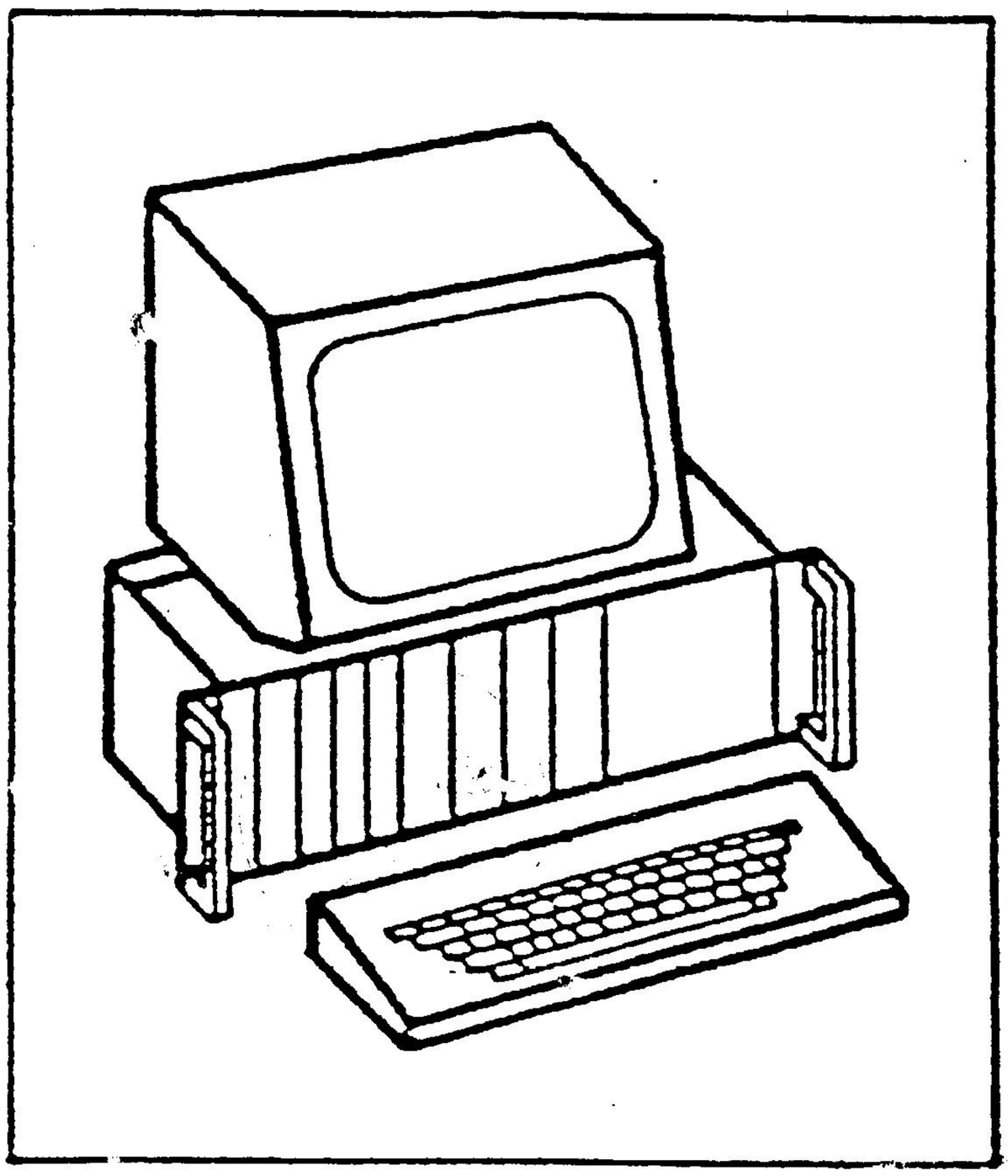
# MIKROCOMPUTER TECHNIK

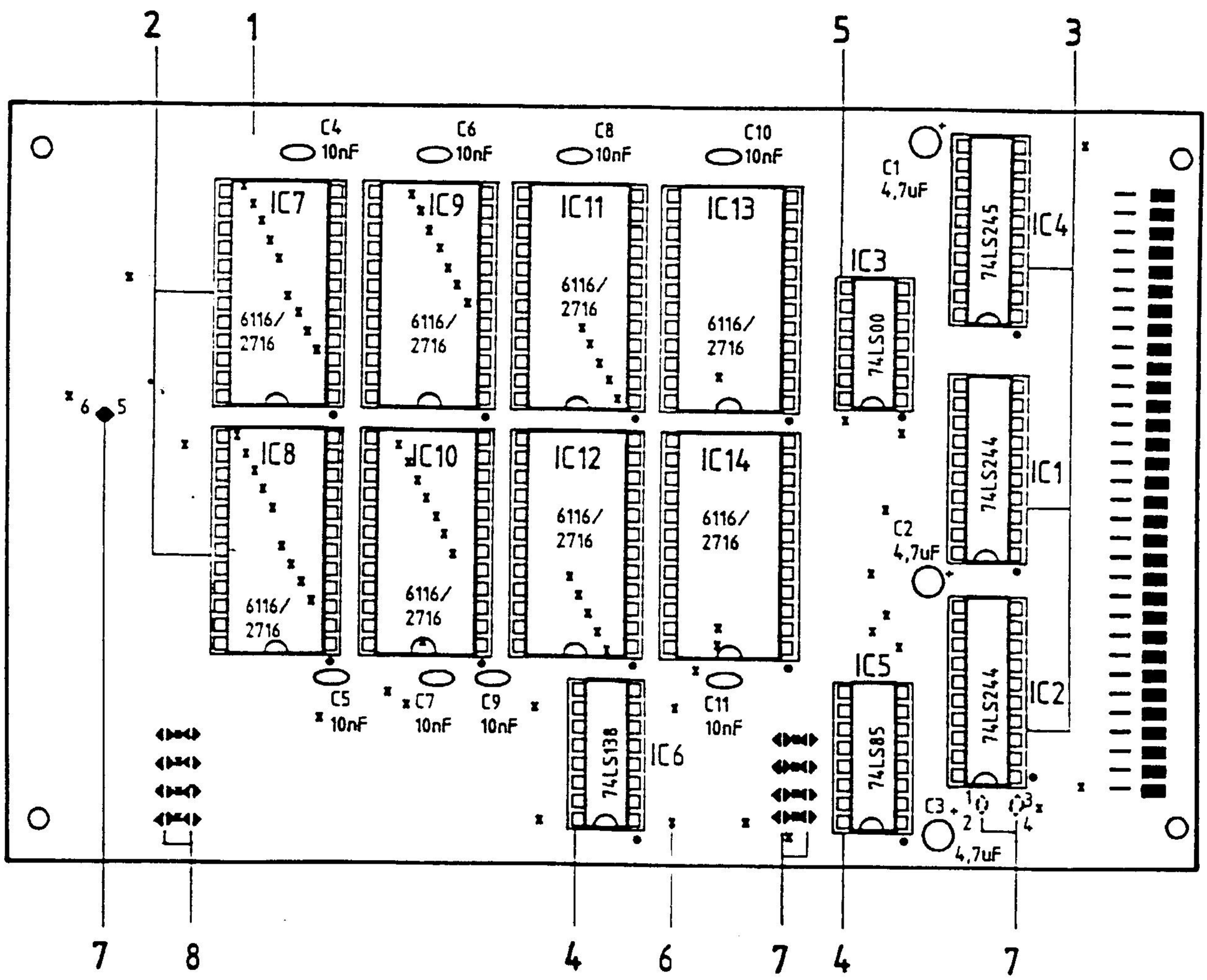


BfZ
MFA

# INHALTSVERZEICHNIS

## Teil 1 Beschreibung der Monitor-Kommandos

Bestückungsplan

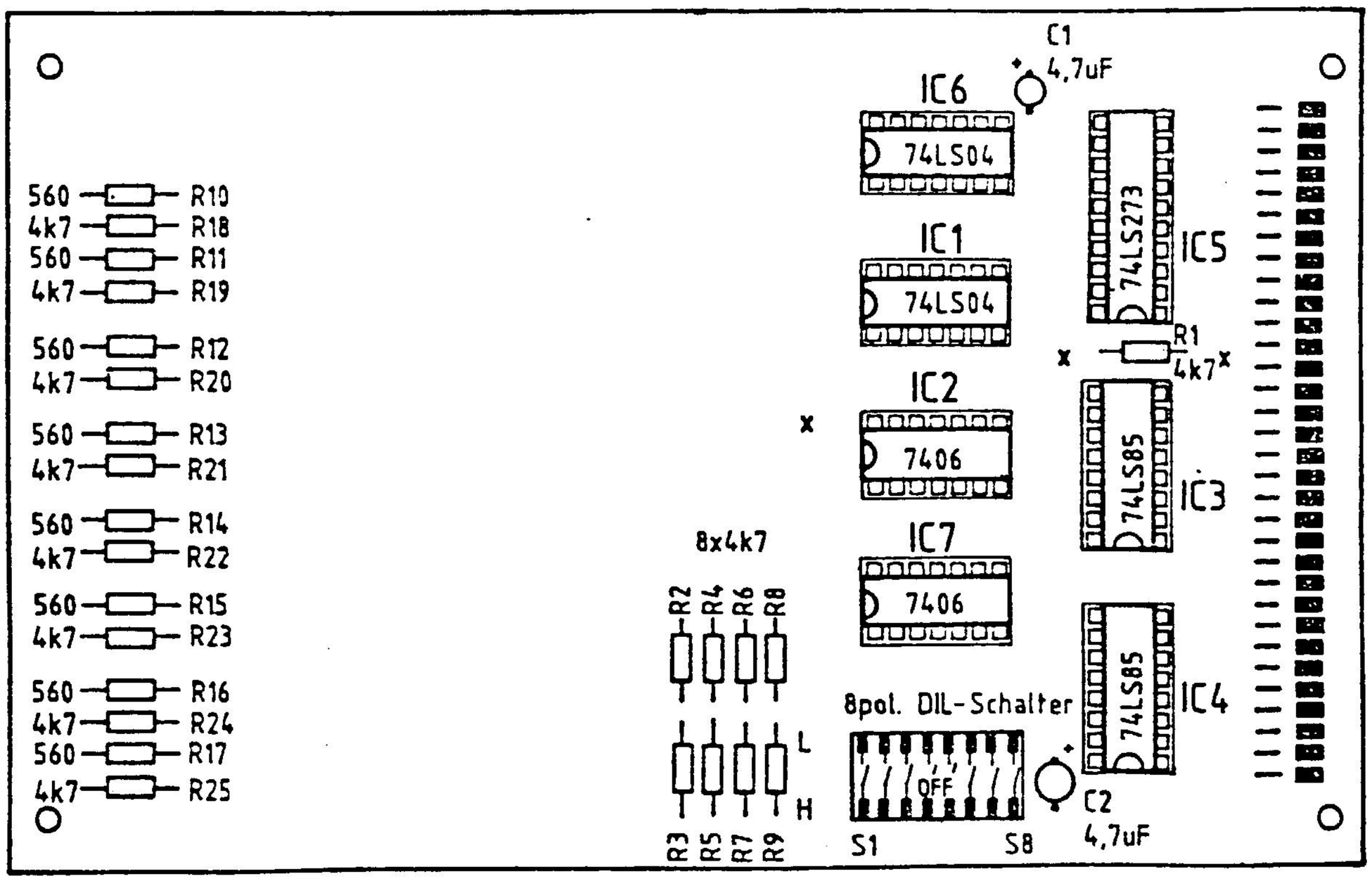



Die Lötbrücken 1-2, 3-4 und 5-6 (Pos. 7) sind zu schließen!

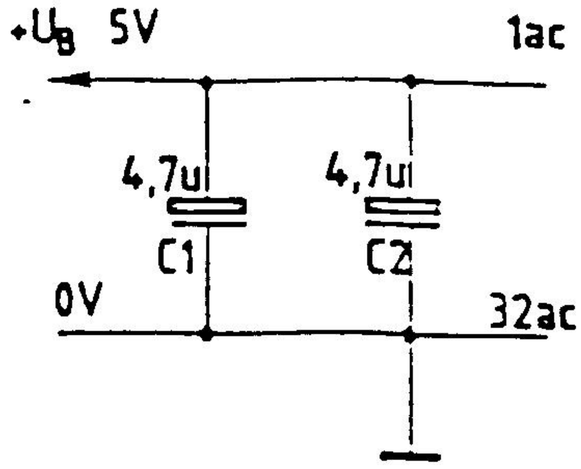
| A17 | A16 | A15 | A14 | Adreßbereich |
|---|---|---|---|---|
| L | L | L | L | 0000 - 3FFF |
| L | L | L | H | 4000 - 7FFF |
| L | L | H | L | 8000 - BFFF |
| L | L | H | H | C000 - FFFF |

| | IC1,IC2 | IC3 | IC4 | IC5 | IC6 | IC7...IC14 |
|---|---|---|---|---|---|---|
| | 74LS244 | 74LS00 | 74LS245 | 74LS85 | 74LS138 | 6116/2716 |
| +5V | 20 | 14 | 20 | 16 | 16 | 24 |
| GND | 10 | 7 | 10 | 8 | 8 | 12 |

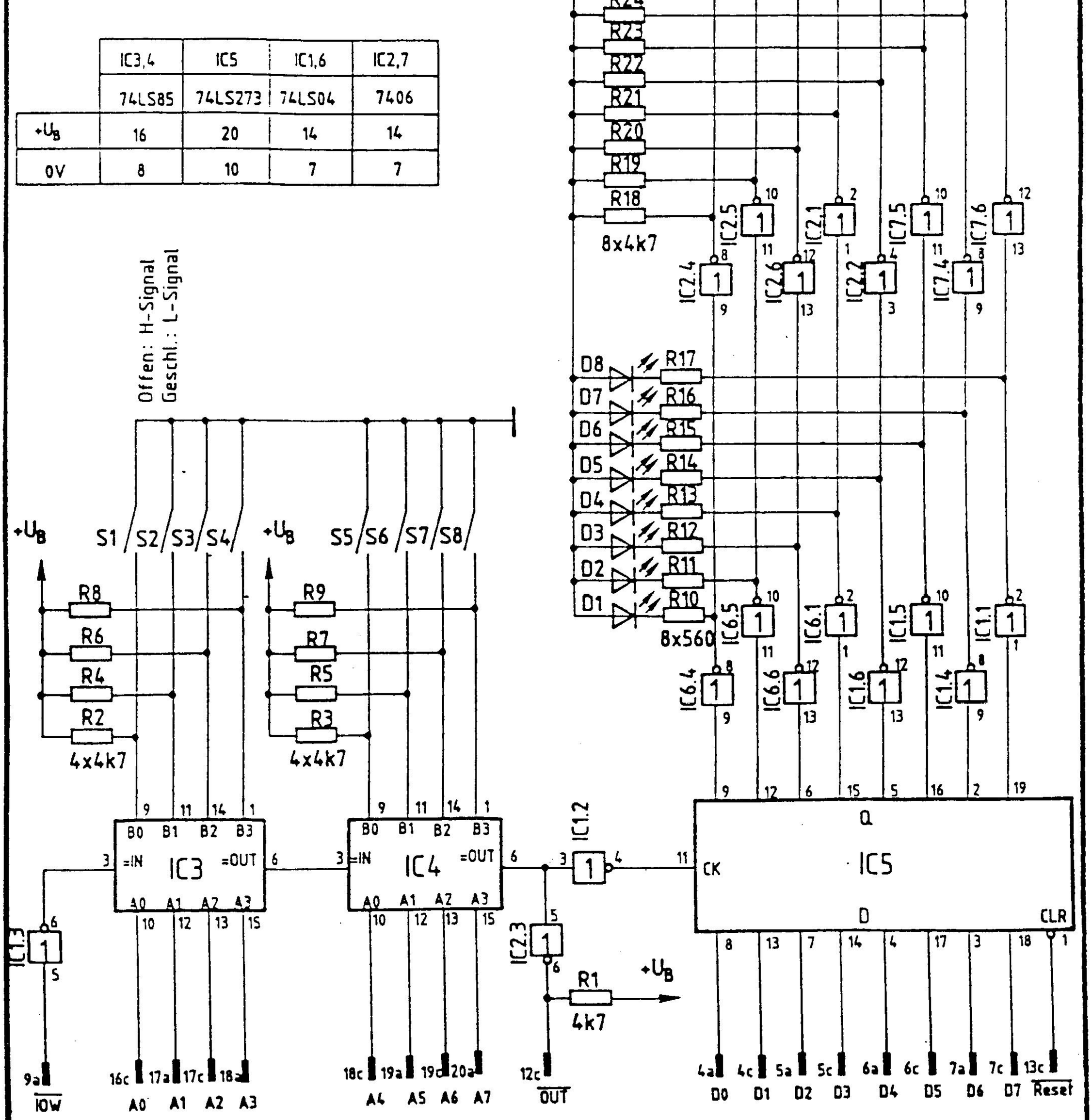
## Bestückungsplan




| | | |
|---|---|---|
| Bfz | Bestückungsplan 8-Bit-Parallel-Ausgabe | 50 |




| | IC3,4 | IC5 | IC1,6 | IC2,7 |
|---|---|---|---|---|
| | 74LS85 | 74LS273 | 74LS04 | 7406 |
| +UB | 16 | 20 | 14 | 14 |
| 0V | 8 | 10 | 7 | 7 |

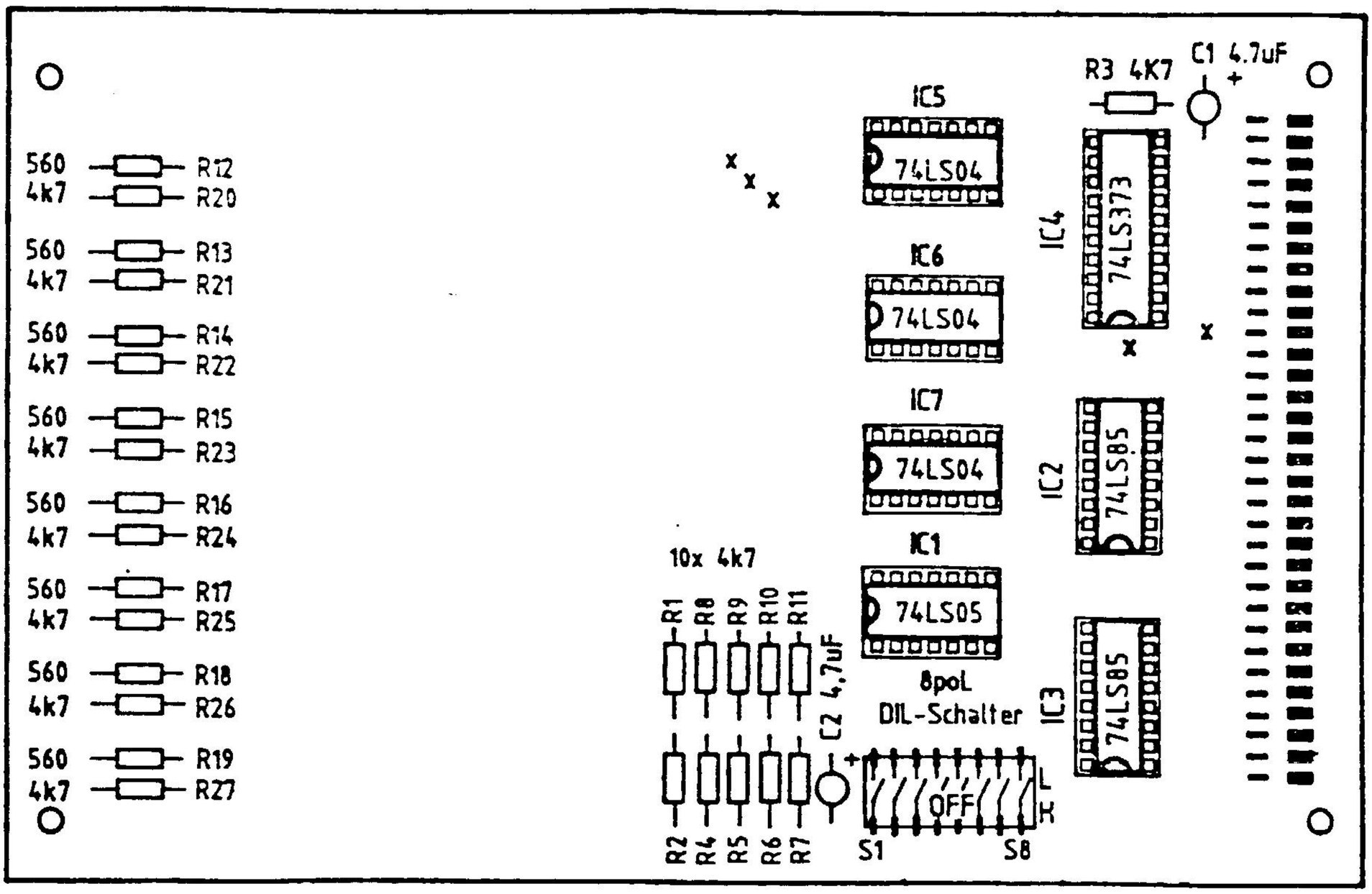



Bfz

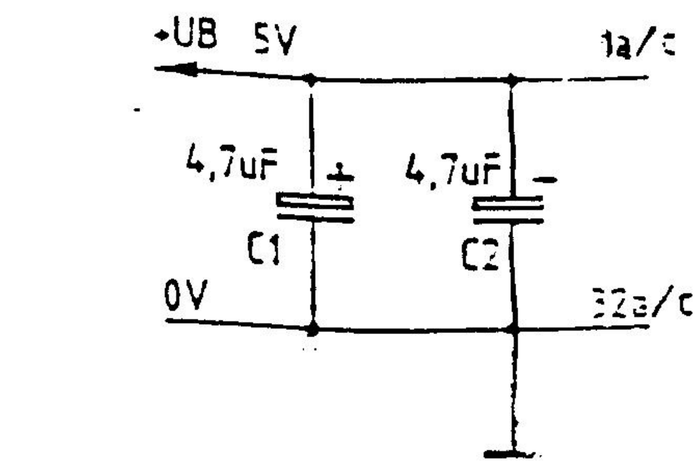
Stromlaufplan "8-Bit-Parallel-Ausgabe"

51

Bestückungsplan

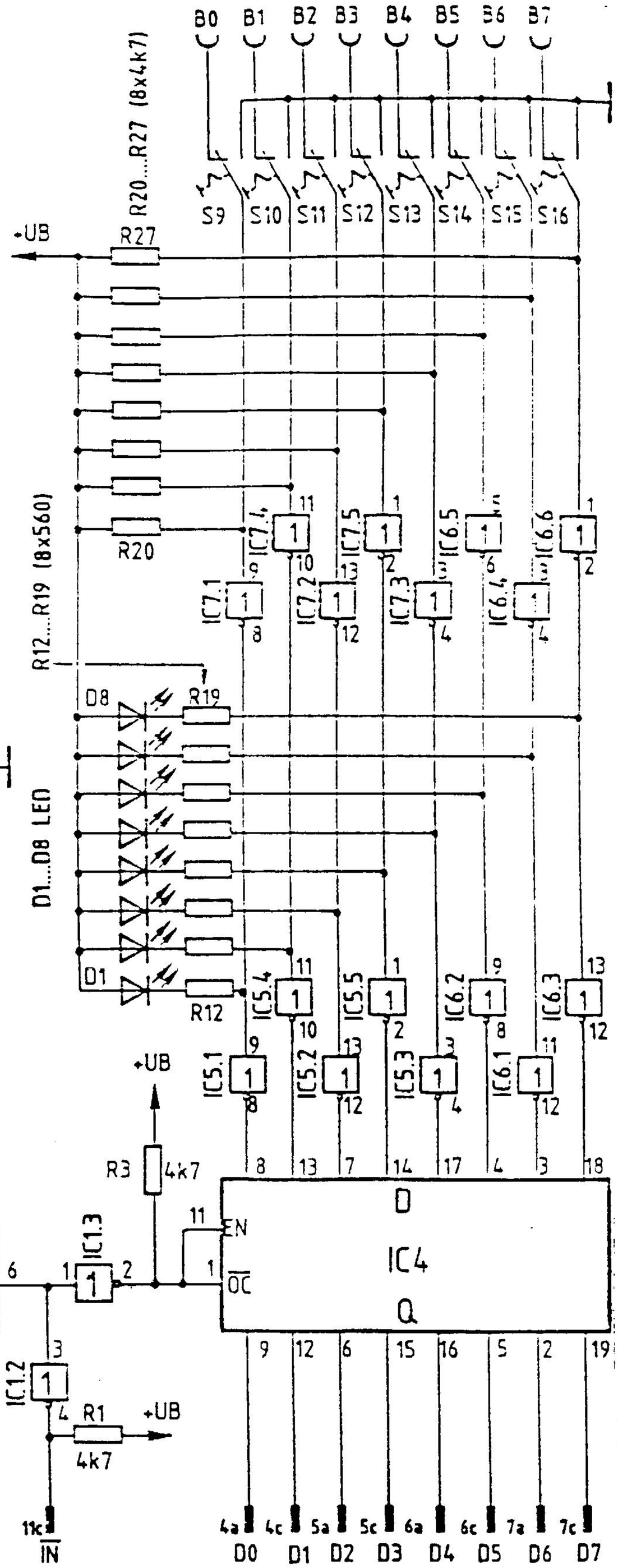



| | | |
|---|---|---|
| Bfz | Bestückungsplan 8-Bit-Parallel-Eingabe | 52 |



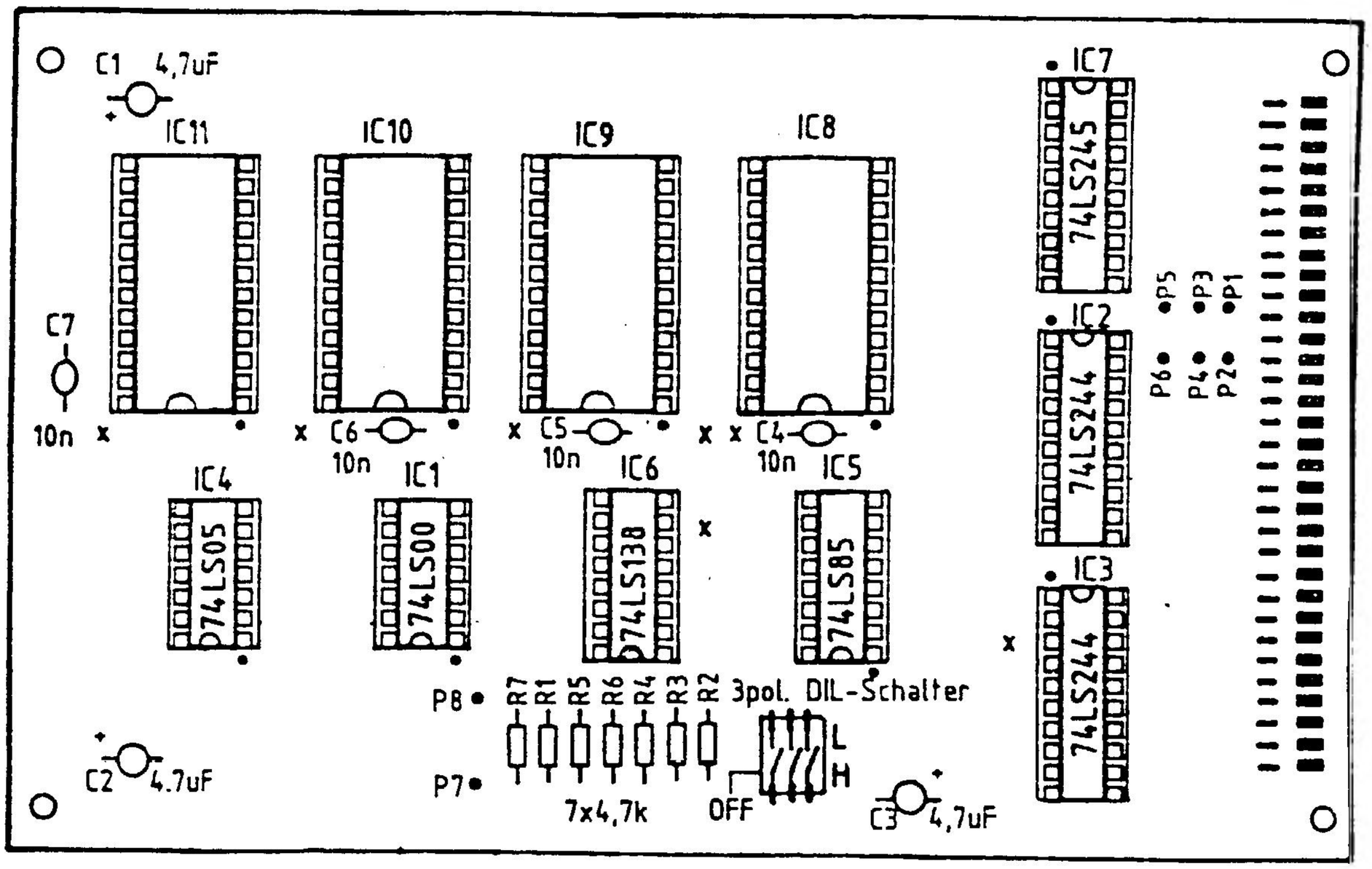

| | IC1 | IC2,3 | C4 | IC5,6,7 |
|---|---|---|---|---|
| | 74LS05 | 74LS85 | 74LS373 | 74LS04 |
| +UB | 14 | 16 | 20 | 14 |
| 0V | 7 | 8 | 10 | 7 |

offen: H-Signal
geschl.: L-Signal




| | | |
|---|---|---|
| Bfz | Stromlaufplan "8 Bit-Parallel-Eingabe" | 53 |

# Bestückungsplan




| | | |
|---|---|---|
| BfZ | Bestückungsplan 8-K-RAM/EPROM | 46 |

Bfz

Stromlaufplan 8-K-RAM/EPROM-Baugruppe

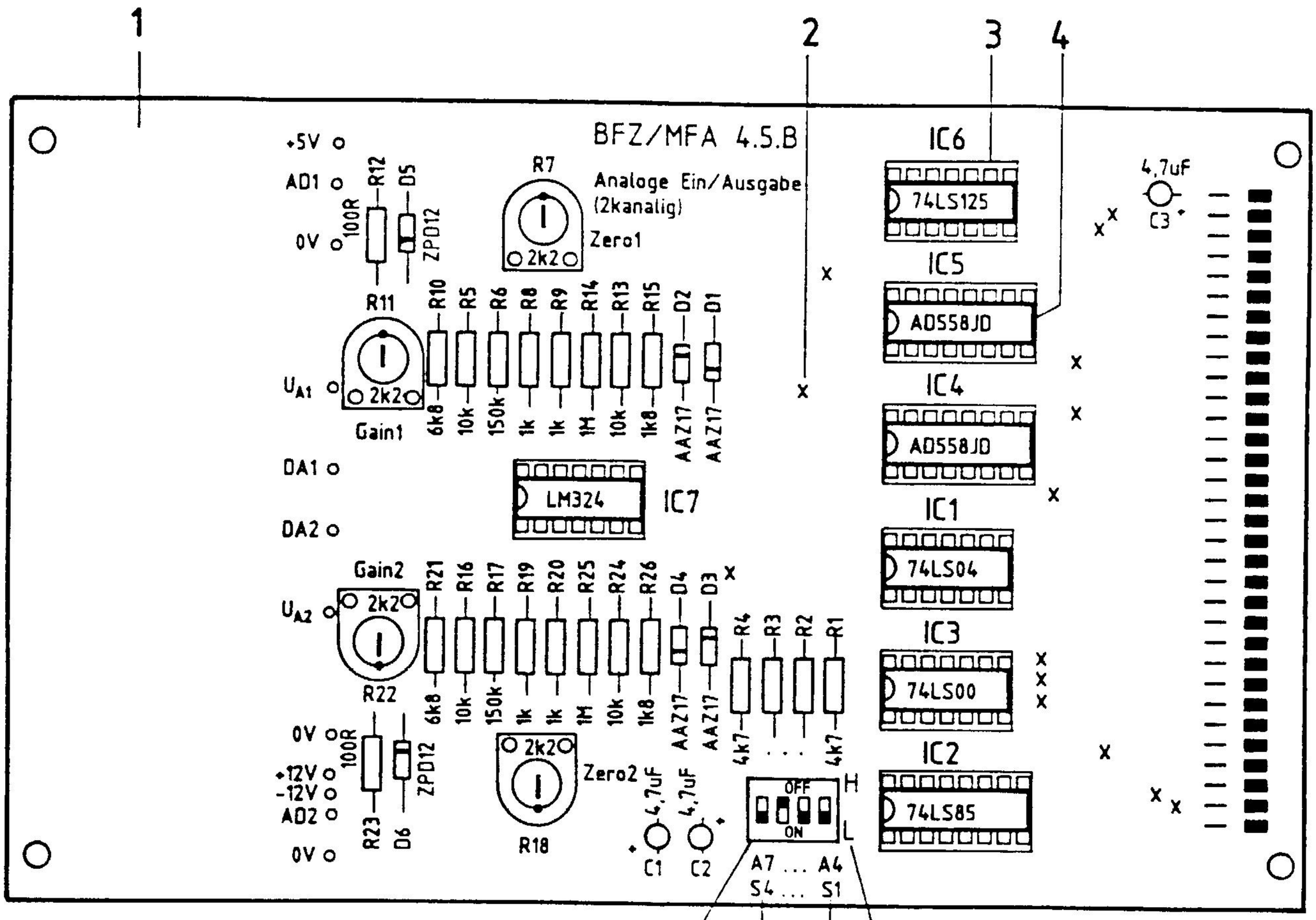
47

| | IC1. | IC2. IC3 | IC4 | IC5 | IC6 | IC7 | IC8... IC11 |
|---|---|---|---|---|---|---|---|
| | 74LS00 | 74LS244 | 74LS05 | 74LS85 | 74LS138 | 74LS245 | RAM/EPROM |
| $+U_B$ | 14 | 20 | 14 | 16 | 16 | 20 | 24 |
| 0V | 7 | 10 | 7 | 8 | 8 | 10 | 12 |

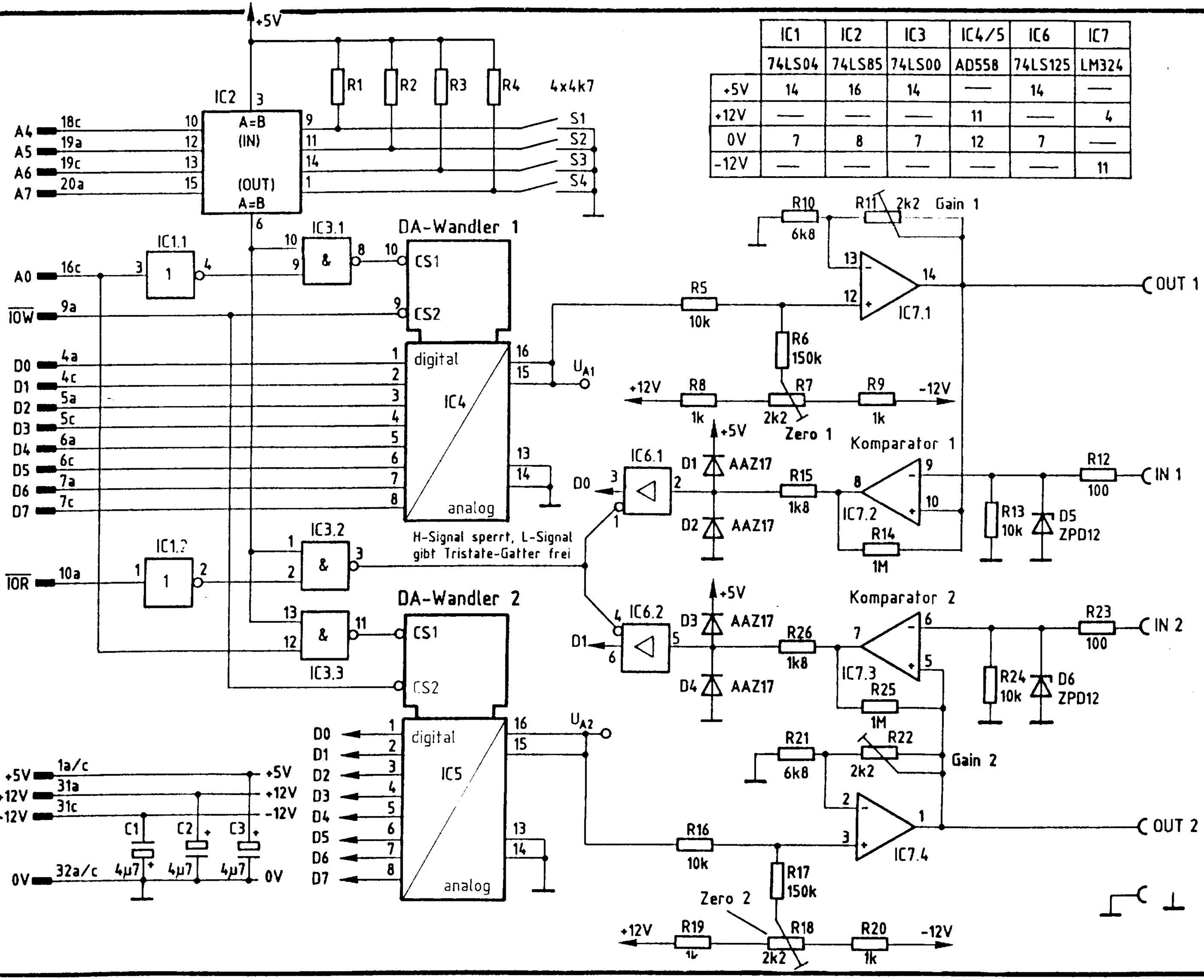
| | |
|---|---|
| RAM | P3- P4, P5-P6 brücken |
| EPROM | P1- P2, P7- P8 brücken |

MEMR 10c
MEMW 9c
P5 P6
IC1.1 &
+5V
R5 4k7
R6 4k7
R7 4k7
P7
P8
IC4.2 1
IC4.3 1
SA13
SA14
SA15
IC5
3x4k7
R2 R3 R4
IC4.1 1
ROM 11a P1 P2
RAM 12a P3 P4
R1 4k7
IC3
A15 24a
A14 23c
A13 23a
A12 22c
A11 22a
A10 21c
A9 21a
A8 20c
IC2
A7 20a
A6 19c
A5 19a
A4 18c
A3 18a
A2 17c
A1 17a
A0 16c
1a/c
32a/c
C1 4,7u
C2 4,7u
C3 4,7u
0V
IC1.2 &
IC6
G1
G2B
G2A
C
B
A
Y7
Y4
Y3
Y2
Y1
Y0
IC8
IC9
IC10
IC11
A10 OE WE CS
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
I/O0 I/O1 I/O2 I/O3 I/O4 I/O5 I/O6 I/O7
C4 10n
C5 10n
C6 10n
C7 10n
IC7
DIR EN
A B
D0 4a
D1 4c
D2 5a
D3 5c
D4 6a
D5 6c
D6 7a
D7 7c

Bestückungsplan



| | IC1 | IC2 | IC3 | IC4/5 | IC6 | IC7 |
|---|---|---|---|---|---|---|
| | 74LS04 | 74LS85 | 74LS00 | AD558 | 74LS125 | LM324 |
| +5V | 14 | 16 | 14 | — | 14 | — |
| +12V | — | — | — | 11 | — | 4 |
| 0V | 7 | 8 | 7 | 12 | 7 | — |
| -12V | — | — | — | — | — | 11 |



## Teil 2 Einige Befehlsbeschreibungen

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

| 1.Befehlsbyte binär | 1.Befehlsbyte hex. | Befehls-länge | Wirkung/Beispiel |
|---|---|---|---|
| 00111010 | 3A | 3 | Der Inhalt der Speicherstelle, deren Adresse auf den dem Befehl folgenden Speicherstellen steht, wird in den ACCU geladen (LDA).<br>Beispiel:<br>3A bewirkt, daß der Inhalt<br>04 der Speicherstelle 1304E<br>13 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).<br>Beispiel:<br>32 bewirkt, daß der ACCU-Inhalt<br>FE unter der Adresse 70FEH ab-<br>70 gespeichert wird. |
| 00111110 | 3E | 2 | Der ACCU wird mit dem Inhalt der dem Befehl folgenden Speicherstelle geladen (MVI).<br>Beispiel:<br>3E bewirkt, daß der ACCU mit<br>F9 dem 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).<br>Beispiel:<br>DB bewirkt, daß der Signalzu-<br>45 stand an den Eingängen der<br>Eingabe-Baugruppe Nr. 45E<br>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).<br>Beispiel:<br>D3 bewirkt, daß der ACCU-Inhalt<br>AC an die Ausgabe-Baugruppe<br>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- | Wirkung/Beispiel |
| binär | hex. | länge | |
| 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).<br>Beispiel:<br>C6 bewirkt, daß zum ACCU-<br>5B Inhalt 5BH hinzuaddiert wird. |
| 11010110 | D6 | 2 | Das dem Befehl folgende Daten-Byte wird vom ACCU-Inhalt abgezogen (SUI).<br>Beispiel:<br>D6 bewirkt, daß der ACCU-<br>3D 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).<br>Beispiel:<br>E6 wenn vor der Befehlsaus-<br>5A führung im ACCU 83H steht, so bewirkt der Befehl,daß im ACCU der Wert 02H 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).<br>Beispiel:<br>F6 wenn vor der Befehlsaus-<br>0F führung 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- | Wirkung/Beispiel |
| binär | hex. | länge | |
| 11000011 | C3 | 3 | Die Programmabarbeitung wird an der Speicherstelle fortgesetzt, deren Adresse in den dem Befehl folgenden Speicherstellen steht (JMP).<br>Beispiel:<br>C3 bewirkt, daß der nächste<br>12 Befehl von der Speicher-<br>F7 stelle F712H gelesen wird. |
| 11001010 | CA | 3 | Dieser Sprungbefehl zu der Speicheradresse, die in den folgenden Speicherstellen steht, wird nur ausgeführt, wenn die letzte Rechenoperation Null ergab (JZ).<br>Beispiel:<br>CA bewirkt, daß der nächste<br>12 Befehl nur dann von der<br>F7 Speicherstelle F712H 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 nicht Null (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

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

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

Befehlsliste: Hexadezimal

**Befehlsliste geordnet in Befehlsgruppen**
**Kurzfassung**

**Transfer 8 Bit**

MOVE IMMEDIATE

| | | |
|---|---|---|
| 06 MVI B, | | D8 |
| 0E MVI C, | | D8 |
| 16 MVI D, | | D8 |
| 1E MVI E, | | D8 |
| 26 MVI H, | | D8 |
| 2E MVI L, | | D8 |
| 36 MVI M, | | D8 |
| 3E MVI A, | | D8 |

LOAD/STORE

| | | |
|---|---|---|
| 3A LDA | | Adr |
| 32 STA | | Adr |
| 0A LDAX B | | |
| 1A LDAX D | | |
| 02 STAX B | | |
| 12 STAX D | | |

MOVE

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 MOV B,B | 50 MOV D,B | 60 MOV H,B | 70 MOV M,B |
| 41 MOV B,C | 51 MOV D,C | 61 MOV H,C | 71 MOV M,C |
| 42 MOV B,D | 52 MOV D,D | 62 MOV H,D | 72 MOV M,D |
| 43 MOV B,E | 53 MOV D,E | 63 MOV H,E | 73 MOV M,E |
| 44 MOV B,H | 54 MOV D,H | 64 MOV H,H | 74 MOV M,H |
| 45 MOV B,L | 55 MOV D,L | 65 MOV H,L | 75 MOV M,L |
| 46 MOV B,M | 56 MOV D,M | 66 MOV H,M | - |
| 47 MOV B,A | 57 MOV D,A | 67 MOV H,A | 77 MOV M,A |
| 48 MOV C,B | 58 MOV E,B | 68 MOV L,B | 78 MOV A,B |
| 49 MOV C,C | 59 MOV E,C | 69 MOV L,C | 79 MOV A,C |
| 4A MOV C,D | 5A MOV E,D | 6A MOV L,D | 7A MOV A,D |
| 4B MOV C,E | 5B MOV E,E | 6B MOV L,E | 7B MOV A,E |
| 4C MOV C,H | 5C MOV E,H | 6C MOV L,H | 7C MOV A,H |
| 4D MOV C,L | 5D MOV E,L | 6D MOV L,L | 7D MOV A,L |
| 4E MOV C,M | 5E MOV E,M | 6E MOV L,M | 7E MOV A,M |
| 4F MOV C,A | 5F MOV E,A | 6F MOV L,A | 7F MOV A,A |

INPUT/OUTPUT

| | | |
|---|---|---|
| D3 OUT | | Nr |
| DB IN | | Nr |

**Transfer 16 Bit**

LOAD IMMEDIATE

| | | |
|---|---|---|
| 01 LXI B, | | D16 |
| 11 LXI D, | | D16 |
| 21 LXI H, | | D16 |
| 31 LXI SP, | | D16 |

LOAD/STORE

| | | |
|---|---|---|
| 2A LHLD | | Adr |
| 22 SHLD | | Adr |
| EB XCHG | | |

STACK

| | |
|---|---|
| C5 PUSH B | C1 POP B |
| D5 PUSH D | D1 POP D |
| E5 PUSH H | E1 POP H |
| F5 PUSH PSW | F1 POP PSW* |
| | E3 XTHL |
| | F9 SPHL |

**Arithmetische Operation 8 Bit**

| ADDITION* | | SUBTRACTION* | | IN-CREMENT** | DE-CREMENT** |
|---|---|---|---|---|---|
| 80 ADD B | 88 ADC B | 90 SUB B | 98 SBB B | 04 INR B | 05 DCR B |
| 81 ADD C | 89 ADC C | 91 SUB C | 99 SBB C | 0C INR C | 0D DCR C |
| 82 ADD D | 8A ADC D | 92 SUB D | 9A SBB D | 14 INR D | 15 DCR D |
| 83 ADD E | 8B ADC E | 93 SUB E | 9B SBB E | 1C INR E | 1D DCR E |
| 84 ADD H | 8C ADC H | 94 SUB H | 9C SBB H | 24 INR H | 25 DCR H |
| 85 ADD L | 8D ADC L | 95 SUB L | 9D SBB L | 2C INR L | 2D DCR L |
| 86 ADD M | 8E ADC M | 96 SUB M | 9E SBB M | 34 INR M | 35 DCR M |
| 87 ADD A | 8F ADC A | 97 SUB A | 9F SBB A | 3C INR A | 3D DCR A |
| C6 ADI D8 | CE ACI D8 | D6 SUI D8 | DE SBI D8 | DECIMAL ADJUST* | 27 DAA |

**Arithmetische Operation 16 Bit**

| ADDITION* | INCREMENT | 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 Operation 8 Bit**

| AND* | XOR* | OR* | COMPARE* | COMPLEMENT |
|---|---|---|---|---|
| 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 CMP E | |
| A4 ANA H | AC XRA H | B4 ORA H | BC CMP H | |
| A5 ANA L | AD XRA L | B5 ORA L | BD CMP L | |
| A6 ANA M | AE XRA M | B6 ORA M | BE CMP M | |
| A7 ANA A | AF XRA A | B7 ORA A | BF CMP A | |
| E6 ANI D8 | EE XRI D8 | F6 ORI D8 | FE CPI D8 | |

**Sprung**

JUMP

| | |
|---|---|
| C3 JMP | Adr |
| C2 JNZ | Adr |
| CA JZ | Adr |
| D2 JNC | Adr |
| DA JC | Adr |
| E2 JPO | Adr |
| EA JPE | Adr |
| F2 JP | Adr |
| FA JM | Adr |
| E9 PCHL | |

CALL

| | |
|---|---|
| CD CALL | Adr |
| C4 CNZ | Adr |
| CC CZ | Adr |
| D4 CNC | Adr |
| DC CC | Adr |
| E4 CPO | Adr |
| EC CPE | Adr |
| F4 CP | Adr |
| FC CM | Adr |

RETURN

| | |
|---|---|
| C9 RET | |
| C0 RNZ | |
| C8 RZ | |
| D0 RNC | |
| D8 RC | |
| E0 RPO | |
| E8 RPE | |
| F0 RP | |
| F8 RM | |

RESTART

| | |
|---|---|
| C7 RST 0 | |
| CF RST 1 | |
| D7 RST 2 | |
| DF RST 3 | |
| E7 RST 4 | |
| EF RST 5 | |
| F7 RST 6 | |
| FF RST 7 | |

**Registeranweisungs- und sonstige Befehle**

| ROTATE* | CONTROL | CARRY* |
|---|---|---|
| 07 RLC | 00 NOP | 37 STC |
| 0F RRC | 76 HLT | 3F CMC |
| 17 RAL | F3 DI | |
| 1F RAR | FB EI | |

**zusätzliche Befehle des MP 8085**

20 RIM
30 SIM

**Erläuterungen zur Tabelle**

| | |
|---|---|
| D16 | = 16-Bit-Konstante |
| D8 | = 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 $r_1, r_2$ | 0 1 c c c s s s | - - - - - | 1 | 5 4 | Move register to register | $r_1, r_2$ = A, B, C, D, E, H oder L Lade Register $r_1$ mit dem Inhalt von Register $r_2$ |
| 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 0 0 0 1 1 | - - - - - | 1 | 18 16 | Exchange top of stack H & L | Vertausche Inhalt des Registerpaares (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 Registerpaares (H, L) |

### b) Speicher, Peripherie → Register

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MOV | $r_1$, M | 0 1 d d d 1 1 0 | - - - - - | 1 | 7 | 7 | Move memory to register | $r_1$ = A, B, C, D, E, H oder L Lade Register $r_1$ mit dem Inhalt des Speicherbytes, das durch den Inhalt des Registerpaares (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 Registerpaares 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<br>PSW | 1 1 r r 0 0 0 1<br>1 1 1 1 0 0 0 1 | - - - - - | 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, $r_1$ | 0 1 1 1 0 s s s | - - - - - | 1 | 7 | 7 | Move register to memory | $r_1$ = A, B, C, D, E, H oder L Inhalt von Register $r_1$ auf den Speicherplatz abspeichern, der durch den Inhalt des Registerpaares (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 Registerpaares 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 Registerpaares 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 Registerpaares (H, L) adressiert ist, mit Konstante (konst ≦ 255) |
| MVI | $r_1$, konst | 0 0 d d d 1 1 0 | - - - - - | 2 | 7 | 7 | Move immediate Register | $r_1$ = A, B, C, D, E, H oder L Lade Register $r_1$ mit Konstante (konst ≦ 255) |

| Mnemonic | | Binär-Code | Beeinflußte Zustands-Bits | Bytes | Taktzyklen 8080 / 8085 | | Englische Befehlsbeschreibung | Funktion des Befehls |
|---|---|---|---|---|---|---|---|---|
| **ARITHMETISCHE OPERATIONEN** | | | | | | | | |
| INR | $r_1$ | 00ddd100 | Z, S, P, -, AC | 1 | 5 | 4 | Increment register | $r_1$ = A, B, C, D, E, H oder L: Zum Inhalt des Registers $r_1$ 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 | $r_1$ | 00ddd101 | Z, S, P, -, AC | 1 | 5 | 4 | Decrement register | $r_1$ = A, B, C, D, E, H oder L Vom Inhalt des Registers $r_1$ 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 | $r_1$ | 10000sss | Z, S, P, CY, AC | 1 | 4 | 4 | Add register to accu | $r_1$ = A, B, C, D, E, H oder L: Inhalt von Register $r_1$ 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 | $r_1$ | 10001sss | Z, S, P, CY, AC | 1 | 4 | 4 | Add register to accu with carry | $r_1$ = A, B, C, D, E, H oder L: Inhalt von Register $r_1$ 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 | $r_1$ | 10010sss | Z, S, P, CY, AC | 1 | 4 | 4 | Subtract register from accu | $r_1$ = A, B, C, D, E, H oder L: Inhalt des Registers $r_1$ 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 | $r_1$ | 10011sss | Z, S, P, CY, AC | 1 | 4 | 4 | Subtract register from accu with borrow | $r_1$ = A, B, C, D, E, H oder L: Inhalt von Register $r_1$ 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) addressiert 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 ≦ 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 ≦ 255 und Carry-Bit addiert |
| SUI | konst | 11010110 | Z, S, P, CY, AC | 2 | 7 | 7 | Subtract immediate from accu | konst ≦ 255 wird vom Akkumulator-Inhalt subtrahiert |
| SBI | konst | 11011110 | Z, S, P, CY, AC | 2 | 7 | 7 | Subtract immediate from accu with borrow | konst ≦ 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 $D_0 - D_3$ Rest 6 addiert.

bei C = 1 " " $D_4 - D_7$ " 6 "

AC: Accillary Carry (Halb Carry) HJ

| Mnemonic | Binär-Code | Beeinflußte Zustands-Bits | Bytes | Taktzyklen | | Englische Befehlsbeschreibung | Funktion des Befehls |
|---|---|---|---|---|---|---|---|

## LOGISCHE OPERATIONEN

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CMA | 0 0 1 0 1 1 1 1 | - - - - - | 1 | 4 | 4 | Complement accu | Akkumulator-Inhalt wird negiert |
| ANA $r_1$ für 8085 | 1 0 1 0 0 s s s | Z. S. P. 0. AC<br>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 | 1 0 1 0 0 1 1 0 | Z. S. P. 0. AC<br>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 | 1 1 1 0 0 1 1 0 | Z. S. P. 0. AC<br>Z. S. P. 0. 1 | 2 | 7 | 7 | And immediate with accu | Akkumulator-Inhalt wird mit der konst ≦ 255 UND-verknüpft |
| ORA $r_1$ | 1 0 1 1 0 s s s | 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 | 1 0 1 1 0 1 1 0 | 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 | 1 1 1 1 0 1 1 0 | Z. S. P. 0. 0 | 2 | 7 | 7 | Or immediate with accu | Akkumulator-Inhalt wird mit konst ≦ 255 ODER-verknüpft |
| XRA $r_1$ | 1 0 1 0 1 s s s | 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 $r_1$ Exclusiv-ODER-verknüpft |
| XRA M | 1 0 1 0 1 1 1 0 | Z. S. P. 0. 0 | 1 | 7 | 7 | Exclusive Or memory with accu | Das über Registerpaar (H. L) adressierte Byte wird Exclusiv-ODER mit dem Akkumulator-Inhalt verknüpft |
| XRI konst | 1 1 1 0 1 1 1 0 | Z. S. P. 0. 0 | 2 | 7 | 7 | Exclusive Or immediate with accu | Der Akkumulator-Inhalt wird mit dem Wert konst ≦ 255 Exclusiv-ODER verknüpft |
| CMP $r_1$ | 1 0 1 1 1 s s s | 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 | 1 0 1 1 1 1 1 0 | 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 | 1 1 1 1 1 1 1 0 | Z. S. P. CY. AC | 2 | 7 | 7 | Compare immediate with accu | Akkumulator-Inhalt wird mit konst ≦ 255 verglichen |

## REGISTERANWEISUNGEN

### a) Akkumulator rotieren

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| RLC | 0 0 0 0 0 1 1 1 | - - - 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 | 0 0 0 0 1 1 1 1 | - - - 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 | 0 0 0 1 0 1 1 1 | - - - 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 | 0 0 0 1 1 1 1 1 | - - - 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 | 0 0 1 1 1 1 1 1 | - - - CY - | 1 | 4 | 4 | Complement carry | Carry-Bit wird negiert |
| STC | 0 0 1 1 0 1 1 1 | - - - 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

# SPRUNGBEFEHLE

## a) Unbedingte Sprünge

| Mnemonic | Binär-Code | Beeinflußte Zustands-Bits | Bytes | Taktzyklen 8080 / 8085 | Englische Befehlsbeschreibung | Funktion des Befehls |
|---|---|---|---|---|---|---|
| PCHL | 1 1 1 0 1 0 0 1 | - - - - - | 1 | 5 6 | H & L to program counter | Programm wird an der Adresse fortgesetzt, die im Registerpaar (H, L) steht |
| JMP adr | 1 1 0 0 0 0 1 1 | - - - - - | 3 | 10 10 | Jump unconditional | Programm wird an der Adresse adr fortgesetzt |

## b) Bedingte Sprünge

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| JC | adr | 1 1 0 1 1 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on carry | Bei Carry-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| JNC | adr | 1 1 0 1 0 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on no carry | Bei Carry-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt |
| JZ | adr | 1 1 0 0 1 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on zero | Bei Zero-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| JNZ | adr | 1 1 0 0 0 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on no zero | Bei Zero-Bit = 0 wird das Programm bei der Adresse fortgesetzt |
| JM | adr | 1 1 1 1 1 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on minus | Bei Sign-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| JP | adr | 1 1 1 1 0 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on positiv | Bei Sign-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt |
| JPE | adr | 1 1 1 0 1 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on panty even | Bei Panty-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| JPO | adr | 1 1 1 0 0 0 1 0 | - - - - - | 3 | 10 7/10 | Jump on panty odd | Bei Panty-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt |

| Mnemonic | Binär-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 | 1 1 0 0 1 1 0 1 | - - - - - | 3 | 17 18 | Call unconditional | Programm wird bei der Adresse fortgesetzt |
| CC | adr | 1 1 0 1 1 1 0 0 | - - - - - | 3 | 11/17 9/18 | Call on carry | Bei Carry-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| CNC | adr | 1 1 0 1 0 1 0 0 | - - - - - | 3 | 11/17 9/18 | Call on no carry | Bei Carry-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt |
| CZ | adr | 1 1 0 0 1 1 0 0 | - - - - - | 3 | 11/17 9/18 | Call on zero | Bei Zero-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| CNZ | adr | 1 1 0 0 0 1 0 0 | - - - - - | 3 | 11/17 9/18. | Call on no zero | Bei Zero-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt |
| CM | adr | 1 1 1 1 1 1 0 0 | - - - - - | 3 | 11/17 9/18 | Call on minus | Bei Sign-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| CP | adr | 1 1 1 1 0 1 0 0 | - - - - - | 3 | 11/17 9/18 | Call on positiv | Bei Sign-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt |
| CPE | adr | 1 1 1 0 1 1 0 0 | - - - - - | 3 | 11/17 9/18 | Call on parity even | Bei Parity-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt |
| CPO | adr | 1 1 1 0 0 1 0 0 | - - - - - | 3 | 11/17 9/18 | Call on parity odd | Bei Parity-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt |
| RST | konst | 1 1 a a a 1 1 1 | - - - - - | 1 | 11 12 | Restart | Programm wird auf der Adresse 8 × konst fortgesetzt ($0 \leq konst \leq 7$) |

## b) Rücksprungbefehle

| | | | | | | |
|---|---|---|---|---|---|---|
| RET | 1 1 0 0 1 0 0 1 | - - - - - | 1 | 10 10 | Return | Programm wird an der Adresse fortgesetzt, die in dem Wort steht, das über dem Stackpointer adressiert ist |
| RC | 1 1 0 1 1 0 0 0 | - - - - - | 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 | 1 1 0 1 0 0 0 0 | - - - - - | 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 | 1 1 0 0 1 0 0 0 | - - - - - | 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 | 1 1 0 0 0 0 0 0 | - - - - - | 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 | 1 1 1 1 1 0 0 0 | - - - - - | 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 | 1 1 1 1 0 0 0 0 | - - - - - | 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 | 1 1 1 0 1 0 0 0 | - - - - - | 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 | 1 1 1 0 0 0 0 0 | - - - - - | 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 |

| Mnemonic | Binär-Code | Beeinflußte Zustands-Bits | Bytes | Taktzyklen 8080 / K085 | Englische Befehlsbeschreibung | Funktion des Befehls |
|---|---|---|---|---|---|---|

## PROGRAMMUNTERBRECHUNG

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EI | 1 1 1 1 1 0 1 1 | - - - - - - | 1 | 4 | 4 | Enable interrupts | INTE-Flipflop wird gesetzt, der Mikroprozessor kann eine Unterbrechungsanforderung annehmen |
| DI | 1 1 1 1 0 0 1 1 | - - - - - - | 1 | 4 | 4 | Disable interrupts | INTE-Flipflop wird rückgesetzt; der Mikroprozessor ignoriert Unterbrechungsanforderungen |

## SONSTIGE BEFEHLE

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| HLT | 0 1 1 1 0 1 1 0 | - - - - - - | 1 | 7 | 5 | Halt | Programm hält an, bis eine Unterbrechungsanforderung eintritt |
| NOP | 0 0 0 0 0 0 0 0 | - - - - - - | 1 | 4 | 4 | No operation | Leerbefehl |

## SAB 8085 BEFEHLE

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| RIM | 0 0 1 0 0 0 0 0 | - - - - - - | 1 | - | 4 | Read interrupt Mask | Lesen Unterbrechungsmaske und seriellen Eingang in Akkumulator |
| SIM | 0 0 1 1 0 0 0 0 | - - - - - - | 1 | - | 4 | Set interrupt Mask | Setzen Unterbrechungsmaske und seriellen Ausgang |

**Legende zu „Binär-Code“**

ddd = Zielregister
rp = Registerpaar
sss = Quellregister

**ddd/sss**

000 ≙ Reg. B
001 ≙ Reg. C
010 ≙ Reg. D
011 ≙ Reg. E
100 ≙ Reg. H
101 ≙ Reg. L
110 ≙ Speicher (M)
111 ≙ Akku

**rp**

00 ≙ B/C
01 ≙ D/E
10 ≙ H/L
11 ≙ Stack Ptr. (SP)

Verschiedenes: I- ACCU löschen mit XRA A (gleichzeitig wird das Carry-Bit gelöscht )

I- FLAGs setzen mit ORA A

| Befehlsgruppe (Bedeutung der Spalten) | Mnemonik | 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)<br>(B) ← (B3)<br>m= (B3)(B2) | x | x | x | x | x |
| | LXI D, m | 1 1 | 10 | 3 | 3 | (E) ← (B2)<br>(D) ← (B3)<br>m= (B3)(B2) | x | x | x | x | x |
| | LXI H, m | 2 1 | 10 | 3 | 3 | (L) ← (B2)<br>(H) ← (B3)<br>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)<br>(m+1) ← (H) | x | x | x | x | x |
| | LHLD m | 2 A | 16 | 3 | 5 | (L) ← (m)<br>(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) | * | * | * | * | * |
| | ADD M | 8 6 | 7 | 1 | 2 | (A) ← (A)+(M) | * | * | * | * | * |
| | ADI n | C 6 | 7 | 2 | 2 | (A) ← (A)+n | * | * | * | * | * |
| | ADC r | | 4 | 1 | 1 | (A) ← (A)+(r)+(C) | * | * | * | * | * |
| | ADC M | 8 E | 7 | 1 | 2 | (A) ← (A)+(M)+(C) C= Carry | * | * | * | * | * |
| | ACI n | C E | 7 | 2 | 2 | (A) ← (A)+n+(C) | * | * | * | * | * |
| | DAD B | 0 9 | 10 | 1 | 3 | (H) (L) ← (H) (L)+(B) (C) | x | x | x | * | x |
| | DAD D | 1 9 | 10 | 1 | 3 | (H) (L) ← (H) (L)+(D) (E) | x | x | x | * | x |
| | DAD H | 2 9 | 10 | 1 | 3 | (H) (L) ← (H) (L)+(H) (L) | x | x | x | * | x |
| | DAD SP | 3 9 | 10 | 1 | 3 | (H) (L) ← (H) (L)+(SP) | x | x | x | * | x |
| | SUB r | | 4 | 1 | 1 | (A) ← (A)-(r) | * | * | * | * | * |
| | SUB M | 9 6 | 7 | 1 | 2 | (A) ← (A)-(M) | * | * | * | * | * |
| | SUI n | D 6 | 7 | 2 | 2 | (A) ← (A)-n | * | * | * | * | * |
| | SBB r | | 4 | 1 | 1 | (A) ← (A)-(r)-(C) | * | * | * | * | * |
| | SBB M | 9 E | 7 | 1 | 2 | (A) ← (A)-(M)-(C) C= Carry | * | * | * | * | * |
| | SBI n | D E | 7 | 2 | 2 | (A) ← (A)-n-(C) | * | * | * | * | * |
| | ANA r | | 4 | 1 | 1 | (A) ← (A)∧(r) | * | * | * | 0 | 1 |
| | ANA M | A 6 | 7 | 1 | 2 | (A) ← (A)∧(M) | * | * | * | 0 | 1 |
| | ANI n | E 6 | 7 | 2 | 2 | (A) ← (A)∧n | * | * | * | 0 | 1 |
| | XRA r | | 4 | 1 | 1 | (A) ← (A)⊻(r) | * | * | * | 0 | 0 |
| | XRA M | A E | 7 | 1 | 2 | (A) ← (A)⊻(M) | * | * | * | 0 | 0 |
| | XRI n | E E | 7 | 2 | 2 | (A) ← (A)⊻n | * | * | * | 0 | 0 |
| | ORA r | | 4 | 1 | 1 | (A) ← (A)∨(r) | * | * | * | 0 | 0 |
| | ORA M | B 6 | 7 | 1 | 2 | (A) ← (A)∨(M) | * | * | * | 0 | 0 |
| | ORI n | F 6 | 7 | 2 | 2 | (A) ← (A)∨n | * | * | * | 0 | 0 |
| | CMP r | | 4 | 1 | 1 | (A) − (r) | * | * | * | * | * |
| | CMP M | B E | 7 | 1 | 2 | (A) − (M) | * | * | * | * | * |
| | CPI n | F E | 7 | 2 | 2 | (A) − n | * | * | * | * | * |
| Register Inkrement Dekrement | INR r | | 4 | 1 | 1 | (r) ← (r)+1 | * | * | * | x | * |
| | INR M | 3 4 | 10 | 1 | 3 | (M) ← (M)+1 | * | * | * | x | * |
| | DCR r | | 4 | 1 | 1 | (r) ← (r)-1 | * | * | * | x | * |
| | DCR M | 3 5 | 10 | 1 | 3 | (M) ← (M)-1 | * | * | * | x | * |
| | INX B | 0 3 | 6 | 1 | 1 | (B) (C) ← (B) (C)+1 | x | x | x | x | x |
| | INX D | 1 3 | 6 | 1 | 1 | (D) (E) ← (D) (E)+1 | x | x | x | x | x |
| | INX H | 2 3 | 6 | 1 | 1 | (H) (L) ← (H) (L)+1 | x | x | x | x | x |
| | INX 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 | * | x |
| | RRC | 0 F | 4 | 1 | 1 | Rechts schieben C → A7 A6 ... A1 A0 → | x | x | x | * | x |
| | RAL | 1 7 | 4 | 1 | 1 | Links rotieren C ← A7 A6 ... A1 A0 ← | x | x | x | * | x |
| | RAR | 1 F | 4 | 1 | 1 | Rechts rotieren C → A7 A6 ... A1 A0 → | x | x | x | * | x |
| Akku beeinfl. | CMA | 2 F | 4 | 1 | 1 | (A) ← (A) | x | x | x | - | - |
| | DAA | 2 7 | 4 | 1 | 1 | (A) in BCD wandeln | * | * | * | * | * |
| 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 | * | x |

| Befehls-gruppe \ Bedeutung der Spalten | Mnemonik | 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 | X | X | X | X | X |
| | JNC m | D 2 | 10/7 | 3 | 3/2 | (C)= 0 Wenn Bedingung erfüllt | X | X | X | X | X |
| | JZ m | C A | 10/7 | 3 | 3/2 | (Z)= 1 (PC) ← m | X | X | X | X | X |
| | JNZ m | C 2 | 10/7 | 3 | 3/2 | (Z)= 0 | X | X | X | X | X |
| | JP m | F 2 | 10/7 | 3 | 3/2 | (N)= 0 | X | X | X | X | X |
| | JM m | F A | 10/7 | 3 | 3/2 | (N)= 1 Wenn Bedingung nicht erfüllt | X | X | X | X | X |
| | JPE m | E A | 10/7 | 3 | 3/2 | (P)= 1 (PC) ← (PC)+ 3 | X | X | X | X | X |
| | JPO m | E 2 | 10/7 | 3 | 3/2 | (P)= 0 | X | X | X | X | X |
| Unterprogrammaufrufe | CALL m | C D | 18 | 3 | 5 | ((SP)- 1) ((SP)- 2) ← (PC)+ 3, (PC) ← m; (SP) ← (SP)- 2 | X | X | X | X | X |
| | RST n | | 12 | 1 | 3 | ((SP)- 1) ((SP)- 2) ← (PC)+ 1, (PC) ← n x 8; (SP) ← (SP)- 2 wobei 0 ≤ n ≤ 7 | X | X | X | X | X |
| | CC m | D C | 18/9 | 3 | 5/2 | (C)= 1 | X | X | X | X | X |
| | CNC m | D 4 | 18/9 | 3 | 5/2 | (C)= 0 Wenn Bedingung erfüllt | X | X | X | X | X |
| | CZ m | C C | 18/9 | 3 | 5/2 | (Z)= 1 ((SP)- 1) ((SP)- 2) ← (PC)+ 3 | X | X | X | X | X |
| | CNZ m | C 4 | 18/9 | 3 | 5/2 | (Z)= 0 (PC) ← m, (SP) ← (SP)- 2 | X | X | X | X | X |
| | CP m | F 4 | 18/9 | 3 | 5/2 | (N)= 0 | X | X | X | X | X |
| | CM m | F C | 18/9 | 3 | 5/2 | (N)= 1 Wenn Bedingung nicht erfüllt | X | X | X | X | X |
| | CPE m | E C | 18/9 | 3 | 5/2 | (P)= 1 (PC) ← (PC)+ 3 | X | X | X | X | X |
| | CPO m | E 4 | 18/9 | 3 | 5/2 | (P)= 0 | X | X | X | X | X |
| Unterprogramm-Rücksprünge | RET | C 9 | 10 | 1 | 3 | (PC) ← ((SP)+ 1) ((SP)), (SP) ← (SP)+ 2 | X | X | X | X | X |
| | RC | D 8 | 12/6 | 1 | 3/1 | (C)= 1 | X | X | X | X | X |
| | RNC | D 0 | 12/6 | 1 | 3/1 | (C)= 0 Wenn Bedingung erfüllt | X | X | X | X | X |
| | RZ | C 8 | 12/6 | 1 | 3/1 | (Z)= 1 (PC) ← ((SP)+ 1) ((SP)), (SP) ← (SP)+ 2 | X | X | X | X | X |
| | RNZ | C 0 | 12/6 | 1 | 3/1 | (Z)= 0 | X | X | X | X | X |
| | RP | F 0 | 12/6 | 1 | 3/1 | (N)= 0 | X | X | X | X | X |
| | RM | F 8 | 12/6 | 1 | 3/1 | (N)= 1 Wenn Bedingung nicht erfüllt | X | X | X | X | X |
| | RPE | E 8 | 12/6 | 1 | 3/1 | (P)= 1 (PC) ← (PC)+ 1 | X | X | X | X | X |
| | RPO | E 0 | 12/6 | 1 | 3/1 | (P)= 0 | X | 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 | X |
| | OUT n | D 3 | 10 | 2 | 3 | (Ausgabe Gerät n) ← (A) | X | X | X | X | X |
| Unterbrech. Steuerung | EI | F B | 4 | 1 | 1 | (INTE) ← 1 | X | X | X | X | X |
| | DI | F 3 | 4 | 1 | 1 | (INTE) ← 0 | X | X | X | X | X |
| Stapelsteuerung | PUSH PSW | F 5 | 12 | 1 | 3 | ((SP)- 1) ← (A), ((SP)- 2) ← (F), (SP) ← (SP)- 2 | X | X | X | X | X |
| | PUSH B | C 5 | 12 | 1 | 3 | ((SP)- 1) ← (B), ((SP)- 2) ← (C), (SP) ← (SP)- 2 | X | X | X | X | X |
| | PUSH D | D 5 | 12 | 1 | 3 | ((SP)- 1) ← (D), ((SP)- 2) ← (E), (SP) ← (SP)- 2 | X | X | X | X | X |
| | PUSH H | E 5 | 12 | 1 | 3 | ((SP)- 1) ← (H), ((SP)- 2) ← (L), (SP) ← (SP)- 2 | X | X | X | X | X |
| | POP PSW | F 1 | 10 | 1 | 3 | (F) ← ((SP)), (A) ← ((SP)+ 1), (SP) ← (SP)+ 2 | * | * | * | * | * |
| | POP B | C 1 | 10 | 1 | 3 | (C) ← ((SP)), (B) ← ((SP)+ 1), (SP) ← (SP)+ 2 | X | X | X | X | X |
| | POP D | D 1 | 10 | 1 | 3 | (E) ← ((SP)), (D) ← ((SP)+ 1), (SP) ← (SP)+ 2 | X | X | X | X | X |
| | POP H | E 1 | 10 | 1 | 3 | (L) ← ((SP)), (H) ← ((SP)+ 1), (SP) ← (SP)+ 2 | X | X | X | X | X |
| Sonstige | HLT | 7 6 | 5 | 1 | 1 | (PC) ← (PC)+ 1 | X | X | X | X | X |
| | NOP | 0 0 | 4 | 1 | 1 | (PC) ← (PC)+ 1 | X | X | X | X | X |
| 8085 Befehle | RIM | 2 0 | 4 | 1 | 1 | Liest Unterbrechungsmaske und seriellen Eingang in Akku | X | X | X | X | X |
| | SIM | 3 0 | 4 | 1 | 1 | Setzt Unterbrechungsmaske und seriellen Ausgang | X | X | X | X | X |

| Symbol | Erklärung |
|---|---|
| r | Register (A, B, C, D, E, H, L, M) |
| m | Zwei- Byte Daten |
| n | Ein- Byte Daten |
| ⟨B2⟩, ⟨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 H und L adressierten Speicherzeile |

## 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 Ausdrucke werden vom Betriebsprogramm ausgeführt.

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

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:

```
                              ←——— CR
KMD> MEMORY
START-ADR = XXXX E000 CR
FORMAT    = H CR

E000  XX 41 ␣
E001  XX
```

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

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

## 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 Ausdrucke 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.

Aufruf:

```
                                        CR
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 Ausdrucke werden vom Betriebsprogramm ausgeführt.

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

## Das NEXT-INSTRUCTION-Kommando

Über dieses Kommando ist eine schrittweise Abarbeitung (Trace-Lauf, trace=Sour, verfolgen ...) eines Anwender-programms 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:

```
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F800 | START: | MVI | A,03 | 03 | 00000 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | FC32 |
| F802 | SCHL1: | DCR | A | 02 | 00100 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | FC32 |
| F803 | | JNZ | SCHL1 | 02 | 00100 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | FC32 |
| F802 | SCHL1: | DCR | A | 01 | 00100 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | FC32 |
| F803 | | JNZ | SCHL1 | 01 | 00100 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | FC32 |
| F802 | SCHL1: | DCR | A | | | | | | | | | | |

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen-
alle weiteren Ausdrucke 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:

```
                          [CR]
KMD>ASSEMBLER
START-ADR=XXXX E000 [CR]
E000
```

```
          EXXX              END
*** RESTART ? (JA/NEIN) N
```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen- alle weiteren Ausdrucke 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          }   SIEHE ASSEMBLER-/DISASSEMBLER-KOMMANDOS
ADR.FELD    }

A  - REGISTER A (ACCU, 8 BIT)

N  - NEGATIV   -FLAG   (NEGATIV-BIT)                    }
Z  - ZERO      -FLAG   (NULL-BIT)                       }  FLAG-BITS DES
H  - HALF CARRY-FLAG   (ZWISCHENUEBERTRAGS-BIT)         }  PROZESSOR-
P  - PARITY    -FLAG   (PARITAETS-BIT)                  }  STATUS-REGISTERS
C  - CARRY     -FLAG   (UEBERTRAGS-BIT)                 }  (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.

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 F8 20 00 FC32
 CR
KMD >
```

Hinweis: Vom Benutzer einzugebende Zeichen sind unterstrichen. Alle weiteren Ausdrucke 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 Programm-Startadresse.

Aufruf:

```
KMD>BREAKPOINT                                  <CR>
  BREAK-ADR1 = XXXX
  BREAK-ADR2 = XXXX
  BREAK-ADR3 = XXXX
  BREAK-ADR4 = XXXX

  EIN/AUS    = A E <CR> oder <SP>
                                                <CR>
KMD>GO
  START-ADR = XXXX E000 <CR>
  BREAK-ADR1= XXXX E030 <SP>
  BREAK-ADR2= XXXX E065 <CR>
  BREAK-ADR3= XXXX      <CR>
  BREAK-ADR4= XXXX      <CR>
```

Änderungen nur unter dem Go-Kommando möglich

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.

## 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 C2 02E0            JNZ   NEXT
E006 C3 4000            JMP   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  SP
 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  SP
 E003         JNZ   NEXT       FE 10100 05 07 FF FA 00 5D 80 FC0E
 E002 NEXT:   DCR   A
```

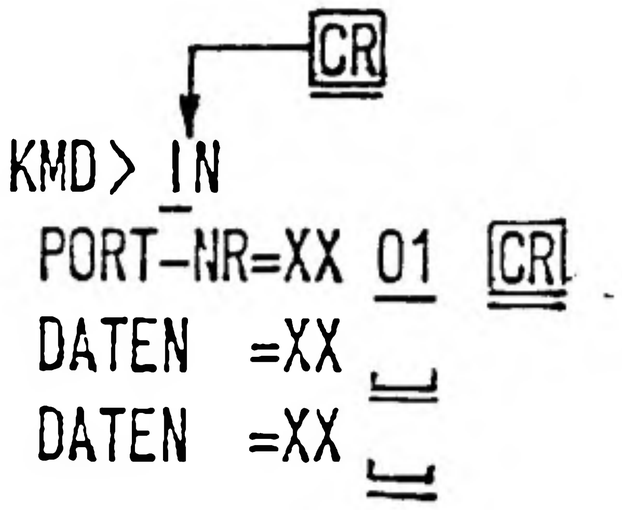
Nach dem Einschalten des `Trace-Interval`und Aufruf des G-Kommandos wird das Programm Befehl für Befehl durchlaufen und vom Tracer überprüft, ob der bearbeitete Programmteil im darzustellenden Adressbereich liegt. Ist dies der Fall,so werden, wie beim N-Kommando die Registerinhalte und die Befehle auf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




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

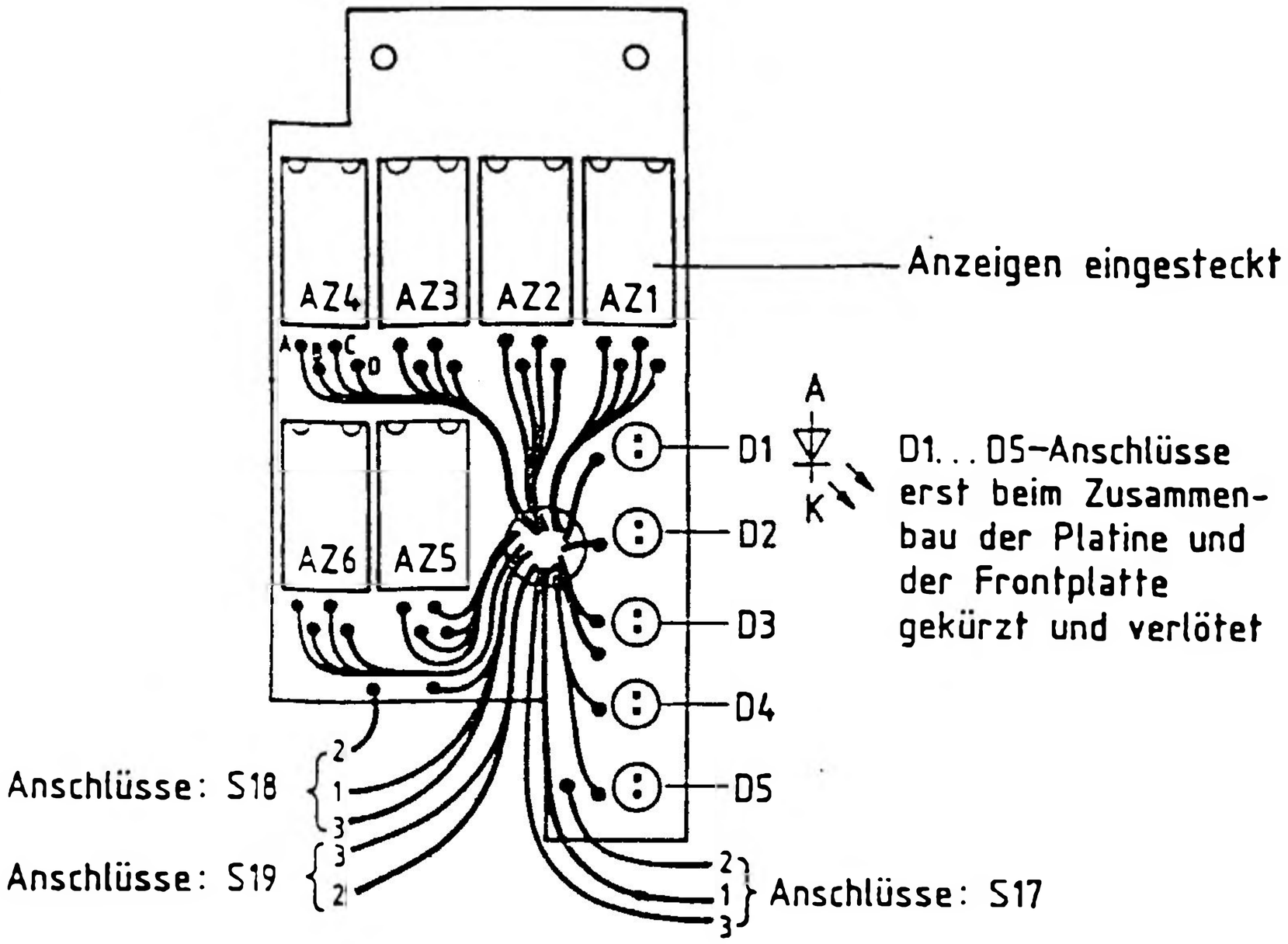
Mit dem LOAD-Kommando werden Daten vom Kassetten-Recorder in den Speicher zurückgelesen. Sollen die Daten nicht in den beim SAVE-Kommando angegebenen Speicherbereich übertragen werden, so kann eine neue Startadresse angegeben werden.

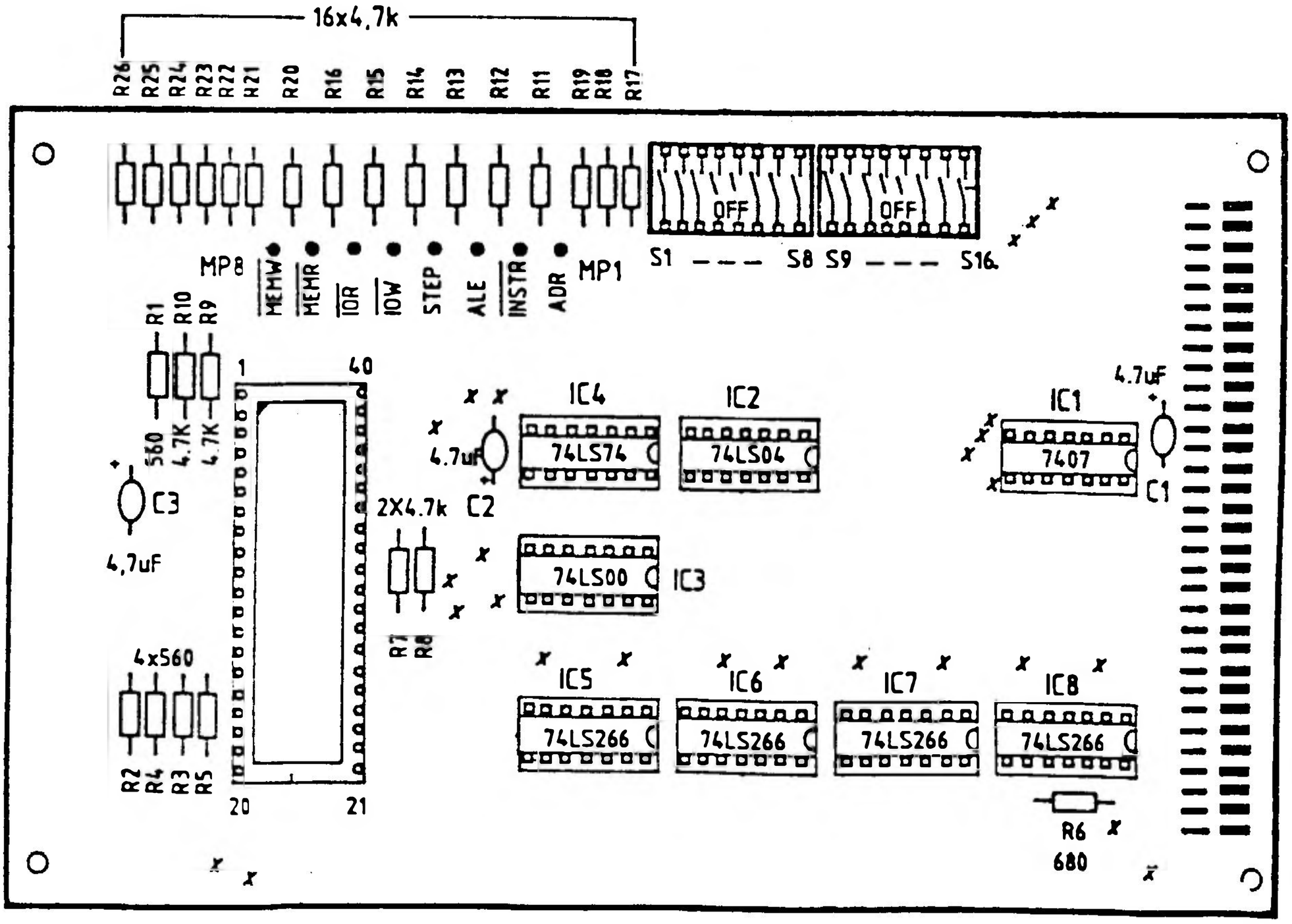
Aufruf: KMD LOAD TAPE CR
START-ADR = XXXX E800 CR
SPACE, DANN BAND EINSCHALTEN SP
(Kommandoausführung)
READY

KMD _

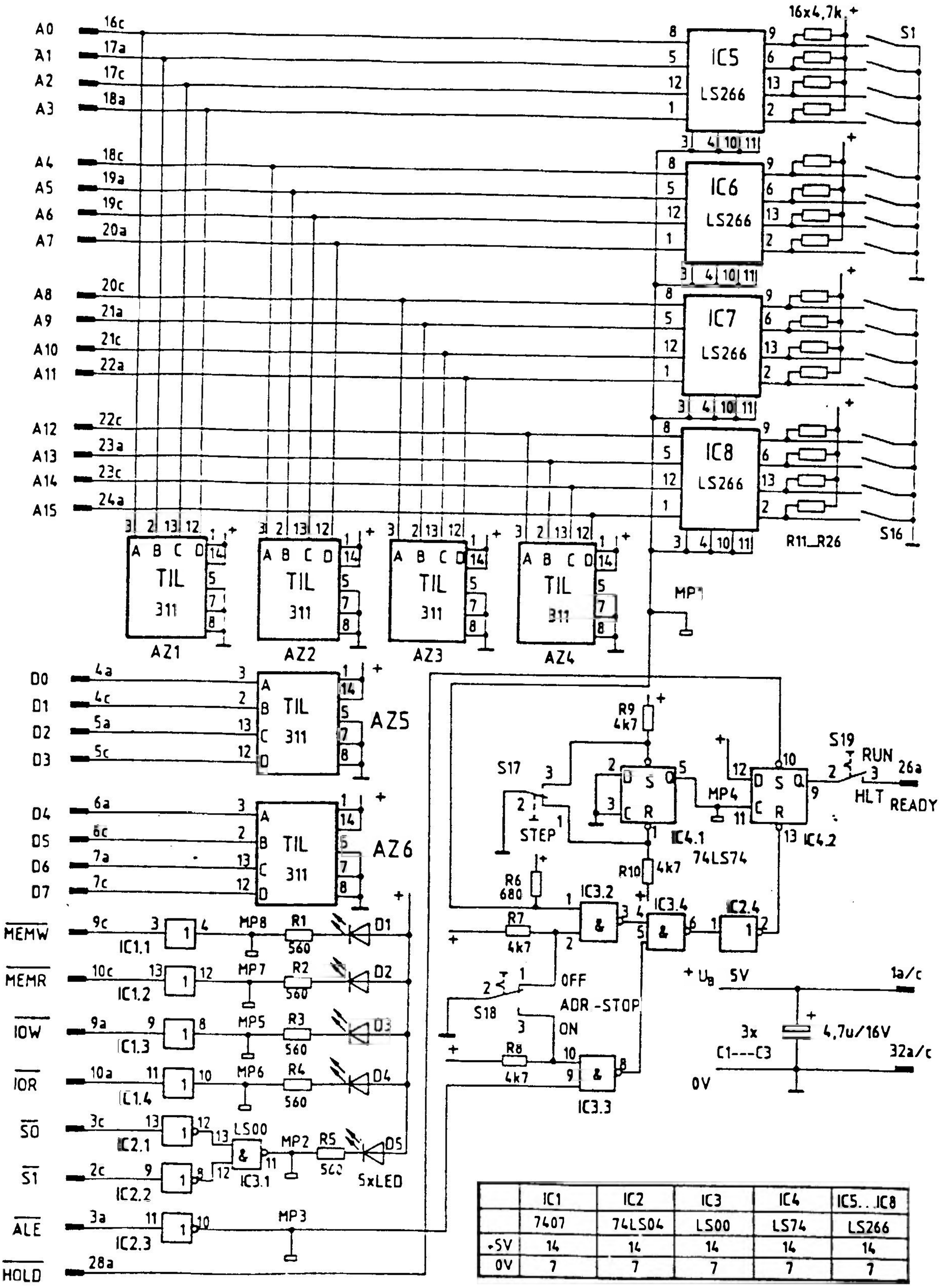
Mögliche Fehlermeldungen: *** NICHT HEX = 7F
*** CHECHSUM-ERROR

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



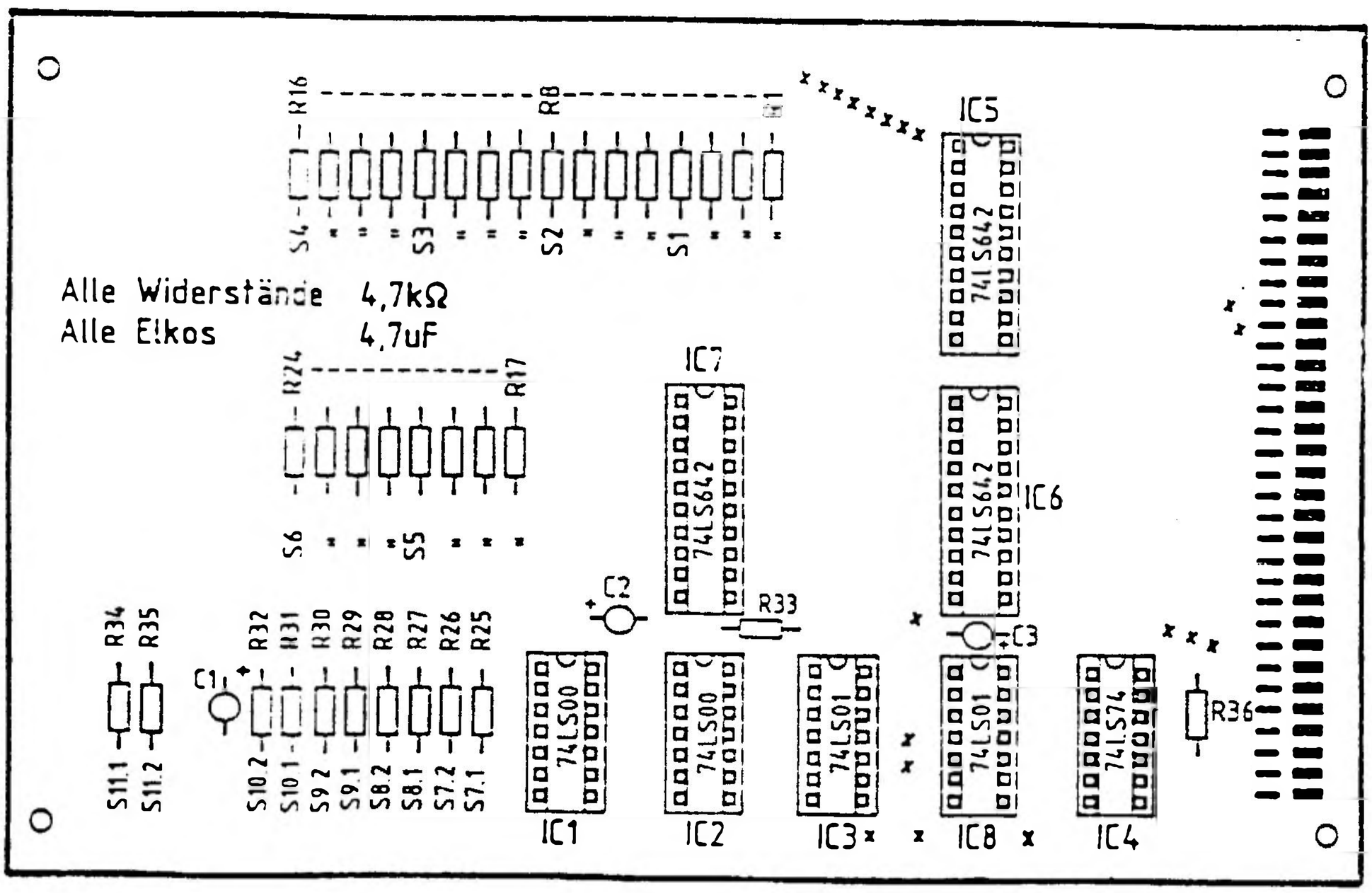

| | | |
|---|---|---|
| Bfx | | |
| | Bestückungsplan Bus-Signalanzeige | 56 |



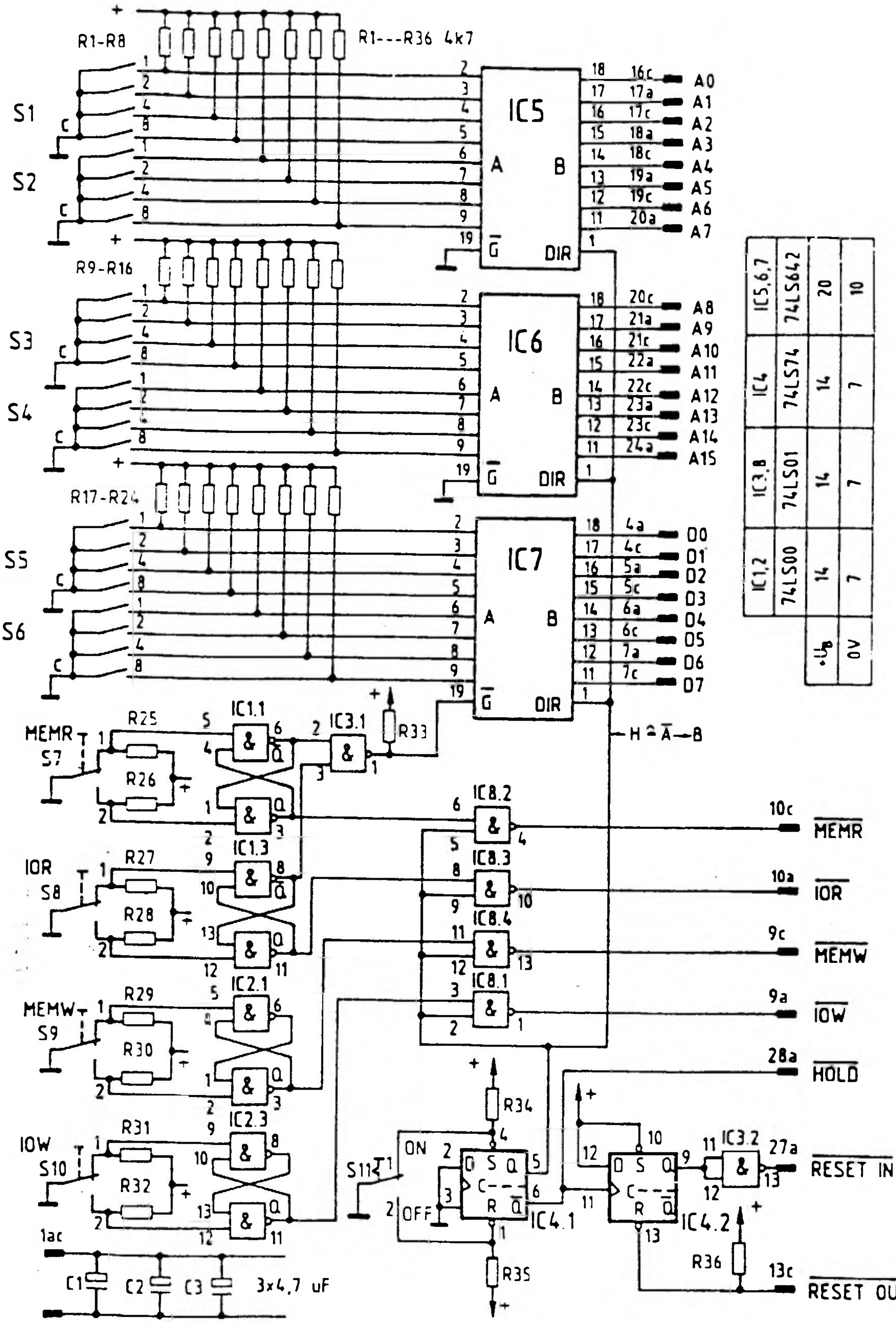

| | IC1 | IC2 | IC3 | IC4 | IC5.. IC8 |
|---|---|---|---|---|---|
| | 7407 | 74LS04 | LS00 | LS74 | LS266 |
| +5V | 14 | 14 | 14 | 14 | 14 |
| 0V | 7 | 7 | 7 | 7 | 7 |

Bfz

Stromlaufplan Bus- Signalanzeige

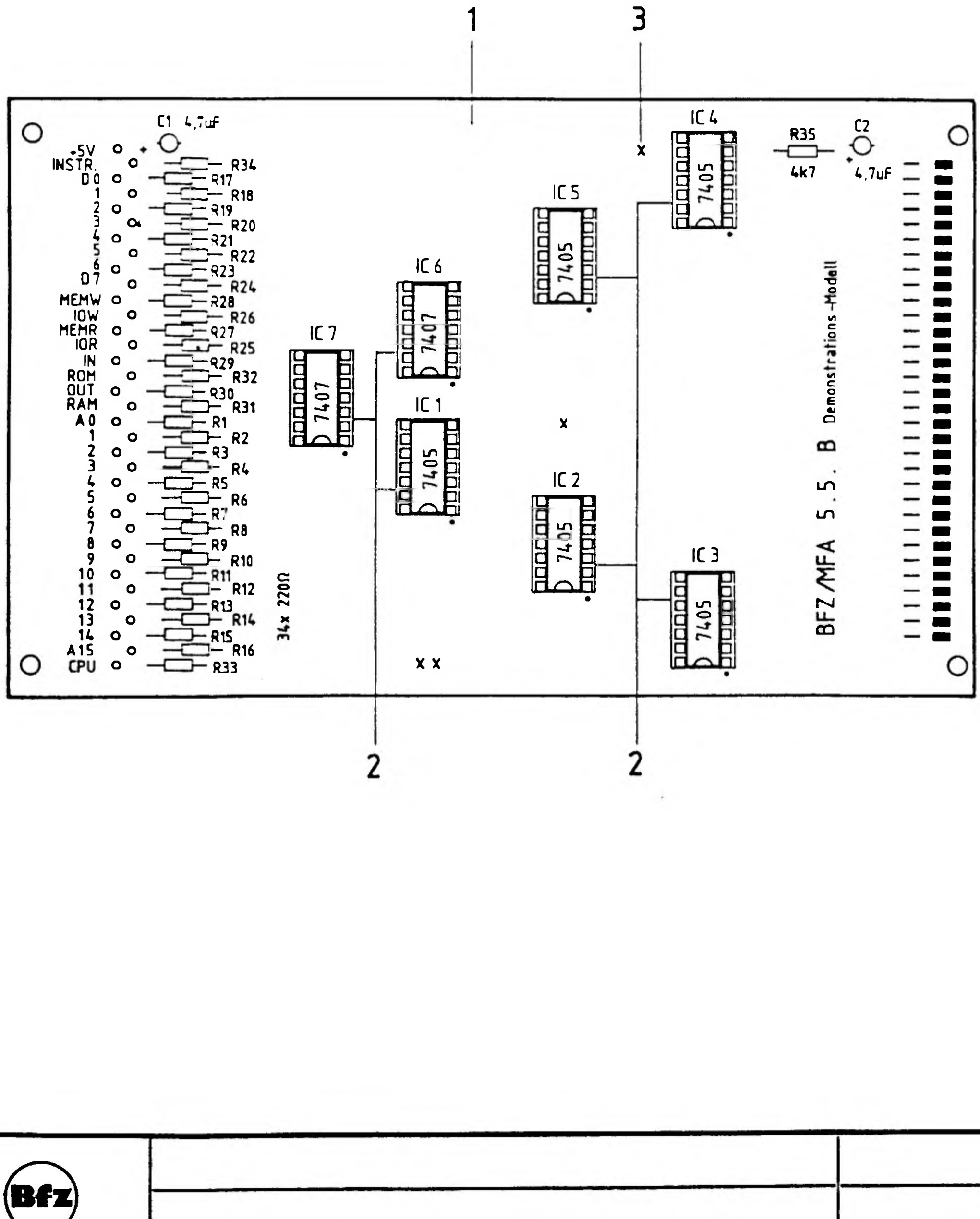


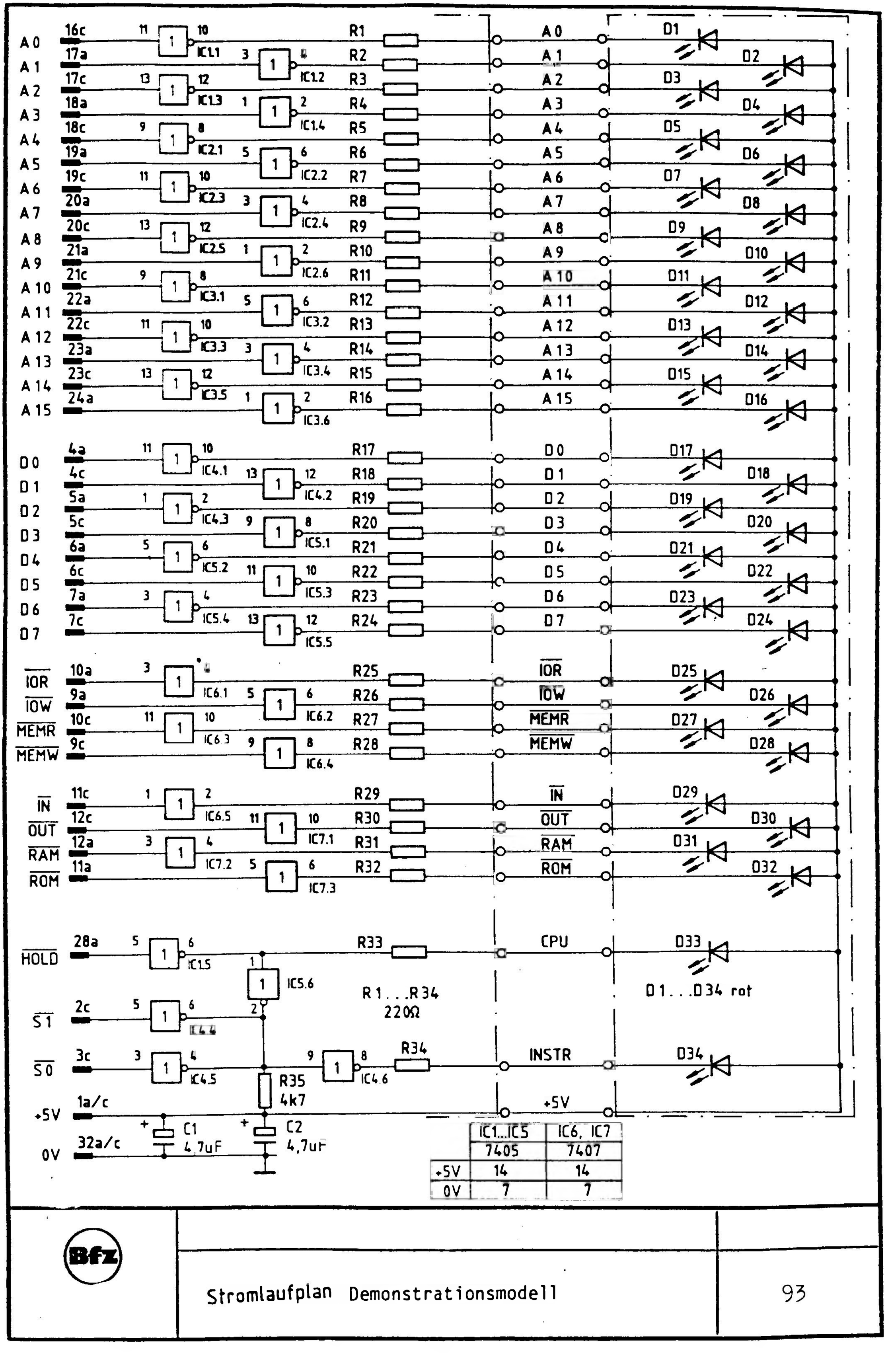

| | | |
|---|---|---|
| Bfz | Bestückungsplan Bus-Signalgeber | 54 |




| | | |
|---|---|---|
| Bfz | Stromlaufplan "Bus-Signalgeber" | 55 |

Bestückungsplan



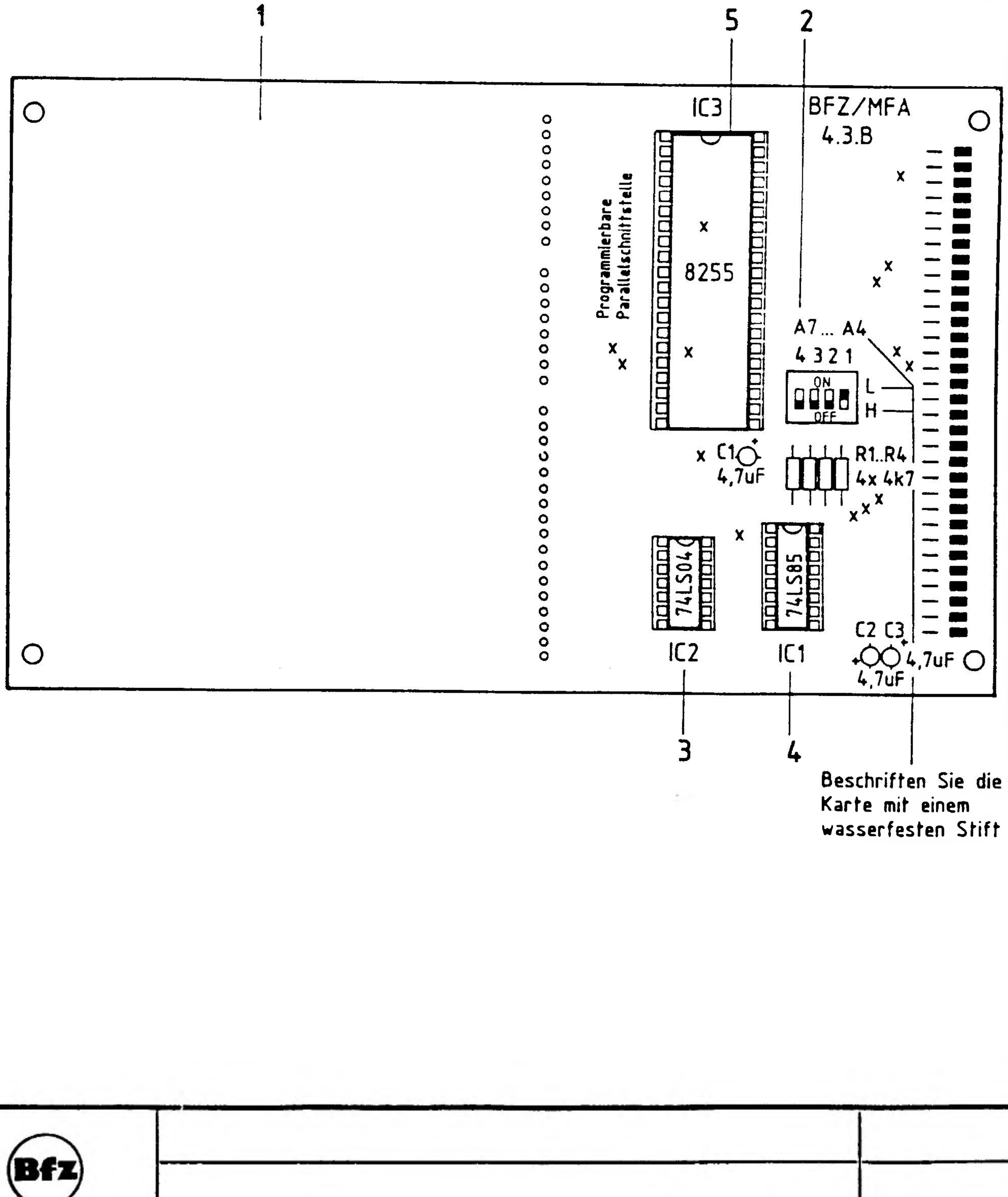


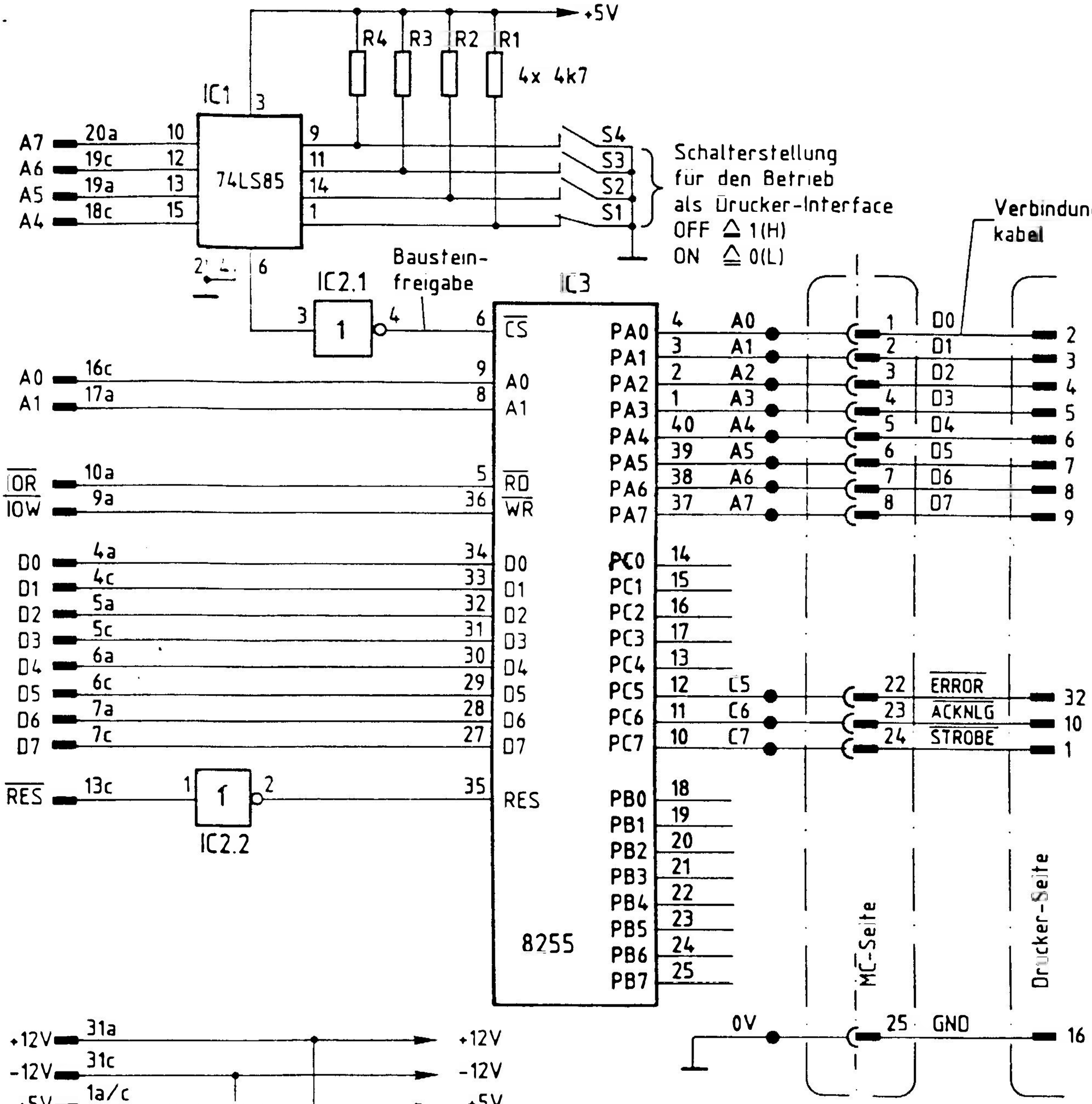

Bfz

Stromlaufplan Demonstrationsmodell

93

Bestückungsplan






| | IC1 | IC2 | IC3 |
|---|---|---|---|
| | 74LS85 | 74LS04 | 8255 |
| +5V | 16 | 14 | 26 |
| 0V | 8 | 7 | 7 |

## Teil 3 MAT 85 Eigenheiten.Tabellen.Symbole

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 Programmbearbeitung 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 Anwender-Programm 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<br>- Breakpoint's eingeschaltet sind und<br>- die Breakpoint-Adressen nicht alle 0<br>- das Anwenderprogramm eine der eingestellten Breakpoint-Adressen erreicht.<br><br>Hinweis: Der Registerausdruck gibt den Inhalt der Register nach der Ausführung des Befehls vor dem Breakpoint wieder.<br>Steht ein RST 4 im Programm, ohne das die Breakpoint's eingeschaltet und eine Adresse eingestellt ist, so erfolgt ein Sprung in die Breakpoint-Fehlerroutine. | ***BREAK-POINT*** und Register-ausdruck<br><br>***BREAK-POINT-ERROR*** und Register-ausdruck |
| 0024 | TRAP, Neustart des Monitorprogramms mit der erneuten Erfassung der Baud-rate. Adresse 0860 | ***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: FC0F/FC E0 |

Anmerkung: Übertragungsraten

0156: 1200
0041C: 2400
0048: 4800

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.

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 [ESC] 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.<br>Beispiel:<br>CALL 0055<br>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.<br>Beispiel: MVI A,23 ; 23 Hexadezimal<br>CALL 005E ; wird als 00100011 ausgegeben |
| WADEZ | 0061 | A | Gibt den Akku-Inhalt (8Bit) als Dezimalzahl auf dem Bildschirm/Drucker aus.<br>MVI A,23 ;23 Hexadezimal wird<br>CALL 0061 ;als 35 ausgegeben |

| 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:<br>0 → ASCII-Zeichen<br>1 → Binärzahl<br>2 → Dezimalzahl<br>3 → Hexadezimalzahl<br>Beispiel:<br>`MVI A,23 ; 23 Hexadezimal wird`<br>`MVI C,1 ; als 00100011 ausge-`<br>`CALL 0064 ; geben` |
| 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.<br>Beispiel:<br>`CALL 006D`<br>`DB ' DIES IST EINE UEBUNG', 0`<br>Gibt den Text DIES IST EINE UEBUNG aus. |
| WCRLFI | 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 [CR] oder [SP] abgeschlossen werden. |
| BSTIME | 0895 | A,D,E | Zeitverzögerung von 0,24 s.<br>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:<br><br>PUSH PSW ;Registerinhalte A,D,E<br>PUSH D ;retten<br>CALL 0895 ;Zeitverzögerung<br>POP D ;Registerinhalte wiederher-<br>POP PSW ;stellen |
| CMP2 | 0EA8 | A | Vergleicht die Inhalte der Register DE mit denen der Register HL.<br>Wenn (HL)>(DE) ist, wird das Carry-Flag auf 1 gesetzt, sonst auf 0.<br>Die zu vergleichenden Inhalte müssen vor Aufruf des Unterprogramms in die Doppelregister D und H geladen werden.<br>Beispiel: LXI D, Zahl 1<br>LXI H, Zahl 2<br>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.<br>[(HL) = (HL) - (DE)] |
| WBUF | 0BA1 | | 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.<br>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                         ;Drucker "AUS"
            START-ADR = 1E03

Ausdruck: *** PRINTER OFF ***
```

EPROM-Programmierroutine:

```
Aufruf:    KMD > GO
            START-ADR = 1E06

Ausdruck: *** EPROM-PROGRAMMER ***

mögliche
Kommamdos  { 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 Speicherbereiches vergleichen.

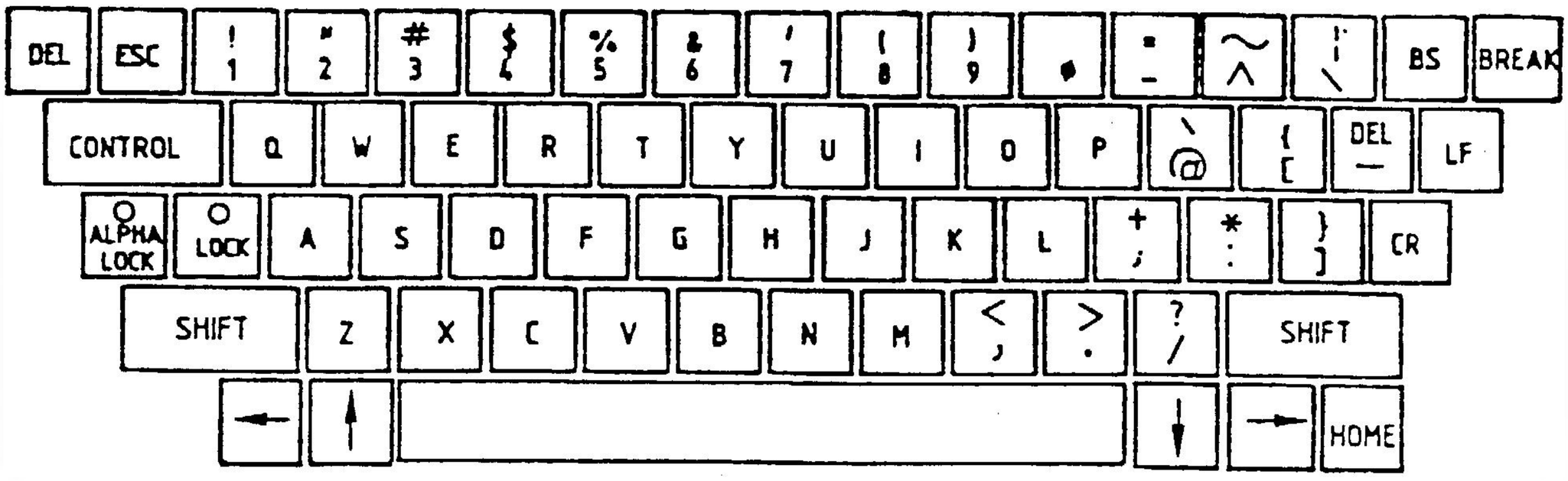
QUIT: Rückkehr in das Monitorprogramm.

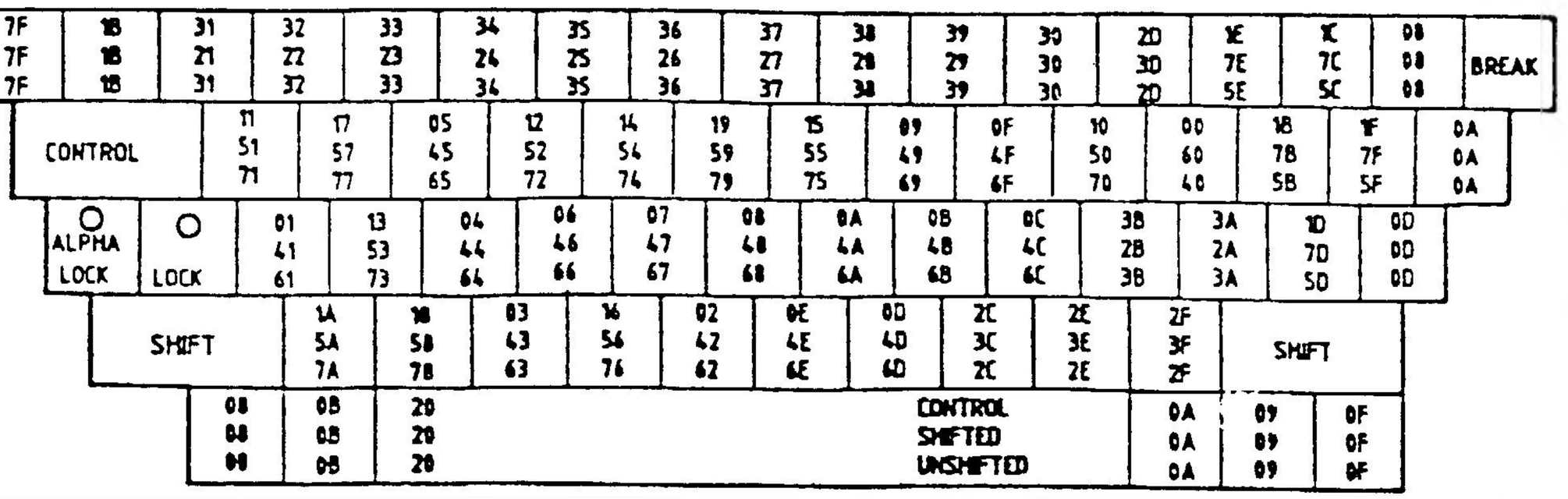
```
KMD > GO
 START-ADR =1E09 1E06

*** 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)

| B6 B4 \ B3 B0 | 0 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

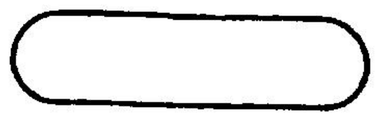
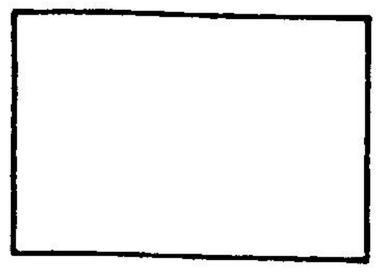
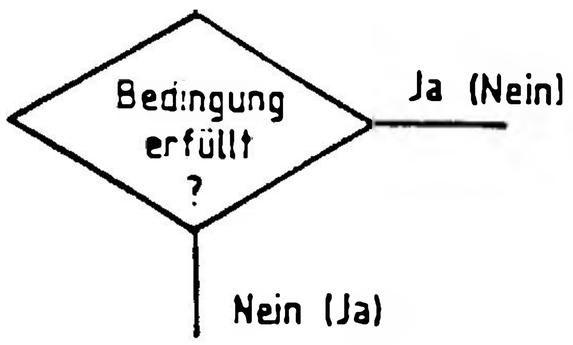
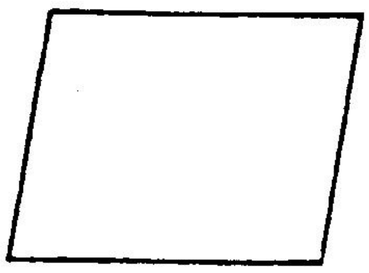
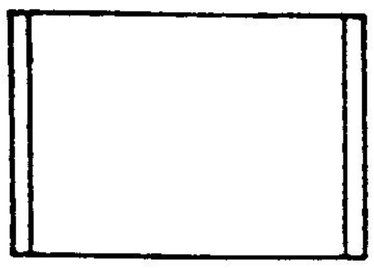
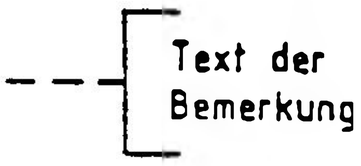
Beschriftung der Tasten und hexadezimale Verschlüsselung der Tastenfunktion



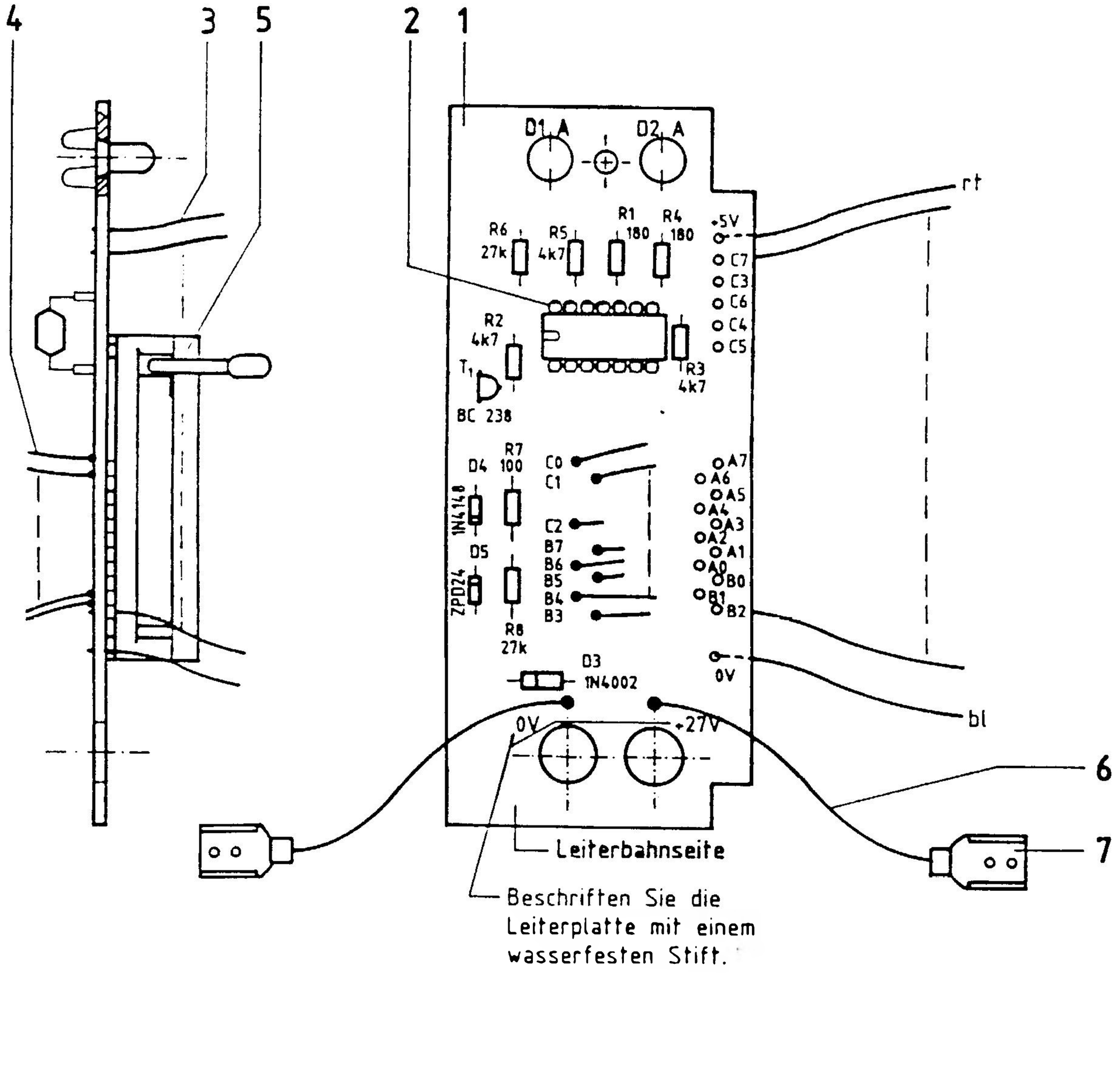



Die Bedeutung der Steuerzeichen (00H-20H und 7FH)

| Code Hex | ASCII | Bedeutung | Steuerung des Cursors im Video-Interface BFZ/MFA 8.2. |
|---|---|---|---|
| 00 | NUL | Null, Nichts | |
| 01 | SOH | Kopfzellenbeginn | |
| 02 | STX | Textanfangszeichen | |
| 03 | ETX | Textendezeichen | |
| 04 | EOT | Ende der Obertragung | |
| 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 | Rückschaltungszeichen | |
| 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 | |

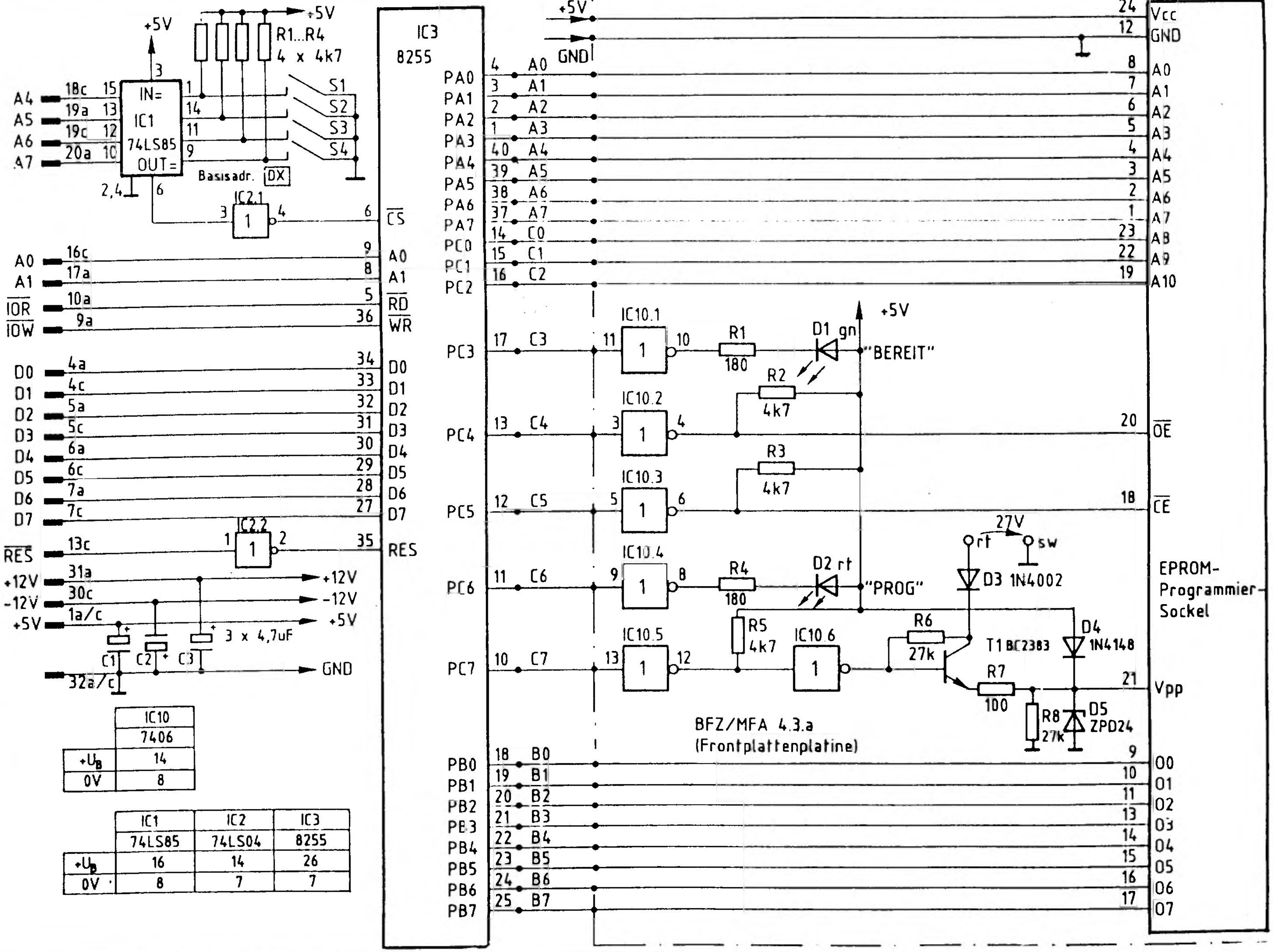
| Symbole | Bedeutung |
|---|---|
|  | Grenzstelle (Beginn, Ende ...) |
|  | allgemeine Operation |
|   | Verzweigung |
|  | Ein-/Ausgabe- Operation |
|  | Unterprogramm |
|  | Flußlinie (Richtung der Abarbeitung) |
|  | Zusammenführung |
|   | Bemerkung |
|  | Nahtstelle |

Bestückungsplan



BfZ

Stromlaufplan EPROM-Programmierer



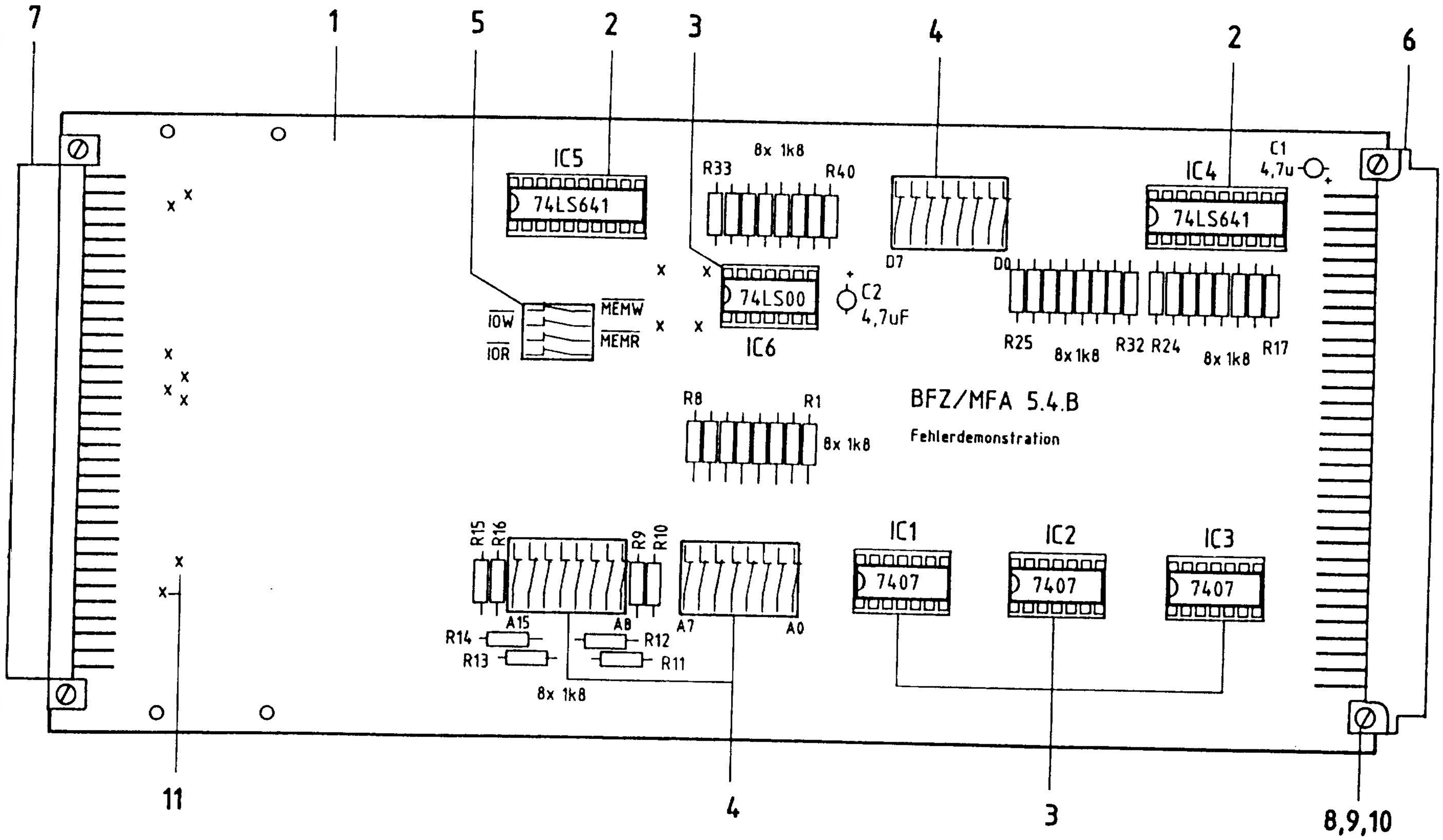

| | IC10 |
|---|---|
| | 7406 |
| $+U_B$ | 14 |
| 0V | 8 |

| | IC1 | IC2 | IC3 |
|---|---|---|---|
| | 74LS85 | 74LS04 | 8255 |
| $+U_B$ | 16 | 14 | 26 |
| 0V | 8 | 7 | 7 |

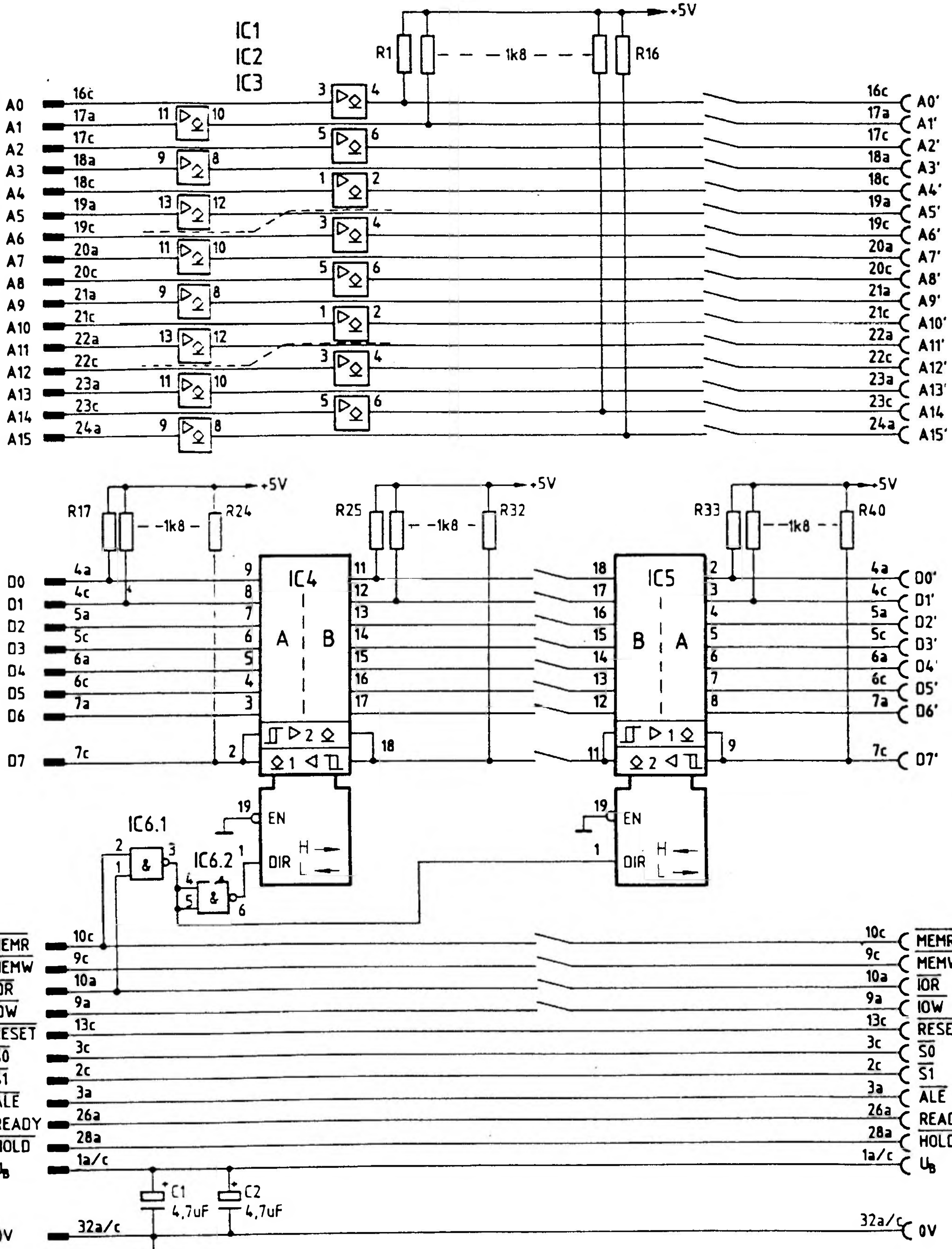
BfZ

Fehlersimulation

06




Bestückungsplan

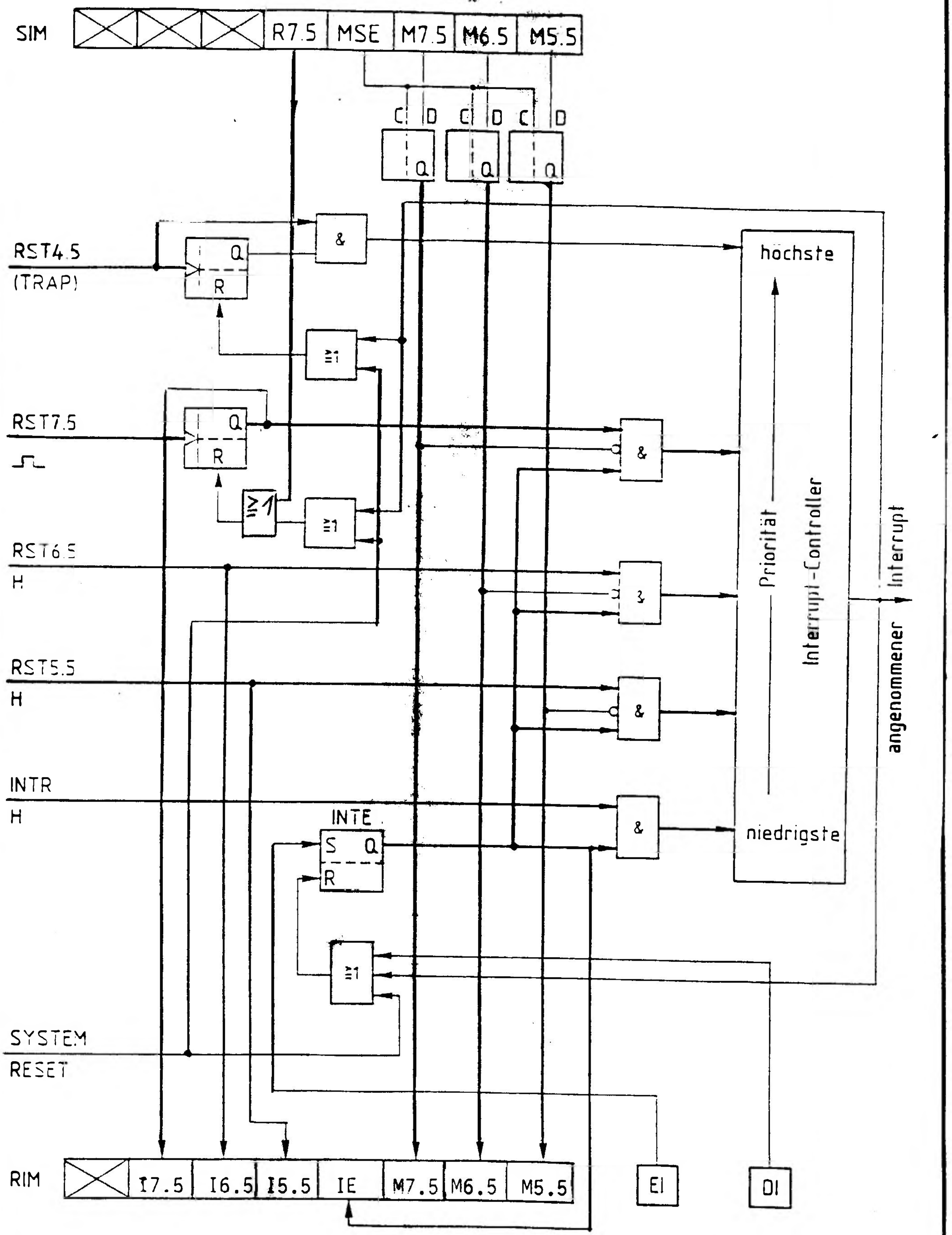



| | IC1 ... IC3 | IC4, IC5 | IC6 |
|---|---|---|---|
| | 7407 | 74LS641 | 74LS00 |
| $+U_B$ | 14 | 20 | 14 |
| 0V | 7 | 10 | 7 |

Bfz

Stromlaufplan Fehlersimulation

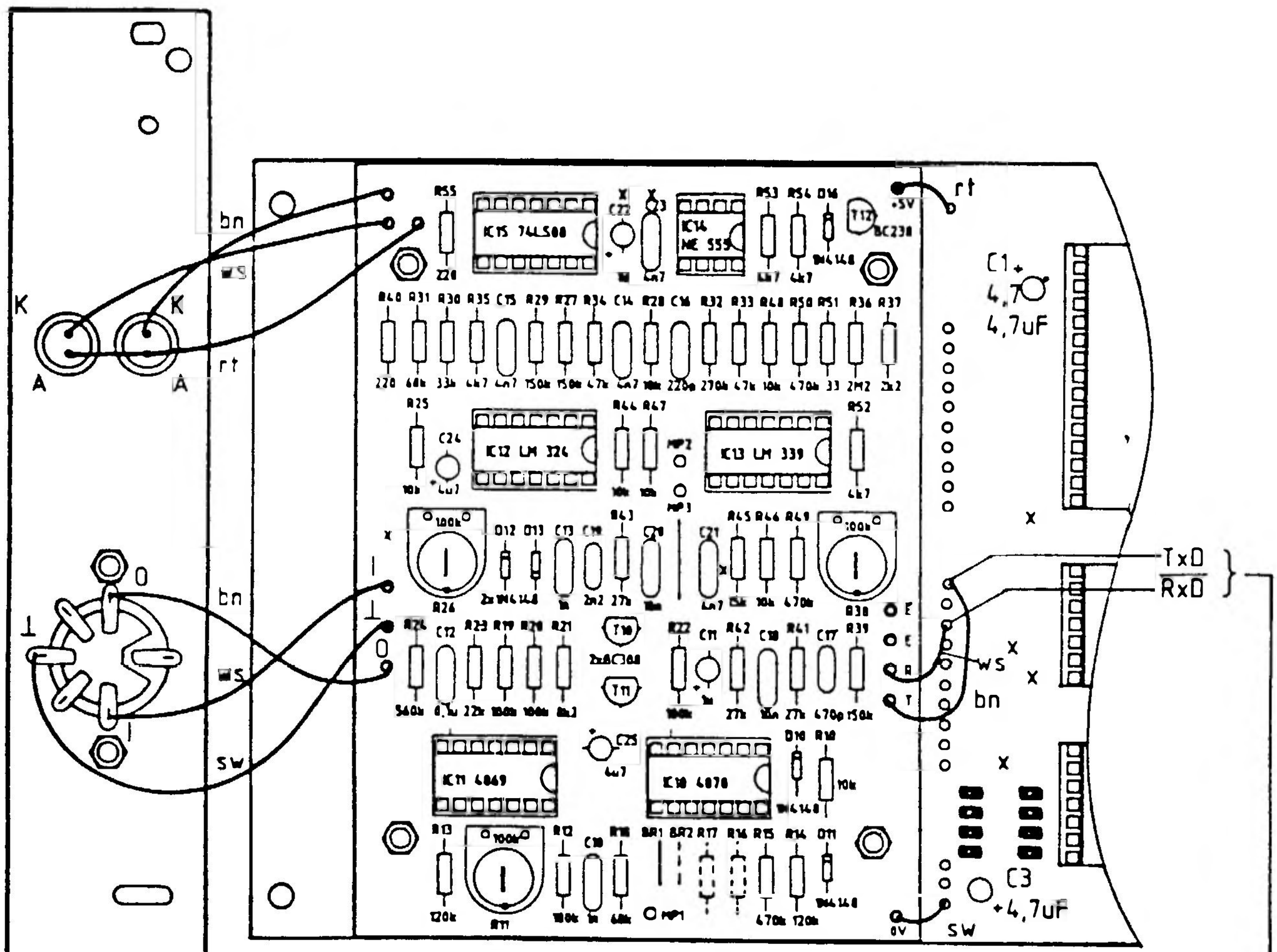
91



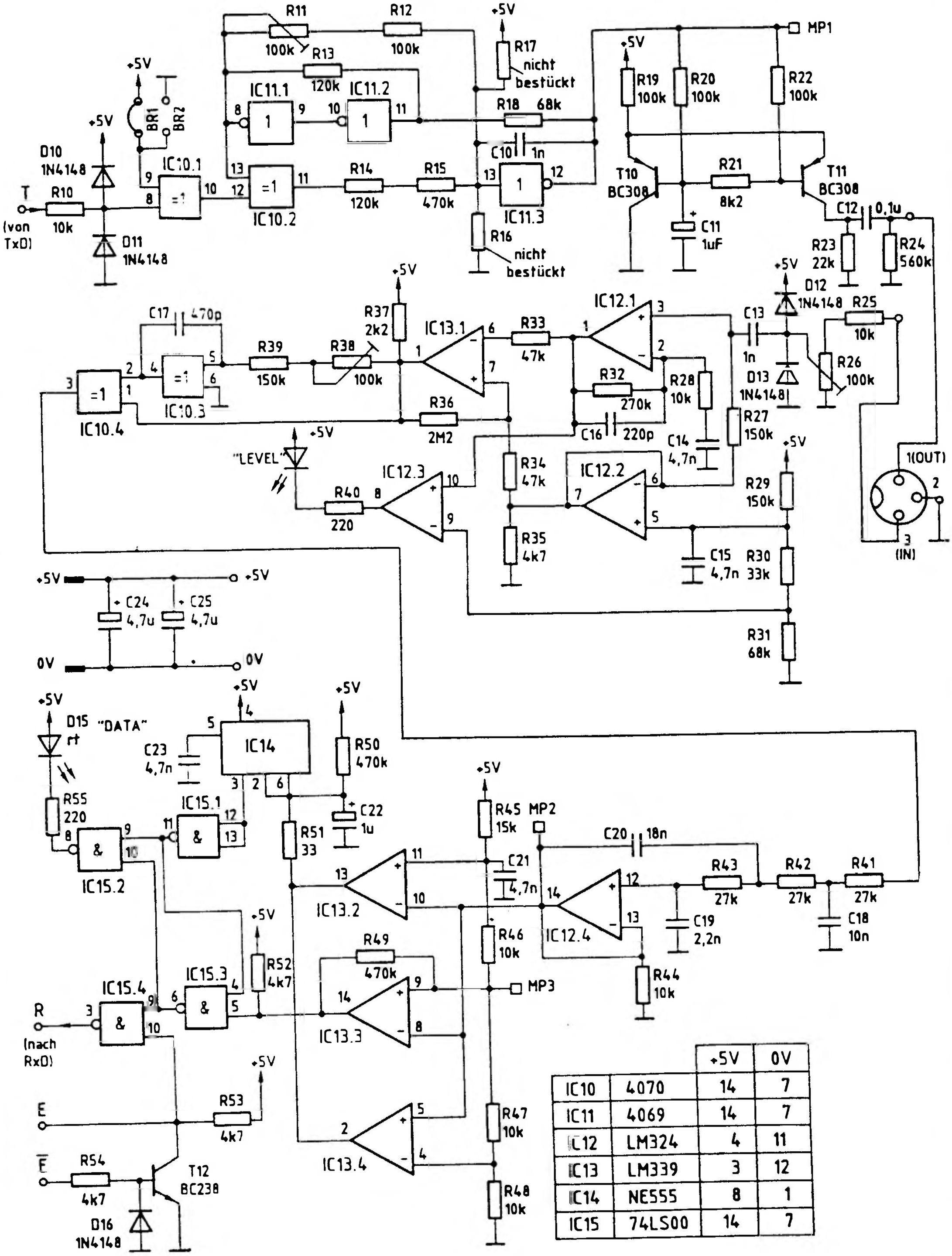

Bfz

Vollständige Interrupt-Steuerung 8085

95



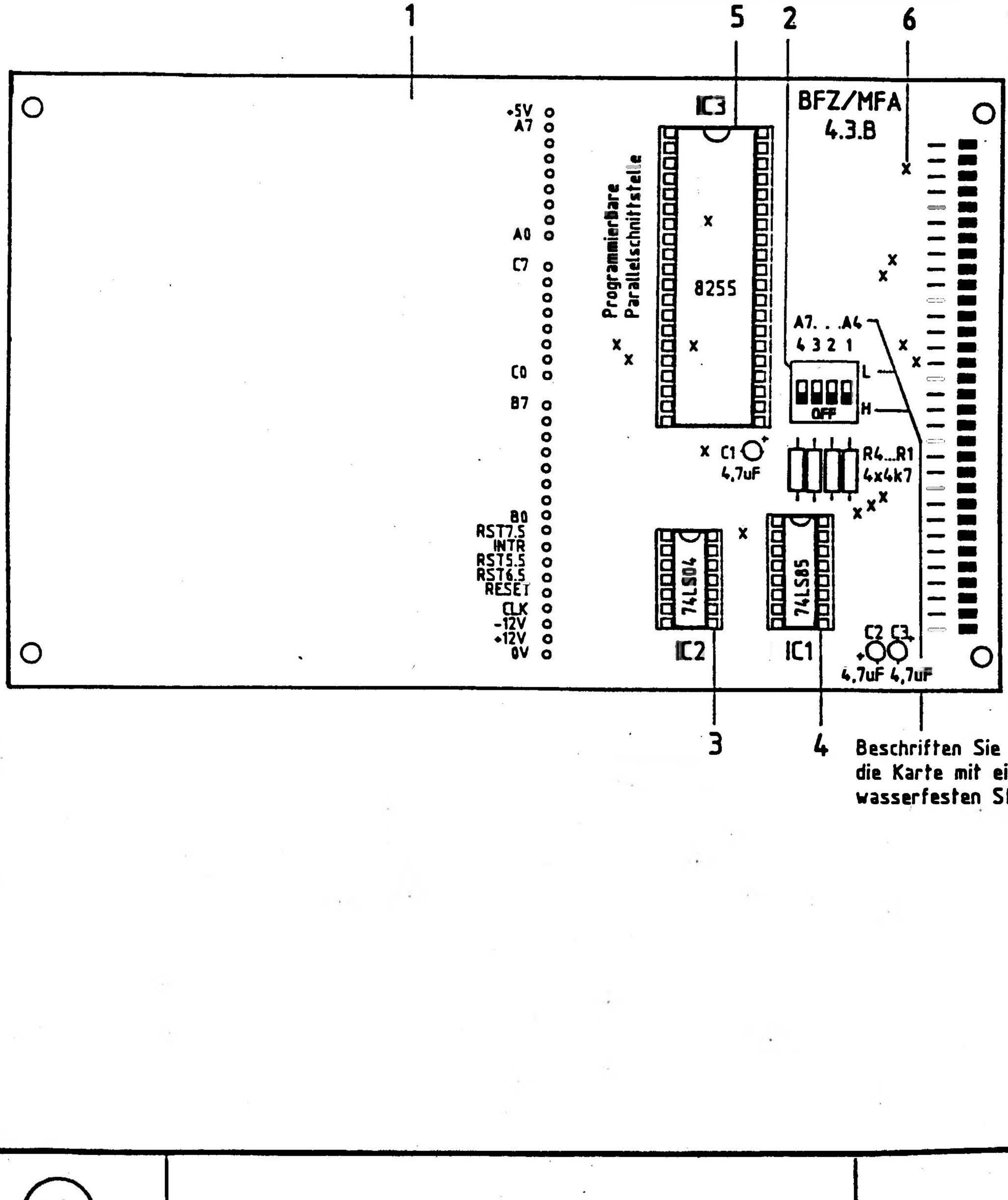

Die beiden Anschlüsse TxD und RxD noch nicht einlöten. Dies geschieht erst bei der Inbetriebnahme.




| | | +5V | 0V |
|---|---|---|---|
| IC10 | 4070 | 14 | 7 |
| IC11 | 4069 | 14 | 7 |
| IC12 | LM324 | 4 | 11 |
| IC13 | LM339 | 3 | 12 |
| IC14 | NE555 | 8 | 1 |
| IC15 | 74LS00 | 14 | 7 |

| Stift-Nr. | Reihe a | Reihe c | Bemerkung |
|---|---|---|---|
| 1 | +5V | +5V | |
| 2 | $\overline{\text{CLKOUT}}$ | $\overline{\text{S1}}$ | |
| 3 | $\overline{\text{ALE}}$ | $\overline{\text{SO}}$ | |
| 4 | DO | D1 | |
| 5 | D2 | D3 | |
| 6 | D4 | D5 | |
| 7 | D6 | D7 | |
| 8 | $\overline{\text{HLDA}}$ | $\overline{\text{INTA}}$ | |
| 9 | $\overline{\text{IOW}}$ | $\overline{\text{MEMW}}$ | |
| 10 | $\overline{\text{IOR}}$ | $\overline{\text{MEMR}}$ | |
| 11 | $\overline{\text{RCM}}$ | $\overline{\text{IN}}$ | |
| 12 | $\overline{\text{RAM}}$ | $\overline{\text{OUT}}$ | |
| 13 | TRAP | $\overline{\text{RESOUT}}$ | |
| 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 | $\overline{\text{RESIN}}$ | RST6,5 | |
| 28 | $\overline{\text{HCLD}}$ | | |
| 29 | PULS OUT. | Rx | |
| 30 | PULS F.INT. | Tx | |
| 31 | +12V | -12V | |
| 32 | GND | GND | |

## Bestückungsplan



| | IC1 | IC2 | IC3 |
|---|---|---|---|
| | 74LS85 | 74LS04 | 8255 |
| +5V | 16 | 14 | 26 |
| 0V | 8 | 7 | 7 |

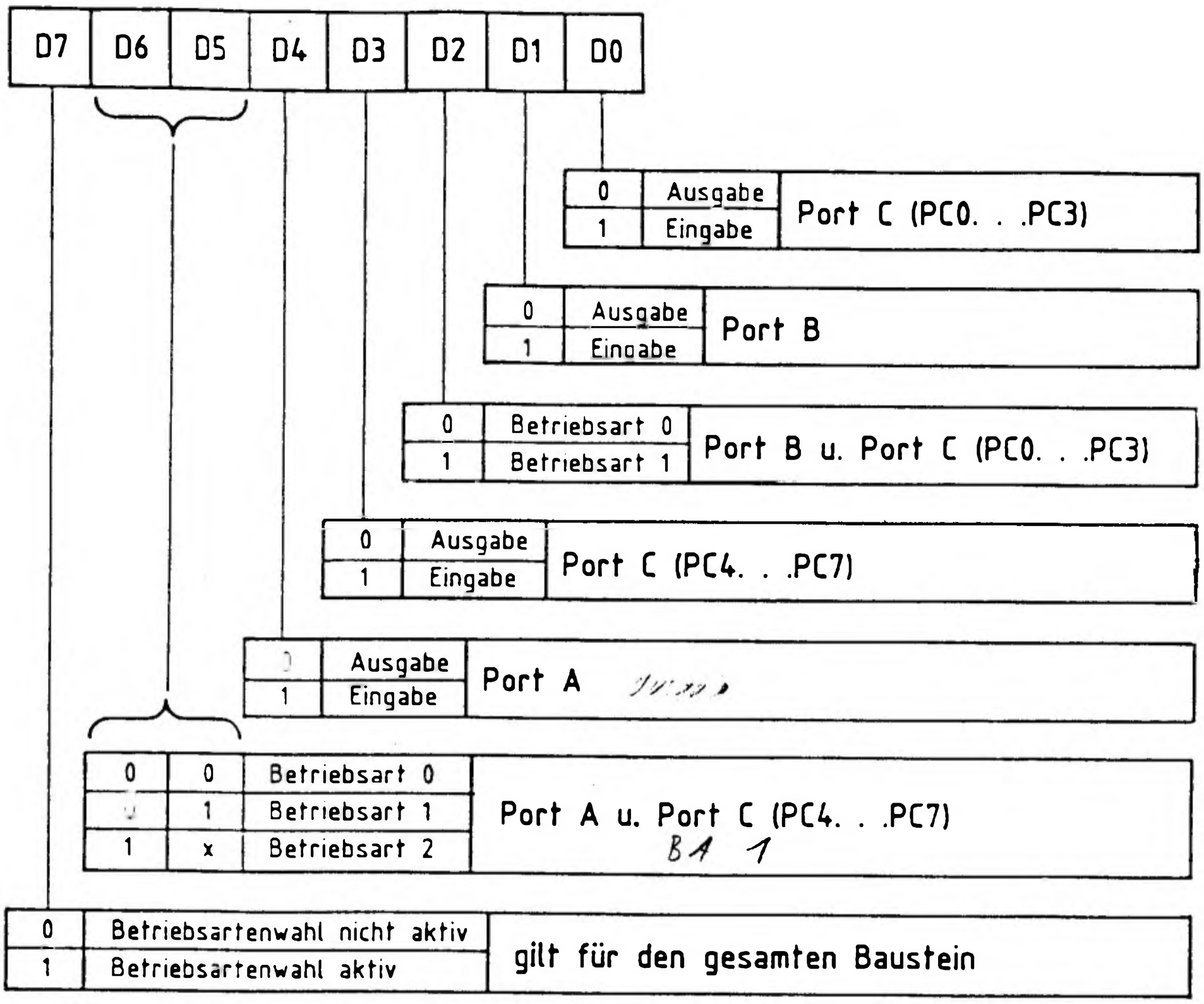
OFF ≙ 1 (H)
ON ≙ 0 (L)

+5V
R4 R3 R2 R1
4x4k7
IC1
74LS85
A7 20a
A6 19c
A5 19a
A4 18c
S4 S3 S2 S1
Bausteinfreigabe
IC2.1
IC3
A0 16c
A1 17a
IOR 10a
IOW 9a
D0 4a
D1 4c
D2 5a
D3 5c
D4 6a
D5 6c
D6 7a
D7 7c
RES 13c
IC2.2
CS
A0
A1
RD
WR
D0
D1
D2
D3
D4
D5
D6
D7
RES
8255
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7
PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7
Port A
Port C
Port B
RESET
CLK
IC2.3
CLK 2a
INTR 25c
RST5.5 26a
RST6.5 27c
RST7.5 25a
+12V 31a
-12V 31c
+ 5V 1a/c
0V 32a/c
C1 C2 C3
3x4,7uF/35V
INTR
RST5.5
RST6.5
RST7.5
+12V
-12V
+ 5V
0V

Bfz

Stromlaufplan "Programmierbare Parallelschnittstelle"

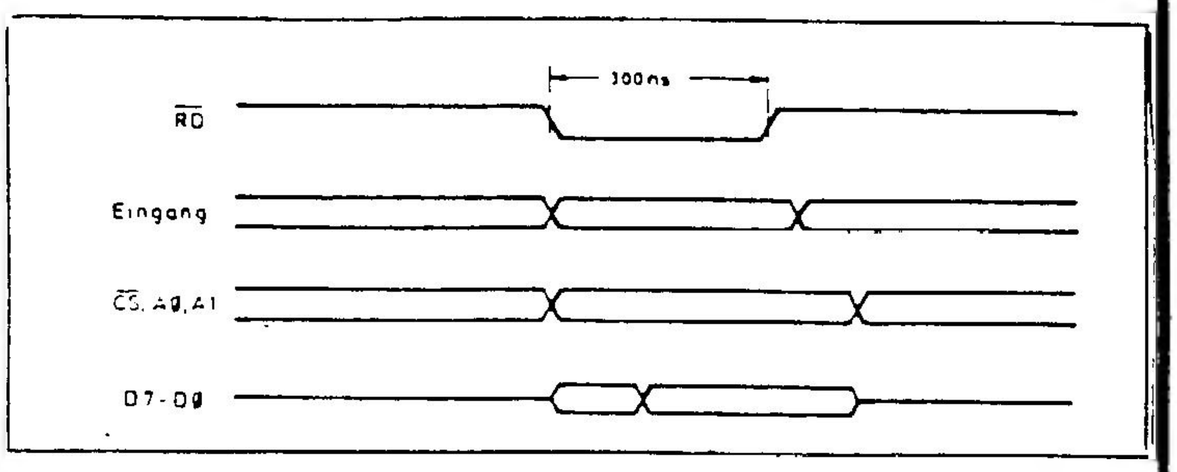
65



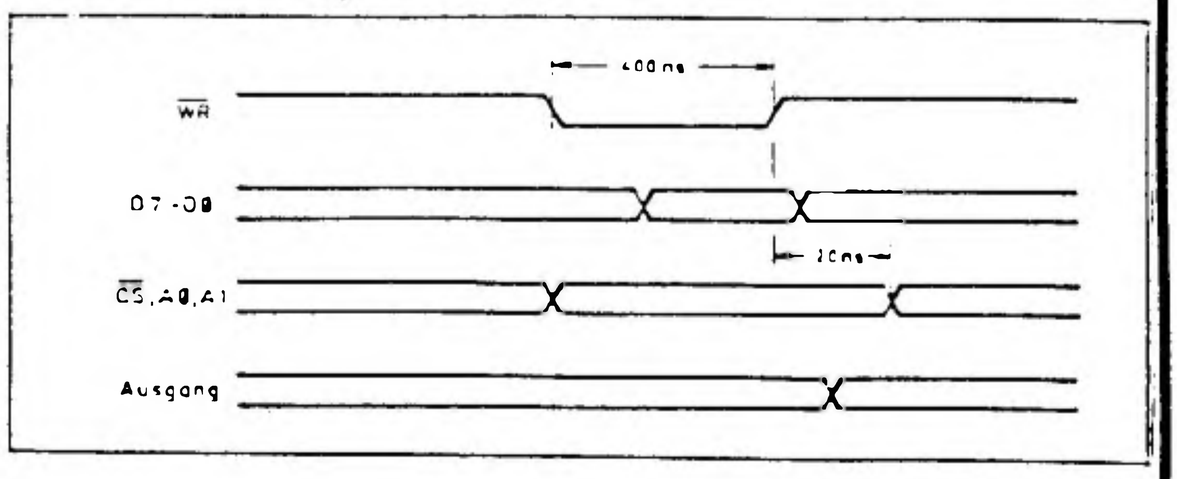

Steuerworte in der Betriebsart 0:

| 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:

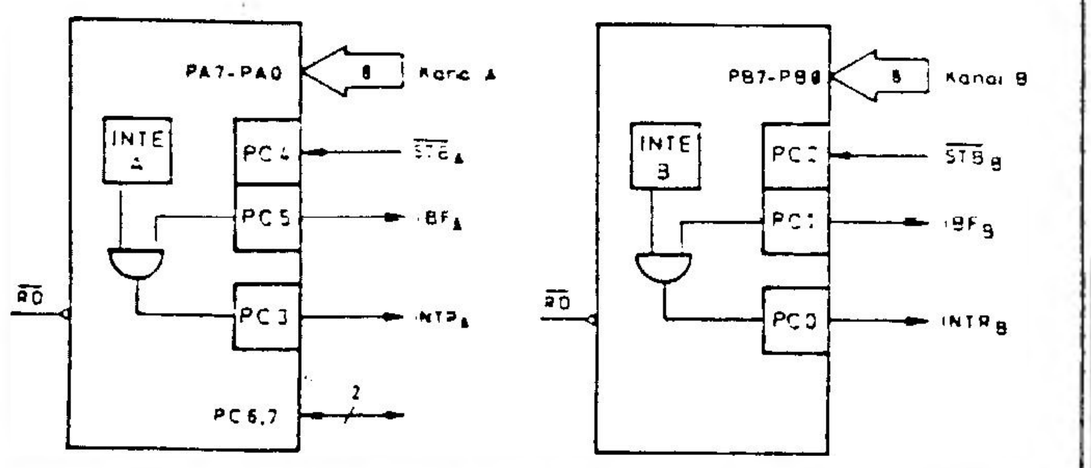
Kanal A

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | • | X | X | X |

Kanal B

| 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




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

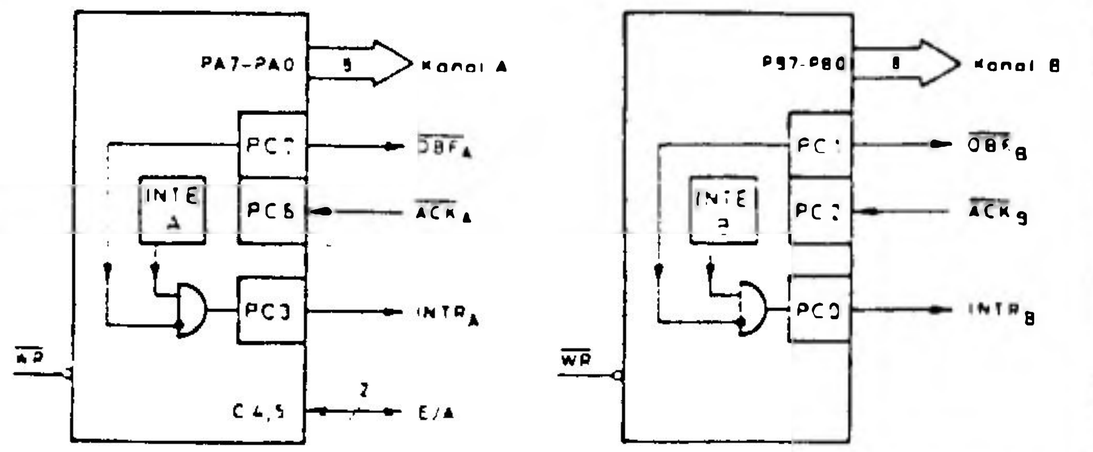
Kanal A

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | • | X | X | X |

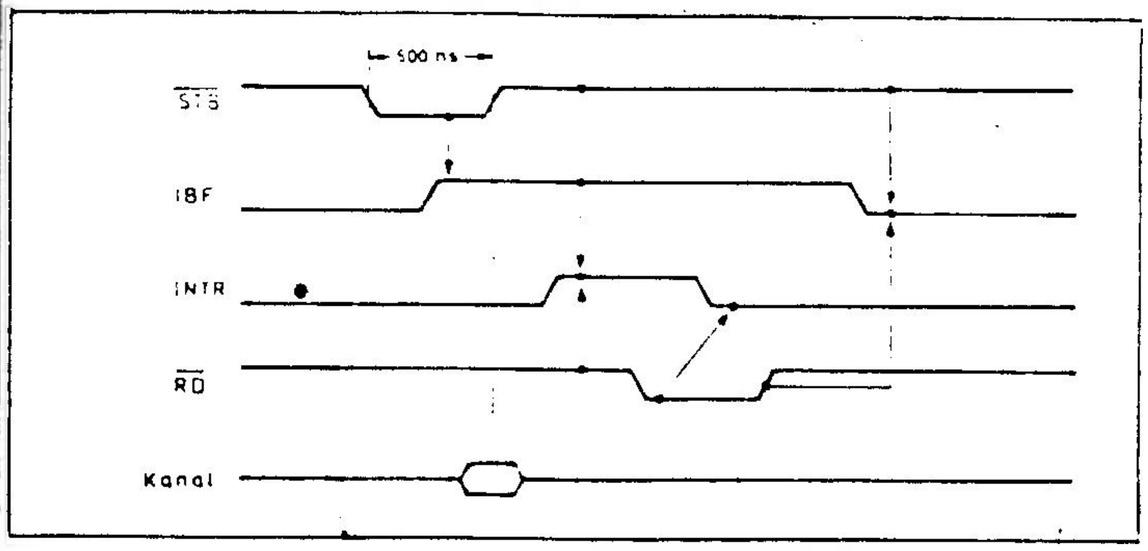
Kanal B

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

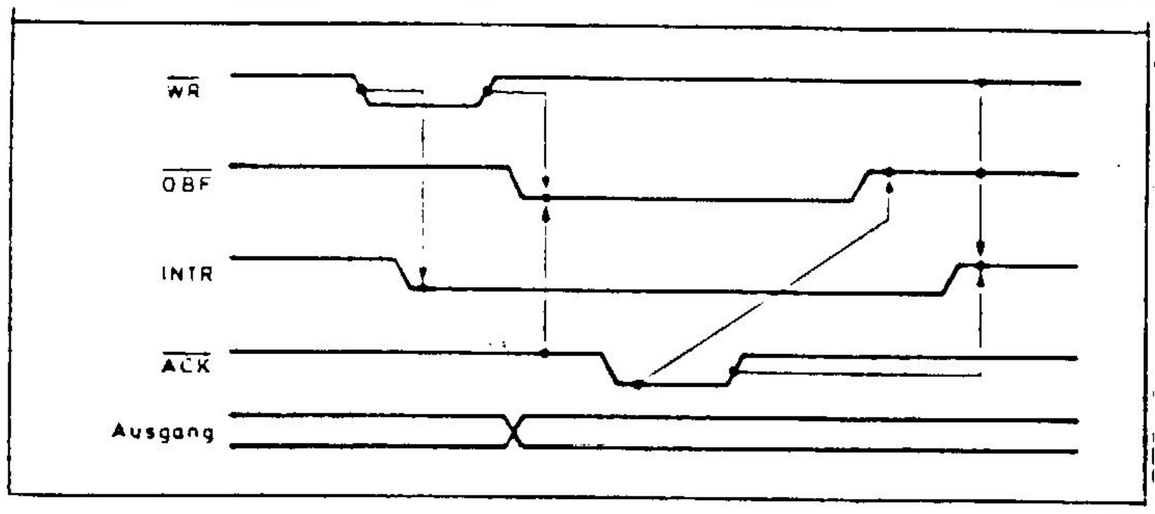
• C5, C4
1 ≙ Eingang
0 ≙ Ausgang über Bit setzen/rücksetzen




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 |
|---|---|---|---|---|---|---|---|
| $\overline{OBFA}$ | INTEA | E/A | E/A | INTRA | INTEB | $\overline{OBFB}$ | INTRB |

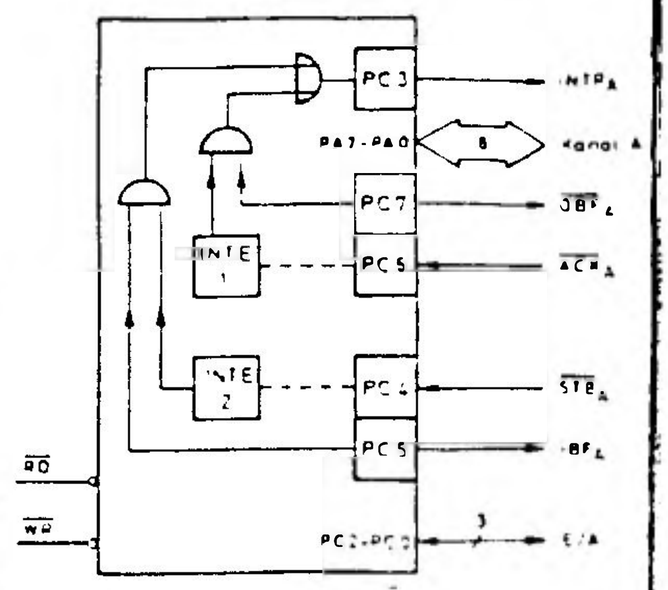
Steuerwort für die Betriebsart 2:

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | X | X | X | • | • | • |

C2—C0
1 ≙ Eingang
0 ≙ Ausgang

Kanal B
1 ≙ Eingang
0 ≙ Ausgang

Betriebsart Gruppe B
0 ≙ Betriebsart 0
1 ≙ Betriebsart 1

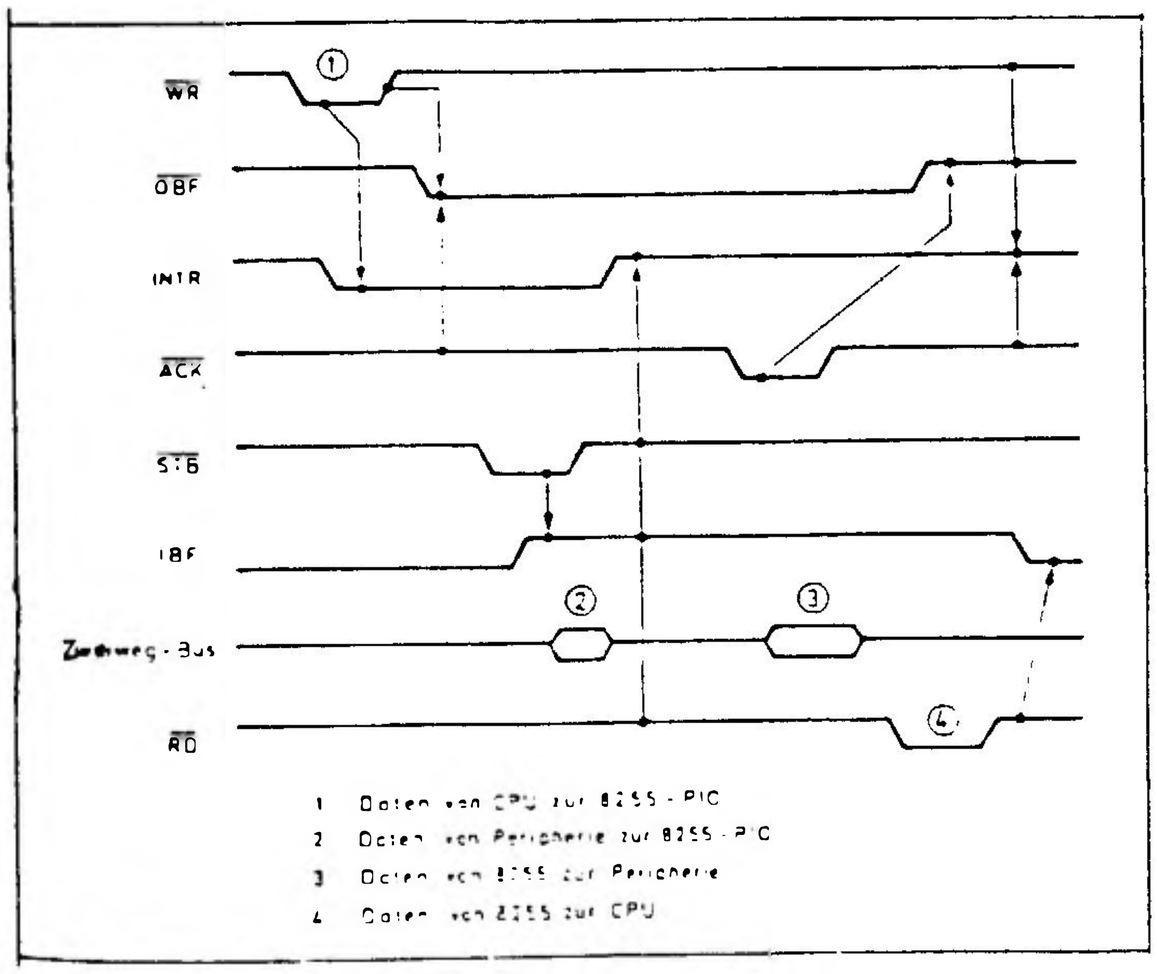


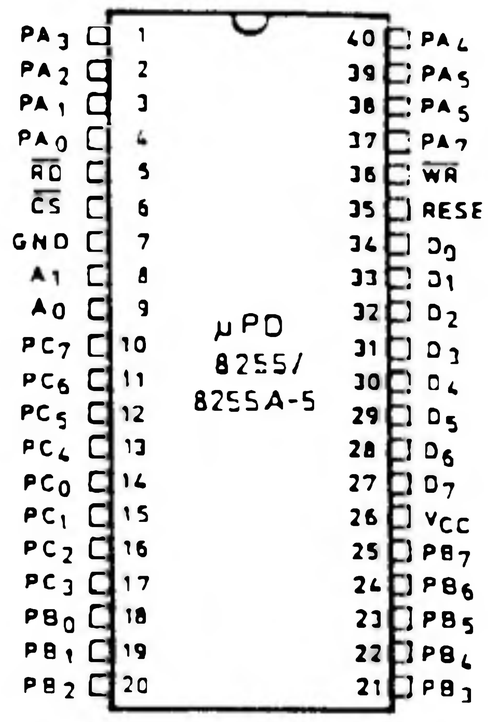

Zustandswort in der Betriebsart 2:

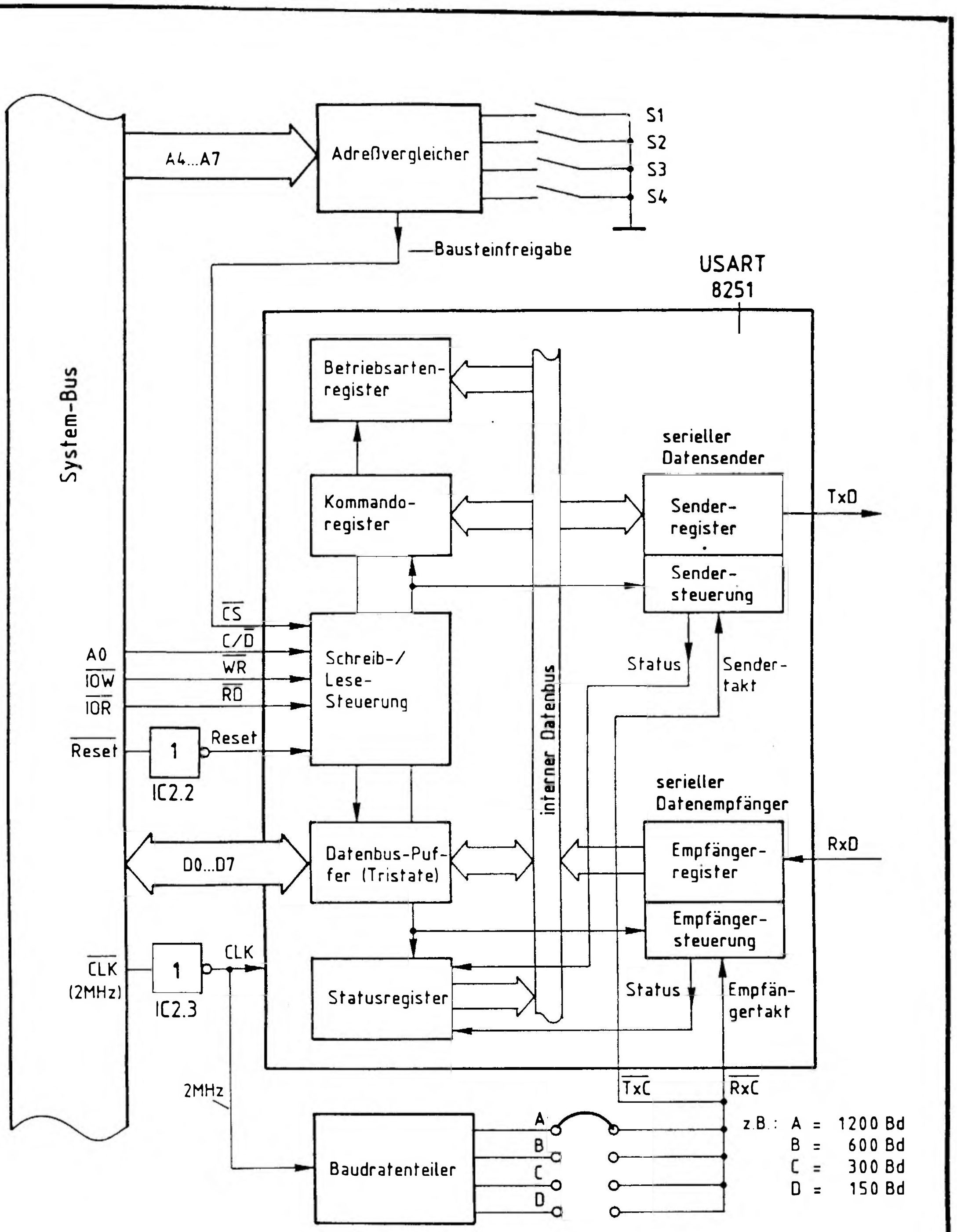
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| $\overline{OBFA}$ | INTE1 | IBFA | INTE2 | INTRA | • | • | • |

• Kanal B: Definiert durch die Betriebsart 0 oder 1

Betriebsart 2 (Zweiweg-Bus)

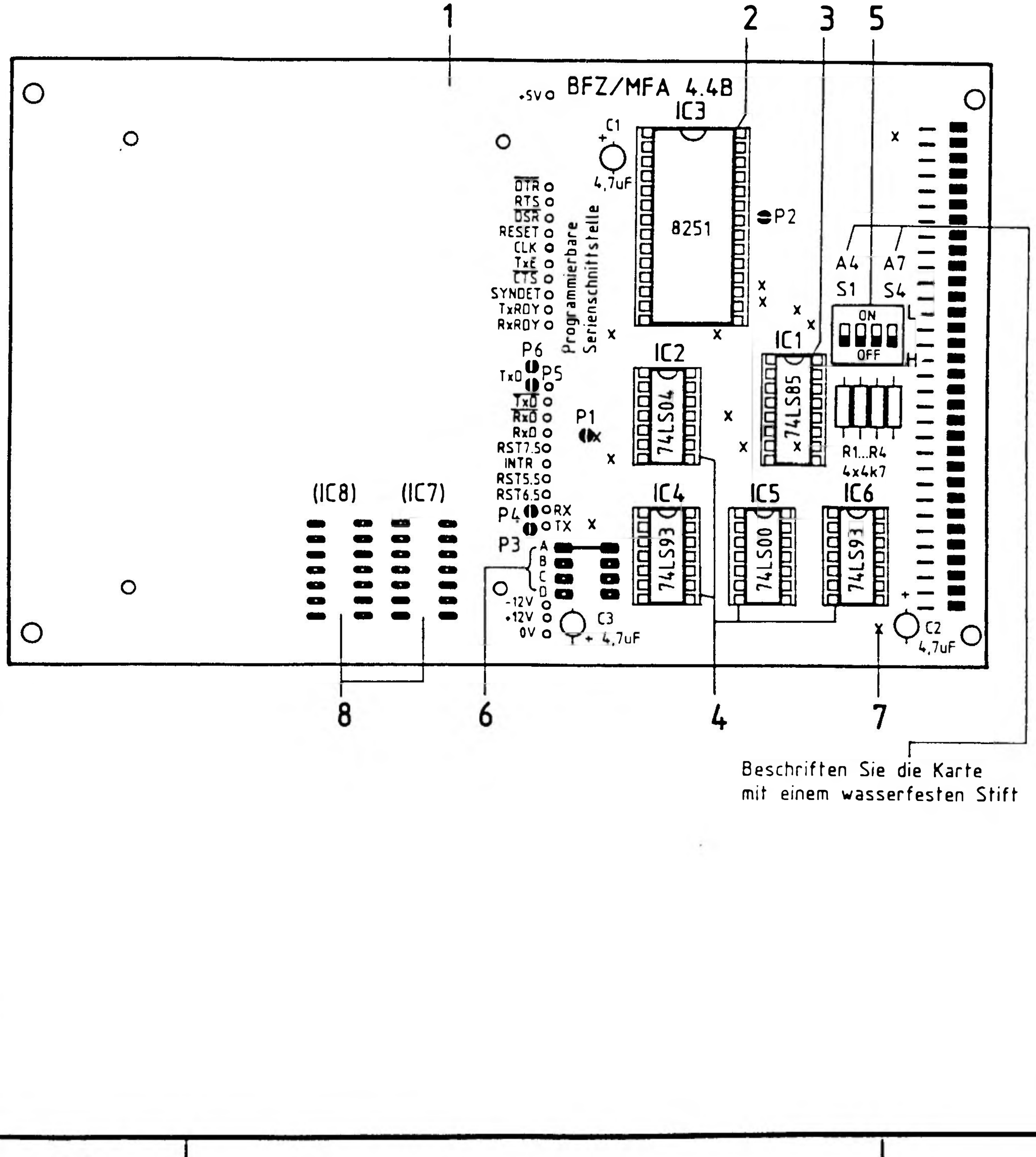


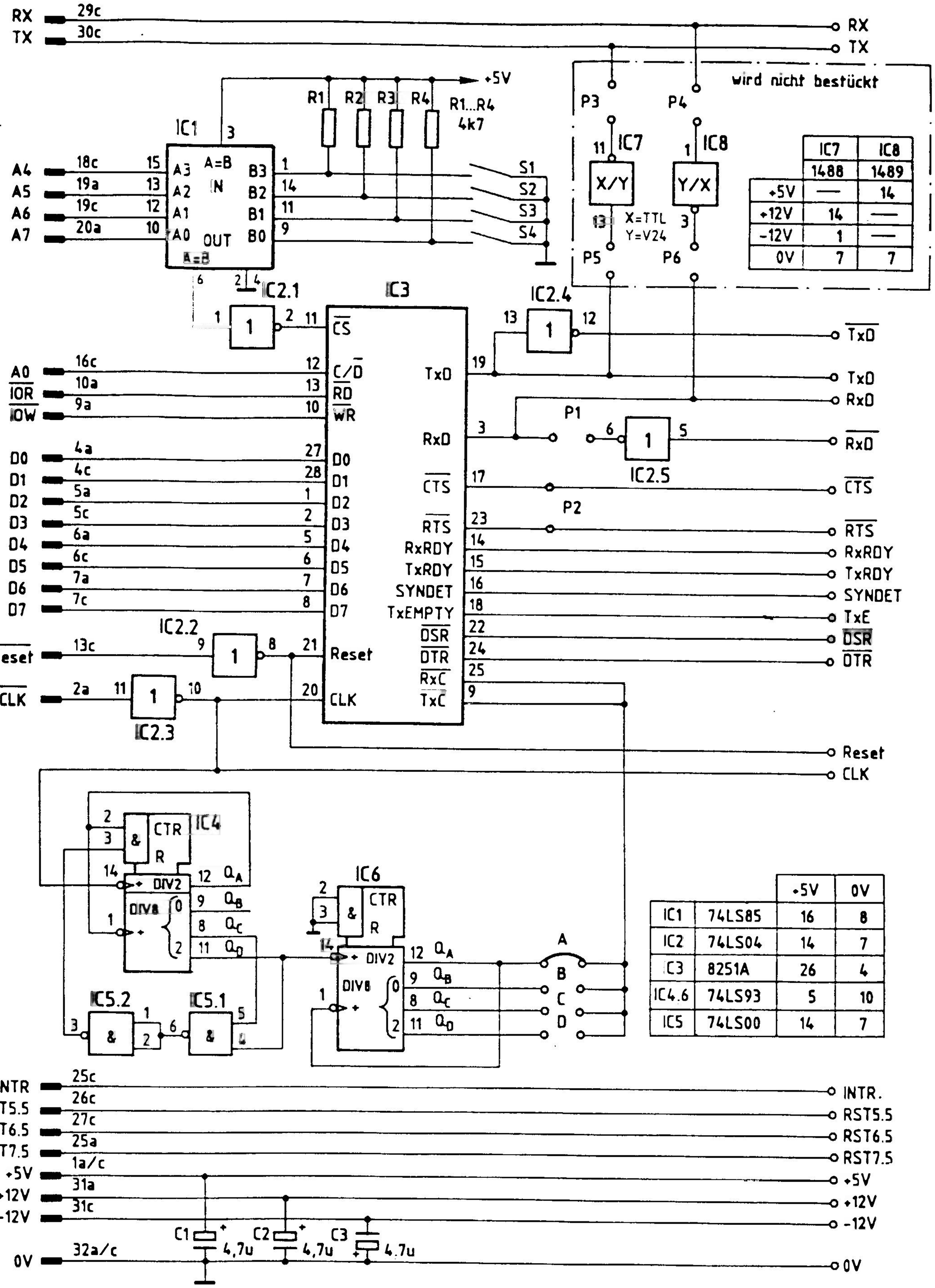



| | | |
|---|---|---|
| Bfz | | |
| | Blockschaltbild Programmierbare Serienschnittstelle | 75 |

## Bestückungsplan




| | | |
|---|---|---|
| Bfz | Programmierbare Serienschnittstelle | 76 |




| | IC7 | IC8 |
|---|---|---|
| | 1488 | 1489 |
| +5V | — | 14 |
| +12V | 14 | — |
| -12V | 1 | — |
| 0V | 7 | 7 |

X=TTL
Y=V24

| | | +5V | 0V |
|---|---|---|---|
| IC1 | 74LS85 | 16 | 8 |
| IC2 | 74LS04 | 14 | 7 |
| C3 | 8251A | 26 | 4 |
| IC4.6 | 74LS93 | 5 | 10 |
| IC5 | 74LS00 | 14 | 7 |

Bfz

Stromlaufplan Programmierbare Serienschnittstelle

77

| 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 |
| 1 | 0 | Asynchronbetrieb | 16 |
| 1 | 1 | Asynchronbetrieb | 64 |

| | | Zeichenlänge |
|---|---|---|
| 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 |

| | |
|---|---|
| 0 | Sender gesperrt |
| 1 | Sender freigegeben |

| | |
|---|---|
| 0 | Ausgang $\overline{DTR}$ führt H-Pegel |
| 1 | Ausgang $\overline{DTR}$ führt L-Pegel |

| | |
|---|---|
| 0 | Empfänger gesperrt |
| 1 | Empfänger freigegeben |

| | |
|---|---|
| 0 | Normaler Betrieb |
| 1 | Break-Signal aussenden |

| | |
|---|---|
| 0 | Fehlerbits des Statusregisters nicht beeinflussen |
| 1 | Fehlerbits des Statusregisters zurücksetzen |

| | |
|---|---|
| 0 | Ausgang $\overline{RTS}$ führt H-Pegel |
| 1 | Ausgang $\overline{RTS}$ führt L-Pegel |

| | |
|---|---|
| 0 | Nächstes Steuerwort wieder in das Kommandoregister schreiben |
| 1 | Nächstes Steuerwort gelangt in das Betriebsartenregister |

| | |
|---|---|
| 0/1 | Nur für Synchronbetrieb von Bedeutung |

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 |

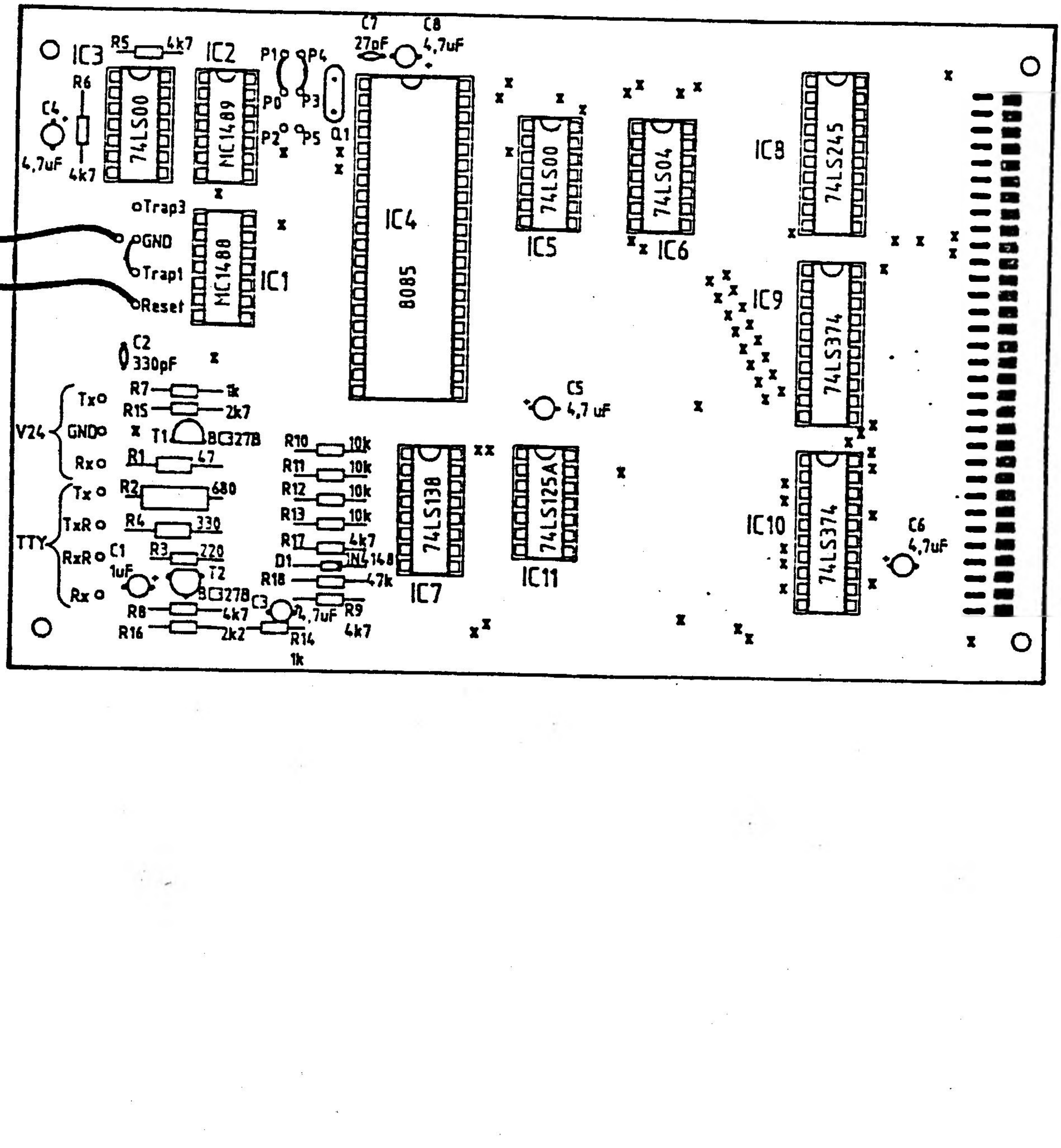
| | |
|---|---|
| 0 | Kein Stopbitfehler beim Datenempfang |
| 1 | Stopbitfehler liegt vor |

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

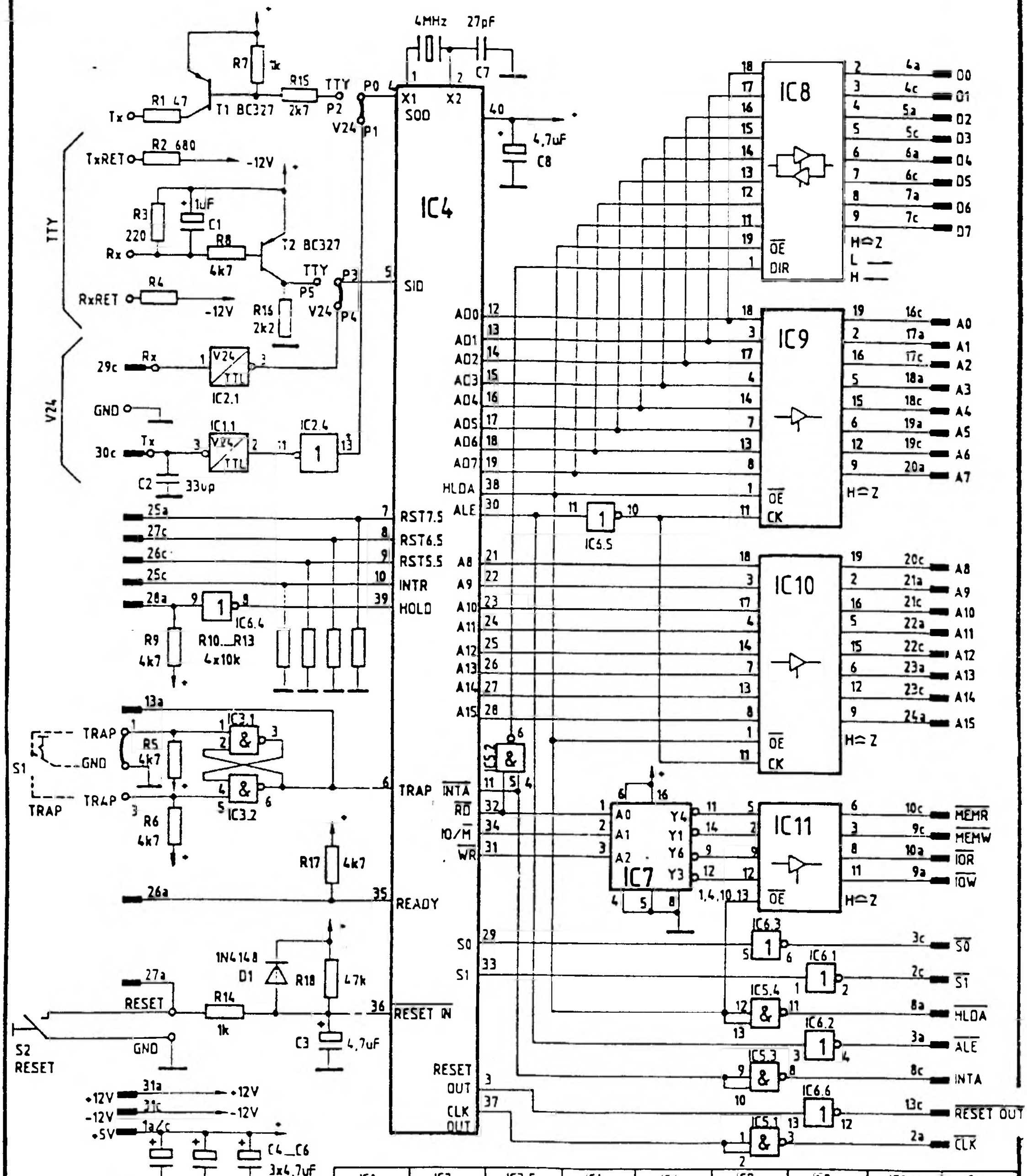
| | |
|---|---|
| 0 | Eingang $\overline{DSR}$ führt H-Pegel |
| 1 | Eingang $\overline{DSR}$ führt L-Pegel |

Das Statusregister

# Bestückungsplan




| | | |
|---|---|---|
| Bfz | | |
| | Bestückungsplan Prozessor 8085 | 44 |




| | IC1 | IC2 | IC3,5 | IC4 | IC6 | IC7 | IC8 | IC9,10 | IC11 |
|---|---|---|---|---|---|---|---|---|---|
| | MC1488 | MC1489 | 74LS00 | 8085 | 74LS04 | 74LS138 | 74LS245 | 74LS374 | 74LS125A |
| •12V | 14 | / | / | / | / | / | / | / | / |
| -12V | 1 | / | / | / | / | / | / | / | / |
| •5V | / | 14 | 14 | 40 | 14 | 16 | 20 | 20 | 14 |
| 0V | 7 | 7 | 7 | 20 | 7 | 8 | 10 | 10 | 7 |

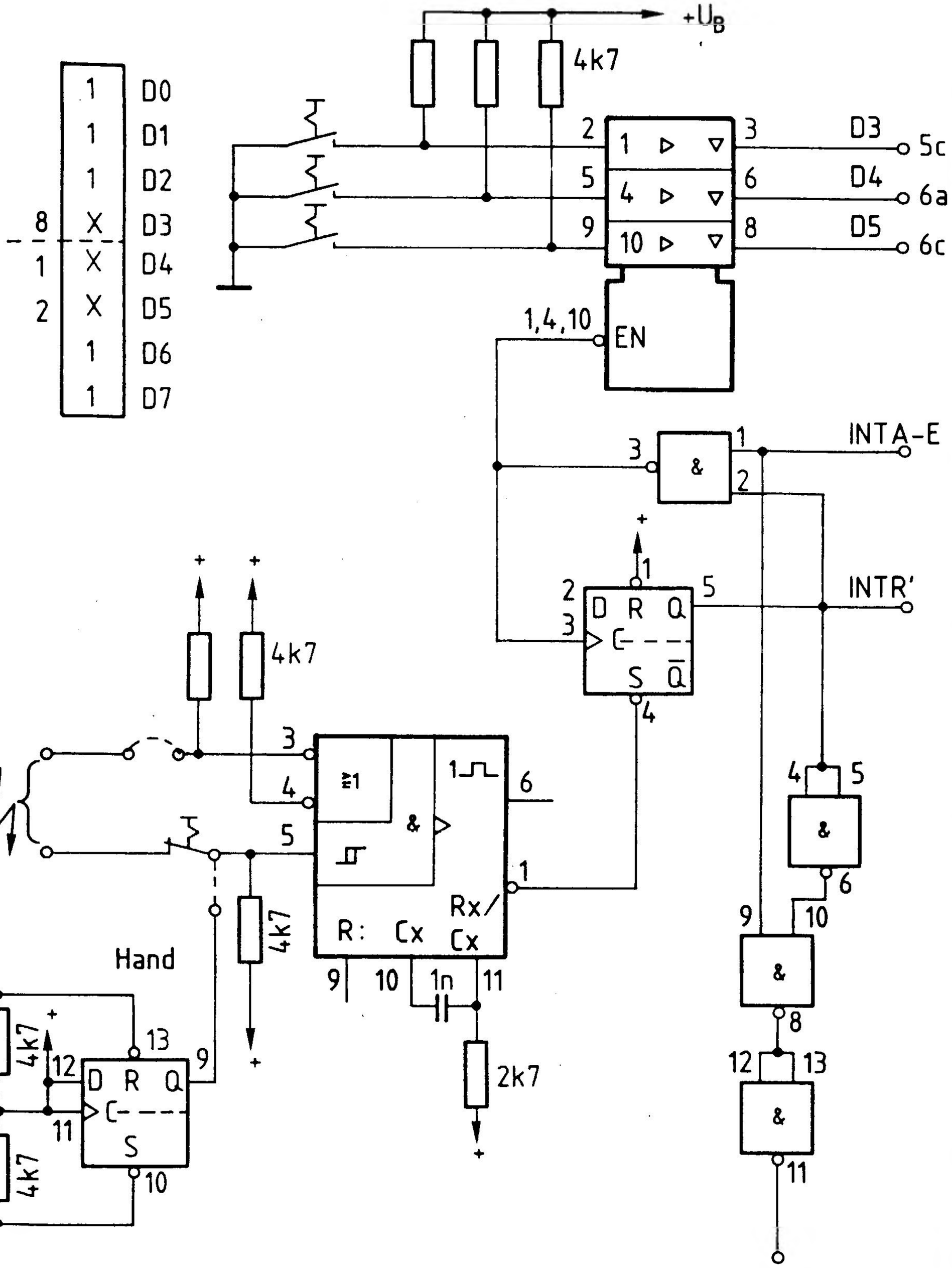
Bfz

Stromlaufplan "Prozessor 8085"

45

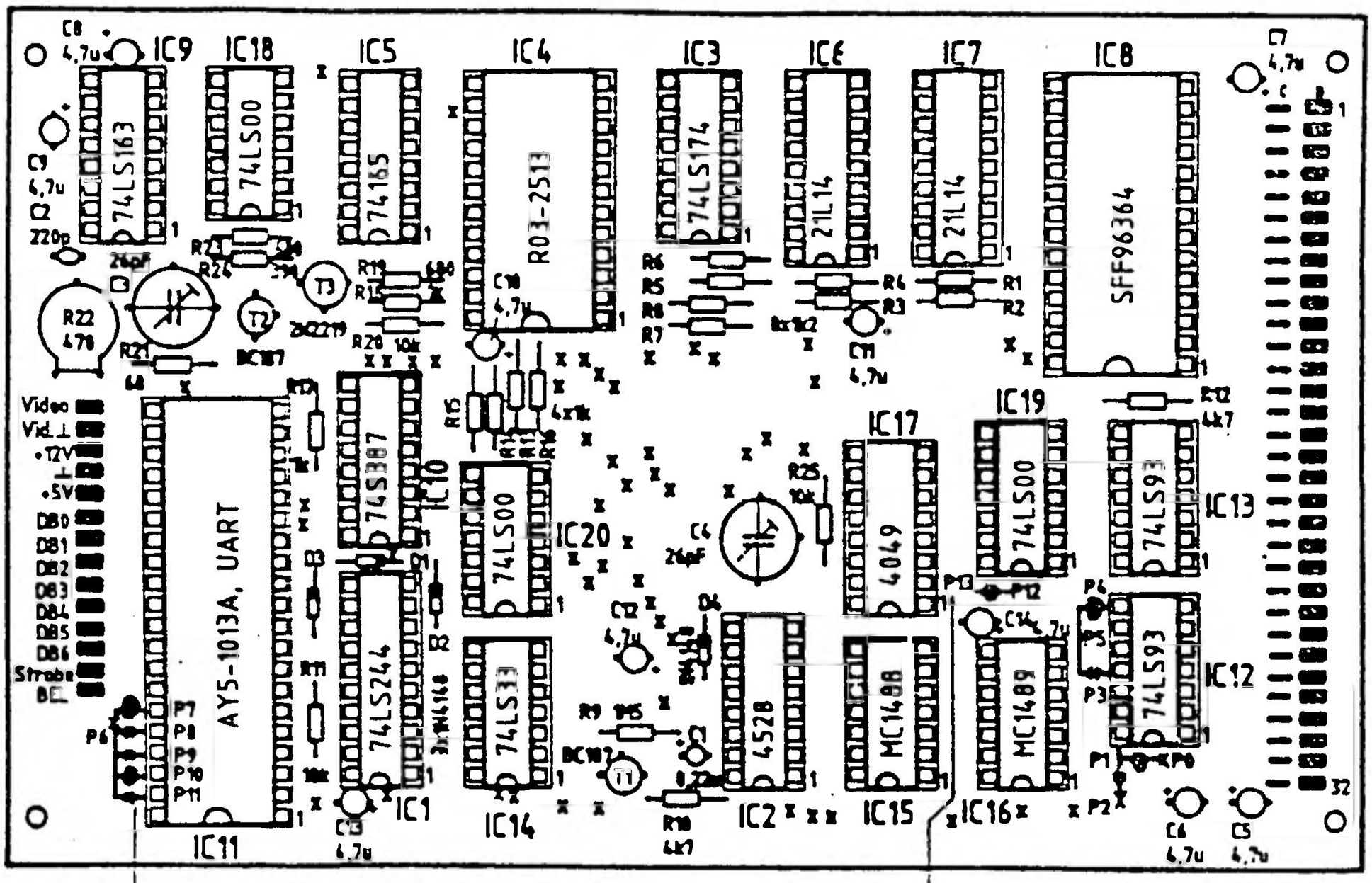
## Teil 5.2 Schaltungen und Bestückungspläne (Erweiterunen)

## Teil 5.1 Schaltungen und Bestückungspläne (Grundgerät)




| | | |
|---|---|---|
| Bfz | Vektor-Interrupt-Karte | 94 |

Bestückungsplan Leiterplatte

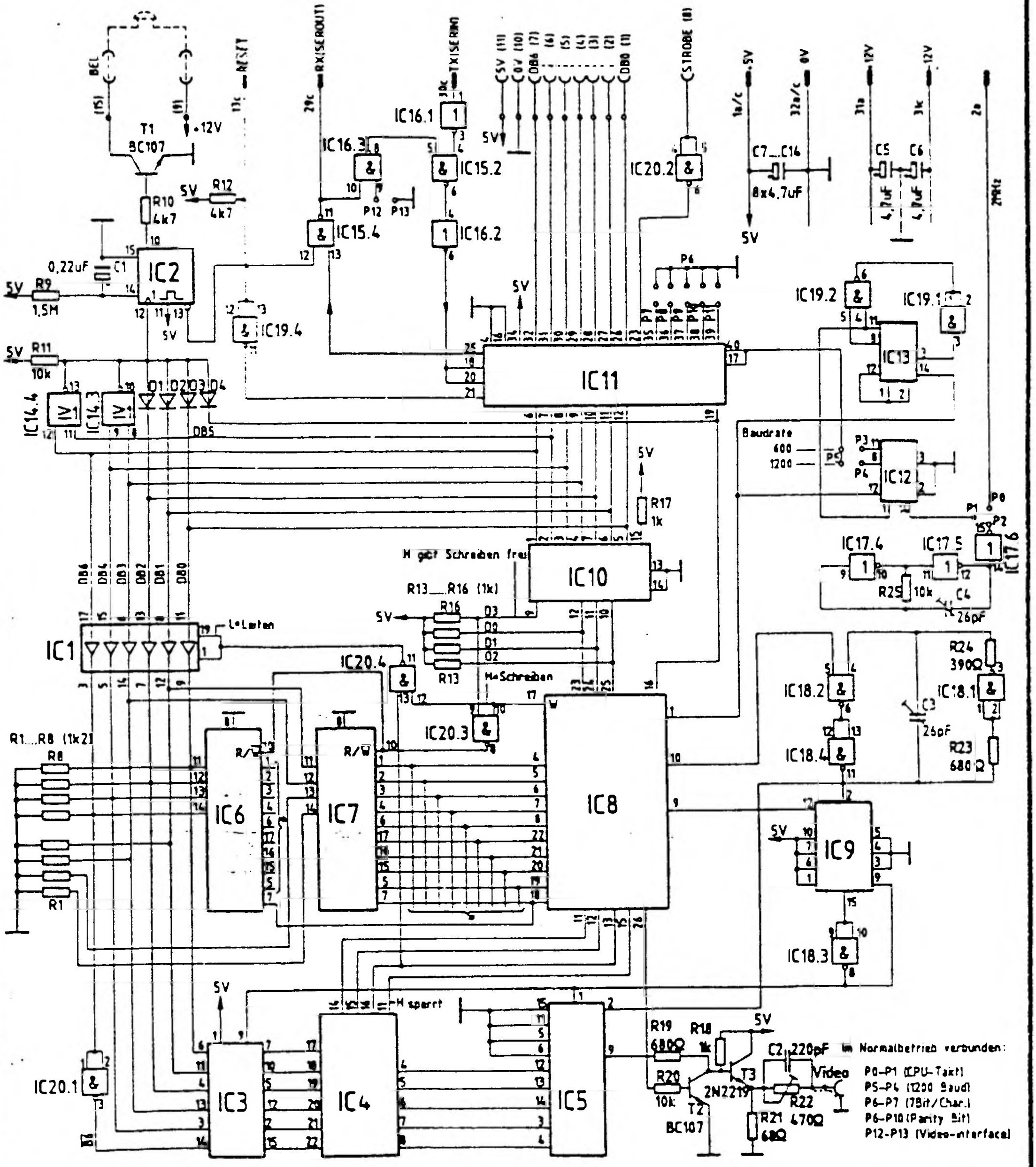



Lötbrücken P..: = Brücke offen
= Brücke geschlossen

Bedeutung der Lötbrücken siehe Stromlaufplan

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

| | | |
|---|---|---|
| Bfz | | |
| | Bestückungsplan Video-Interface | 58 |




| | IC1 | IC2 | IC3 | IC4 | IC5 | IC6,7 | IC8 | IC9 | IC10 | IC11 | IC12,13 | IC14 | IC15 | IC16 | IC17 | IC18,19,20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Typ | 74LS244 | 4528 | 74LS174 | BR03-2513 | 74165 | 2k14 | 96364 | 74LS163 | 74S387 | AYS-1013 | 74LS93 | 74LS33 | MC1488 | MC1489 | 4049 | 74LS00 |
| -5V | 20 | 16 | 16 | 24 | 16 | 18 | 28 | 16 | 16 | 1 | 5 | 16 | / | 14 | 1 | 14 |
| 0V | 10 | 8 | 8 | 10 | 8 | 9 | 14 | 8 | 8 | 3 | 10 | 7 | 7 | 7 | 8 | 7 |
| -12V | / | / | / | / | / | / | / | / | / | / | / | / | 14 | / | / | / |
| -12V | / | / | / | / | / | / | / | / | / | 2 | / | / | 1 | / | / | / |

Bfz

Stromlaufplan Video-Interface

59

RESET SERIN SEROUT

2

IC16 V24 TTL

IC15 V24 TTL

ASCII-Tastatur

7

2-MHz-CPU-Takt

Zeichen gesendet

16 x Baudrate in Hz

IC11

IC12,13,19

IC17

Parallel/Serien-Wandler

UART

Serien/Parallel-Wandler

Baudraten-Teiler

P3 600 (1/208)

P5 1200 (1/104)

P4

(1/2)

P0

P1

P2

2-MHz-Oszillator

(nur für Inbetrieb-nahme)

T1 IC2,14

D1...D4

BEL (07H,Klingel)

7

interner Datenbus

7

6

7

IC1

IC10

Puffer

Zeichen einschreiben

Steuerzeichen-Decoder

Zeichen von der Tastatur empfangen

1 MHz

3

23... 25

16

1

IC6,7

IC8

IC18

3-Bit-Steuerwort

Video-Synchr.

17

R/W

Neues Zeichen einschreiben bzw. Zeichen lesen

Steuerung des Strahl-rücklaufs nach 64. Zeichen

10

Punkt-Oszillator ca. 13MHz

Bildbreite

6

10

D0...D5

A0...A9

Zeichen-Adresse für Bild-wiederhol-speicher

CRTC

IC9

Teiler durch 8

~1,6MHz

9

Monitor

Bildwieder-holspeicher (1K-RAM)

6

6

Zählt Zeichen pro Zeile

Zeichen-Punkt-reihe

Cursor-Ausgabe

Synchr.

11... 13

15

26

3

neues Zeichen laden

13MHz

Bild-Helligkeit

Laden

Reihen-Adresse

Cursor-Einblendung

Laden | Schieben

Horiz. u. Vertik. Synchron-Impulse

Video

Zeichen-zwischen-Speicher

(im ASCII-Code)

6

Zeichen-Adresse

5-Bit-Punkt-muster (parallel)

5

Punkte pro Reihe mit Zwi-schenraum

1

8-Bit-Punkt-muster (seriell)

Zeichengenerator

8-Bit-Schiebe-register

Video-Mischer

IC3

IC4

IC5

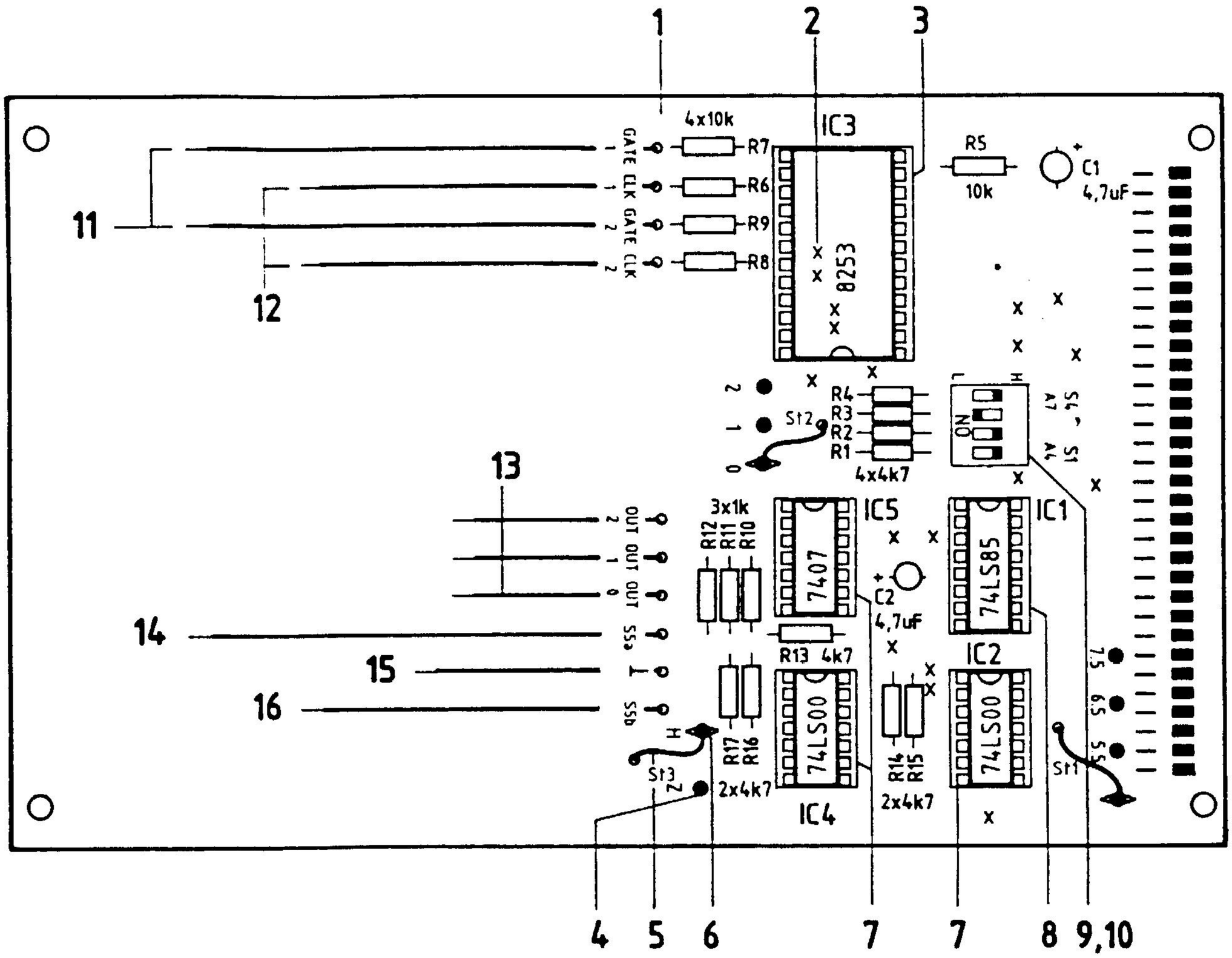
T2,T3

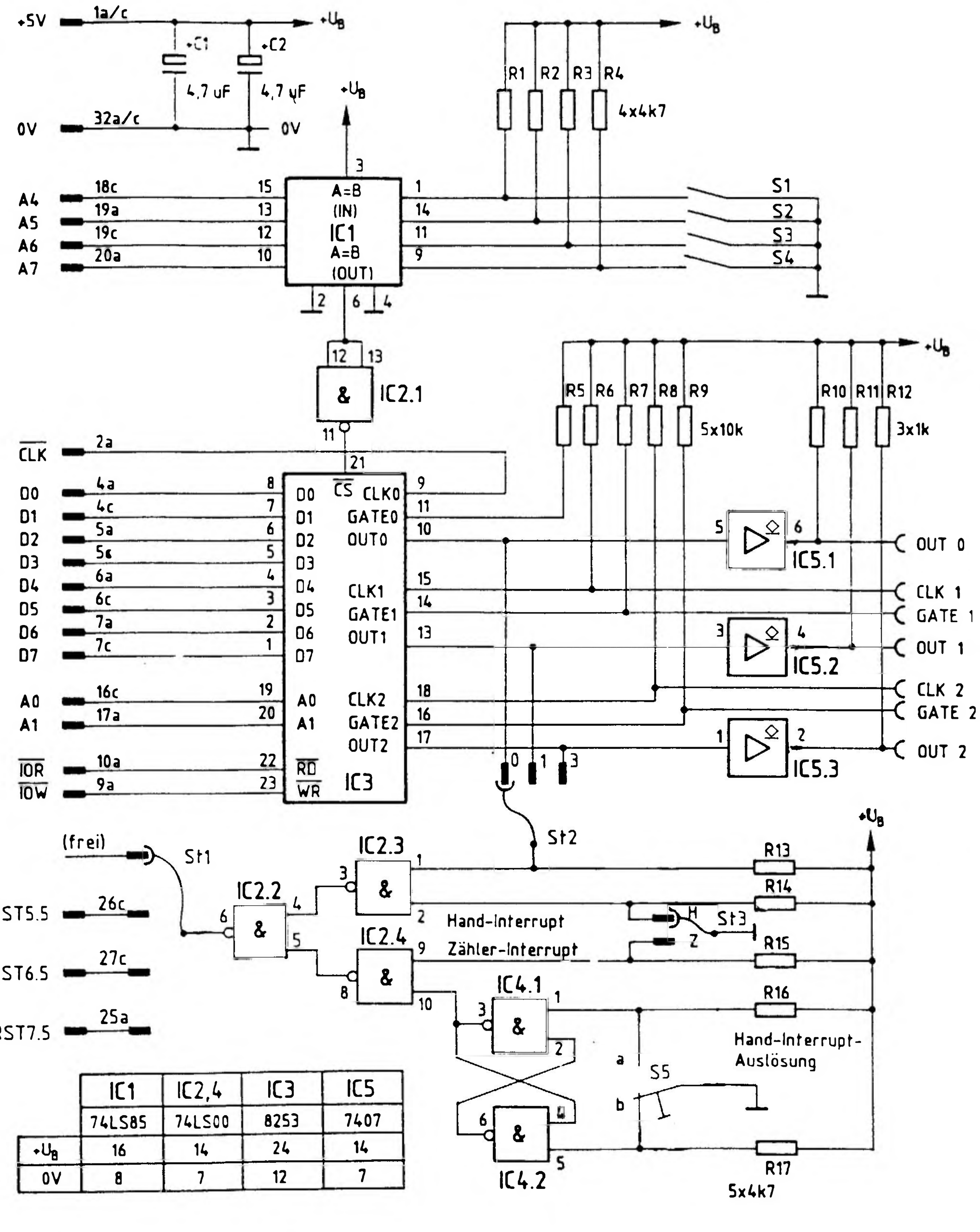
Bfz

Blockschaltbild Video-Interface

60

Bestückungsplan






| | IC1 | IC2,4 | IC3 | IC5 |
|---|---|---|---|---|
| | 74LS85 | 74LS00 | 8253 | 7407 |
| $+U_B$ | 16 | 14 | 24 | 14 |
| 0V | 8 | 7 | 12 | 7 |

Aufbau des Steuerwortes:

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Datenbit |
|---|---|---|---|---|---|---|---|---|
| SC1 | SC0 | RL1 | RL0 | M2 | M1 | M0 | BCD | Bit-Bezeichnung |

bei "1": Zähler zählt im BCD-Code
bei "0": Zähler zählt binär

Festlegung der Betriebsart
("M" von "mode", Betriebsart)

Definition des Lese- und Ladevorgangs
("RL" von "read/load", lesen/laden)

Auswahl des zu programmierenden
Zählers ("SC" von "select counter",
Zählerauswahl)

Festlegung der Betriebsarten:

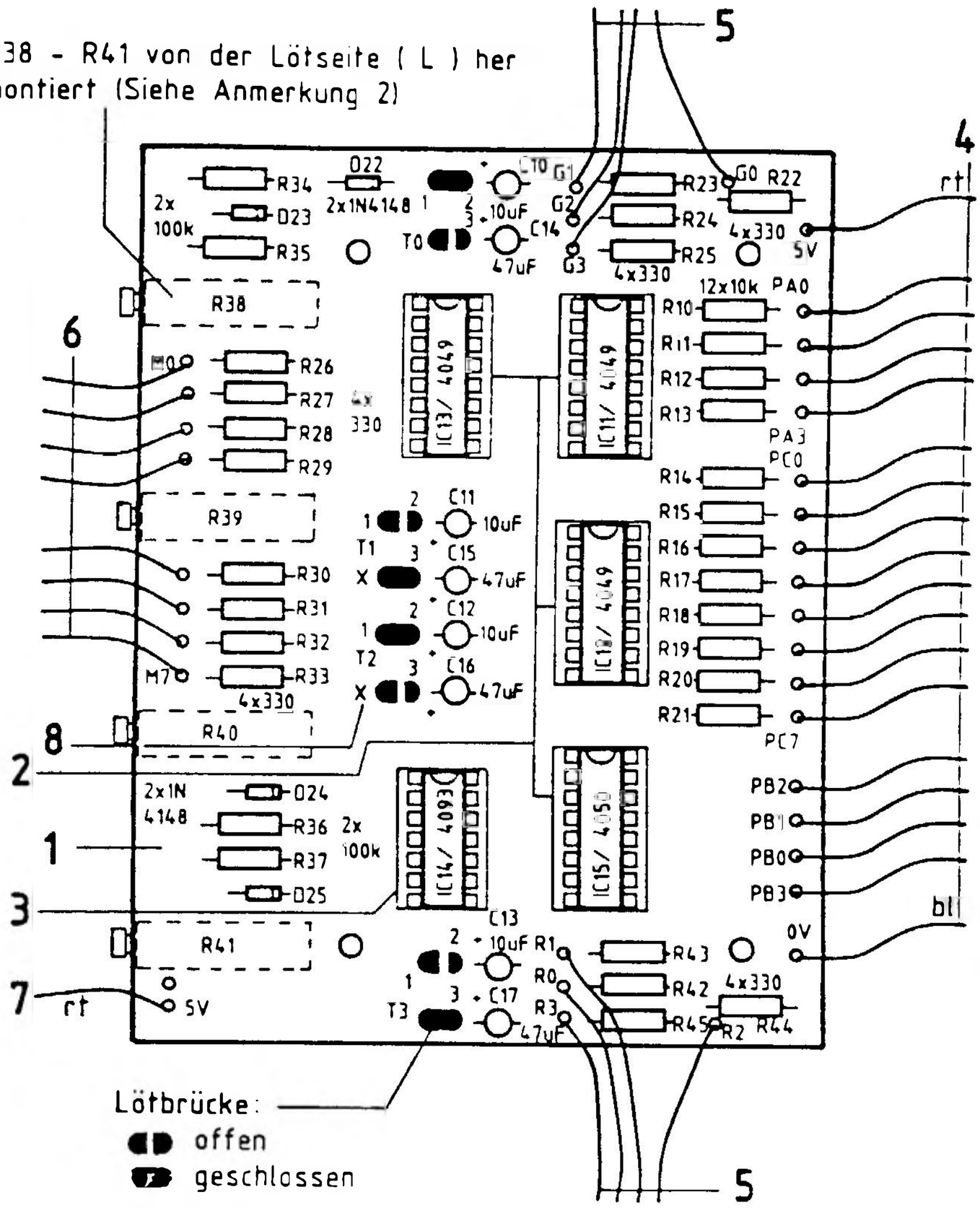
| M2 | M1 | M0 | Nummer und Bezeichnung der Betriebsart |
|---|---|---|---|
| 0 | 0 | 0 | 0 Unterbrechen beim Zählerstand Null |
| 0 | 0 | 1 | 1 Monostabile Kippstufe, retriggerbar |
| X | 1 | 0 | 2 Programmierbarer Frequenzteiler |
| X | 1 | 1 | 3 Programmierbarer symmetrischer Rechteckgenerator |
| 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 RL0 und RL1:

| Bitkombination Nr. | RL1 (D5) | RL0 (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 niederwertigen, danach des höherwertigen Bytes eines 16-Bit-Zählers |

## Bestückungsplan




| | | |
|---|---|---|
| Bfz | | |
| | Zeitwerk (4fach) | 70 |

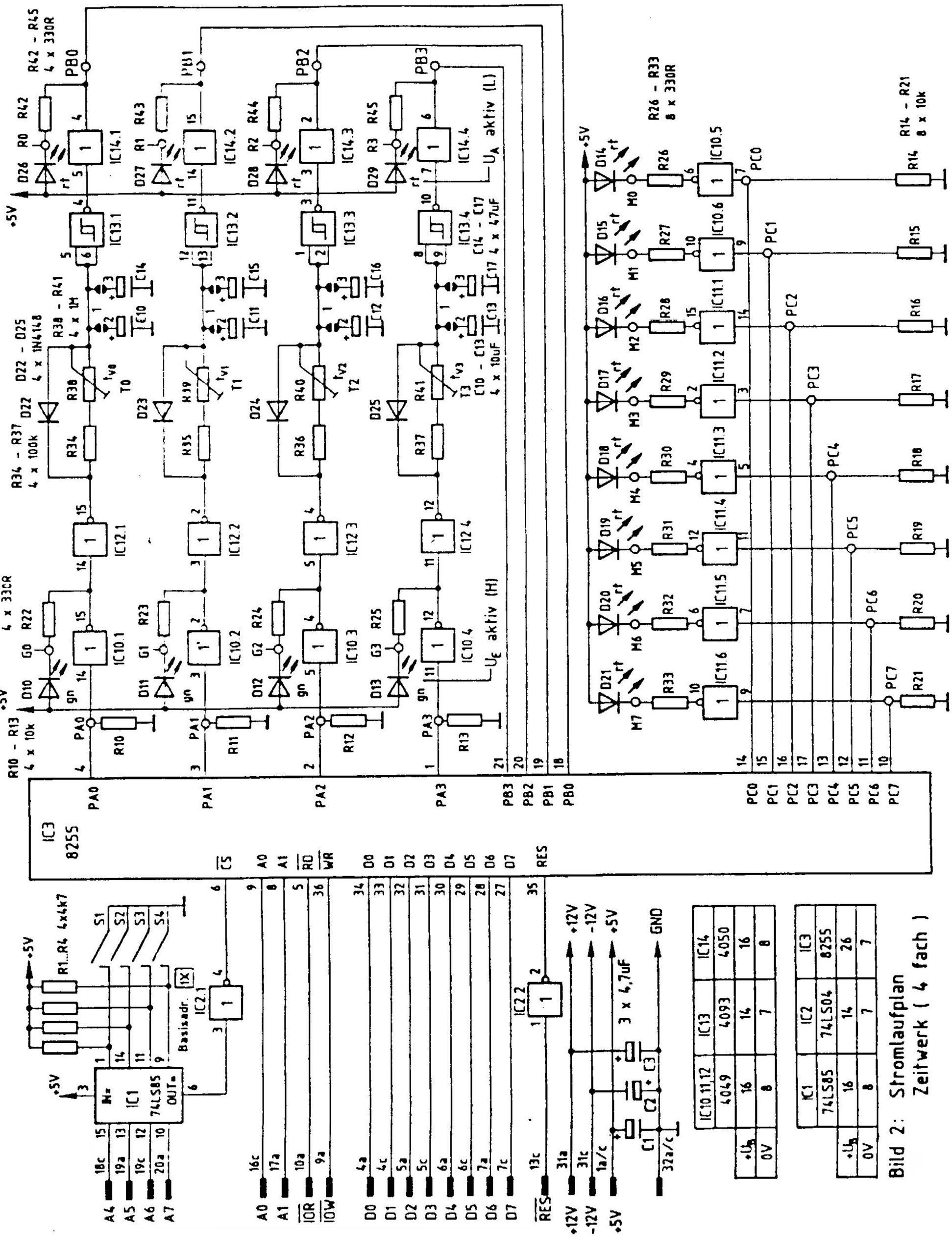



Bild 2: Stromlaufplan
Zeitwerk ( 4 fach )

| | | |
|---|---|---|
| Bfz | Stromlaufplan Zeitwerk ( 4 fach ) | 71 |