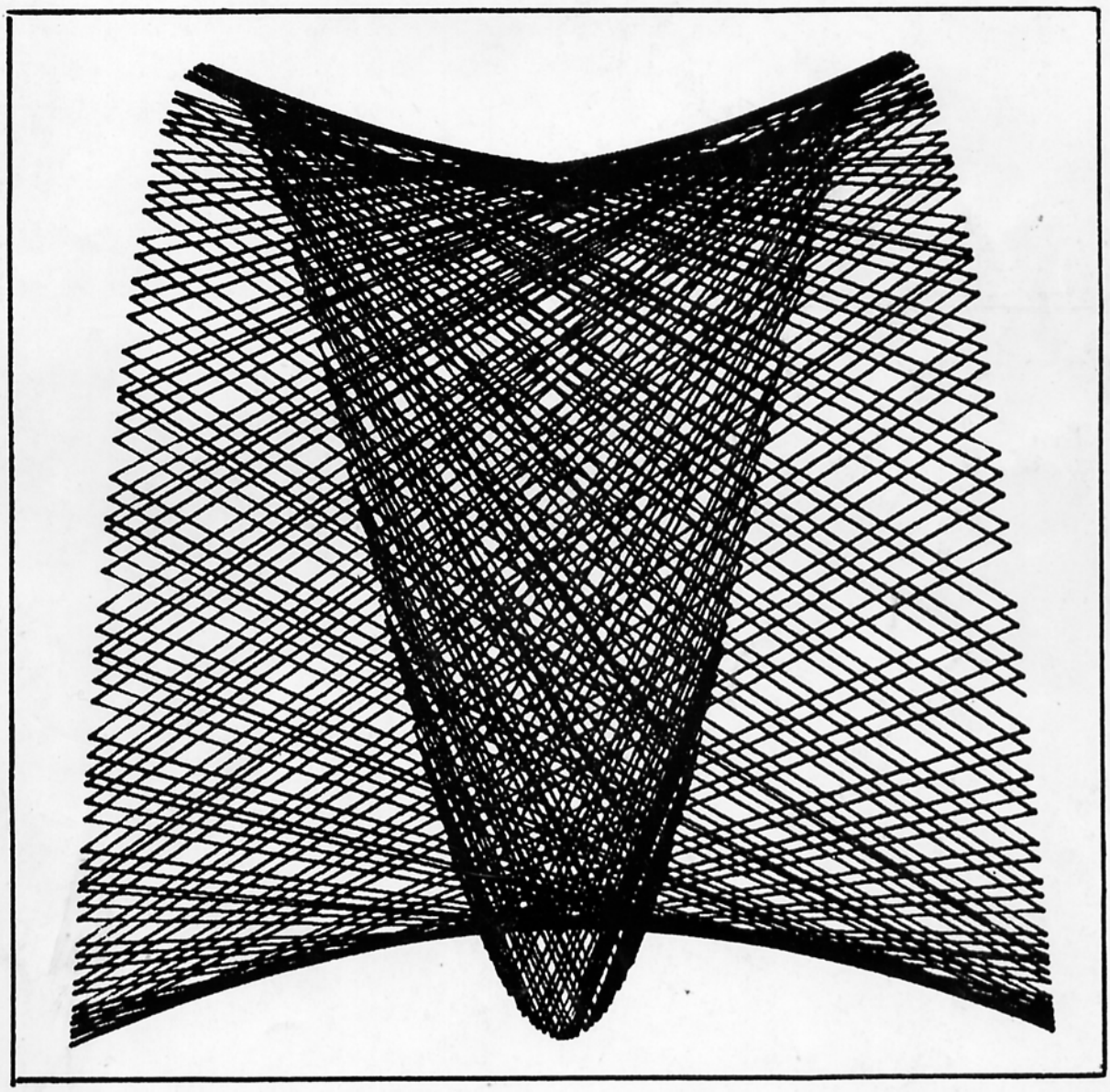
NDR

mc

Arbeitsmaterial zur Serie Mikroelektronik

Schaltplaene & Unterlagen



Franzis-Software-Service

Uwe u. Claudia Koch
Frankenstraße 25
5880 LÜDENSCHEID

Lieber Leser,

in diesem Heft haben wir ganz schnell wichtige Informationen zur Fernsehserie Mikroelektronik zusammengefasst

- Schaltpläne und Beschreibungen zu den einzelnen Versuchen
- Tabellen mit den Farbcodes für Widerstände und Kondensatoren
- Schaltpläne der einzelnen Baugruppen - insbesondere der 68008-Platinen
- Bestückungspläne und Stücklisten
- Datenblätter der LSI-Bausteine
- Fehlerkorrekturen zum Buch "Mikrokomputer selbstgebaut" von R.D. Klein

Weil es so schnell gehen mußte, konnten wir das Heft nicht immer so schön gestalten, wie wir wollten. Aber es ist ja auch als Arbeitshilfe gedacht - da ist die Schönheit nicht ganz so wichtig.
Als erstes sehen Sie auf den folgenden Seiten, was Ihnen das Grundprogramm alles auf dem Bildschirm bietet - das Menü, Ändern des Speichers, EPROM programmieren und Einzelschrittbetrieb.

Viel Erfolg beim Bau des NDR-Klein Computers,

Ihr Franzis-Software-Service

Uwe u. Claudia Koch
Frankenstraße 25
5880 LÜDENSCHEID

ROK-Grundprogramm

1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter

1 = Laden CAS
2 = Speichern CAS
3 = Pruefen CAS
W = weiter

1 = IO lesen
2 = IO setzen
3 = Einzelschritt
W = weiter

aendern

```
Adr:ram
8800 : 21 50 00
8803 : FF
8804 : 00
8803  cd schreite
```

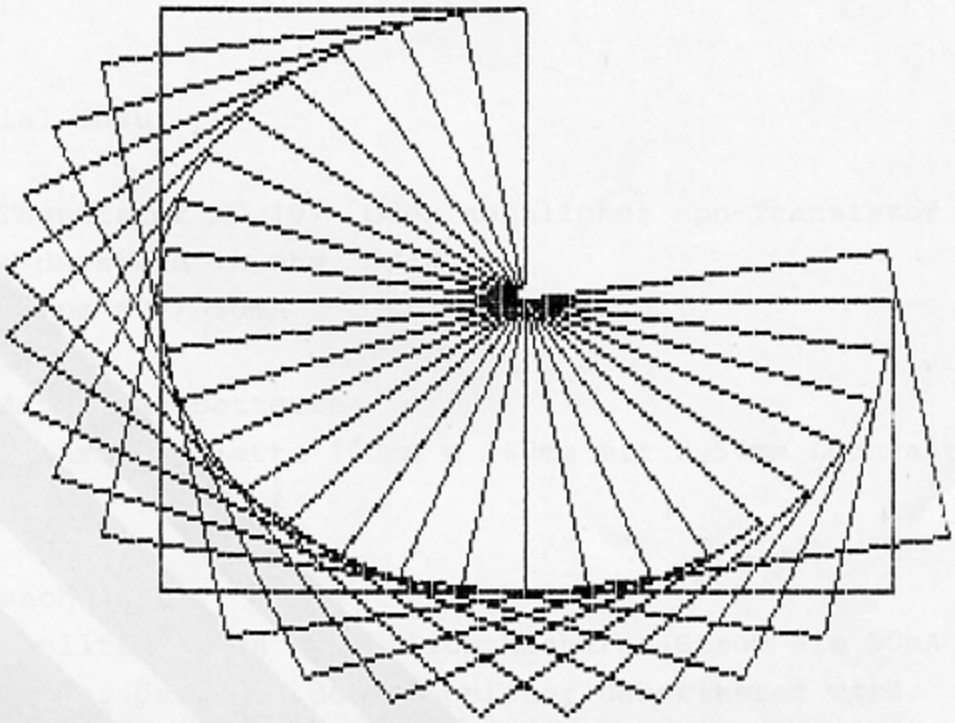
```
+  =Adr + 1   - =Adr - 1
M  =Menue     R =Adr
cr =ein Befehl weiter
```

EPROM PROG

von	8800
bis	88ff
nach	0

Bereit = B 00FF

PROM FEHLER M=Men



```
R=Adr S=Seite N=n-Mal B=Bis cr=Befehl ausfuehren M=Menue 881C : CD 00 88 | CD QUADRAT
af   bc   de   hl   af' bc' de' hl' ix   iy   sp   i  r
0044 0010 8819 8035 0000 0000 0000 0000 0000 0000 8FFA 00 67
```

Schaltplaene Schaltplaene Schaltplaene Scha

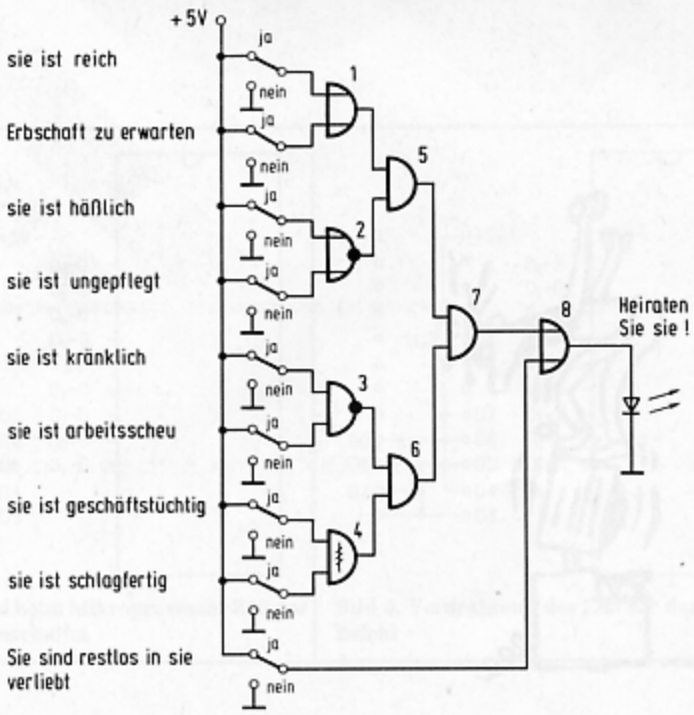
1.1 Rechnen als Schalten

In dieser Folge wird der Transistortreiber aufgebaut.
Der Aufbau kann auf einer Lochrasterplatte erfolgen.
"Brettschaltung"

Material dazu:

1x Transistor BC 107 oder aehnlicher npn-Transistor
1x Widerstand 1k Ohm 1/4W
1x Lampe 6V / 50mA
1x Taster
1x 4.5V Flachbatterie
1x Lochrasterplatte 100mm x 160mm mit 2.54mm Lochraster

zu besachten:
Lampe sollte keinen wesentlich hoeheren Strom als 50mA benoetigen, da sonst der Transistor ueberlastet wird.



Schaltfunktion eines "Heirats-indikators" zur Veranschaulichung logischer Verknüpfungen.

1.2 Verknuepfungen

Der Treiber wird erweitert und es wird ein Und-Glied aufgebaut. Aufbau ebenfalls auch einer Loschrasterplatte.

Material:

- Material der Folge 1.1 plus:
-- Versuch Inverter
1x Transistor BC 107 oder aehnlicher npn-Transistor
1x Widerstand 1k OHM 1/4 Watt
1x Lampe 6V /50mA

-- Versuch Und-Glied
1x IC 7408 oder 74LS08
1x IC-Fassung 14polig
2x Widerstand 330 Ohm
2x Taster (vom vorherigen Versuch ist schon eine vorhanden)
- Treiber aus Folge 1.1

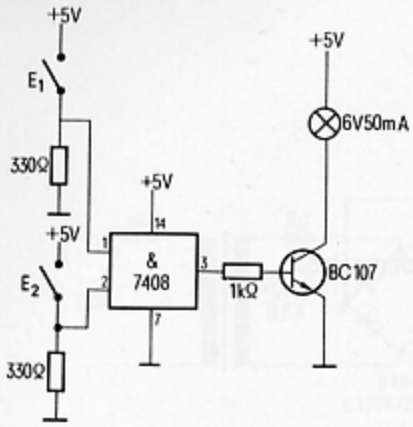


Bild 1. Versuchsschaltung mit einem UND-Gatter

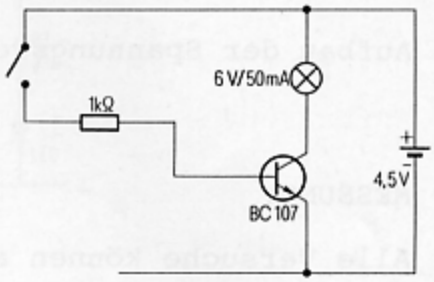


Bild 2. Treiberschaltung mit npn-Transistor

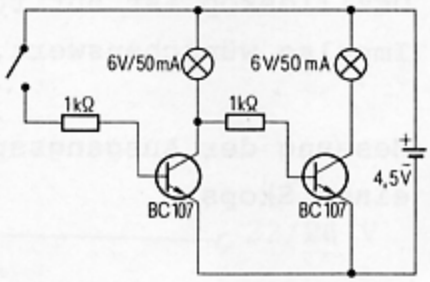


Bild 3. So entsteht ein NICHT-Glied aus einer Treiberschaltung

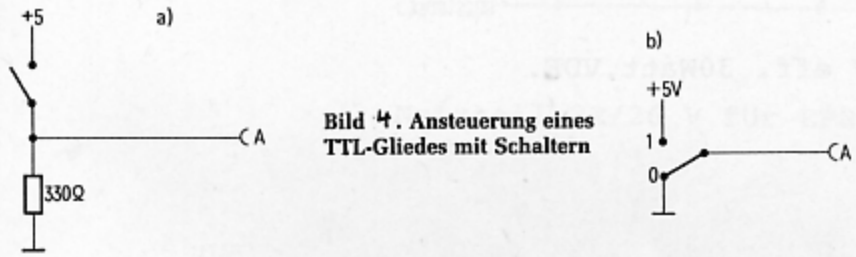


Bild 4. Ansteuerung eines TTL-Gliedes mit Schaltern

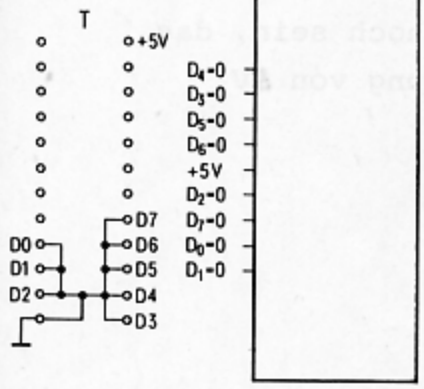


Bild 5. So wird beim Mikroprozessor Z80 der NOP-Befehl beschaltet

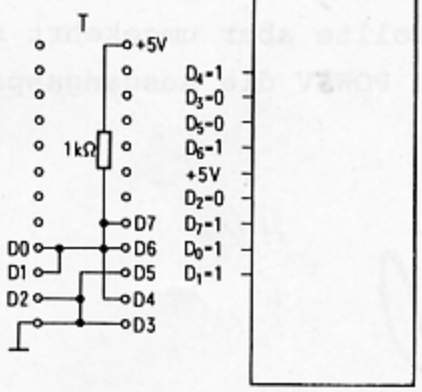


Bild 6. Verdrahtung des Z80 für den OUT-Befehl

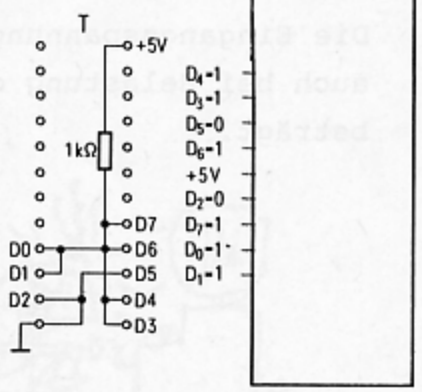


Bild 7. Der IN-Befehl des Z80 wird verdrahtet

2.1 Gleich riecht er

Aufbau der Spannungsversorgung für den Computer.

MESSUNG:

Alle Versuche können auch mit einem einfachen Prüfstift (siehe Anhang) und Multimeter durchgeführt werden. Ein Oszilloskop ist nur für eine genauere Betrachtung der Impulse wünschenswert.

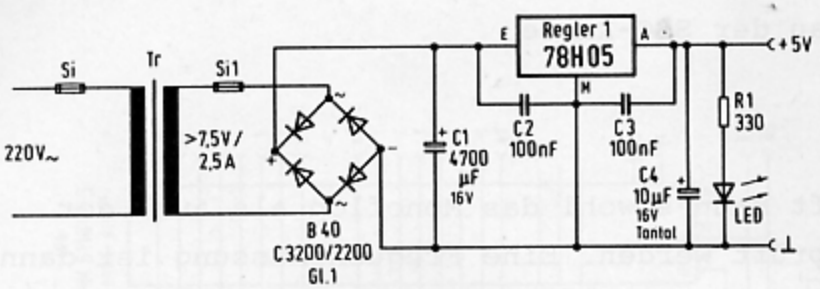
Messung der Ausgangsspannung mit einem Multimeter anstelle eines Skops.

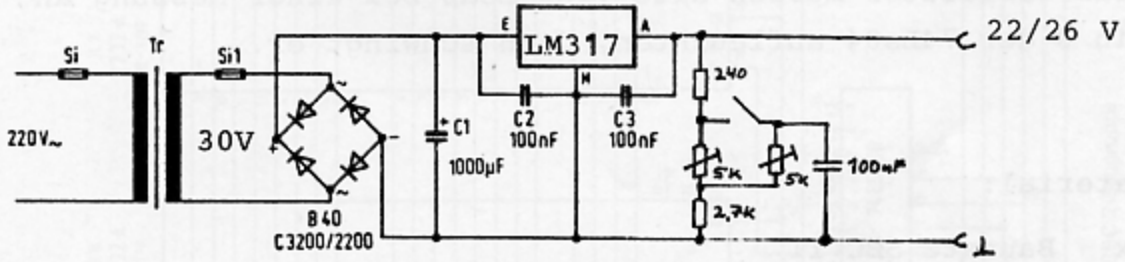
Material:

1x Bausatz POW 5V
1x Trafo mit ca. 10V eff. 30Watt.VDE.

zu beachten:

Bei der Wechselspannung am Eingang ist zu beachten, das der Wert niemals höher liegt, als die auf dem Elektrolytkondensator angegebene. Z.B. Der Elko hat max. 16V, dann darf die Spitzenspannung, das ist (1.4142 * Effenktivspannung) des Trafos nicht hoeher als 16V liegen.
Die Eingangsspannung sollte aber umgekehrt so hoch sein, das auch bei Belastung der POW5V die Ausgangsspannung von 5V beträgt.





Netzteil 22/26 V für EPROM-Programmierer

2.2 Geschafft: Er schwingt

Erster Teilaufbau der SBC-Karte

Messung:

Mit dem Prüfstift kann sowohl das Monoflop als auch der Oszillator überprüft werden. Eine Frequenzmessung ist dann natürlich nicht möglich, jedoch ist es sehr unwahrscheinlich, das der Quarz auf einer falschen Frequenz schwingt.

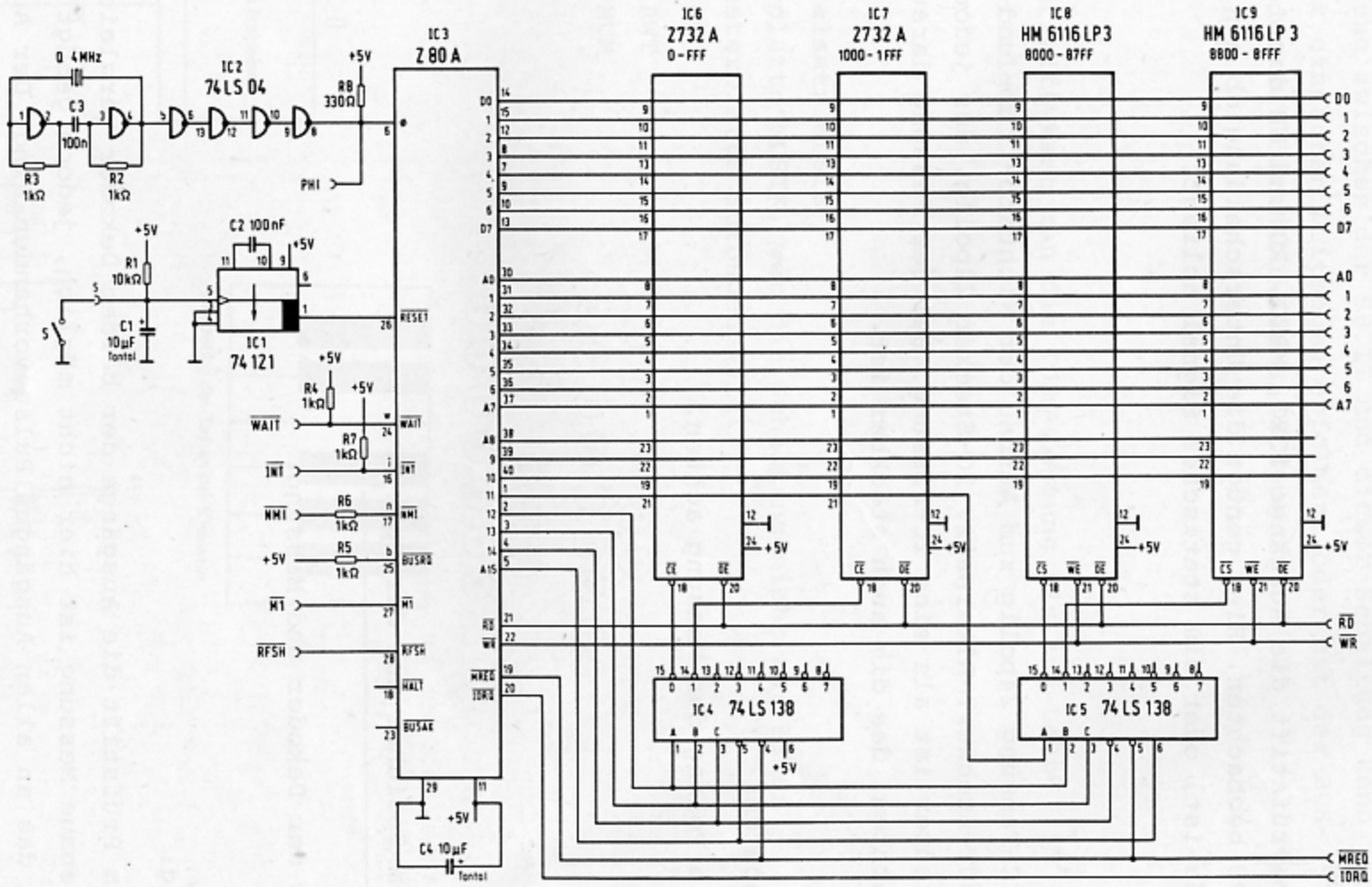
Beim Monoflop muß bei jedem Reset-Vorgang an Pin 1 ein kurzer Plus erscheinen. Der Prüfstift zeigt den kurzen Plus durch einen Wechsel der Leds L3 und L4 an.
Beim Oszillator müssen alle vier Leds bei einer Messung an Pin 8 der 74Ls04 aufleuchten, dann schwingt er.

Material:

1x Bausatz SBC II
1x Reset-Taster

zu beachten:

Polung von ICs beachten, da sonst Zerstörung droht. Ferner auf gute Lötstellen achten, heiß löten und solange warten bis das Lötzinn anfängt zu fließen. Gute Lötstellen haben nur wenig Lötzinn und dennoch ist das gesamte Lötauge benetzt. Keine Angst vor zu langem Löten, da ja immer Sockel verwendet werden kann nichts passieren.



Schaltung der SBC 2-Baugruppe

2.3 Der Nichtstu-Befehl

Weiterer Aufbau der SBCII.

Messung:

Mit dem Prüfstift die Ausgänge IORQ, MREQ, RD und WR der Z80-CPU beobachten. Hier genügt die Unterscheidung ob ein Plus da ist, oder ein statisches Signal anliegt.

Material:

3x IC-Fassung 24polig zum Aufbau der Nichtstu-, IN- und OUT-Stecker. Alternativ IC-Stecker 24polig, der jedoch teurer ist als eine IC-Fassung. Bei der Fassung darauf achten, das die auch steckbar ist.

zu beachten:

Auf die Polung der Fassung achten.

2.4 Dem Speicher auf der Spur

Einbau der Dekoder und Messung.

Messung:

Mit dem Prüfstift die Ausgänge der beiden Dekoder vergleichen. Eine genaue Messung ist hier nicht möglich, jedoch genügt zu sehen, das an allen Ausgängen Pulse vorhanden sind. Der Ausgang des ROM-Dekorders Pin 15 zeigt ein anderes Muster als die anderen Ausgänge.

2.5 EPROM mach Musik

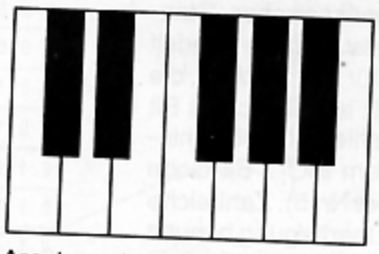
Eine kleine Peripherieschaltung wird aufgebaut.
Die Verbindung zwischen der SBCII und dieser Schaltung kann entweder mit einfachen Leitungen erfolgen, oder mit der BUS-Baugruppe oder mit der MINI-Bus-Baugruppe.

Messung:

Mit dem Prüfstift kann man die IORQ-Leitung bis zum Lautsprecher verfolgen.

Material:

1x Materialsatz MUSIK
1x Leiterplatte MUSIK, wer die Schaltung nicht auf einer Lochrasterplatte aufbauen will.
1x EPROM MUO
1x EPROM RWT
1x EPROM MUM



Anordnung der Tasten der Klaviatur

2.6 Alarmstufe ROT

Aufbau eriner weiteren Peripheriekarte.

Messung:

Wieder mit dem Prüfstift.

Material:

1x Bausatz AMPEL (incl. Leiterplatte Ampelversuch)

oder

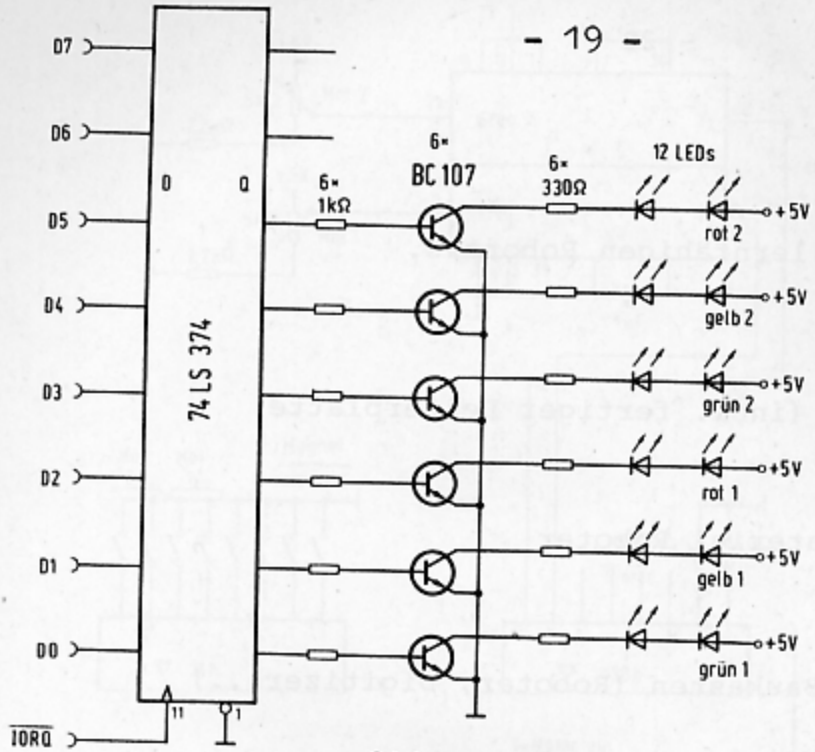
1x IOE Bausatz +Ampelmaterial
1x EPROM AST
1x Bus-Platte. Empfehlenswert ist es sie gleich
für den Vollausbau mit allen Kontakten zu bestellen.

Was ist eigentlich ASCII?

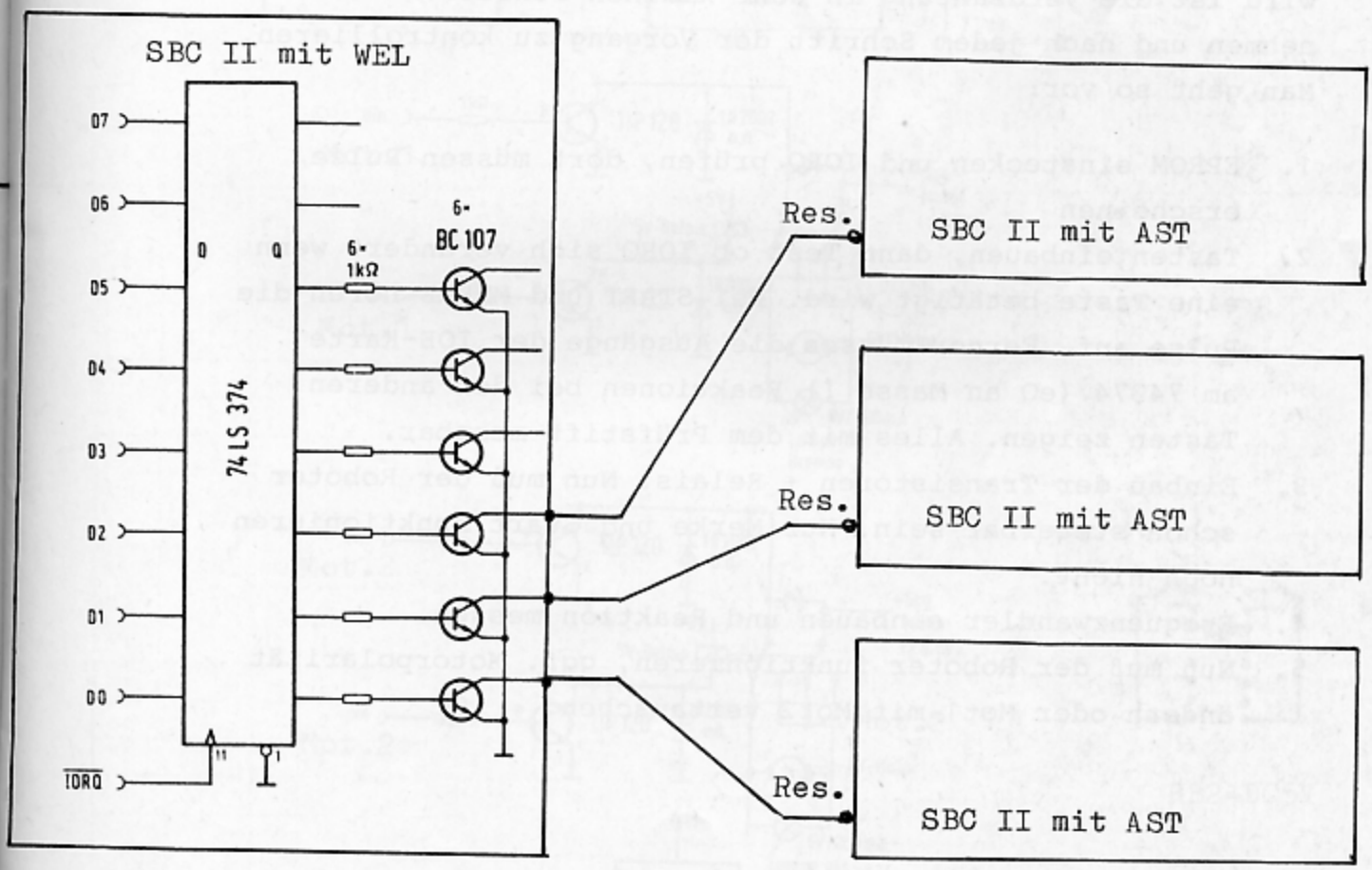
ASCII ist die Abkürzung für American Standard Code for Information Interchange, also amerikanischer Standard-Code für den Informationsaustausch. Es handelt sich um eine 7-Bit-Verschlüsselung für 128 Zeichen, die bei Mikrocomputern allgemein üblich ist. Ein achtes Bit wird oft als Paritätsinformation zur Fehlererkennung hinzugefügt. Dieser Code entspricht dem ISO-7-Bit-Code (internationales Telegraphenalphabet Nr. 5). Zahlreiche Zeichen werden zur Steuerung der Übertragung benutzt (z. B. STX = Start Text, EOT = End of Transmission). In Mikrocomputern verwendet man einige davon, um beispielsweise den Cursor (zeigt auf die Stelle, an der das nächste Zeichen gedruckt werden soll) zu positionieren. Seit kurzem ist auch Funkfernschreiben im ASCII-Code erlaubt. Mikrocomputerbesitzer können damit dieses Hobby ohne zusätzliche Code-Umsetzer betreiben. Eine Zusammenstellung der ASCII-Zeichen und ihrer dualen Codierung zeigt die Tabelle.

Die Spalte CNTRL bezeichnet die Taste, die ein bestimmtes Steuerzeichen auslöst, wenn sie gleichzeitig mit der CNTRL-Taste gedrückt wird.

Bit 7 6 5 / 4 3 2 1	000	CNTRL	100	CNTRL	010	110	001	101	011	111
0 0 0 0	NUL	@	DLE	P	SP	0	@	P	`	p
0 0 0 1	SOH	A	DC 1 X-ON	Q	!	1	A	Q	a	q
0 0 1 0	STX	B	DC 2 TAPE	R	"	2	B	R	b	r
0 0 1 1	ETX	C	DC 3 X-OFF	S	#	3	C	S	c	s
0 1 0 0	EOT	D	DC 4 TAPE	T	$	4	D	T	d	t
0 1 0 1	ENQ	E	NAK	U	%	5	E	U	e	u
0 1 1 0	ACK	F	SYN	V	&	6	F	V	f	v
0 1 1 1	BEL	G	ETB	W	'	7	G	W	g	w
1 0 0 0	BS	H	CAN	X	(	8	H	X	h	x
1 0 0 1	HT	I	EM	Y	)	9	I	Y	i	y
1 0 1 0	LF	J	SUB	Z	*	:	J	Z	j	z
1 0 1 1	VT	K	ESC	[	+	;	K	[	k	{
1 1 0 0	FF	L	FS	/	,	<	L	\	l	|
1 1 0 1	CR	M	GS	]	-	=	M	]	m	} ALT MODE
1 1 1 0	SO	N	RS	^	.	>	N	^	n	~
1 1 1 1	SI	O	US	-	/	?	O	_ ←	o	DEL RUB OUT



Schaltung zur Ampelsteuerung



Schaltung "grüne Welle"

2.7 Roboter steuern

Aufbau eines kleinen lernfähigen Roboters.

Material:

1x Bausatz ROBOTER (incl. fertiger Leiterplatte)

oder

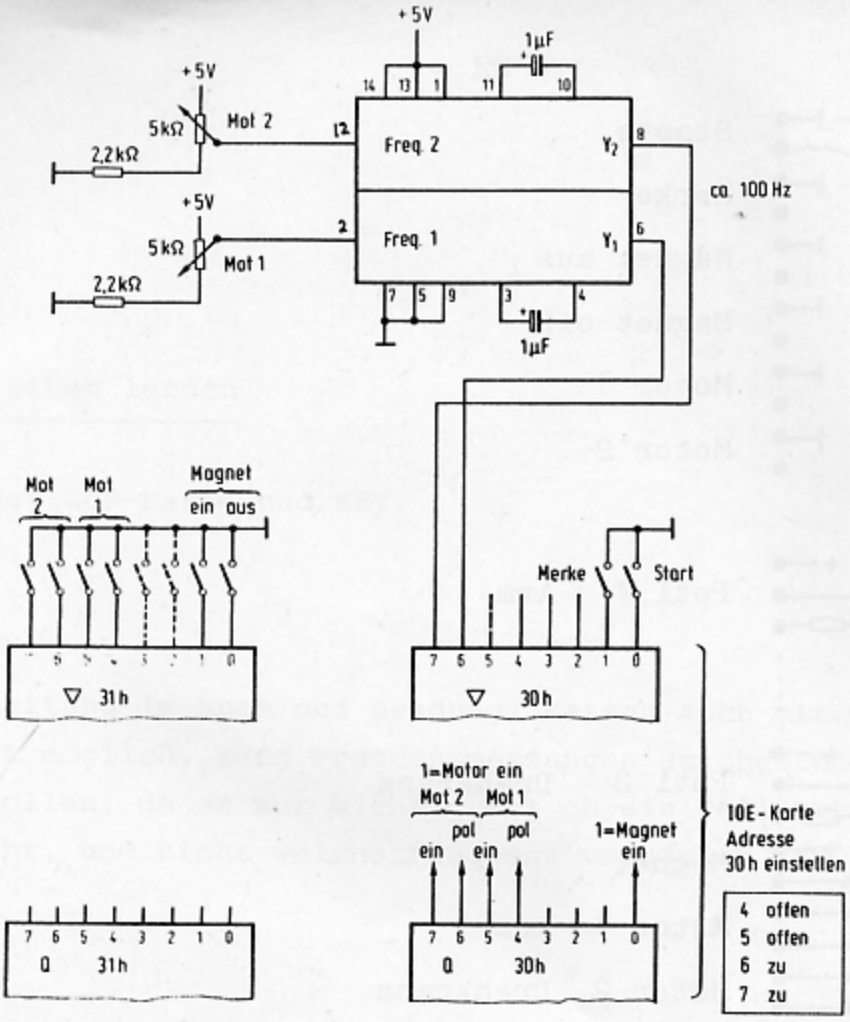
1x IOE-Bausatz + Material Roboter

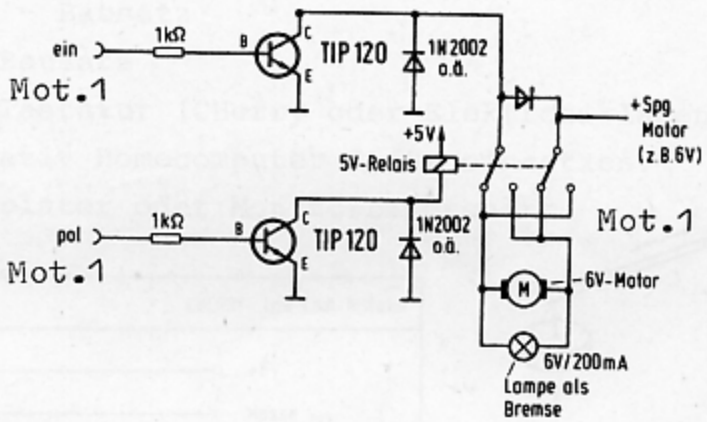
1x Fischertechnik Baukasten (Roboter, Digitizer...)
1x EPROM ROB

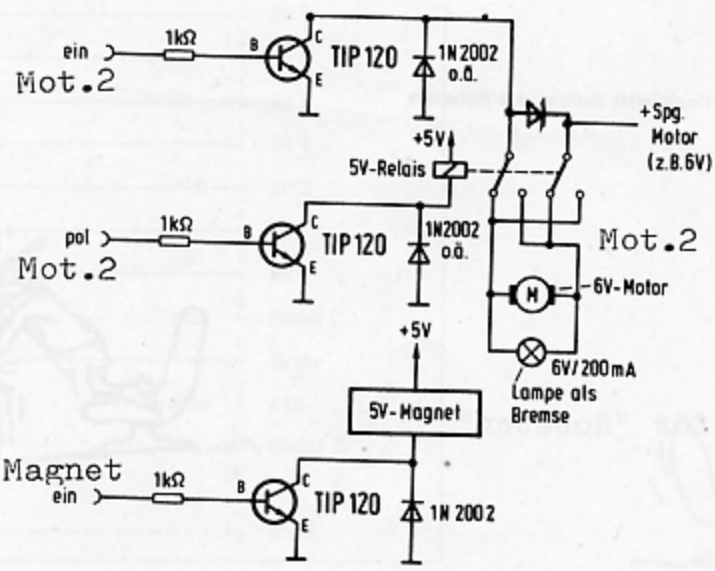
zu beachten:

wenn nicht die fertige gelayoutete Leiterplatte verwendet wird ist die Verdrahtung in sehr kleinen Schritten vorzunehmen und nach jedem Schritt der Vorgang zu kontrollieren. Man geht so vor:

1. EPROM einstecken und IORQ prüfen, dort müssen Pulse erscheinen
2. Tasten einbauen, dann Test ob IORQ sich verändert wenn eine Taste betätigt wird. Bei START und MERKE hören die Pulse auf. Ferner müssen die Ausgänge der IOE-Karte am 74374 (eO an Masse !) Reaktionen bei den anderen Tasten zeigen. Alles mit dem Prüfstift messbar.
3. Einbau der Transistoren + Relais. Nun muß der Roboter schon steuerbar sein. Nur Merke und Start funktionieren noch nicht.
4. Frequenzwandler einbauen und Reaktion messen.
5. Nun muß der Roboter funktionieren, ggf. Motorpolarität ändern oder Mot1 mit Mot2 vertauschen.

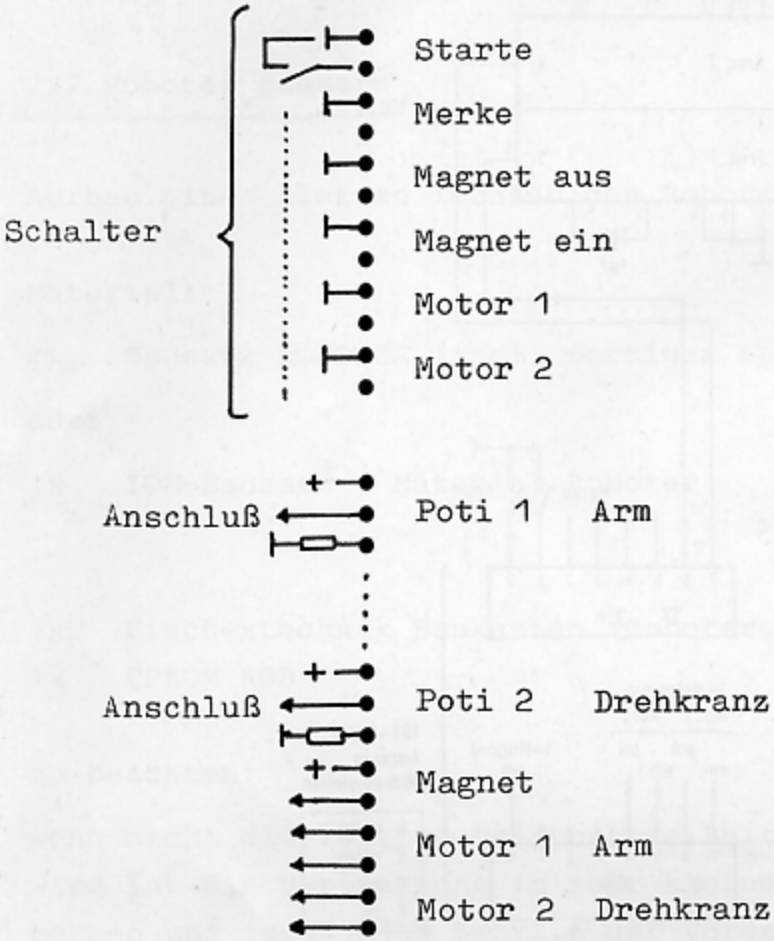


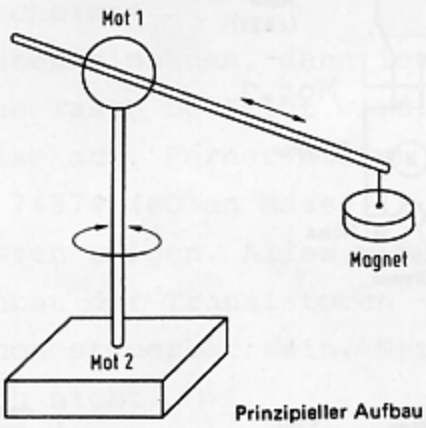




HB2-DC5V

Roboter





Prinzipieller Aufbau des Roboters

Anschlußbelegung für "Roboter"

2.8 Schreiben lernen

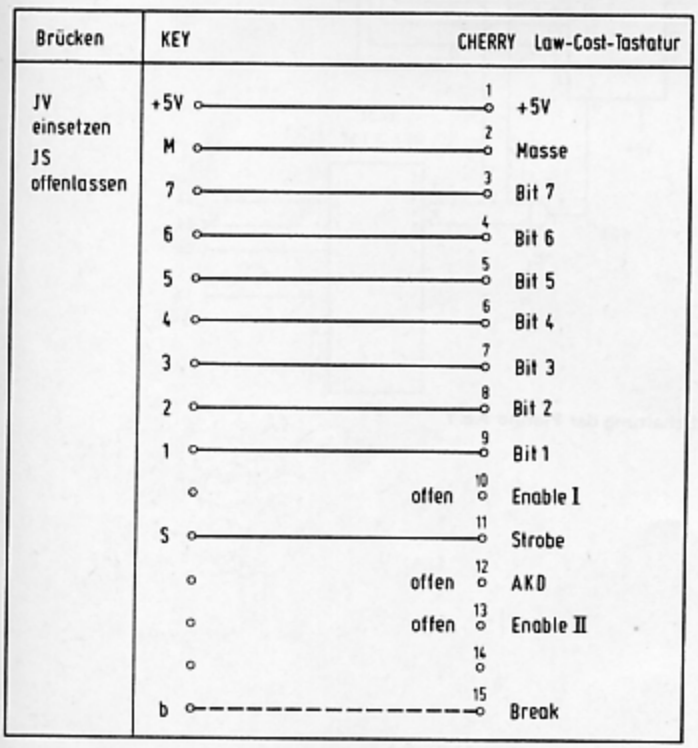
Aufbau der GDP-Karte und KEY.

Messung:

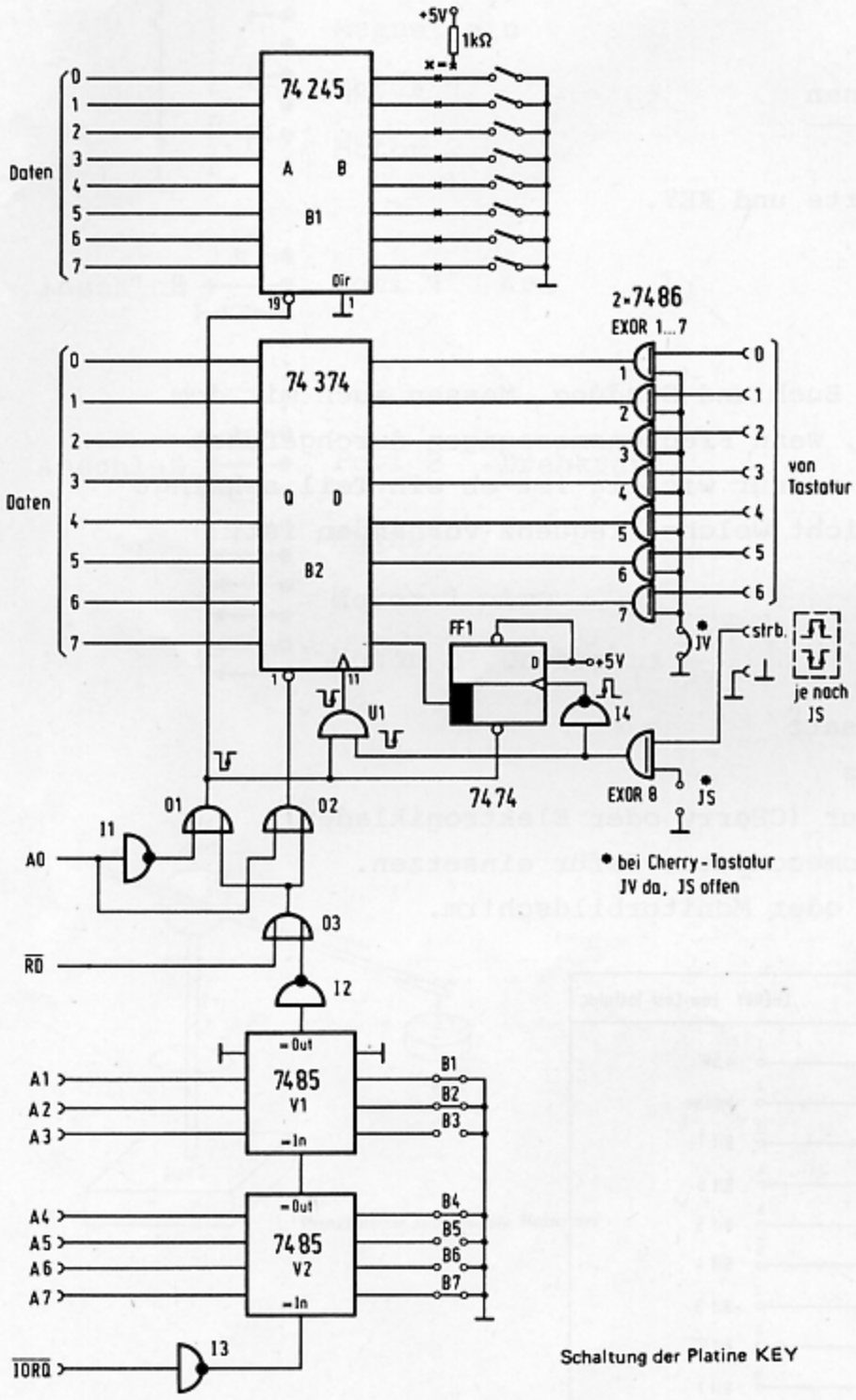
nach Anleitung im Buch und Sendung. Messen auch mit dem Prüfstift möglich, wenn Frequenzmessungen durchgeführt werden sollen, da es nur wichtig ist ob ein Teil schwingt oder nicht, und nicht welche Frequenz vorhanden ist.

Material:

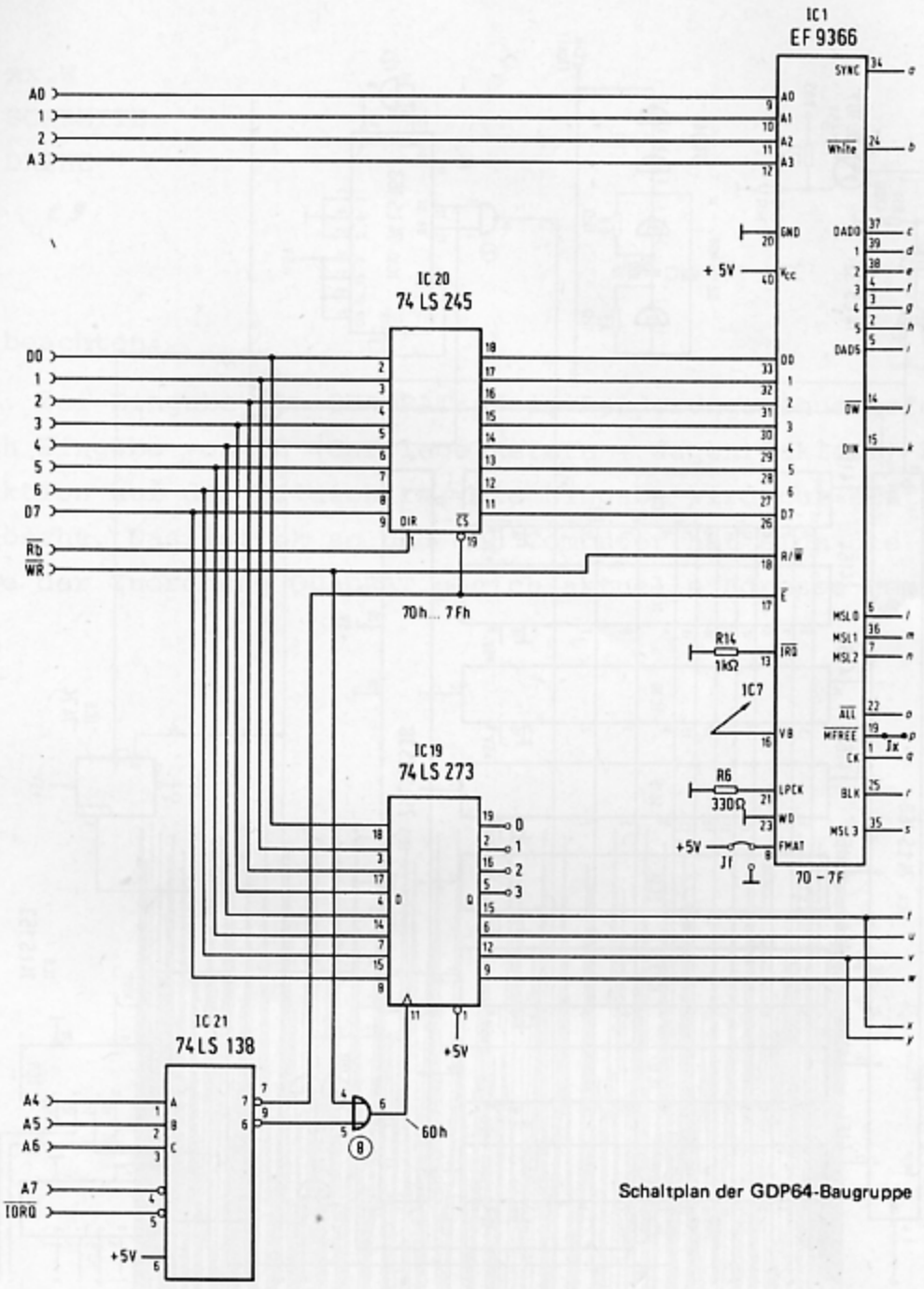
1x GDP 64 - Bausatz
1x KEY - Bausatz
1x ASCII-Tastatur (CHerry oder Elektronikladen)
alternativ Homecomputer dafür einsetzen.
1x HF-Modulator oder Monitorbildschirm.



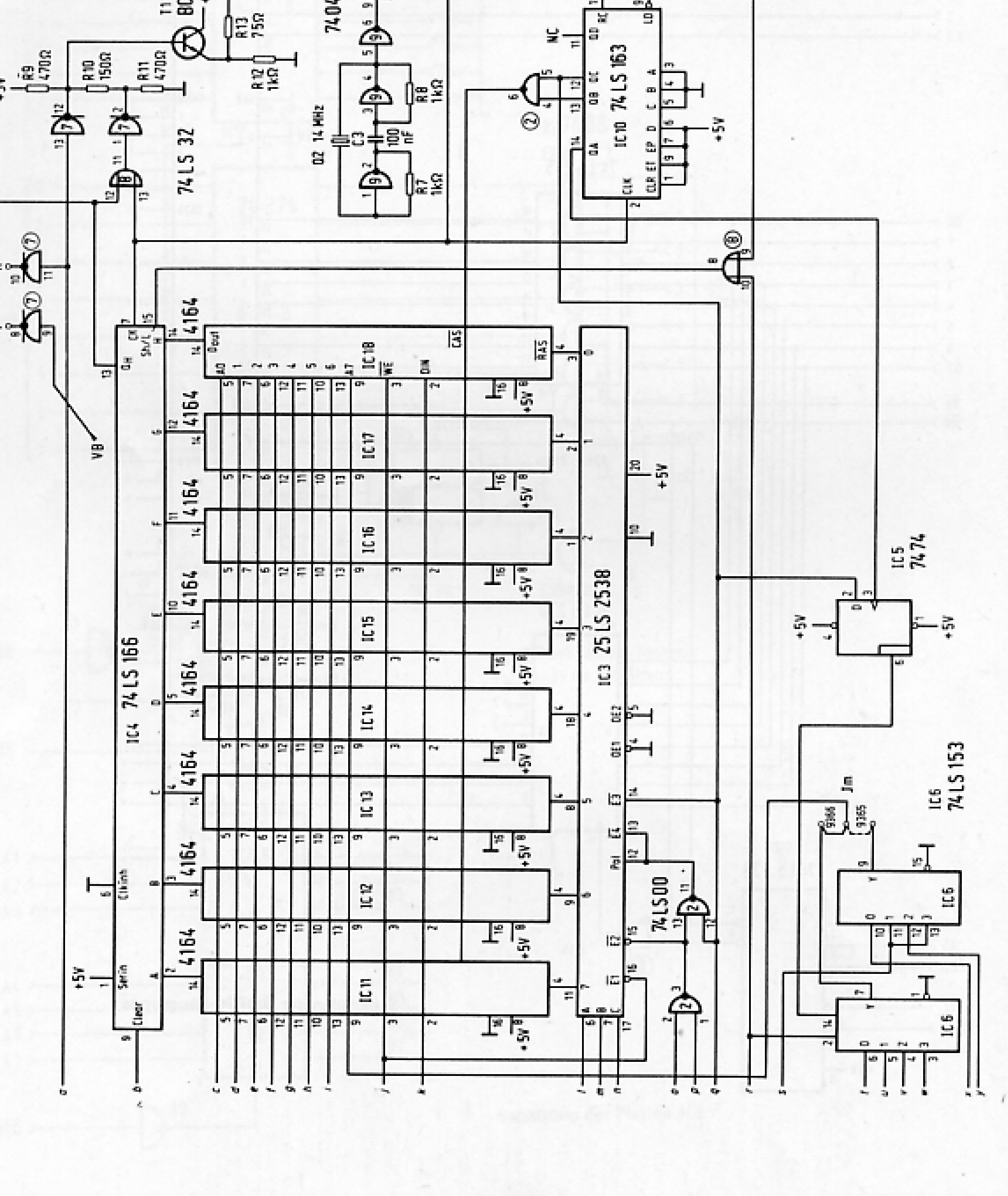
Verbindung zwischen KEY und Tastatur



Schaltung der Platine KEY



Schaltplan der GDP64-Baugruppe


Video (D)
BC107
T1
BAS
R9 470Ω
R10 150Ω
R11 470Ω
R12 1kΩ
R13 75Ω
74LS32
7404
Q2 14 MHz
C3 100 nF
R7 1kΩ
R8 1kΩ
Clk (Takt)
IC10 74LS163
NC
+5V
IC4 74LS166
4164
IC11
IC12
IC13
IC14
IC15
IC16
IC17
IC18
IC3 25LS2538
74LS00
IC5 7474
IC6 74LS153
VB

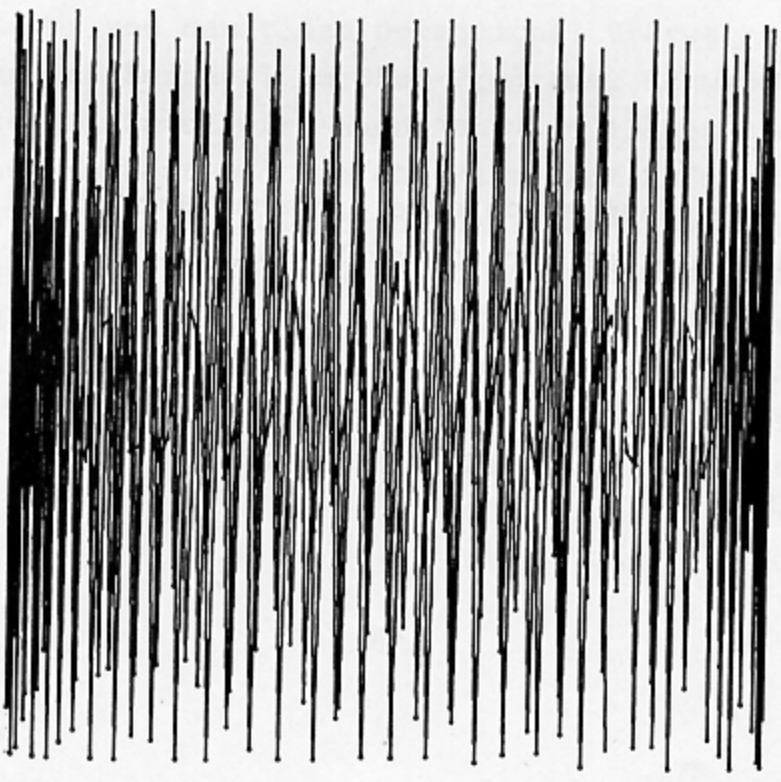
2.9 Zeichen-Sprache

-- nur Software --
Versuche mit den Befehlen

21 xx.W
CD SCHREITE
CD DREHE
C$ C9

zu beachten:

z.B. bei Eingabe von QUADRAT:=$ im Aenderungsmenue erfolgt nach Eingabe von CR (Carriage Return = Wagenrücklauf) keine Reaktion auf dem Bildschirm, die Eingabe wird nur einfach gelöscht. Das ist ok so und der Computer hat sich die Eingabe der Zuordnung QUADRAT gleich aktuelle Adresse gemerkt.



2.10 Blumen mit Schleife

-- nur Software --
Neue Befehle
CD SCHLEIFE
CD ENDSCHLEIFE

Literatur:

In käuflichen LOGO-Büchern findet man zahlreiche Ideen zur Programmierung von weiterer Graphik.
ggf. benötigt man noch weitere Z80 Befehle, wie z.B.
2A xx.W Lade von Speichern
22 xx.W Speichere nach Speicher xx ist die Adresse.

Hinweis:

mit den Zuordungen LADE:= 21 RUFE:CD und ENDE:=C9
kann man danach auch eingeben

```
LADE 50
RUFE SCHRIFTE
ENDE
```

Der Computer kann dann schon eine einfache Sprache direkt verstehen, hier gibt es zahlreiche Möglichkeiten für eigene Ideen.

2.11 Statt Musik gibt's Daten

Aufbau einer Kassettenschnittstelle

Messung:

Prüfstift und EPROM SKOP. Das Eprom 2716 wird anstelle des letzten RAM-Bausteins (IC9, nr3) eingesteckt. Dann wird beim Startmenue die Adresse 8800 eingegeben. Es meldet sich das Skop-Menue. Eine IOE-Karte wird benötigt und auf Adresse 30H eingestellt.

1. Messung: An Bit O I/O O des 74LS245 wird der Ausgang c des 6850 (Pin 4) angeschlossen und das Menue 1) aufgerufen. Damit wird die Periodendauer gemessen. Zeigt sich keine Schwingung auf dem Bildschirm, so liegt kein Takt an. Wenn eine Schwingung angezeigt wird, so wird die Periodendauer in ys ausgegeben. Nun dreht man am Trimmer Tr1 solange bis eine Periodendauer von 833ys (+5ys) angezeigt wird.

2. Messung: An Bit O I/O O des 74LS245 auf der IOE-Karte wird an Punkt b (Pin 3/6850) angeklemmt und damit der RCLK-Eingang gemessen. Bit 1 des 74LS245 wird an Punkt a (Pin 2 6850) angeklemmt und damit das Datensignal überwacht. Nun wählt man das Menue 2) im Skop-Programm. Es werden beide Kanäle angezeigt. Zuvor wird nach Abb. 5.6.12 des Buches die Testschaltung angebaut und die CAS-Karte auf Aufnahme geschaltet. Es erscheinen Pulsgruppen auf dem Bildschirm. Oben wird der RCLK-Eingang angezeigt und unten die Daten. In ys wird die O-Signaldauer des RCLK-Eingangs angezeigt. Nun kann man mit dem Trimmer TR2 diese Zeit auf 3/4 der Periodendauer, also 625ys eingestellt. Die Genauigkeit der Messung beträgt nur etwa 5ys, daher kann man den Wert nicht

exakt erreichen, was jedoch nicht sehr kritisch ist.

Material:

1x CAS-Bausatz

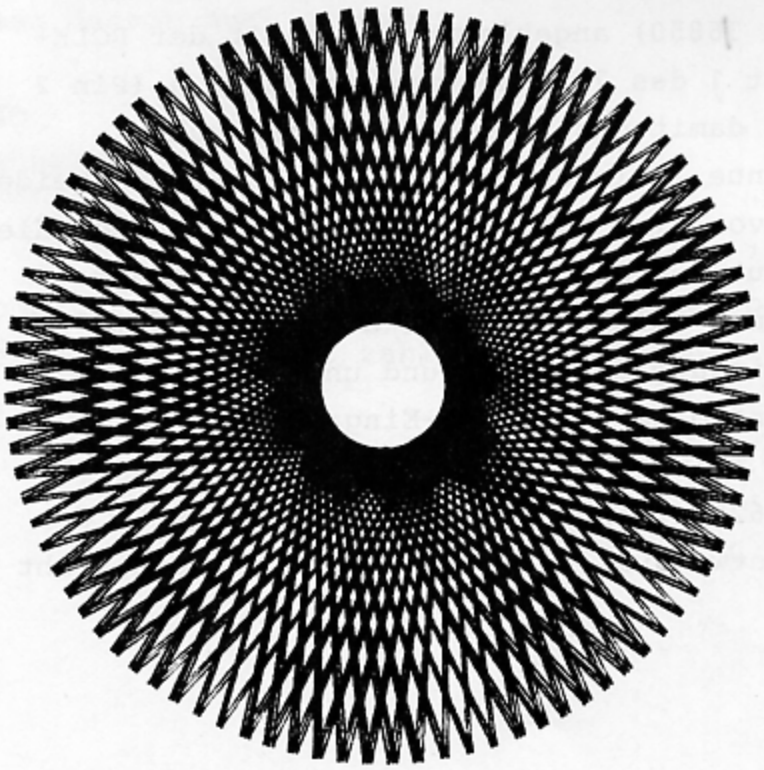
1x Kassettenrekorder MONO, ohne Sprachfilter.
Beim Kauf auf Rückgaberecht achten und erst mal ausprobieren. Nicht alle Rekorder sind geeignet.
Der Rekorder darf keine Phasenverzerrungen verursachen, z.B. aufgrund von speziellen Filtern (Sprache etc.)

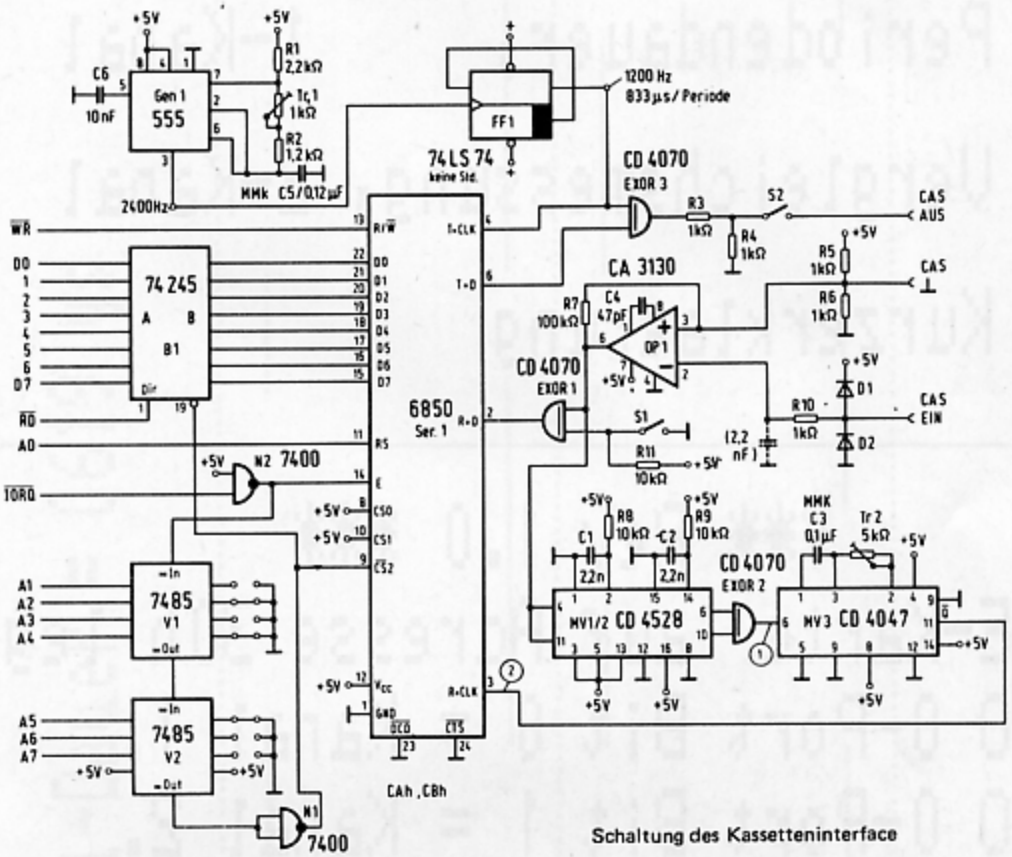
1x Kondensator 100nf für Teststecker

1x Widerstand 1kOhm

2x Schalter für Aufnahme und Polarität

1x Tonband-Buchse passend zum Überspielkabel





Schaltung des Kassetteninterface

Für 2400 Baud gelten folgende Bauteilewerte:

C_3 47nF

Tr_1 2kOhm (oder R_2 angleichen)

C_1 47nF

C_1 330pF

C_2 330pF

ROK-Digital-Scop

1 = Periodendauer, 1-Kanal

2 = Vergleichsmessung, 2-Kanal

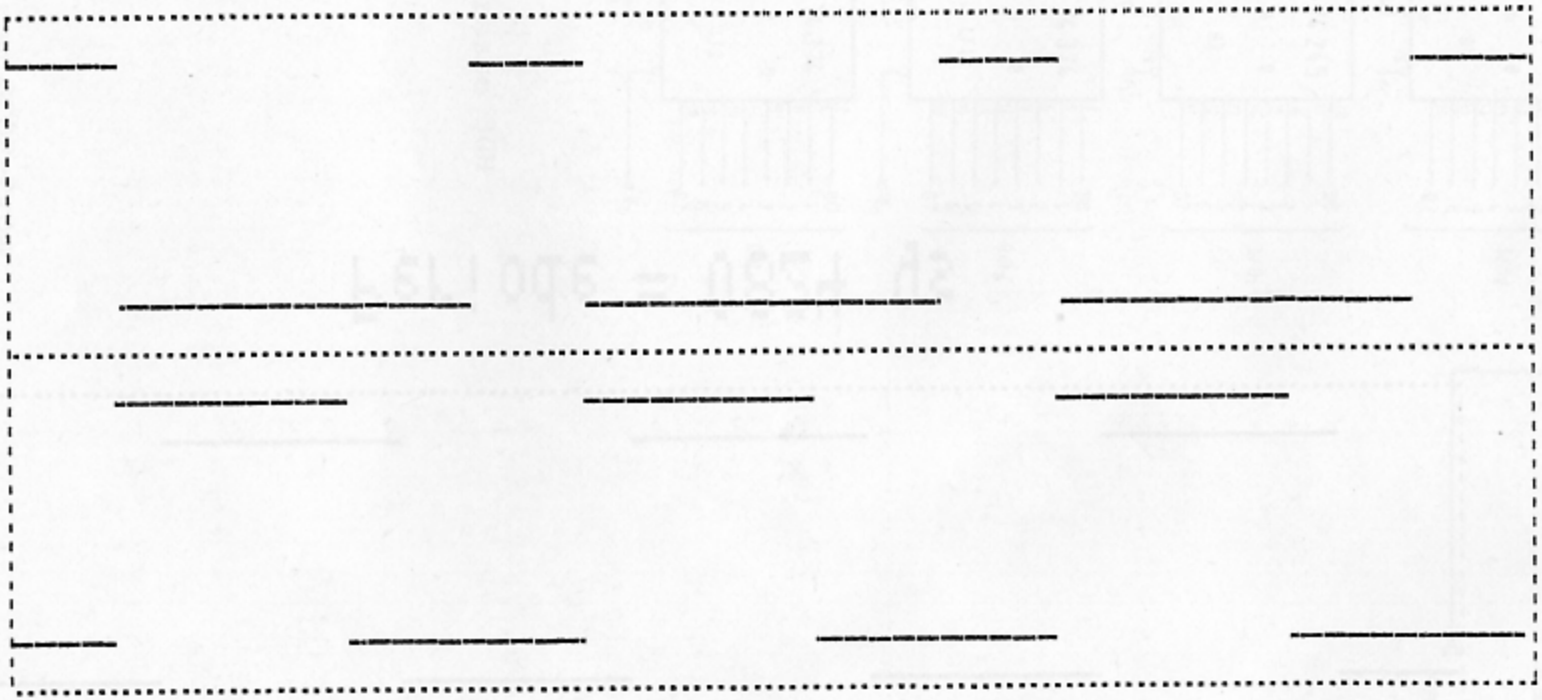
3 = Kurzerklaerung

*** Rev 1.0 ***

1. IOE-Karte auf Adresse 30h legen.
2. I/O 0-Port Bit 0 = Kanal 1.
3. I/O 0-Port Bit 1 = Kanal 2.
4. Kanal 1 ist auch Triggereingang.
5. Das Signal erscheint erst nachdem ein Signalwechsel am Triggereingang von 0 auf 1 erfolgt.
6. Die Messungen erfolgen auf ca. 5ys bis 6ys genau.

M=Menue

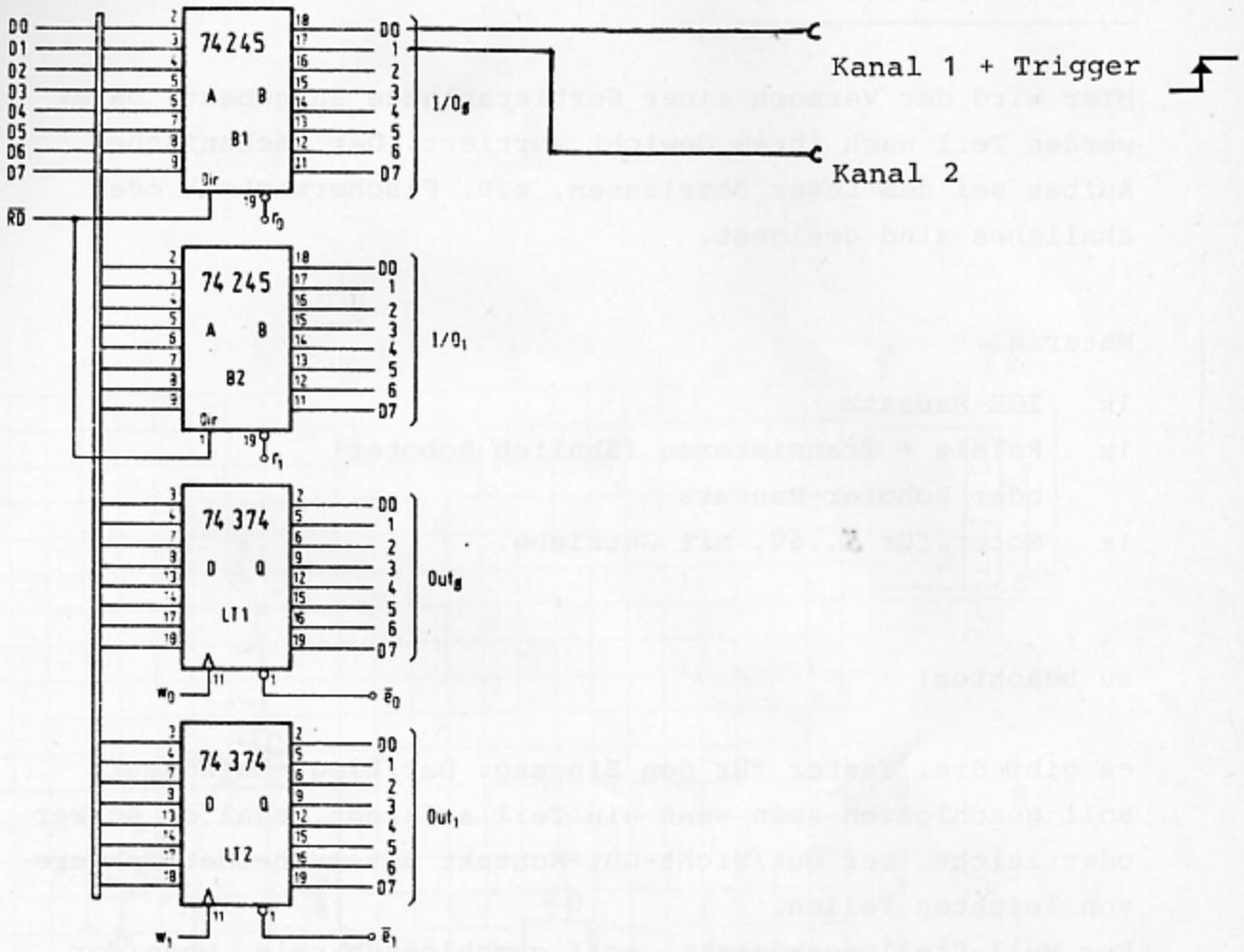
0-Signal-Dauer = 0619 µs



S=Speichern W=Weiter M=Menue

Periode = 0824 ys

S=Speichern W=Weiter M=Menue



IOE auf Adresse 30H

Anschluß zum SCOP-Prom

2.12 Gut und Schlecht

Hier wird der Versuch einer Sortieranlange aufgebaut. Dabei werden Teil nach ihrem Gewicht sortiert. Der mechanische Aufbau sei dem Leser überlassen, z.B. Fischertechnik oder ähnliches sind geeignet.

Material:

1x IOE-Bausatz
1x Relais + Transistoren (ähnlich Roboter)
oder Roboter-Bausatz
1x Motor für 5..6V, mit Getriebe.

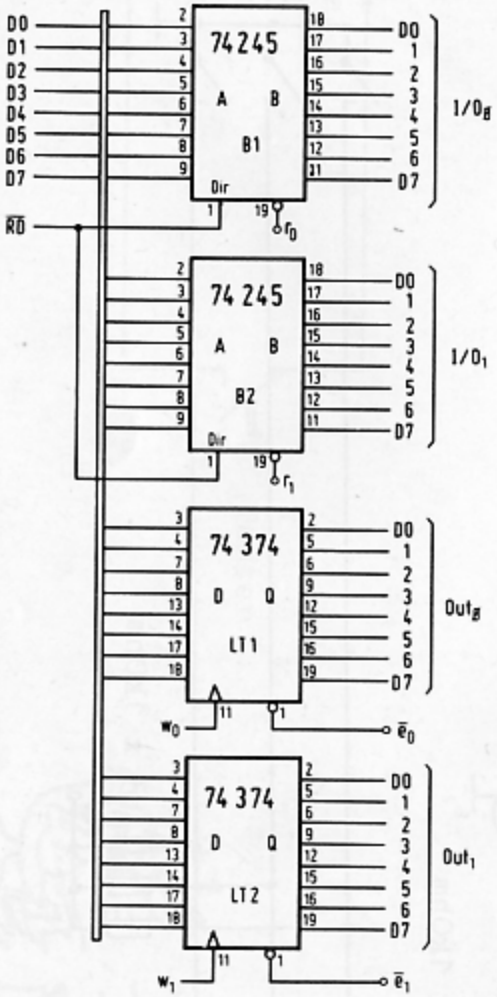
zu beachten:

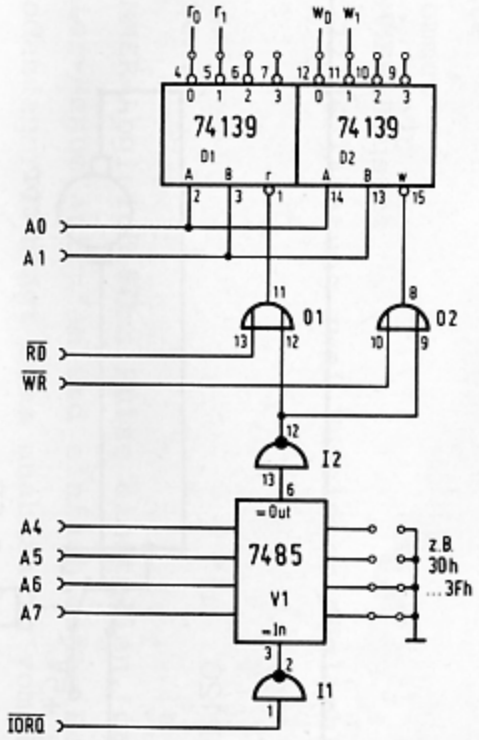
es gibt drei Taster für den Eingang. Der Liegt-Aufkontakt soll geschlossen sein wenn ein Teil aufliegt, egal ob schwer oder leicht. Der Gut/Nicht-Gut-Kontakt unterscheidet schwere von leichten Teilen.
Der Null-Stellungskontakt soll geschlossen sein, wenn der Wiege-Arm die Nullstellung erreicht hat.
Die Motorpolarität spielt bei der im Fernsehn gezeigten Anordnung keine Rolle, es wird dadurch nur bestimmt, ob gute Teile nach Rechts oder Links ausgeworfen werden.

-- Softwarehinweis
Nun werden vom Z80 weitere Befehle gelernt. Man kann von hier aus in die Z80-Programmierung vertieft einsteigen.

-

Das Programm muß zweimal eingegeben werden, da erst beim zweiten Mal Vorwärtsmarken bekannt sind, beim ersten Mal werden dafür O.W eingegeben.





Schaltplan der Universal- IO-Karte

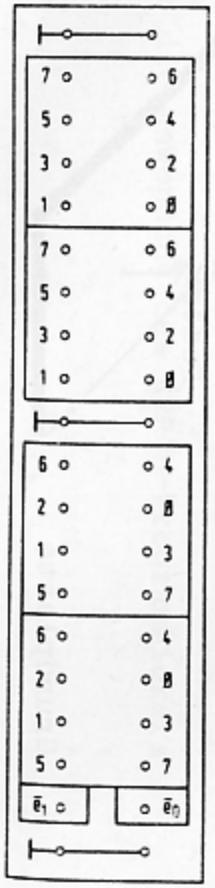
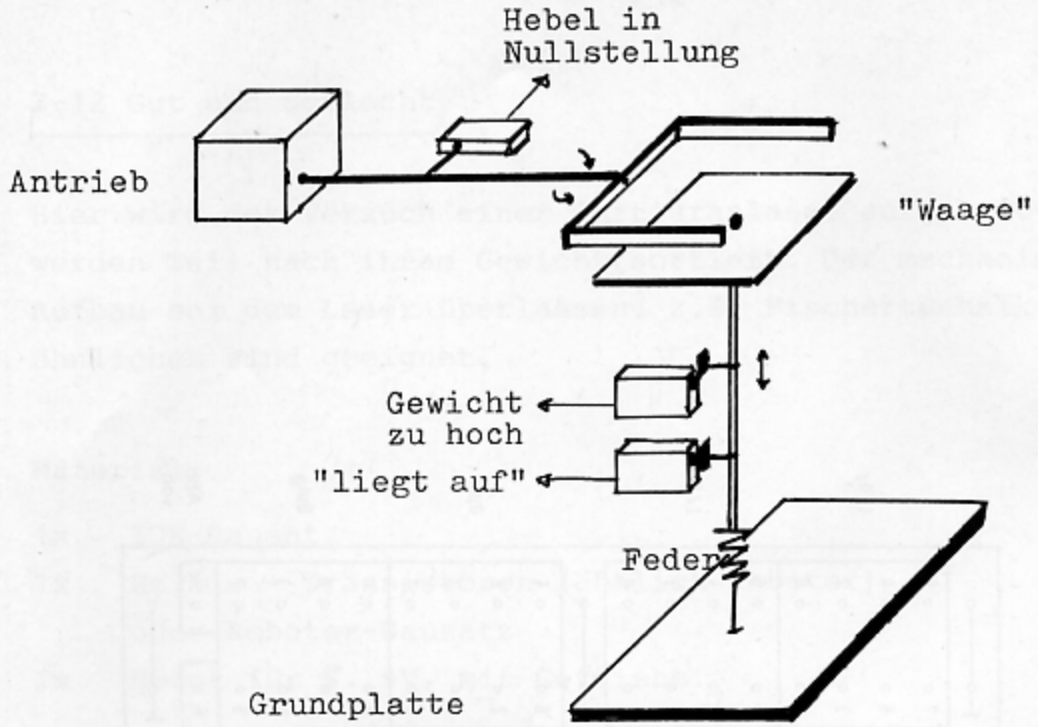
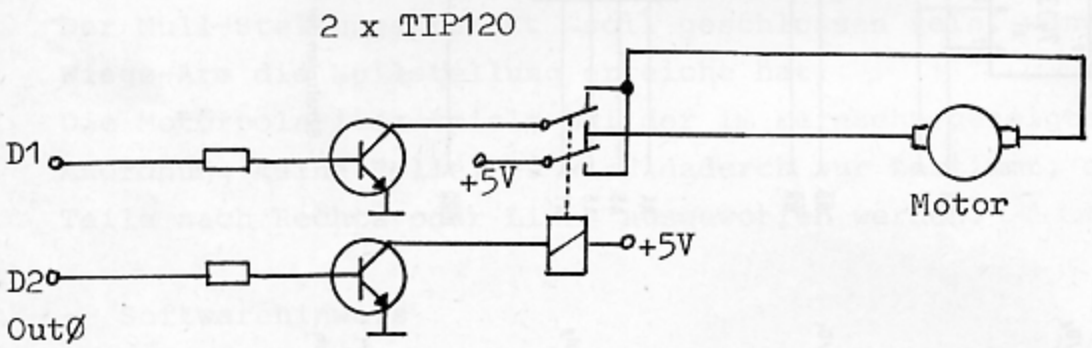
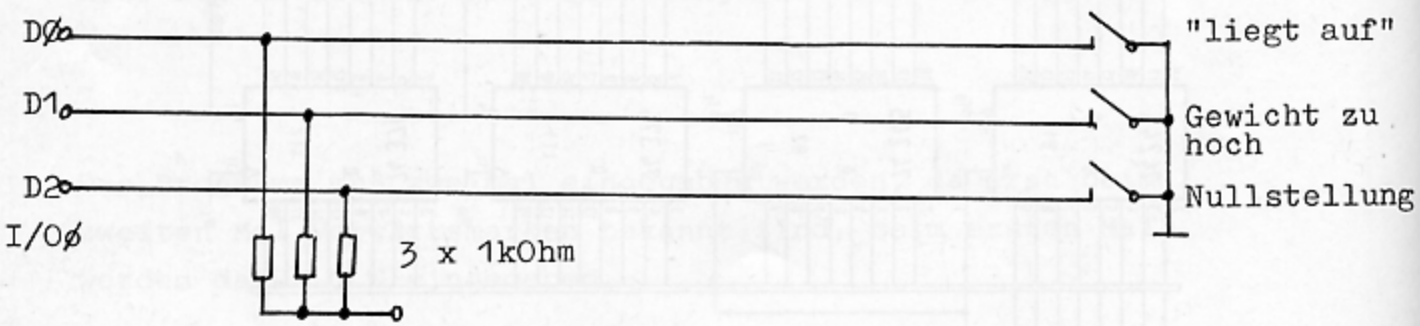


Abb.
von c





2 x 1kOhm



Versuch "Waage"

2.13 Schrecksekunde

Hier soll mal auch ein anderer Prozessor gezeigt werden. Die Versuche lassen sich aber auch mit dem Z80 durchführen.
Dazu wird eine Taste mit dem INT-Eingang der SBCII verbunden.
Das Programm kann man anhand des Listings eingeben. Achtung alle Adressen, also dort wo zwei Bytes direkt nebeneinander stehen mit xxxx.W eingeben, daß .W nicht vergessen.

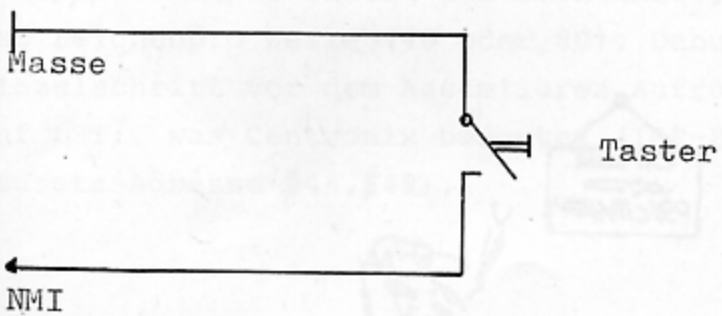
Ferner wird hier die Anwendung des EPROM-Programmierers gezeigt.

Material:

1x Taste
- für Prommer
1x PROMMER-Bausatz
1x POW 22/26 (Achtung zwei Versionen möglich)

zu beachten:

Beim PROMMER-Abgleich 50ms Pulse einstellen, dazu das EPROM-Programmier-Menue aufrufen und einfach einen Bereich angeben. Die Periodendauer beträgt 60ms unabhängig vom Trimmer.



3.1 Mehr Bits

Aufbau der 68008-Baugruppen CPU68K und ROA64.

Material:

1x CPU 68K - Bausatz
1x ROA 64 - Bausatz
1x Grundprogramm 68K in vier EPROMs 2764
1x RAM-Baustein HM 6264
-- besser gleich zwei RAM-Bausteine anschaffen
da sonst zu wenig Platz für Programme.

zu beachten:

Nach dem Start des Prozessors muß sich das Grundprogramm auf dem Bildschirm melden. Das Menue ist ähnlich zum Z80-Menue, jedoch kommen ein EDITOR-Menue, ein ASSEMBLER-Menue und ein BIBLIOTHEKs-Menue hinzu.
Zum Bedienen des Editors ihn einfach mal aufrufen. Mit CTRL-J wird eine Liste der verfügbaren Befehle auf den Bildschirm gebracht.
Den Editor kann man durch die Sequenz CTRL-K Shift-X wieder verlassen. Auf keinen Fall durch RESET, da in diesem Fall das gerade angezeigte Bild verloren geht.

3.2 Sprachprobleme

-- Software

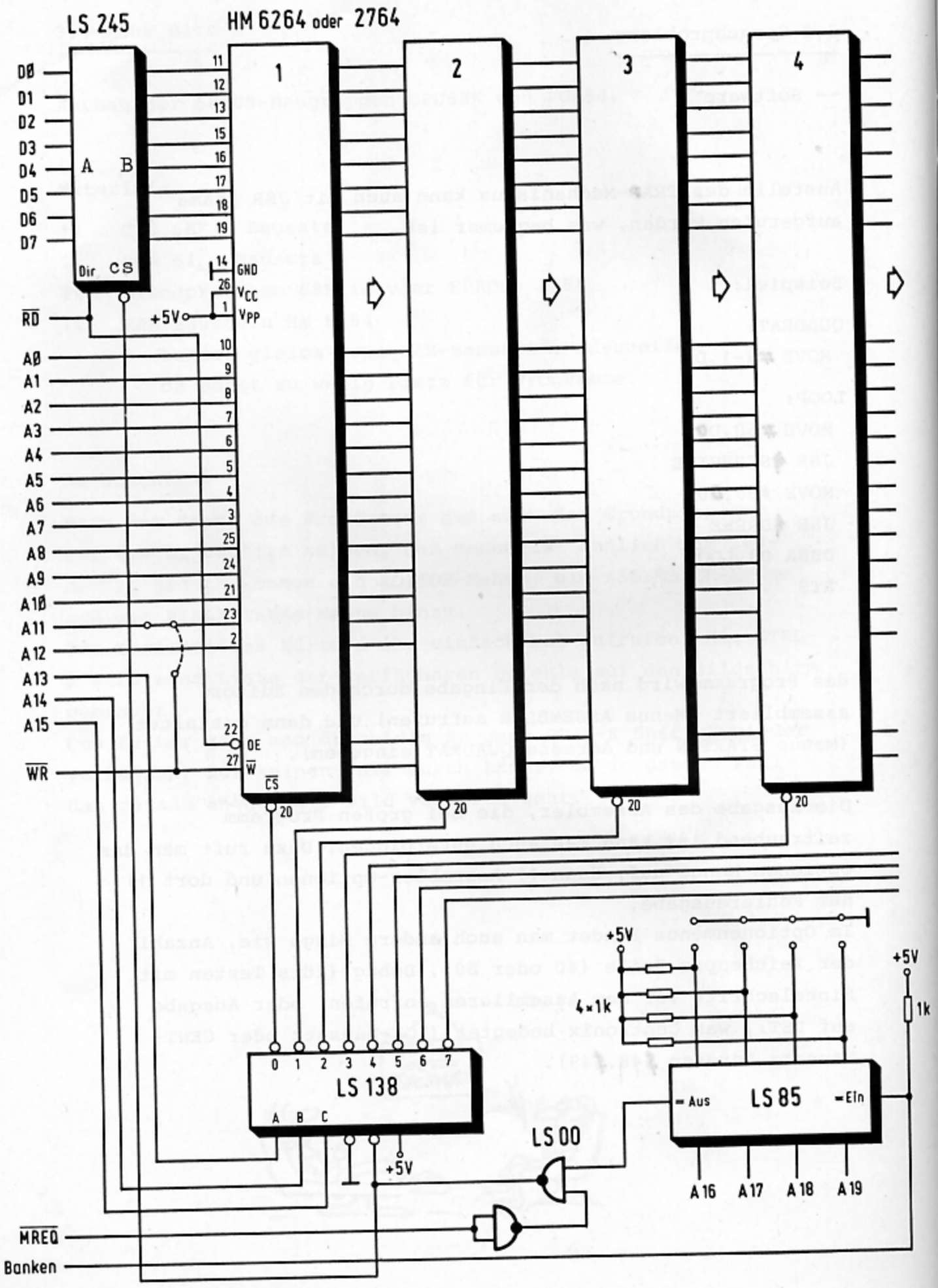
Anstelle des TRAP-Mechanismus kann auch mit JSR §name aufgerufen werden, was bequemer ist.

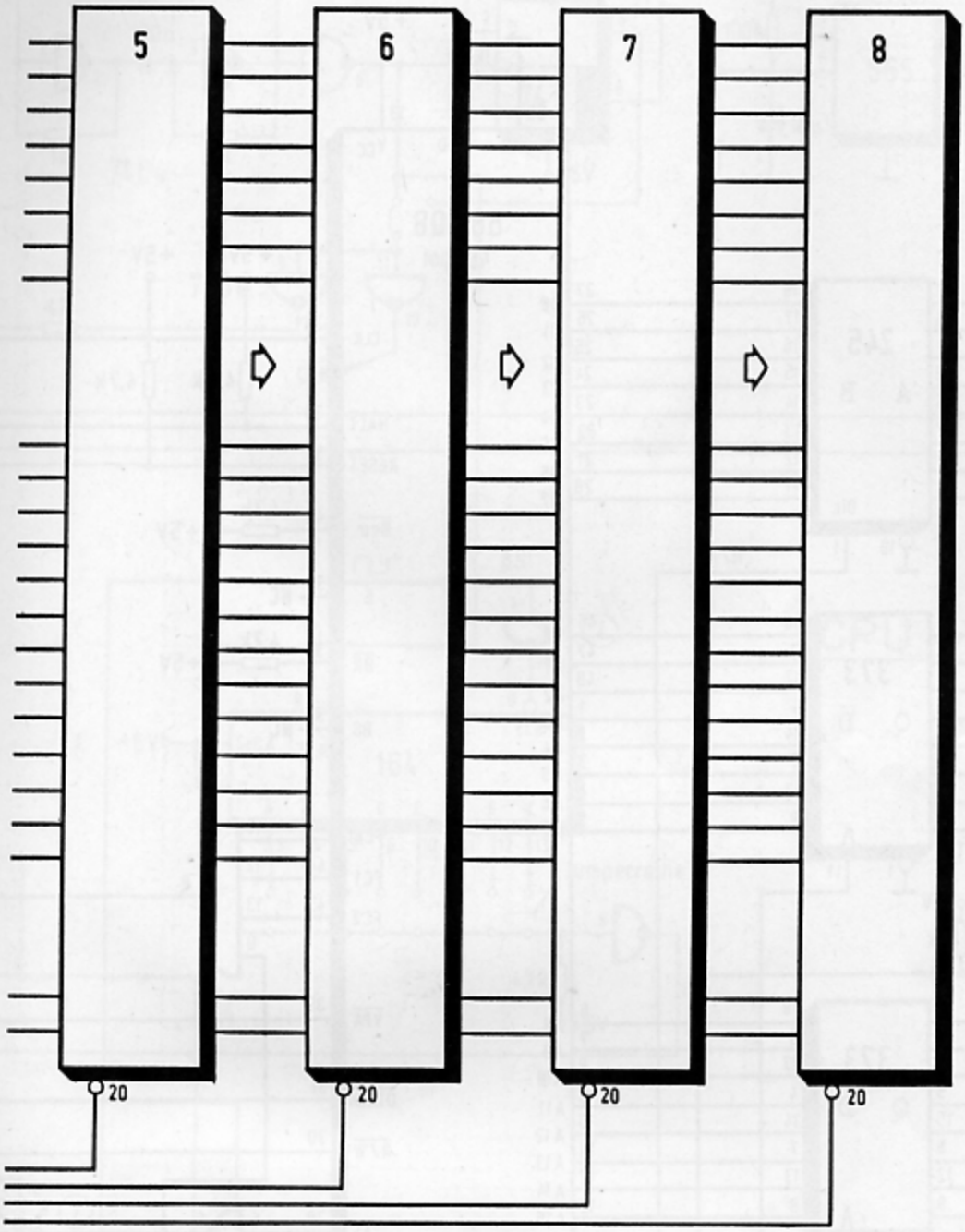
Beispiel:

```
QUADRAT:
 MOVE #4-1,D3
LOOP:
 MOVE #50,D0
 JSR §SCHREITE
 MOVE #90,D0
 JSR §DREHE
 DBRA D3,LOOP
 RTS
```

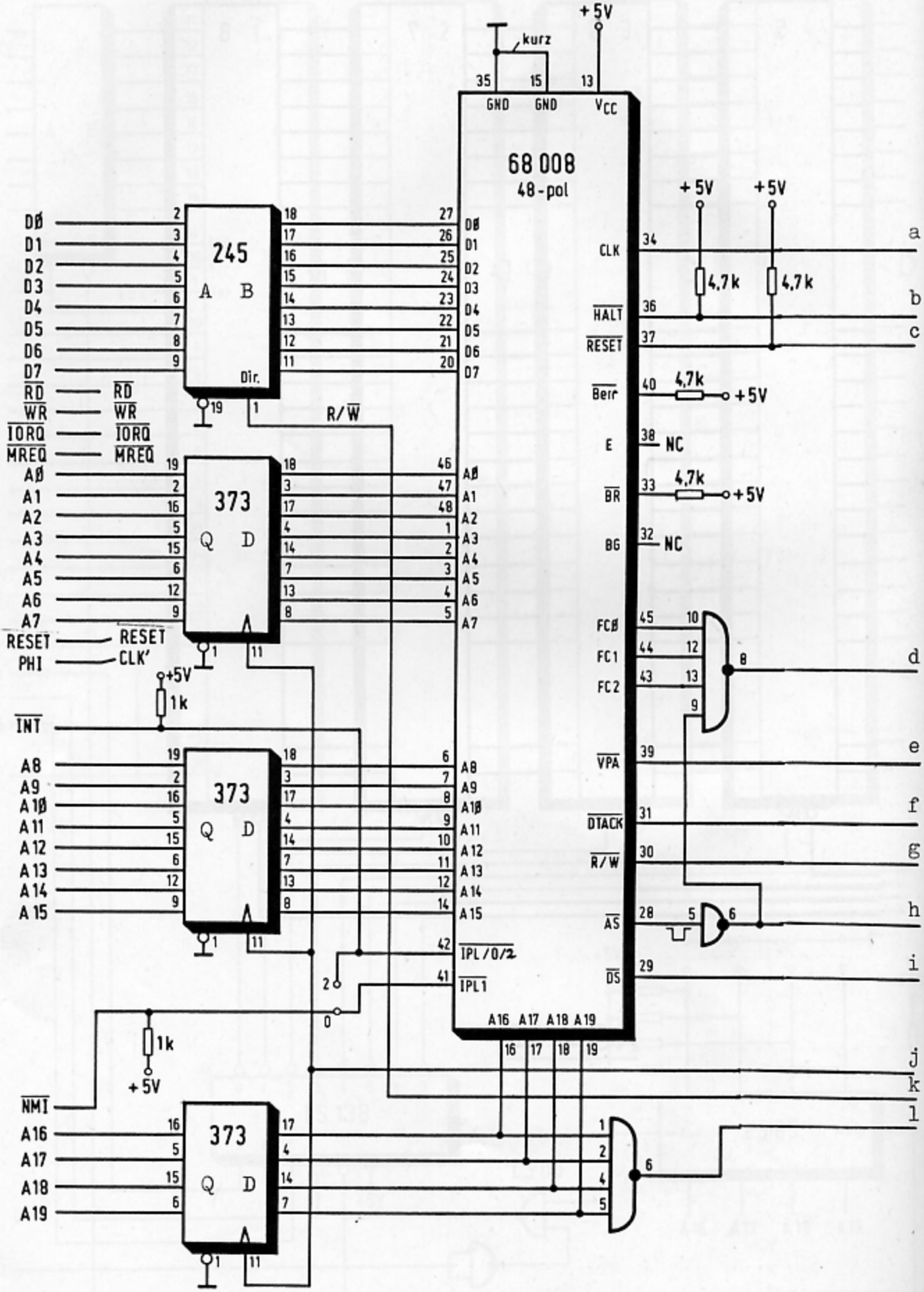
das Programm wird nach der Eingabe durch den Editor assembliert (Menue ASSEMBLER aufrufen) und dann gestartet (Menue STARTEN und Adresse QUADRAT eingeben).

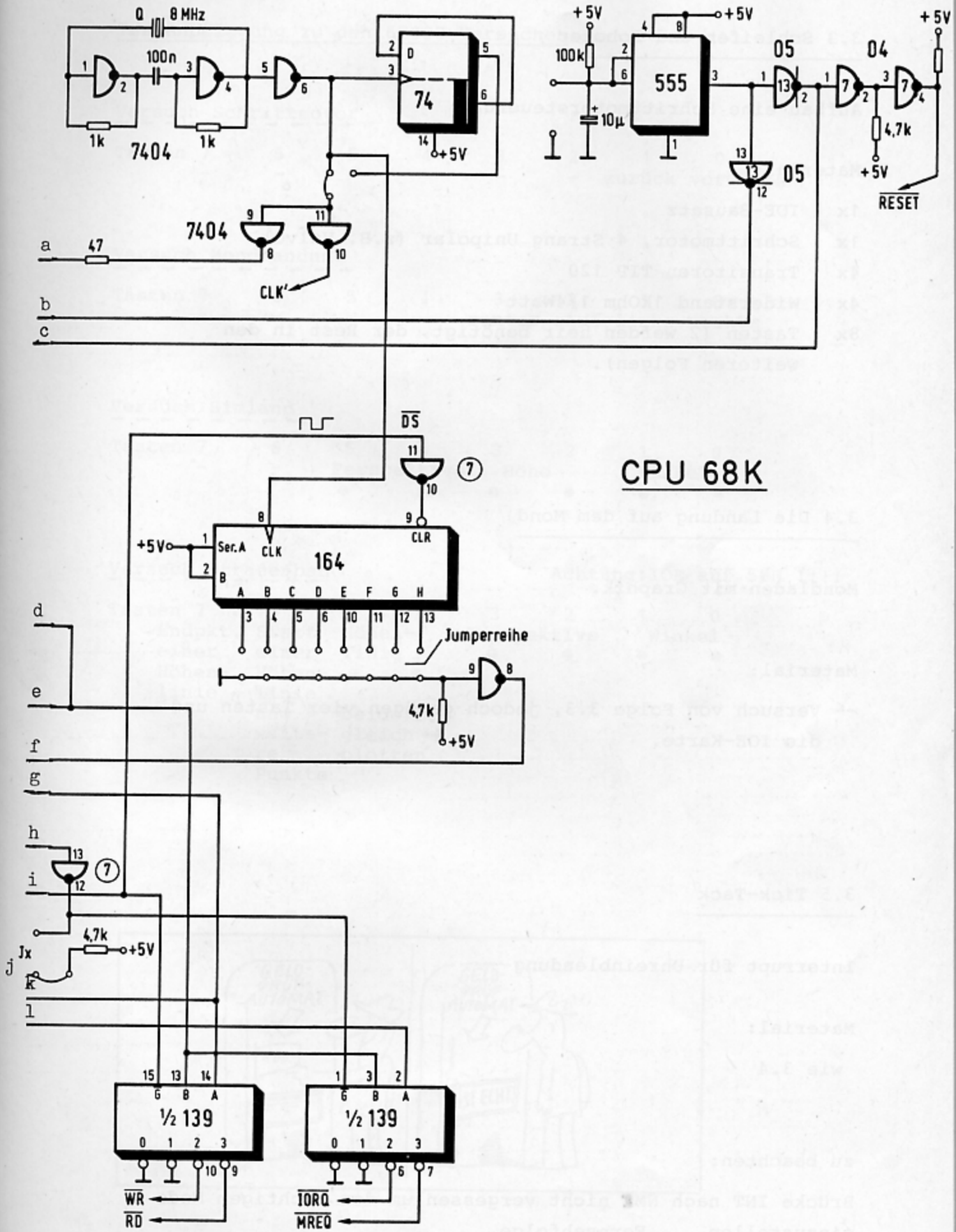
Die Ausgabe des Assembler, die bei großen Programm zeitraubend ist kann man auch unterbinden. Dazu ruft man das Optionen-Menue auf, dann 1)-Assembler-Optionen und dort 1) nur Fehlerausgabe.
Im Optionenmenue findet man auch andere Dinge wie, Anzahl der Zeichenpro Zeile (40 oder 80), Debug (fürs Testen mit Einzelschritt vor dem Assemlieren aufrufen) oder Ausgabe auf LST:, was Centronix bedeutet (IOE-Bausatz oder CENT-Bausatz Adresse §48,§49).


LS 245
HM 6264 oder 2764
D0
D1
D2
D3
D4
D5
D6
D7
A
B
Dir
CS
RD
WR
+5V
GND
VCC
VPP
OE
W
CS
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
1
2
3
4
LS 138
LS 85
LS 00
Aus
Ein
4×1k
1k
A16
A17
A18
A19
MREQ
Banken



ROA 64


+5V
kurz
35 GND
15 GND
13 VCC
68 008
48-pol
245
A B
Dir.
373
Q D
373
Q D
373
Q D
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
IORQ
MREQ
A0
A1
A2
A3
A4
A5
A6
A7
RESET
PHI
CLK'
R/W
INT
A8
A9
A10
A11
A12
A13
A14
A15
NMI
A16
A17
A18
A19
1k
4,7k
CLK
HALT
RESET
Berr
E
BR
BG
FC0
FC1
FC2
VPA
DTACK
R/W
AS
DS
IPL0/2
IPL1
A16 A17 A18 A19
NC
a
b
c
d
e
f
g
h
i
j
k
l


Q
8 MHz
100n
1k
7404
1k
74
+5V
100k
555
10µ
05
04
4,7k
RESET
7404
47
a
CLK'
b
c
DS
7
CPU 68K
+5V
Ser.A
CLK
164
CLR
B
A B C D E F G H
Jumperreihe
d
e
f
g
h
i
Jx
j
k
l
½ 139
WR
RD
IORQ
MREQ

3.3 Schleifen und Roboter

Aufbau eine Schrittmotorsteuerung

Material:

1x IOE-Bausatz
1x Schrittmotor, 4 Strang Unipolar (z.B. Valvo)
4x Transitoren TIP 120
4x Widerstand 1kOhm 1/4Watt
8x Tasten (2 werden heir benötigt, der Rest in den weiteren Folgen).

3.4 Die Landung auf dem Mond

Mondladen mit Graphik.

Material:

-- Versuch von Folge 3.3, jedoch genügen vier Tasten und die IOE-Karte.

3.5 Tick-Tack

Interrupt für Uhreinblendung

Material:

wie 3.4

zu beachten:

Brücke INT nach NMI nicht vergessen um den richtigen Mode einzustellen. Fernsehfolge.

Tastenbelegung zu den 68008-Versuchen

Versuch Schrittmotor

Tasten	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	zurück	vorwärts

Versuch Mondlandung

Tasten	7	6	5	4	3	2	1	0
	-	-	-	-	runter	rauf	links	rechts

Versuch Simland

Tasten	7	6	5	4	3	2	1	0
	-	-	Perspektive		Höhe		Winkel	
			⊖	⊕	⊖	⊕	⊖	⊕

Versuch Straßenbau

Achtung:IOE auf 5ØH !!!!

Tasten	7	6	5	4	3	2	1	0
	Endpkt. einer Höhen-linie	Start einer Höhen-linie und weitere Punkte	Höhen-linie		Perspektive		Winkel	
			⊖	⊕	⊖	⊕	⊖	⊕
			beide zugleich plotten ➡					

3.6 Ausflug nach Simland

3D-Simulation.

Material:

für die Handsteuerung Material IOE + 8 Tasten, wie Folge 3.3.

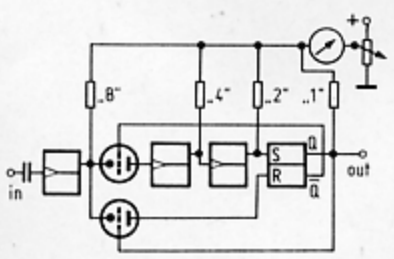
Zum Entspannen ein Artikel aus der Funkschau:

Die verbotene Erfindung

Die Anfänge der Digitaltechnik liegen vielfach im Dunkeln. Einer unserer Leser erinnert sich an ein frühes elektronisches Flip-Flop.

Für den Aufbau eines Kurzzeitmessers wurde 1939 der theoretische Entwurf eines Dualzählers zum Patent angemeldet [1]. Es gelang auch rasch, Flip-Flops mit Röhren und Master-Slave-Schaltungen (diese mit vier Doppelröhren pro Bit!) aufzubauen. Erst später fand man heraus, daß es auch mit weniger Röhren geht, wenn man kleine Kapazitäten als Zwischenspeicher (und Kipphilfe) verwendet.

Als Bitanzeige dienten damals „Magische Augen" und Schauzeichen aus der Fernmeldetechnik. Das war letztlich unbefriedigend, weil jedes Meßresultat aus der binären Anzeige umgerechnet werden mußte. Wo lag die Möglichkeit, sofort eine dezimale Anzeige zu erhalten?



Dezimalschaltung mit Röhren (BCD-Schaltung) nach DRP 761 021 aus dem Jahr 1941

Der Zeitdruck verbot eigentlich weitere Untersuchungen: Es sollten so schnell wie möglich verkaufsreife Geräte entwickelt werden. Doch die fixe Idee bohrte weiter: Man lasse von den 16 möglichen Kombinationen von vier Flip-Flops einfach sechs weg, breche also die Zählung bei „9" ab (heute nennt man das „BCD"). Auf dem Arbeitsplatz entstanden immer umfangreichere Drahtverhaue, entsprechend wuchs das Mißtrauen des Chefs.

Eines Tages war es dann soweit: Der Zeiger des Milliamperemeters rückte mit jedem Impuls ein Stückchen weiter – und fiel beim zehnten Impuls auf Null zurück (*Bild*). Der Weg für eine Neukonstruktion war frei. Natürlich durfte von dieser „verbotenen" Erfindung nichts veröffentlicht werden, die Patentunterlagen [2] blieben im Tresor des Patentamtes (wo sie bei Kriegsende wohl verlorengingen).

Dr. Winfried Wisotzky

Literatur

[1] *Dr. Poleck* (Siemens): Anordnung zur Messung kurzer Zeiten mit sehr hoher Genauigkeit. Patentschrift DRP 745 854 vom Mai 1939.

[2] *Dr. Wisotzky* (Siemens): Einrichtung zum Umwandeln einer gegebenen Folge von elektrischen Impulsen in eine ebensolche Folge von einem Zehntel der ursprünglichen Häufigkeit. Patentschrift DRP 761 021 vom September 1941.

3.7 Krater in PASCAL

PASCALS-Cpmpiler.

Material:

1x ROA64
1x PASCALS in 4 EPROMs
4x RAM-Bausteine HM 6462

zu beachten:

siehe Sonderdruck PASCALS-Programm.

3.8 Die Türme von Hanoi

-- wie 3.7

3.9 Taschenrechner selbst gebaut

-- wie 3.7

3.10 Straßenbau

Koppeln zweier Rechner.

Material:

1x 68008 Vollausbau
(CPU68K+ROA64(3..4RAMs)+IOE+IOE+IOE+BUS+BUS)
1x Z80-System (SBCII+IOE+IOE)

zu beachten:

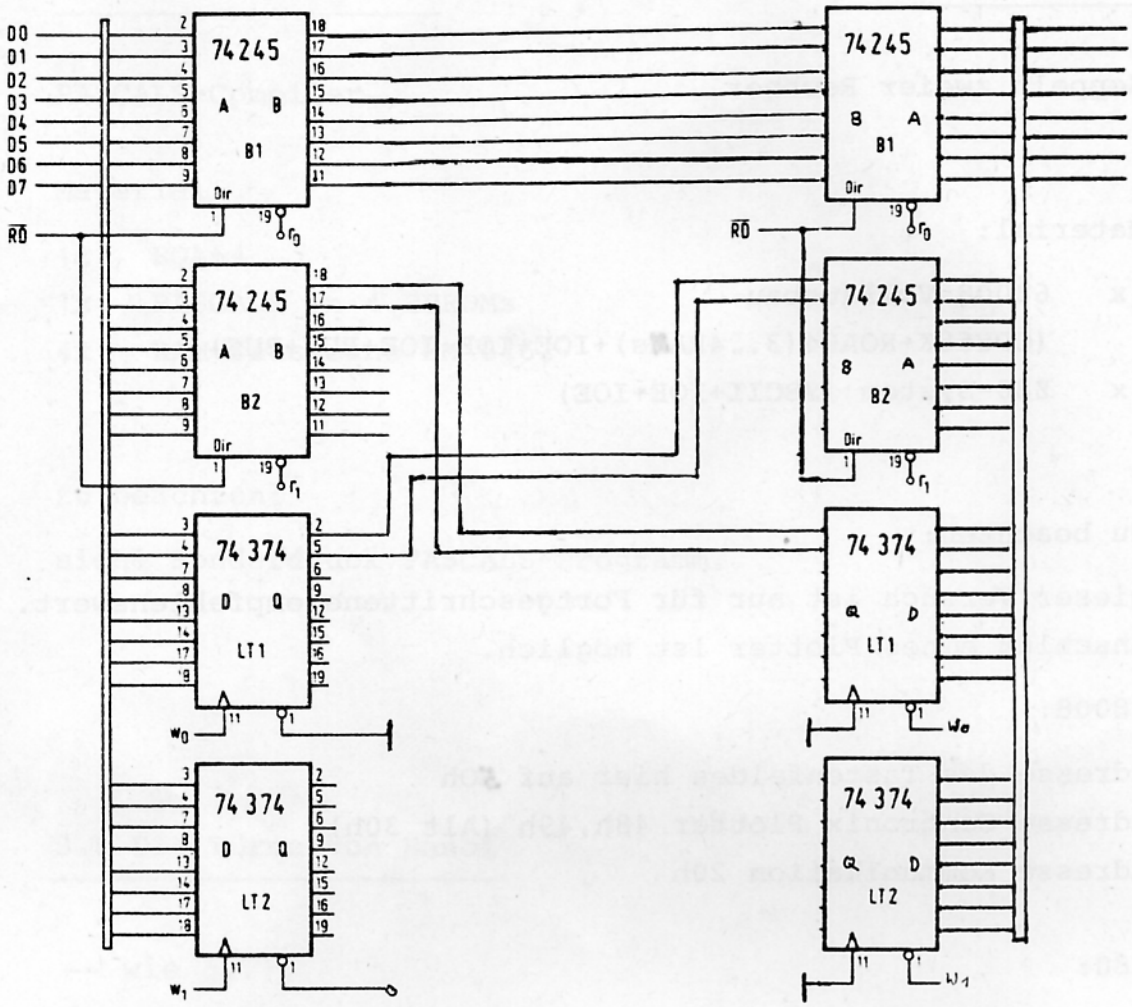
Dieser Versuch ist nur für Fortgeschrittene empfehlenswert.
Anschluß eines Plotter ist möglich.

68008:

Adresse des Tastenfeldes hier auf 50h
Adresse Centronix Plotter 48h,49h (Alt 30h)
Adresse Kommunikation 20h

Z80:

Adresse Kommunikation 20h
Adresse AD-Umsetzer 30h

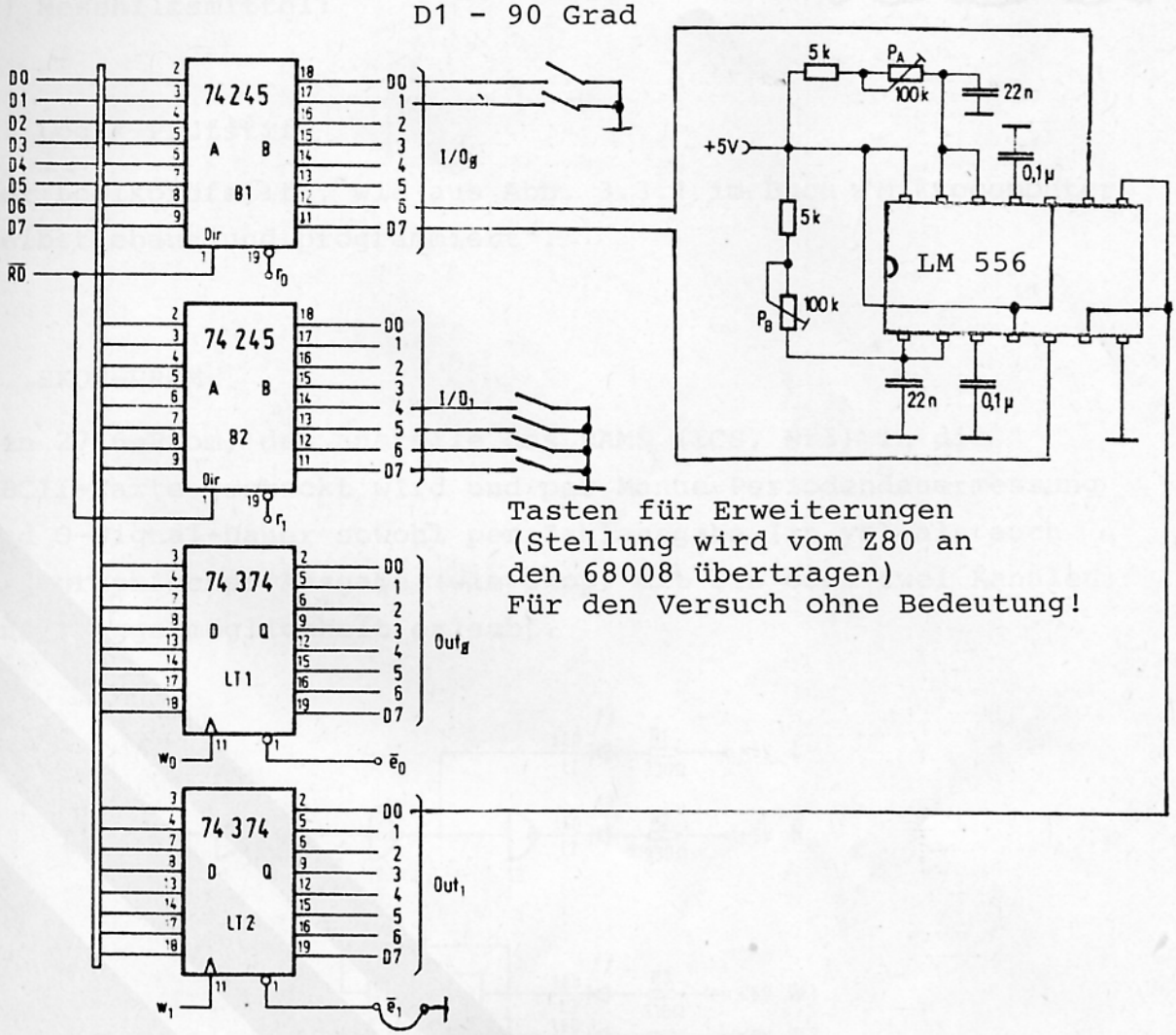


IOE des 68008
auf Adresse 20H

IOE des Z80
auf Adresse 20H

Rechnerkopplung für "Strassenbau"

Kalibriertasten
D0 - 0 Grad
D1 - 90 Grad

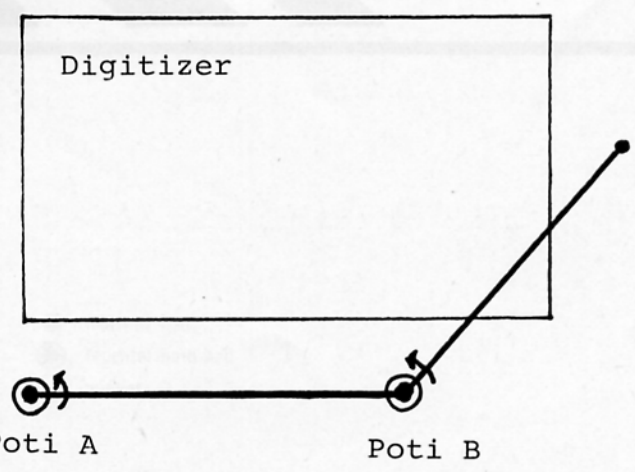


Tasten für Erweiterungen
(Stellung wird vom Z80 an
den 68008 übertragen)
Für den Versuch ohne Bedeutung!

Straßenbau
A/D-Umsetzer

Z80-IOE
auf Adresse 30H

Die Pulsdauer des Digitizers
muß bei Drehung in Pfeilrich-
tung steigen!
(sonst Poti-Abgriff ändern)



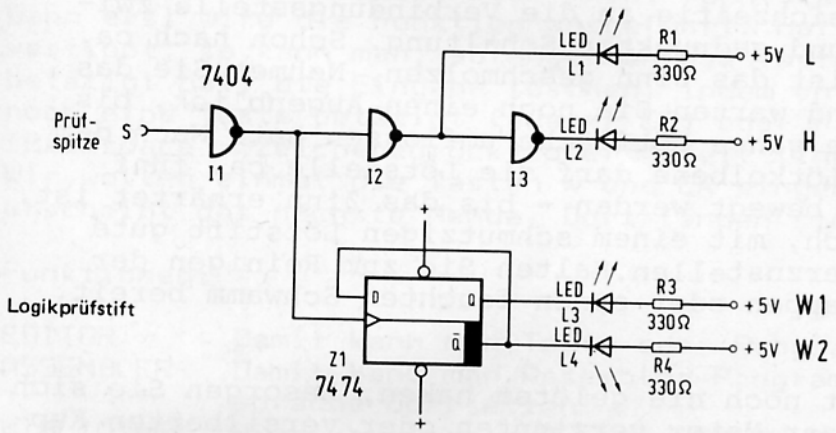
A) Messhilfsmittel:

1. Logik-Prüfstift

Der Logikprüfstift, wie aus Abb. 3.3.1 im Buch "Mikrocomputer selbstgebaut und programmiert".

2. SKOP-PROM

Ein 2716-Prom, das anstelle des RAMS (ICS, Nr3) in die SBCII-Karte gesteckt wird und per Menue Periodendauermessung und O-Signal-Dauer sowohl per Zahlausgabe (in ys) als auch per graphischer Ausgabe (wie Skop) mit ein oder zwei Kanälen und Triggermöglichkeit erlaubt.



	L L1	H L2	W1 L3	W2 L4
S= 0-Signal	✱	○	✱	○
S= 0-Signal	✱	○	○	✱
S= 1-Signal	○	✱	✱	○
S= 1-Signal	○	✱	○	✱
S= ⎍⎍	⊛	⊛	⊛	⊛
S= ⎍⎍	✱	○	⊛	⊛
S= ⎍⎍	○	✱	⊛	⊛

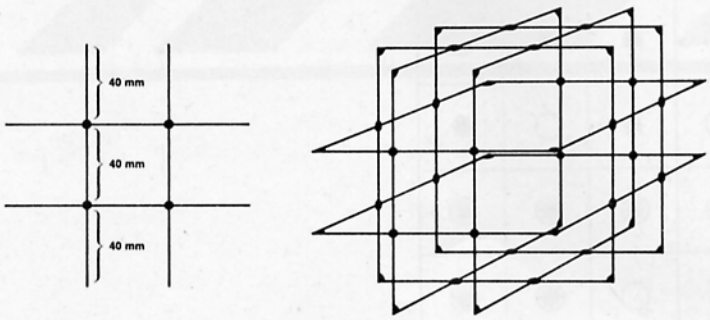
✱ leuchtet hell
⊛ leuchtet halb hell
○ dunkel

Vom Löten

Löten ist eigentlich ganz einfach! Einige Regeln sollen Ihnen helfen:

1. Verwenden Sie niemals säurehaltige Lötpasten oder Lötwasser, sondern Zinnlot mit 60% Zinn und 40% Blei mit Kolophoniumkern. Dieses Lot ist im Fachhandel als Elektroniklot erhältlich.
2. Verwenden Sie einen kleinen elektrischen Lötkolben von 15 - 30 Watt Leistung und feiner Spitze - am besten mit einer Dauerlötspitze. Lötkolben zu großer Leistung bringen zuviel Hitze auf die Leiterbahn, die sich dann ablösen kann. Zu kleine Kolben liefern zuwenig Hitze, sodaß die Gefahr sogenannter kalter Lötstellen größer wird.
3. Richtiges Löten geht schnell. Legen Sie Lötkolbenspitze und Lötzinn gleichzeitig an die Verbindungsstelle zwischen Bauteil und gedruckter Schaltung. Schon nach ca. drei Sekunden ist das Zinn geschmolzen. Nehmen Sie das Lötzinn fort und warten Sie noch einen Augenblick, bis das Lötzinn die ganze Lötfläche umflossen hat. Nach dem Wegnehmen des Lötkolbens darf die Lötstelle ca. fünf Sekunden nicht bewegt werden - bis das Zinn erhärtet ist.
4. Es ist unmöglich, mit einem schmutzigen Lötstift gute Verbindungen herzustellen.Halten Sie zum Reinigen der Spitze einen Lappen oder einen feuchten Schwamm bereit.

Wenn Sie überhaupt noch nie gelötet haben, besorgen Sie sich Lötzinn und ein paar Meter verzinnten oder versilberten Kupferdraht (aus dem Elektronik- oder Bastelladen) und stellen Sie sechs Gitter her, wie sie das linke Bild zeigt. Daraus produzieren Sie einen Würfel wie im Bild rechts. Das ist nicht so schwer, wie es aussieht!
Danach sind Sie fit im Löten und können sich an die Platinen wagen.



68000 (68008) Grundprogramm

1. Einfuehrung in die Bedienung
Das Grundprogramm befindet sich in vier EPROMs auf Adresse 0 bis 7FFF, belegt also 32K Byte Speicher. Als Arbeitsspeicher benötigt es einen RAM-Speicher von Adresse 8000 bis 9FFF. Wird die CPU 68K durch einen RESET gestartet, so meldet sich das Hauptmenue, ähnlich wie beim Z80 Grundprogramm.
Im Hauptmenue gibt es die Befehle:

Aendern:	Zum Aendern von Speicherinhalten.
Starten:	Damit kann ein Programm gestartet werden.
Ansehen:	Ein Speicherbereich kann angesehen werden.
Symbole:	Eigens definierte Symbole, Namen werden alphabetisch ausgeben.
W	steht für WEITER.

Um ein Menue auszuwählen, wird die Zahl oder der Buchstabe eingetippt. Dann erscheint die Zahl oder der Buchstabe in dem Fenster, in dem sich auch der blinkende Cursor befindet. Will man die Funktion ausgeführt bekommen, so muß man nun anschliessend die Taste CR (Carriage Return = Wagenrücklauf) betätigen, sie befindet sich meißt rechts am Tastaturrand. Dann erst wird die Funktion ausgeführt. Hat man sich vertippt, so kann man, solange man die Taste CR nicht betätigt hat, die Eingabe löschen, indem entweder einfach noch eine Taste betätigt (ausser CR) oder die Taste BS (Backspace = Zeichenzurück) oder "Pfeil links" eingibt.
Wir wollen einmal die Tasten W und CR eingeben. EWs erscheint das nächste Menue. Dort finden sich folgende

Funktionen:

EDITOR	Damit kann man Texte oder Programme eingeben
ASSEMBLER	Damit kann man Assembler-Programme in Maschinensprache übersetzen.
BIBLIOTHEK	Hier kann man z.B. den PASCAL-Compiler aufrufen, falls der PASCAL-Promsatz vorhanden ist. Es ist auch möglich eigene Bibliotheken zu erstellen, in der sich Programme befinden, die man oft benötigt.
OPTIONEN	Damit kann man verschiedene Dinge einstellen, z.B. Ausgabe auf Drucker, Zeichen pro Zeile (80 oder 40) Debug (Fehlersuche) etc.

Mit W kommt man ins nächste Menue.
Dort finden sich die Funktionen:

SPEICHERN CAS	Programme, Daten oder Texte können auf einem Kassettenrekorder gespeichert werden.
LADEN CAS	Laden von dem Kassettenrekorder
PRUEFEN CAS	Der Text, oder Daten im Speicher werden mit der Aufnahme auf dem Kassettenrekorder vergleichen.

Mit W geht wieder ins nächste Menue.

Dort kann man:

EPROM PROG	Mit der Baugruppe PROMMER ein EPROM programmieren.
EPROM LESEN	Oder den inhalt eines EPROMs einlesen.
SPEICHERBEREICHE	Der Aufruf dieser Funktion gibt eine Übersicht über vorhandene RAM-Speicherplätze. Nicht alle angezeigten RAM-Plätze dürfen jedoch von eigenen Programen benutzt werden, da sie ggf vom Grunprogramm belegt sind.

Mit W geht weiter.

Hier gibt es die Menues:

IO SETZEN	Damit können Peripherieports direkt angesprochen werden, als Kommuntikation mit der Aussenwelt.
IO LESEN	Eingabe von Peripherieports
EINZELSCHRITT	und Programme schrittweise getestet werden. (Debug vor dem Assemblieren einschalten)

Mit W gehts weiter und man landet wieder im Hauptmenue.

2. Befehle des Grundprogramms

Nach dieser groben Übersicht wollen wir jetzt ins Detail gehen.

Hauptmenue:
AENDERN: Aendern von Speicherinhalten.
Zuerst wird die Adresse angefragt. Hier hat man die Möglichkeit die Adresse Dezimal,Sedezimal, Binaer oder Symbolisch anzugeben.
Dezimal ist die Eingabe wenn einfach eine Zahl angegeben wird, also nicht wie beim Z80-Grundprogramm.
Sedezimal ist die Eingabe, wenn ein Dollar-Zeichen, der Zahl vorangestellt wird.
Binaer ist die Eingabe, wenn ein %-Zeichen vorangestellt ist.
Symbolisch ist die Eingabe, wenn eine Zeichenkette eingegeben wird, die mit einem Buchstaben anfängt, Z.B. QUADRAT. Dieses Symbol muß allerdings definiert sein, entweder durch ein Assemblerprogramm, oder eine Zuweisung.

Zuweisung:
Es ist möglich bei jeder Adresseingabe im Menue zunächst eine Zuweisung durchzuführen, um Symbole zu definieren.
Beispiel:
SPEICHSTART := $9C00
Mit der Taste CR wird die Eingabe abgeschlossen, das Menuefenster bleibt und ist für eine neue Eingabe bereit.
Nun kann man auch im Menueteil rechnen, folgende Operationen sind erlaubt:

\+ - * / ()

* und / werden nur mit 16Bit Genauigkeit durchgeführt, sind daher für Adressrechnung nicht sinnvoll.
Möglich ist also z.B.:
$9000+300 oder SPEICHSTART + $5A
300*(TEST-500) etc.
Symbole müssen definiert sein, sonst gelangt man zurück ins Hauptmenue (bei kritschen Menues wie START) oder ein Wert wird angenommen.
Die Symbole werden auch definiert wenn man Sie fehlerhaft eingibt, sie erscheinen dann mit dem Attribut 5 in der Symboltabelle. Die Symboltabelle kann man nur durch eine Funktion im Optionenmenue löschen. Wichtig, da die Symboltabelle auch Platz braucht und ggf.mit Programmteilen in Konflikt gerät wenn sie zu lang wird. Die Länge erfährt man beim assemblieren eines Programms.
Es gibt auch noch weitere Sonderzeichen die eine Bedeutung haben. Das Zeichen § (Klammeraffe) ist für die Kennzeichnung von reservierten Symbolen (Unterprogrammen) aus dem Grundprogramm zuständig.
Beispiel §SCHREITE , §DREHE ...
Dabei erhält man die Adresse dieses Unterprogramms.
Mit ! (Ausrufezeichen) kann man den Index für den Trap-Befehl von reservierten Symbolen erfahren, Also !SCHREITE !DREHE... Damit wird eine Zahl ermittelt.
Mit MOVE.B #!SCHREITE,D7
TRAP #1 wird Auch das Unterprogramm §SCHREITE aufgerufen.

TRAP-Befehle bleiben unverändert, auch wenn die Versionsnummer der Grundprogramms sich ändert, Die Adressen §SCHREITE jedoch können sich ändern. Bei einer neuen Revision müssen Programme, die § verwenden neu übersetzt werden, Programme mit ! jedoch nicht.
Wird das Symbol ? eingegeben, so erhält man die Adresse des Textbereiches, also die Startadresse, bei der Text durch den Editor eingegeben wird.

Zurück zum AENDERN.
Nachdem die Adresse eingegeben würde erscheinen vier Bytes beginnend bei der angegebenen Speicheradresse.
Wird die Taste CR betätigt, so erscheint der nächste Eintrag. Mit - kann man einen Schritt zurück.
Mit R kann man eine neue Adress eingeben und mit M gelangt man ins Menue zurück. Die eingabe wird immer mit einem CR abgeschlossen. Zum AENDERN gibt man den neuen Wert ein.
Mit .L, .W, .B (müssen groß geschrieben werden) kann man entweder ein Langwort, ein Wort oder ein Byte neu eingeben.
Beispiel:
vorher:
Speicherzellen: 9000: 45 46 47 48
Eingabe: $1F.L
nachher:
Speicherzellen: 9004: 49 4A 4B 4C
Nun viermal - CR eingeben
Speicherzellen: 9000: 00 00 00 1F

Programme oder Datenfelder werden aber sinnvollerweise nicht mit dem AENDERN-Menue eingeben, sondern mit Hilfe des Assemblers. Daher besitzt die Funktion nicht die Bedeutung, wie sie es beim Z80 getan hatte.

STARTEN:
Damit kann ein Programm gestartet werden. Dazu wird die Startadresse eingeben, dir Angabe kann auch hier wieder DEZIMAL, SEDEZIMAL oder Symbolisch erfolgen. Am besten ist eine symbolsche Angabe. Vertippt man sich nämlich bei einer Eingabe mit einer Zahl, so wird dort das Programm gestartet, vertippt man sich bei einem Symbol, merkt es das Grundprogramm und das Programm wird nicht gestartet.
Anwenderprogramme werden mit RTS abgeschlossen, dann gelangt man nach der Ausführung wieder zurück ins Hauptmenue.

ANSEHEN:
Der Inhalt eines Speicherbereichs wird auf dem Bildschirm in sedezimaler Schreibweise ausgegeben. Dazu muß man die Startadresse angeben. Dann kann man mit + oder - blättern, mit R eine neue Adresse eingeben oder mit M ins Menue zurückgelangen.

SYMBOLE:
Die Symboltabelle wird alphabetisch auf dem Bildschirm (oder Drucker) ausgegeben. Der erste Eintrag ist der Name des Symbols. Dann wird der Wert des Symbols ausgegeben und als dritter Eintrag die Bedeutung des Symbols. 5 bedeutet dabei das Symbol ist undefiniert, wurde also nur verwendet, aber noch nicht definiert, 1 bedeutet, das Symbol besitzt einen Byte-Wert, 2 einen Wort-Wert und 3 einen Langwortwert.
Namen die durch den Assembler definiert wurden besitzen immer den Wert 3.

EDITOR:
Siehe Abschnitt 3, dafür gibt es ein eigenen Abschnitt.
Der Editor dient der Programm - und Texteingabe.
Der Editor ist bildschirmorientiert, im Gegensatz zum zeilenorienttierten Bildschirm kann man damit viel komfortabler arbeiten, blättern, mit dem Cursor einfach auf Positionen fahren und verbessern.
Der Editor wurde an das CP/M-Programm WORDSTAR angelehnt, das heißt die Befehle die in unserem Editor vorhanden sind sind ähnlich zu denen im Wordstar. Wer mit unserem Editor gearbeitet hat, kann leicht auf Wordstar umsteigen und umgekehrt.

ASSEMBLER:
Auch hierfür ein eigenes Kapitel (siehe 4.). Der Assembler hat die Aufgabe, durch den Editor eingegebene Programme in Maschinensprache zu übersetzen. damit wird es unnötig sich sedezimale Befehlscodes zu merken, was auch beim 68008 unmöglich ist, da es so viele gibt. Einfacher ist es daher leicht zu merkende Kürzel einzugeben und die Übersetzung dem Computer zu überlassen.

BIBLIOTHEK:
Damit können Zusatzprogramme gestartet werden. Es wird nach Aufruf, der jeweilige Programmname gezeigt und wenn man J eingibt kann man dieses Programm starten. Ist PASCAL/S vorhanden, so wird es dort erscheinen. Mit M gelangt man ins Menue zurück.

OPTIONEN:

Dadurch wird ein Untermenue aufgerufen, über das man weitere Menues aufrufen kann.
Es gibt:

ASSEMBLER OPTIONEN	Mit denen man die Zeichenausgabe umschalten kann
TEXTSTART	Damit kann man die Startadresse für Texteingaben angeben
LOESCHEN SYMBOLE	Damit kann man die Symboltabelle löschen
ZEICHEN/ZEILE & DEBUG	Einstellen der Zeichenzahl pro Zeile und von Debug für Einzelschritt.

ASSEMBLER OPTIONEN:

1 NUR FEHLERAUSGABE	Der Assembler, oder Pascal-Compiler geben das Quellprogramm nicht auf dem Bildschirm erneut aus, sondern nur Fehlermeldungen, falls vorhanden
2 AUSGABE AUF CRT	Die Ausgabe erfolgt auf den Bildschirm. Quellen werden beim Übersetzen mit ausgegeben.
3 AUSGABE AUF LST	Die Ausgabe erfolgt auf den Drucker. Dabei wird ein Centronix-Drucker verwendet (siehe Schaltung)
4 WIE 3 OHNE LF	Für manche Druckertype erforderlich, die bei Ausgabe von CR automatisch LF (Linefeed= Zeilenvorschub) mit ausgeben.

TEXTSTART:

1 ALTER TEXT	Startadresse eines schon vorhandenen Textes angeben
2 NEUER TEXT	Neues Textgebiet angeben, ein alter Text wird falls er dort stand gelöscht.

Alle Assemblierungen und Compileraufrufe(PASCAL) beziehen sich auf diesen Textstart, wie auch der Editor. Im Speicher kann man mehrere Texte unterbringen.
Beim Stromeinschalten wird der Textstart auf die Adresse $9000 gelegt. Dort beginnt also die Texteingabe.
Maschinen-Programme werden ab Adresse $9C00 gespeichert. Wird der Text zu groß kann er damit in Konflikt geraten. Mit einem Befehl an den Assembler (ORG) kann der Start des Maschinenprogramms geändert werden. Der Assembler gibt auch immer Auskunft, wie groß die jeweiligen Gebiete sind. Das Textende erfährt man vom Editor, wenn man den Cursor hinter das letzte Zeichen positioniert und der Bildschirm sonst leer ist.
Der Speicherbereich $8000 bis $8fff ist für das Grundprogramm reserviert. Dort darf man keine Texte hinlegen. Wird PASCAL/S verwendet, so ist der Speicherbereich $18000 bis $1ffff zusätzlich reserviert, dort wird der Compilercode hingelegt, wie auch Variable etc. In diesem Bereich dürfen auch keine Texte liegen.

LOESCHEN ALLER SYMBOLE:
Durch Eingabe von 1 kann man die Symboltabelle löschen. Es empfhielt sich dies von Zeit zu Zeit zu tun, da sonst falls unterschiedliche Programme oder Fehleingaben vorlagen die Symboltabelle sehr groß wird und ggf. das Textgebiet ab $9000 gefährdet. Oder man legt den Text auf höheren Adressen ab.
Das Ende der Symboltabelle wird beim Assemblieren mit ausgeben, so daß man sich dort orientieren kann.

ZEICHEN/ZEILE & DEBUG:

1 40 ZEICHEN	Die Ausgabe wird auf 40 Zeichen gestellt. Editor und Assembler sowie PASCAL und alle CO-Ausgaben sind davon betroffen. Nach RESET ist das Grundprogramm auf 40 Zeichen eingestellt. Wird über die Zeile hinaus getippt, so verschwinden die weiteren Zeichen
2 80 ZEICHEN	Nun werden 80 Zeichen pro Zeile dargestellt. Die Darstellung ist enger und bei Verwendung eines TV-Gerätes als Wiedergabemonitor nicht gut lesbar. Für Textverarbeitung jedoch unumgänglich. Die Ausgabe wird dadurch auch schneller.
3 DEBUG AN	Soll vor dem Assemblieren aufgerufen werden, wenn man anschließend im Einzelschritt Programme testen will. Dann erhält man die Quellzeile als Zusatz beim Einzelschritt angezeigt. Debug belegt aber auch Platz, ca. 8 Bytes pro Quellzeile. Die Information wird

hinter dem Quelltext abgelegt. Der Assembler gibt das DEBUG-Ende aus um vor Speicherkonflikten zu warnen.

4 DEBUG AUS — Debug wird auch ausgeschaltet wenn man den Editor aufruft. Daher ist diese Funktion normalerweise nicht nötig.

Weitere Grundmenues:

SPEICHERN CAS:
Das Abspeichern von Programmen oder Daten ist damit möglich.

1 TEXT — Quelltexte, die durch den Editor eingeben wurden

2 DATEN — Speicherbereiche abspeichern.

Bei 1 TEXT wird die Startadresse angefragt. Mit ? als Eingabe wird der aktuelle Textbereich abgespeichert, ? steht z.B. für die Adresse $9000, die man auch so eingeben kann. Mit ? erreicht man immer den zuletzt bearbeiteten Text (oder durch Optionen eingestellt).
Dann wird ein Name eingegeben um beim Laden eine Kontrolle zu haben.
Der Kassettenrekorder wird vor den Namenseingabe gestartet.
Bei 2 DATEN wird nach einer Start und einer Endadresse gefragt. Die Endadresse ist beim Textspeichern nicht nötig, da sie das Grundprogramm kennt, beim Datenspeichern kann ein beliebiger Bereich gespeichert werden.

LADEN CAS:

1 TEXT — Nach einer Startadresse wird gefragt. Man gibt ? ein oder $9000 z.B. als direkte Adresse. (Dollarzeichen nicht vergessen.

2 DATEN — Anfangs- und Endadresse stehen fest, es werden die gleichen Adressen verwendet, wie beim SPEICHERN.

PRUEFEN CAS:
wie LADEN CAS, jedoch wird der vorhandene Inhalt des Speichers mit dem aufgezeichneten verglichen und Abweichungen gemeldet. Diese Funktion sollte man nach jedem Speichern durchführen um sicherzugehen das auch richtig aufgezeichnet wurde.
Auch beim LADEN können Fehlermeldungen erfolgen, da beim Aufzeichungen auch Prüfsummen mit aufgezeichnet werden um sicherzustellen, das die geladenen Werte auch fehlerfrei sind.

EPROM PROG:
damit kann man zusammen mit der PROMMER - Karte EPROMs programmieren. Dabei wird zunächst gefragt:

VON — Dies ist die Adresse von wo aus programmiert werden soll

BIS — Ist die letzte Adresse im Speicher

NACH — Ist die Adresse des EPROMs von der ab in das EPROM programmiert werden soll. Normalerweise wird hier 0 eingegeben.

Beispiel:
VON $9C00
BIS $9CFF
NACH 0
Dann wird gefragt B=Bereit und wenn man B eingeibt beginnt die Programmierung. Erst beim erscheinen von B darf man das EPROm in die Fassung stecken, dann erst geht auch die rote LED aus, falls sie vorher noch an war.

EPROM LESEN:
VON Ist hier die Adresse im EPROM, als STD=0
BIS Adresse im EPROM
NACH Adresse im 68008 Speicher. Dort wird begonnen die Daten abzulegen.
Beispiel:
VON 0
BIS $FF
NACH $9C00
das EPROM darf bei erscheinen des Menues eingelegt werden.

SPEICHERBEREICHE:
Diese Funktion kann man einfach mal aufrufen. Es erscheinen dann alle RAM-Speicherplätze. Dabei wird nicht berücksichtigt, welche Plätze durch das Grundprogramm oder Compiler etc. verwendet werden. Es ist praktisch nur ein schneller Speichertest um Hardwarefehler zu finden.
ERSTER BEREICH, gibt den ersten zusammenhängenden RAM-Speicherbereich an. An das Ende dieses Bereiches wird der Stack-Pointer gelegt. Daher sollten am Ende dieses Bereichs keine Programme mehr liegen.
EPROMS werden nicht angezeigt.

IO LESEN:
Damit kann man sich Ports ansehen. Es wird zuerst nach einer Adresse gefragt. Alle IO-Ports liegen bei der CPU68K ab Adresse $FFFFFF00 bis $FFFFFFFF. Dies ist wichtig, da der Z80 z.B. einen vom Hauptspeicher unabhängigen Adresseraum besitzt, der 68008 aber nicht. Mit IO LESEN kann man daher auch RAM-Speicherplätze ansehen.

Beispiel:
Symboldefinition als Abkürzung eingeben:
TAST := $FFFFFF68
Dann TAST eingeben (mit CR abschliessen jeweils).
Nun erscheint der Inhalt dieses Ports in binaerer und sedezimaler Schreibweise auf dem Bildschirm. In unserem Beispiel der Wert 8D und 10001101. Nun kann man verschiedene Befehle geben.
Mit R kann man eine neue Adresse eingeben.
Mit D schaltet man auf Dauerfunktion, es wird damit ständig der aktuelle Wert des Portes angeben, wichtig für Tests.
Mit S kann man diese Dauerfunktion wieder anhalten.
Mit M gelangt man wieder zum Teilmenue mit IO LESEN etc.

IO SETZEN:
Es wird hier neben der Adresse auch nach einem Bytewert gefragt (Data:). Dieser Wert wird an die Adresse ausgegeben.
Danach kann man mit R eine neues Ausgabe durchführen oder mit M ins Menue zurück.

EINZELSCHRITT:
Damit ist es möglich ein Programm schrittweise auszuführen. Es wird dazu die Startadresse angegeben.
Vor der Übersetzung durch den Assembler sollte man mit Optionen DEBUG AN einstellen um Zusatzinformationen beim Einzelschritt zu erhalten.
Auf dem unteren Bildrand erscheinen alle Registerinhalte des 68008 und rechts und die QUELLZEILE, die zu diesem Befehl gehört.
Mit CR kann man einen Befehl ausführen. Dabei werden JSR - Aufrufe komplett ausgeführt und BSR-Aufrufe schrittweise. Damit ist es möglich Programme hierarchisch zu testen, also erst die Unterprogramme und dann die Programme, die diese Unterprogramme aufrufen. Will man es nicht, so verwendet man BSR als Unterprogrammaufruf.
Mit N kann man eine Anzahl von Schritten vorgeben, die dann ausgeführt werden.
Mit B kann man eine Adresse vorgeben, bis zu der ausgeführt werden soll.
Mit S kann man die Bildseite anwählen. 0 bis 3 ist erlaubt. Wird keine Zahl angegeben, so ist auch wieder der Flip-Mode möglich, es werden also ggf. mehrere Bildseiten im Wechsel dargestellt.
Mit L kann man den Bildschirm zwischendurch löschen.
Der Einzelschritt rettet nicht nur alle Register des 68008 sondern auch alle Register des Graphik-Prozessors, so daß man auch Programme, die direkt mit diesem arbeiten austesten kann. Doch das nur für Fortgeschrittene.

3. Der Texteditor
Damit ist es möglich Programme und Texte einzugeben. Er wird einfach aufgerufen und es erscheint ein leeres Bildfeld. Links sind DEL-Zeichen zu sehen, sie geben an, das die Zeile leer ist.
Unten ist eine Status-Zeile, die verschiedene Informationen ausgibt, wie Textstartadresse, Textende etc.
Nun kann man einfach Texte eingeben. Mit CR wird eine Zeile angeschlossen und man gelangt damit in die nächste Zeile. So kann man einen ganzen Text eingeben. Nun gibt es aber auch viele Tasten um Fehler zu korrigieren, Zeilen einzufügen und zu löschen, Texte zu suchen und gegen andere zu ersetzen.
Als erstes der wichtigste Befehl, nämlich um wieder aus dem Editor herauszukommen. Man betätigt dazu die Taste CTRL und dann zusammen mit dieser die Tasten SHIFT K, dannach die Taste SHIFT X ohne CTRL. Dann ist man wieder im Hauptmenue.
Mit RESET darf man nicht aus dem Editor heraus, sonst ist alles was auf dem Bildschirm sichtbar war gelöscht.
Befehle:

CTRL-J bedeutet Eingabe der Tasten CTRL (Control) und SHIFT J (Taste J und die Taste Shift für Großschreibung).
Damit erhält man eine Liste aller verfügbaren Befehle.

Cursor Positionierung
geschieht nicht mit den Pfeiltasten, da zu Wordstar kompatibel, sondern durch CTRL-Sequenzen:

CTRL E	Zeile nach oben
CTRL X	Zeile nach unten
CTRL S	Zeichen nach links
CTRL D	Zeichen nach rechts

CTRL A Wort nach links
CTRL F Wort nach rechts

Diese Zeichen kann man sich am Besten merken, wenn man sich die Tastatur ansieht:

```
    E

A   S   D   F

    X
```

Dann bilden die Tasten ein Fadenkreuz, die Richtung zeigt auch jeweils die Richtung ihrer Bedeutung an.

Dann gibt es

CTRL Z Zeile nach oben scrollen (Cursor bleibt, aber Bildausschnitt verschiebt sich).

CTRL W Zeile nach unten scrollen.

CTRL Y Die Zeile in der der Cursor steht wird gelöscht

CTRL N Es wird eine Zeile an der Cursorstelle eingefügt. Der Cursor steht an auf dieser neuen Zeile

CTRL T Alles rechts der Cursorposition löschen (Wordstar löscht nur ein Wort)

CTRL R Eine halbe Bildschirmseite zurückblättern

CTRL C Eine halbe Bildschirmseite vor blättern

CTRL V Einfügemode einstellen. Alle Zeichen werden nun eingefügt, falls rechts davon schon welche stehen. Ferner wird bei CR automatisch eine Zeile eingefügt

CTRL P Umschalten vom amerikanischen Zeichensatz auf deutschen und umgekehrt. In der Statuszeile ist jeweils sichtbar welcher Mode (amer, deut) gerade eingestellt ist. Bei deutschen Zeichen (ä ö ü Ä Ö Ü ß) wird die Scrollgeschwindigkeit vermindert, da die Zeichen Punkt für Punkt geplottet werden müssen. Bei normalem Gebrauch in Texten jedoch kaum merkbar.

CTRL Q C An das Ende des Textspeichers gehen

CTRL Q R An den Anfang des Textspeichers gehen

CTRL Q F Suchen eines Textstring, wird über ein Menue abgefragt.

CTRL Q A Suchen eines Textstrings und ersetzen durch einen neuen. Wird mit J auf das Blinken des Cursor geantwortet, so wird das nächste Auftreten gesucht und zuvor die Zeichenkette ersetzt, wird mit N geantwortet, so wird nicht ersetzt, aber weitergesucht, bei anderen Zeichen wird der Suchmode wieder verlassen, so auch wenn die Zeichenkette nicht gefunden wurde.

CTRL G löscht das Zeichen rechts neben dem Cursor und schiebt die Zeile nach

DEL löscht das Zeichen links neben dem Cursor und schiebt die Zeile nach

CTRL L Wiederholt das letzte Suche oder Ersetze -Kommando, falls eines gegeben wurde

CTRL K X Beendet die Arbeit im Texteditor und

kehrt in das Menue des Grundprogramms zurück.

4. Assemblerbefehle

Der 68008 (68000) besitzt eine Vielzahl von Assemblerbefehlen, diese kann man dem "16 - Bit Microprocessor USER'S MANUAL MC68000UM/AD3" von Motorola entnehmen. Wir wollen hier mal nur eine kurze Zusammenfassung von Besonderheiten aufführen.

A) Pseudobefehle

Hier sind alle Befehle gemeint, die der Steuerung des Assemblers dienen und nicht Maschinenbefehle sind.

ORG adresse

Damit wird festgelegt, von wo ab der Maschinencode gespeichert wird. Wird die Anweisung in einem Programm nicht gegeben, so wird die Adresse $9C00 angenommen.

Beispiel:

```
ORG $9E00
ORG $18000
```

Als Adresse kann jeder Ausdruck verwendet werden, also Dezimale Angabe, Sedezimale oder Symbolische und Adressrechung (+ - sinnvoll).

symbol EQU wert

Hiermit wird das Symbol "symbol" definiert und ihm der Wert "wert" zugeordnet. Ähnlich zu der Anweisung :=, wie sie im Grundprogramm verwendet wurde.

Beispiel:

```
IO EQU $FFFFFF30
RAMENDE EQU $9FFF
```

```
DC.B  'Text'
DC.B  daten,daten,daten ...
DC.W  daten,daten,daten ...
DC.L  daten,daten,daten ...
```

Damit kann der Speicher mit einzelnen Datenwerten belegt werden. Ebenfalls können Texteingaben erfolgen.
Ein .B gibt an das Bytes belegt werden, .W das Worte gemeint sind und .L daß Langworte verwendet werden.

Beispiel:

```
DC.B 'Hallo',0
DC.B $55,$AA,$66
DC.W $1234
DC.L START,$12345678
```

Wird DC.B verwendet, so muß man darauf achten, das nachfolgende Befehle auf einer geraden Adresse zu liegen kommen, da der 68008 Wortorientiert arbeitet und Befehle immer auf geraden Adressen beginnen müssen
Ein nachfolgender Befehl DS 0 korrigiert automatisch auf Wortgrenze.

```
DS.B n
DS.W n
DS.L n
```

Reservieren von Speicherbereichen, N ist die Anzahl der Bytes, Worte oder Langworte.

Beispiel:

```
DS.B 100   reserviert 100 Bytes
DS.L 25    reserviert auch 100 Bytes
```

OFFSET adresse
Damit wird die Maschinencodeadresse verschoben.
Die Adresse bei Offset wird auf die aktuelle Adresse des Maschinencodes addiert und dorthin wird der Code gespeichert. Wichtig, wenn man Programme für Adressen schreiben will, auf denen sich kein RAM befindet um sie z.B. auf EPROM zu speichern.
Beispiel:
ORG $0
OFFSET $9C00
Der Maschinencode wird für Adresse 0 übersetzt und dennoch beginnend bei Adresse $9C00 abgelegt. Man kann ihn dann in einem EPROM ablegen und das EPROM auf Adresse 0 z.B. legen.

END
Dort ist das Assemblerprogramm zuende.
Man kann END auch weglassen, dann wird das Textende als Programmende genommen.

CTRL S stoppt die Assemblerausgabe in einem Lsiting
CTRL Q setzt sie wieder fort. Um in das Grundprogramm zu gelangen und den Assembler abzubrechen, darf man RESET betätigen, ohne das Informationen verloren gehen.

symbol: Angabe einer Symboldefinition
Dem Symbol wird der aktuelle Programmzähler zugewiesen.
Beispiel:
START: MOVE #4-1,D3
QUADRAT:
Symbole werden bis zu 8 Zeichen erkannt, längere Namen werden zwar angenommen jedoch nach 8 Stellen nicht mehr voneinander unterschieden.

*
"*" als Wert ist die aktuelle Programmzähleradresse. Als Programmzähler ist hier die Adresse des Maschinencodes gemeint.
Beispiel: MOVE.L *,D0 lädt die eigene Adresse nach D0. * ist immer die Startadresse einer Assemblerzeile.

Kommentare
Werden durch ; oder * oder Leerzeichen vom Programm getrennt.
Beispiel:
ORG $9C00 * Dies ist die Startadresse

B) Adressierarten und Assemblerbefehle
Zuerst eine alphabetische Liste aller Kürzel für den 68008-Assembler:
ABCD, ADD, ADDA, ADDI, ADDQ, ADDX, AND, ANDI, ASL
ASR, BCHG, BCLR, BRA, BSET, BSR, BTST, CHK, CLR
CMP, CMPA, CMPI, CMPM, DBRA, DC, DIVS, DIVU, DS
END, EOR, EORI, EXG, EXT, ILLEGAL, JMP, JSR, LEA
LINK, LSL, LSR, MOVE, MOVEA, MOVEM, MOVEP, MOVEQ
MULS, MULU, NBCD, NEG, NEGX, NOP, NOT, OFFSET
OR, ORG, ORI, PEA, RESET, ROL, ROR, ROXL, ROXR
RTE, RTR, RTS, SBCD, STOP, SUB, SUBA, SUBI, SUBQ
SUBX, SWAP, TAS, TRAP, TRAPV, TST, UNLK
ferner Bcc, Scc, DBcc für die bedingten Befehle und EQU als besonderer Pseudobefehl.

Nicht erschrecken bei dieser Vielfalt, denn man muß nicht alle Befehle auf einmal wissen um mit dem 68008 Programme zu schreiben, Hier der Hinweis auf die Sammlungen 68008 Grundprogramme und PASCAL/S in der sich zahlreiche Assemblerbeispiele befinden, die man direkt einmal eintippen kann und testen. Insbesondere die 68008 Aufbauprogramme, die ja direkt als Beispiel geschrieben sind.

Nun zur Zusammenfassung der Adressierarten:

D0,D1,D2,D3,D4,D5,D6,D7	Angabe der Datenregister
Beispiel: MOVE D0,D1	
A0,A1,A2,A3,A4,A5,A6,A7	Angabe der Adressregister
Beispiel: MOVEA A0,A2	

Dn steht für Datenregister D0 bis D7 und An von nun an für Adresseregister A0 bis A7.

(An)	Indirekte Adressierung Beispiel: MOVE.B (A0),D0
(An)+	Indirekte Adressierung mit nachfolgender Erhöhung des Adressregisters Beispiel: MOVE.B (A0)+,(A1)+
-(An)	Indirekte Adressierung mit vorausgehender Erniedrigung des Adressregisters Beispiel: MOVE.B -(A0),-(A1)
wert(An)	Indirekte Adressierung mit Displacement Beispiel: MOVE.B 3(A0),5(A1)
wert(An,Dn.L) wert(An,An.L) wert(An,Dn.W) wert(An,An.W)	Indirekte Adressierung mit Index. Beispiel: MOVE.W 0(A1,D0.W),5(A2,A3.L)
wert.W	Wertangabe als Wort Beispiel: MOVE.B $1234.W,D0
wert.L	Wertangabe als Langwort Beispiel: MOVE.B $1234.L,D0 Interpretation als Speicheradresse
#wert	Laden eines direkten Wertes. Beispiel: MOVE.B #$55,D1 oder MOVE.L #$12345678,D0
wert(PC)	PC-Relative Adressierung Hier wird automatisch der Abstand verwendet: Beispiel: LEA TEST(PC),A0 mit TEST: ...
wert(PC,Dn.L) wert(PC,An.L) wert(PC,Dn.W) wert(PC,An.W)	Wie bei indirekter indizierter Adressierung, jedoch relativ zum Programmzähler.

Anstelle von Wert oder Adresse dürfen auch Symbole etc. stehen.

Bedingungen:
Für die bedingten Befehle wie Bcc DBcc und Scc
cc steht für die nachfolgenden Bedingungscodes:

T	True, immer erfüllt
F	False, nie erfüllt
HI	High, -C * -Z
LS	Low or Same, C + Z
CC (HS)	Carry Clear, -C
CS (LO)	Carry Set, C

```
NE          Not Equal, -Z
EQ          Equal, Z
VC          Overflow Clear, -V
VS          Overflow Set, V
PL          Plus, -N
MI          Minus, N
GE          Greater or Equal, N*V + -N*-V
LT          Less Than, N*-V + -N*V
GT          Greater Than, N*V*-Z + -N*-V*-Z
LE          Less or Equal, Z+n*-V+-N*V

Beispiel:
      BCS  START
      ST D0
      DBEQ D4,LOOP

Attribute:

Beispiele:
            MOVE.B #'A',D0
            MOVE.W #$1234,D1
            MOVEA.L #START,A0
            BRA.S LOOP              * Kurzer Sprung
            BRA LOOP                * Weiter Sprung (+-32767)
            BRA.W LOOP              * Auch weiter Sprung
            MOVE #1,D2              * Wort ist voreingestellt.
```

C) Anmerkungen
Nicht vorhanden sind Makrofähigkeiten und Strukturierungsbefehle wie REPEAT UNTIL etc., wie sie im PASCAL/S-Listing verwendet wurden.
Sie werden bei Bedarf durch ein Zusatzeprom verfügbar sein.
Die Quellistings wurden auf einem Cross-Assembler der auf einem Z80 lief erstellt, denn damals gab es den NDR-Klein-Computer leider noch nicht. Die meißten Teile bis auf die Strukturierungsbefehle sind jedoch auf dem NDR-Klein-Computer-Assembler vorhanden, so daß man die Listing sicher als gutes Lernbeispiel verwenden kann.

5. Unterprogramme aus dem Grundprogramm
Über 60 Unterprogramme aus dem Grundprogramm sind für den Anwender verfügbar geworden.
Erst mal die vollständige Liste aller Befehle:

```
SCHREITE ,DREHE,SENKE, SET MOVETO,DRAWTO,WRITE,READ,
CI,CSTS,RI,PO,CLR,CLPG,WAIT,SCHR16TE,CLRSCREEN,CO,
LO,SIN,COS,SIZE,CMD,NEWPAGE,SYNC,WERT,ZUWEIS,
CIINIT2,CI2,CO2,SETFLIP,DELAY,FIRSTTIM,
SETPEN,ERAPEN,GRAPOFF,CMDPRINT,PRINT2X,PRINT4X,
PRINT8X,PRINT8B,PRINT4D,HIDE,SHOW,CRT,LST,USR,NIL,
SETERR,GETERR,SETPASS,EDIT,FIGUR,SETFIG,
GETRAM,AUTOFLIP,CURSEIN,CURSAUS,CHAR,
PROGZGE,ASSEMBLE,GETSTX,PUTSTX,GETORG,
PUTORG
```

```
SCHREITE        D0=Anzahl der Schritte
DREHE           D0=Winkel in Grad,Vorzeichenbehaftet
HEBE            Schreibstift heben
SENKE           Schreibstift senken
SET             D1=X, D2=Y, D3=Winkel absolut in Grad
MOVETO          D1=X, D2=Y
```

DRAWTO	D1=X, D2=Y
WRITE	D0=Textgröße, Schriftgröße wie GDP Register 3 ($11 ist kleinste) D1=X, D2=Y und A0=Adresse Text, der mit 0 beendet wird.
READ	D0=Groesse,D1=X, D2=Y, D3=Anzahl maximal einzugebender Zeichen, D4 ist Ergebnis wirklich eingegebene Zahl, D5 ist Ergebnis letztes getipptes Zeichen, A0 ist Zieladresse
CI	Ein Zeichen von Tastatur nach D0 holen
CSTS	0 in D0 wenn kein Zeichen da
RI	Zeichen von Kassette nach D0 holen
PO	Zeichen auf Kassette von D0 ausgeben
CLR	gesamten Bildschirm löschen
CLPG	Die aktuelle Schreibseite löschen
WAIT	Warten bis GDP bereit für Befehle
SCHR16TE	In D0 die Anzahl der 16tel Schritte
CLRSCREEN	Bildschirmlöschen mit Vorbereitung für Cursorausgabe
CO	Zeichen auf den Bildschirm bringen, D0=Zeichen (incl. Scroll etc.)
LO	Zeichen auf dem Drucker ausgeben
SIN	Berechnet D0 := 256*sin(D0)
COS	Berechnet D0 := 256*cos(D0)
SIZE	Zeichen pro Zeile setzen, $11 ist 80 Zeichen/Zeile und $21 ist 40/Zeile. Wert ins D0 Register
CMD	Befehl von D0 an GDP Port 0 ausgeben.
NEWPAGE	D0=Schreibseite (0..3) D1=Leseseite(0..3)
SYNC	Wenn =0 dann Synchronpuls (20ms) nicht aufgetreten, für Flimmerfreie Graphik verwendbar

Spezialbefehle für besondere Anwendungen, nur für Spezialisten !!!

WERT	A0 zeigt auf Symbolischen Ausdruck, in D0 erhält man den Wert, siehe Heft "68008 Grundprogramme"
ZUWEIS	Arithmetische Zuweisung A0 zeigt auf Textstart, "68008 Grundprogramme"
CIINIT2	Textpointer auf STXTXT setzen
CI2	Lesen von RAM-Quelle
CO2	Umschaltbares Schreibziel, z.B. auf Drucker etc.
SETFLIP	D0=Flip, D1=FLIP1, Automatischer Bildseitenwechsel durch AUTOFLIP
DELAY	D0 ist 0.1 Sekunden Verzögerung
FIRSTTIM	Initialisierung für Schildkrötengraphik
SETPEN	GDP auf Schreibmode
ERAPEN	GDP auf Löschmode
GRAPOFF	Schildkröte nicht mehr sichtbar
CMDPRINT	Datenwerte von A0 ab an GDP ausgeben bis der Wert 0 auftaucht
PRINT2X	Wert in D0 als Sedezimalen Text von A0 an ablegen, mit 2 Stellen
PRINT4X	dto. 4 Stellen

PRINT6X	dto. 6 Stellen
PRINT8X	dto. 8 Stellen
PRINT8B	8 Stellen Binärausgabe
PRINT4D	4 Stellen Dezimal ausgeben (+-32767)
HIDE	Schildkröte aus
SHOW	Schildkröte an
CRT	CO2 umschalten auf Bildschirmausgabe
LST	CO2 umschalten auf Druckerausgabe
USR	CO2 umschalten auf Benutzervektor $8024
NIL	CO2 auf Papierkorbausgabe umschalten
SETERR	Fehlercode setzen (siehe PASCAL5)
GETERR	Fehlercode holen (siehe PASCAL5)
SETPASS	Nummer des Übersetzungsdurchgangs setzen
EDIT	Editor als Unterprogramm aufrufen
FIGUR	A0 ist die Adresse der Figur, D0 = Vergrösserungsfaktor, D1 = X, D2= Y 0,1,2,3,4,5,6,7 sind Vektorrichtungen 8 = Hebe, 9=Senke, 10=Ende Alte Figur wird automatisch gelöscht Wenn D0=0 dann wird die alte Figur nur gelöscht.
SETFIG	Figur dauerhaft auf den Bildschirm setzen
GETRAM	Speicherbereiche holen (siehe 68008 Grundprogramme)
AUTOFLIP	Bildwechselautomatik
CURSEIN	Cursor darstellen
CURSAUS	Cursor löschen
CHAR	Zeichenausgabe spez
PROGZGE	A0 Adresse Zeichengenerator, wird ausgeben
ASSEMBLE	Assembler als Unterprogramm aufrufen
GETSTX	Textstartadresse holen nach D0
PUTSTX	Neuen Textstart festlegen, dort muss schon ein Text stehen, denn das Ende wird automatisch gesucht, Wert in D0
GETORG	Default ORG - Adresse nach D0
SETORG	D0 nach Default - ORG - Adresse

6. Fehlermeldungen

Beim Start von Anwenderprogrammen können verschiedene Fehler auftreten, z.B.
INTERRUPT, BUSERROR, ADRERROR, ILLEGAL INSTR,
DIV BY ZERO, CHECK, TRAPV, PRIVILEGE, TRACE, LINEA,
LINEF.
Dies kann teilweise durch Hardwarevorgänge oder Softwarefehler entstehen. Sollte z.B. insbesondere ADRERROR oder ILLEGAL INSTR im Normalbertrieb des Grundprogramms plötzlich auftreten, also nicht bei Adressangabe oder Programmstart, so kann ggf. der EPROM-Speicher zu langsam sein, dann mit einem größerem WAIT-Zyklus (siehe Schaltplan CPU68008) versuchen.
EPROMs 250ns laufen ohne Waits
EPROMs 350ns laufen mit zwei Waits (Jumper auf zweite Position)

Die Bibliotheksfunktion im 68008-Grundprogramm

Wenn Sie im PROM ein Programm mit einem Bibliotheks-Vorspann versehen, kann dieses Programm nachher über das Bibliotheksmenue aufgerufen werden. Das Programm muß auf einer 2kByte-Grenze beginnen. Der Kopf eines Bibliotheksprogramms hat folgendes Aussehen:

```
DC.B    $55,$AA,$01,$00 80
DC.B    "NAME____"              genau 8 Buchstaben
DC.L    Startadresse
DC.L    Programmlänge           (siehe unten)
DC.B    0  oder 1               Reloc
DC.B    0,0,0                   (Reserve für Erweit.)
DC.L    0,0                     hier weitere Entries
                                oder 0,0
```

Die Startadresse hängt davon ab, ob das Programm verschieblich (relokativ) ist (Byte Reloc = 1), oder absolut (Byte Reloc = 0). Bei relokativen Programmen bezieht sich die Startadresse auf den Abstand zum ersten Byte des Kopfes ($55). Setzt man vor den Kopf eine Marke (z.B.:HEAD) und vor den ersten Befehl des Programms auch eine (z.B.:BEGIN), dann läßt sich die Startadresse definieren als:

```
DC.L    BEGIN-HEAD
```

Bei absoluten Programmen ist die Startadresse gleich der absoluten Speicheradresse des Programmanfangs. PROMs mit relokativen Programmen lassen sich also in jede beliebige Fassung stecken.
Hat man mehrere kleine Programme in einem PROM (oder im RAM) kann man die Köpfe der weiteren Eintrittspunkte (entries) hintereinandersetzen. Die letzte Langwortdefinition ist dann nicht 0,0, sondern wieder $55, ...- bis alle entries definiert sind.

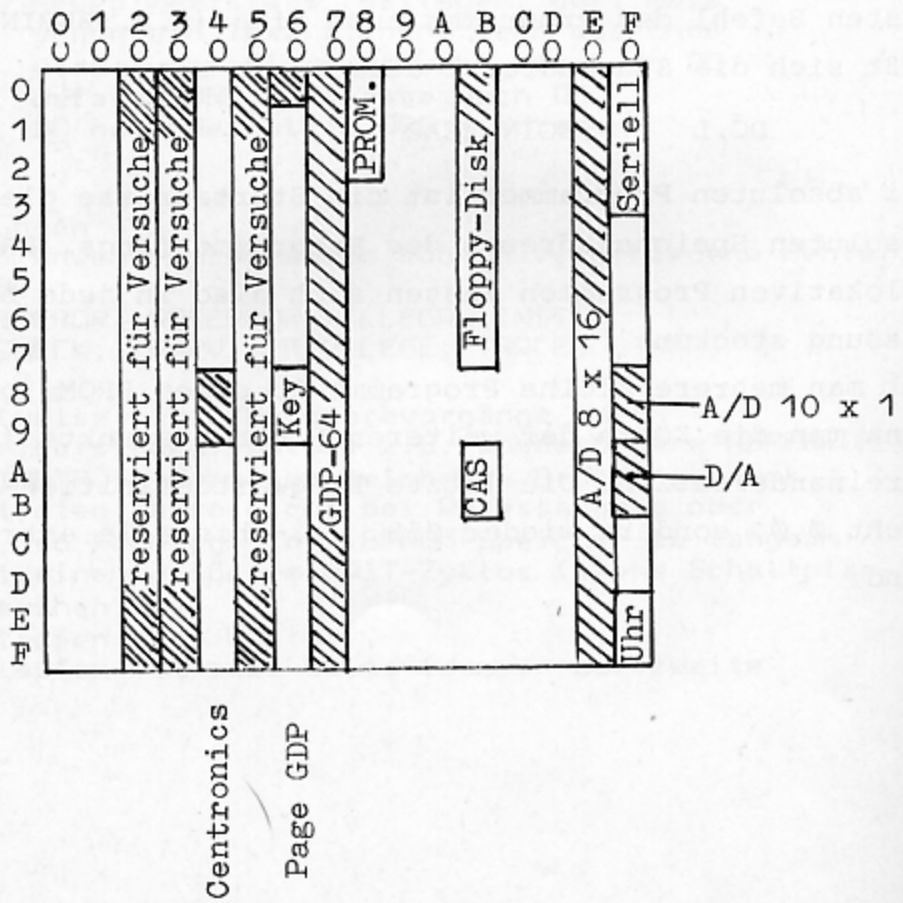
Die Centronics-Schnittstelle

IOE-Karte: ~~I/O~~ OUT Ø : Datenbits Ø - 7
I/O 1 : Bit Ø = Busy
OUT 1 : Bit Ø = $\overline{\text{Strobe}}$

Belegung des Centronics-Steckers:

Pin 1	$\overline{\text{Strobe}}$	Pin 19 - 29 Masse
Pin 2	Data Ø	
Pin 3	Data 1	
Pin 4	Data 2	
Pin 5	Data 3	
Pin 6	Data 4	
Pin 7	Data 5	
Pin 8	Data 6	
Pin 9	Data 7	
Pin11	Busy	

I/O Adreßbelegung



Jumper

KEY: DIL-Schalter bei kleinem Monitor wie Buch Seite 125, bei dem Monitor für die GDP64-Karte ohne Belang. JMP2 ist schon auf der Platine verdrahtet.

IOE: - siehe Adresstabelle

CAS: -

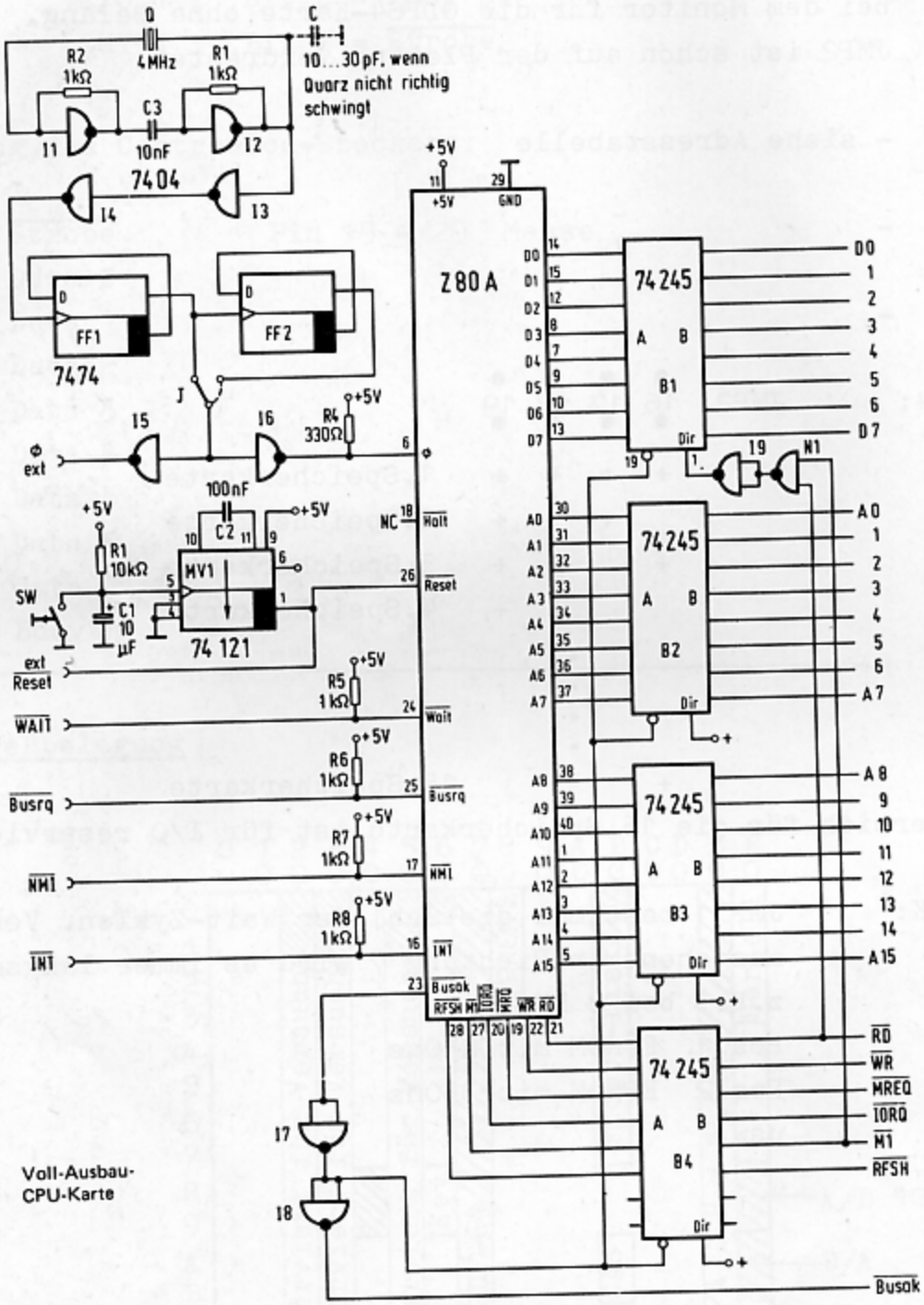
GDP: -

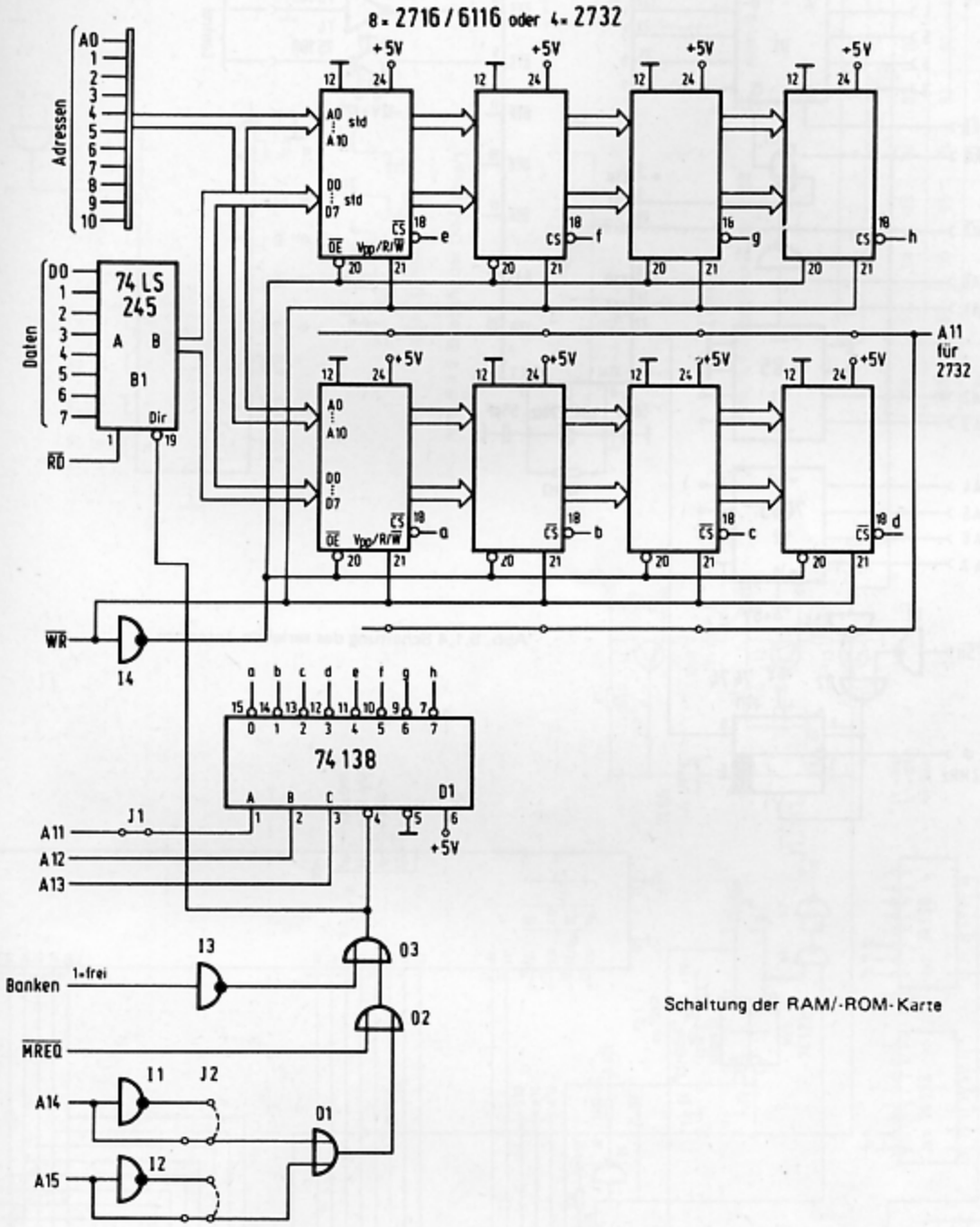
ROA64: JMP2

16	17	18	19	
+	+	+	+	1.Speicherkarte
	+	+	+	2.Speicherkarte
+		+	+	3.Speicherkarte
		+	+	4.Speicherkarte
		:		
+				15.Speicherkarte

Der Bereich für die 16.Speicherkarte ist für I/O reserviert.

CPU68K: JMP 1 bestimmt die Zahl der Wait-Zyklen. Von 1 ausgehend in Richtung 7 wird es immer langsamer.
z.B.: bei 8 MHz
Pos.1 EPROM mit 250ns
Pos.2 EPROM mit 350ns
usw.


Q
C
R2 1kΩ
4 MHz
R1 1kΩ
10...30 pF, wenn Quarz nicht richtig schwingt
C3 10nF
I1
I2
7404
I4
I3
+5V
11
29
+5V
GND
Z80A
FF1
FF2
7474
D
J
I5
I6
R4 330Ω
Φ ext
100nF
C2
MV1
74 121
R1 10kΩ
C1 10 µF
SW
ext Reset
NC
Halt
Reset
R5 1 kΩ
WAIT
Wait
R6 1kΩ
Busrq
R7 1kΩ
NMI
R8 1kΩ
INT
Busak
RFSH M1 IORQ MREQ WR RD
74245
A B
B1
B2
B3
B4
Dir
I9
M1
I7
I8
D0 1 2 3 4 5 6 D7
A0 1 2 3 4 5 6 A7
A8 9 10 11 12 13 14 A15
RD
WR
MREQ
IORQ
M1
RFSH
Busak
Voll-Ausbau-CPU-Karte



Schaltung der RAM/-ROM-Karte

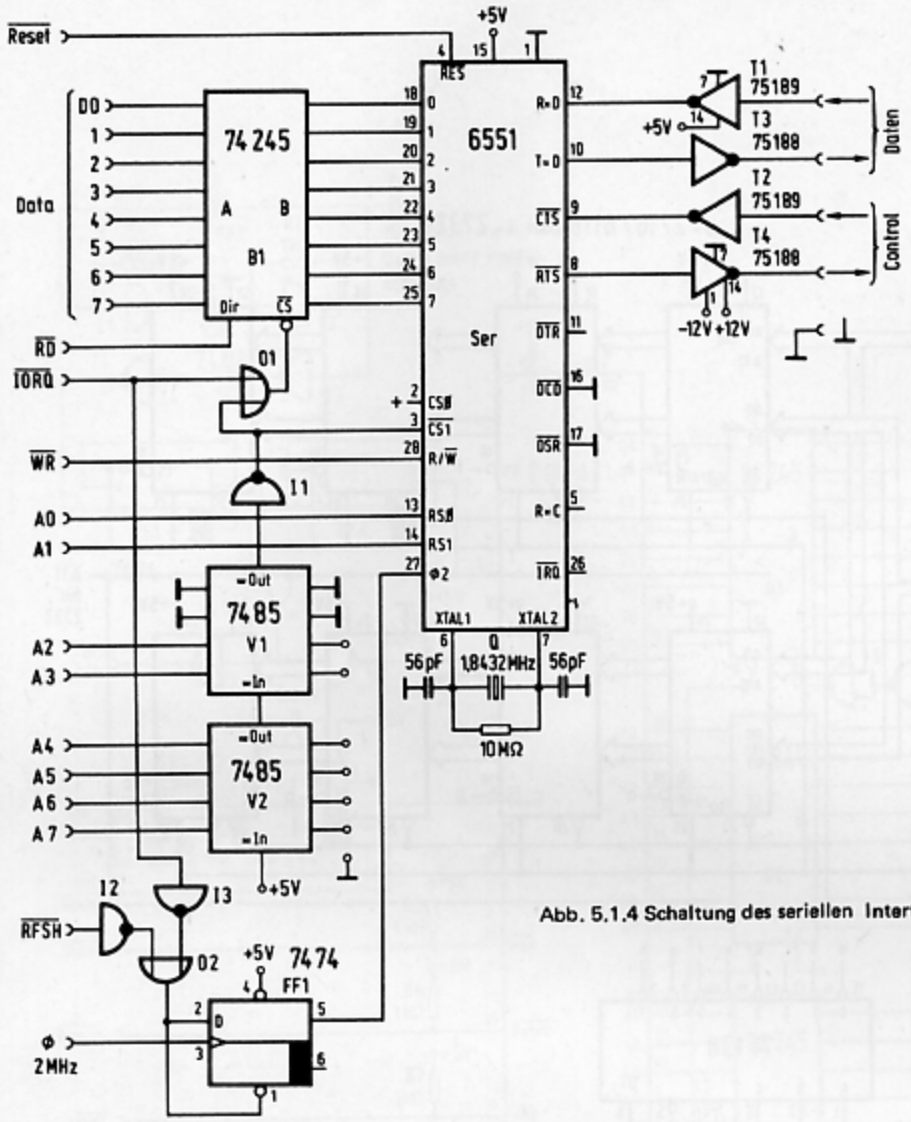
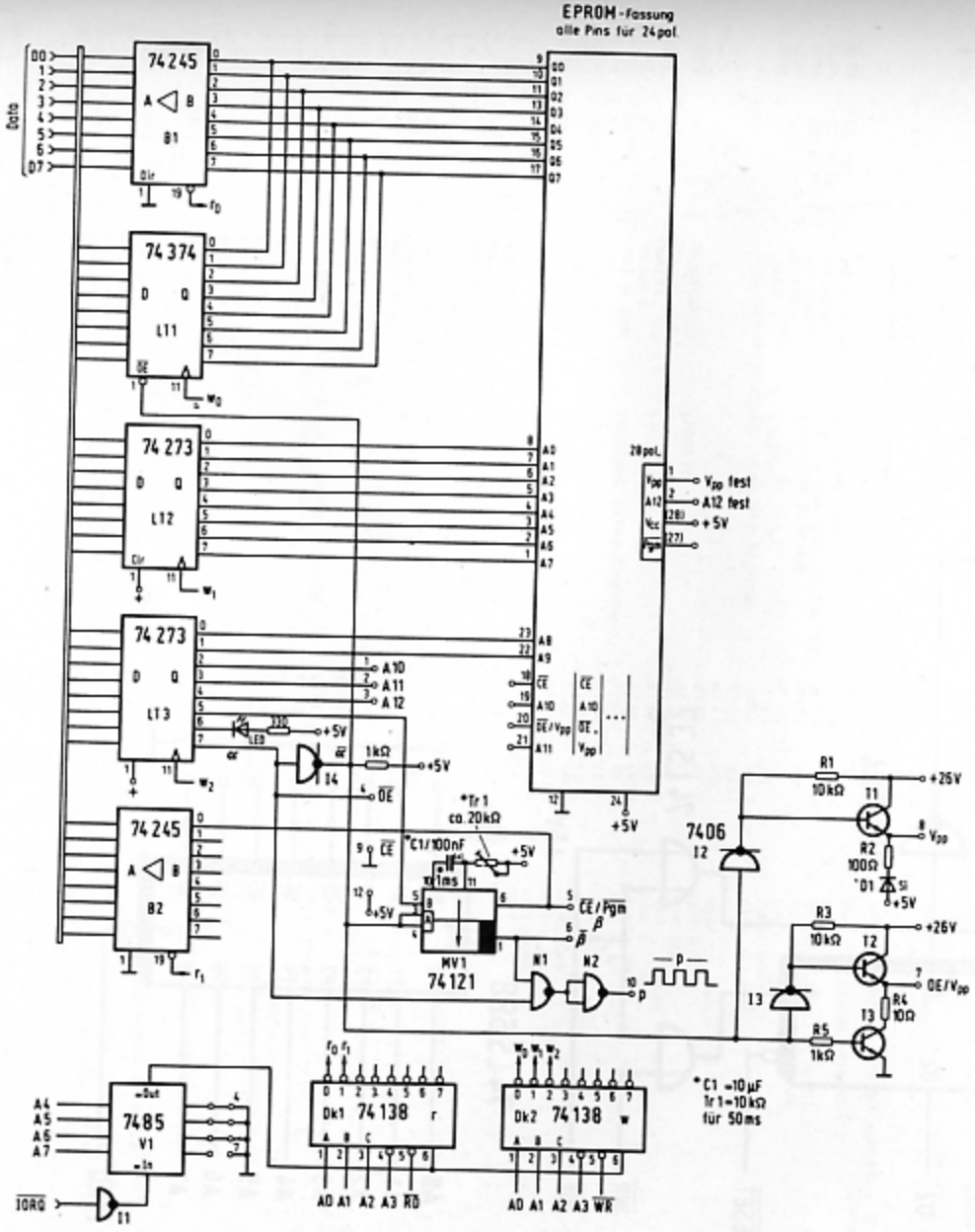


Abb. 5.1.4 Schaltung des seriellen Interface



Schaltung der EPROM-Programmier-Karte

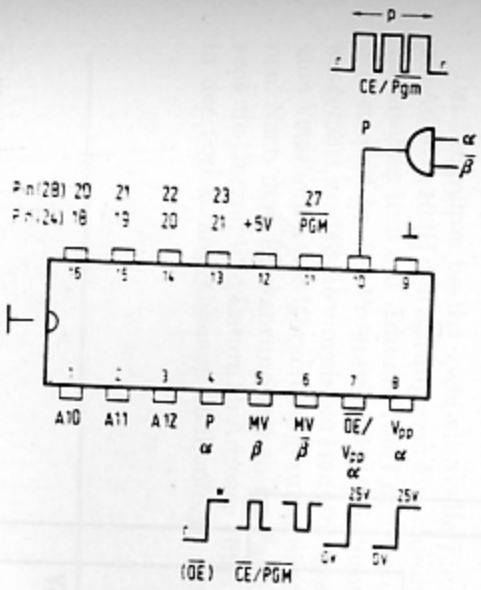


Abb. 5.4.2 Belegung des Dil-Steckers

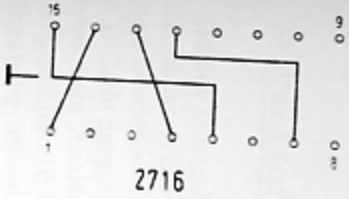


Abb. 5.4.3 Dil-Stecker für 2716

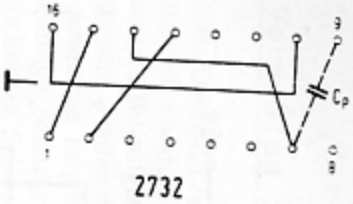


Abb. 5.4.4 Dil-Stecker für 2732

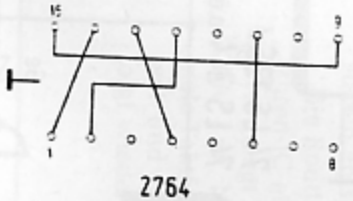


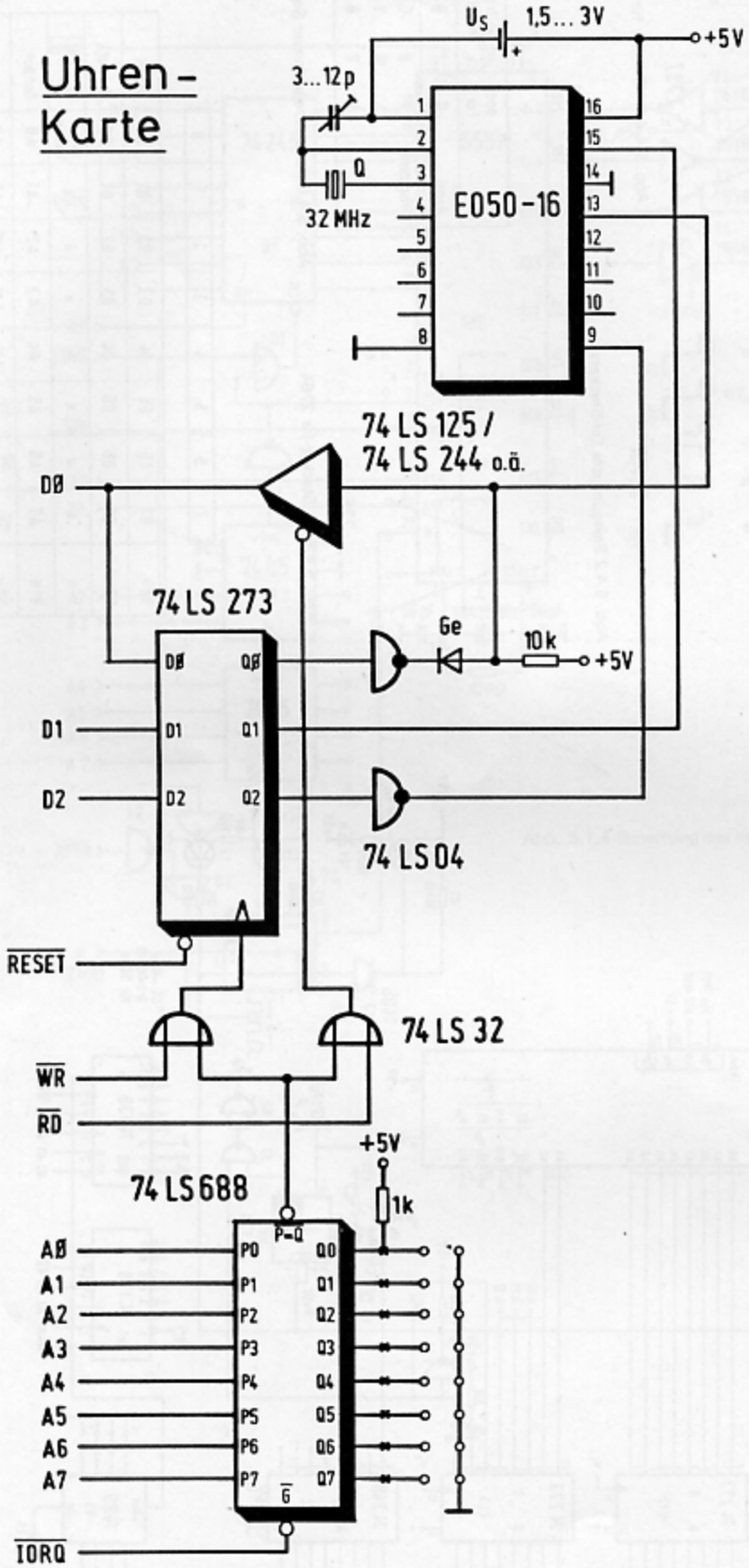
Abb. 5.4.5 Dil-Stecker für 2764

	ena (α)	led	trg (β)
Read	0	1	0
Programm	1	0	0
	1	0	1
	1	0	0

Abb. 5.4.6 Read//Programmier Sequenzen

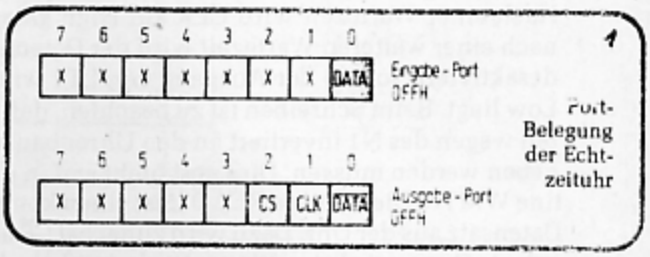
	7	6	5	4	3	2	1	0	
80 H	D7	D6	D5	D4	D3	D2	D1	D0	Input
80 H	D7	D6	D5	D4	D3	D2	D1	D0	Output
81 H	x	x	x	x	x	x	x	Busy 1=warten	Input
91 H	A7	A6	A5	A4	A3	A2	A1	A0	Output
92 H	ena ($\overline{CE}$)	led 1=on	trg ⊓_	A12	A11	A10	A9	A8	—

Uhren-Karte


Us
1,5... 3V
+5V
3...12p
Q
32 MHz
E050-16
74 LS 125 /
74 LS 244 o.ä.
D0
74 LS 273
Ge
10k
+5V
D1
D2
74 LS 04
RESET
74 LS 32
WR
RD
+5V
74 LS 688
1k
P=Q
A0
A1
A2
A3
A4
A5
A6
A7
IORQ

Vorläufiger Vorschlag zum Betrieb der Uhrenplatine

Da das Uhren-IC nur einen Datenein-/ausgang besitzt, müssen die Daten bitseriell verarbeitet werden. Dazu werden sie in Gruppen zu 8 Bit aufgeteilt und im BCD-Code dargestellt. Insgesamt werden dann sieben Bytes zum Auslesen des gesamten Inhalts benötigt. Das geschieht in der Reihenfolge: Stunden, Minuten, Monatstag, Monat, Jahr, Wochentag, Sekunden..



Das Programm besitzt zwei sinnvolle Einsprünge: auf Adresse 103H den Sprung zum Einschreiben eines Datensatzes und auf Adresse 106H den Sprung zum Stellen der Uhr. Die Programme sind als Subroutinen ausgeführt, so daß sie von einem Hauptprogramm aus zum Verarbeiten der Uhrzeit aufgerufen werden können. Nach dem Rücksprung stehen im Falle des Lesens die Uhrzeit und Datum ab Adresse 109H (7 Byte) in der vorher angegebenen Reihenfolge zur Verfügung. Beim Stellen der Uhr müssen diese Speicherstellen entsprechend vorbelegt sein.

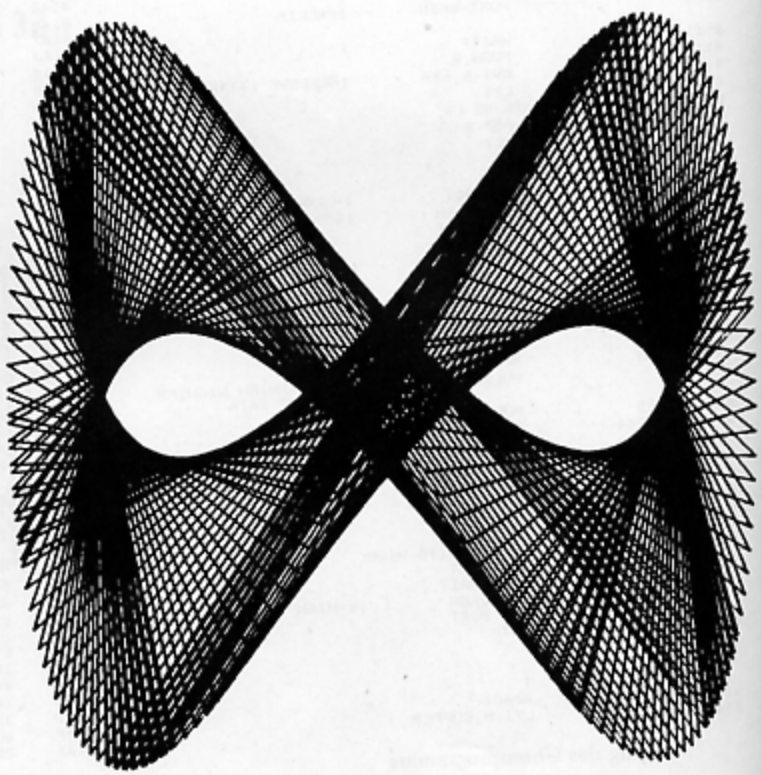
Die Routine PULSRD hat die Aufgabe, ein Bit vom Uhren-IC einzulesen. Dazu wird als erstes CLK auf Low gelegt und $\overline{CS}$ ebenfalls. Die Bitzuordnung gibt *Bild 1* an. Nach Ablauf einer Wartezeit, da die maximale Taktfrequenz begrenzt ist, wird CLK auf High gelegt, und das Datenbit kann eingelesen werden. Dann folgt wieder eine Wartezeit.

```
TDL Z80 CP/M DTSK ASSEMBLER VERSION 2.21
.MAIN. -
                        .PHEX
                        .PABS
0100                    .LOC 100H
                        ;*************************
                        ;* UHRENPRG  TEST1       *
                        ;* 880621    RDK         *
                        ;*************************
                        ;
                        ;
0100    C3 F01E         JMP 0F01EH      ;IN MONITOR
0103    C3 0146         JMP READ        ;LESEN VON UHR IN BUFFER
0106    C3 0175         JMP WRITE       ;STELLEN DER UHR
0109                    BUFFER:
0109                    .BLKB 8         ;ZEIT
                        ;HRS MIN DATE MONTH YEAR DAY SECONDS



00F0                    PORT=0F0H       ;PRELIM

0111                    WAIT:           ;
0111    C5              PUSH B
0112    0688            MVI B,88H       ;RESERVE 13T*8.5YS*88=TX
0114                    LP:
0114    10FE            DJNZ LP
0116    C1              POP B
0117    C9              RET
                        ;
                        ;
0118                    PULSRD:         ;->LOW
0118    3E04            MVI A,4H        ;CS- = LOW
011A    D3F0            OUT PORT
011C    CD 0111         CALL WAIT
011F    3E06            MVI A,6H
0121    D3F0            OUT PORT
0123    DBF0            IN PORT
0125    F5              PUSH PSW
0126    CD 0111         CALL WAIT
0129    F1              POP PSW
012A    C9              RET
012B                    PULSWR: ;->LOW OUTDAT ->HIGH HIGHIMP
                                ;IN AKKU LSB IST DATA
012B    F5              PUSH PSW
012C    3E04            MVI A,4
012E    D3F0            OUT PORT
0130    F1              POP PSW
0131    E601            ANI 1
0133    F604            ORI 4
0135    D3F0            OUT PORT
0137    CD 0111         CALL WAIT
013A    F602            ORI 2   ;TO HIGH
013C    D3F0            OUT PORT
013E    CD 0111         CALL WAIT
0141    E6FE            ANI 0FEH        ;0=HIGH IMPEDANZ
0143    D3F0            OUT PORT
0145    C9              RET



                        ;
0146                    READ:
0146    21 0109         LXI H,BUFFER
```

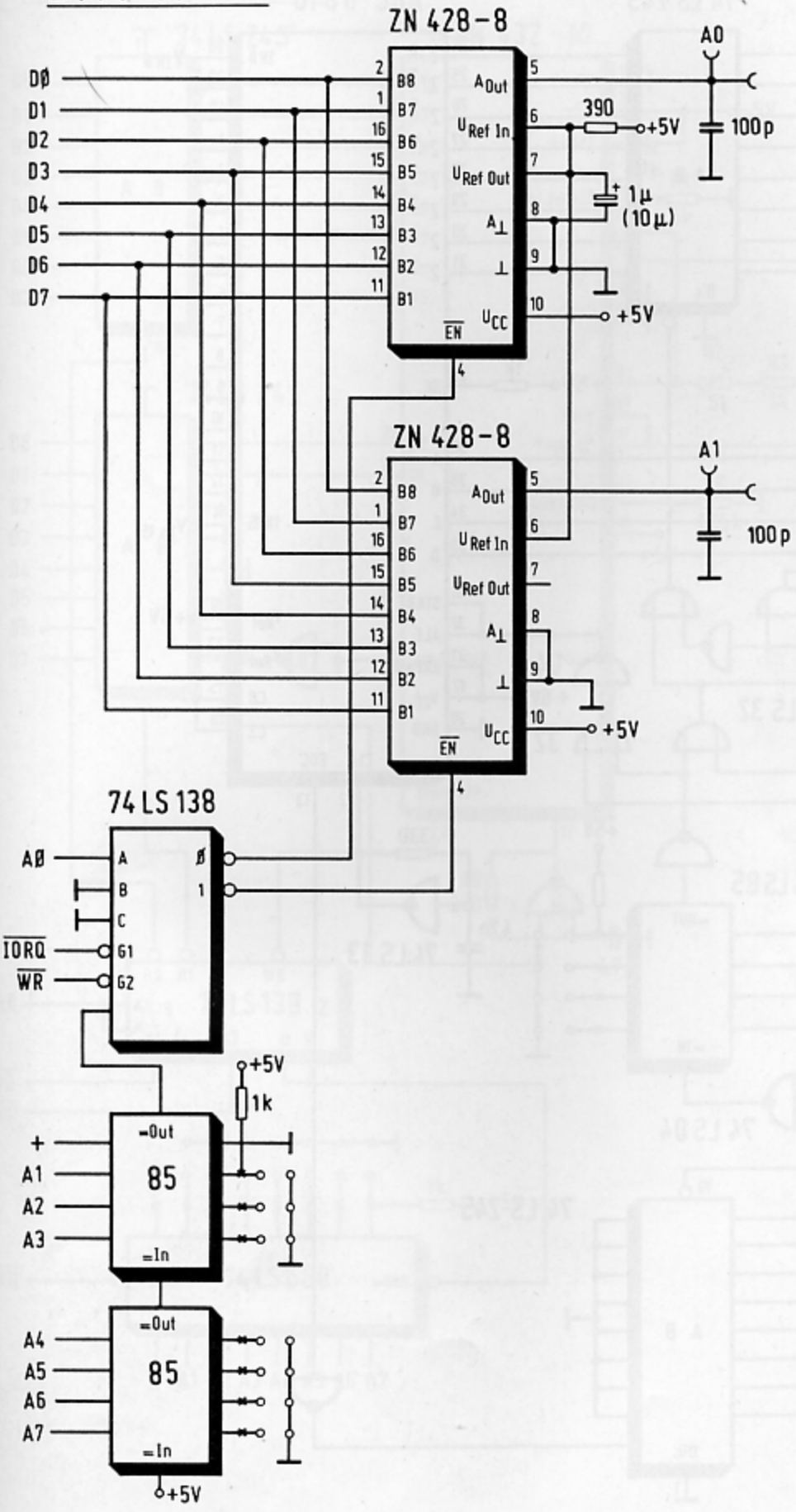
Listing des Uhrenprogramms

```
0149    0603            MVI B,3
014B    3E04            MVI A,4
014D    D3F0            OUT PORT        ;DREI START ADDR=HIGH
014F                    LP1:
014F    AF              XRA A           ;0(1) AUSGEBEN INVERTED
0150    CD 012B         CALL PULSWR
0153    10FA            DJNZ LP1
0155    AF              XRA A
0156    CD 012B         CALL PULSWR     ;0(1) AUSGEBEN READ
                                                COMMAND
0159    0E07            MVI C,7 ;ALLE DATENWERTE
015B                    LPH:
015B    0608            MVI B,8 ;8 BITS JEWEILS
015D    AF              XRA A   ;ERGEBNISBYTE
015E                    LP2:
015E    F5              PUSH PSW
015F    CD 0118         CALL PULSRD
0162    E601            ANI 1
0164    0F              RRC     ;NACH BIT 7
0165    5F              MOV E,A
0166    F1              POP PSW
0167    0F              RRC     ;LSB FIRST
0168    B3              ORA E   ;NA.....
0169    10F3            DJNZ LP2
016B    77              MOV M,A
016C    23              INX H
016D    0D              DCR C
016E    20EB            JRNZ LPM
0170    3E08            MVI A,8
0172    D3F0            OUT PORT
0174    C9              RET
                        ;
0175                    WRITE:
                                ;SCHREIBEN NACH UHR
0175    3E04            MVI A,4
0177    D3F0            OUT PORT
0179    21 0109         LXI H,BUFFER
017C    0603            MVI B,3
017E                    LP11:
017E    AF              XRA A
017F    CD 012B         CALL PULSWR
0182    10FA            DJNZ LP11
0184    3E01            MVI A,1 ;8 NACH UHR
0186    CD 012B         CALL PULSWR     ;INVERTED DATA!
0189    0E07            MVI C,7 ;GESAMT
018B                    LPMM:
018B    0608            MVI B,8
018D    7E              MOV A,M ;DATEN HOLEN
018E    23              INX H
018F                    LP22:
018F    07              RLC     ;MSB FIRST TO BIT 0
0190    F5              PUSH PSW
0191    2F              CMA     ;INVERTED
0192    E601            ANI 1
0194    CD 012B         CALL PULSWR
0197    F1              POP PSW
0198    10F5            DJNZ LP22
019A    0D              DCR C
019B    20EE            JRNZ LPMM
019D    3E06            MVI A,6
019F    D3F0            OUT PORT
01A1    C9              RET
                        ;
                        ;
                        .END
```

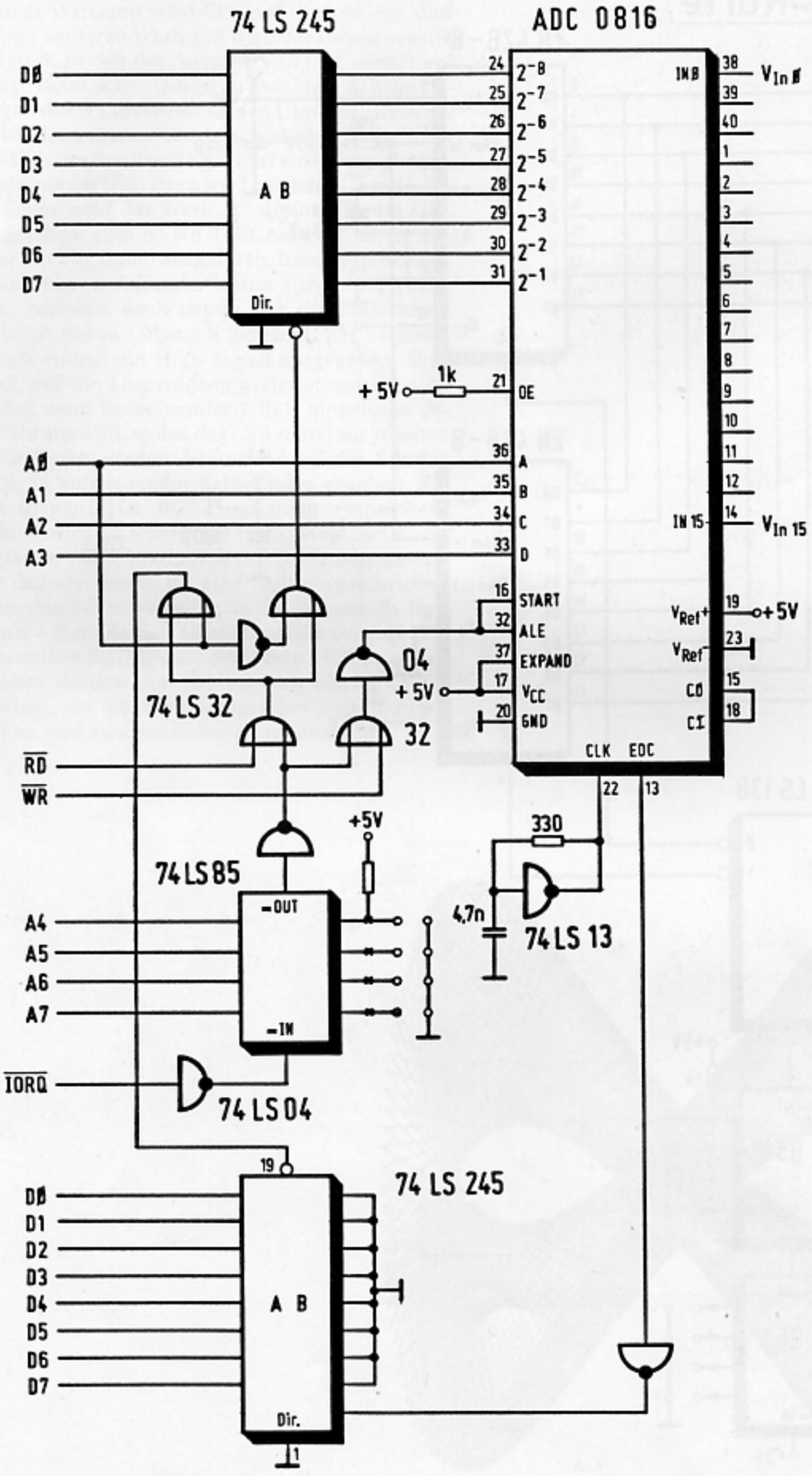
PULSWR schreibt ein Bit in das Uhren-IC ein. Dazu wird zunächst $\overline{CS}$ auf Low gelegt, dann werden die Daten angelegt. Auch CLK ist zu diesem Zeitpunkt Low. Nach einer Wartezeit wird CLK auf High gelegt, und nach einer weiteren Wartezeit wird der Datenausgang desaktiviert, so daß der Ausgang von LT1 wieder auf Low liegt. Beim Schreiben ist zu beachten, daß die Daten wegen des N1 invertiert an den Uhrenbaustein gegeben werden müssen. Dies geschieht erst in der Routine WRITE. Die Routine READ liest einen kompletten Datensatz aus der Uhr. Dazu wird zunächst $\overline{CS}$ auf Low gelegt. Dann wird der Wert „1" dreimal in die Uhr eingeschrieben. Dies ist ein 3-Bit-Adresse. Ist sie auf 111, werden alle Daten ausgelesen, bzw. eingeschrieben. Mit anderen Adressen lassen sich gezielt Zeit, Jahr etc. auslesen, doch wurde hier der Einfachheit halber nicht davon Gebrauch gemacht. Als nächstes wird noch einmal ein High-Signal ausgegeben. Dies bedeutet, daß ein Lesevorgang gestartet werden soll. Es werden dann nacheinander 7 Byte eingelesen. Jedes besteht aus 8 Bit, wobei das LSB zuerst ausgegeben wird. Die Daten werden, beginnend bei der Adresse BUFFER, in aufsteigender Reihenfolge abgelegt. Sie werden in gepackter BCD-Darstellung gespeichert. Am Schluß wird $\overline{CS}$ wieder auf High gelegt. Beim Unterprogramm WRITE verhält es sich im Prinzip genauso, nur daß als viertes Bit eine Null eingeschrieben wird, um den Schreibvorgang anzukündigen. Es werden dann 7 Byte an den Uhrenbaustein ausgegeben, wobei jeweils 8 Bit beginnend mit dem MSB invertiert ausgegeben werden. Am Schluß wird wieder $\overline{CS}$ auf High gelegt, um einen nachfolgenden Zugriff zu ermöglichen und synchronisieren zu können.■



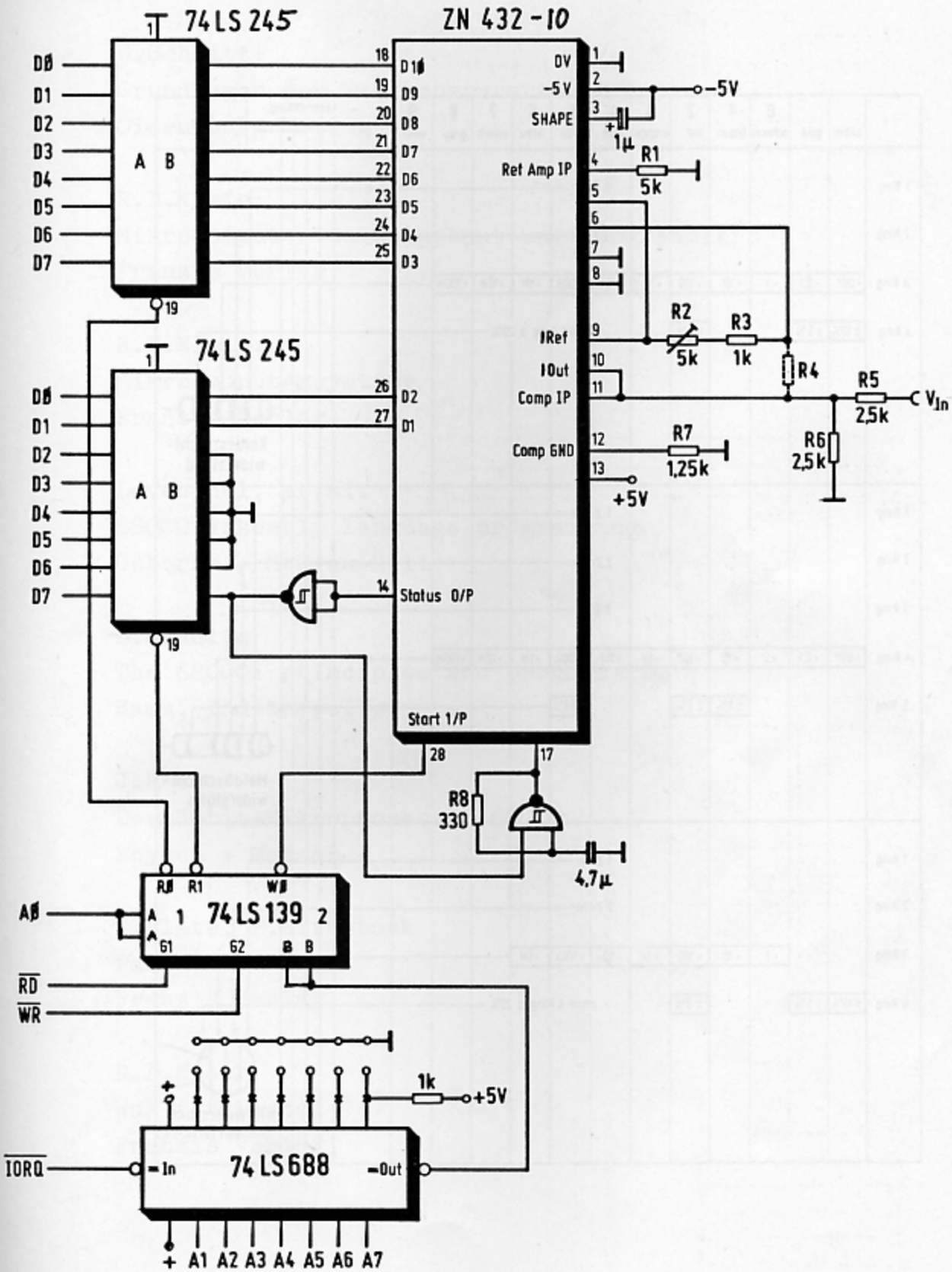
D/A-Karte


ZN 428-8
D0
D1
D2
D3
D4
D5
D6
D7
B8
B7
B6
B5
B4
B3
B2
B1
EN
A Out
U Ref In
U Ref Out
A⊥
⊥
UCC
A0
390
+5V
100p
1µ
(10µ)
ZN 428-8
A1
74 LS 138
A0
A
B
C
G1
G2
IORQ
WR
0
1
+5V
1k
85
=Out
=In
A1
A2
A3
A4
A5
A6
A7

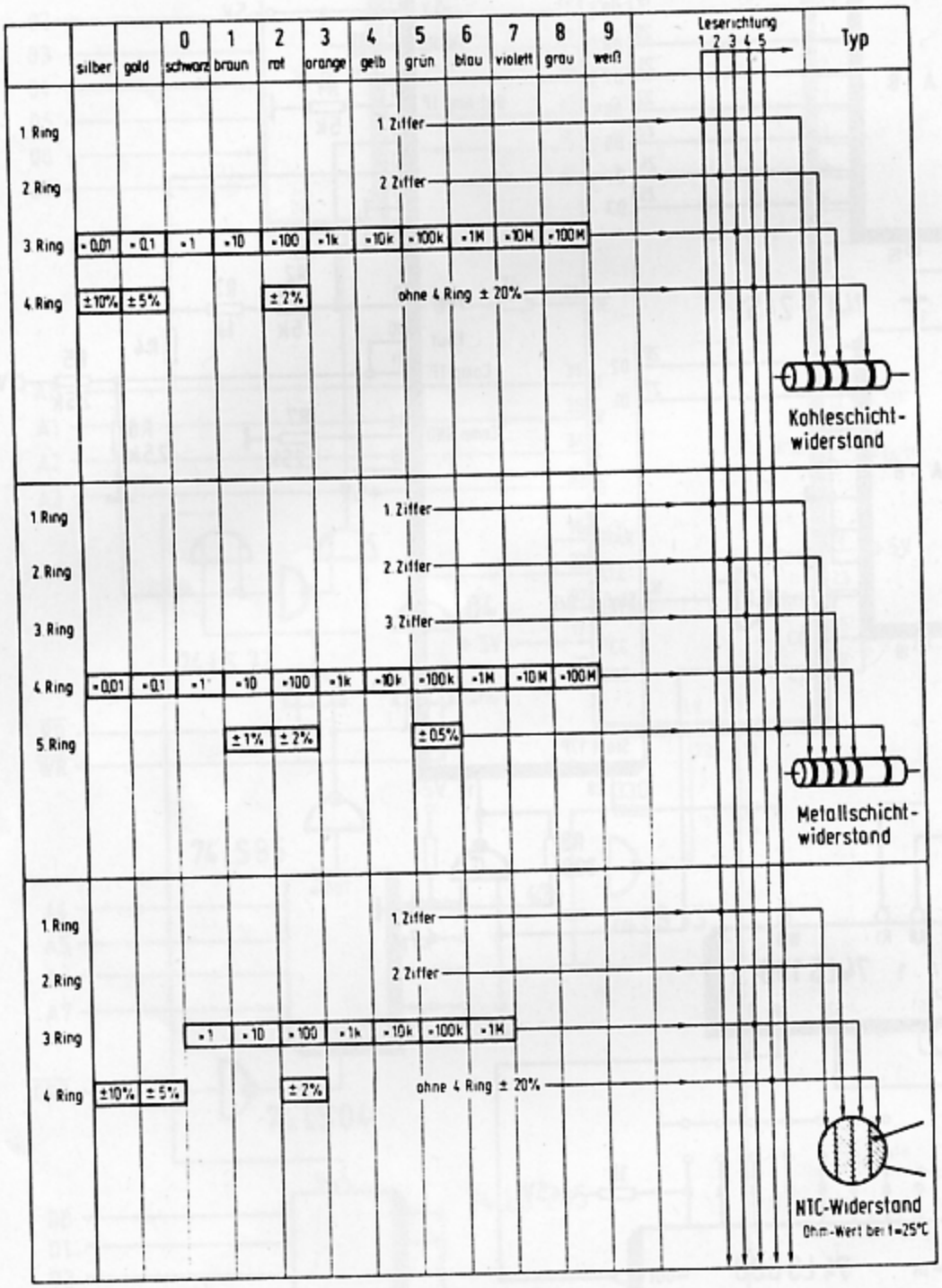
AD 8×16 - Karte


74 LS 245
ADC 0816
DØ
D1
D2
D3
D4
D5
D6
D7
A B
Dir.
2⁻⁸
2⁻⁷
2⁻⁶
2⁻⁵
2⁻⁴
2⁻³
2⁻²
2⁻¹
IN Ø
IN 15
VIn Ø
VIn 15
+5V
1k
OE
AØ
A1
A2
A3
A
B
C
D
START
ALE
EXPAND
VCC
GND
VRef+
VRef-
CO
CI
CLK
EOC
74 LS 32
04
32
RD
WR
74LS85
-OUT
-IN
A4
A5
A6
A7
330
4,7n
74 LS 13
IORQ
74 LS04
74 LS 245

AD 10×1-Karte


74LS245
ZN 432-10
74LS245
D0
D1
D2
D3
D4
D5
D6
D7
A B
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
0V
-5V
SHAPE
Ref Amp IP
IRef
IOut
Comp IP
Comp GND
Status O/P
Start 1/P
+5V
1µ
R1
5k
R2
5k
R3
1k
R4
R5
2,5k
R6
2,5k
R7
1,25k
R8
330
4,7µ
V_{In}
74LS139
A0
RD
WR
IORQ
1k
74LS688
In
Out
A1 A2 A3 A4 A5 A6 A7

Farbcode bei Widerständen



Literatur

G.Schmitt:
Grundlagen der Mikrocomputertechnik
Oldenbourg Verlag

R.D.Klein:
Mikrocomputer selbstgebaut und programmiert
Franzis Verlag

R.D.Klein:
Mikrocomputersysteme
Franzis Verlag

Leventhal, et.al.
68000 assembly language programming
Osborne - McGraw-Hill

L.Scanlon
The 68000: principles and programming
Sams, Indianapolis

J.Koch
Der 16bit-Mikroprozessor 68000
Boysen + Maasch

J.Plate, P.Wittstock
Pascal
Franzis Verlag

R.D.Klein
Was ist Pascal ?
Franzis Verlag

Bezugsquellen:

1. Platinen und Bauteile, Bausätze:

Graf Elektronik Systeme GmbH
Magnusstraße 13
Postfach 1610
8968 Kempten

Giesler & Danne GmbH
Wilhelm Mellies Straße 88
4930 Detmold 18

2. Begleitbuch: Klein, Mikrocomputer selbstgebaut - Zeitschrift mc:

Jede Buchhandlung oder

Franzis Verlag GmbH
Karlstr. 37-41
8000 München 2

3. Arbeitsmaterialien, Programmlistings:

Franzis-Software-Service
Karlstraße 41
8000 München 2

4. Videokassetten mit allen Folgen des Kurses:

Franzis-Verlag GmbH
Karlstr. 37-41
8000 München 2

Achtung!

Nur bei den angegebenen Firmen erhalten Sie die Original-Bausätze und -Unterlagen zur Fernsehserie.

Berichtigungen Berichtigungen Berichtigung

Korrekturen zum Begleitbuch

(R.D.Klein: Mikrocomputer selbstgebaut und programmiert)

Seite	Korrektur
25	Abb 2.3.3 7406 anstatt 7606
27	Abb 2.3.7 Die Steuereingänge des 74245 sind nicht invertiert (am Treiberbaustein)
81	Abb 5.5.13 Der Ausgang 7 des 74LS 138 liegt an Pin 7 und nicht wie gezeichnet an Pin 8.
86	Abb 4.6.1 Die Gatter I7 und I8 sind aus dem 7400, nicht aus dem 7404.
91	Abb 4.7.1 IC B1 : Pin 1 = Dir (geht an $\overline{RD}$) Pin 19= Enable
104	Zeile 6 muß lauten: C3 1223
112	Zeile 13 muß lauten: ...Leitung PHI2(Pin 27)...
124	Abb 5.2.1 Der Widerstand gegen +5V ganz oben sollte statt 1k einen Wert von 3,3k haben (Wert ist unkritisch)
136	Abb 5.3.9 J1 muß heißen: J2 J2 muß heißen: J1 R1 = 390 , R5 = 390 Beim Baustein 2716 sind die Bezeichnungen $\overline{CS}$ und $\overline{OE}$ vertauscht. usgang D (Pin 6) von Z1 muß an +5V liegen. Bei Gen.1 ist der Masseanschluß Pin 7 und nicht wie gezeichnet Pin 8. Ebenso liegt +5V an Pin 14 und nicht an Pin 16.
161	Abb 5.3.28 Der Kondensator 10 Mikrofarad links oben ist mit falscher Polarität gezeichnet - der Pluspol muß nach rechts zeigen.
165	letzte Zeile streichen
166	Zeile 2 Nach der Zeile einfügen: "so ergeben sich 4 Bildseiten mit 256 x 512 Bildpunkten, "
167	Kasten unter b4 b3 Δy statt Δx bei "Dimension" : Δx or Δx

171	Stückliste	EF9366 statt EF0366
185	Zeile 12	$\overline{e_1}$ und $\overline{e_2}$ statt $\overline{e_0}$ und $\overline{e_1}$
191	Abb 5.6.7	R10 zwischen rechter Diode und Widerstand 100k hat einen Wert von 1k statt 10k
373	Abb 7.4.3	Poti Mot2 geht an Pin 12 des 74LS327 Poti Mot1 geht an Pin 2 des 74LS327
374	Abb 7.4.4	Die Diode 1N2002 vom Kollektor des obersten Transistors TIP120 muß an +Spg. Motor und nicht an +5V liegen, da sonst die Motorspannung auf 5V begrenzt wird.
406	Firmentab.	Die Firma Albatros existiert nicht!
324	ff	Bei der Version 1.2 des Grundprogramms arbeiten die Funktionen "HEBE" und "SENKE" nicht richtig.

Änderung an der GDP64-Karte

Um zusammen mit der CPU 68008 höhere Geschwindigkeit erreichen zu können ist die folgende Änderung der GDP64-Karte empfehlenswert.

Bei 12 MHz ein Wartezyklus, bei 8MHz kein Wartezyklus.

Der Inverter ist noch frei in IC 9. Pin 12 wird mit Pin 18 des EF9366 verbunden und Pin 13 mit $\overline{RD}$. Die Leitung von $\overline{WR}$ zu Pin 18 des EF9366 wird unterbrochen. Die Verbindung zum IC 21 muß natürlich erhalten bleiben.

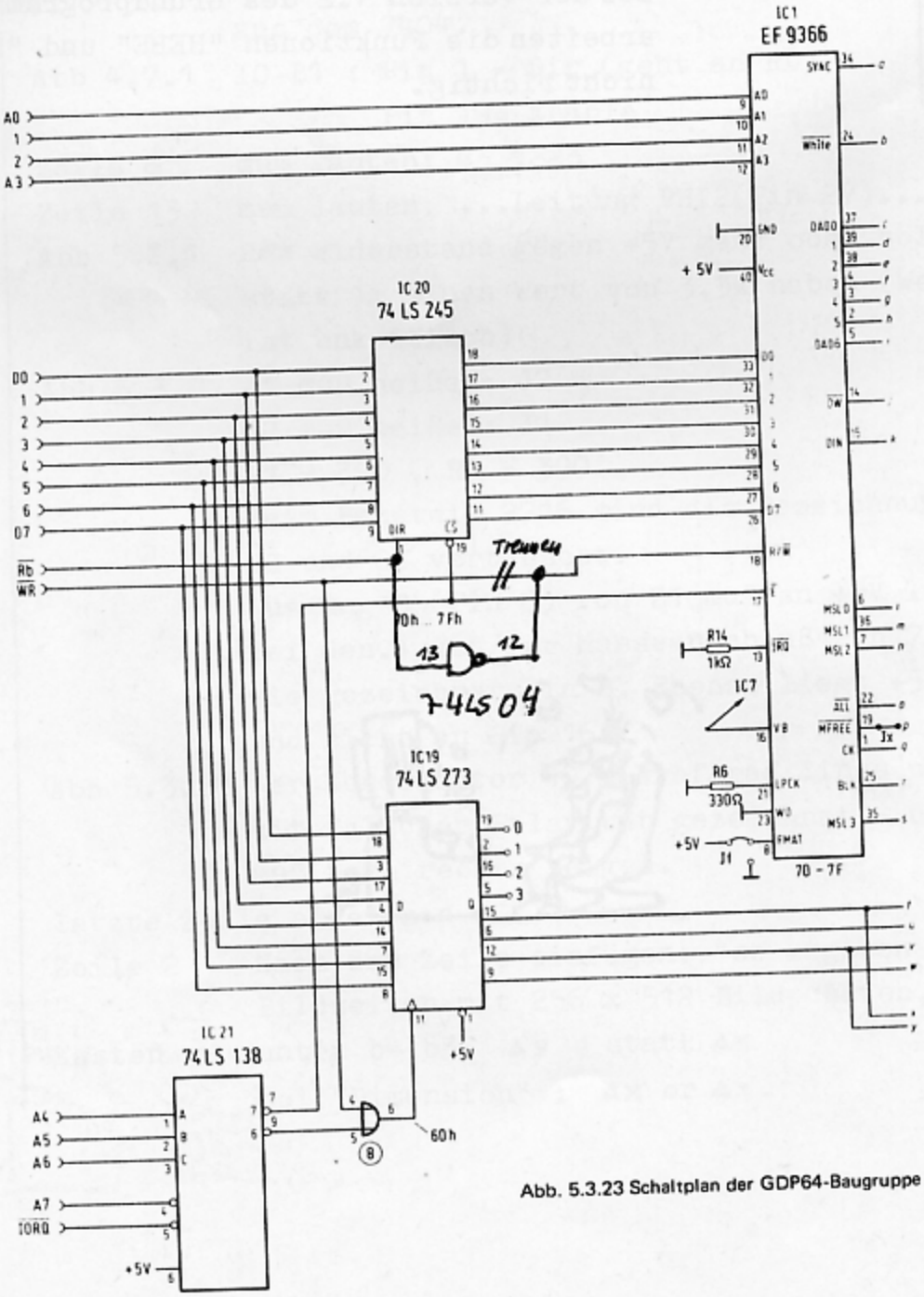
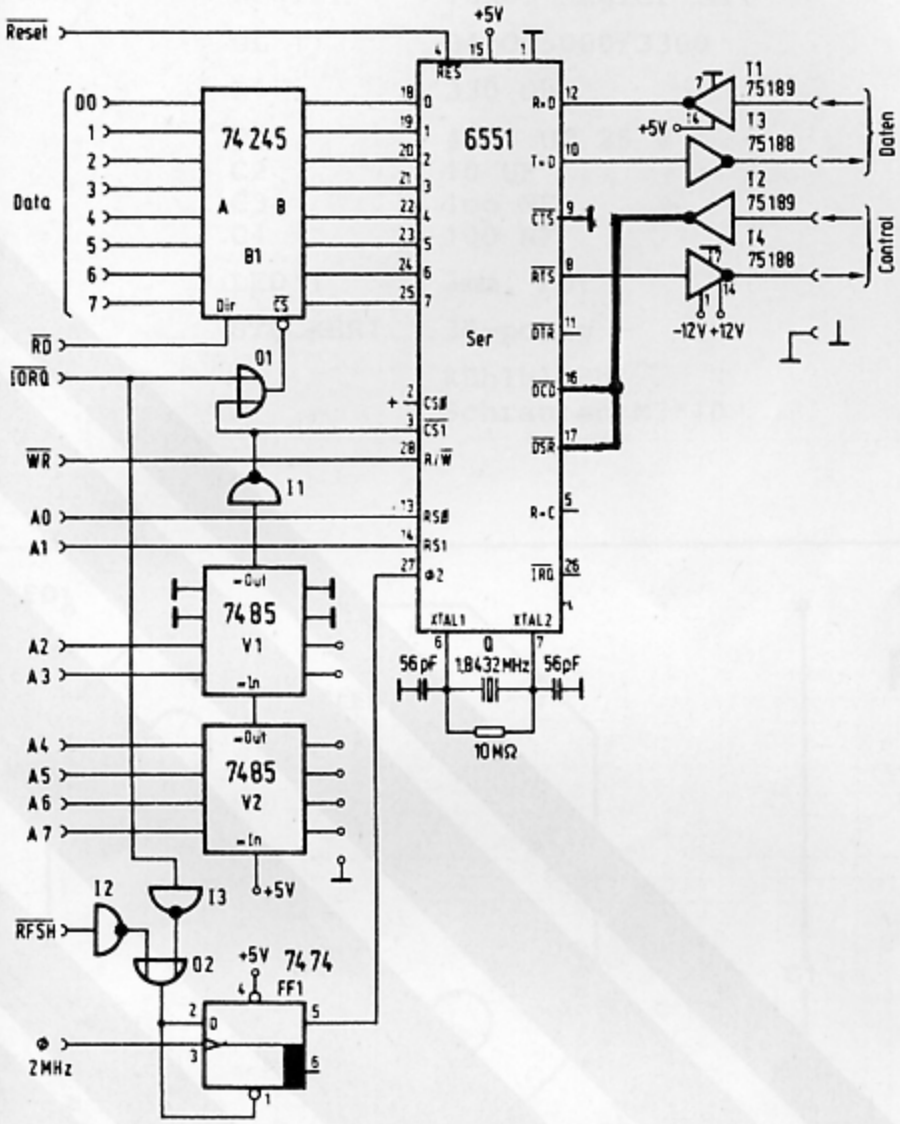
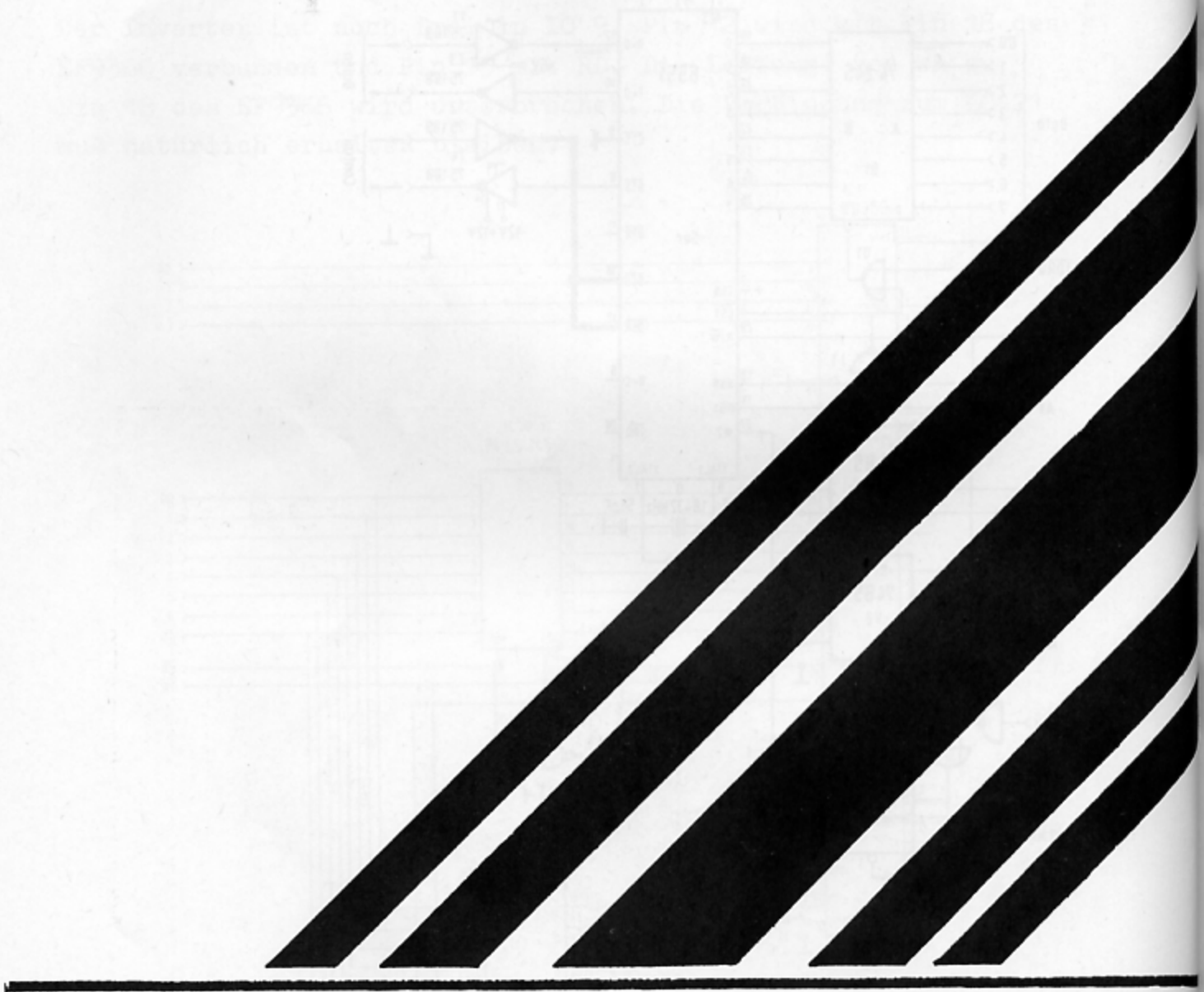


Abb. 5.3.23 Schaltplan der GDP64-Baugruppe

Änderung des seriellen Interface



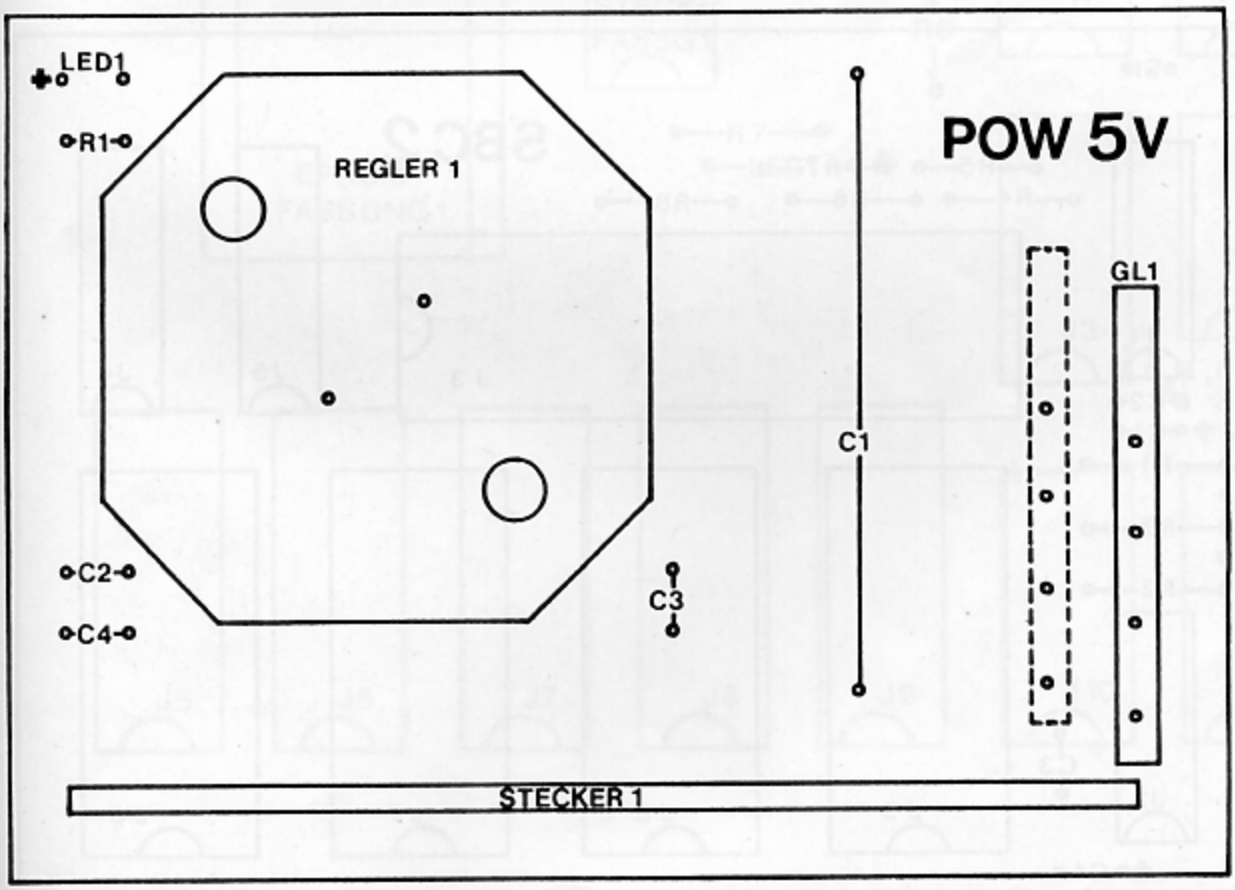


StuecklistenStueckl i stenStueckl i stenStue

P O W 5 V

STÜCKLISTE

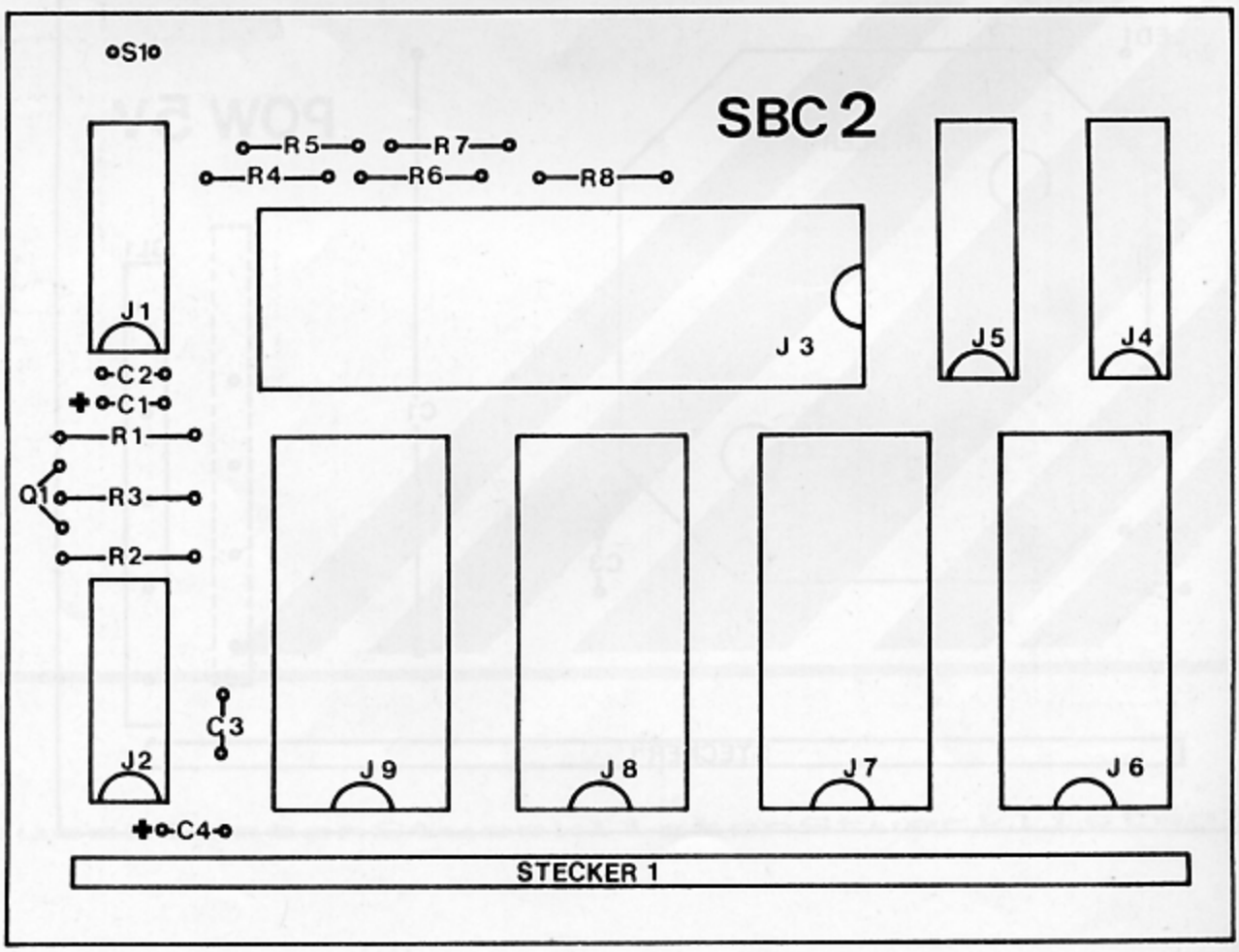
Regler 1	78H05 Regler 3A
GL 1	B4oC 5000/3300
R1	330 Ohm
C1	4700 UF 25 V
C2	10 UF
C3	1oo NF
C4	100 NF
LED 1	3mm, rot
STECKER1	36-polig
	Kühlblech
	Schrauben M3*10



S B C 2

STÜCKLISTE

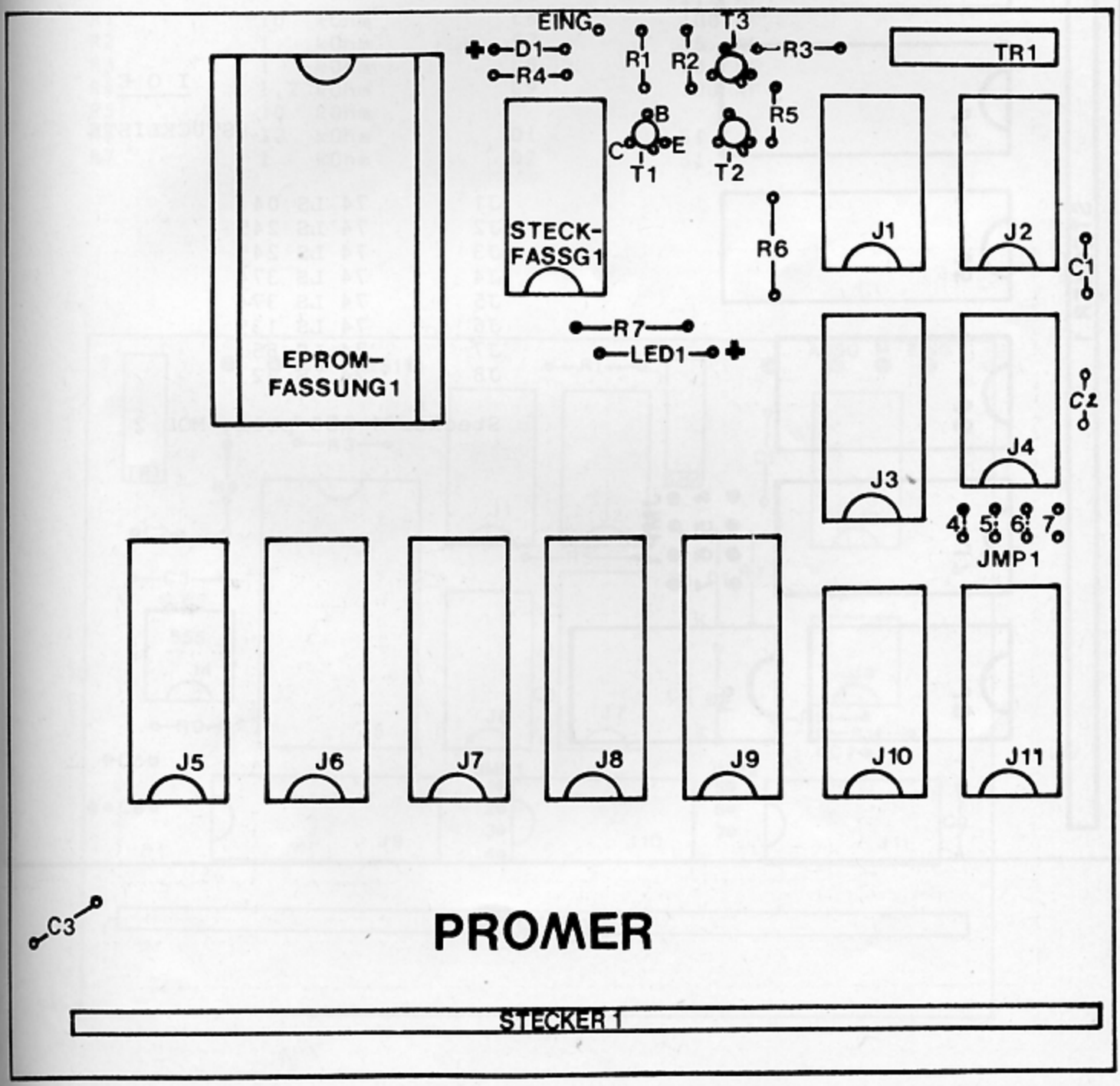
J1	74 121	
J2	74 LS 04	keinen 74 04 verwenden!
J3	Z 80 A	
J4	74 LS 138	
J5	74 LS 138	
J6	(2732 A, Grundprogramm oder Basic)	
J7	(2732 A, Grundprogr. od. Basic)	
J8	6116 LP3	
J9	6116 LP3	
R1	10 kOhm	
R2	1 kOhm	
R3	1 kOhm	
R4	1 kOhm	
R5	1 kOhm	
R6	1 kOhm	
R7	1 kOhm	
R8	330 Ohm	
C1	10 UF	
C2	100 NF	
C3	10 NF	
C4	10 UF	
Q1	4,000 MHz	
STECKER1	36-polig	

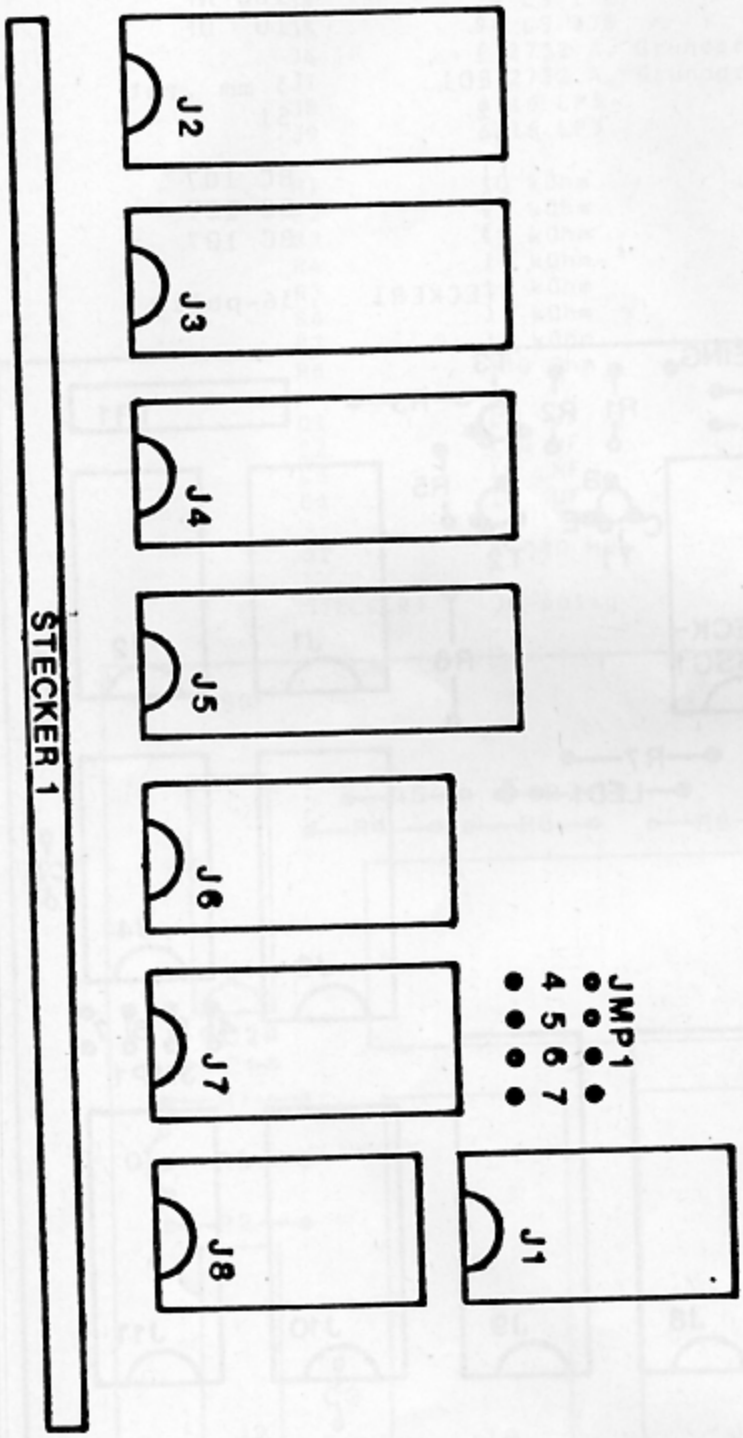


P R O M E R

STÜCKLISTE

J1	74 06
J2	74 121
J3	74 LS 138
J4	74 LS 00
J5	74 LS 245
J6	74 LS 245
J7	74 LS 374
J8	74 LS 273
J9	74 LS 273
J10	74 LS 138
J11	74 LS 85
R1	10 kOhm
R2	10 kOhm
R3	1 kOhm
R4	100 Ohm
R5	10 Ohm
R6	1 kOhm
R7	330 Ohm
TR1	10 kOhm
C1	10 UF
C2	100 NF
C3	10 UF
LED1	3 mm ,rot
D1	SI
T1	BC 107
T2	BC 107
T3	BC 107
STECKER1	36-polig





I O E

STÜCKLISTE

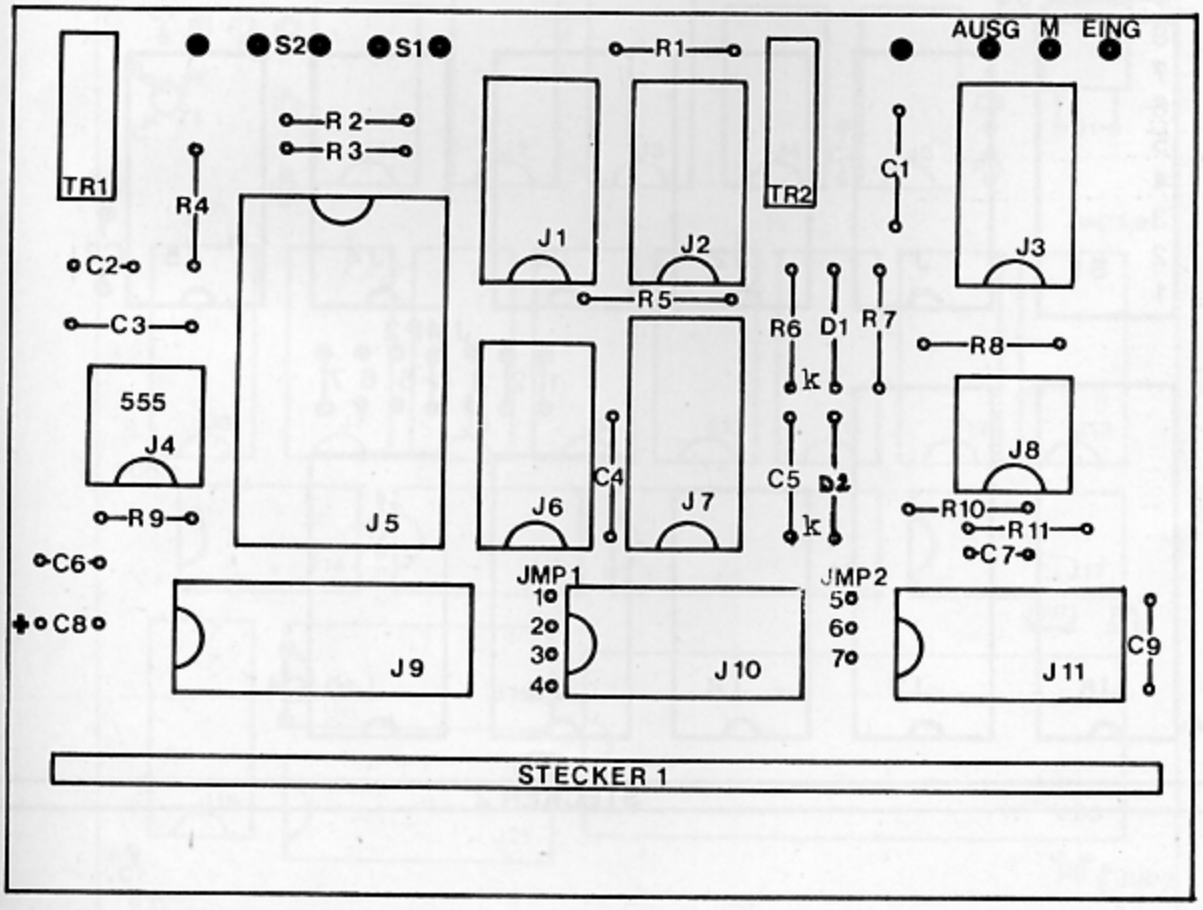
J1	74 LS 04
J2	74 LS 245
J3	74 LS 245
J4	74 LS 374
J5	74 LS 374
J6	74 LS 139
J7	74 LS 85
J8	74 LS 32

Stecker 1 36-polig MOD 2

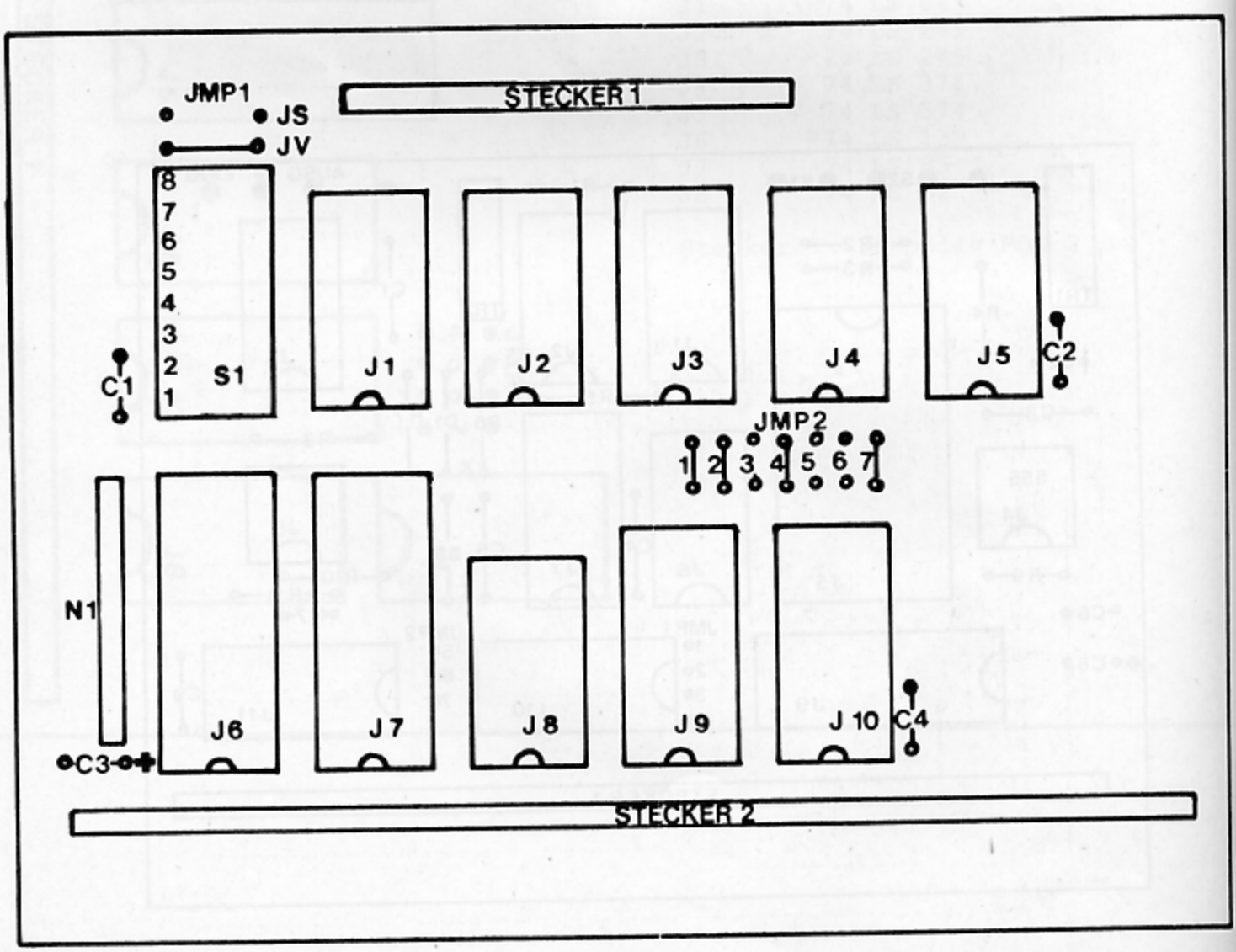
C A S

STÜCKLISTE

J1	4070	R8	100 kOhm
J2	4047	R9	2,2 kOhm
J3	74 LS 74	R10	1 kOhm
J4	555	R11	1 kOhm
J5	6850	TR1	1 kOhm
J6	74 LS 00	TR2	5 kOhm
J7	4528		
J8	3130	C1	100 NF
J9	74 LS 245	C2	10 NF
J10	74 LS 85	C3	100 NF
J11	74 LS 85	C4	2,2 NF
		C5	2,2 NF
R1	10 kOhm	C6	100 NF
R2	1 kOhm	C7	56 NF
R3	1 kOhm	C8	10 UF
R4	1,2 kOhm	C9	100 NF
R5	10 kOhm		
R6	10 kOhm	D1	SI
R7	1 kOhm	D2	SI



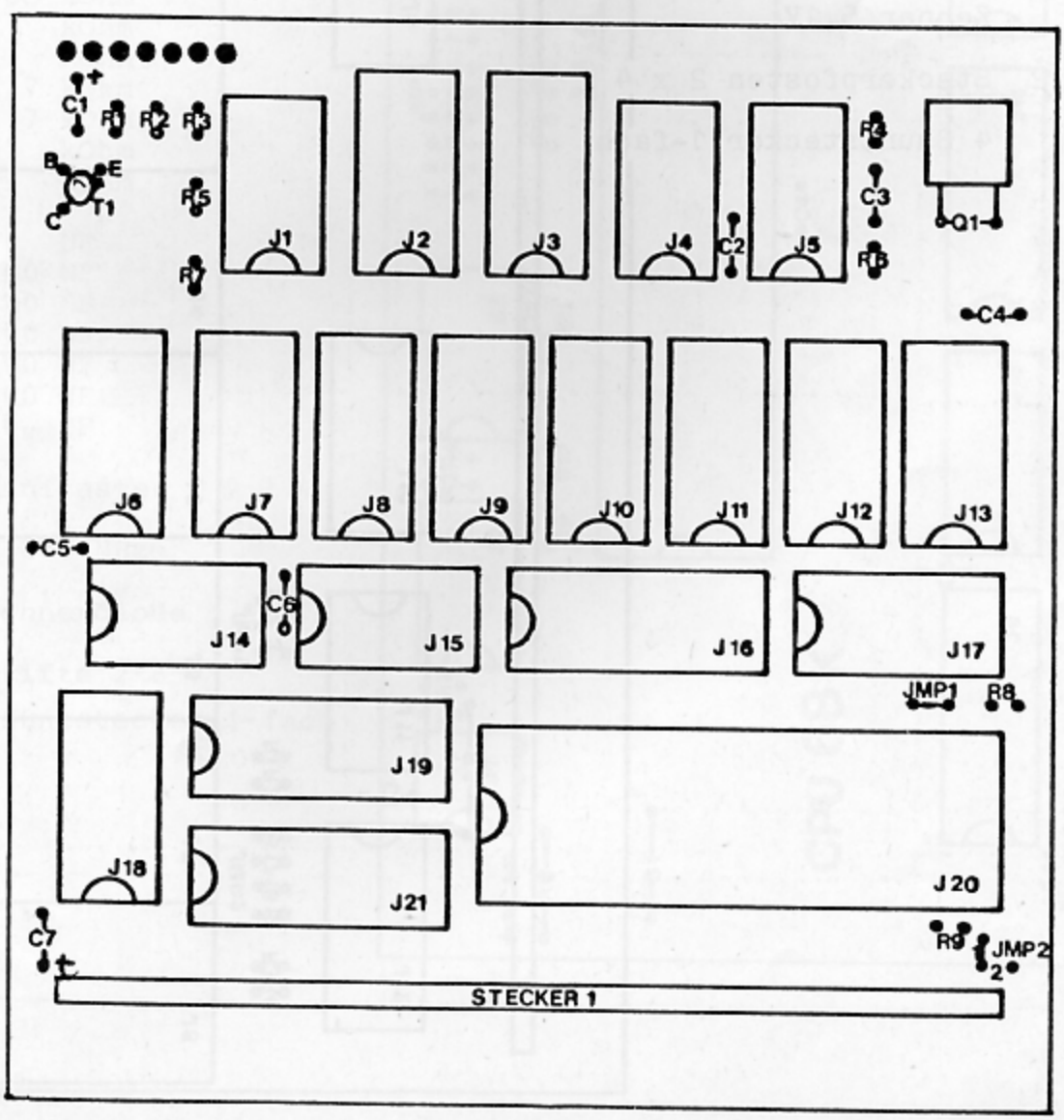
J1	74 LS 86	K E Y
J2	74 LS 86	
J3	74 LS 04	STÜCKLISTE
J4	74 LS 74	
J5	74 LS 00	
J6	74 LS 245	
J7	74 LS 374	
J8	74 LS 32	
J9	74 LS 85	
J10	74 LS 85	
C1	100 NF	
C2	100 NF	
C3	10 UF	
C4	100 NF	
STECKER1	15-polig	
STECKER2	36-polig	
N1	Netzwerkwiderstand 8*3,3 kOhm	
S1	DIL-Schalter 8-fach	



STÜCKLISTE

J1	74 05
J2	74 LS 166
J3	74 LS 163
J4	74 LS 00
J5	74 04
J6	4164 200 NS
J7	4164 200 NS
J8	4164 2oo NS
J9	4164 200 NS
J10	4164 200 NS
J11	4164 200 NS
J12	4164 200 NS
J13	4164 200 NS
J14	74 LS 74
J15	74 LS 32
J16	25 LS 2538
J17	74 LS 153
J18	74 LS 138
J19	74 LS 245
J20	GDP 9366
J21	74 LS 273

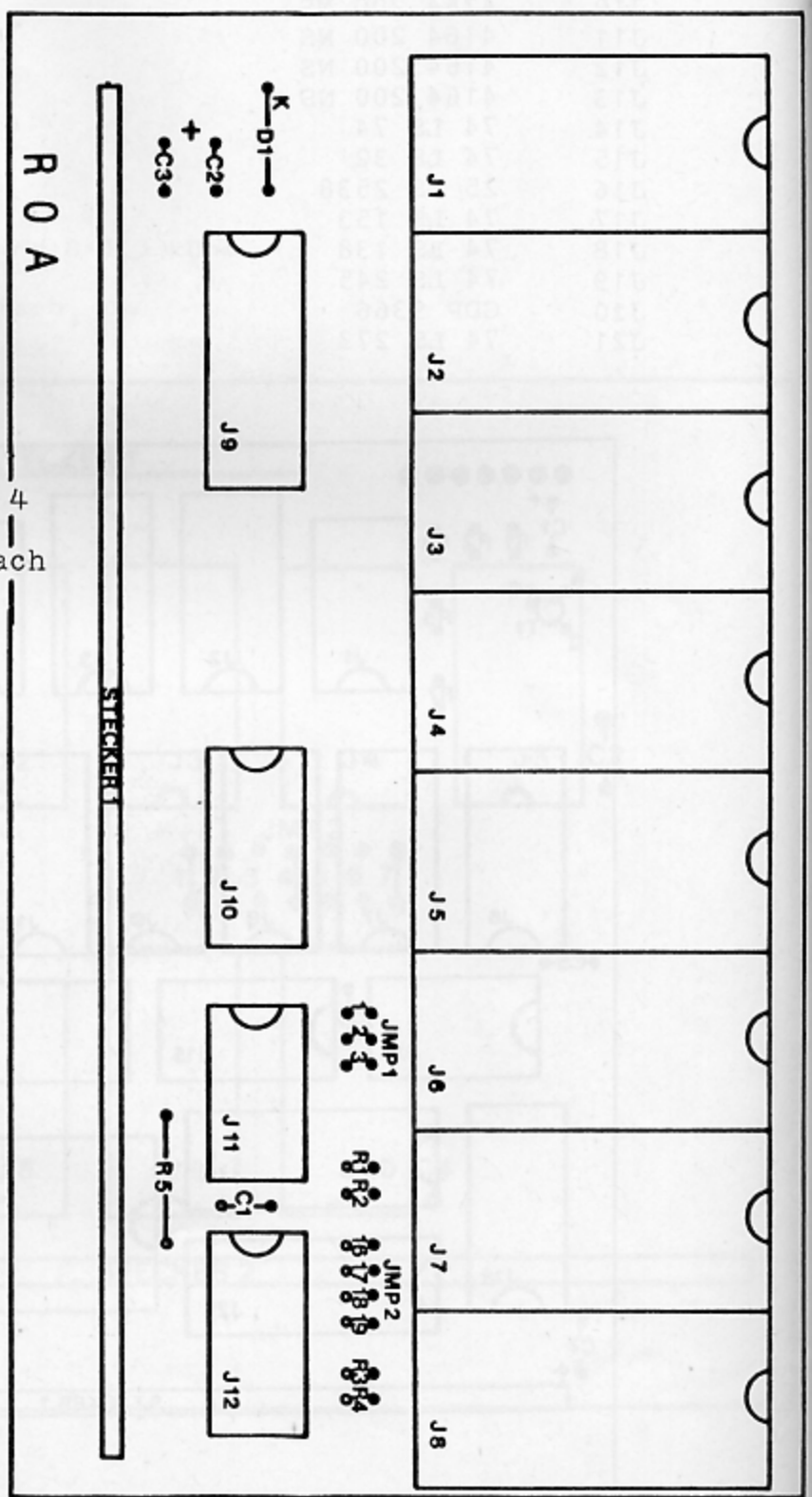
R1	75 Ohm
R2	1 kOhm
R3	470 Ohm
R4	1 kOhm
R5	15o Ohm
R6	1 kOhm
R7	470 Ohm
R8	33o Ohm
R9	1 kOhm
C1	10 UF
C2	100 NF
C3	100 NF
C4	100 NF
C5	100 NF
C6	100 NF
C7	10 UF
T1	BC 1o7
Q1	14,00 MHz
STECKER 1	36-polig



R O A 6 4

STÜCKLISTE

J1	ROM oder RAM	(Grundprogramm/Pascal)
J2		-"-
J3		-"-
J4		-"-
J5		-"-
J6		-"-
J7		-"-
J8		-"-
J9	74 LS 245	
J10	74 LS 138	
J11	74 LS oo	
J12	74 LS 85	
R1	1 kOhm	
R2	1 kOhm	
R3	1 kOhm	
R4	1 kOhm	
R5	1 kOhm	
C1	100 NF	
C2	100 NF	
C3	10 UF	
D1	Zehner 5,1V	
JMP2	Steckerpfosten 2 x 4	
	4 Shuntstecker 1-fach	



DatenblaetterDatenblaetterDatenblaetterD

Inhalt

1984

Bearbeitet vom Franzis-Software-Service
Für den Inhalt verantwortlich: Dipl.-Inform. Jürgen Plate

Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
Printed in Germany. Imprimé en Allemagne.

Druckfehlerberichtigungen zum Schaltplanheft

S. 12,	12. Zeile:	...30 Watt. VDE
	17. Zeile:	...(1.4142 * Effektivspannung)...
	vorletzte Zeile:	...POW 5V...von 5V...
S. 14,	2. Zeile:	...der SBC-II-Karte
	10. Zeile:	...der Leds L3 und L4...
S. 16,	4. Zeile:	...Ausgänge IORQ, MREQ, RD...
S. 27,	7. Zeile:	...C9
S. 28,	10. Zeile:	...2A xx.W
	11. Zeile:	...22 xx.W
	13. Zeile:	...ENDE:=C9
	15. Zeile:	...LADE 50
	16. Zeile:	...RUFE SCHREITE
S. 29,	17. Zeile:	...(Pin 3 des 6850)...
S. 36,	10. Zeile:	...Motor für 5..6V...
S. 39,	3. Zeile von unten:	...Abgleich 50 ms...
S. 41,	3. Zeile:	...TRAP...

	6. - 14. Zeile:	Quadrat:
		MOVE #4-1,D3
		LOOP:
		MOVE #50,D0
		JSR SCHREITE
		MOVE #90,D0
		JSR DREHE
		DBRA D3,LOOP
		RTS

	Letzte Zeile:	...Adresse $ 48,$ 49).
S. 46,	Vorletzte Zeile:	...INT nach NMI
S. 51,	5. Zeile:	...(3...4RAMs) +...
	11. Zeile	... auf 50H.
S. 73,	7. Zeile:	...DC.B $55,$AA,$01,$80
S. 74,	2. Zeile:	...IOE-Karte: OUT 0:Datenbits