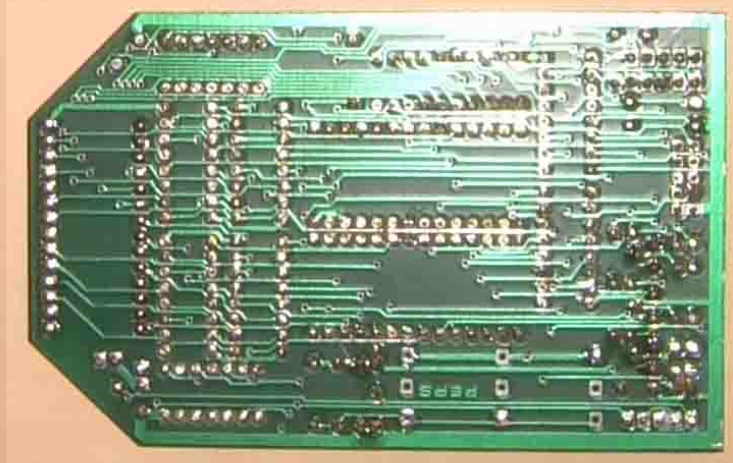
PEPS
HANDBUCH



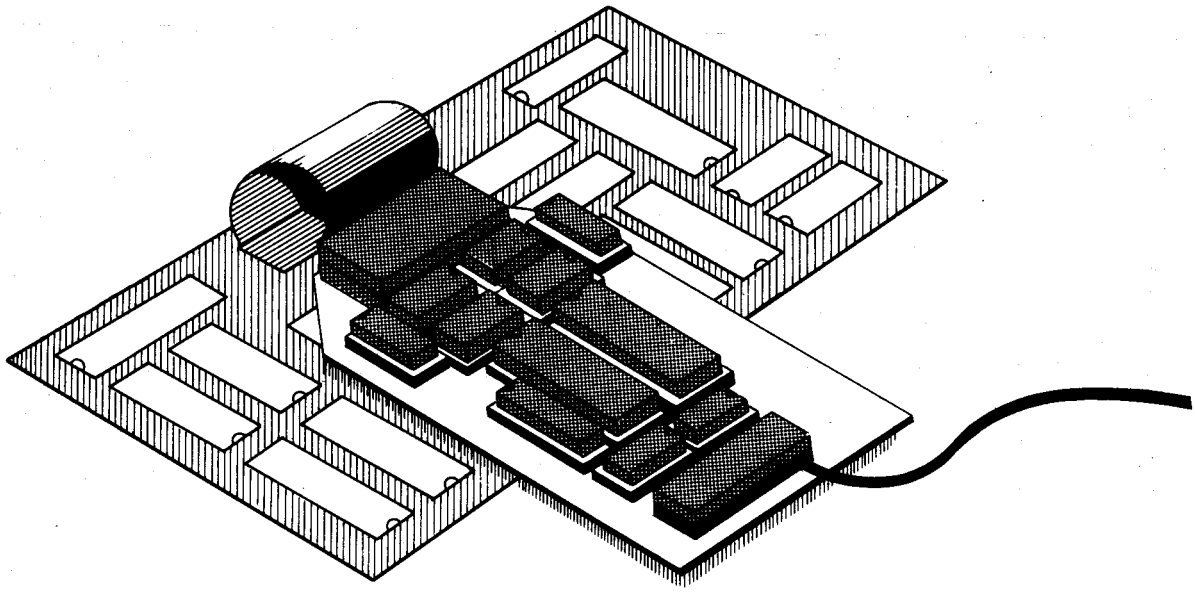
CONITEC
DATENSYSTEME

P E P S

F u n k t i o n e n

PEPS ist ein Instrument zum Entwickeln von EPROM-software ("Firmware") für Microcomputer oder intelligente Steuerschaltungen (z.b. CEPAC-80). Dabei übernimmt PEPS (Programmierbarer EPROM-Simulator) die Funktion des EPROMS und erlaubt so - ähnlich wie ein CPU-Emulator - das schnelle Austesten und Ändern der Software.

PEPS wird einfach in den EPROM-Sockel eingesteckt oder über ein kurzes Flachbandkabel damit verbunden (Bild). Das interne RAM kann vom Entwicklungssystem aus mit dem auszutestenden Programm geladen werden. Zur Verbindung ist nur ein fünfadriges Kabel nötig.



Da PEPS für das Testobjekt wie ein EPROM aussieht, liegt eine weitere Anwendung auf der Hand: Das Übertragen von Daten zwischen Rechnern, die sonst absolut (Schnittstellen-)inkompatibel sind. Für solche Fälle kann PEPS batteriegepuffert werden, so daß nicht einmal eine Kabelverbindung notwendig ist.

Während der Datenübertragung oder während des Testlaufs ist PEPS ständig mit Entwicklungs- und Testsystem verbunden. Die Stromversorgung erfolgt wahlweise über die Testschaltung oder den Host-Rechner; die Stromaufnahme ist nur etwa halb so groß wie die des simulierten EPROMs.

PEPS enthält zwei Sockel für RAM-Bausteine (6116 oder 6264); damit lassen sich EPROMs von 2-16 KByte simulieren (2716, 2732, 2764, 27128). Beim Einsatz von 24-poligen IC's (6116) bleiben die Pins 1, 2, 27 und 28 frei.

Die Programmierung von PEPS erfolgt über vier Steuerbits, die an vier Ausgangsleitungen des Hostrechners anzuschließen sind, z.b. an eine Z80-PIO oder eine Centronics-Schnittstelle. Der Programmierzusatz PROMMER-80

ist ebenfalls dafür geeignet.

S t e u e r l e i t u n g e n

Über M0 und M1 wird der Betriebsmodus von PEPS eingestellt. Es gibt vier Modi:

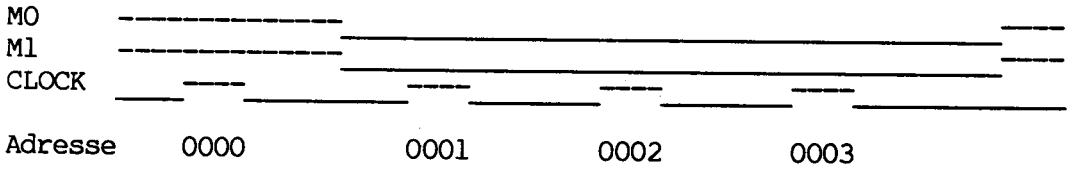
M1	M0	Modus
0	0	Adresse hochzählen (Modus 0)
0	1	Daten einschieben (Modus 1)
1	0	Daten einschreiben (Modus 2)
1	1	Simulieren (Modus 3)

Adresse zurücksetzen

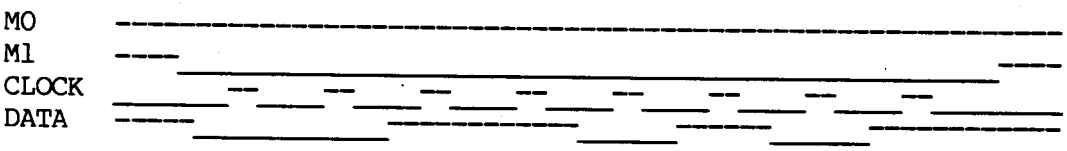
Die CLOCK-Leitung löst die durch M0 und M1 vorgewählte Aktion aus. Diese Leitung ist high-aktiv und muß im Ruhezustand oder beim Modus-Wechsel einen Low-Pegel annehmen. Über die serielle DATA-Leitung wird im Modus 1 das Datenbyte eingelesen.

Um ein Datenbyte an eine beliebige PEPS-Adresse zu schreiben, sind drei Schritte erforderlich.

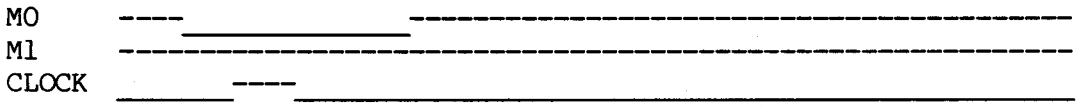
Adresse einstellen: Im Modus 3 setzt ein Low-Pegel am CLOCK-Eingang den internen Adresszähler auf Null. Im Modus 0 wird die eingestellte Adresse bei jeder positiven CLOCK-Flanke um eins erhöht. Im Prinzip läßt sich so durch Rücksetzen und wiederholtes Hochzählen jede beliebige Adresse einstellen. Die einmal eingestellte Adresse bleibt bei Modus-wechsel erhalten.



Datenbyte einschieben: Dazu muß Modus 1 gewählt werden. Nun wird das Byte seriell über die DATA-Leitung übertragen. Bei jeder negativen CLOCK-Flanke wird dabei der jeweilige Pegel am DATA-Eingang als Bit übernommen. Bit 0 wird immer zuletzt übertragen. Ein einmal eingeschobenes Byte bleibt beim Moduswechsel erhalten. Das dargestellte Impulsdiagramm zeigt das Einschreiben des Bytes 35h = 00110101b.



Daten einschreiben: Modus 2 ist einzustellen. Das im Modus 1 eingelesene Byte wird nun mit einem High-Pegel am CLOCK-Eingang an die in Modus 0 eingestellte Adresse geschrieben.



Zur Beschleunigung der Übertragung lassen sich einige Schritte abkürzen. Da normalerweise die Daten blockweise übertragen werden, kann man das Einstellen der Adresse auf einfaches Hochzählen nach jedem übertragenen Byte beschränken. Sollen identische Daten - z.b. Nullen - in verschiedene Speicherzellen geschrieben werden, genügt einmaliges Einlesen des Bytes, gefolgt von wiederholtem Schreiben.

Nach Beendigung des Datentransfers kann der Simulier-Modus - Modus 3 - eingestellt werden. Er wird mit einem High-Pegel am CLOCK-Eingang aktiviert. PEPS verhält sich nun wie ein EPROM, das mit den entsprechenden Daten "gebrannt" wurde.

S c h a l t u n g

Der Schaltplan ist ziemlich simpel. Die beiden RAM's Z8 und Z9 bilden den eigentlichen EPROM-Simulator. Ihre Adressen erhalten sie über die vier umschaltbaren Multiplexer Z2-5 entweder vom EPROM-Sockel oder von den beiden 7-Bit-Adresszählern Z1 und Z4. Das CMOS-Schieberegister Z11 nimmt die seriellen Daten vom Host-Rechner auf und dient als Zwischenspeicher zum späteren Einschreiben in's RAM. Für das Umschalten der Modi ist der Dekoder Z10 zuständig. Der Tristate-Treiber Z7 trennt den internen Datenbus vom den Datenleitungen des EPROM-Sockels.

Für Spezialanwendungen kann PEPS batteriegepuffert werden, so daß die eingeschriebenen Daten auch bei Abschalten der Spannung erhalten bleiben. Zum Schutz des RAM bei Spannungsabfall dienen die Transistoren T1-T3.

Im Normalfall - also ohne Batteriepufferung - genügt es, an passiven Bauelementen die Kondensatoren C1-C3, C5, R12 und die Leuchtdiode zu bestücken. D2, R8 und T1-T3 werden durch Drahtbrücken ersetzt, alle anderen passiven Bauteile entfallen.

P E P S

J u m p e r

PEPS kann vier verschiedene EPROM-Typen simulieren. Die Bestückung und die Einstellung der Jumper J1 und J2 richtet sich nach dem jeweiligen Typ.

EPROM	Sockel Z8	Sockel Z9	J2	J3
2716	6116 (2K)	leer	gebrückt	A,D
2732	6116 (2K)	6116 (2K)	gebrückt	A,C
2764	6264 (8K)	leer	offen	B,D
27128	6264 (16K)	6264 (16K)	offen	B,D

J1 und J2 sind zu öffnen, wenn die Stromversorgung für PEPS vom Entwicklungssystem aus erfolgen soll. J4, J5 und J6 sind für eventuelles Programmieren von einem EPROM-Programmierer aus vorgesehen und bleiben im Normalfall unbestückt.

V e r b i n d u n g

PEPS wird normalerweise mit einem 24- oder 28-poligen Flachbandkabel mit dem EPROM-Sockel verbunden. Alternativ dazu kann ein DIL-Stecker senkrecht zur Platine unter N1 gelötet werden; dann läßt sich die Karte direkt in den Sockel einstecken.

Für die Verbindung zum Entwicklungssystem ist der zehnpolige Stecker N-12 vorgesehen. Es genügt, die vier Steuerleitungen und eine Masseleitung anzuschließen; die +5V-Leitung ist nur dann erforderlich, wenn das Entwicklungssystem die Stromversorgung mit übernehmen soll.

Die Steuerleitung können an vier beliebige Ausgangsleitungen angeschlossen werden. Beim Anschluß an eine Centronics-Schnittstelle werden vier Datenleitungen für M0, M1, DATA und CLOCK verwendet.
Die Daten müssen zwischen den einzelnen Strobe-Impulsen stabil bleiben. Das ist in der Praxis für alle uns bekannten Centronics-Schnittstellen gewährleistet.

Die Software für den PEPS

Damit mit dem EPROM-Simulator überhaupt gearbeitet werden kann, benötigt man ein Programm, daß die Daten vom Hostrechner (also dem jeweiligen Entwicklungssystem) zur PEPS-Hardware überträgt.

Das hier vorgestellte Programm, wir werden es im weiteren PEPS.COM nennen, ist für CP/M-80-Computer (z.B. PROF-80) bestimmt. Der größte Teil des Programms dient der Befehlsdekodierung und der Datenaufbereitung. Die eigentliche Übertragung zum PEPS übernimmt ein relativ kurzes Unterprogramm mit Namen LPEPS. Besitzer eines anderen Computers mit 8080

CPU (oder einer CPU, die 8080-Code verarbeiten kann) können dieses Unterprogramm übernehmen; es muß lediglich die Parallel-Ausgabe angepaßt werden, dazu jedoch später mehr. Wer einen Rechner mit anderer CPU sein Eigen nennt, kann das Unterprogramm LPEPS für seinen Rechner abwandeln.

Wie wird PEPS an den Computer angeschlossen ?

Die PEPS-Hardware benötigt zu seiner Ansteuerung insgesamt vier Signalleitungen, die wie folgt mit einem Parallel Port verbunden werden:

Bit 0 mit M0
Bit 1 mit M1
Bit 2 mit Data
Bit 3 mit Clock

Zusätzlich muß noch Masse und falls gewünscht die 5-Volt-Spannungsversorgung mit der PEPS-Hardware verbunden werden.

PEPS.COM sollte angepaßt werden

PEPS.COM funktioniert auch ohne spezielle Anpassung, wenn der Hostrechner eine Centronics-Schnittstelle besitzt, die über den CP/M List-Kanal anzusprechen ist. Diese Lösung hat jedoch zwei Konsequenzen:

- Die Centronics-Schnittstelle muß so beschaltet werden, das der BIOS-Treiber immer "Drucker bereit" erkennt und ein Zeiichen ausgeben kann. Um dies zu erreichen, genügt es in den meisten Fällen, den BUSY-Eingang der Centronics-Schnittstelle auf Masse zu legen.

- Die zweite Konsequenz ist weitaus unangenehmer: Für jedes Byte, das zur PEPS-Hardware übertragen wird, sind insgesamt 19 Daten-Ausgaben an den Parallel-Port notwendig. Mit anderen Worten: die Dauer der Parallel-Ausgabe-Routine bestimmt entscheidend die Dauer der gesamten Übertragung zur PEPS-Hardware. Die Datenausgabe über den CP/M-List-Kanal dauert aber relativ lange, so daß man mit etwa 15 Sekunden pro Kilo-Byte rechnen muß.

Wer regelmäßig größere Datenmengen in den PEPS "schaufelt", sollte PEPS.COM deshalb besser an seinen Ausgabe-Port anpassen.

Wie wird PEPS.COM angepaßt ?

Bei Adresse 180h steht ein Unterprogramm namens "POUT" (siehe Listing), welches die niederwertigen vier Bit des Akkus an den PEPS-Ausgabe-Port übergibt. Wie oben erwähnt, gibt dieses Programm standardmäßig die Daten an den CP/M-List-Kanal. In den meisten Fällen genügt es, diese Routine durch folgende Befehlsfolge zu ersetzen:

```
OUT  PEPSPORT
RET
```

Mit PEPSPORT ist die Adresse des Parallel Ports gemeint, mit der die PEPS-Hardware verbunden ist. Die Übertragung eines Kilobytes dauert in diesem Falle etwa 250 Millisekunden (6MHz Z80-CPU).

Anmerkung für ganz Eilige:

> Die Übertragungszeit eines Kilobytes an den PEPS kann maximal auf 125 Millisekunden reduziert werden (6MHz Z80 CPU vorausgesetzt), wenn die "CALL POUT" Befehle (insgesamt 7 Stück) in den Routinen LPEPS und WRITEBYTE direkt durch "OUT PEPSPORT" Befehle ersetzt werden.

Zu beachten ist noch, daß POUT kein einziges Register verändern darf.

Je nach Hardwareausführung kann es notwendig sein, daß der Ausgabe-Port vor seiner Benutzung initialisiert werden muß. Zu diesem Zweck besteht die Möglichkeit, bei Adresse 1C0h ein Unterprogramm einzufügen, welches die geforderte Initialisierung übernimmt.

Änderungen "patcht" man am besten mit einem Debugger (DDT oder SID) ein.

Wie wird PEPS.COM bedient ?

Zunächst einige grundsätzliche Bemerkungen:

- Alle Daten werden ab Adresse 0000h in die PEPS-Hardware eingeschrieben.

- Bei Bestückung mit 2K RAM's werden die Daten Modulo 1000h eingeschrieben. D.h. eine Längenangabe über 1000h führt zum Überschreiben der vorhergehenden Daten.

- Bei Bestückung mit 8K RAM's werden die Daten Modulo 4000h eingeschrieben. Eine Längenangabe über 4000h führt in diesem Falle ebenfalls zum Überschreiben der zuerst übertragenen Daten.

PEPS.COM läßt sich auf zwei Arten starten:

- von der CP/M Befehlsebene

- von einem Debugger

Wird PEPS.COM von der CP/M-Befehlsebene gestartet, dann gibt es insgesamt vier Möglichkeiten:

P E P S

- A>PEPS

  PEPS.COM meldet sich und fragt nach der Startadresse und der Länge der zu übertragenden Daten. Nach dieser Eingabe werden die Daten aus dem Arbeitsspeicher zur PEPS-Hardware übertragen.

- A>PEPS dddddddd.xxx

  PEPS.COM liest die Datei, die anstelle von dddddddd.xxx angegeben wird, und überträgt sie, so wie sie sich auf der Diskette befindet, zur PEPS-Hardware.

- A>PEPS dddddddd.HEX

  Bei dddddddd.HEX muß es sich um eine Datei handeln, die im Intel-Hex-Format aufgezeichnet wurde. PEPS.COM liest diese Datei, wandelt die Hexadezimal-Daten in Binär-Daten um und sendet diese zur PEPS-Hardware.

- A>PEPS dddddddd

  Diese Eingabemöglichkeit entspricht der Vorhergehenden; man spart lediglich das Eintippen von ".HEX".

Will man PEPS.COM von einem Debugger starten, dann muß PEPS.COM vorher in den TPA-Bereich geladen werden (und zwar ab 100h). Nachdem dies geschehen ist, (die Daten, mit denen die PEPS-Hardware geladen werden soll, müssen natürlich auch irgendwo im Arbeitsspeicher stehen) hat man zwei Möglichkeiten, die Datenübertragung zu starten:

- #G103

  Es wird die Eingabe von Startadresse und Länge verlangt, danach werden die Daten aus dem Arbeitsspeicher zur PEPS-Hardware übertragen und schließlich wird wieder zur Debugger Befehlsebene zurückgekehrt.

- #G106

  Wie bei dem vorhergehenden Aufruf. Startadresse und Länge können nicht eingegeben werden, sondern werden entsprechend dem letzten G103-Aufruf gesetzt. Wenn man mehrmals Daten übertragen will, dann erspart dies einiges an Tipperei.

Fehlermeldungen und Warnungen

error, no valid hex number

Die eingetippte Startadresse oder Länge war länger als vier Zeichen oder enthielt ungültige Zeichen. Die Eingabe muß wiederholt werden.

error, file dddddddd.xxx not found

Die angegebene Datei befindet sich nicht auf der Diskette.

error, wrong character in Intel HEX-file

Die angegebene HEX-Datei enthielt ungültige Zeichen. PEPS.COM bricht seine Ausführung ab und kehrt auf CP/M Befehlsebene zurück.

checksum error in Intel HEX-file

Bei mindestens einer Aufzeichnung der angegebenen HEX-Datei stimmt die Prüfsumme nicht. PEPS.COM bricht seine Ausführung ab und kehrt auf CP/M Befehlsebene zurück.

warning, no end record in Intel HEX-file

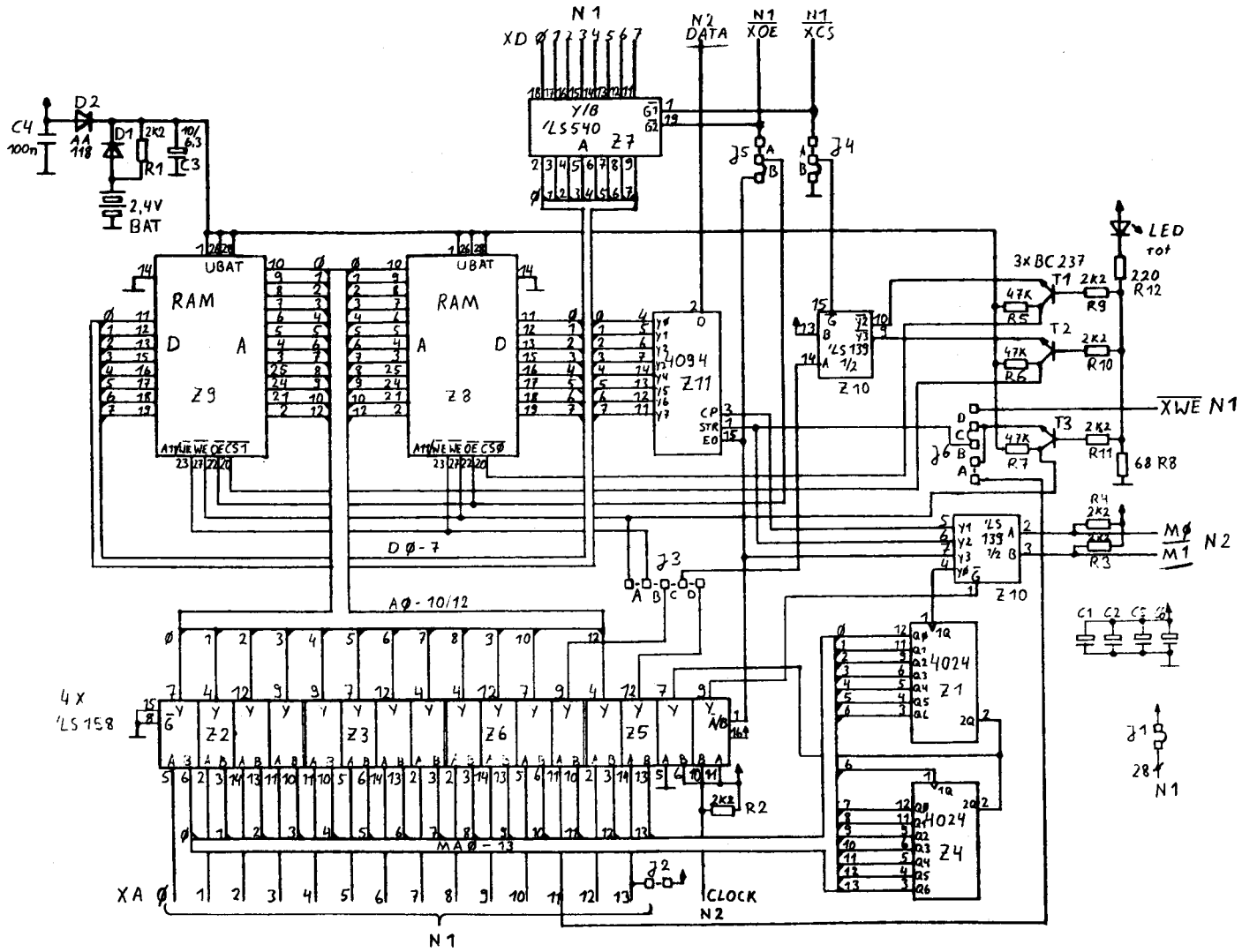
Die angegebene HEX-Datei enthält keine Endaufzeichnung. PEPS.COM simuliert die Endaufzeichnung und arbeitet normal weiter.

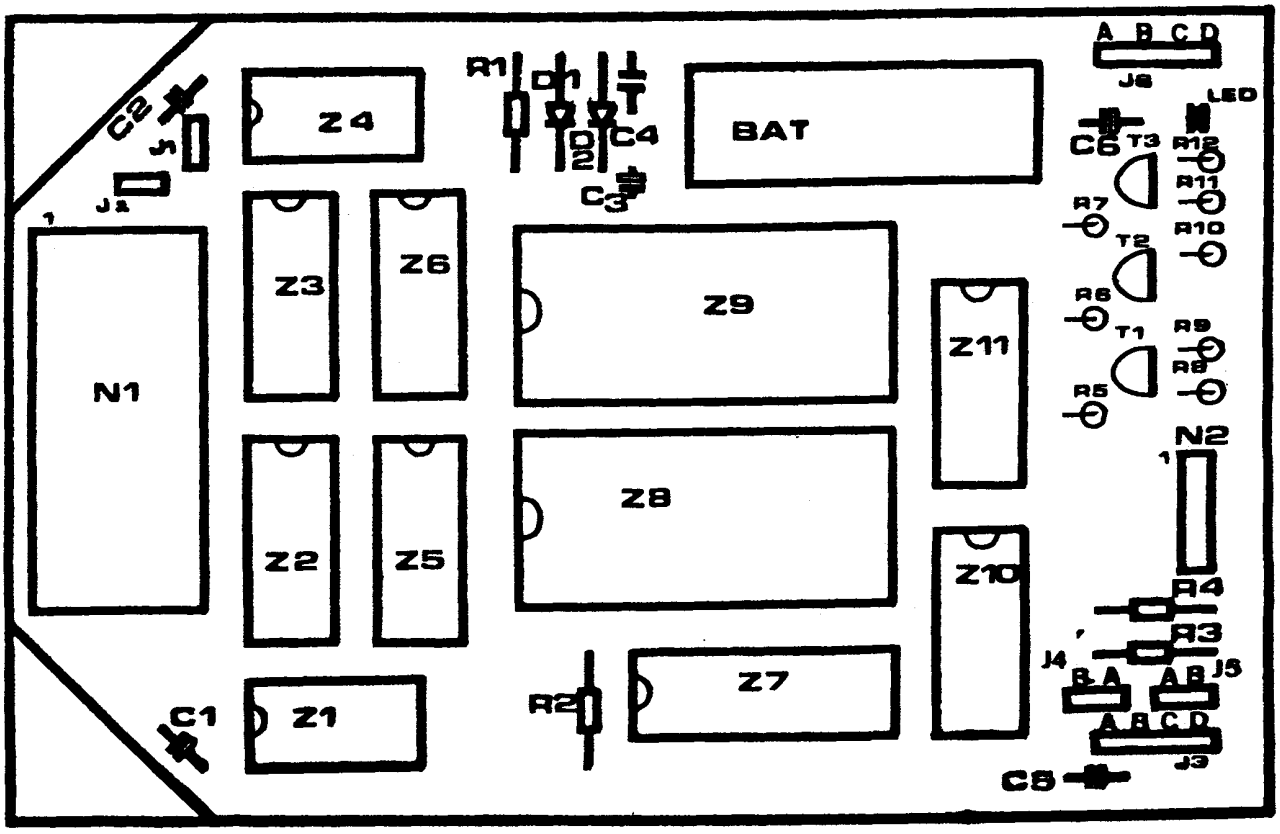
warning, buffer overflow

Die angegebenen Datei ist länger als 4000h. Da dies zum Überschreiben der vorher gesendeten Daten führen würde, begrenzt PEPS.COM die Länge auf 4000h und gibt diese Warnung aus. Anmerkung: wer seinen PEPS mit 2K RAM's bestückt hat, wünscht sich wahrscheinlich, daß die Länge auf 1000h begrenzt wird. Dies ist leicht möglich, indem man die Variable BUFL (siehe Listing) auf den Wert 1000h setzt und neu assembliert.

warning, data in lower buffer

Um diese Warnung zu verstehen, muß man wissen, wie PEPS.COM HEX-Dateien verarbeitet: PEPS.COM liest die erste Aufzeichnung einer HEX-Datei und legt die Daten in seinem internen Buffer ab. Alle weiteren Aufzeichnungen werden relativ zur ersten Aufzeichnung im Buffer abgelegt. Nun kann es vorkommen, daß eine spätere Aufzeichnung eine Anfangsadresse besitzt, die kleiner ist als die Anfangsadresse der ersten Aufzeichnung. PEPS.COM ignoriert einfach alle Aufzeichnungen, die kleiner als die Erste sind und gibt die oben angeführte Warnung aus.


N1
XD Ø 1 2 3 4 5 6 7
N2 DATA
N1 XOE
N1 XCS
Y1Ø 'LS540 A Z7
J5
J4
C4 100n
D2 AA 118
D1
2K2 R1
10/6,3 C3
2,4V BAT
UBAT
RAM D A Z9
RAM A D Z8
A11 WE OE CS1
A11 WE OE CS0
DØ-7
4094 Z11
CP STR EO
'LS139 1/2 Z10
3x BC237
T1
T2
T3
47K R5
47K R6
47K R7
2K2 R9
2K2 R10
2K2 R11
220 R12
68 R8
LED rot
XWE N1
J6
R4 2K2
R3 4K7
MØ N2
M1
'LS 139 1/2 B Z10
C1 C2 C3 C4
J3
AØ-10/12
4X 'LS158
Z2
Z3
Z6
Z5
R2 2K2
4024 Z1
4024 Z4
MAØ-13
J2
J1
28
N1
XA Ø 1 2 3 4 5 6 7 8 9 10 11 12 13
CLOCK N2
N1



PEPS Stückliste

Widerstände

R8	68 Ohm	
R12	220 Ohm	
R1-4,9-11	2.2 kOhm	
R5-6,7	47 kOhm	

Kondensatoren

C1,2,5,6	100 nF	Keramik
C3,4	10 uF	Tantal

Halbleiter

D1,2	AA118
T1-3	BC237
LED	LED rot

ICs

Z10	74LS139	Dekoder
Z2,3,5,6	74LS158	Multiplexer
Z7	74LS540	Buffer 8 Bit
Z1,4	4024	Counter 7 Bit
Z11	4094	Shifter 8 Bit
Z8-9	6116 LP-3	RAM 2Kx8
Z8-9	6264 LP-3	RAM 8Kx8

Stecker

N1	Präzisionsfassung 28pol.
N2	Pfosten 5x2 pol.