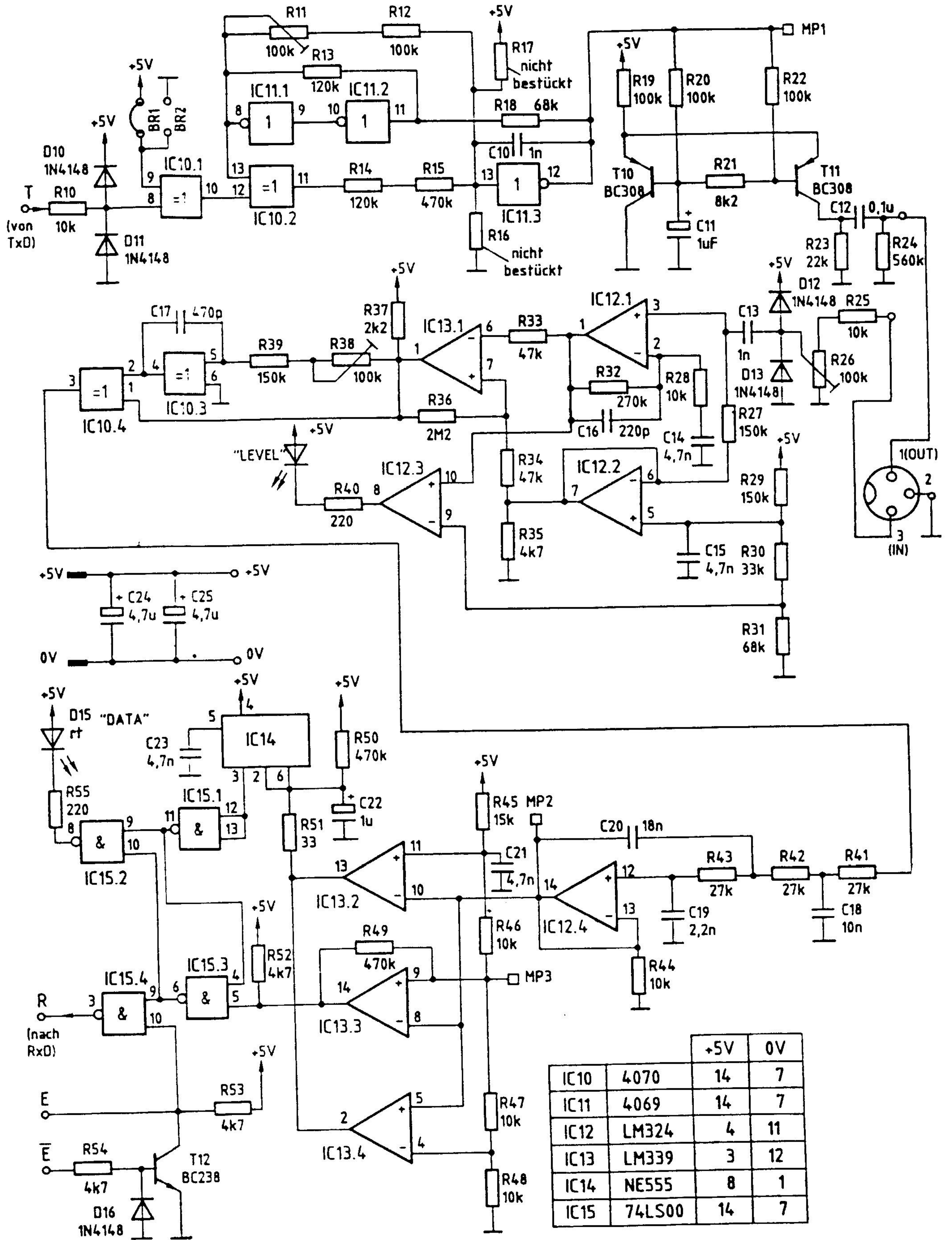
	IC1 ... IC3	IC4, IC5	IC6
	7407	74LS641	74LS00
+U <sub>b</sub>	14	20	14
0V	7	10	7







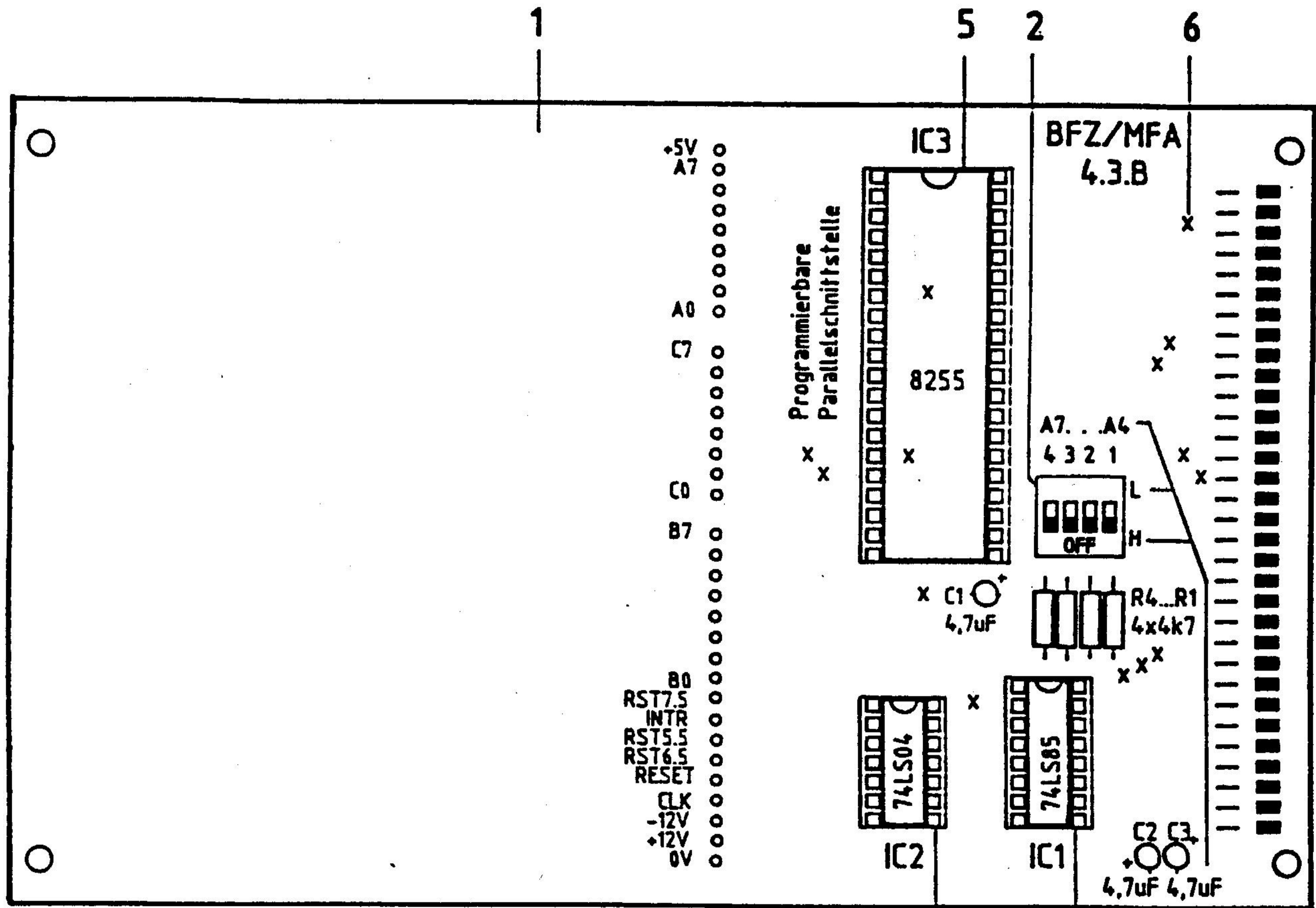
Die beiden Anschlüsse TxD  
und RxD noch nicht einlöten.  
Dies geschieht erst bei der  
Inbetriebnahme.



Stift- Nr.	Reihe		Bemerkung
	a	c	
1	+5V	+5V	
2	CLKOUT	S1-	
3	ALE	S0	
4	D0	D1	
5	D2	D3	
6	D4	D5	
7	D6	D7	
8	HLD $\bar{A}$	INT $\bar{A}$	
9	IOW	MEMW	
10	IOR	MEMR	
11	RCM	IN	
12	RAM	OUT	
13	TRAP	RESCUT	
14			
15		A 16	
16	A17	A 0	
17	A 1	A 2	
18	A 3	A 4	
19	A 5	A 6	
20	A 7	A 8	
21	A 9	A10	
22	A11	A12	
23	A13	A14	
24	A15		
25	RST7,5	INTR	
26	READY	RST5,5	
27	RESIN	RST6,5	
28	HOLD		
29	PULS OUT	Rx	
30	PULS F.INT.	Tx	
31	+12V	-12V	
32	GND	GND	

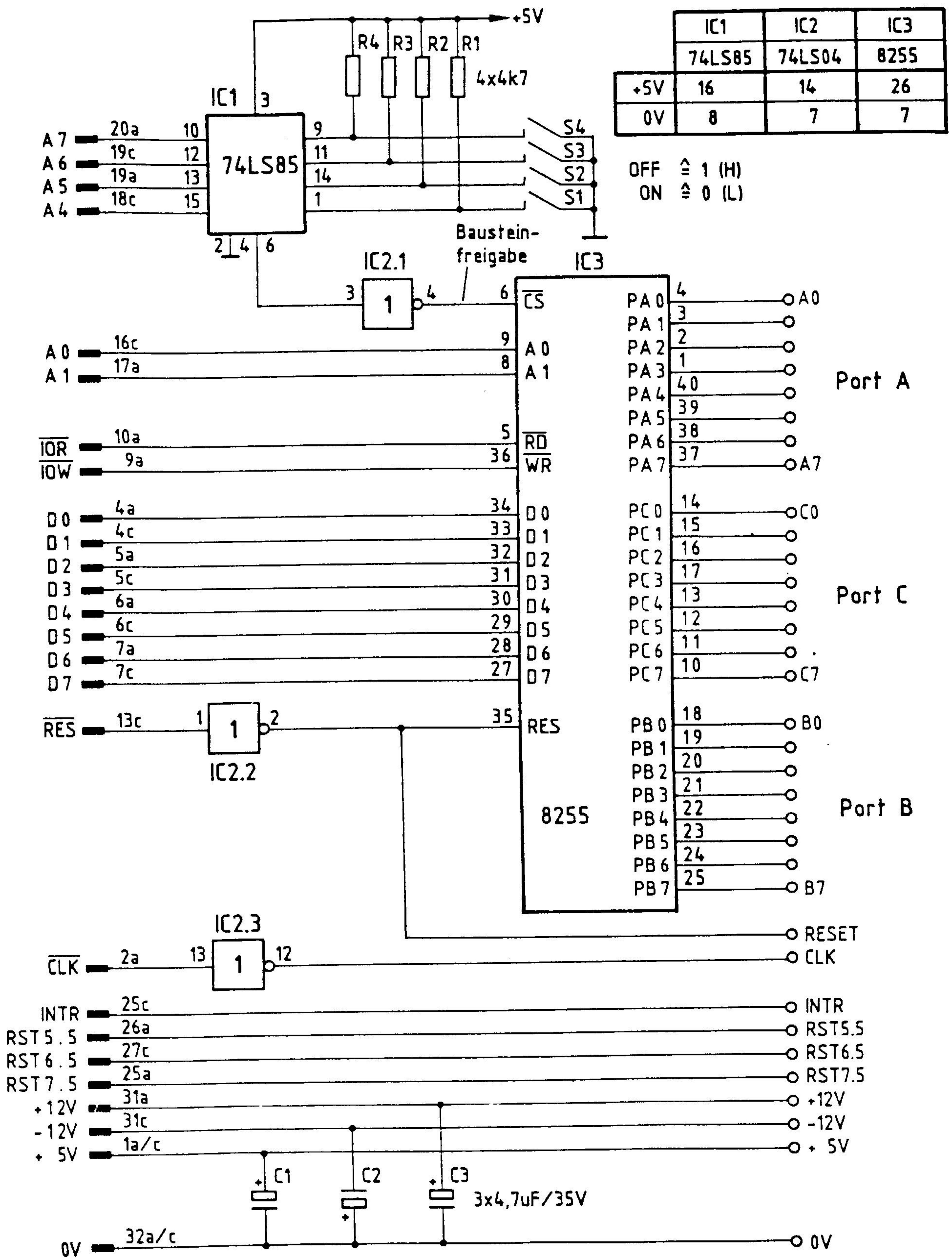


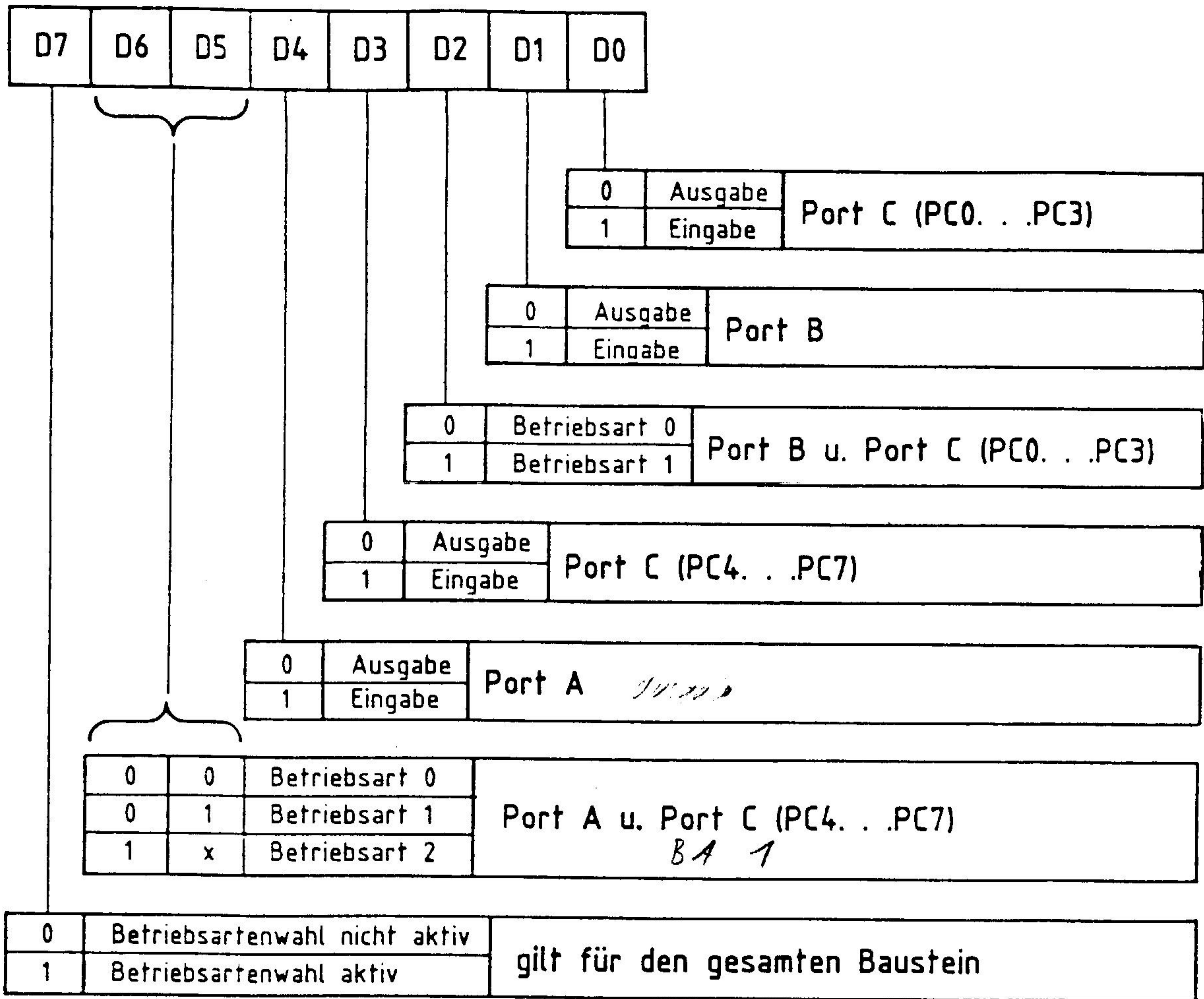
# Bestückungsplan



3 4 Beschriften Sie die Karte mit einem wasserfesten Stift



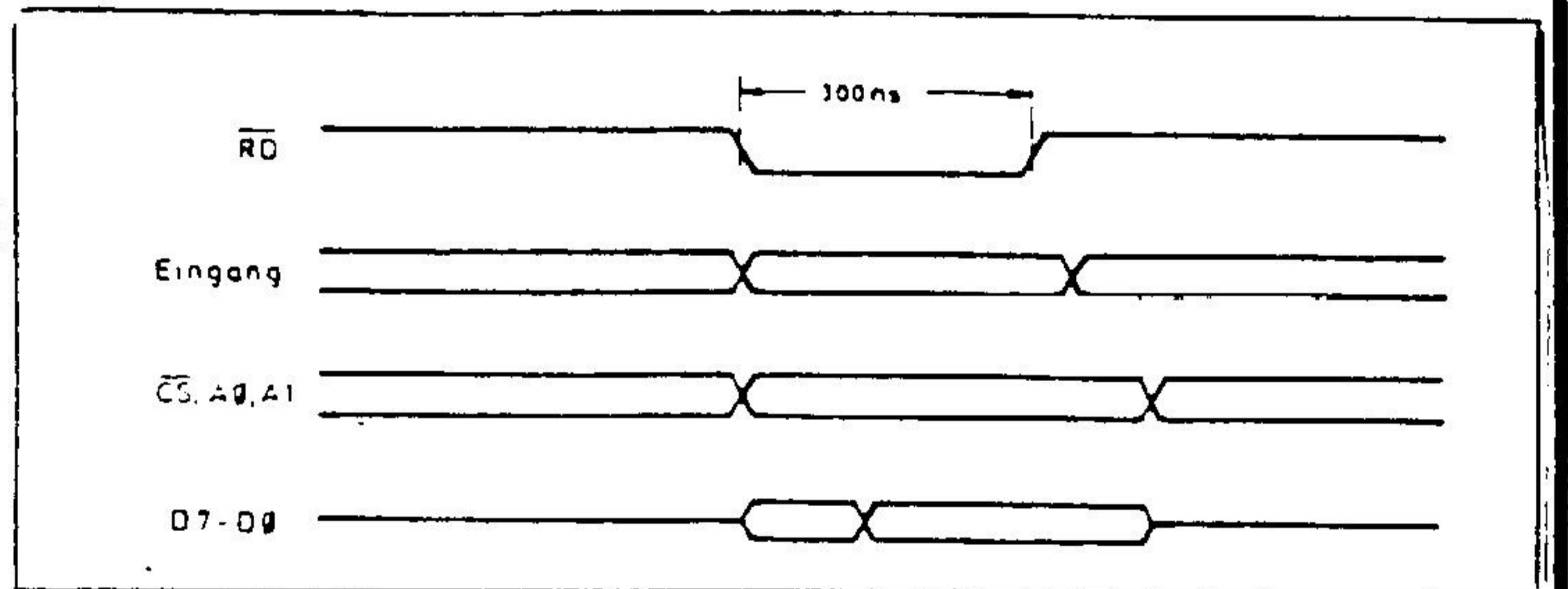




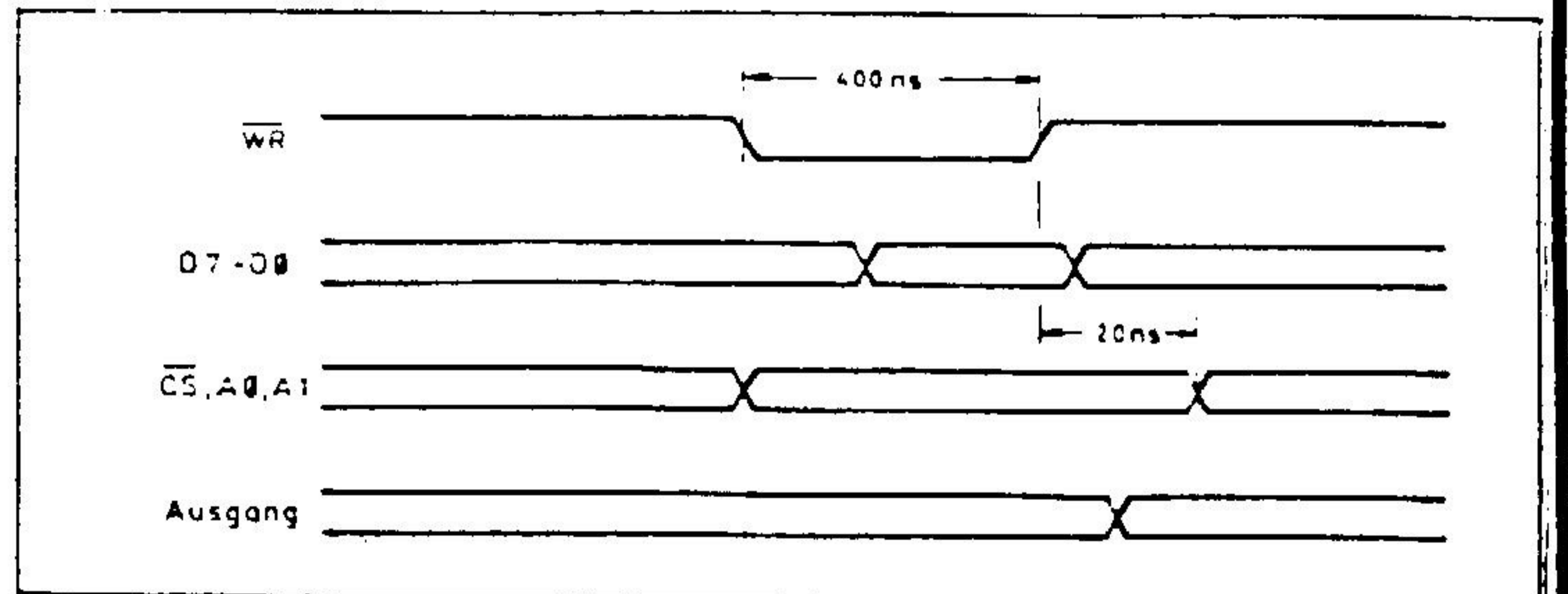
Steuerworte in der Betriebsart 0:

Steuerwort	Kanal A	Kanal C C7—C4	Kanal B	Kanal C C3—C0
80h	Ausg.	Ausg.	Ausg.	Ausg.
81h	Ausg.	Ausg.	Ausg.	Eing.
82h	Ausg.	Ausg.	Eing.	Ausg.
83h	Ausg.	Ausg.	Eing.	Eing.
88h	Ausg.	Eing.	Ausg.	Ausg.
89h	Ausg.	Eing.	Ausg.	Eing.
8Ah	Ausg.	Eing.	Eing.	Ausg.
8Bh	Ausg.	Eing.	Eing.	Eing.
90h	Eing.	Ausg.	Ausg.	Ausg.
91h	Eing.	Ausg.	Ausg.	Eing.
92h	Eing.	Ausg.	Eing.	Ausg.
93h	Eing.	Ausg.	Eing.	Eing.
98h	Eing.	Eing.	Ausg.	Ausg.
99h	Eing.	Eing.	Ausg.	Eing.
9Ah	Eing.	Eing.	Eing.	Ausg.
9Bh	Eing.	Eing.	Eing.	Eing.

Betriebsart 0 (einfache Eingabe):



Betriebsart 0 (einfache Ausgabe):





Die Bedeutung der einzelnen Bits des Steuerworts in der Betriebsart 1 bei der Dateneingabe:

Die Bedeutung der einzelnen Bits des Steuerworts in der Betriebsart 1 bei der Datenausgabe:

Kanal A                      Kanal B

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	*	X	X	X

D7	D6	D5	D4	D3	D2	D1	D0
1	X	X	X	X	1	1	X

\* C7, C6  
1 ≙ Eingang  
0 ≙ Ausgang über Bit setzen/rücksetzen

Kanal A                      Kanal B

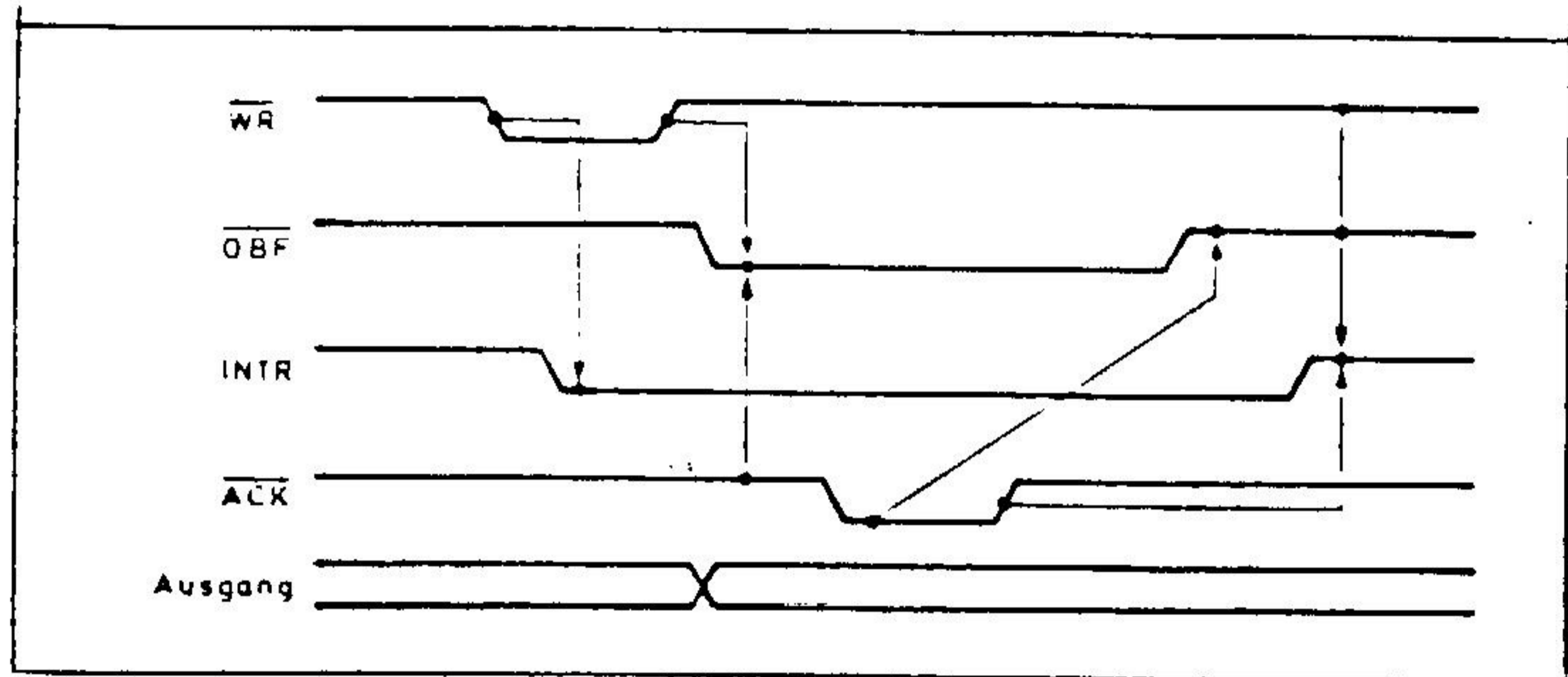
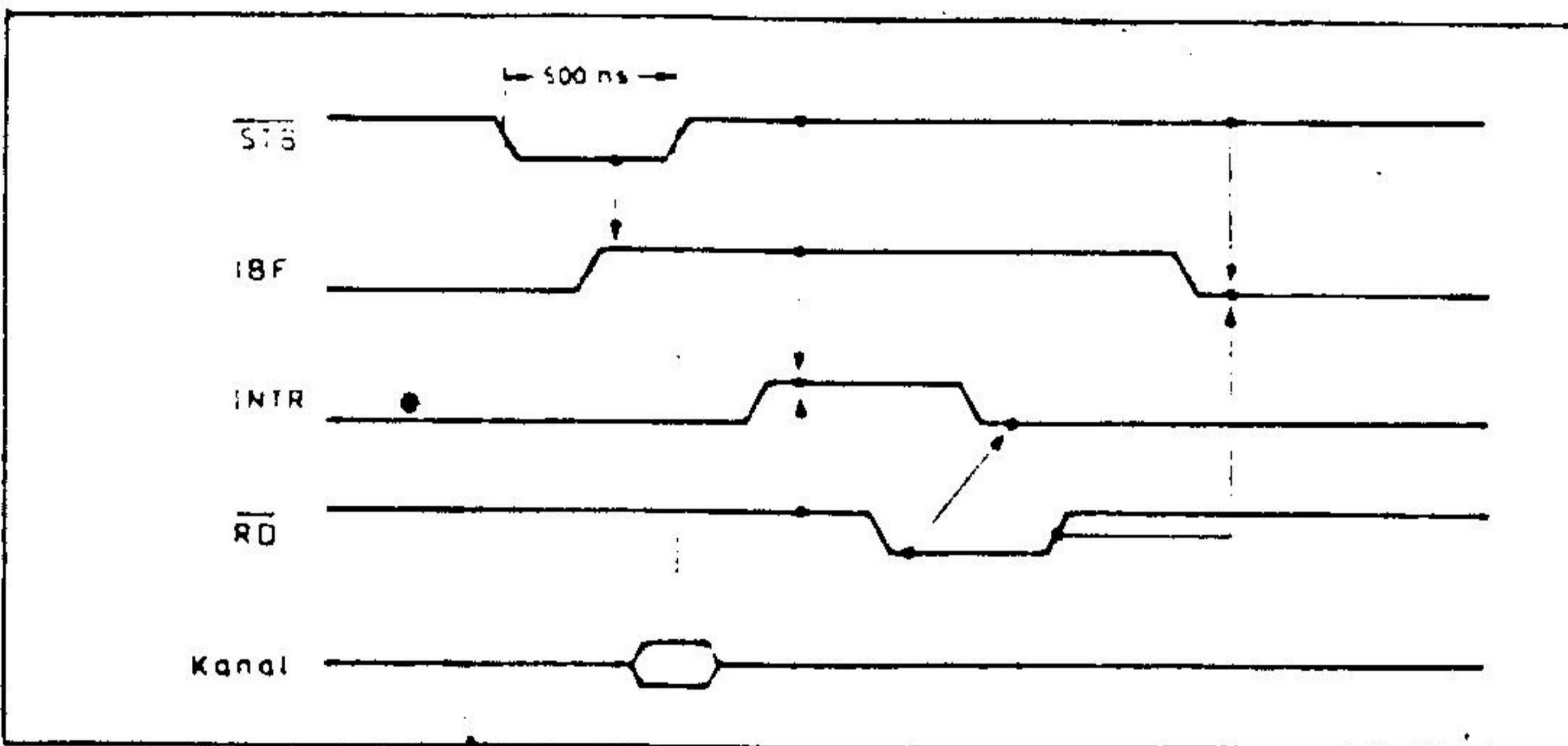
D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	*	X	X	X

D7	D6	D5	D4	D3	D2	D1	D0
1	X	X	X	X	1	0	X

\* C5, C4  
1 ≙ Eingang  
0 ≙ Ausgang über Bit setzen/rücksetzen

Betriebsart 1 (getastete Eingabe):

Betriebsart 1 (getastete Ausgabe):



Bedeutung des Zustandswortes, das in der Betriebsart 1 von Port C gelesen werden kann:

Port A und Port B als Eingang:

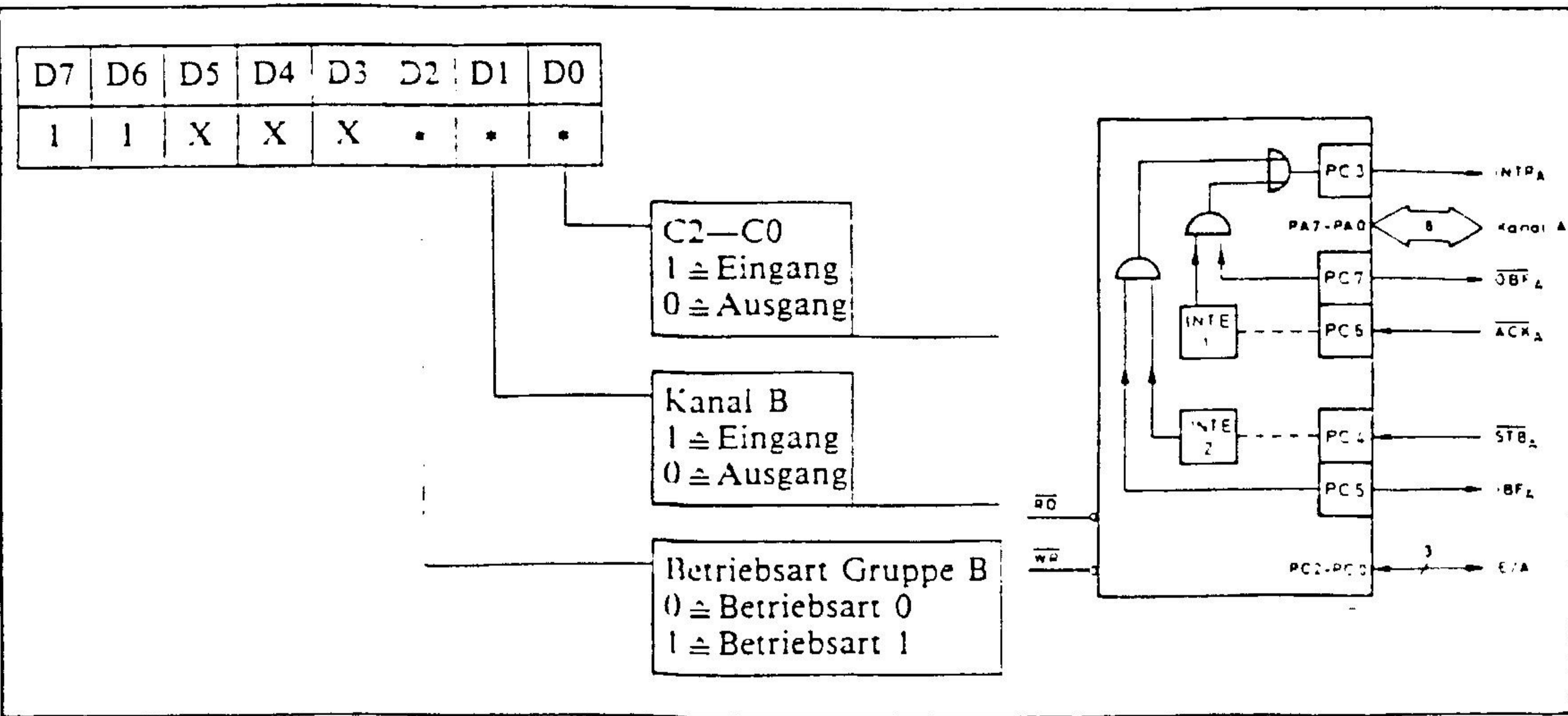
D7	D6	D5	D4	D3	D2	D1	D0
'E/A'	'E/A'	IBFA	INTEA	INTRA	INTEB	IBFB	INTRB

Port A und Port B als Ausgang:

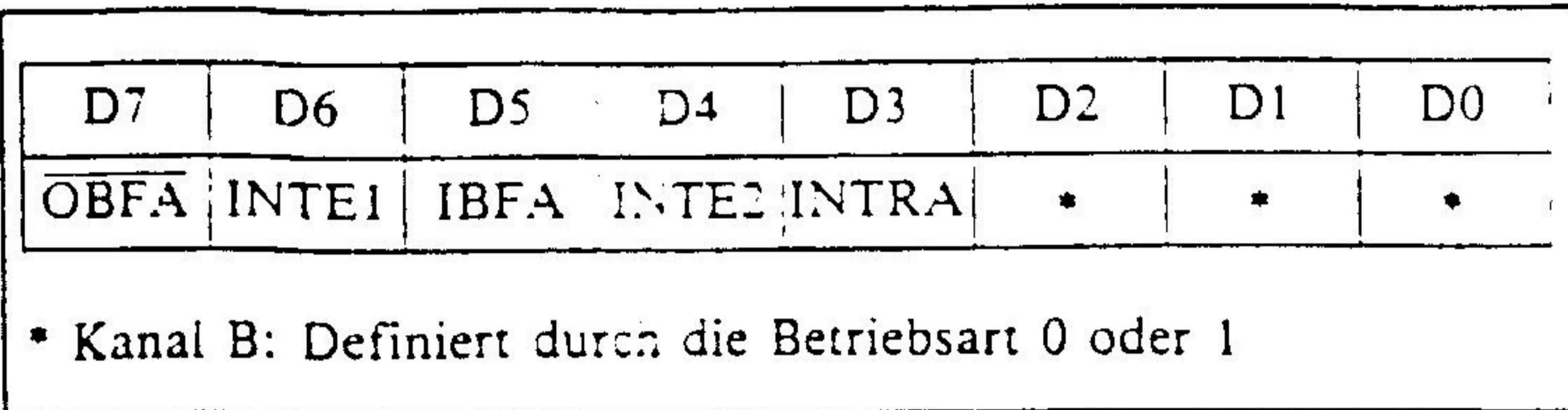
D7	D6	D5	D4	D3	D2	D1	D0
OBF A	INTEA	E/A	E/A	INTRA	INTEB	OBF B	INTRB



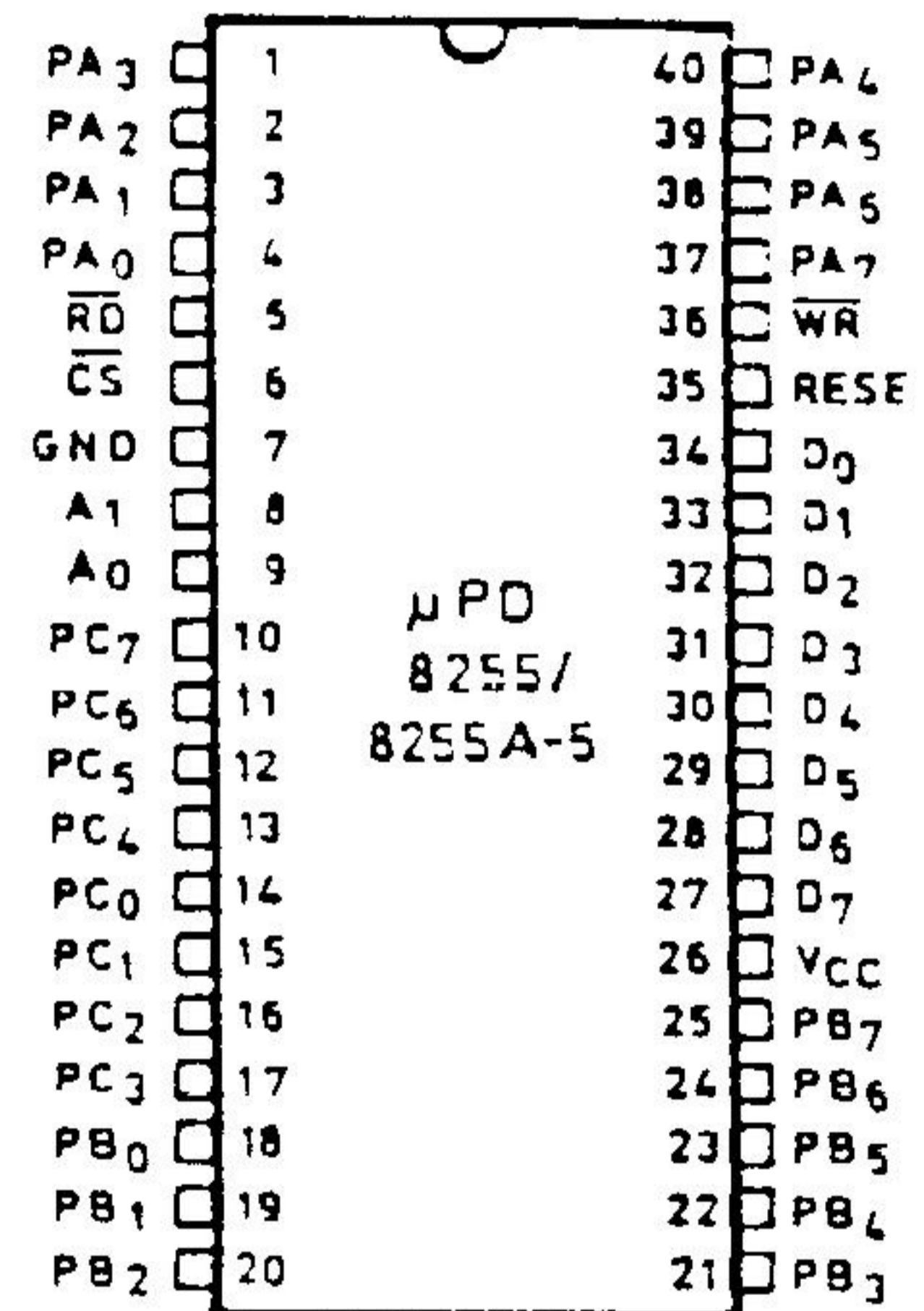
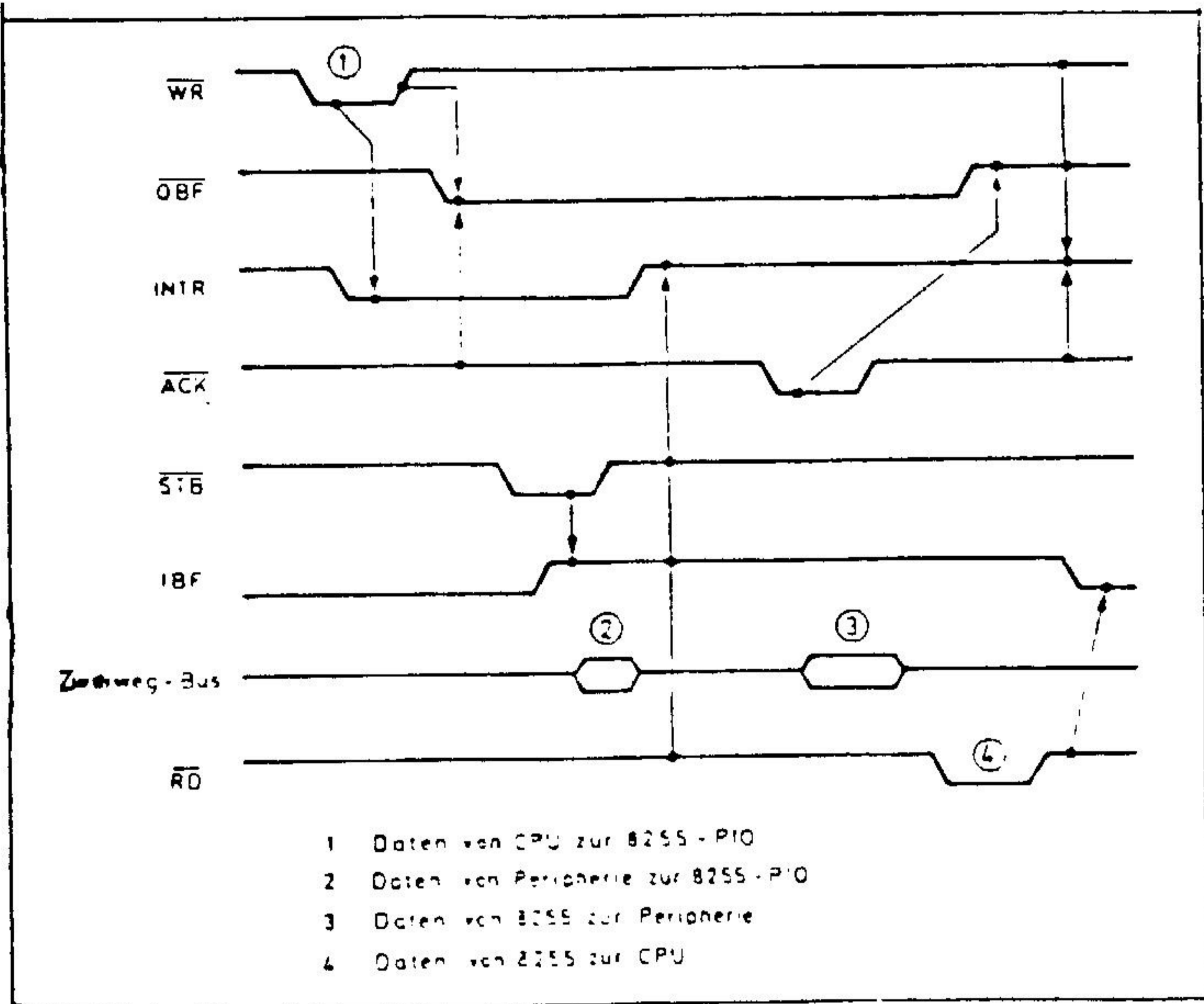
Steuerwort für die Betriebsart 2:

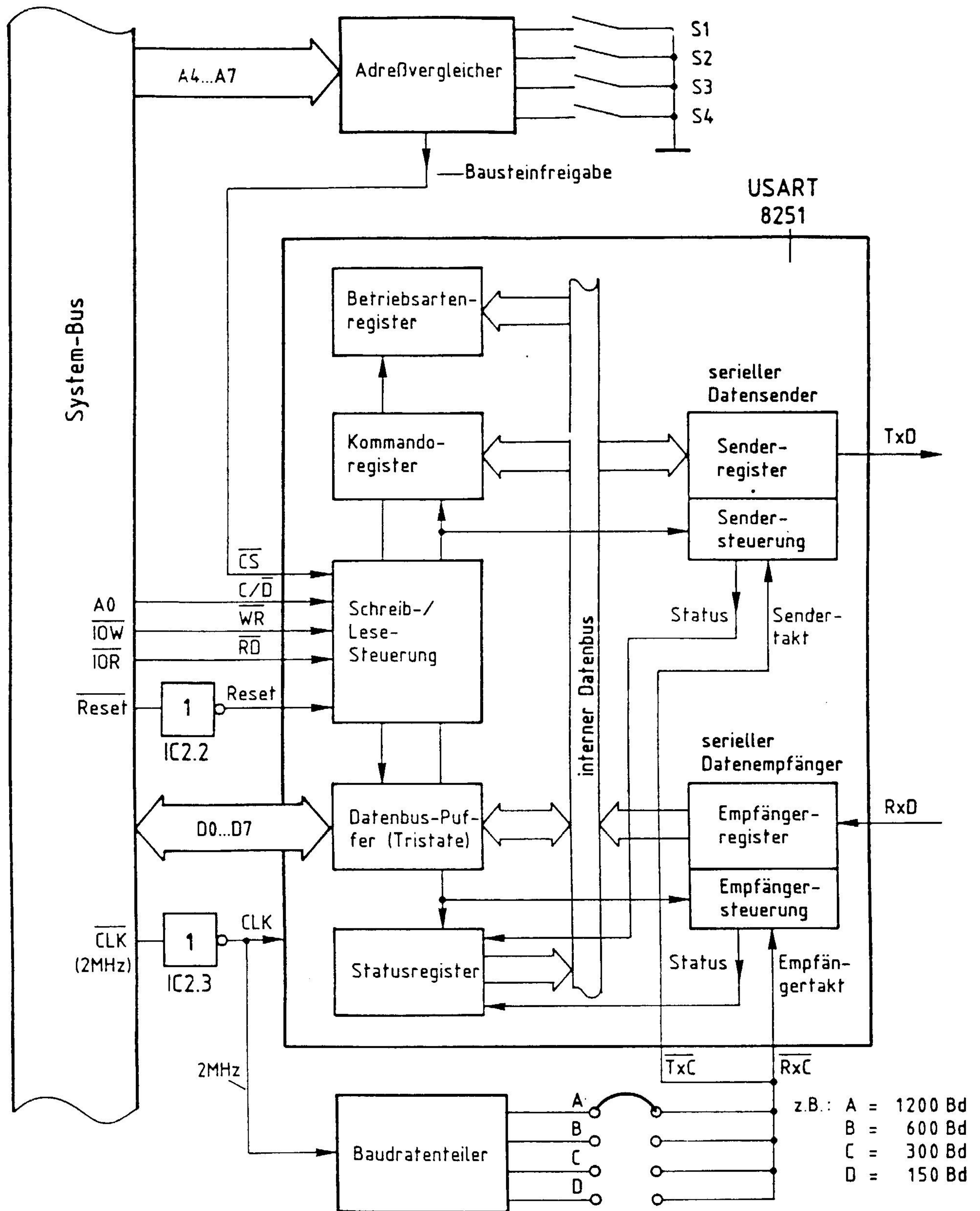


Zustandswort in der Betriebsart 2:

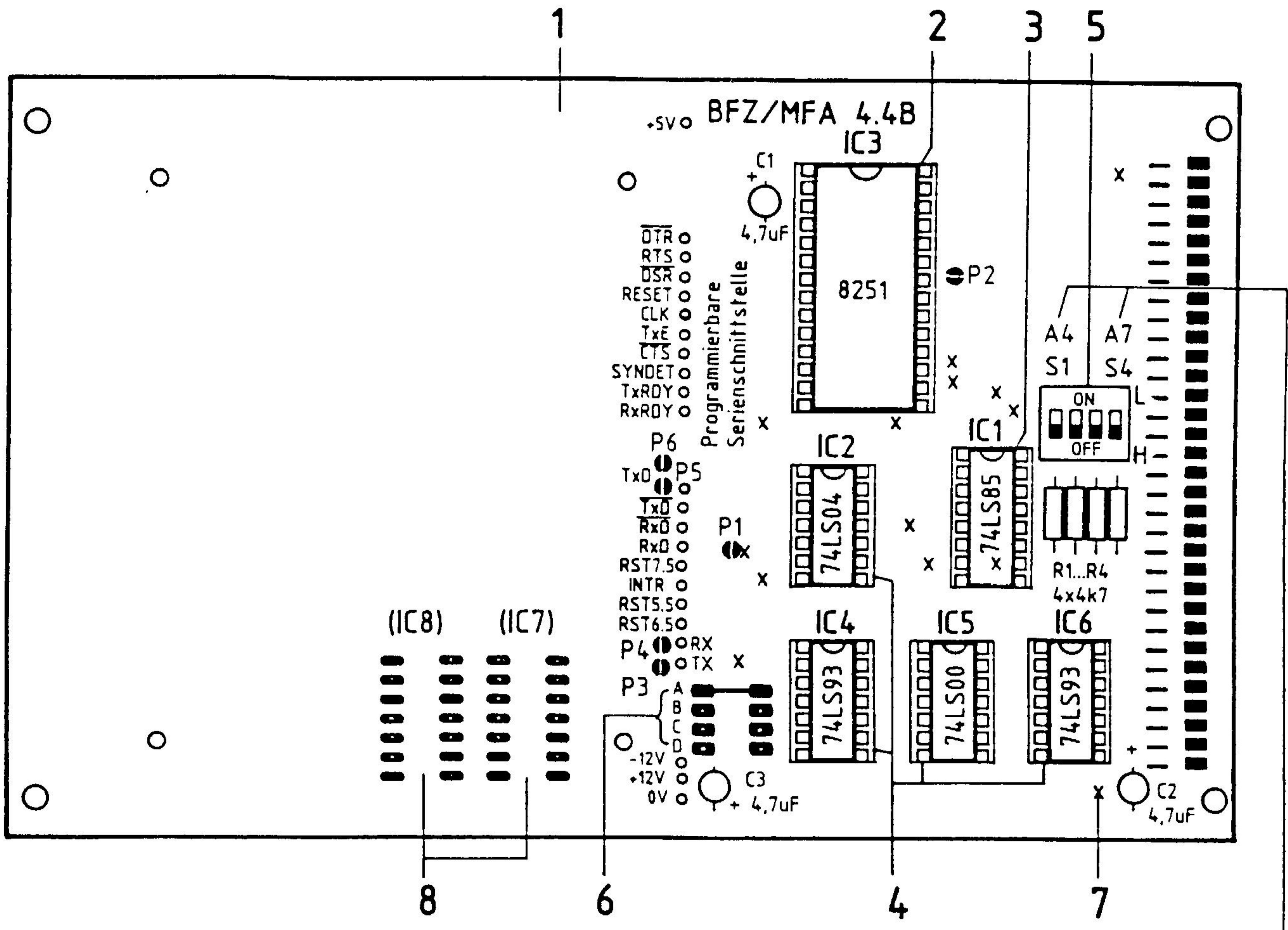


Betriebsart 2 (Zweiweg-Bus)



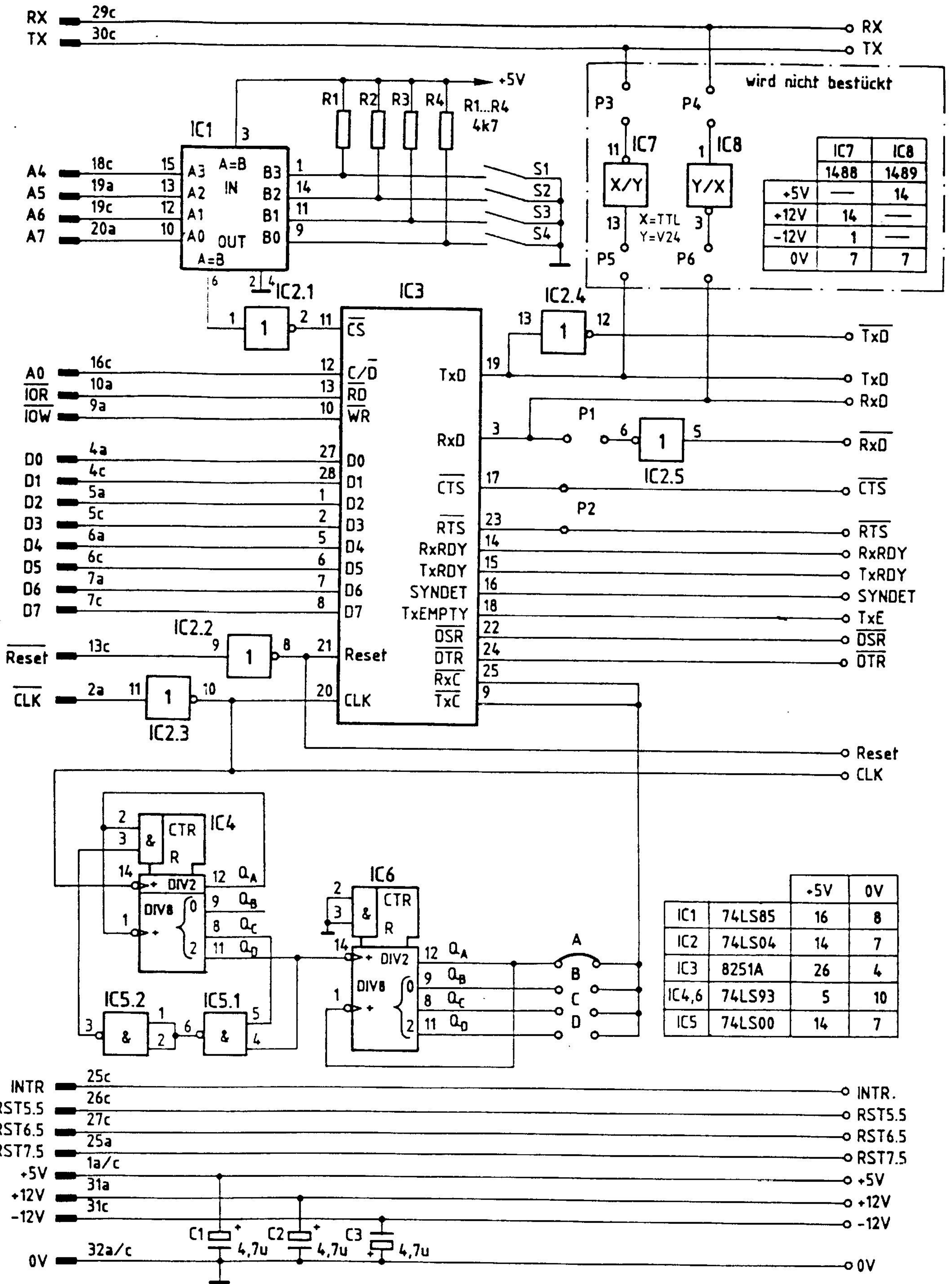


# Bestückungsplan



Beschriften Sie die Karte mit einem wasserfesten Stift





D7	D6	D5	D4	D3	D2	D1	D0
S2	S1	EP	PEN	L2	L1	B2	B1

		Betriebsart	interner Teilungsfaktor
0	0	Synchronbetrieb	1
0	1	Asynchronbetrieb	1 <i>9600</i>
1	0	Asynchronbetrieb	16 <i>4800</i>
1	1	Asynchronbetrieb	64 <i>1200</i>

Zeichenzlänge		
0	0	5 Bit
0	1	6 Bit
1	0	7 Bit
1	1	8 Bit

0	ohne Paritätsbit
1	mit Paritätsbit

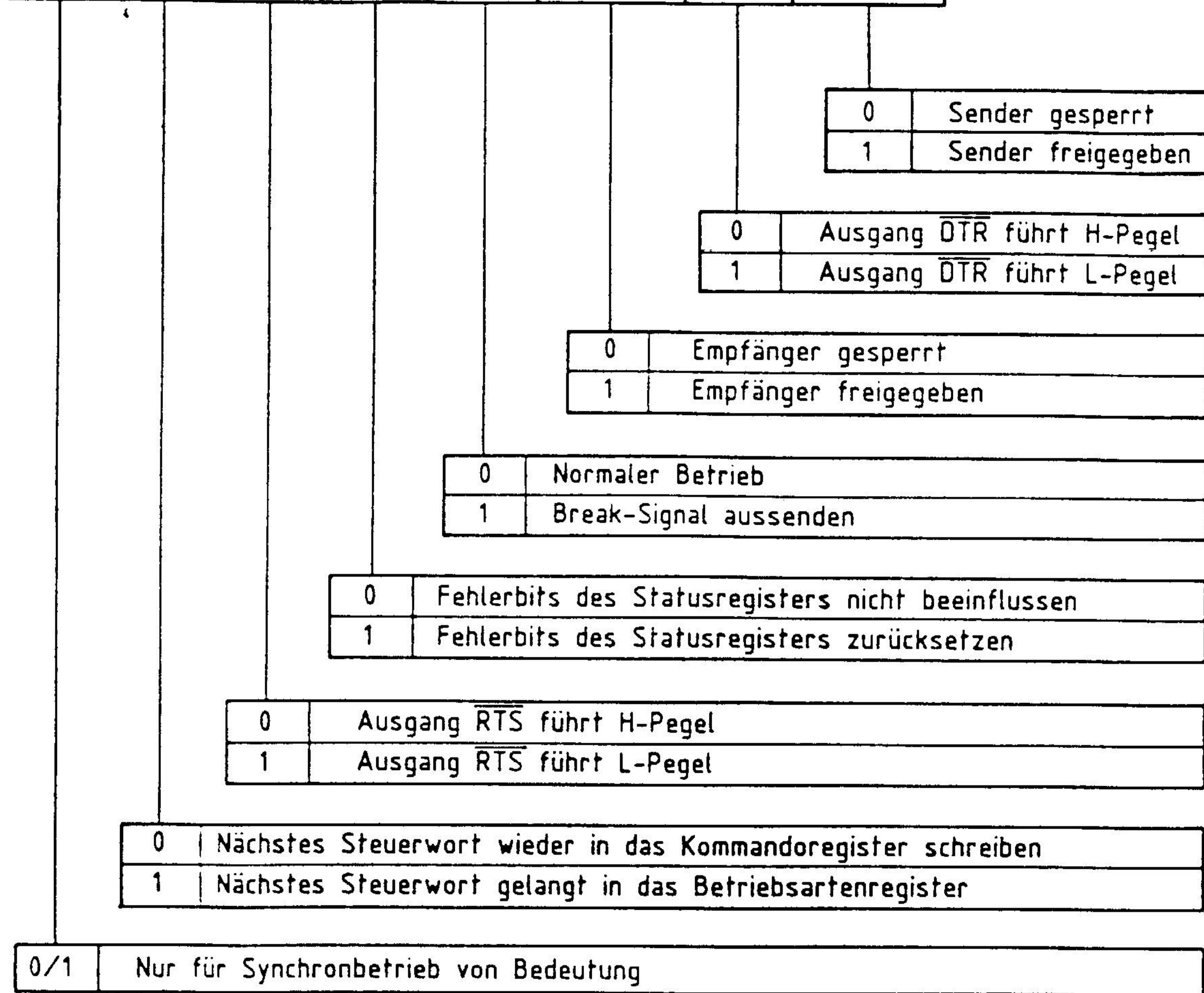
0	ungerade Parität
1	gerade Parität

Anzahl der Stopbits		
0	0	nicht zulässig
0	1	1 Stopbit
1	0	1½ Stopbits
1	1	2 Stopbits

Das Betriebsarten-Wort



D7	D6	D5	D4	D3	D2	D1	D0
EH	IR	RTS	ER	SBRK	RxENABLE	DTR	TxENABLE



Das Kommando-Wort



D7	D6	D5	D4	D3	D2	D1	D0
DSR	SYNDET/BD	FE	OE	PE	TxEMPTY	RxRDY	TxRDY

0	Senderregister besetzt
1	Senderregister frei

0	Empfängerregister hat kein Zeichen
1	Empfängerregister hat Zeichen

0	Datensender gibt noch Zeichen aus
1	Datensender hat keine Zeichen mehr

0	Kein Paritätsfehler beim Datenempfang
1	Paritätsfehler liegt vor

0	Kein Überlauffehler beim Datenempfang
1	Überlauffehler liegt vor

0	Kein Stopbitfehler beim Datenempfang
1	Stopbitfehler liegt vor

0	} hat im Synchron- und Asynchronbetrieb unterschiedliche Bedeutungen, wird nicht benötigt
1	

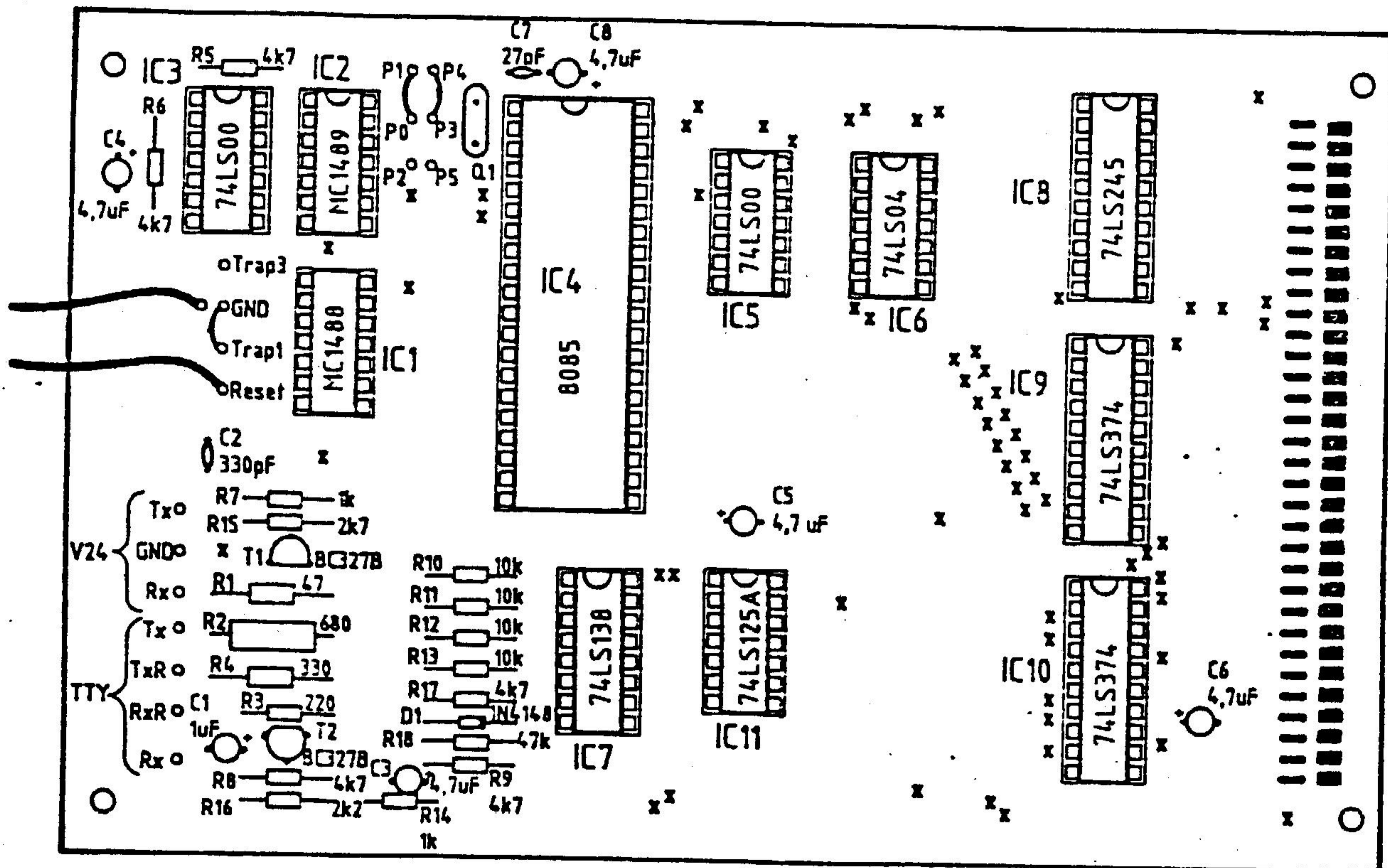
0	Eingang $\overline{DSR}$ führt H-Pegel
1	Eingang $\overline{DSR}$ führt L-Pegel

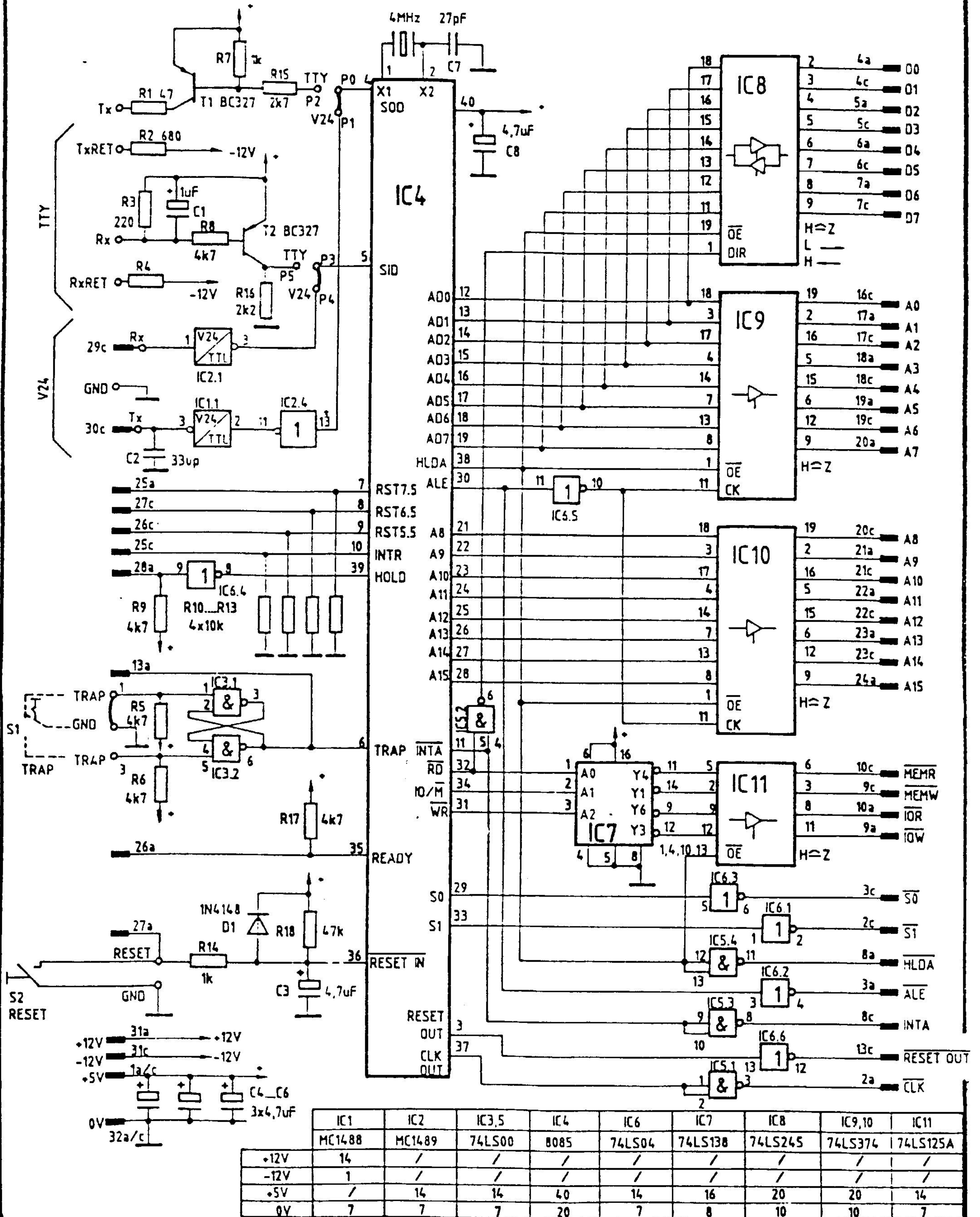
### Das Statusregister





# Bestückungsplan



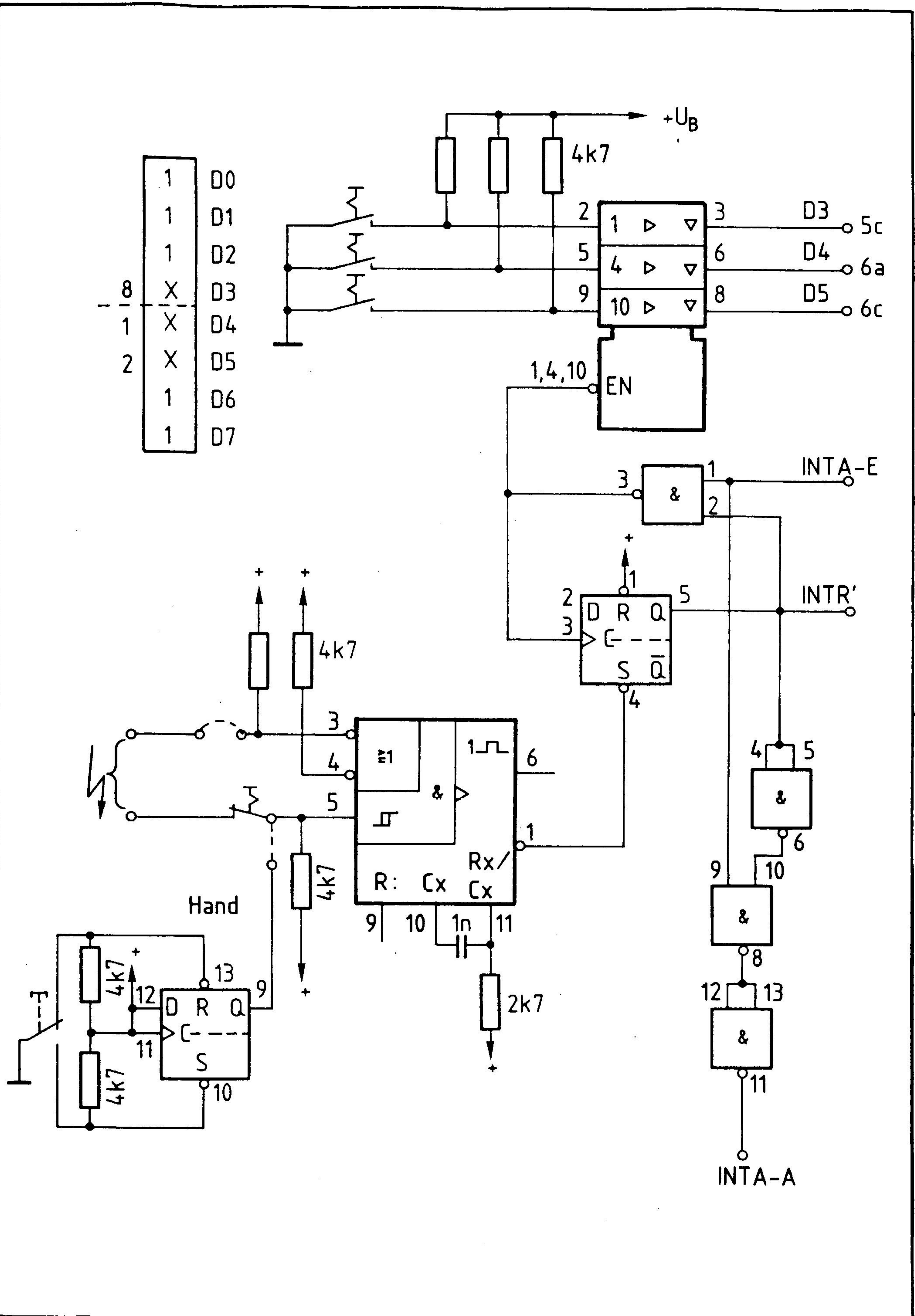


Teil 5.2 Schaltungen und Bestückungspläne (Erweiterungen)

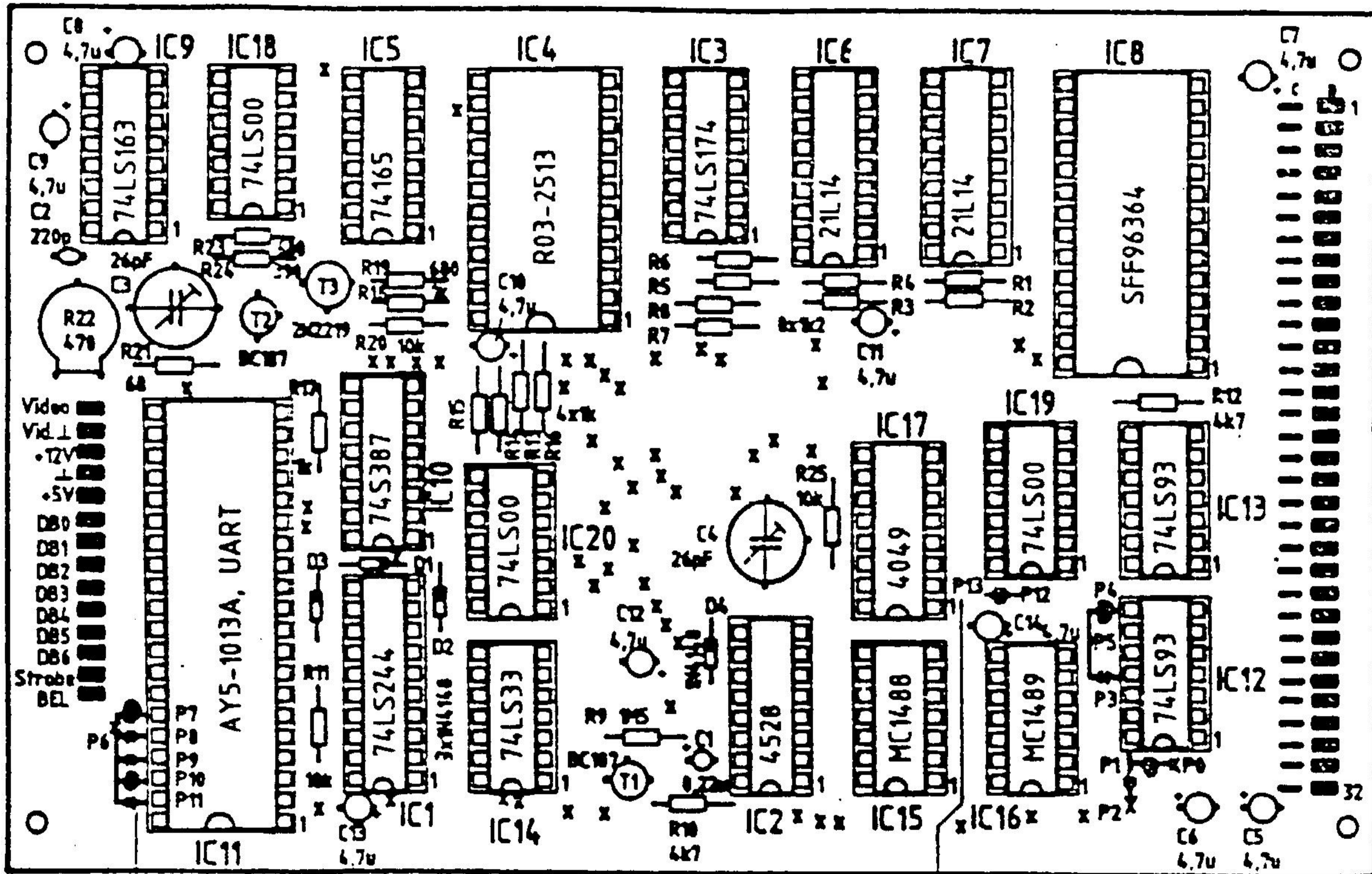
-Programmierbare Parallel-Schnittstelle	S.64
-Eprom-Programmierer	S.66
-Drucker-Interface	S.68
-Zeitwerk 4 fach	S.70
-Initialisierung und Diagramme des Parallel- schnittstellen-Bausteins 8255	S.72
-Blockschaltbild Programmierbare Serienschnittstelle	S.75
-Programmierbare Serienschnittstelle	S.76
-Initialisierung des Serienschnittstellen- Bausteins 8251	S.78
-Kassetten-Interface	S.82
-Analoge Ein-/Ausgabe (2 kanalig)	S.84
-Programmierbarer Zähler und Zeitgeber	S.86
-Steuerwort des Zähler-Bausteins 8253	S.88
-Fehlersimulation	S.90
-Demonstrationsmodell	S.92
-Vektor-Interrupt-Karte	S.94
-Interruptsteuerung 8085	S.95
-Pin-Belegung der Busleiste	S.96



Teil 5.1 Schaltungen und Bestückungspläne (Grundgerät)

-Prozessor 8085	S.44
- 8-K-Ram/Eprom	S.46
- 16-K-Ram/Eprom	S.48
- 8-Bit-Parallel-Ausgabe	S.50
- 8-Bit-Parallel-Eingabe	S.52
-Bus-Signalgeber	S.54
-Bus-Signalanzeige	S.56
-Video-Interface	S.58
-Blockschaltbild Video-Interface	S.60



# Bestückungsplan Leiterplatte

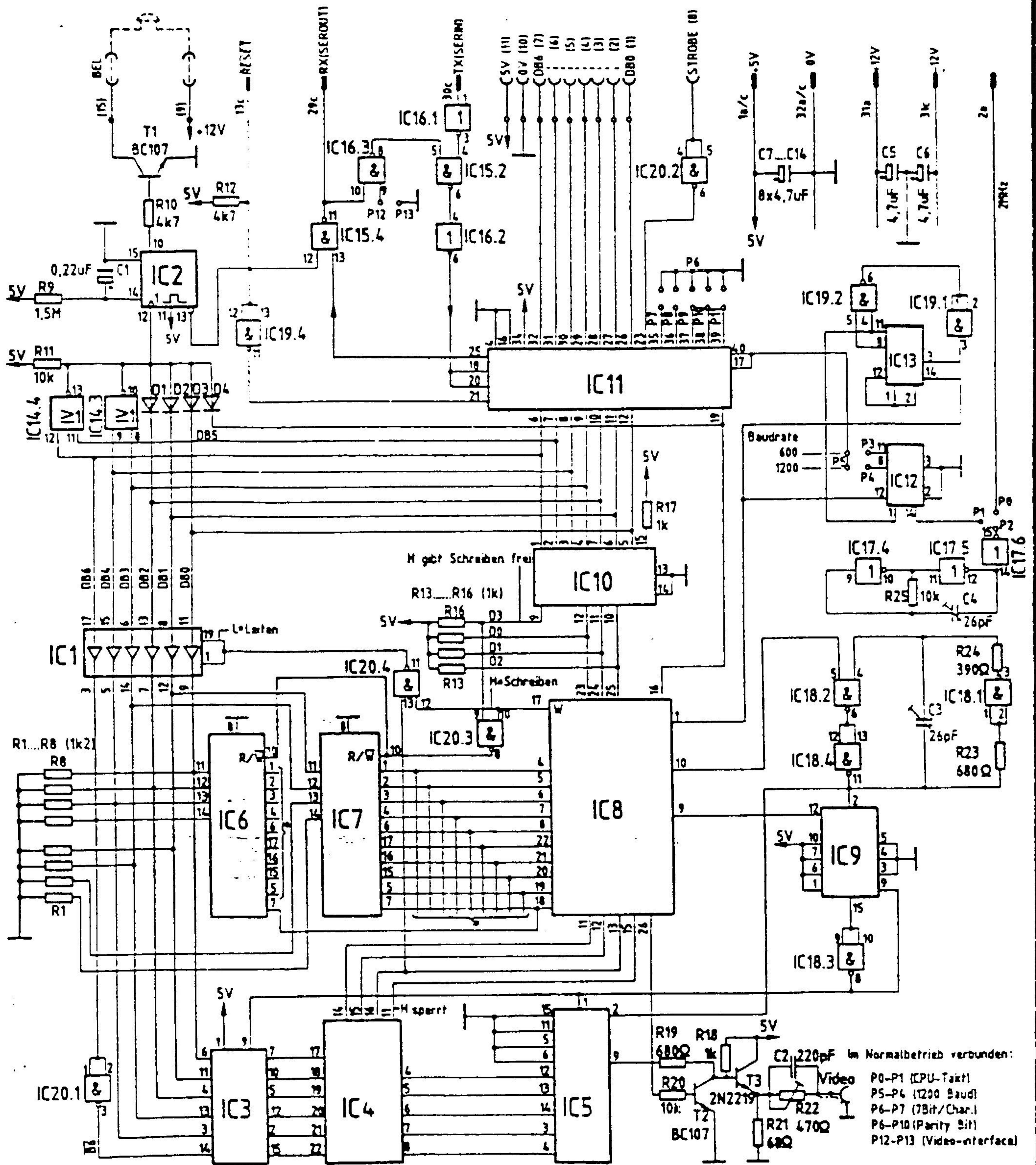


Lötbrücken P.:  = Brücke offen  
 = Brücke geschlossen

Bedeutung der Lötbrücken siehe Stromlaufplan

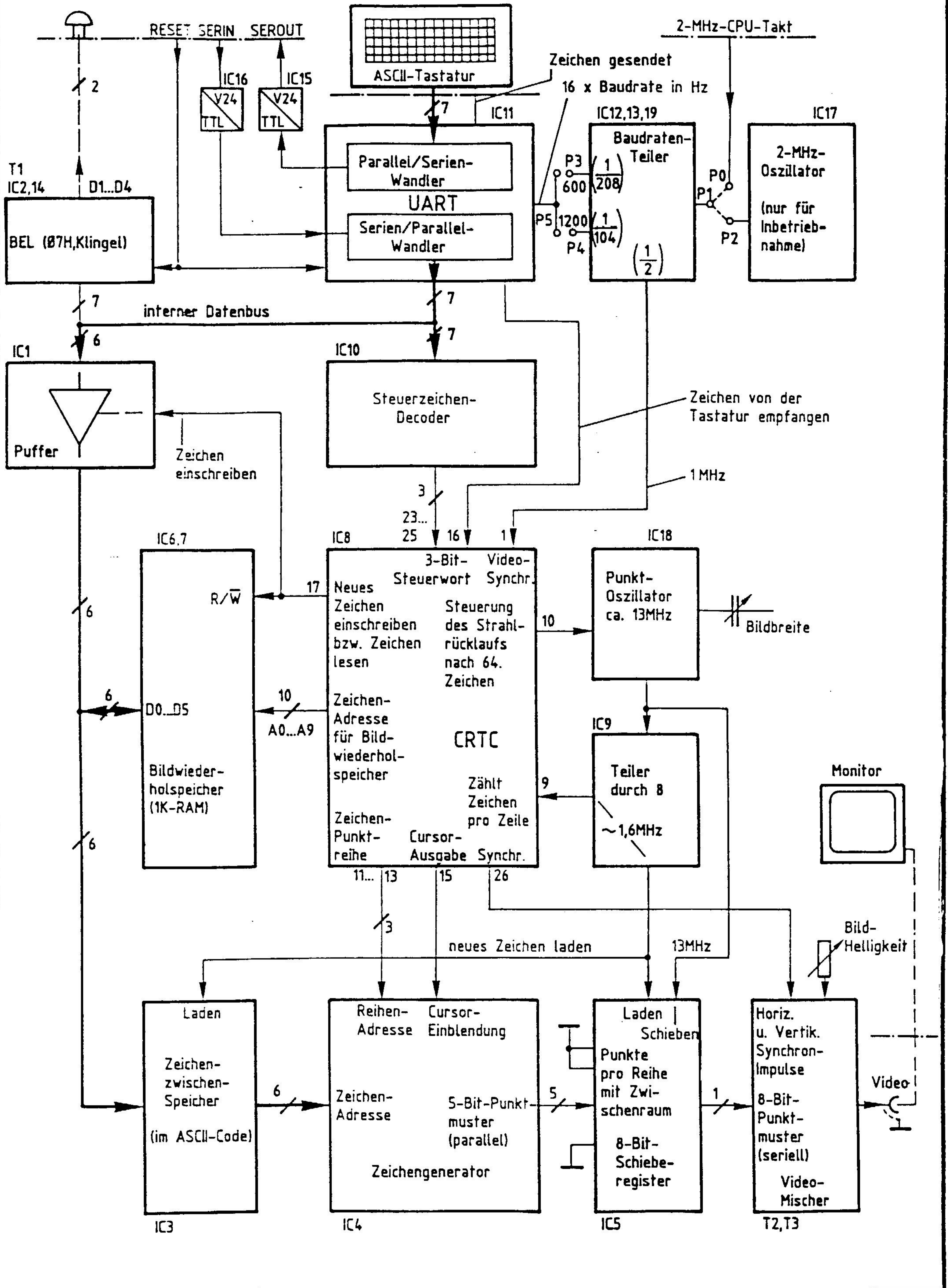
P13 und P12 befinden sich hier auf der Lötseite der Leiterplatte.  
 Die Brücke bleibt zunächst offen!





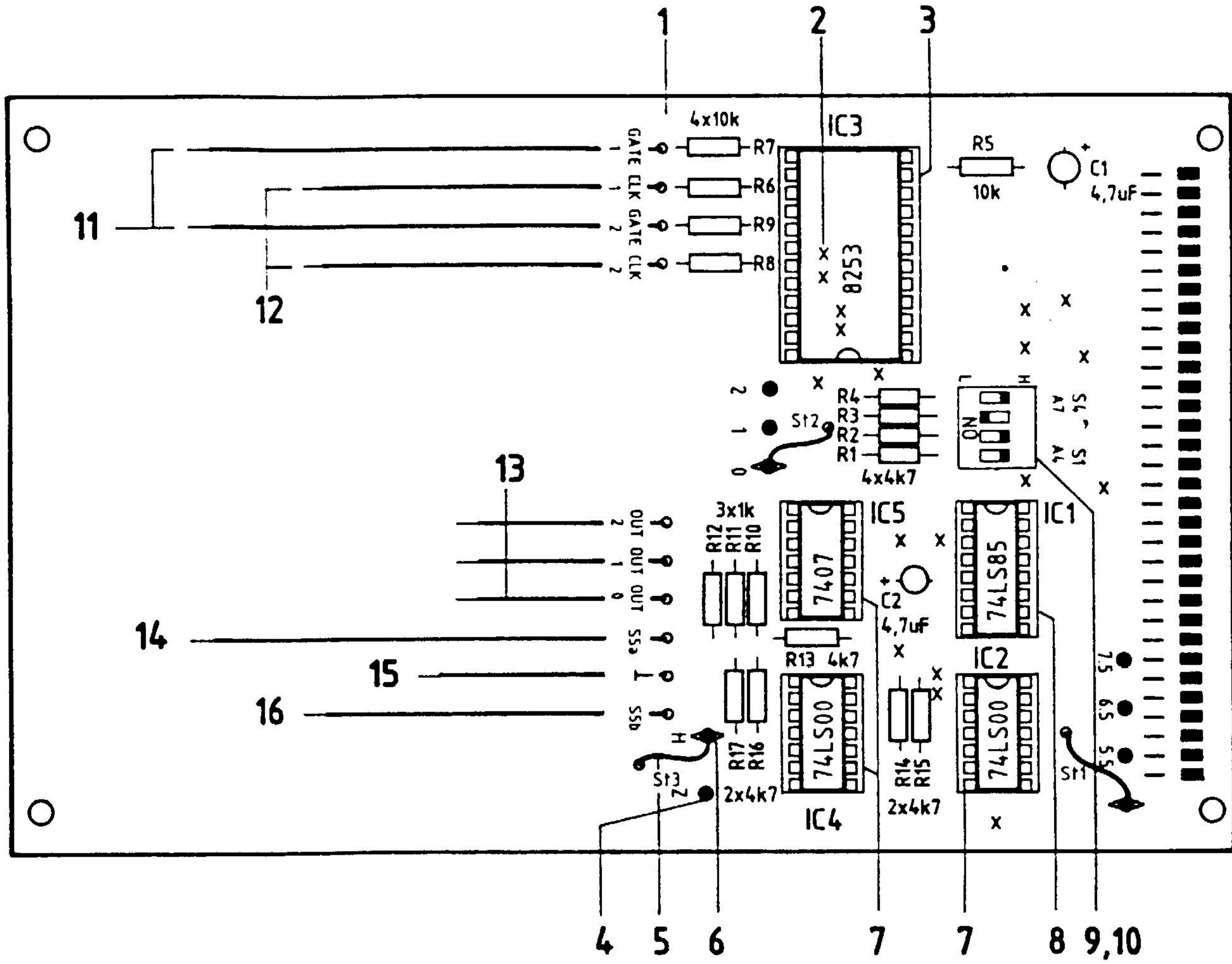
	IC1	IC2	IC3	IC4	IC5	IC6,7	IC8	IC9	IC10	IC11	IC12,13	IC14	IC15	IC16	IC17	IC18,19,20
Typ	74LS244	4528	74LS174	R03-2513	74165	2714	96364	74LS163	74LS387	AYS-1013	74LS93	74LS33	MC1488	MC1489	4069	74LS00
-5V	20	16	16	24	16	18	28	16	16	1	5	16	/	16	1	16
0V	10	8	8	10	8	9	14	8	8	3	10	7	7	7	8	7
-12V	/	/	/	/	/	/	/	/	/	/	/	/	16	/	/	/
-12V	/	/	/	/	/	/	/	/	/	2	/	/	1	/	/	/

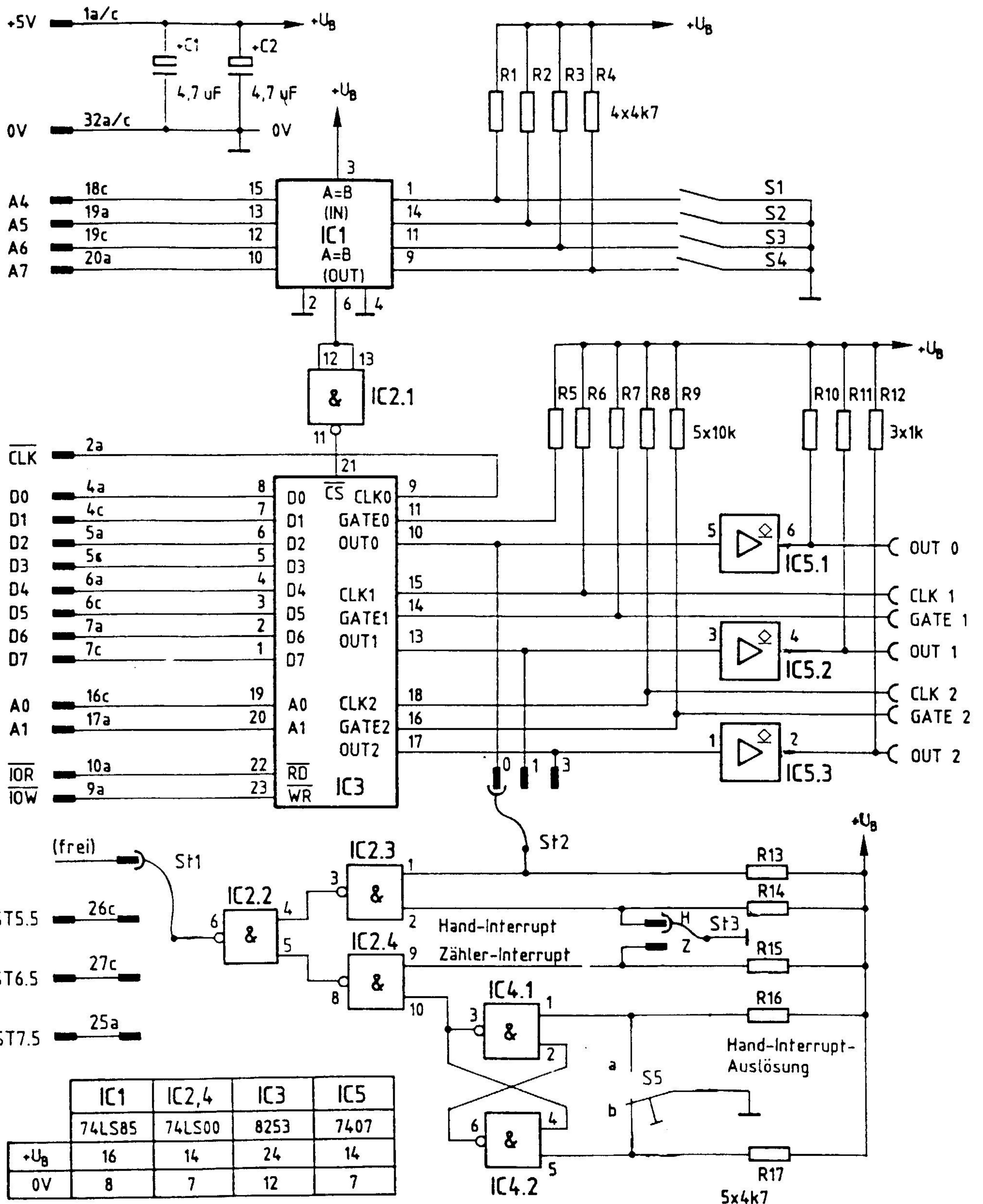






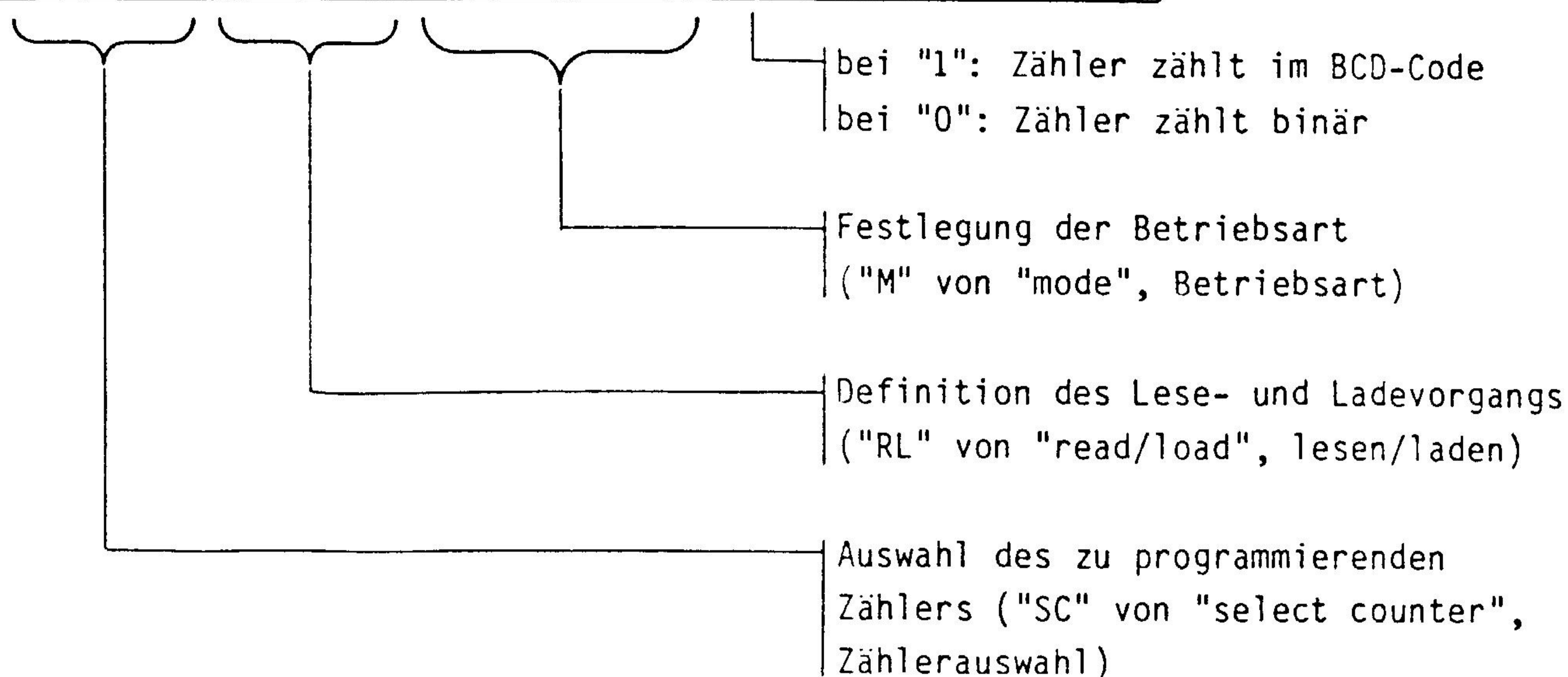
# Bestückungsplan





### Aufbau des Steuerwortes:

D7	D6	D5	D4	D3	D2	D1	D0	Datenbit
SC1	SC0	RL1	RLO	M2	M1	M0	BCD	Bit-Bezeichnung



### Festlegung der Betriebsarten:

M2	M1	M0	Nummer und Bezeichnung der Betriebsart
0	0	0	0 Unterbrechen beim Zählerstand Null
0	0	1	1 Monostabile Kippstufe, retriggerbar
X	1	0	2 Programmierbarer Frequenzteiler
X	1	1	3 Programmierbarer symmetrischer Rechteckgenerator
1	0	0	4 Verzögerter Impuls, Programmtriggerung
1	0	1	5 Verzögerter Impuls, externe Triggerung

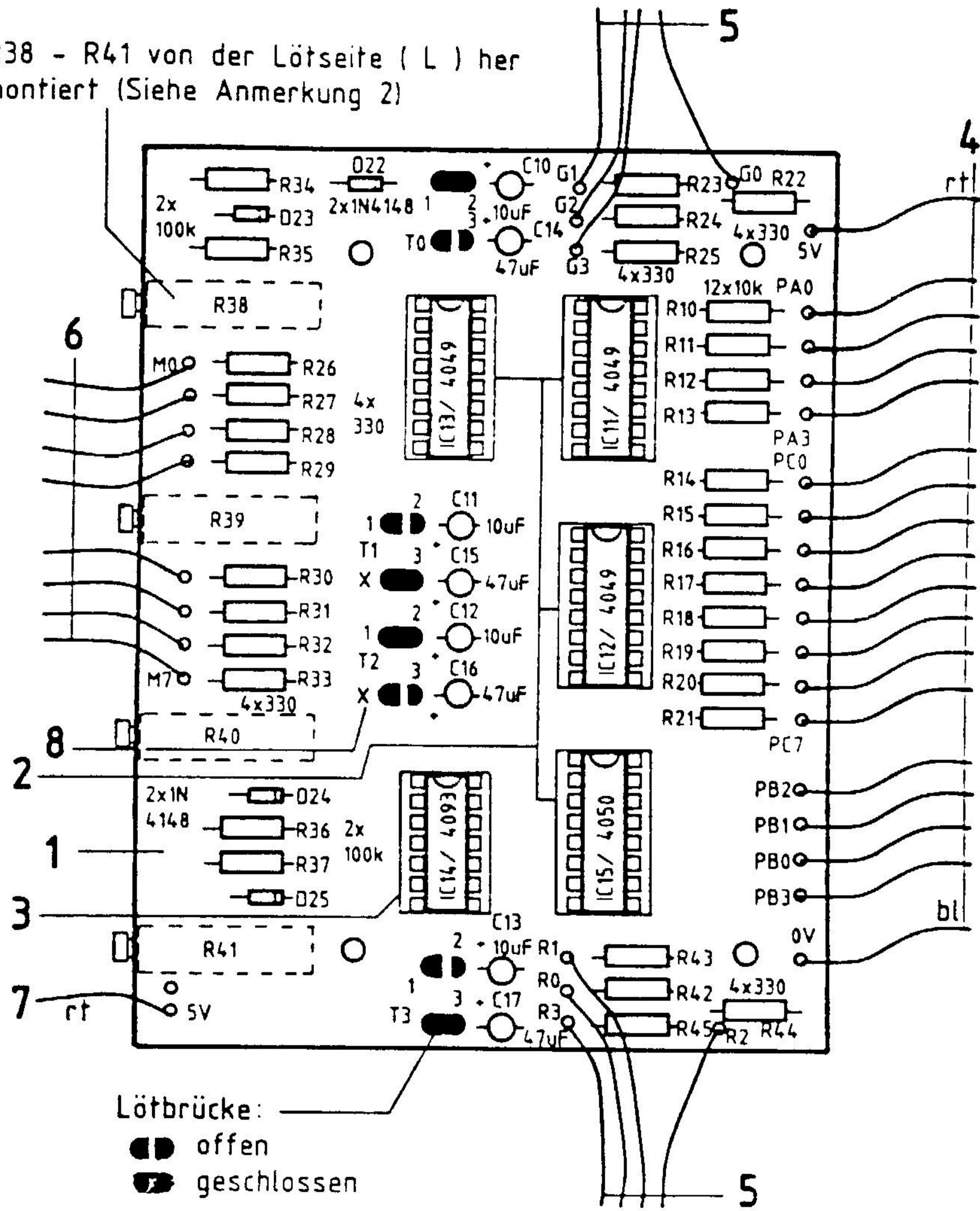
("X" bedeutet wahlweise "1" oder "0")

### Wirkung der Steuerwort-Bits RLO und RL1:

Bitkombi- nation Nr.	RL1 (D5)	RLO (D4)	Wirkung auf das Lesen und Laden des durch SC0 und SC1 ausgewählten Zählers
1	0	0	Zählerstand zum Lesen eines Zählers zwischenspeichern
2	0	1	Lesen/Laden nur des niederwertigen Bytes eines 16-Bit-Zählers
3	1	0	Lesen/Laden nur des höherwertigen Bytes eines 16-Bit-Zählers
4	1	1	Lesen/Laden beider, zuerst des nieder- wertigen, danach des höherwertigen Bytes eines 16-Bit-Zählers

# Bestückungsplan

R38 - R41 von der Lötseite ( L ) her montiert (Siehe Anmerkung 2)



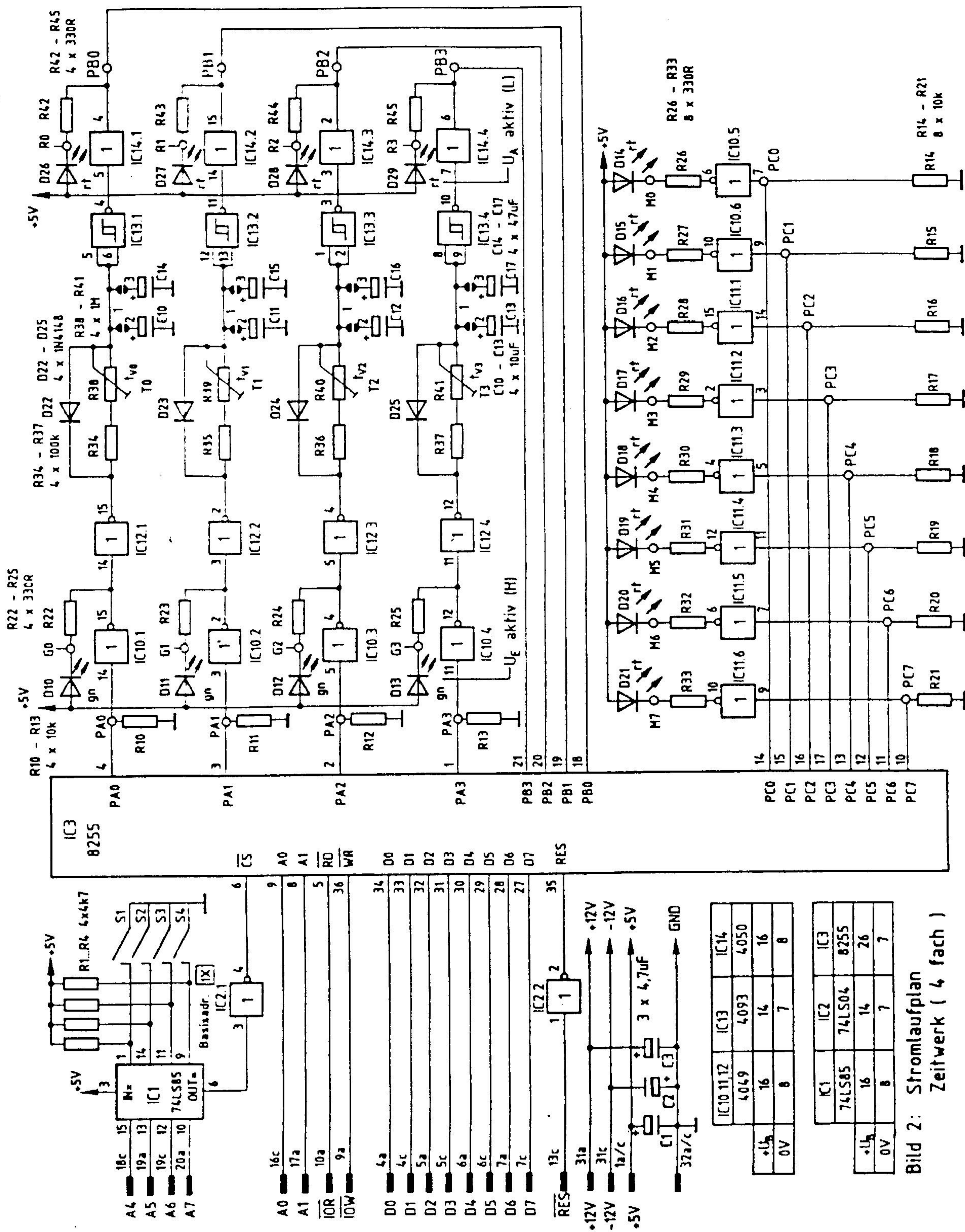


Bild 2: Stromlaufplan Zeitwerk ( 4 fach )