

In der vorletzten Ausgabe der Clubzeitung wurde die von Harald Steves "entdeckte" Commodore Diskette mit ihrem DOS vorgestellt.

Aber auch Jan Boerrichter und einige andere Personen haben sich mit dem Anschluß dieser Diskette an den DAI befaßt: Hier wird nun seine Version vorgestellt:

Momentan wird von Commodore das Diskettenlaufwerk VC1541 zu einem sehr günstigen Preis verkauft. Dieses Laufwerk ist ein intelligentes System mit einem eigenen Betriebssystem. Dies war für uns Grund genug uns mit dem Anschluß am DAI zu beschäftigen.

Die VC1541 ist kein schnelles Diskettensystem, aber der Vorteil einen direkten Zugriff auf Dateien zu haben und auf ein ausgezeichnetes Datei-Verwaltungssystem zurückgreifen zu können machen dieses Gerät zu einem sehr nützlichen Speichermedium für den DAI - vor allem wenn man den Preis bedenkt.

Nach einigen Monaten in denen die Hard- und Software entwickelt wurde, um das Laufwerk am DAI anzuschliessen ist nun das "DAI DOS 1541" fertig. Es besteht aus:

- Einem Commodore VC1541 Diskettenlaufwerk
- Einer Interface Karte um das Laufwerk am DCE Bus anzuschliessen
- Einer EPROM/RAM Karte, die auf dem X-Bus installiert wird

Das DAI DOS 1541 ist vollständig kompatibel mit den DAI BASIC Versionen V1.0 und V1.1. Es wird kein RAM Speicher des DAI verwendet!

Es können bis zu 4 VC1541 Diskettenlaufwerke, außerdem noch 4 Memocom DCR's durch das DOS gesteuert werden. Zusätzlich können noch ein "High-Speed-Dataloader" und ein paralleler Drucker angeschlossen werden. Alle diese Peripherieeinheiten werden durch das DOS 100% unterstützt.

Im DAI muß nur eine kleine Änderung vorgenommen werden: eine Leitung muß gelegt werden, um den RESET mit dem X-Bus zu verbinden.

DAI DOS 1541 Befehle:

- Die DCR Kommandos können dem DCR Handbuch entnommen werden. Der DAI Befehlsschatz wird durch die folgenden Befehle erweitert;

Diskettenbefehle:

FDD x : Auswählen eines Laufwerks 0-3
 FORMAT : Eine Diskette formatieren
 SCRATCH : Eine Datei der Diskette löschen
 RENAME : Eine Datei umbenennen
 COPY : Mehrere Dateien verbinden (merge)
 VALID : Reorganisation des Diskettenplatzes
 DIR : Das Inhaltsverzeichnis lesen.
 Über die Cursor Tasten kann eine Datei ausgewählt und gestartet werden
 FVER : Überprüft ein File auf der Diskette auf Schreibfehler

Dateien werden automatisch geöffnet und geschlossen.

Andere Kommandos:

CAS x : Auswählen der Audio Cassette 0-2
RDL x : Daten vom EPROM 0-3 lesen
USR : Sprung zu einem Maschinenprogramm, dessen Start-
adresse im DAI I/O Vektorbereich abgelegt ist
BOOT : Eine Maschinenroutine und ein BASIC Programm laden
UBL : Wie BOOT, aber für Files mit gleichem Namen für
BASIC und MP
LNON : Aktiviert AUTO-Zeilenummerierung
LNOFF : Deaktiviert " "
<tab> : Löscht den Bildschirm
/C : Standard Cursor Zeichen
/D : Nur Ausgabe zum Bildschirm
/E : Eingabe vom Edit-Buffer
/F : wie IMP FPT
/H : Anzeige der I/O Adressen, externer Speicher
und den IMP-Typ (FPT/INT)
/I : wie IMP INT
/M : MODE 0
/P : Anwählen des Parallel Druckers
/S : " " seriellen Druckers
/T : Standard Textfarben
/O : Kombination von /C, <tab> /I, /M und /T

Alle Kommandos können sowohl im direkten Eingabe-Mode, als auch von BASIC Programmen aus benutzt werden.

Es wurden auch noch einige Fehler Meldungen durch das DOS und von der Diskette zu den bisherigen DAI Fehlermeldungen hinzugefügt. U.a. sind zwei ON ERROR GOTO Möglichkeiten vorhanden.

Ein ausführliches Handbuch beschreibt detailliert alle Funktionen. Zusammen mit dem MDCR und VC1541 Handbuch werden Sie alle Fähigkeiten dieses Systems verstehen lernen.

Die Preise:

Das VC1541 Diskettenlaufwerk kann in jedem Computerladen gekauft werden. Hier existieren große Preisunterschiede!

Die Interface und die EPROM Karte zusammen kosten ca. 285 holländische Gulden. Ein Interface-Kabel (nicht notwendig, falls Sie bereits ein Flachkabel von der DCR haben) kostet ca. 50 Gulden extra, kann jedoch auch einfach von Ihnen selbst hergestellt werden.

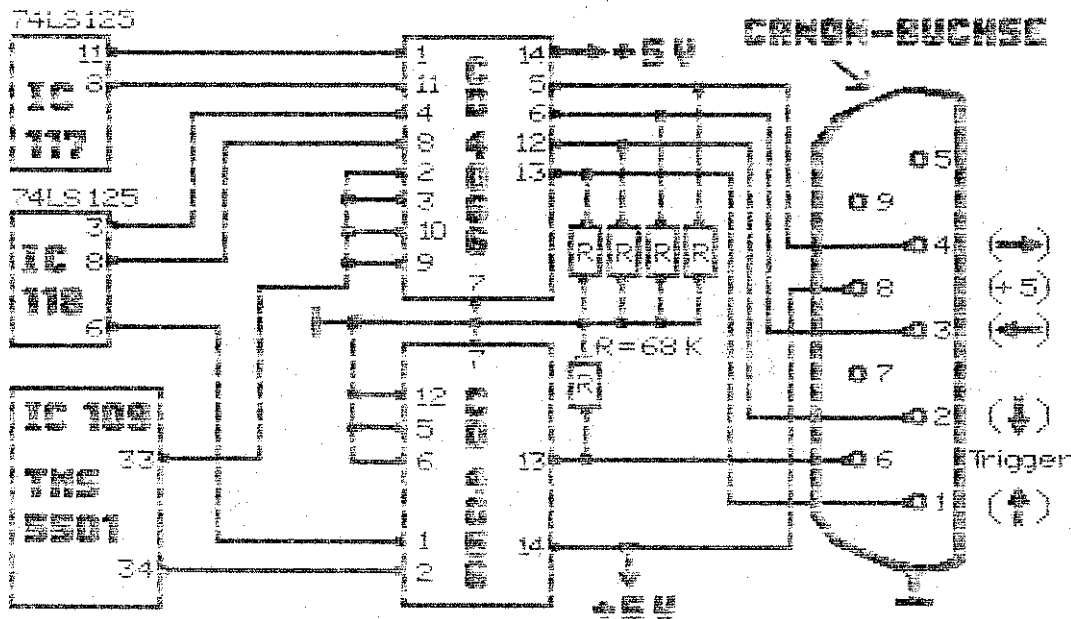
Bezugsmöglichkeiten:

Wenden Sie sich an: Jan Boerrigter
Fabritiusstraat 15
6174 RG Sweikhuizen - Niederlande
☎ 04493-2093 (19-21 Uhr)

Hier erfahren Sie den genauen Preis. Die Auslieferung nach Eingehen einer Bestellung erfolgt in ca. 4 Wochen!

Hardy ~~Handbuch~~ ~~Neuer Joystick~~ Strobel

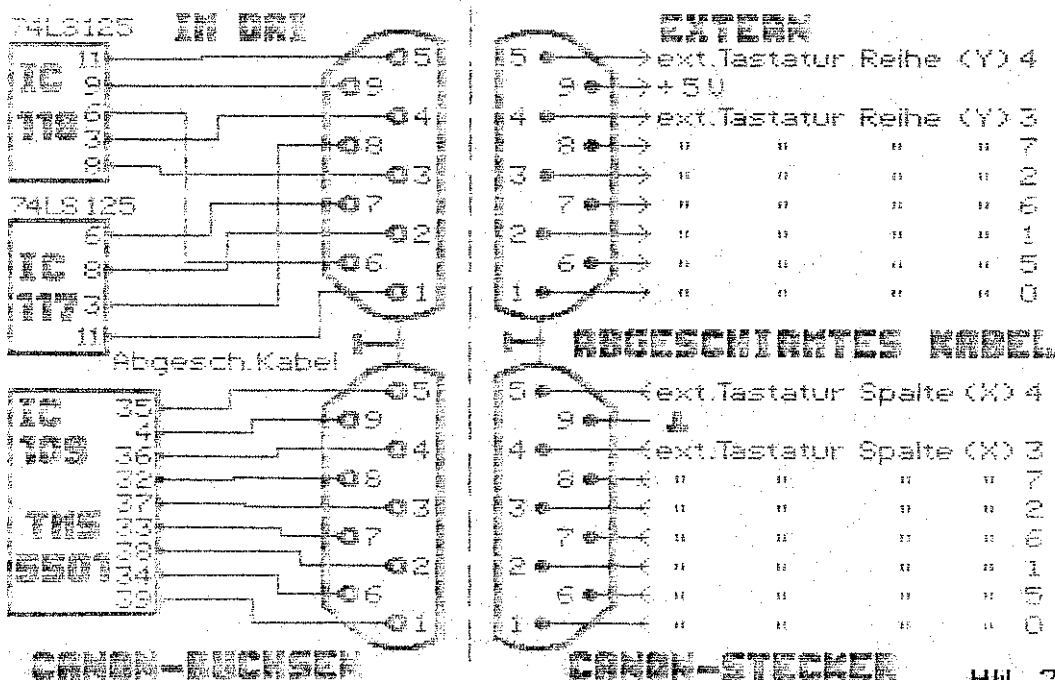
Die Canonbuchse wird nach der Standardbelegung für Joysticks verdrahtet. Es können deshalb Modelle von ATARI 400/800 oder VC 64 und ähnliche benutzt werden. Die Funktionen liegen parallel zu den CURSOR-Tasten bzw. zur SPACE-Taste. Die Kabel von den ICs zur Buchse müssen abgeschirmt sein!



IC Nummern nach HW 3 (großes Blatt) !!

Hardy ~~Handbuch~~ ~~Neuer Joystick~~ Strobel

Die externe Tastatur sollte eine X/Y MATRIX wie die DAI-Tastatur besitzen. Angeschlossen wird sie hier über zwei CANON-BUCHSEN. Diese sind leicht erhältlich, haben ein Metallo Gehäuse, die Anschlüsse sind eindeutig nummeriert und die beiden Verbindungskabel zur Tastatur können zwei 10polige Spiralkabel sein, die sind leichter erhältlich als 20poliges Kabel. Sollten nach dem Aufbau Fehlfunktionen bei der DAI-Tastatur auftreten helfen vielleicht folgende Maßnahmen: Wenn möglich Verbindungskabel kürzen - die Widerstandsmatrix RB 5 (10 KOHM) durch eine Matrix mit 1 KOHM ersetzen oder die IC-Anschlüsse von IC 109 PIN 32-39 über zusätzliche Pull-Down Widerstände von 1,2 KOHM an Masse legen. Bitte gehen Sie nach dem Aufbau in den Editor und drücken folgende Tasten gleichzeitig: CURSOR-SHIFT-REPT. sollte der Editor dadurch verlassen werden, schreiben Sie mir bitte.



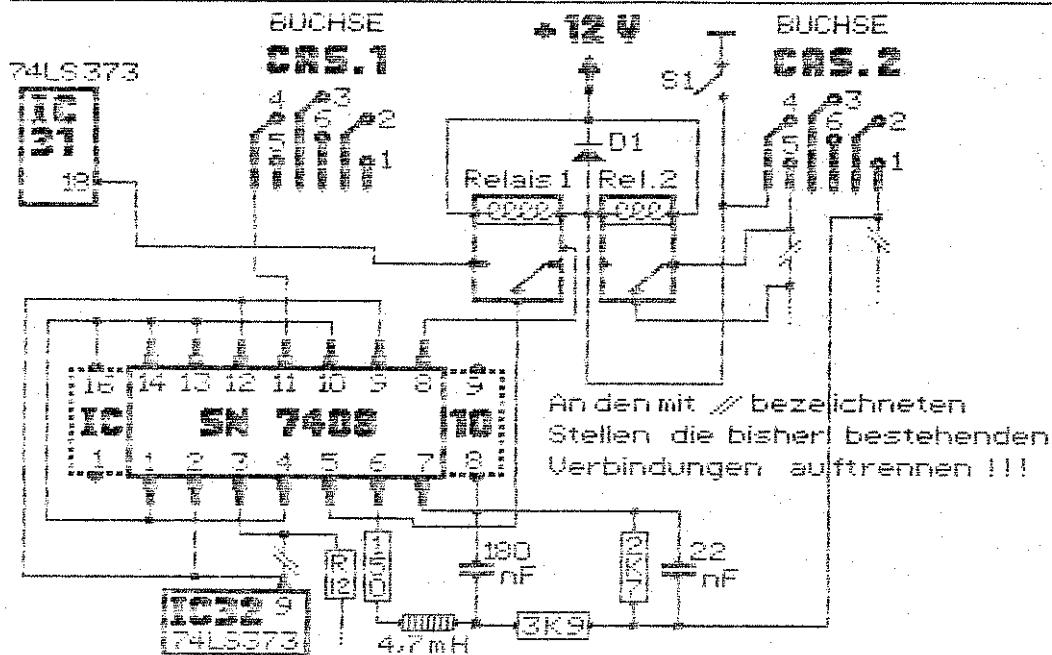
Hardy ~~AUTO-COPY-KASSETTE~~ Strobel

Das Kopieren mit dem Kassettenrecorder verursacht beim DAI häufig Probleme, vor allem beim Erstellen von Sicherheitskopien. Einige Programme lassen sich gar nicht kopieren und BAND zu BAND KOPIEN sind oft BAD. Die folgende Schaltung vermeidet diese Nachteile ! Die Vorteile sind :

1. Programme mit Kopierschutz werden kopiert,
2. es wird mit demselben Pegel aufgenommen (CAS 2) wie beim Original SAVE,
3. keine Impedanzprobleme mehr, wenn gleichzeitig auf zwei Rekorder gesaved wird,
4. schaltbarer Betrieb - NORMAL oder KOPIEREN,
5. sehr preiswert - ca. 10 DM !

Ein Programm welches kopiert werden soll muß natürlich OK sein ! Zwei Rekorder vom selben Typ bringen die besten Ergebnisse.

ANSCHLUSNUMMERN DER KASSETTENBUCHSEN VON INNEN GEGEHEN !!



Das IC SN7408 wird am besten direkt auf dem IC 10 angeordnet, die IC-Anschlußbeinchen des IC SN7408 vorher abbiegen ! Pin 9 von IC 32 sowie die Anschlüsse 1 und 5 von CAS-Buchse 2 von der Platine trennen (durchkneifen), dann die Trennstellen der Schaltung entsprechend neu verbinden ! Am TTL-Ausgang (CAS-Buchse 1 Pin 4) stehen die Impulse von der Speicherstelle #FD06 BIT 0 zur Verfügung (CAS-Output). Belastung maximal zwei TTL-Eingänge. S1 offen (beide Relais in Ruhestellung) : Normaler Betrieb ! S1 geschlossen : Kopieren von CAS1 nach CAS2 !

- Stückliste :
- 1 IC SN7408
 - 1 Diode 1N4148 (D1)
 - 1 Widerstand 150 OHM
 - 1 Widerstand 3,9 KILO-OHM
 - 1 Widerstand 2,7 KILO-OHM
 - 1 Spule 4,7 mH
 - 1 Kondensator 180 nF / 16V
 - 1 Kondensator 22 nF / 16V
 - 1 Schalter (S1)
 - 2 Reed-Relais 1 X UM (12V)

Fragen beantwortet gerne (Rückporto nicht vergessen) :

Hardy Strobel, Neuselsbrunn 51, 8500 Nürnberg 50, 0911/863080

Ich kaufe übrigens auch defekte DAIs !

Die 34 KB SRAM/EPROM-Karte

Autoren: Nils Kay , Wolfgang Schnaack

10.11.84

Wir haben eine SRAM/EPROM-Karte für den DAI entwickelt. Die Karte stellt maximal 34 KB Speicherraum zur Verfügung. Davon sind 2 KB als CMOS-SRAM und bis zu 32 KB EPROM vorgesehen. Ferner ist ein Erweiterungsbus in Slot-Technik vorgesehen. (Mit Decodierung von 13 Slots ! Eine in Arbeit befindliche Buskarte nutzt davon nur 8 .)

Betrieb: DAI mit Reset-Pull-up (IB 8.1) und eingesetzter SRAM/EPROM-Karte => keine Systemabstürze, die nicht auf Software-Fehler zurückführbar waren !!

Vorteile:

- Programme im EPROM und SRAM (= statischer RAM) laufen mit dem vollen 2 MHz Systemtakt (Zeitverlust im normalen Arbeitsspeicher : ca. der Faktor 1.7 (bedingt durch Video-controller und Refresh)) !
- Auf Programme im EPROM kann immer zugegriffen werden (ohne Wartezeiten wie Cassette, MDCR !)
- Der volle Arbeitsspeicher bleibt erhalten !
- Durch die Batterie/Accu-Pufferung gilt obiges auch für den SRAM !
- Bereitstellung eines gepufferten Erweiterungsbus, mit dem Memory-Mapped (!) 13 Zusatzkarten betrieben werden können, ohne den Umweg DCE-Bus.

Einbau:

Die Karte wird auf den X-Bus gesteckt und zusätzlich müssen (leider) noch 5 Leitungen auf die Hauptplatine gelötet werden.:
3 Leitungen an IC55=74LS155:Pin 10,11,12;
Leiterbahn von Pin 12 trennen und an dieser Leiterbahn die 4. Leitung anlöten;
die 5. Leitung wird an Pin 7 des DCE-Buses gelötet. Die Leitungen werden dann noch auf die SRAM/EPROM-Karte gesteckt .

Daten:

8...32 KB EPROM (1-2 * 2764 bzw 27128)
Auf dem Adressbereich #F000-#F7FF:=16*2 KB
mit Bankswitching (belegt Adresse #45)

2 KB CMOS-SRAM (6116 LP-3) mit Batterie / Accu-pufferung. Adressbereich #F900-#FAFF:=4*0.5 KB
(Ruhestromaufnahme < 500 nA !)

Erweiterungsbus zum Anschluss von max. 13 Karten (wie z.B. Uhr, A/D-Wandler, Schnittstellen) vorgesehen. Jeder Karte werden zur Verfügung gestellt (gepuffert !!!) :

- CPU-Datenbus
- CPU-Adresse 0-3 (= 16 Unteradressen)
- Rd (NOT)
- Wr
- Wr (NOT)
- Reset
- Phi2, TTL
- 9 Bit Ausgabeport
- alle 13 CS(NOT)-Signale (d.h.:Der Steckplatz ist völlig unabhängig von der Kartenadresse !)

Für die Chip-Select (CS) Bildung wurde der Bereich
#FB00-#FBFF aufgeschlüsselt: #FBxy

y: Unteradresse

x=0: AMD9511-Zugriff

x=1: Bankswitch-Port :

 EPROM-Banks : Bits 4-7, Bit 3=unbenutzt

 SRAM-Banks : Bits 0-1, Bit 2=Freigabe

x=2: Output-Port (8-Bit D-Flip-Flop :74LS377)

x>2: aktiviert eine der 13 CS(NOT) Leitungen

Einschränkungen:Bei Ken-DOS nicht anwendbar (eigene Karte !)

Bei MDCR ??? (Wir haben selbst keine MDCR !)

Zur Anpassung der MDCR ggf: Mitteilung über Hard-
und Software nötig !

Lieferung:

-Aufgebaute und getestete Karte

(komplett, ohne EPROMs, mit genauer Einbau-
anleitung)

299.-DM

Lieferzeit: 2 bis mehrere Wochen, je nach Stückzahl

Bestellung:

Per Nachnahme:

Nils Kay, Schöne Aussicht 14, 2211 Oldendorf ; oder

Wolfgang Schnaack, Ridderser Weg 54, 2214 Hohenlockstedt

Entwicklung:

Erweiterungskarte mit 8 Slots,

Netzteil: 5V/5A, +-12V/1A

ca: 200.-

Schnelles Cassetteninterface

(4000 Baud auf Normalrecorder !!)

Verwendet den USART 8251, der

als 2. serielle Schnittstelle

genutzt werden kann !

ca: 89.-

Assembler auf EPROM 2764

ca: 90.-

W. Schnaack

Tel.: 04826/2907

nur am Wochenende !

Tips zur Entstörung der Netzleitungen
Autor: Uwe Wienkop

Wohl die meisten Computerbesitzer kennen das lästige Übel, man sitzt am Computer und denkt an nichts Böses, da schaltet ein Familienmitglied eine Neon-Röhre ein, oder der Nachbar benutzt seinen uralten Rasierapparat. Die Reaktion des DAI: Er zeigt einen wunderschönen grünen Bildschirm mit weißer Schrift "DAI PERSONAL COMPUTER" und das eingegebene Programm ist futsch.

Bei mir war es zwar nicht der DAI, sondern meine Diskettenstation war von diesen Störungen derart begeistert, daß sie sich auch sofort angesprochen fühlte, einmal loslief und dann bis auf RESET nicht mehr ansprechbar war. Dies war natürlich frustrierend, zumal ich schon einen sogenannten Entstörfilter in der Steckdose hatte und auch im Netzstecker der Diskettenstation befindet sich ein "Entstörsatz".

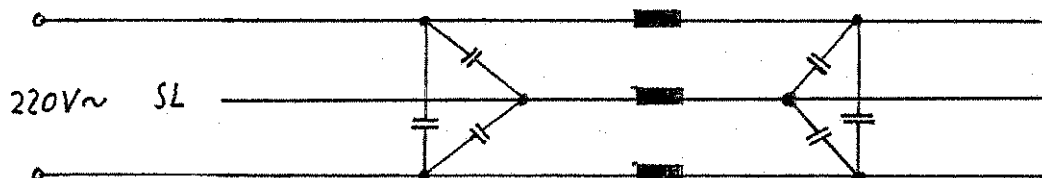
Mein Problem war: Einkommende Signale auf dem Schutzleiter werden weder durch Entstörfilter in der Steckdose noch durch den in der Diskettenstation ausgefiltert. Diese Signale haben freien Durchgang - was sie auch taten. Der DAI benutzt übrigens keinen Schutzleiter (Eurostecker!). Jedoch die Diskettenstation und auch der Drucker enthalten diese Leitung!

In einem Elektronikbuch habe ich untenstehende Schaltung entdeckt und sie dann auch gleich ausprobiert. Damit waren meine Probleme mit Netzstörungen beseitigt!

Als Bauteile wurden bei mir sogenannte Entstörkondensatoren und -Spulen verwendet. Die Kondensatoren haben bei mir eine Kapazität von 8 nF. Hier sollte man ruhig noch einmal im Elektronikgeschäft wegen den Spulen und den Kondensatoren nachfragen.

Alle Bauteile zusammen haben bei mir knapp 20,- DM gekostet - allerdings kam das Gehäuse noch extra. Jedoch war dieser Preis günstiger als der eines Netzfilters und funktionieren tut diese Schaltung außerdem noch!

Ich habe diese Schaltung vor (!) eine Dreifach Steckdose in die Leitung eingesetzt, so daß nun Computer, Diskette und Fernseher an dieser Steckdose hängen und nun ungestört arbeiten können.



Anschluß eines kleinen Verstärkers an den DAI
Autor: Uwe Wienkop

Die meisten DAI Besitzer kennen wohl die Fähigkeiten ihres Computers in Bezug auf Tonausgabe. Jedoch werden die wenigsten schon einmal ein Musikstück oder ein Spiel ungestört, d.h. ohne das lästige Rauschen des Fernsehers, genossen haben. Ich hatte so ein Problem und so habe ich nach einer billigen Möglichkeit gesucht diesem Übel abzuhelpen. Ich habe im Oppermann Bausatz-Angebot einen kleinen Verstärker mit akzeptablen Werten gefunden. Dieser Bausatz kostet z.Z. 11,90 DM ist jedoch auch als Fertigteil für ca. 19 DM erhältlich.

Seine Werte:

Eingangsempfindlichkeit	ca. 40 mV
Frequenzbereich	40 Hz bis 20.000 Hz
Klirrfaktor bei max. Leistung	10 %
" " 70% der Maximal Leistung	weniger als 1%

Die Ausgangsleistung dieses Verstärkers richtet sich nach der angelegten Betriebsspannung. Diese kann zwischen 6 und 16 Volt variieren. Die maximale Leistung beträgt entsprechend 1W bei 6V und 6.5 W bei 16V. Dies hört sich zwar nach sehr wenig an, ist jedoch völlig ausreichend.

Diese Spannung kann man z.B. durch entsprechende Schaltung von Batterien erreichen. Es ist jedoch ebenfalls möglich ein Netzteil zu verwenden. Ich habe hierfür den Bausatz B161 ebenfalls von Oppermann verwendet. Dieses Netzteil liefert 11-18 V bei einem Ampere. Der Preis hierfür beträgt ca. 23 DM inklusive Trafo.

Rechnet man alle Preise zusammen, so ergibt sich folgende

Aufstellung:	B 75	=	11,90	DM
	B 161	=	23,00	DM
Anschlußkabel mit Stecker		=	3,00	DM
Ein-/ Ausschalter		=	2,50	DM
		Σ	40,40	DM

Als Lautsprecher kann ein beliebiger 4 Ohm Lautsprecher z.B. aus einem alten Fernseher oä genommen werden.

Ich besitze die obige Anordnung jetzt seit mehr als einem Jahr und diese billige Anlage hat bei mir stets zur vollsten Zufriedenheit funktioniert.

Zum Aufbau sind noch folgende Anmerkungen zu machen: Der Stecker für den Stereo Ausgang des DAIs ist entsprechend den Angaben des Handbuchs zu löten. Hier kann gleich an dieser Stelle eine Brücke zwischen dem linken und dem rechten Kanal gelötet werden. Somit werden beide Kanäle zusammengelegt. Man hat also "nur" Mono Wiederhabe.

Natürlich kann man auch zwei Verstärker Bausätze für Stereo Ausgabe verwenden.

Sonst sind alle Dinge, die beim Zusammenlöten zu beachten wären, auf den Bausätzen vermerkt!

Fehler an der RS232-Schnittstelle

=====

Bei einigen DAI-Geräten ist am Eingang der RS232-Schnittstelle (Leitung DTR) ein Fehler festgestellt worden. Es handelt sich dabei um den Widerstand R19 (56K), dieser Widerstand kann nur einseitig angeschlossen sein. Nachprüfen kann man das, indem man eine Widerstandsmessung (bei ausgeschaltetem DAI) durchführt.

Zeichnung: a) Widerstand von Punkt 'x' gegen +5V
b) Widerstand von Punkt 'x' gegen 0V (Masse)

Sollte beide Male ein Widerstand von ca. 160K herauskommen, so ist der Widerstand nur einseitig verbunden. Verbinden Sie ihn dann, wie in der Zeichnung angegeben. Zur Sicherheit sollte man über den Widerstand R18 (3K3) noch einen Widerstand (820 Ohm) legen (parallel). Dieser Widerstand ist unbedingt nötig, wenn Sie einen Drucker (oä) an der Schnittstelle angeschlossen haben, welcher keine NEGATIVE Spannung hat.

In den DAI Schaltplänen sind folgende Fehler gefunden worden: Bitte berichtigen Sie Ihre Exemplare entsprechend!

Blatt 5: Die Anschlußbelegung des ROMs MK36000 ist verkehrt
Die korrekte Belegung ist:

A0 pin 8	A8 pin 23	D0 pin 9
A1 pin 7	a9 pin 22	d1 pin 10
a2 pin 6	a10 pin 19	d2 pin 11
a3 pin 5	a11 pin 18	d3 pin 13
a4 pin 4	a12 pin 21	d4 pin 14
a5 pin 3	CS pin 20	d5 pin 15
a6 pin 2	+5V pin 24	d6 pin 16
a7 pin 1	Masse pin 12	d7 pin 17

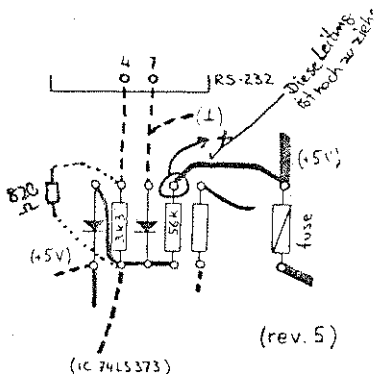
Blatt 7: Die Polarisation der Dioden D43, D44 und D45 muß umgedreht werden

Blatt 10: R101 hat einen Wert von 5.6 K Ohm

evtl. RS232 Fehler (Skizze)

Herman MOEYS
3200 KESSEL-LD (LEUVEN)

Zeichnung aus:
DAInamic Belgium
und ergänzt durch
Hardy Strobel



Wer sich vielleicht für eine RGB-Karte interessiert kann ab Mitte Februar welche bei mir bekommen. Bei dieser Karte handelt es sich um eine Eigenentwicklung mit besseren Signalverhältnissen als bei der Original DAI-RGB Karte.

Mindestbezug: RGB-Platine, gebohrt, mit Lötstopplack und Spezial PROM (Die Farben bleiben gleich!)

Sie können dann den Rest selbst bestücken, wenn Sie wollen ?!
Der Preis für den Mindestbezug beträgt 30 DM und ist deshalb so hoch, da die PROMs noch von einer Firma bestellt werden müssen. Deshalb sind auch Lieferzeiten bis zu 3 Wochen möglich.

RGB-Karte:

- mehr als 1000 Punkte verarbeitungsfähig, da die RGB Treiber nicht aus "laschen" BC Typen bestehen, sondern superschnelle Schalttransistoren sind (Schaltzeit max t_{on} 7 ns / t_{off} 20 ns 2,7 pF Eingangskap.)
- Die Karte wird einfach auf den DAI aufgesteckt, wie die HF-Karte

Sollten Sie eine komplett aufgebaute Karte vorziehen, so benötige ich folgende Daten:

- 1) RGB-Eingangsimpedanz des Monitors
- 2) Sync. signal -> pos. oder neg.
- 3) Falls benötigt: Schaltspannung; frei wählbar zwischen 0-12 V
- 4) Auf Wunsch regelbare RGB-Signale
- 5) Tonsignal-Eingangsimpedanz / fest oder regelbares Tonsignal
- 6) Falls sie eine Buchse wünschen: Art? (Normal; 6-pol.)
- 7) Unbedingt: Die Zusendung Ihrer alten HF-Karte, da ein Bauteil davon benötigt wird (Sie wird wieder zurückgeschickt)
- 8) Spezielle Wünsche werden berücksichtigt - soweit möglich

Der Preis für eine Fertigplatine schwankt (abhängig von Ihren Wünschen): ca. 50 DM

Wenn Sie weitere Informationen wünschen, so schreiben Sie mir:

Hardy Strobel / Neuselsbrunn 51
8500 Nürnberg 50 / ☎ 0911 863080

Falls es noch nicht bekannt ist, ich repariere auch DAI-Computer; allerdings nur nach Rücksprache (Am besten man schreibt mir). Dieses Angebot gilt nur für Clubmitglieder!

Außerdem kaufe ich defekte DAIs auf, denn ich benötige noch ca. 3 Computer! (evtl. Gründung einer DAI Gruppe Unterfranken)

Ich kenne ein Geschäft, das würde den DAI als Geschäftscomputer nehmen, aber nur unter der Voraussetzung, daß eine Festplatte (ca. 20 MB) angeschlossen werden kann.

Frage: Hat jemand so etwas schon einmal ausprobiert?

Frage: Wer hat sich schon einmal Gedanken über die PROMs im DAI gemacht; d.h. Wer kann sagen, was eigentlich im DAI passiert, wenn aus den PROMs die Daten herauskommen?

Verbesserung der DAI-Hardware:

Ich habe bei einigen Geräten festgestellt, daß die NOISE-Fkt. bzw. RND(0) nichts taugt. Der Fehler ist, daß das "Rausch Flip-Flop" nicht besonders gut getaktet wird. Zum Ausprobieren, ob Ihr Hardware Random etwas taugt:

10 ? RND(0):GOTO 10 eingeben

Sollten Sie in der Mehrzahl als Ergebnis 0.0 / 1.0 erhalten, so sollten Sie folgende Verbesserung durchführen:

IC 16, 74LS74 von Pin 3 nach Pin 7 einen Widerstand (10K bis 6.8M0hm (ausprobieren, am besten mit Trimmer) einlöten.

Der Widerstand sollte so gewählt werden, daß Sie viele 0.xxxx Zahlen und auch einige 0.xxxE-2 und 0.xxxxE-3 Zahlen erhalten.

Fehler Bezeichnung: Schaltplan SOUND Rev.5 sheet 7

IC 16, 74LS74, Flip-Flop beim Random Generator, Pin 7 ist falsch, der D-Eingang heißt Pin 2, Pin 7=Masse

MEMORY EXTENSION UNIT (MEU)

Die nachfolgende Bauanleitung ermöglicht es den Speicherplatz des DA1 erheblich zu vergrössern.

Das Prinzip ist dabei denkbar einfach: Bankswitching, d.h. es werden Speicherblöcke ausgeblendet und dafür neue eingeblendet. Als Bankswitchingbereich wurden die Adressen von 0000-BFFF ausgewählt. Das dort befindliche RAM kann in sechs Blöcken zu je acht Kilobyte ausgeblendet werden. Als Ersatz *kann* dann in jedem dieser 8K-Blöcke ein EPROM oder RAM vom Typ 2716, 2732, 2764, 6116 oder 6164 eingesetzt werden. Damit wäre die Erweiterung auf 6*8KB=48KB neuen Speicherplatz beschränkt. Um dies zu vermeiden kann nicht nur eine EPROM-Karte mit max. 48KB angeschlossen werden, sondern insgesamt acht Stück, die Software-mässig umgeschaltet werden können. Somit stehen 8*48KB=384KB neuer Speicher-raum zur Verfügung.

Das Bankswitching und das Cardswitching wird von einem Ein-Ausgabebaustein vom Typ 8255 übernommen. Die Programmierung erfolgt folgendermassen:

1. Auf Adresse F903 wird der 8255 so programmiert, dass Port B und Port CL als Ausgänge arbeiten (z.B.: POKE#F903, #80).

2. Mit einem POKE auf F901 anwählen, welche 8K-Blöcke des RAM's ausgeblendet werden sollen, dabei gilt:

Block-Nr.	Bereich	KennNr.	POKE#F901, 255-KennNr-KennNr-...
1	0000-1FFF	1	Beispiel: Sie wollen die Blöcke 2, 4 und 5 ausblenden. Dann gilt:
2	2000-3FFF	2	
3	4000-5FFF	4	POKE#F901, 255-2-8-16 oder POKE#F901, 229
4	6000-7FFF	8	
5	8000-9FFF	16	
6	A000-BFFF	32	

3. Mit einem POKE auf F902 wird eine EPROM-Karte angewählt, und die Speicher-erweiterung freigegeben. Dies geschieht folgendermassen:

POKE#F902, 8+Kartennummer(0-7)

↑
wenn Bit 3 gesetzt ist, ist die Speichererweiterung freigegeben.

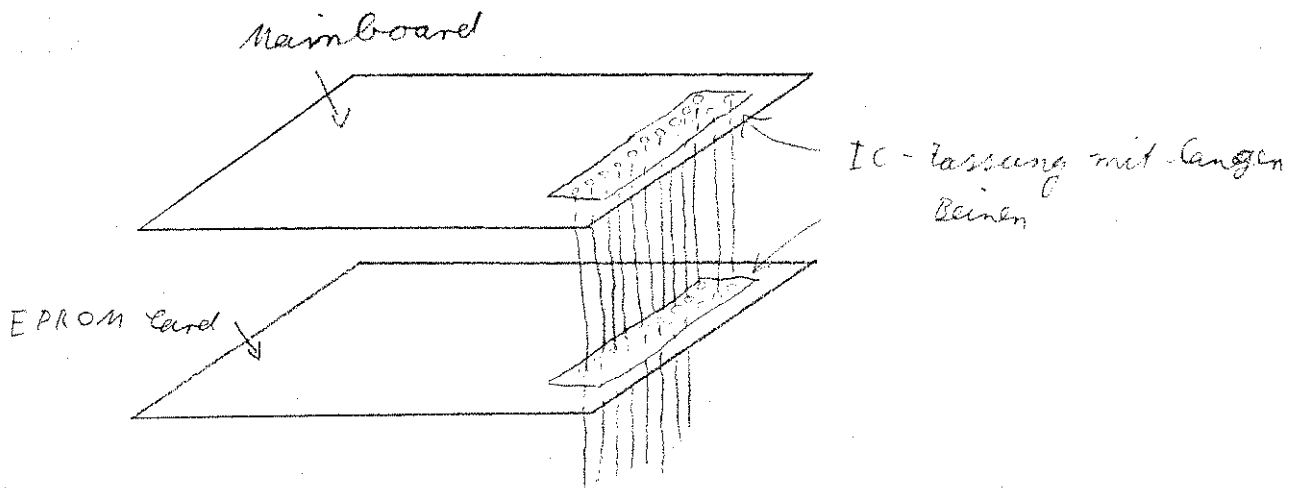
Während die Blöcke im Bereich von 2000-BFFF fast ungehindert auch vom BASIC aus benutzt werden dürfen (Ausnahmen: der Block in dem sich der Cursor befindet und der Block in dem das BASIC-Programm läuft) ist der erste Block (0000-1FFF) fuer das BASIC unbenutzbar, weil dann saemtliche BASIC-Printer und Restart-Routinen ausgeblendet sind. Von ml-Programmen aus ist alles benutzbar, weil hier die Interrupts unwirksam gemacht werden können (DI).

Neben diesen Nachteilen gibt es aber auch mehrere Vorteile:

Wenn man mal von dem riesigen Speichervolumen von 384KB absieht, wird der ml-Programmierer feststellen, dass die Programme in den neuen RAM's oder ROM's ca. doppelt so schnell laufen wie im normalen RAM. Dies liegt daran, dass der Screen-Driver nicht auf die Memory Extension Unit zugreift. Daher bleibt auch der Bildschirminhalt erhalten und sichtbar, obwohl er aus CPU Sicht ausgeblendet ist.

Doch nun zur Hardware:

Das System besteht aus mindestens zwei Platinen (Main-Board und eine EPROM-Karte). Die Verbindung der Platinen erfolgt ueber einen Bus, an den alle weiteren Karten angeschlossen werden. Als Busstecker eignen sich am besten 40-polige Wire-wrap IC-Sockel (IC-Sockel mit ca. 2 cm langen Beinchen), was dann folgendermassen aussieht:

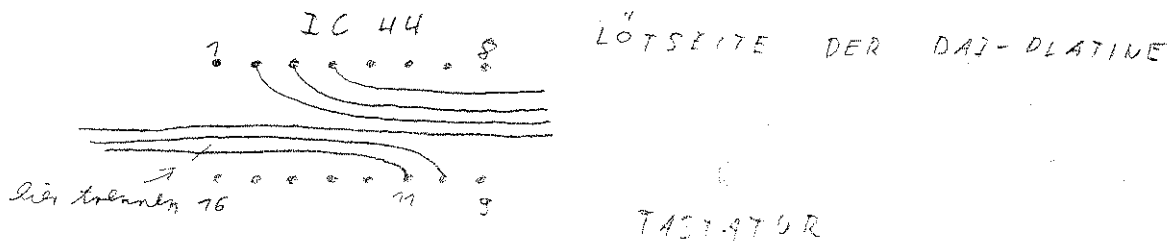


Die Verbindung zwischen DAI und MEU wird mit einem (nicht zu langen ca. 1m) 50-pol Flachbandkabel und zwei Quetschsteckern realisiert. Auf dem Main Board sind zwei 50-pol Pfostenleisten. Diese sind Pin-kompatibel mit dem X-Bus. Es werden zwei benötigt, um noch weitere Erweiterungen, die auf den X-Bus sollen anschließen zu können (z.B. EPROM-Karte fuer MDCR).

Es müssen (leider) noch drei weitere Leitungen vom DAI zur MEU gelegt werden:
 1. Eine RESET-Leitung (z.B. von IC94 (8224) Pin 1);
 2. Die CS(neg)-Leitung fuer den Bereich F900-F9FF von IC45 (74LS155) Pin 10;
 3. Eine RAMOP-Leitung (NICHT identisch mit der vom X-Bus, wegen Änderung siehe unten) von IC44 (74S288 green) Pin 11.

Diese Leitungen werden ueber Steckverbindungen mit den entsprechenden Anschlüssen auf dem MEU-Main-Board verbunden.

Weiterhin muss auf der Platine des DAI eine Leiterbahn unterbrochen werden und zwar die zwischen IC 46 (74S288 blue) Pin 2 und IC 44 (74S288 green) Pin 11. Allerdings muss die Verbindung zwischen IC 46 Pin 2 und dem X-Bus bestehen bleiben. Man trennt also die Leiterbahn am besten direkt vor IC 44 Pin 11 (es fuehrt nur eine Leitung dort hin, und zwar auf der Loetseite; siehe auch Skizze).



Dazu muss die Platine natuerlich ausgebaut werden. Dabei gilt es besonders vor-sichtig zu sein. Dies gilt auch fuer das Loeten. Es sollten nur Elektronik-Loet-kolben mit feiner Spitze und max. 30 Watt Verwendung finden.

Diese drei Kabel und das Flachbandkabel werden seitlich am Rechner herausge-fuehrt. Der Aufbau der Platinen ist bei etwas Loeterfahrung unproblematisch. Es sollten alle IC gesockelt werden (gedrehte Fassungen). Ausserdem ist zu beachten, dass zuerst alle Drahtbruecken eingelotet werden, weil diese z.T. unter IC's verlaufen. Uebrigens sind alle uebrigen Ports des 8255 frei verfuegbar.

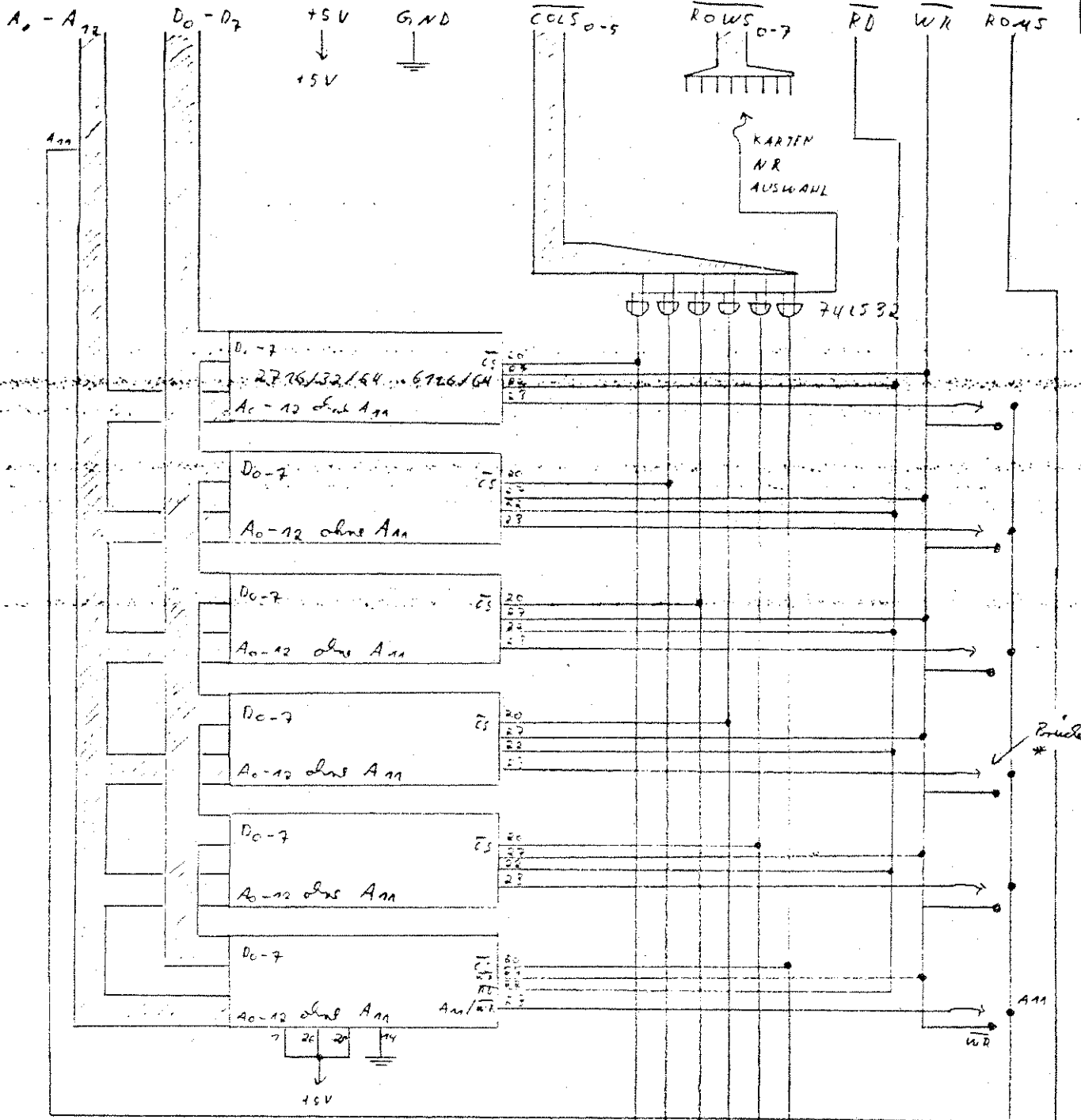
Fuer Fragen und Verbesserungsvorschlaege habe ich jederzeit ein offenes Ohr, und ich koennte mich bereit erklaren die Platinen zu liefern (evtl. auch bestueckelt). Viel Spass beim nachbauen.

Norbert Rinnen

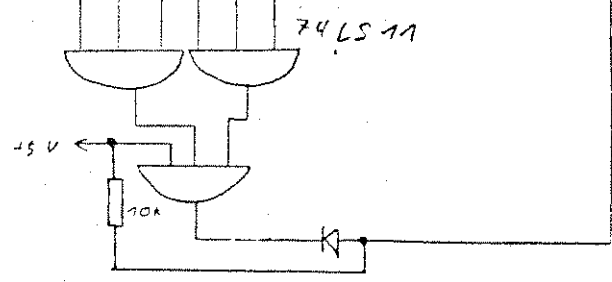
Norbert Rinnen
 Moenikestr. 10
 5060 Bergisch-Gladbach 1
 Tel. 02204/81963

M E U - 8K EPROM / RAM CARD

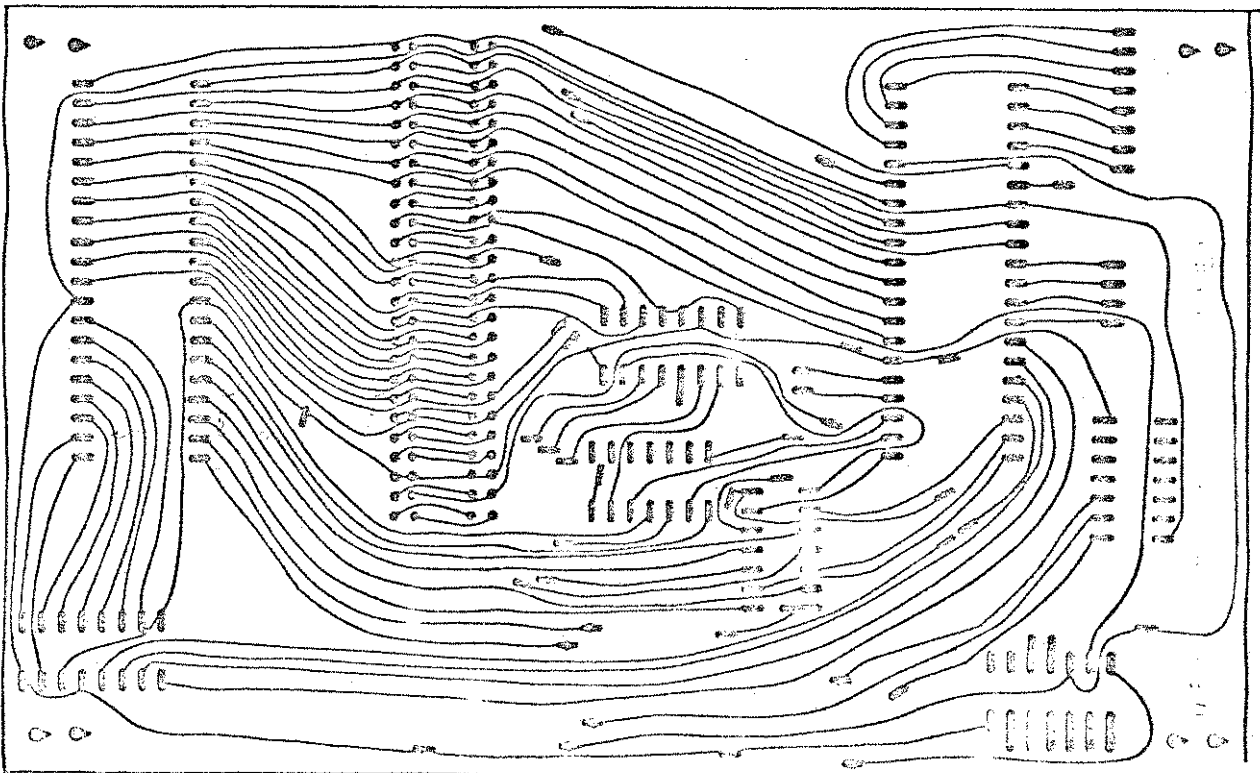
40-POL VERBINDUNGSSTECKER



* Die Tabelle ist bei Verwendung des 6A116 mit WR zu verbinden, sonst mit A11.

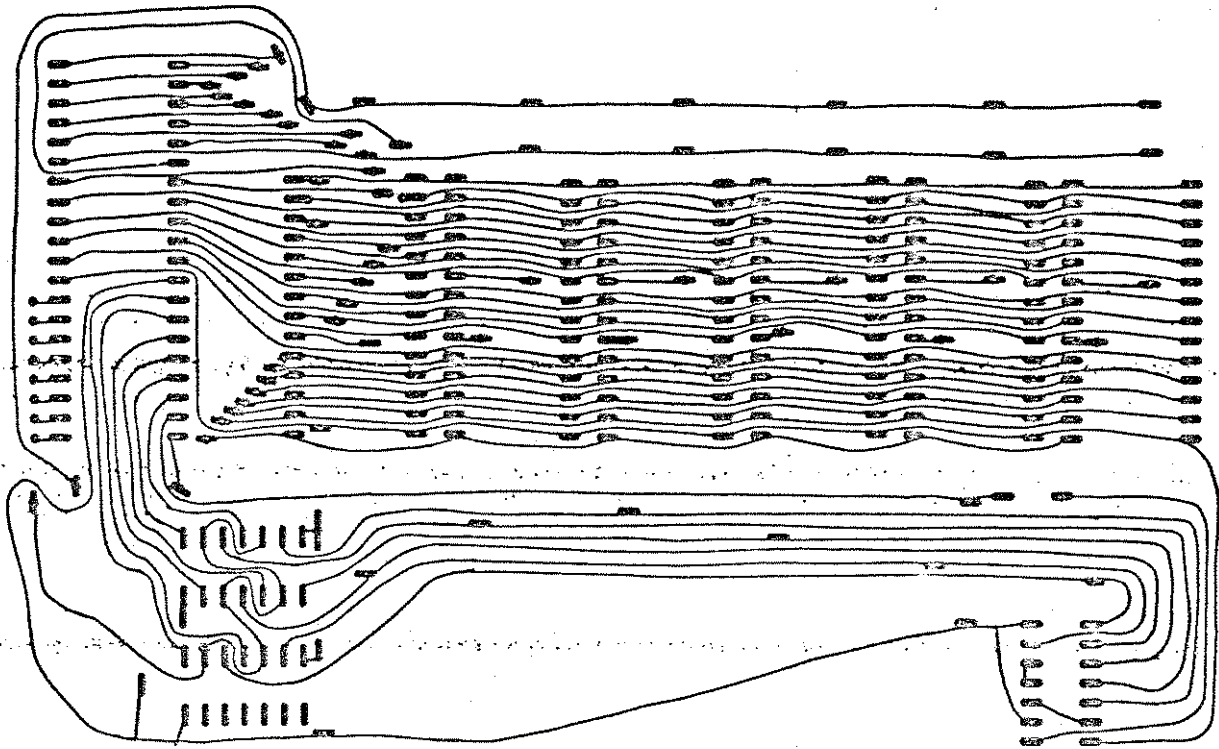


LÖTSEITE



MEMORY EXTENSION UNIT
TO MAIN BOARD

LÖTSEITE



MEU
EPROM CARD 6 x 4 KB

EPROM PROGRAMMER

Mit dem nachfolgend vorstellten EPROM-Programmer ist es auf einfache und preiswerte Weise moeslich EPROMs folgender Typen zu programmieren : 2716, 2732, 2732a, 2764 und 27128.

DIE SOFTWARE

Mit der Software fuer den Programmer sind folgende Operationen moeslich:

1. Leertest:

Bei diesem Test stellt der Rechner fest, ob das EPROM geloescht ist, d.h. ,ob alle Speicherzellen den Wert #FF enthalten. Stoesst er bei diesem Test auf eine ungeloeschte Speicherzelle, so gibt er deren Adresse aus und fuehrt den Test fort.

2. Lesen:

Mit dieser Operation ist es moeslich den Inhalt des EPROMs in den Speicher des Rechners einzulesen. Hierzu verlangt der Rechner die Eingabe einer Startadresse, ab der der Inhalt des EPROMs abgespeichert wird.

3. Programmieren:

Wird diese Funktion angewaehlt, so verlangt der Rechner drei Eingaben: a. Anfangsadresse des zu programmierenden Speicherbereichs; b. Adresse, ab der das EPROM programmiert werden soll. c. Anzahl der zu programmierenden Bytes. Beispiel: Man moechte in ein EPROM vom Typ 2764 folgende Daten programmieren: 3248 Bytes ab Adresse #B350 sollen im EPROM ab Adresse #400 abgespeichert werden. Dazu gibt man zunaechst die Startadresse der zu programmierenden Daten ein (B350), dann die Adresse, ab der das EPROM programmiert werden soll (400) und schliesslich die Anzahl der zu programmierenden Bytes (CB0). Der Rechner teilt dann mit, dass das EPROM programmiert wird, und wann er fertig ist.

Vorher wird ein Loeschtest durchgefuehrt und evtl. darauf hingewiesen, dass das EPROM nicht vollstaendig geloescht ist. Nach dem Programmieren wird ueberprueft ob der Programmiervorgang erfolgreich war.

4. Verifizieren:

Mit dieser Funktion ist es moeslich den Inhalt des EPROMs mit einem Speicherbereich des Rechners zu vergleichen. Die Eingabe der Parameter ist gleich der beim Programmieren. Tritt eine Unstimmigkeit zwischen Rechnerspeicher und EPROM auf, so wird diese Adresse ausseseben. Der Vergleichstest kann durch den Druck einer Taste abgebrochen werden.

Die Eingaben der Parameter erfolgt im UTILITY-Format und muessen durch 'RETURN' abgeschlossen werden. Bei jeder Fehleingabe (z.B. es wurde keine Hex-Zahl eingeseben) wird die Operation abgebrochen, eine Fehlermeldung ausseseben und zurueck zum Menue-Programm verzweigt.

EPROM PROGRAMMER

DIE HARDWARE

Der EPROM-Programmer besteht aus einer Platine, die mit einem RWC-Stecker ausgerüstet ist, (wahlweise kann auch ein 34-poliges Flachbandkabel mit DCE-Bus Stecker angeleitet werden). Ein gleichzeitiger Betrieb des EPROM-Programmers und anderer RWC-Karten ist nicht moeslich.

Die Hardware selbst ist sehr einfach gehalten und daher auch preiswert. Die Adressierung des EPROMs erfolgt ueber Zaehlbau-
steine, der Datenbus ist direkt verbunden, und das Anlesen der
Programmiererspannung erfolgt ueber Transistoren, die in einer
Sicherheitsschaltung so verknuepft sind, dass eine groesst-
moesliche Sicherheit gegen Beschaedigungen des EPROMs erreicht
wurde (selbst bei Systemabstuerzen passiert meistens nichts).
Die Anpassung der Programmiererspannung an die verschiedenen
EPROMs ist ebenfalls ueber Transistoren geregelt und voll
unter Software-Kontrolle.

Auf der Platine ist alles ausser dem Netztrafo untergebracht.
Es muss also nur noch eine Wechselspannung von ca. 25 V
aneeleast werden.

Fuer Interessenten, die die Schaltung nicht in Wire-Wrap-
Technik erstellen wollen, besteht die Moeslichkeit eine
gebohrte Platine (oder auch bestueckt und getestet) bei mir zu
bestellen. Die Preise belaufen sich auf ca. 25,-DM fuer eine
unbestueckte und ca. 100,-DM fuer eine bestueckte Platine.

Die Software ist zum Selbstkostenpreis beim Club erhaeltlich,
bzw. wird beim Bestellen einer Platine mitgeliefert.

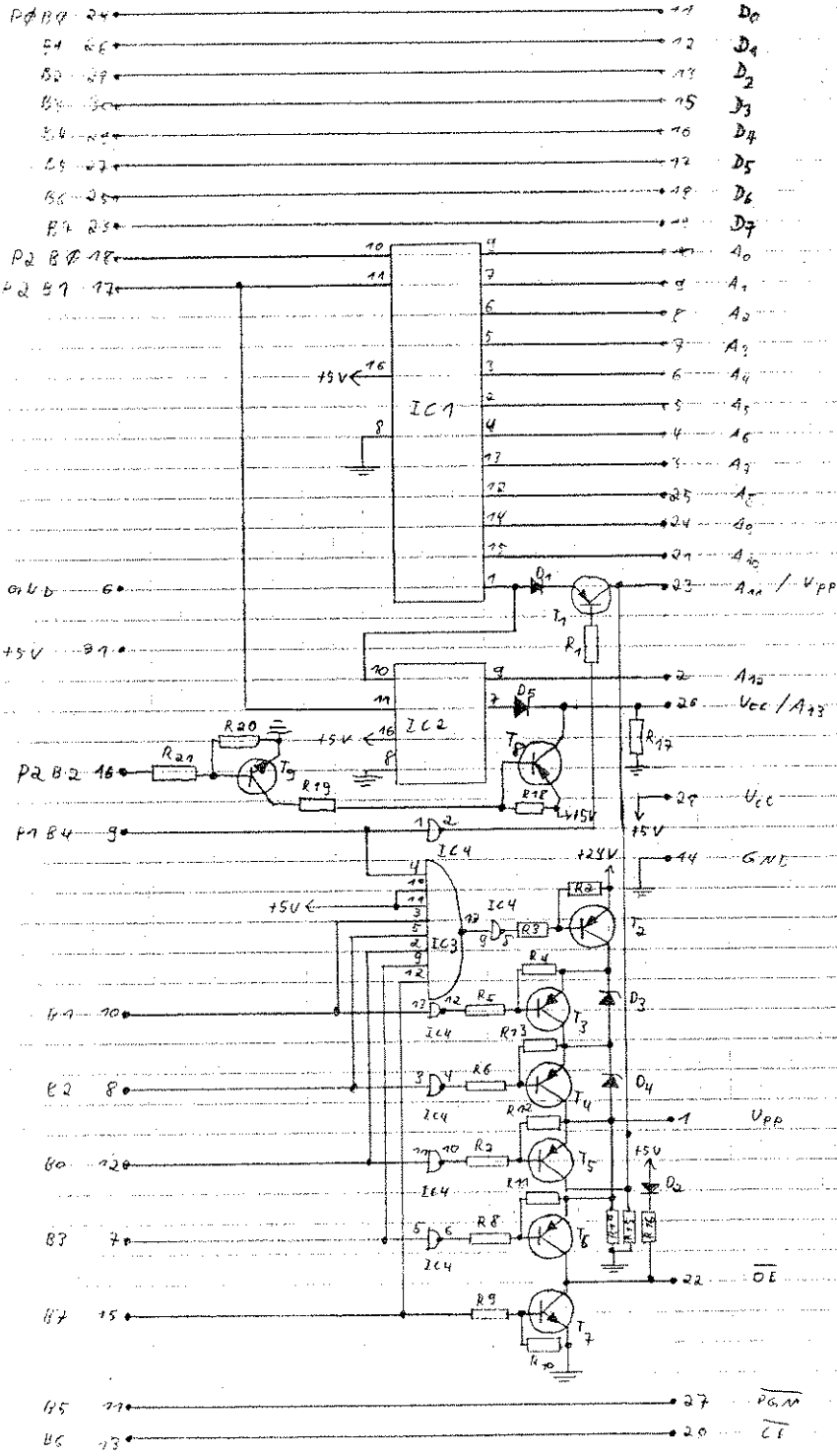
Norbert Rinnen

Norbert Rinnen
Moerikestr. 10
5060 Bergisch-Gladbach 1
Tel. 02204/81963

EPROM PROGRAMMER 2716 / 32 (A) / 64 / 128 -3-

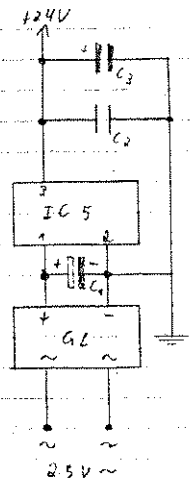
RWC - 0111

28-PIN IC SOCKET



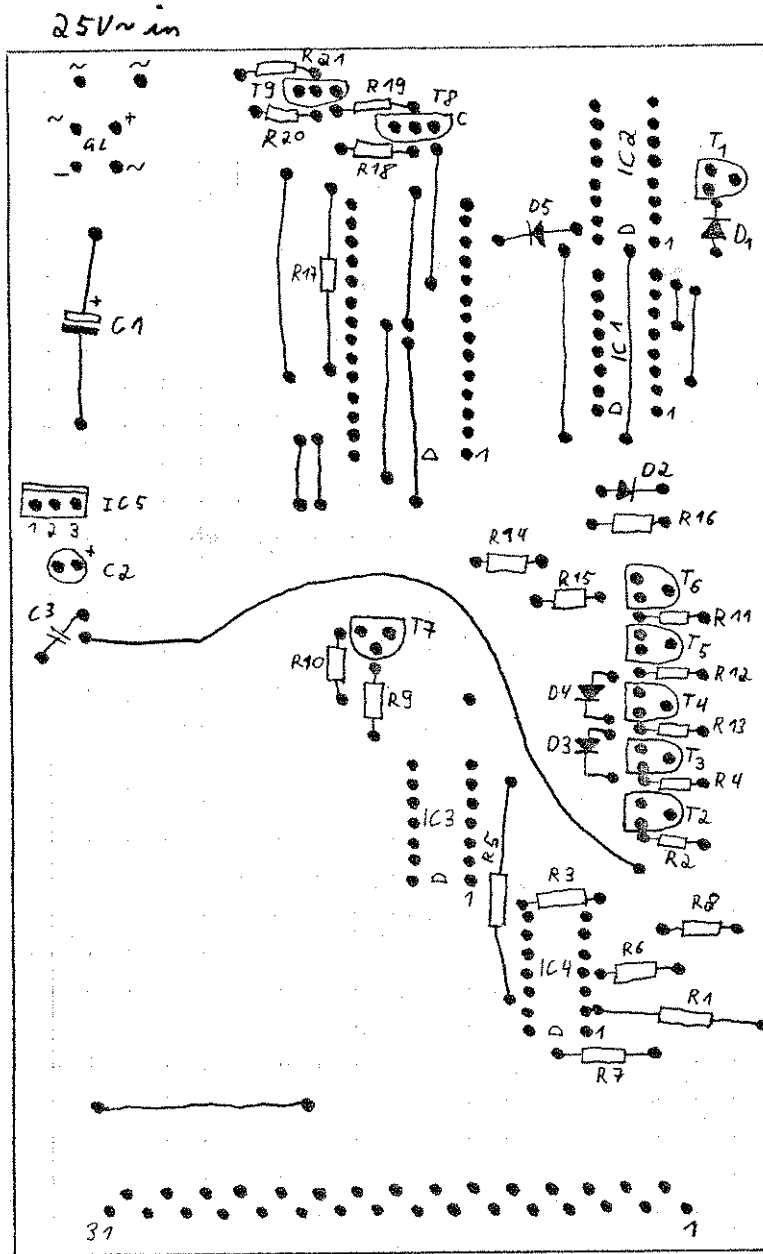
- IC1 CD4040
- IC2 CD4040
- IC3 CD4068
- IC4 7406
- T₁-T₆ BC177
- T₇ BC107
- D₁, D₂ 1N4148
- D₃ 2PD4,3
- D₄ 2PD75
- R₁ 22K
- R₂ 10K
- R₃ 22K
- R₄ 10K
- R₅-R₈ 22K
- R₉ 4,7K
- R₁₀-R₁₃ 10K
- R₁₄ 4,7K
- R₁₅ 10K
- R₁₆ 2,2K
- D₅ 1N4148
- R₁₇ 10K
- T₈ BC177
- R₁₈ 10K
- R₁₉ 1K
- R₂₀ 10K
- T₉ BC107
- R₂₁ 4,7K
- G1 B80C1500
- IC5 7824
- C₁ 2200µF/40V
- C₂ 100µF
- C₃ 4,7µF/35V

HW 32,3

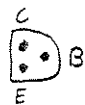


250V ~

BESTÜCKUNGSPLAN (EPROM -PROGRAMMIERER)



Transistoren:



oder



von oben gesehen.

Aufgrund der Tatsache, dass im DAI zunächst nur 2KB fuer Betriebssystemerweiterungen (DAIDOS, TOS, COMDOS, NCU, ...) zur Verfuegung stehen, kam die Idee auf, im Bereich von #F000 bis #F7FF mehrere 2KB-Blocke unterzubringen, und diese mit einer Bankswitchschaltung umzuschalten. Die dabei entstehenden Probleme (z.B. benoetigt man einen nicht umschaltbaren Bereich, von dem aus umgeschaltet wird) wurden folgendermassen geloest:

1. Es existieren 64 Blocke zu je 2KB im Bereich von #F000 bis #F7FF, die mit Hilfe eines 8255 umgeschaltet werden. Diese Blocke koennen EPROMs und/oder RAMs sein.
2. Im Bereich von #F900 bis #FAFF sind vier Blocke zu je 0.5KB RAM untergebracht, die ebenfalls mit dem 8255 umgeschaltet werden.
3. Der Bereich von #FB00 bis #FBFF wurde folgendermassen aufgeteilt:

FB00-FB03 : Mathe Prozessor
 FB04-FB07 : 8255 zum Umschalten der Banks (Port A fuer die 2KB Blocke, Port B fuer die 0.5KB Blocke, Port C fuer die Commodore Floppy)
 FB08-FB0B : 8255 fuer freie Verwendung (z.B. Anschluss von MDCR oder DAI-Floppy, damit der DCE-Bus wieder frei wird)
 FB0C-FB0F : CS (nes.) zur freien Verwendung
 FB10-FB13 : "
 FB14-FB17 : "
 FB18-FB1B : "
 FB1C-FB1F : "
 FB20-FBFF : Hier liegt eine Kopie des Bereiches F920-F9FF (von RAM-Bank0), damit man einen Bereich hat, der nie ausgeblendet ist.

Die Arbeitsweise der Hardware ist aus dem Schaltplan ersichtlich. Zur Umschaltung der Banks muessen die Ports A und B des 8255 als Ausgaenge geschaltet werden. Mit einem POKE#FB04,Banknummer*4 wird die EPROM-Bank (2KB) Nummer 'Banknummer' eingeschaltet. Mit POKE#FB05,Banknummer*64 wird die RAM-Bank (0.5KB) Nummer 'Banknummer' eingeschaltet.

In die EPROM/RAM Fassungen koennen IC's der Typen 2764, 27128 und 6264 verwendet werden.

Die gesamte Elektronik ist bei mir auf zwei Platinen untergebracht, die aneinanderschraubt sind. Diese werden dann einfach auf den X-Bus gesteckt. Weiterhin wird ein IC-Sockel auf ein IC gesteckt (auf IC45, CS F900-FA00) und ein IC-Sockel in die Fassung des Mathe Prozessors. Hier kann auch eine kleine weitere Platine mit einem Wire-Wrap-Sockel aufgesetzt werden, damit der Mathe Prozessor weiterhin verwendet werden kann.

Fuer Interessenten gibt es die Moeglichkeit die Platinen bei mir zu bestellen. Die Kosten belaufen sich auf ca. DM 110,00 fuer eine fertige Platine (ohne IC's aber mit allen IC Sockeln, sedrehte fuer die EPROMs). Welche IC's benoetigt werden ist auf dem Bestueckungsplan angegeben.

Norbert Rinnen
 Moerikestr. 10
 D-5060 Bensisch-Gladbach 1
 Tel. 02204/81963

UNIPROM-1 Hardware-Info :

Um die Hardware aufzubauen, muss man auch nicht tief in die Tasche greifen: alle ICs kosten weniger als DM 30,- :

- 1 x 4021, 4 x 4094
- 1 x 7406, 1 x 555
- 8 x Transistoren
- 9 x Dioden (2 LEDs)
- 4 x Kondensatoren
- 23 x Widerstände

Die Verbindung DAI - UNIPROM-1 wird ueber den DCE-Bus realisiert. Folgende Pinbezeichnungen ergeben sich dabei :

- +5V - Pin 1
- GND - Pin 4
- IN - Pin 26 (P2B0)
- CK - Pin 27 (P2B1)
- S - Pin 28 (P2B2)
- OUT - Pin 20 (P2B4)

Die Platine (siehe DM 19,80) kann beim Autor bestellt werden; ebenfalls alle notwendigen Bauteile.

Der UNIPROM-1 zeichnet sich durch seine Flexibilitaet und seiner Preisueberstaeigkeit aus. Die 50 ms-langen Programmierimpulse und die 5 Schieberegister sind aber Grund eines Nachteils: die Geschwindigkeit: um den 27128 (16K) beispielsweise zu lesen o. ae. werden schon 120 Sekunden benoetigt, das Programmieren verlanat allerdings schon fast 16 Minuten.

Andreas J. Bathe

UNIPROM-1 DRIVER V1.0

Mun gibt es fuer alle DAI-Fans die Software zu dem universellen EPROM-Programmieraeraat UNIPROM-1. Interessierte Leser seien auf den Artikel 'Davor ist kein EPROM sicher' im mc-Heft 8/1984 hingewiesen.

Das Steuerprogramm wurde vollstaendig in Assembler geschrieben und befaest 2,5 KBytes. Um bei langwieriger Datenarbeit im Monitor einen klaren Blick zu behalten, wird der Schriftmodus mit 44 Zeichen pro Zeile erzeugt.

Im DRIVER stehen folgende Unterprogramme zur Verfuegung :

- 1) EPROM-Auswahl
- 2) Loeschtest
- 3) Lesen des EPROM-Inhalts
- 4) MONITOR-Aufruf
- 5) Programmieren
- 6) Ueberpruefen

Folgende EPROM-Typen sind 'direkt' programmierbar ('direkt' deshalb, da praktisch auch andere EPROM-Typen vom Programm aus bedient werden koennen) :

- 2508, 2516, 2532, 2564
 - 2759A(B), 2716, 2732(A), 2764, 27128
- sowie alle C-Typen der 27xx-Serie

Der UNIPROM-1 MONITOR V1.5 verfuegt ueber 5 Befehle :

- 1) C adr1-adrh berechnet die Checksumme im Speicherbereich adr1 bis adr h
 - 2) D adr1-adrh dislayed Bereich adr1 bis adr h und berechnet wiederum von jeder Zeile die entsprechende Checksumme
 - 3) F adr1-adrh xy gleiche Funktion wie in Utility
 - 4) P adr1-adrh xy printed alle Adressen im Bereich adr1 bis adr h mit Inhalt xy
 - 5) S adr gleiche Funktion wie in Utility
- B, Pfeil oben Zurueck zum Driver, Clear-Screen

Das Programm laesst nur moegliche Eingaben zu und ignoriert alle Eingaben, welche zum Absturz fuehren koennen. Datenzugriff ist nur im Bereich 0000h+Offset bis max. 7FFFh+Offset erlaubt, der Offset betraegt 1000h.

Das Programm kann auf Audio-Cassette bezogen werden bei :
Andreas J. Bathe, Helmstrasse 8 in 1000 Berlin 62
Es kostet DM 35,- zuzuealich Versandkosten.


MIDI - Interface für den DAI Strobel Ein Beitrag von Hardy Strobel


Einige kurze Erklärungen zum MIDI-Interface:

Durch dieses MIDI-Interface ist es dem DAI möglich mit einem Synthesizer in Verbindung zu treten. Es können Daten vom Synthesizer empfangen oder Daten zum Synthesizer gesendet werden.
Durch die hohe Übertragungsrate (31.25 kbaud, 8 Bits/2 Stoppbits) ist die Arbeit mit dem DAI nur möglich, wenn das Programm im Stack oder EPROM arbeitet.

Verwendete ICs und deren Funktionen:

- 74LS245: Dieses IC dient als Datenverstärker, Datenfreigabe
- 74LS121: Dieses IC erzeugt den Enable-Puls für den ACIA
- 74LS85: Überprüft die Kartenadr. mit der vorgegebenen Adr. vom Dip-Schalter
- 74LS08: Mit diesem IC werden die $\overline{RD}/\overline{WR}/\overline{BE}$ Signale gebuffert und das Freigabesignal \overline{E} für den LS245 erzeugt
- 74LS04: Die 3 Gatter dieses ICs arbeiten als Oszillator für den ACIA, die Taktfrequenz beträgt 2MHz
- 2 Gatter arbeiten als doppelter Buffer für die Sendedaten (MIDI-Out)
- 1 Gatter dient als Inverter (Verstärker) um einen möglichen Interrupt im DAI zu aktivieren (siehe Steckbrücke)
- 2 Gatter dienen als Buffer für die Unteradr. 0
- 1 Gatter hat die Aufgabe aus dem BE und dem CS des LS85 das CS Signal zu erstellen.
- 1 Gatter ist frei. Die Eingänge sollten an Masse hängen um Schwingungen zu vermeiden.
- 1 ist kein IC, auch wenn er so aussieht; Dieser Optokoppler trennt das Signal vom Synthesizer galvanisch.
- 6850(ACIA): Ist der Hauptbaustein; er erzeugt die Baudrate, prüft auf Fehler... Kurz: Er übernimmt die ganze Sender- und Empfangseinrichtung.
- Suchse A/B: Sind 2x5 pol. D9-Buchsen
- 4fachDp: Mit ihm wird eingestellt mit welcher Kartenadresse die Schaltung arbeitet.
- z.B. D/C geschlossen, A/B offen -> Karten-Adr: #03
D/C/B geschlossen, A offen -> Karten-Adr: #01
- Steckbrücke: Falls Sie mit Interrupt arbeiten (wollen) Steckbrücke auf A;
Ein Interrupt 7 wird im DAI ausgelöst, s.
Cursorblinken
Steckbrücke auf B: ein externer Interrupt wird ausgelöst.

Hinweise zum DCE finden Sie bei:  FM4 (Clubzeitung)

" " externen Interrupt:  HW13

Anmerkung der Redaktion: Es wurden uns von Hardy Strobel freundlicherweise die Unterlagen über den MC6850 und ein Auszug aus der Home-Computer ("MIDI-Software selbst gemacht") zur Verfügung gestellt. Wer hieran Interesse hat kann Kopien (27 Seiten) bei Hans Wientken neben 5 DM (incl. Porto) erhalten!

Nun ein kleines BASIC Beispielprogramm:

MIDI-IN

- 1) alles verbinden, - DCE, MIDI-Interface, MIDI-OUT
- 2) DAI -> ein, Synthesizer -> ein
(Kartenadresse auf z.B. #E: D/C/B offen, A geschlossen)
- 3) 10 OUT #E0,#3: REM ACIA MASTER-RESET
20 OUT #E0,#12: REM ASIA-Teiler 64-8 Bit + 2 Stopp
30 AZ=INP#E1:2HEX\$(AZ):GOTO 30

So, nun hämmern Sie auf dem Synthesizer herum, es müßten nun HEX-Zahlen erscheinen.

MIDI-OUT

- 4) Nun etwas zu senden
10 OUT #E0,#3:OUT#E0,#12
20 OUT #E1,RND(128)+127:GOTO 20

Auf dem Synthesizer müßten nun Töne zu hören sein (z.B. Kopfhörer!)

Also:

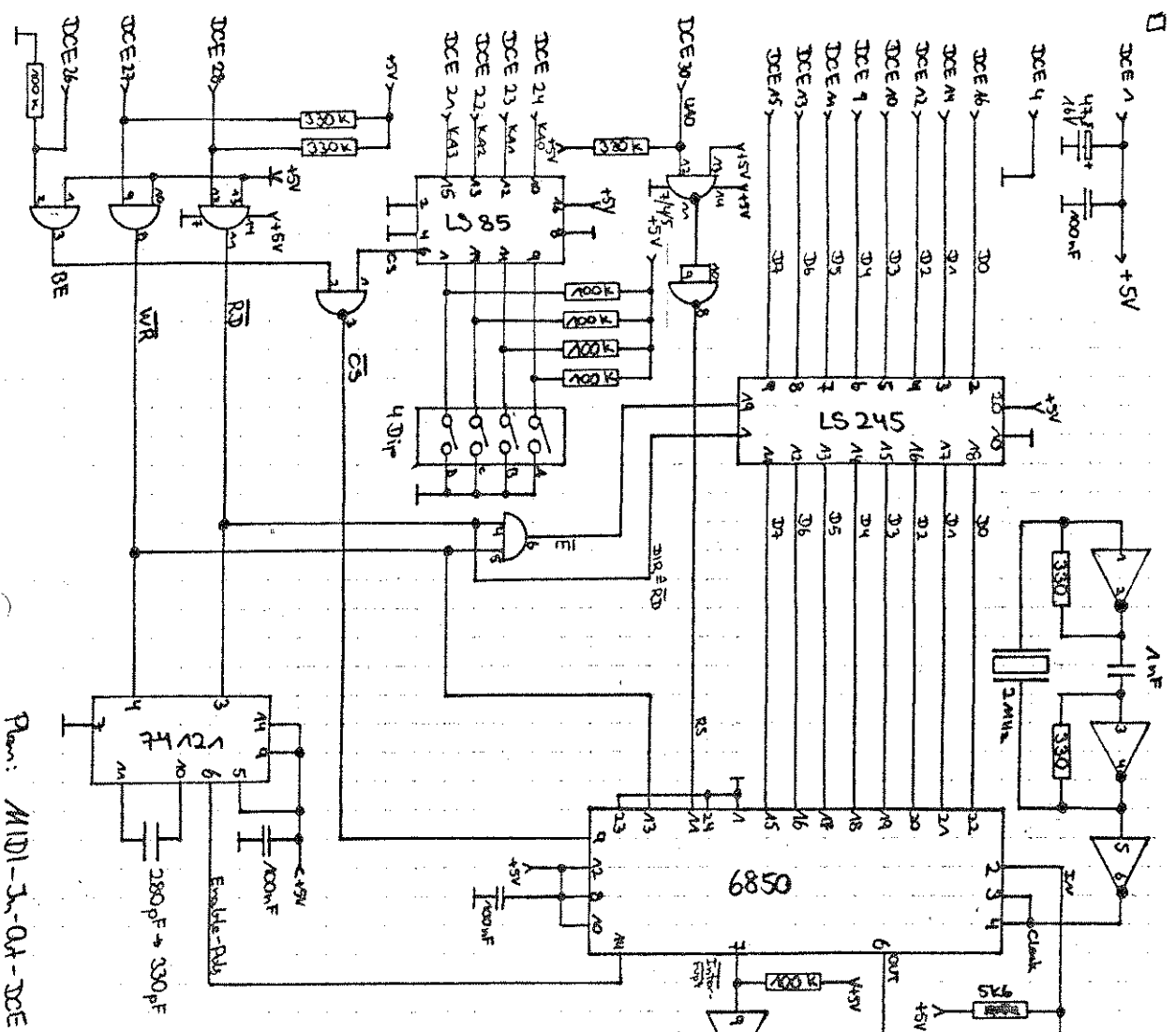
- OUT #x0,AZ -> Sie schreiben in das Controlregister des ACIA
- OUT #x1,AZ -> Sie schreiben Daten in das Senderregister des ACIA
- AZ=INP#x0 -> Sie lesen das Statusregister des ACIA
- AZ=INP#x1 -> Sie lesen das Datenempfangsregister des ACIA
(X = Kartenadresse)

MIDI Kurzkodel:

Einige Codes, die Musikfreunde werden sie hoffentlich wissen.

- #00-#7F : Keyboard, spez. des Synth.
- #80, Data 1/2 : Note OFF, Note, Velocity
- #90, " : Note ON, " "
- #A0, " : Polyph, Pitch, Nr, Value
- #B0, " : Controller, Nr, Value
- #C0, #7A, #00/#7F : Local Keyboard OFF/ON
- #D0, #7B, #00 : All Notes OFF
- #E0, #7C, #00 : OMNI OFF
- #F0, #7D, #00 : OMNI ON
- #80, #7E, #00 : Mono on/Poly off
- #80, #7F, #00 : Poly on/Mono off
- C0, Prog. No : Program change
- D0, Value : Channel pressure
- E0, LSB, MSB : Pitch bend
- F0, Manuf. Id., Data: System Information
- F8 : Timing clock
- FA : Song start
- FB : Song continue
- FC : Song stop
- FE : Active sensing
- FF : System reset

P.S.: Noch etwas für die Nachbauer. Sollten Sie Verbesserungen haben, so schreiben Sie doch an die DAI-Redaktion (Uwe Wienkop) -- (Er freut sich!)

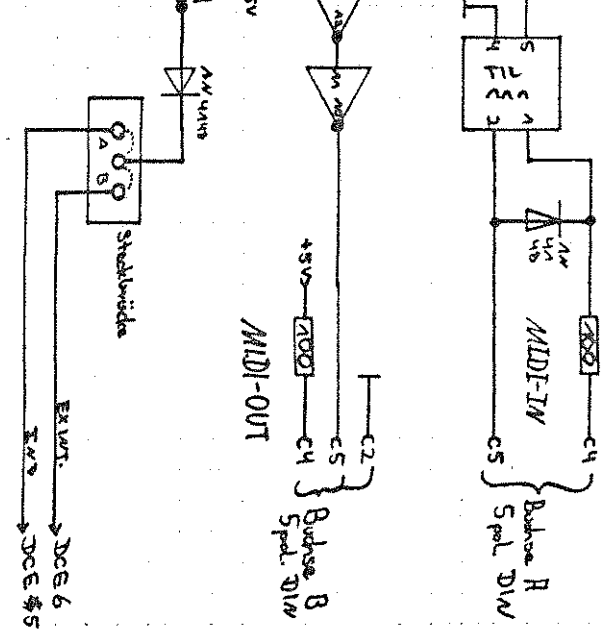


Plan: MIDI-5 pin-DCE

- Benutzteile:**
- 1x 6850 "40A"
 - 1x 74LS245
 - 1x 74LS85
 - 1x 74LS121
 - 1x 74LS00
 - 1x 74LS08
 - 2 x Diode 1N4148
 - 1 x 74LS04
 - 1 x Quarz 2MHz
 - 1 x Optokoppler TIL AM
 - 1 x 47pF/16V
 - 3 x 100nF
 - 1 x 10nF
 - 1 x 220pF → 330pF
 - 3 x 330kΩ
 - 6 x 100kΩ
 - 1 x 200kΩ 5k6
 - 2 x 330Ω
 - 2 x 100Ω
 - 1 x 4-fach Dip Schalter
 - 1 x Steckwädhle 3Pin

Dr. G. G. G.

10. AUG. 1985 13. AUG. 1985
 Hardy Strobel
 Neuseibrunn 51
 8500 Nürnberg 50
 HW 36, 3



Featur der Stragic-Karten:

- gebufferte Daten
- gebuffertes Strobo
- pro Ausgang (Daten) max. 24mA-low
- 4 Rückmeldeleitungen
- DCE-Bus-Anschluß
- BASIC fähig mit OUT, INP, WAIT und natürlich auch ML-Prog.

-
- ### Speechcard SP
- Phonengenerator (64 Laute)
 - 8-10 Bytes/sec
 - Interruptfähig: IN7 / Ex.Int
 - getrennter Interfacereset
 - Frequenzbereich 0 Hz-5 KHz
 - Kling richtig nach Ami-Slang (Richtung TEXAS)
 - DCE - Bus Anschluß
 - BASIC OUT, INP, WAIT und ML
 - Preisgünstig: SP ca. 40DM

-
- ### Soundcard:
- Voll Software kontrolliert
 - 3 unabhängige Tongeneratoren von 30,5 Hz- 125 KHz
 - 1 Rauschgeneratore (3900 Hz-125KHz)
 - 3 Mixer
 - 3 Amplitudenkontrollregister
 - 1 Envelope-Einheit (.12Hz-7800Hz)
 - 2 I/O Port (2x8 Bit) u.v.m
 - Preisgünstig: AV ca. 20 DM
 - DCE - Bus-Anschluß
 - BASIC OUT, INP, WAIT und ML

-
- ### Speechcard SC:
- Phonengenerator (64 Laute)
 - ca. 90 Bytes/sec
 - Interruptfähig: IN7 / Ex.Int
 - Frequenzbereich 0 Hz-5 KHz
 - Frei progr. z.B. Filterfrequenz, Geschwindigkeit usw.
 - Registerauswahl - 5 Register
 - DCE - Bus-Anschluß
 - BASIC OUT, INP, WAIT und ML
 - leider etwas teurer: SC ca. 160 DM, dafür hat er aber Deutsche Umlaute und klingt besser

Erklärungen zu den Schaltplänen:

Centronics - Interface:

- Beuteilerklärung:
- 74LS245 - Treiber, In-Out für die Daten
- 74LS30 - Gatter, dient zur Erzeugung des CS-Signals
- 74LS57 - Treiber, dient zur Bufferung des RD/WR-Signals und zur Statusausgabe auf den Bus

Senden:

- Daten werden gesendet durch: OUT(#F0),xx
- wobei #F0 die feste Kartenadr. ist
- und xx die Daten sind 0-255

Status des Druckers lesen: AZ=INP(#F0)

- wobei #F0 die Kartenadr. ist
- Das Interface hat die feste Adr.: #F0-#FF

Status-Bits:

- 7: BUSY vom Drucker: 0 Drucker kann Daten empfangen
1 Drucker beschäftigt, Datensenden sinnlos
- 6: ERROR vom Drucker: 0 Drucker gibt ERROR Meldung
1 Alles ok.
0 noch Papier vorhanden
1 Kein Papier mehr
- 5: P.EMPTY vom " " 0 Daten angenommen
1 Es können Daten angenommen werden
- 4: ACK vom Drucker: 0 Daten angenommen werden
1 Es können Daten angenommen werden
- 3-0 sind immer auf 1 - auf jeden Fall unwichtig

Einige Drucker haben die Rückmeldesignale nicht, das ist Pech. Aber "BUSY" haben sie auf alle Fälle - unbedingt anschließen!

Zu "ACK": Der Anschluß wird nur interessant, wenn in Maschinensprache programmiert wird, da der Impuls nur ca. für 10 usec auf "Null" geht.

Die Verbindung INIT ist ein durchgeschleifter Hardware Reset.

Stragic Sprachkarte V4.0

Beuteilerklärung:

- SP0254-A1-2 - Sprachgenerator
- 74LS125 - Buffer → RD, WR, Statuslogik, Interruptgeneration
- 74LS138 - Kartenadr.
- L11386 - Kleinverstärker
- Kartenadr: z.B. Brücke 5 am LS138 verbunden → Adr #4
oder Brücke 2 am LS138 verbunden → Adr #1

Reset:

- Resetkaster am DAI → löst beim SP einen Masterreset aus.
- Unter Adr. 0 am DCE löst beim SP einen Interfacereset aus

Ansteuerung:

- 1) Reset DAI (z.B. Kartenadr #4 am LS138)
- 2) OUT#40,0; REM SP macht Interfacereset, alle Filter gelöscht usw.
- 3) OUT#41,0; REM SP fertig zum Betrieb (senden)
- 4) OUT#41,Daten (Status des SP lesen)
- 5) AZ=INP(#41)
- 6) IF AZ IAND #80 # #80 THEN 5)
- 7) NEXT

(5. und 6. können z.B. durch WAIT #41,#80 ersetzt werden)

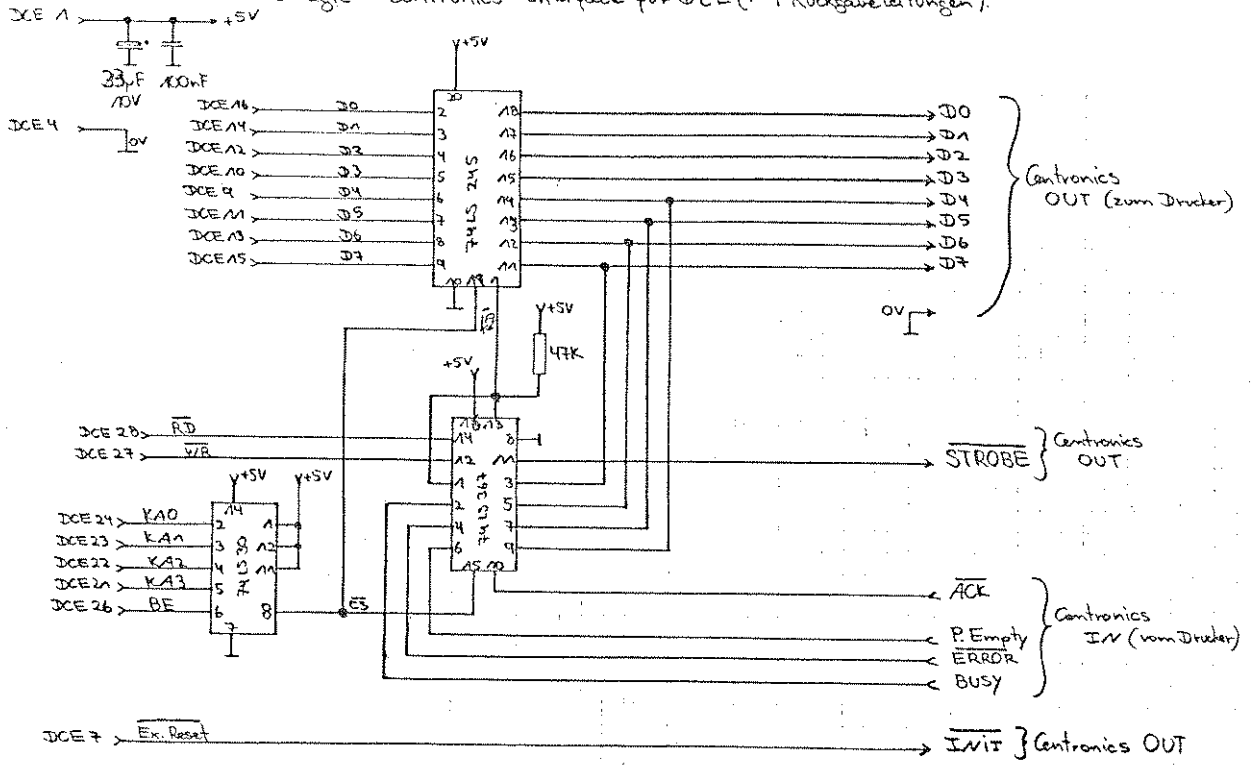
Status des SP

- 7: SBY: "0" SP arbeitet noch an einem Phonem, kein Dateneingang
- "1" SP bereit zum Dateneingang

Wichtig:

Nach dem Sprechen senden: OUT #41,2
Bei Fehler (z.B. Datencrash) OUT #40,0;OUT #41,0
Steckbrücke für Interrupt: Wenn Sie möchten löst der SP nach jedem fertig gesprochenen Phonem einen Interrupt aus, je nach dem wie die Brücke steckt.

Stragic - Centronics-Interface für DCE (+ 4 Rückableitungen).

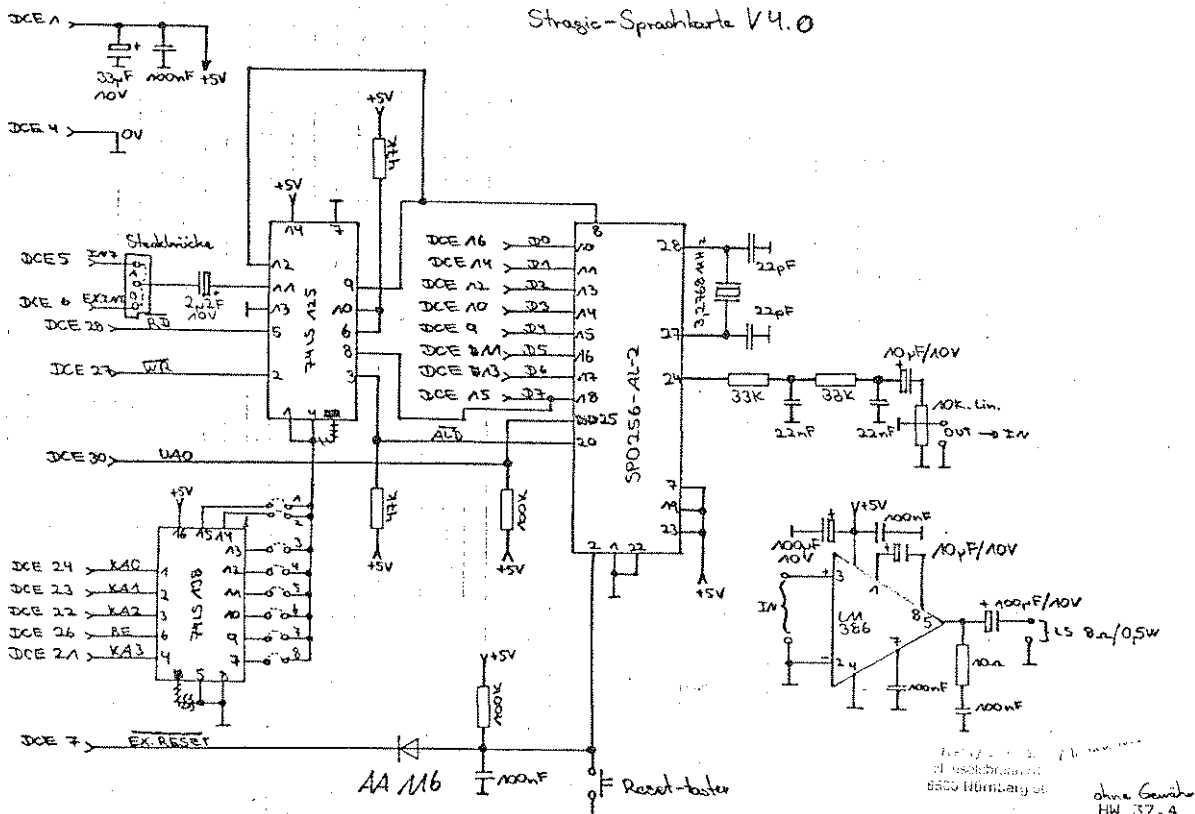


Herdy Strobl
Hausestr. 51
8530 Nürnberg 9

20. 05. 1985

ohne Gewähr HW 37,3

Stragic-Sprachkarte V4.0



Herdy Strobl
Hausestr. 51
8530 Nürnberg 9

ohne Gewähr HW 37,4

HIGH SPEED LOADER

 Autor: H.Rison

Der High Speed Loader (HSL) wurde vor einigen Jahren von Hein Kop und mir entwickelt, um häufig benutzte Programme auf schnelle Weise aus EPROMs zu laden, welche über den DCE Bus angeschlossen waren. Das System wurde um eine EPROM Karte herum aufgebaut, die beim "elektur Printservice" erhältlich war. Die Karte konnte vier EPROMs der Typen 2716 oder 2732 enthalten.

Diese EPROM-Karte wurde auf einer Hauptplatine, welche die restliche Elektronik enthielt, über einen 32 Pin Stecker befestigt.

Das Ergebnis war: Wir konnten ein 10K Programm in einer Sekunde laden. Dies war für Programme wie SPL eine ideale Lösung. Außerdem wurden einige Spiele ins EPROM gebracht und die Kinder freuten sich: Eine EPROM Karte mit dem Spiel wurde eingesteckt und es genügte ein Kommando, um ein Programm zu laden - wie ATARI.

Wie auch immer, die hohen EPROM Kosten damals bewirkten, daß nur wenige Karten hergestellt wurden. Diese liefen aber sehr zufriedenstellend. Ein weiterer Nachteil war, daß die EPROM Karte nur für die Typen 2716 und 2732 aufgelegt war.

Mit der Entwicklung des DAI-DOS 1541 Systems entschieden wir uns die Software des HSLs in der des DOS Systems zu integrieren, da SPL während der Durchführung dieses Projekts häufig benötigt wurde.

Von Zeit zu Zeit werden wir aber immer wieder von Leuten auf den HSL angesprochen, so daß wir nun einen Übergang vom Floppy Interface bauten, wobei gleich mehrere Probleme wie "Power on" Initialisierung gelöst wurden.

Das Endergebnis ist auf den folgenden Seiten in Form eines Schaltplans zu erkennen.

Wir haben jedoch nicht vor eine völlig neue Hauptplatine zum Anschluß an den DCE Bus zu entwickeln, oder aber die Nachfrage müßte so groß sein, daß wir einfach keine Wahl hätten.

Der HSL kann momentan am DCE Bus zusammen mit NCR(s) und VC1541 Disketten Laufwerken angeschlossen werden. Er hat eine eigene Adreßdecodierung und stört nicht den Betrieb von anderen externen Geräten.

Technische Beschreibung

Die Gerätedecodierung wird über P84 mit Hilfe eines 74LS139 vorgenommen, so daß der HSL nur angewählt ist, wenn P84 auf 1 ist.

Nur ein Ausgang des 74LS139 wird verwendet; die anderen können zur Decodierung von weiteren am DCE Bus angeschlossenen Geräten dienen. Der Adreßdecoder für die EPROMs besteht aus zwei 74LS393, getaktet über PC0 und zurückgesetzt durch PS1. Beide Signale werden zweimal invertiert über einen 74LS14. Dies wird gemacht um Fakt- und Reset Signale aufzurufen, welche durch das Verbindungskabel und andere externe Geräte deformiert wurden. Aus dem gleichen Grund wurden auch die zwei 10K Kondensatoren eingebaut. In einigen Fällen ist es noch notwendig einen 500 Ω Widerstand in Reihe mit beiden Leitungen zu legen.

HW 3B.1

Die Datenleitungen werden extra über einen 74LS245 gebuffert, der nur aktiv ist, solange der HSL angesprochen wird. Hiermit werden werden nicht benötigte Daten vom DCE Bus unterdrückt.

Die Auswahl der EPROMs wird durch eine Kombination von 74LS157 und 8255 PROM vorgenommen. Mit dieser Kombination und zusätzlich FR2 und PS3 haben wir die Möglichkeit ein Programm auf mehr als ein EPROM zu verteilen und es später ohne Unterbrechung als Ganzes zu lesen.

Beispiel:
 EPROM 1 (2732) enthält ein Programm "SFDOLER"
 EPROM 2,3,4 enthalten "SPL"

Mit dem Kommando HSL0 wird nun das SFDOLER Programm aus dem 1. EPROM geladen. Mit HSL1 wird dann "SPL" aus den 3 EPROMs geladen. Auf die Anfrage HSL2 wird eine Fehlermeldung "NOT AVAILABLE" ausgegeben.

Dies wird durch fünf Startbytes erreicht:

1. Byte : #FD Nicht verwendetes Befehlsbyte, bezeichnet ein neues Programm

2./3. Byte : Startadresse im DAI in Low/High Reihenfolge

4./5. Byte : Gesamtgröße des Programms in High/Low Form

Es ist jedoch nicht möglich mehr als ein Programm auf einem EPROM zu speichern, so daß evtl. noch leerer Speicher leider nicht verwendet werden kann.

Der Inhalt des verwendeten PROMs (82553) ist wie folgt:

Adresse: 0 1 2 3 4 5 6 7 8 9 A B C D E F
 Daten: E D B 7 D B 7 F B 7 F F 7 F F F F

Die Hauptplatine konnte von "elektur printservice", Postbus 75, 6190 Ab Beek(L) Nummer EPS 82558-2 bezogen werden. Der Verbindungsstecker ist in jedem guten Elektronikgeschäft erhältlich.

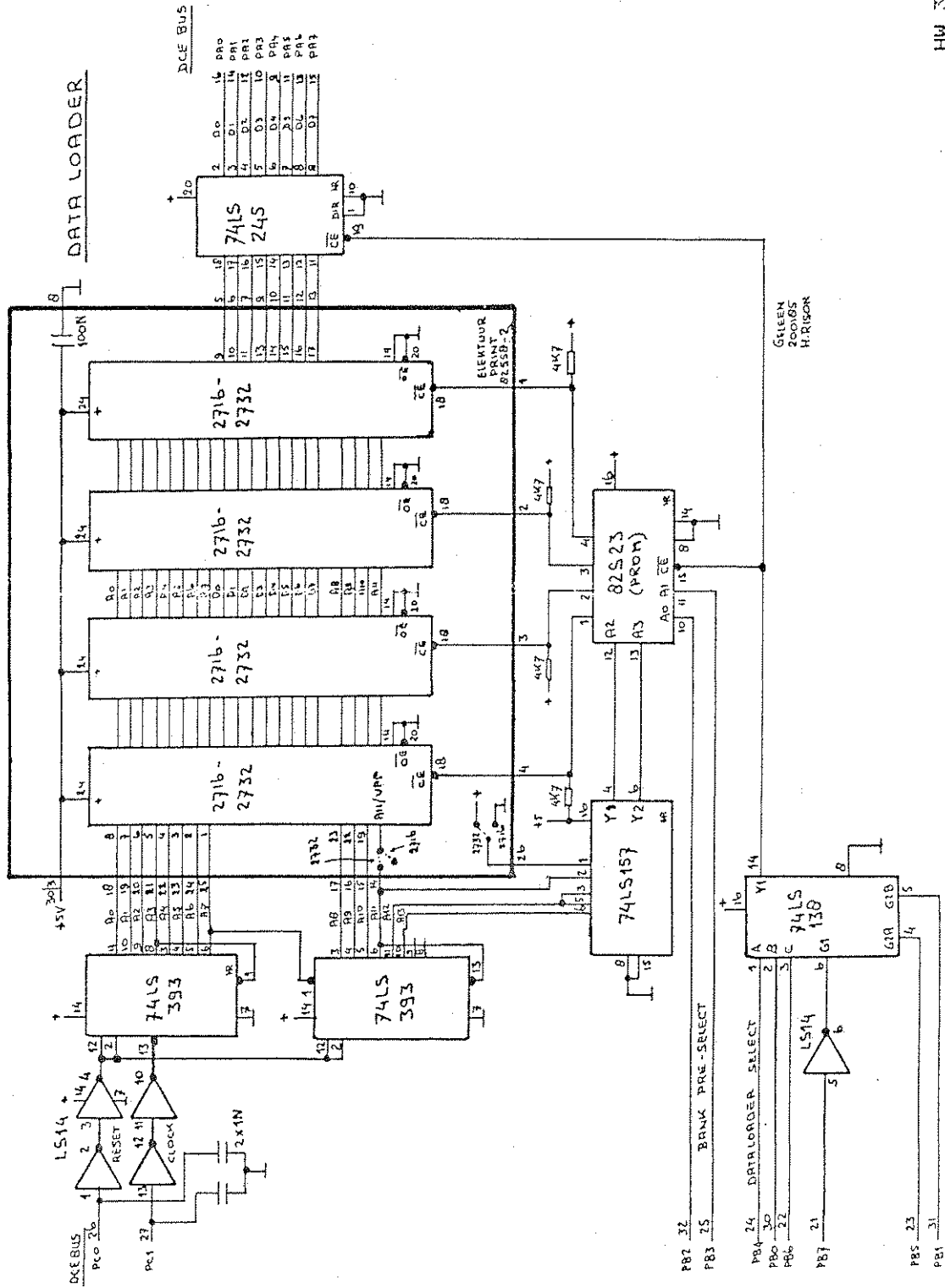
Wenn Sie eine eigene Entwicklung planen, so benötigen Sie diese Karte und den Stecker natürlich nicht.

Sollten Sie Änderungen- oder Verbesserungsvorschläge haben, so lassen Sie uns diese bitte wissen. Wir haben auf dem Papier z.Z. eine Version, welche die EPROM Typen 2716, 2732, 2764 und 27128 verwendet!

Für eine Erweiterung, um mehr als 4 Programme von einer EPROM Karte zu lesen, ist allerdings eine Änderung der DOS Software notwendig, da dies damals noch nicht vorgesehen war.

H.Rison
 Luxemburgstraat 17
 6164 ES Geleen - NL

HW 3B.2

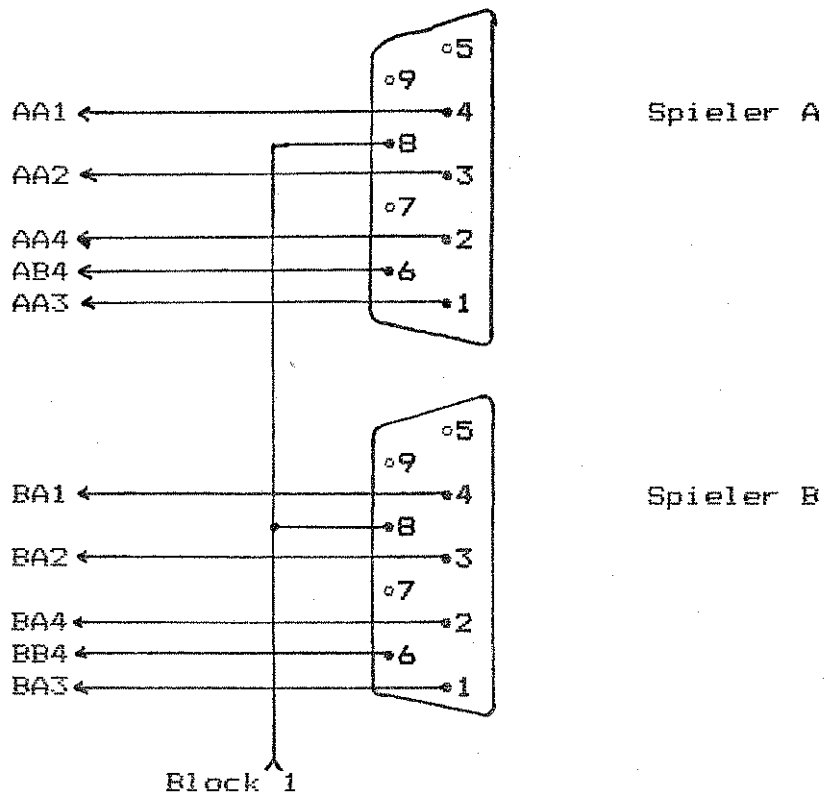


Beispiel: Kartenadr. 5, Spielerpaar 3
A% = INP(#52):PRINT HEX\$(A%)

Ausgabe: 98 (Das Aufgliedern nach der Tabelle ergibt:
Spieler A $\hat{=}$ Unten
Spieler B $\hat{=}$ Fire + Oben

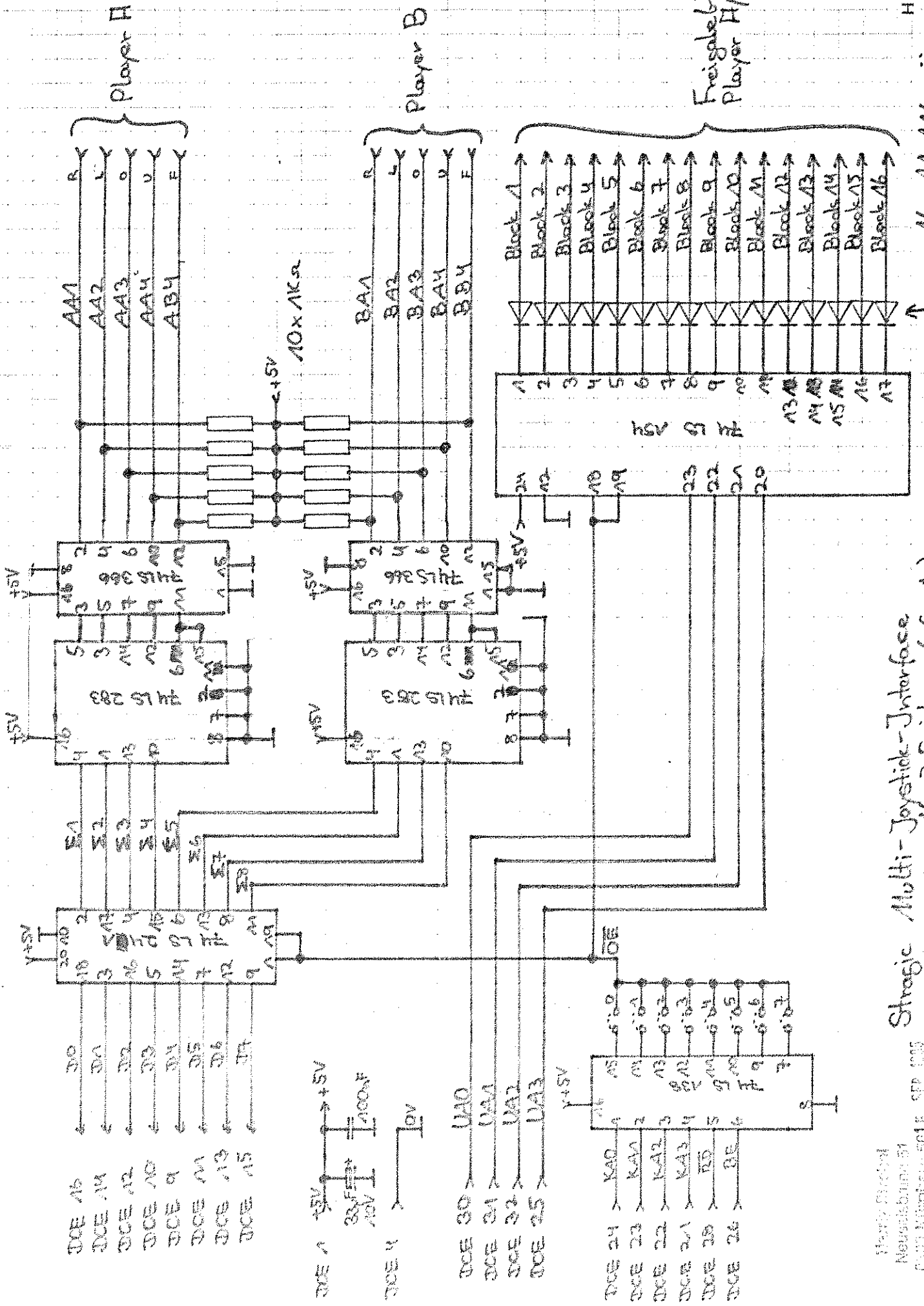
Es sind nur die Codes gültig, wie sie in der Tabelle stehen!
Noch einmal die gültigen Codes: Hex 0,1,2,4,5,6,7,8,9,D

Anschluß eines Joystickpaares an das Multi-Interface:
Max. 16 Paare



Belegung der Canon 9 pol Buchsen für Atari Joysticks uä.:

Pin 1: \uparrow	Pin 6: Fire
Pin 2: \downarrow	Pin 8: Common
Pin 3: \leftarrow	Pin 7,9,5: leer
Pin 4: \rightarrow	



Walter Brunsel
Neuselbrunn 81
08531 Müritzer 0915 077 005

Stragic Multi-Joystick-Interface
max. 16x2 Spieler (a. Ger. v. Stragic)

↑ max. 16x AA M6 o.ä.

Stragic - DAI - DCE -IEEE 488 Interface
(Talker, Listener, Controller)
Autor: Hardy Strobel

Mit diesem Interface ist es dem DAI möglich über den DCE mit IEEE 488 gesteuerten Geräten in Verbindung zu treten.

Das Interface besteht hauptsächlich aus einem GPIB-Interface Controller, der alle anfallenden Timing und Datenübertragungsprozesse vornimmt.

Der GPIB kann als Talker, Listener und Controller arbeiten.

Beschreibung der Bauteile:

74HC04 - Clockgenerator, Inverter (1 Gatter frei)
74HC138 - Kartenadr. 0-7 (Einstellung bekannt)
SN75160 + 3 Busterminderungsbausteine für das
SN75161 3 IEEE 488 Format
μPD7210 - Intelligenter IEEE 488 Interface-Controller

(Der μPD kostet zwischen 30,- und 40,- DM und ist in jedem guten Elektronikgeschäft erhältlich), z.B.

Frank Electronic GmbH
Postfach: 840073
8500 Nürnberg 84

Steuerung des Interfaces:

Das Interface kann mit den BASIC Befehlen OUT, INP und WAIT gesteuert und programmiert werden, z.B.:

OUT (#nm),x n = Kartenadr. 0-7
 m = Register 0-7, d.h. WR Register
 x = Daten zum GPIB
INP (#nm) n = Kartenadr. 0-7
 m = Register 0-7, d.h. RD Register

Es gibt also 8 \overline{RD} Register und 8 \overline{WR} Register. Welche Möglichkeiten der μPD bietet, steht in den Datenblättern (s.u.).

Interrupt: Den Interrupt können Sie wählen, bereits bekannt

Clock: Das IC kann bis zu 8 MHz vertragen, für den DAI genügt aber 2 MHz Quarz, sie können aber auch ein Vielfaches von 2 MHz einsetzen.

Tun Sie Ihrem DAI etwas Gutes, nehmen Sie 74HCxxx ICs. Das Netzteil im DAI freut sich über den geringen Stromverbrauch!

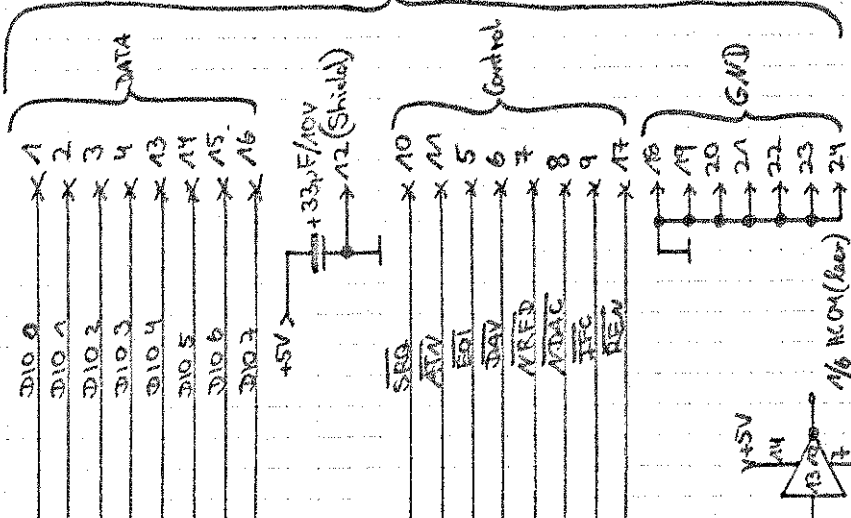
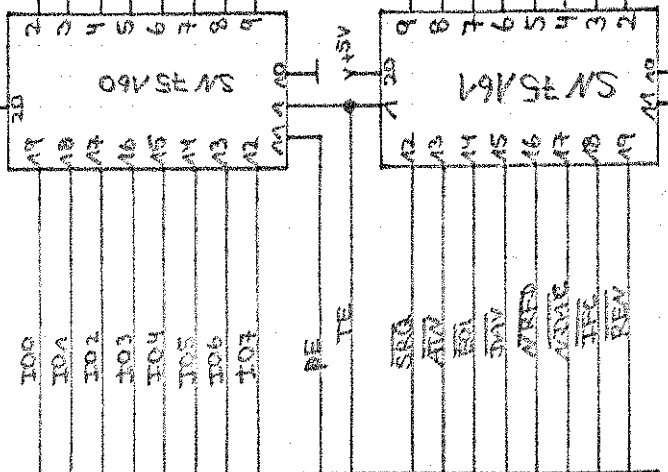
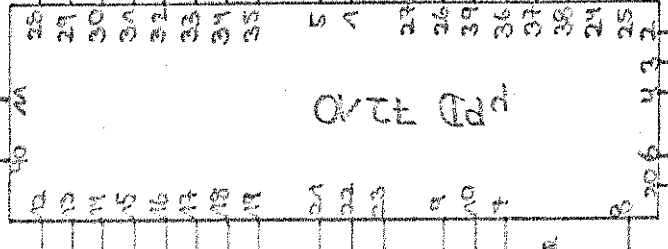
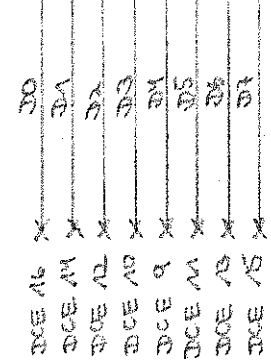
☞ Die Datenblätter sind bei der Redaktion unter der Adresse:

Uwe Wienkop / Laerfeldstr. 54 / 4630 Bochum 1

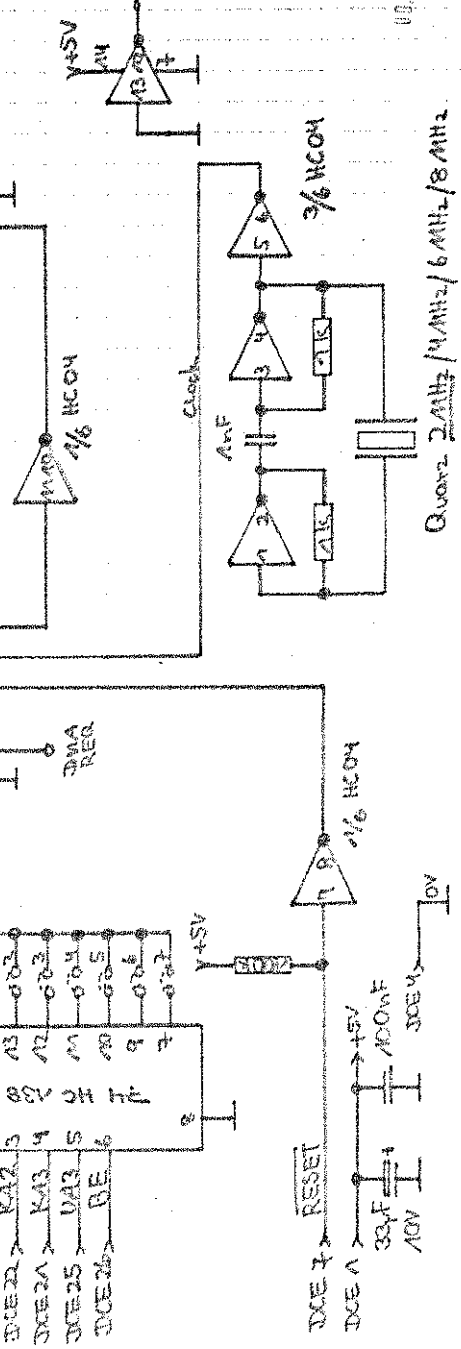
erhältlich. Es wurden uns von Herrn Strobel freundlicherweise die Unterlagen zu den folgenden Chips zur Verfügung gestellt:

MM 58174 Microprocessor Compatible Real Time Clock
DAC1000-1008 10 Bit, μP Compatible,
 Double-Buffered D to A Convertors
ADC0808 8 Bit μP Compatible A/D Converters with 8 Channel
 Analog Multiplexer
ADC0801-0804 8 Bit μP Compatible A/D Converters
INS8048-Series Microcomputer/Microprocessor Family

Diese Unterlagen (* 50 Seiten) können gegen 7 DM (incl. Porto) bei der obenstehenden Adresse bezogen werden.



IEEE 488 - Steckverbinder



Stragic-DAL-DCE-IEEE 488
Interface.
(Talker, Listener, Controller)
O. Gewähr

Hardy Strobel
Neulsebrunn 51
90. 017. 1895
8500 Nürnberg 50
HW 40, 2

Real - Time Uhr
Autor: Hardy Strobel

Diese Real-Time Uhr wird am DCE Bus angeschlossen und bietet Uhrfunktionen, wie: Zehntelsekunden, Sekunden, Minuten, Stunden, Tage, Wochentage, Monate, Jahr und einen Interrupt.

Kurzbeschreibung der Bauteile:

74LS138 - Kartenadresse
74LS245 - Buffer, nur 4 Ein-/ Ausgänge benutzt
MM58174 - Uhrbaustein
Transistor - Interruptinverter

Die Real-Time Uhr kann, wie üblich, durch OUT, INP und WAIT gesteuert werden:

Programmieren mit:

OUT (#nm),x n $\hat{=}$ Kartenadresse 0-7
 m $\hat{=}$ Registeradresse 0-F
 X $\hat{=}$ Daten 0-F (nur die niederen 4 Bits
 sind gültig)

Lesen mit INP (#nm): n $\hat{=}$ Kartenadresse 0-7
 m $\hat{=}$ Registeradresse 0-F
(Auch hier sind nur die niederen 4 Bits gültig)

Interruptsteckbrücke: A) INT 7
 B) Ext. INT

Bei Benutzung des Akkus bleibt der 180 Ω , falls eine Lithiumbatterie verwendet wird, entfällt der Widerstand.

Zugriffsmöglichkeiten:

Register:	:	HEX	:	Zugriffsmode:
Test	:	0	:	Write only
Sekunden 1/10	:	1	:	Read only
1er	:	2	:	Read only
10er	:	3	:	Read only
Minuten 1er	:	4	:	Read or Write
10er	:	5	:	Read or Write
Stunden 1er	:	6	:	Read or Write
10er	:	7	:	Read or Write
Tage 1er	:	8	:	Read or Write
10er	:	9	:	Read or Write
Wochentag	:	A	:	Read or Write
Monat 1er	:	B	:	Read or Write
10er	:	C	:	Read or Write
Jahre	:	D	:	Write only
Stop/Start	:	E	:	Write only
Interrupt	:	F	:	Read or Write

Genauere Einstellungen + Verwendung => siehe Datenblatt; beim Club erhältlich (siehe hierzu die Bemerkungen beim IEEE Interface)

10 Bit (9/8 Bit) D/A Wandler

Autor: Hardy Strobel

Diese Schaltung dient dazu, ein digitales Eingangssignal in ein analoges Ausgangssignal umzuwandeln. Einstellbarer Bereich $-5V \rightarrow +5V$

☞ Diese Schaltung wurde aus Herstellerapplikationen zusammengesetzt, also theoretisch, d.h. sie wurde nicht praktisch getestet. Garantie: 99% Fehlerfreiheit

Erklärung der Bauteile:

74LS04 Inverter
74LS138 Kartenadresse 0-7
4013 2D-Latches für die UA2/3
4052 2x4 Kanalanalogmultiplexer
LF356 OP für Spannungseinstellung
DAC xxxx D/A Wandler

Technische Daten: mit DAC 1000 - 10 Bit
" DAC 1001 - 9 Bit
" DAC 1002 - 8 Bit
 $V_{out} - -5V \rightarrow +5V \text{ max. } 30 \text{ mA}$
4 Kanal Out $-5V \rightarrow +5V \text{ max. } 5 \text{ mA}$
Umsetzzeit: 500 ns.

Um eine möglichst konstante V_D zu erhalten wurde V_D aus der DAI +12V mit einem 78M05 erzeugt. V_{Ref} ist im Bereich von $-5V \rightarrow +V_D$ einstellbar!?

Das Interface reagiert nur auf den BASIC Befehl OUT. Die Befehle INP, WAIT haben keine Funktion.

Die Steckbrücke "Konstant" dient dazu, um festzulegen, von welchem Bit gezählt wird (D/A) von links (MSB)-A, von rechts (LSB)-B. Sie sollte auf B stecken (zumindest ist das üblich)

Die Steckbrücke Kanalwahl: A $\hat{=}$ Ausgangsspannung geht auf den 4fach Multiplexer
B $\hat{=}$ Ausgangsspannung geht direkt zu V_{out} ; Der Kanal ist egal

EX.RESET bewirkt nur, daß der Kanal 0 gewählt wird.

Hinweis: einige Widerstände sind Met.film 1%, die Trimmer sind Keramik (linear)

Die Einstellung der Regler V_{off} (am OP) und V_{Ref} am DAC sollten sie vom Datenblatt selbst lesen und auf Ihren Bedarf einstellen.

$V_{off} \hat{=}$ 0V für den OP-Offset

$V_{Ref} \hat{=}$ Ausgangsspannungsbereich V_{out}

Ein Zyklus, um eine D/A Wandlung zu vollziehen sieht so aus:

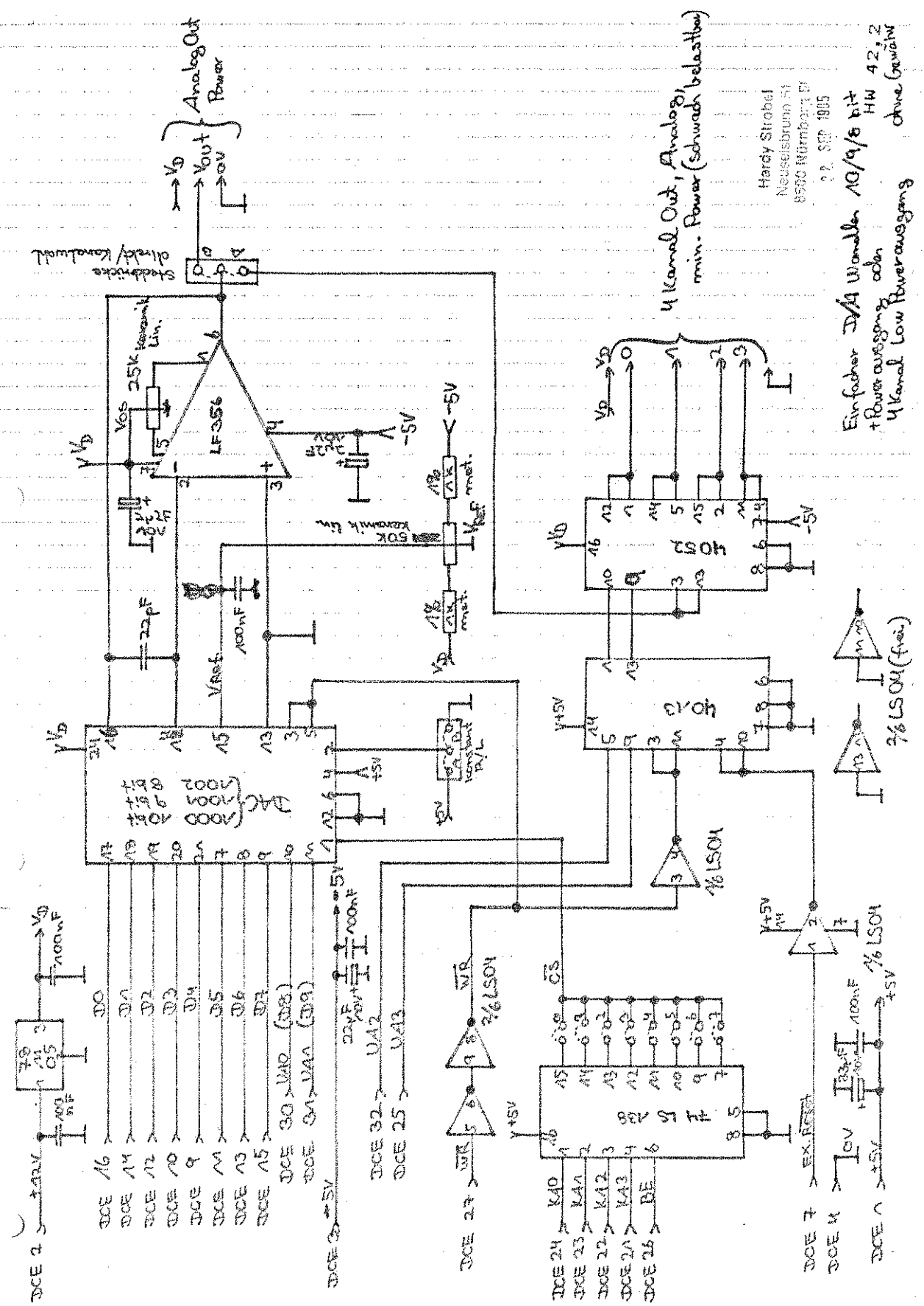
OUT(#nmo),x
x $\hat{=}$ 8 Bit Daten
m $\hat{=}$ 2 Bit Kanalwahl 0..3
o $\hat{=}$ 2 Bit Daten für 10 u. 9 Bit
n $\hat{=}$ Kartenadresse 0..7, Bit 7=0!

n	m	o	x
7 6 5 4	3 2 1 0	7 6 5 4 3 2 1 0	Bits

o und x werden zu einem 10 Bit langen Datenwort zusammengefaßt, also ox, um eine 10 oder 9 Bit Wandlung zu ermöglichen.

Beispiele:

- * Sie verwenden einen DAC 1002, Kartenadr. 5, Volle Ausgangsspannung, Kanal 1: OUT(#57),#FF
- * Sie verwenden einen DAC 1001, Adr. 4, 0V-Ausgang, Kanal 3: OUT(#4C),0
- * Sie verwenden einen DAC 1000, Adr.0 volle V_{out} , kein Kanal gesteckt: OUT(#03),#FF (die 3 ist eigentlich egal)



Städtrichter
 direkt Kanalwahl

4 Kanal Out, Analog,
 min. Power (Schwach belastbar)

Harvey Strobel
 Neuseelsbrunn 51
 8580 Nürnberg, FR
 22. SEP 1985

Ein einfacher D/A Wandler 10/9/8 bit
 + Powerausgang oder
 4 Kanal Low Powerausgang ohne Gewinn

74LS04 (free)

ASCII parallele Tastaturen

Autor: Hardy Strobel

Mit diesem Interface ist es möglich beliebige 7 Bit ASCII Tastaturen am DAI zu betreiben.

Erklärung der Bauteile:

- 74LS138 - Dekoder, nur beim Lesezyklus aktiv
- 74LS224 - 16x7 Bit FIFO Speicher, Lese-Ladekontrolle
async. Betrieb, min 0 Hz, max. 10 MHz
- 74LS368 - geschaltete invertierende Buffer

Der FIFO Speicher hat eine Kapazität von 16x7 Bit (asynchron), d.h. Sie können z.B. mit 10 Hz Daten eingeben und mit 10 MHz Daten auslesen. Die eingelesenen Daten werden solange gespeichert, bis sie ausgelesen werden, oder ein EX.RESET oder ein prog. CLEAR erfolgt.

EIN/AUSgänge:

- Die +5V Spannung wurde aus Sicherheitsgründen aus der +12V Spannung des DAI durch einen 7805 gewonnen.
- Kartenadresse: OE wird nur dann "0", wenn die Kartenadresse, der BE und ein RD-Zugriff erfolgt. Adr.<8, d.h. aus dem Interface kann nur gelesen werden.
- EX.RESET, Beim Betätigen des RESET-Tasters des DAI löscht sich der 16x7 Bit Speicher

Das Interface kann durch die BASIC Befehle INP und WAIT gesteuert werden; OUT hat keine Funktion!

Die Daten werden vom Port A gelesen und dann wie folgt aufgelöst:

```
+---+---+---+---+---+---+---+---+
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+---+---+---+---+---+---+---+---+
```

Die Bits 0-6 enthalten den ASCII Code der gedrückten Taste (#00-#7F)

Bit 7 ist ein Status Bit:

- 0 - Daten gültig
- 1 - Daten ungültig, Buffer leer

Beispiel: AZ=INP(#50): REM KartenAdr. 5; UnterAdr. 0
AZ > #7F Daten ungültig
AZ ≤ #7F Daten gültig

- Mit der UnterAdr. 0 kann ein Indikatorimpuls an die Tastatur abgegeben werden. Die meisten Tastaturen (µP) arbeiten dann ein Sonderprogramm ab. (Nicht alle Tastaturen haben einen DSR Eingang)

UAD = "0" → $\overline{DSR} \hat{=} "1"$ es passiert nichts
UAD = "1" → $\overline{DSR} \hat{=} "0"$ Sonderprog. wird abgearbeitet

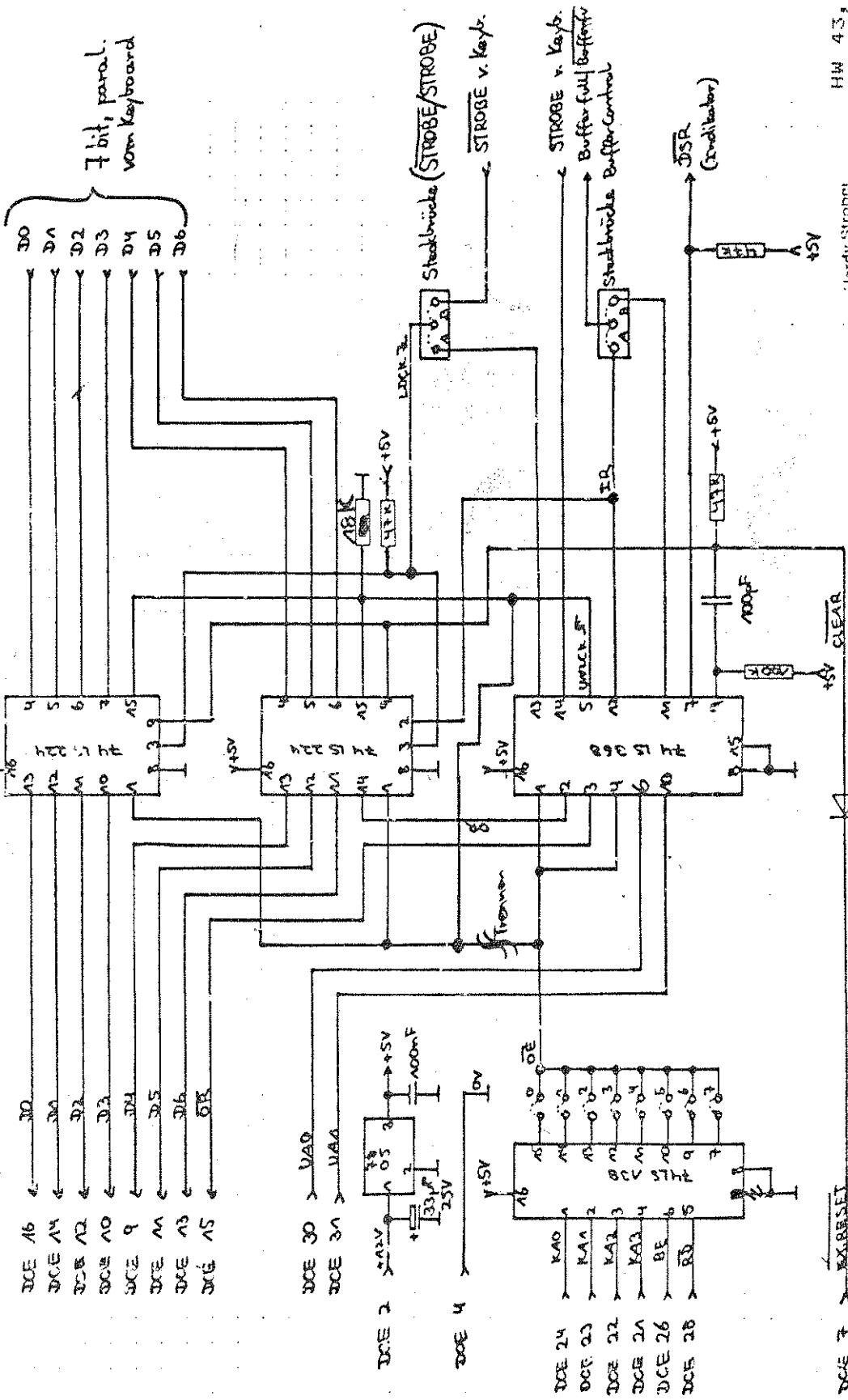
Beispiel: AZ=INP(#51) → \overline{DSR} aktiv

- Mit der UnterAdr.1 kann ein prog.barer CLEAR erfolgen (RESET)
 - UA1 $\hat{=} "0"$ → kein Clear
 - UA1 $\hat{=} "1"$ → Clearimpuls, gelesene Daten sind ungültig
- STROBE-Control-Steckbrücke
 - Steckbrücke auf A: Interface übernimmt den Tastaturcode mit einem positiven Strobesignal
 - Steckbrücke auf B: Interface übernimmt den Tastaturcode mit einem negativen Strobesignal
- Buffer full Control:
 - Steckbrücke auf A: Bufferfull = "0" $\hat{=} Full$, wenn versucht wird von der Tastatur Daten einzulesen
 - Steckbrücke auf B: Bufferfull = "1" $\hat{=} Full$, wenn versucht wird von der Tastatur Daten einzulesen

Das Interface hat also insgesamt 4 Lesezustände:
Kartenadresse = 5 (Brücke 5 am LS 138 geschlossen)

- 1) AZ=INP(#50): Normales Lesen, Daten <#80 sind gültig
- 2) AZ=INP(#51): Lesen normal, aber Aktivierung des \overline{DSR} ;
Daten <#80 sind gültig
- 3) AZ=INP(#52): CLEAR, Daten ungültig
- 4) AZ=INP(#53): CLEAR, Daten ungültig, aber \overline{DSR} aktiv

SRAGIC 7-bit, parallele Speicherinterface für ASCII-paralele Textdaten.



7 bit, paral. vom Keyboard

Überschreibetaster
Interface 22.10.85

AA 116 o.ä.

Einfacher 8 Bit - 8 Kanal A/D Wandler

Autor: Hardy Strobel

Technische Daten: Bereich 0 - +5V regelbar
Impedanz ca. 4,5 K Ω
Umsetzzeit < 100 μ s.

Kurzbeschreibung: 74HC138 - Kartenadresse
74LS125 - Clockgenerator, Buffer,
RD-Generierung
ADC0808/0809 - A/D-8 Bit / 8 Kanal

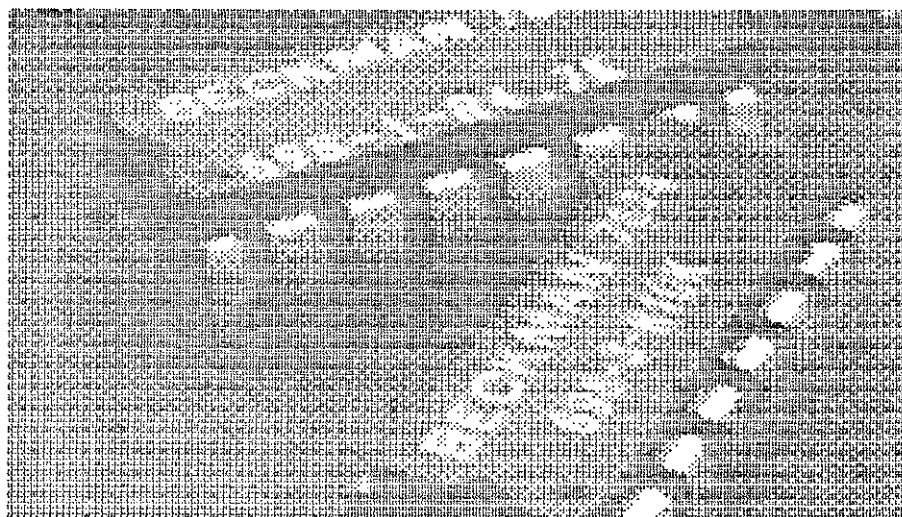
Bedienung: Das Interface kann mit folgendem BASIC Befehl arbeiten:

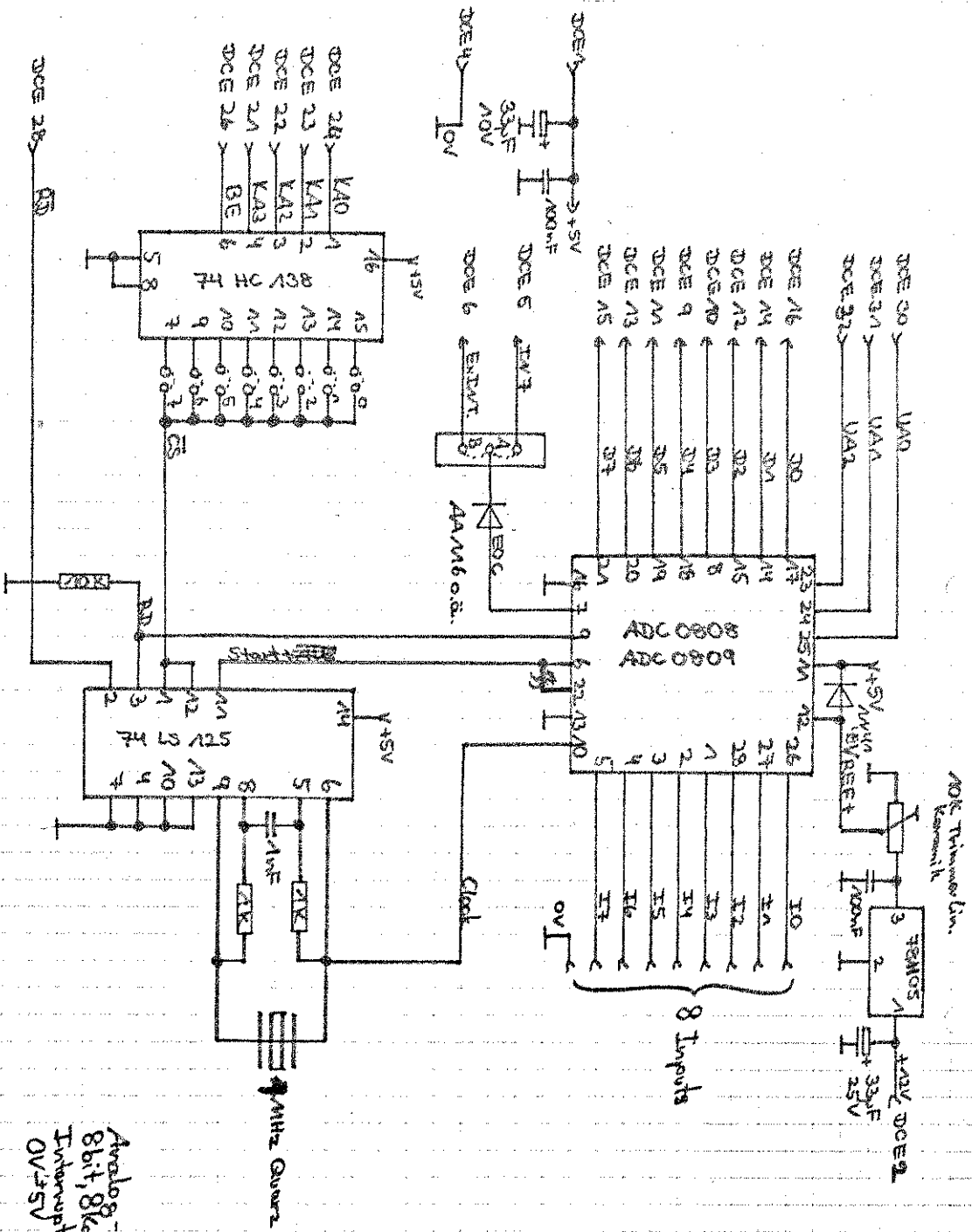
```
AZ=INP(#nm)    n  $\hat{=}$  Kartenadresse 0-7  
                m  $\hat{=}$  Kanal 0-7
```

Daten: AZ kann Daten von #00-#FF enthalten.

Interruptsteckbrücke: EOC wird "1", wenn der Umsetzvorgang beendet ist

Die Einstellung des zu wandelnden Spannungsbereichs wird mit V_{REF} vorgenommen. Bitte fragen Sie bei Interesse beim Club nach den Datenblättern.





Analog-Digitalwandler,
 8bit, 8kanal, < 100µs
 Interrupt, regelbarer Bereich
 0V-5V

Marty Stohrer
 Neuselbriun 51
 93500 Nürnberg
 O. G. 1985
 HW 44, 2
 O. G. 1985

Parallele Druckerschnittstelle ohne Interface

Um einen Drucker mit Centronics-Parallel-Schnittstelle am DAI zu betreiben, benötigt man nur ein Kabel und ein kleines Maschinenprogramm. Dies gilt allerdings mit der Einschränkung, daß am DCE-Bus allenfalls noch die DCR angeschlossen ist, d.h. Floppy-Benutzer brauchen ein DCE-Interface. Meiner Meinung nach müßte es auch mit Real-World-Karten, wie im Artikel FW4 beschrieben, ohne Interface funktionieren, allerdings konnte ich das bis jetzt noch nicht ausprobieren.

Eine Centronics-Druckerschnittstelle besteht in der Hauptsache aus

- 7..8 Datenleitungen zum Übertragen der Bytes
- 1 BUSY-Leitung, die anzeigt, ob der Drucker bereit ist, ein neues Zeichen zu empfangen
- 1 STROBE(neg.)-Leitung, die dem Drucker bei low-high Wechsel anzeigt, daß der Zustand der Datenleitungen gültig ist
- 1 ACKNLG(neg.)-Leitung

Dazu kommen meist noch weitere Steuer- und Statusleitungen wie z.B. ERROR oder INIT o.ä.

Die Datenausgabe sieht etwa so aus:

1. warten, bis der Drucker bereit ist
2. Daten anlegen
3. STROBE auf 0 setzen
4. warten, bis ACKNLG auf 0 geht
5. STROBE wieder 1 setzen

Nun haben wir im DAI den 8255, der den DCE-Bus bedient. Er kann im MODE 1 die Schritte 3 bis 5 selbsttätig durchführen (siehe hierzu auch das Datenblatt). Hier bietet sich nun folgende Belegung an: (Beispiel EPSON-Drucker)

EPSON	Signal	DAI-Pin	Port+Bit
1	STROBE	17	C7
10	ACKNLG	18	C6
11	BUSY	20	C4
2	DATA 0	16	A0
3	DATA 1	14	A1
4	DATA 2	12	A2
5	DATA 3	10	A3
6	DATA 4	9	A4
7	DATA 5	11	A5
8	DATA 6	13	A6
9	DATA 7	15	A7
19..30	GND	4	MASSE

Das Kabel sollte möglichst kurz sein und unbedingt 'twisted-pair' (jede Signalleitung ist mit einer Masseleitung verdreht) oder ein Flachbandkabel mit je einer Masseleitung zwischen den Signalleitungen sein!

Der Port A läuft in MODE1-Output, d.h. die Bits 6 und 7 des Port C sind festgelegt für ACK und STROBE. Bleiben für weitere Signale die Bits 0 bis 3 und 4 und 5. Die Bits 0 bis 3 werden von der DCR benutzt, also legen wir BUSY auf Bit 4 und Bit 5 bleibt für einen weiteren Eingang frei. Daraus ergibt sich das Kontrollwort für den 8255 zu #A9.

Vor dem Drucken muß das Kontrollwort eingeschrieben werden:

POKE #FE03,#A9

Die Ausgabe geschieht dann einfach durch POKE #131,3 oder 4

Das Programm zur Zeichenausgabe:

```

02DD C30003          JMP  OUTCH

0300 EF03          OUTCH RST5 / DATA :03  ZEICHEN AUF SCHIRM
0302 F5           OUTCH1 PUSH PSW          ZEICHEN MERKEN
0303 3A03FE       BUSY  LDA  :FE03         STATUS-WORT DES 8255
0306 E680                ANI  :80         BIT 7 GIBT DEN ZUSTAND
                                      DER STROBE-LEITUNG WIE-
                                      DER, DIE ERST WIEDER 1
                                      WIRD, WENN ACK AUF 1
                                      GEGANGEN IST.

0308 CA0303       JZ    BUSY              WARTEN
030B F1           POP  PSW              ZEICHEN WIEDER IN A
030C 3200FE       STA  :FE00            DATEN AUF PORT A, LÖST
                                      DAS HANDSHAKE AUS

030F C9           RET

```

oder Abfrage der BUSY-Leitung:

```

0303 3A02FE       BUSY  LDA  :FE02         PORT C
0306 E610                ANI  :10         BUSY-BIT DIREKT
0308 C20303       JNZ  BUSY              WARTEN, BIS 1

```

Hierbei muß der Drucker allerdings bei einem Carriage-Return automatisch ein Line-Feed ausführen; ist das nicht der Fall, wird das Programm ab 030F geändert:

```

030F FE0D                CPI  :0D         IST ES CAR-RET ?
0311 C0                RNZ                NEIN, ALSO SCHLUSS
0312 3E0A                MVI  A,:0A         LINE-FEED AUSGEBEN
0314 C30203       JMP  OUTCH1

```

Beim Betrieb mit der DCR ist zu beachten, daß bei jeder Operation das Kontrollwort überschrieben wird, zusätzlich wird beim Aufruf von 'DCR' oder 'CAS' der Sprung bei #2DD vernichtet, u.U. muß also beides vor dem Drucken neu 'gepoket' werden.

Das Binden dieses Programmes mit 'Formatdruck' geschieht, indem im Source-Text Zeile 110 oder im BASIC-Programm Zeile 60030 das 3. und 4. DATA die Adresse von OUTCH erhalten.

Nun kann man auch sehen, warum der Betrieb mit DCR möglich ist, mit Floppy jedoch nicht: da die Ausgangssignale des Druckers dauernd am DCE-Bus anliegen, sind diese Leitungen blockiert, d.h. von einem anderen Gerät kommende Signale auf denselben Leitungen werden verfälscht. Da die Floppy jedoch außer Bit 4 alle Bits des Port C benutzt, ist der gleichzeitige Betrieb verhindert. Die DCR hingegen braucht nur die Bits 0 bis 3.

Ein DCE-Interface würde nur dann, wenn die eingestellte DCE-Adresse auf Port B anliegt, die Druckerleitungen durchschalten, ansonsten ist der Drucker gewissermaßen 'abgeklemmt'.

Kürzlich mußte ich leider feststellen, daß die Floppy gar nicht voll DCE-kompatibel ist, daß also selbst mit Interface der Betrieb des Druckers unmöglich ist. Das liegt an einem Fehler im Slave-DOS (EPROMs in der Floppy), der bewirkt, daß sie im Wartemodus die obere Hälfte ihres Port C auf Output läßt, was die gleiche Wirkung hat, wie ein Drucker ohne Interface.

Für Eprom-Brenner: SDOS0 Adresse #070F von #93 auf #9B ändern.

In diesem Beitrag soll nun der Betrieb eines Druckers mit Centronics-Schnittstelle mit Hilfe des DAINashop-Interfaces beschrieben werden. Er nimmt Bezug auf HP1, diesen deshalb vorher noch einmal ansehen!

Das Interface erlaubt den gleichzeitigen Anschluß von Floppy (nach SDDS-Änderung), DCR, Drucker und Real-World-Karten.

Die Belegung des DCE-Bus ist für die Daten- und Steuerleitungen die gleiche, bis auf BUSY, dessen Anschluß der Interface-Beschreibung entnommen werden muß. Die BUSY-Leitung ist bei dem hier vorgestellten Programm überflüssig. Dies setzt allerdings voraus, daß der Drucker ein Zeichen empfangen kann, trotzdem er nicht druckbereit ist. Meines Wissens ist das bei allen EPSON- und ITOH-Druckern der Fall. Andernfalls muss anstelle des Status-Bit die BUSY-Leitung abgefragt werden.

Das Maschinenprogramm macht alle POKEs von Kontrollworten und DCE-Adressen überflüssig und hat zudem zwei Einsprungpunkte für die Ausgabe auf Drucker und Bildschirm oder auch nur auf den Drucker.

02DD		C3FE02	JMP	SCRPAR	
02FE	SCRPAR	EF	RST	5	AUSGABE AUF
02FF		03	DATA	:03	BILDSCHIRM
0300	PAR	F5	PUSH	PSW	ZEICHEN RETTEN
0301		3A01FE	LDA	:FE01	DCE-ADRESSE
0304		FE02	CPI	:02	=PRINTER-ADRESSE?
0306		CA1303	JZ	BUSY	8255 SCHON OK
0309		3EA1	MVI	A,:A1	MODE 1 FÜR 8255
030B		3203FE	STA	:FE03	EINSCHREIBEN
030E		3E02	MVI	A,:02	PRINTER-ADRESSE
0310		3201FE	STA	:FE01	EINSCHREIBEN
0313	BUSY	3A02FE	LDA	:FE02	STROBE-LEITUNG
0316		E680	ANI	:80	ABFRAGEN
0318		CA1303	JZ	BUSY	WARTEN, BIS HIGH
031B		F1	POP	PSW	ZEICHEN ZURÜCK
031C		3200FE	STA	:FE00	UND AUSGEBEN
031F		C9	RET		

Zum Einbinden in FORMATDRUCK (NP2,1) die Adresse von PAR in Zeile 110 einsetzen. In DAINatext ist ein Vorläufer von FORMATDRUCK, hier die Adresse von PAR (low,high-Byte) in #35C,#35D einsetzen und diese Routine hinter den ML-Teil legen.

BASIC-Lader: (lädt ML-Programm in ein Array)

```

40000 DATA DRUCKROUTINE
40010 DATA #EF,#03,#F5,#3A,#01,#FE,#FE,#02,#CA,#00,#00,#3E
40020 DATA #A1,#32,#03,#FE,#3E,#02,#32,#01,#FE,#3A,#02,#FE
40030 DATA #E6,#80,#CA,#00,#00,#F1,#32,#00,#FE,#C9
40040 RESTORE
40050 READ A$: IF A$<>"DRUCKROUTINE" THEN 40050
40060 DIM DR$(8):SCRPAR%=VARPTR(DR$(0))
40070 FOR ADR%=SCRPAR% TO SCRPAR%+33:READ BYTE%:POKE ADR%,BYTE%:NEXT
40080 POKE #2DD,#C3:POKE #2DE,SCRPAR% MOD 256:POKE #2DF,SCRPAR%/256
40090 BUSY%=SCRPAR%+21
40100 POKE SCRPAR%+9,BUSY% MOD 256:POKE SCRPAR%+10,BUSY%/256
40110 POKE SCRPAR%+27,BUSY% MOD 256:POKE SCRPAR%+28,BUSY%/256
40120 RETURN
    
```


Die Software der Commodore-Floppy

Die Software zum Antreiben der Floppy befindet sich wie bei der MDCR in einem EPROM im Adressenbereich F000-F7FF. Die Software ist soweit kein DOS, da die VC 1541 ein 16kB DOS-ROM und 2kB RAM im Gerät besitzt. Allerdings werden zum Ansteuern des eigentlichen DOS einige Befehle (neben der LOAD/SAVE-Logik) benötigt, die auf 'NCU'-Basis (New-Command-Unit, siehe Beschreibung in dieser Ausgabe) aufgebaut sind, d.h., die neuen Befehle sind gleichwertig zu den Standardbefehlen und können wie diese auch in Basic-Programmen eingegeben werden.

Diese Beschreibung dient in erster Linie dazu, diese neuen Befehle zu erklären, da die Möglichkeiten des DOS (Sequentielle, Relative und Random-Access-Files, Blocklesen/schreiben etc.) im Handbuch (bei mir ein Deutsches und ein Englisches!) ausführlich beschrieben sind. Zusätzlich gibt es ja auch noch das 'Floppy-Buch'. Einige Tricks und Programme werden bei Gelegenheit veröffentlicht.

Die Software mit den NCU-gesteuerten Befehlen (es handelt sich allerdings um eine Mini-NCU-Version, die nur die wichtigsten Dinge beinhaltet) ist durch die an den DCE-Bus gehängte Reset-Logik sofort betriebsbereit. Beim Reset wird die NCU initialisiert, der Befehl DISK (s.u.) durchgeführt, die Baudrate wird neu gesetzt (nur interessant für Besitzer von Druckern mit Baudrate [#FFFF51:#C0]) und die Fehlerroutine auf 'Rückkehr ins Basic' gestellt (durch Ändern ist bei Disk-Fehlermeldungen eine Rückkehr in verschiedene MP-Systeme wie Assembler möglich. Die Rückkehradresse muß in die beiden Bytes #00CE/CF eingetragen werden).

Die neuen Befehle sind teilweise von Commodore übernommen (und bis auf INPUT# auch besser), teilweise sind aber auch ganz neue Befehle dabei, die das Arbeiten deutlich erleichtern:

CLS	OPEN	INPUT#	TEXT#
DISK	CLOSE	GET#	
CAS	STATUS	MGET#	
DIR	ON.EOF	PRINT#	(Kurz: P#)
GMP	COM	PUT#	
IF		MPUT#	

a) CLS (=Clear Screen) löscht den Bildschirm.

b) DISK schaltet die LOAD/SAVE-Vektoren auf Diskettenbetrieb um und legt die Standard-Primäradresse auf 8. Die Primäradresse ist die in der Floppy hardware-mässig festgelegte Gerätenummer, mit der alle an den Rechner und eine Leitung angeschlossenen Geräte unterschieden werden. Durch unterschiedliche Primäradressen können zwei Floppys oder Commodore-Drucker etc. angeschlossen werden. Dadurch, daß eine Standard-PrimAdr. definiert wurde, ist es nicht wie bei Commodore nötig, diese Adresse bei LOAD/SAVE o.ä., wie im Handbuch beschrieben, anzugeben. Ein Programm/UT-File/Daten o.ä. wird also immer von dem Gerät gelesen (o. geschrieben), dessen PrimAdr. in dem Byte #00C9 steht. Der Befehl DISK löscht ausserdem noch die ON.EOF-Eintragung (s.u.).

c) CAS schaltet die Vektoren wieder auf Cassettenbetrieb.

d) DIR listet die Directory auf. Im Gegensatz zu Commodore muß die Dir. nicht als Basicprogramm geladen und dann gelistet werden, sondern wird unmittelbar ausgedruckt. Der Ausdruck kann aber wie ein Listing mit Space/Break gestoppt oder abgebrochen werden. Die von Commodore vorgesehene Möglichkeit, nur bestimmte Namen aufzulisten, wird durch anhängen eines einsprechenden Strings genutzt (wie LOAD / LOAD "..."):

```
DIR
```

```
DIR "SG%"    (alles, was mit SG anfängt. Die 'Joker' %? gibt es generell bei  
              Filenamen)
```

Der DIR-Befehl holt sich die Directory von der Standard-PrimAdr (8).

e) GMP (Go Machine Program) ist eine Abkürzung für 'CALLM #400', da viele Utilities bei #400 beginnen.

f) IF ist kein neuer Befehl, aber notgedrungenenerweise muß er neu interpretiert werden. Ein Abfallprodukt: Bei 'IF ... THEN ...' kann das THEN entfallen. Hier muß noch gesagt werden, daß alle Befehle natürlich nur dann laufen, wenn man die Disk-NCU im Rechner hat. Weitergabe von Programmen an Nicht-Floppy-Besitzer mit oben aufgeführten Befehlen ausser 'IF' ist ohne MP-Zusatz (spezielle NCU) nicht möglich. Da es aber ausser CLS,GMP,PRINT#0 und PUT#0 reine Disk-Befehle sind, dürfte dies kein Problem sein.

g) Der Befehl OPEN hat ähnlich zu Commodore die Syntax

```
OPEN Kanal PrimAdr SekAdr  
o. OPEN Kanal PrimAdr SekAdr "...."
```

Die Bedeutung der Sekundäradresse steht im Handbuch (man kann ≈ 6 Files auf einmal bearbeiten); die Kanalnummer ist die Nummer, mit der man bei den Befehlen, die oben in der 3. Spalte stehen, das gewünschte Gerät (mit PrimAdr und SekAdr) anspricht. PrimAdr und SekAdr werden ja durch den OPEN-Befehl definiert, die physikalischen Adressen interessieren danach nicht mehr (Ausnahme sind die Blockschreib/lese-Kommandos, s. Handbuch). Im Gegensatz zu Commodore muß die Kanalnr zwischen 1 und 6 liegen, d.h., man kann 5-6 Files auf einmal bearbeiten (zusätzlich noch einen LOAD/SAVE-Vorgang und DIR). Jeder geöffnete Kanal muß wieder mit CLOSE geschlossen werden. Geschieht dies nicht, weil z.B. ein Basic-Fehler auftritt, so zeigt die Floppy dies durch eine LED an (ausser SekAdr=15 und nach STATUS). Bei dem Versuch, den gleichen Kanal nochmals zu öffnen, erscheint 'INVALID NUMBER' und der alte Kanal wird vorsichtshalber automatisch geschlossen. Bei OPEN kann eine beliebige PrimAdr zwischen 0-15 gewählt werden. Existiert dieses Gerät nicht oder ist es nicht eingeschaltet (dies gilt auch für die implizierte Adresse 8 bei LOAD/SAVE o.ä.), so gibt es die Fehlermeldung 'INVALID DEVICE' (Commodore: 'DEVICE NOT PRESENT').

h) Mit 'CLOSE Kanal' werden die einzeln geöffneten Kanäle wieder geschlossen. Wie im Handbuch beschrieben, bedeutet ein Schliessen der SekAdr 15 ein Schliessen aller Files in der Floppy. Deshalb den 'Kommandokanal 15' als letzten Schliessen. Diese Eigenschaft nutzt 'CLOSE 0' aus: CLOSE 0 schliesst im Rechner alle Kanäle und die (nicht notwendigerweise geöffnete) SekAdr 15 des Standard-PrimAdr-Gerätes. Ist also nur die Floppy bei allen OPEN's angesprochen worden, reicht ein CLOSE 0 (sonst nicht!).

Wie bei den anderen Kanalabhängigen Befehlen ausser OPEN gilt, daß, wenn der Kanal nicht geöffnet wurde, der Basic-Fehler 'INVALID NUMBER' erscheint.

j) STATUS ist eine Abkürzung der im Handbuch beschriebenen Prozedur, sich eine eventl. vorhandene Fehlermeldung ausdrucken zu lassen. STATUS druckt die Fehlermeldung (bzw. OK) auf den Bildschirm und/oder Drucker.

k) 'ON.EOF' und 'ON.EOF z' sind eine Möglichkeit, beim Erreichen des Fileendes beim Lesen von Files zu einer bestimmten Zeile zu springen. Bei Commodore muß ständig ein Flag abgetestet werden (es steht in der Adresse #0007), hier wird automatisch der INPUT#,GET#,MGET# oder TEXT# Befehl abgebrochen und ein entsprechendes GOTO durchgeführt. Mit ON.EOF ohne Zeilennummer muß diese Funktion wieder abgeschaltet werden (auch der DISK-Befehl führt dies aus), da dies in der Mini-NCU-Version nicht automatisch geschehen kann. Existiert beim 'Erreichen' von EOF (genauer: es wird versucht, das erste Byte nach EOF bzw. [#0007]#0 zu lesen) die Zeile nicht bzw. ist die Funktion ausgeschaltet oder tritt der Fehler beim Ausführen einer Kommandozeile auf, so erscheint der Basic-Fehler 'OUT OF DATA' und der Kanal wird autom. geschlossen. Bei ON.EOF-Goto bleibt der Kanal offen.

l) COM "..." ist eine Möglichkeit, die umständliche Commodore-Methode zu umgehen, einfache Befehle wie Formatieren, Löschen, Rename etc. an die Floppy zu geben. Bei Commodore muß die KommandoSekAdr geöffnet (OPEN x 8 15), der Befehl gedruckt (PRINT# x;"...") und der Kanal x wieder geschlossen werden (CLOSE x). Bei den komplizierteren Befehlen ist dies meistens ebenfalls nötig, aber die Befehle, die 'schnell mal eben' gegeben werden müssen, sind so wesentlich einfacher.

m) TEXT# Kanal

Hiermit werden Bytes aus dem (Lese-geöffneten) Kanal gelesen und direkt ausgegeben (je nach Zustand von [#131] auf Bildschirm, Drucker etc.). Die Ausgabe endet mit Erreichen eines CHR\$(13), welches nicht mehr ausgegeben wird. Mit diesem Befehl kann schnell überblickt werden, ob eine (Text-) Datei in Ordnung ist oder es wird ein 'gebufferter' Text gedruckt.

n) INPUT# Kanal; variablen, ...

Dies entspricht einem INPUT von Zahlen und Strings von der Diskette. Der INPUT# erfolgt (aus EPROM-Platzmangel) so, daß eine Zeile bis zum CR gelesen und ausgedruckt wird und dann ein 'normaler' INPUT erfolgt. 'SOME INPUT IGNORED' wird unterdrückt, ein Nachladen von Text bei Zeichenmangel erfolgt automatisch. Bei SYNTAX ERROR o.ä. wird das Programm abgebrochen, die Kanäle bleiben geöffnet. Der Nachteil dieser Methode 'über den Bildschirm' ist, daß der Fehlerkanal nicht so leicht wie im Handbuch gelesen werden kann (sinnvoll nur mit GET#). Der Vorteil: Man kann durch INPUT# 0; ... ein normales INPUT über die Tastatur durchführen und kann somit die Daten leicht wahlweise direkt oder abgespeichert eingeben/einlesen. In beiden Fällen sieht man die Eingabedaten auf dem Bildschirm. Ausserdem ist zu berücksichtigen, daß mit MPUT#/MGET# Zahlenvariablen viel genauer und platzsparender gespeichert werden können.

```
INPUT# N;ANZ,N$(0)
```

```
INPUT# 5;E,T$,TR,SEC
```

o) GET# Kanal; variablen, ...

Es wird pro Variable je ein Byte gelesen und der Variablen entweder als Zahlenwert 0-255 oder bei #-Variablen als 1-Zeichen-String zugewiesen.

```
GET# 1;AL,AH,H,H,H,A1$,A2$
```

p) MGET# Kanal,Adr Anzahl

Mit MGET# werden 'Anzahl' Bytes gelesen und ab der angegebenen Adresse abgespeichert. Bei Eintreten von EOF steht in #0048/49, wieviele Bytes nicht gelesen wurden. Dieser Befehl eignet sich hervorragend, um grössere Datenmengen auf einmal zu lesen (da er gegenüber einzelnen GET#'s sehr schnell ist), um unbenötigte Daten schnell zu überlesen und insbesondere beim Sektoren-Lesen (Block-Read).

```
MGET# 2,#B350 #C000-#B350    liebt MODE 0-Bild  
MGET# N,#8000 256           für Block-Read etc.
```

q) PRINT# Kanal; (Daten wie bei PRINT, Eingabe Kürzel: P# Kanal;...)

Die Daten (Zahlen- und Textstrings) werden an den schreibgeöffneten Kanal gegeben und auf Diskette abgelegt. Optionen: Positive Zahlen erscheinen ohne Vorblank, ein Komma zwischen den Daten bedeutet, daß ebenfalls ein Komma ausgegeben wird, und als Bonus (auch bei PUT#,MPUT#): mit Kanal=0 werden die Daten nicht auf Disk, sondern an den Bildschirm o.ä. ausgegeben. Vorblank- und Komma-Option können so auch allgemein genutzt werden.

```
P# 2,"Daten: ";PI,34        (- mit CHR$(13) als Abschluss
```

```
P# 0,"Ohne Vorblank",N;
```

```
PRINT# 5,"B-R:2,0",TR,SEC
```

r) PUT# Kanal; ...

PUT# druckt Strings wie der PRINT#-Befehl, Zahlen werden allerdings als CHR#()-Zeichen aufgefaßt. Komma und Semikolon haben keine Bedeutung

```
PUT# 2,NA,"#",120,13
```

```
PUT# 0,27,"E",27,"L008"
```

```
PUT# N,LEN(A$),A$
```

s) MPUT# Kanal,Adr Anzahl

Schreiben von 'Anzahl' Bytes ab Adresse. Wichtig für grosse Datenmengen wie Bildschirme etc.

```
Trick: MPUT# x,VARPTR(A) 4    speichert Zahl platzsparend ab  
       MGET# x,VARPTR(...) 4  das Gegenstück
```

Weitere Anmerkungen:

Der Fehler 'TYPE x', z.B. TYPE 1, tritt auf, wenn bei Ladebefehlen wie LOAD, LOADA, UT-R bei passendem Namen der Typ (Basic,Code,Data) nicht paßt.

Das Source-Listing und MP kann bei mir angefragt werden.

Die Software ist eine Gemeinschaftsproduktion von H.Steves (Bus-Ansteuerung) und mir (NCU, Befehle).

Lock und Unlock für die Commodore-Floppy

Jeder Besitzer der VC-1541 kennt den Stern vor einem Dateityp zur Kennzeichnung eines offenen Files. Das "Kleiner"-Zeichen hinter dem Filetyp (z.B. "PRG<") ist dagegen weniger geläufig. Derartig gekennzeichnete Dateien lassen sich durch den SCRATCH-Befehl nicht mehr löschen. In den Handbüchern der Floppy wird dieser Löschschutz leider verschwiegen. Der Grund hierfür ist darin zu suchen, daß das Disketten-Betriebssystem keine Befehle wie "LOCK" und "UNLOCK" zum Setzen und Rücksetzen des Löschschutzes bereitstellt.

Verantwortlich für den Scratch-Schutz ist Bit 6 im Filetypbyte. Ist es gesetzt, so wirkt der SCRATCH-Befehl für diese Datei nicht.

Das unten aufgelistete Programm nutzt die Möglichkeit, im Floppy-RAM 6502-Maschinenprogramme auszuführen. Gestartet wird das Programm mit den Userbefehlen UC und UD. Das hat im Gegensatz zu ME (Memory-Execute) den Vorteil, daß Parameter mit übergeben werden können. In diesem Fall einen oder mehrere Filenamen.

Geschützt wird eine Datei mit COM "UC:Filename". Aufgehoben wird der Schutz mit UD. Da die Syntax dem SCRATCH-Befehl entspricht, können auch Joker oder mehrere Filenamen verwendet werden. Mit dem Stern können z.B. alle Files geschützt werden. Das Maschinenprogramm ist auf der Diskette in einem Userfile (Typ:USR) abgelegt. Diese Methode hat gegenüber dem Abspeichern in einem Sektor und Aufruf mit BE (Block-Execute) den Vorteil, daß das Programm bei einem VALIDATE nicht verlorengeht. Außerdem existiert ein Directoryeintrag und man muß sich die Sektorposition nicht merken.

Leider teilen die Userfiles das Schicksal des Löschschutzes, sie sind in den Handbüchern nicht dokumentiert. Das liegt wahrscheinlich an der etwas umständlichen Handhabung. Außerdem sind Userfiles üblicherweise bei einem Reset oder beim Einschalten selbststartend. Diese Möglichkeit ist zwar im DOS vorgesehen, funktioniert aber anscheinend nicht.

Zunächst zum Aufbau eines Userfiles. Die ersten beiden Bytes bilden die Startadresse des Programms. Zuerst das Lo- und dann das Hbyte. Das zweite Byte ist ein Zähler für die Anzahl der nachfolgenden Datenbytes. Nach den Daten folgt noch ein Prüfsummenbyte. Nach der Prüfsumme können weitere Programme mit Startadresse u.s.w. folgen. Da die Anzahl der Daten nur in einem Byte abgelegt ist, müssen längere Programme in mehrere Blöcke zerlegt werden.

Ein Userfile wird mit dem "Und-Befehl" ('&') in das RAM der Floppy geladen (z.B. COM "&UN/LOCK"). Das '&' muß auch das erste Zeichen des Filenamens sein, da es beim Auswerten des Namens mitgelesen wird (wahrscheinlich ein Fehler im DOS). Nach dem Laden wird das zuletzt gelesene Programm automatisch gestartet. Nachdem es mit RTS abgeschlossen ist, wird der vorherige Programmblock gestartet u.s.w., bis das erste Programm erreicht ist. Das kann unangenehm sein, falls ein langes Programm zerlegt wurde (s.oben). Es muß entweder im letzten Block der Stackpointer angepasst werden, oder das erste Byte eines Programmblocks ist jeweils ein RTS. Auf diese Weise kann auch der Autostart des gesamten Files verhindert werden.

Neben den bekanntesten Fehlermeldungen gibt es beim Arbeiten mit Userdateien noch zwei weitere:

50 RECORD NOT PRESENT	Es liegt ein Prüfsummenfehler vor
51 OVERFLOW IN RECORD	Der letzte Programmblock enthält zuwenig Datenbytes (Das Längenbyte ist fehlerhaft)

Das Userfile wird mit dem folgenden BASIC-Programm erzeugt:

```

10 OPEN 1:8:2 "&UN/LOCK,U,W"
20 READ ADDR:IF ADDR>#FFFF THEN CLOSE 1:END
30 CHKSUM=0:BYTE=ADDR IAND #FF:GOSUB 70
40 BYTE=ADDR SHR 8:GOSUB 70
50 READ BYTE:GOSUB 70:FOR I=1 TO BYTE
60 READ BYTE:GOSUB 70:NEXT:PUT# 1,CHKSUM:GOTO 20
70 CHKSUM=(CHKSUM+BYTE) MOD #FF:PUT# 1,BYTE:RETURN
100 DATA #610,#102,#A0,#05,#B9,#70,#06,#99,#00,#05,#98,#10,#F7,#60
110 DATA #A9,#29,#8D,#56,#06,#A9,#BF,#8D,#57,#06,#4C,#33,#06,#A9,#09
120 DATA #8D,#56,#06,#A9,#40,#8D,#57,#06,#A9,#0A,#8D,#2A,#02,#20,#EE
    ,#C1,#20,#98,#C3
130 DATA #20,#20,#C3,#20,#CA,#C3,#A9,#00,#85,#86,#20,#9D,#C4,#30,#15
    ,#20,#B7,#DD,#90,#0B
140 DATA #A0,#00,#B1,#94,#EA,#EA,#20,#B9,#C8,#E6,#86,#20,#8B,#C4,#10
    ,#EB,#20,#10,#06
150 DATA #A5,#86,#85,#80,#68,#68,#A9,#00,#4C,#78,#C8,#4C,#29,#06,#4C
    ,#1C,#06,99999

```


Dieser Artikel beschreibt das Prinzip von relativen Dateien und wie mit ihnen beim DAI-DOS 1541 gearbeitet werden kann.

Bitte beachten Sie hierbei: Das Prinzip der relativen Dateien ist für alle Diskettensysteme gültig, egal ob es sich um die Commodore Floppy (mit dem deutschen oder holländischen Interface System) oder PRODATA Floppys usw. handelt. Nur werden die verschiedenen notwendigen Arbeitsschritte vom DOS mehr oder weniger automatisch erledigt, bzw. unterstützt - je nach DOS.

Prinzip der relativen Dateien:

Relative Dateien sind sogenannte random access Dateien. Sie sind diskettenorientiert, was bedeutet, daß nicht der ganze Inhalt einer Datei auf einmal in den RAM des Computers eingelesen wird, sondern er befindet sich nur auf der Diskette. Es wird also nur ein kleiner, augenblicklich benötigter, Teil in den Hauptspeicher gelesen. Der Vorteil liegt auf der Hand: Es können Dateien bis zur maximalen Diskettenkapazität verarbeitet werden. Dies ist z.B. bei der Commodore Floppy max. 164 KB - also viel mehr als sonst beim DAI möglich ist, nämlich max. 32KB INT/FPT Arrays oder 20KB String Arrays.

Eine relative Datei besteht aus einer Reihe von Records. Ein solcher Verbund ist ein Informationsblock, der aus verschiedenen einzelnen Feldern bestehen kann, z.B. bei einer Adreßdatei: Name, Straße, Stadt usw. Die Datei besteht nun aus hintereinandergereihten Records, jeweils bestehend aus einem (oder mehreren Feldern).

Neben der Kapazität gibt es noch einen Vorteil:

Änderungen können sehr schnell und einfach vorgenommen werden. Man lädt einfach den betreffenden Record, verbessert die Information und speichert nur diesen Record wieder weg. Der Rest der Datei muß hierbei nicht gelesen und später wieder weggeschrieben werden!

Datei / Record Struktur:

Bei relativen Dateien müssen die Records eine feste Länge besitzen. Dies wird vom VC1541 Operating System diktiert. Weiterhin sind nur Strings (¶) als Records erlaubt. Hierbei sind max. 254 Zeichen zugelassen.

Die max. Anzahl der Records beträgt 65535, was soviel heißt wie: "Records bis die Diskette platzt".

In unserer obigen Adreßdatei wollen wir nun für den Namen 20, für die Straße ebenfalls 20 und für die Stadt 15 Zeichen für das betreffende Feld annehmen. Dies ergibt eine Record Länge von insgesamt 55 Zeichen. Diese Struktur (die Längendef.) muß man sich gut überlegen, denn wenn sie einmal besteht kann sie später nicht mehr geändert werden!

Jeder Eintrag in ein Feld muß nun exakt diese Länge besitzen. Sollte er zu kurz sein, so müssen einfach Leerzeichen angehängt werden, um auf die voreingestellte Länge (s.o) zu kommen.

Kommunikation DAI => VC1541

Um mit relativen Dateien mit dem holländischen DAI-DOS1541 System arbeiten zu können ist ein zusätzliches Monitorprogramm "FDDMON/O" notwendig. Dieses kleine Maschinenprogramm erlaubt den Zugriff auf alle Möglichkeiten des VC1541 Diskettenlaufwerks. Dieses Programm muß zusammen mit dem BASIC Programm geladen werden. Hierfür gibt es der BOOT Befehl, z.B. BOOT "FDDMON/O,MAILING LIST/B"

In Abweichung vom normalen Arbeiten mit Dateien, wie z.B. BASIC Programmen, müssen relative Dateien explizit vom Benutzer geöffnet und geschlossen werden. Für diese Arbeiten wird der Kanal 3 in Verbindung mit Kanal 15 (Fehler-/Kommandokanal) der VC1541 benutzt. Der Fehlerkanal wird ebenfalls hier nicht automatisch gelesen.

Es kann immer nur eine relative Datei zur gleichen Zeit bearbeitet werden. Benötigt man mehrere Dateien, so muß die erste zunächst geschlossen, danach die zweite geöffnet werden. Ansonsten erscheint die Fehlermeldung "NO CHANNEL".

Der Aufruf der Unterprogramme erfolgt durch CALLM, deshalb ist es sinnvoll sich hierfür synonyme Namen, wie OPEN3, FRNT3, ERROR, CLOSE3 und INP3 zu überlegen.

Öffnen von Dateien:

```
10  FILENAME$="MAILING LIST"  
20  RECLENGTH=56 (1 größer als die reale Länge)  
30  OPEN$=FILENAME$+",L,"+CHR$(RECLENGTH)  
40  CALLM OPEN3,OPEN$
```

Positionierung innerhalb der Datei:

```
50  RECNR=200 (Als Beispiel)  
60  HB=RECNR SHR 8  
70  LB=RECNR IAND 255  
80  POS$="P"+CHR$(3)+CHR$(LB)+CHR$(HB)+CHR$(0)  
    ^Kanal-Nr.           ^ Byte 0 innerhalb  
                        des Records  
90  CALLM PRNT15,POS$
```

Schreiben in rel. Dateien:

```
1) evtl. Datei öffnen, falls nicht bereits geschehen  
2) Positionieren  
3) Feldinhalt eingeben (errechnen) und auf die richtige Länge  
   bringen (ggf. Spaces anfügen!)  
4) OP$=NAME$+STRASSE$+STADT$  
   CALLM PRNT3,OP$
```

Diese Eingabeprozedur kann für andere Records wiederholt werden.
Anschließend muß die Datei geschlossen werden:

Das Schließen von Dateien:

```
120 CALLM ERROR: CALLM CLOSE3
```

Man sollte immer den Fehlerkanal vor dem Schließen der Datei lesen.
Eine evtl. Fehlermeldung wird dann auf dem Bildschirm angezeigt.

Lesen aus rel. Dateien:

Daten können auf zwei Arten gelesen werden:

- 1) Byte für Byte oder als ein (GET)
- 2) String (INPUT)

Das Arbeiten mit Records wirkt vielleicht ein wenig gekünstelt, aber dies ist notwendig, um eine Verbindung zwischen dem BASIC und Monitorprogramm herzustellen.

Es werden zwei Variablen benötigt:

```
1  IN$=" "  
2  IP$=SPC(255)
```

IN\$ wird als Zwischenspeicher für GET und IP\$ als Speicherbereich für INPUT verwendet. Das Lesen geschieht folgendermaßen:

- 1) evtl. Datei öffnen, falls nicht bereits geschehen
- 2) Positionieren
- 3) CALLM GET3,IN\$ oder
 CALLM INP3,IP\$

Wenn das erste Zeichen des Records gleich CHR\$(#FF) ist, so ist dieser Record, gemäß der Commodore Struktur leer. Dies kann z.B. durch CALLM GET3,IN\$:IF ASC(IN\$)=#FF GOTO leer überprüft werden. Danach muß allerdings neu positioniert werden, da das erste Zeichen ja schon gelesen wurde!

Bei CALLM INP3,IP\$ stehen nachher die gelesenen Zeichen in IP\$. Aus diesem String muß man sich dann die gewünschten Zeichen gemäß der definierten Länge "herausschneiden", z.B. NAME\$=LEFT\$(IP\$,20) usw.

Anschließend ist die Datei zu schliessen (s.o)!

Literaturangaben:

Für eine detaillierte Erklärung mit vielen Beispielen wird auf "Das große Floppy-Buch", herausgegeben von DATA-Becker GmbH, Düsseldorf, verwiesen.

Bezugsmöglichkeiten für das Monitorprogramm:

Die Diskette 'FDD TOOLKIT/1' mit dem Programm "FDDMON/O" kann bei Micro Service, Fabritiusstr.15, 6174 RG Sweikhuizen, Niederlande, bestellt werden. Es kostet 60 Gulden, zahlbar per Eurocheque oder Internationaler Zahlungsanweisung. Bitte vermerken Sie den Programmnamen auf der Bestellung. Dieses Toolkit enthält weiterhin noch einige andere nützliche Programme, um die Diskettenstation VC1541 noch besser nutzen zu können.

Relative Files

(von A. Koldehoff)

Das Prinzip von relativen Dateien ist bereits hinreichend in einem Artikel der letzten Clubausgabe erläutert worden (siehe dazu HP 6,1 und 6,2 von Jan Boerrichter; Absätze: „Prinzip der relativen Dateien“ und „Datei / Record Struktur“). Ich möchte hier vielmehr auf die Kommunikation DAI → VC-1541 in Verbindung mit dem deutschen Interface-System eingehen.

Kommunikation DAI / VC-1541

Gleich zu Anfang der wichtigste Unterschied zum holländischen DAI-DOS1541 System: Es ist kein zusätzliches Monitorprogramm notwendig, um relative Dateien zu verwalten. Das deutsche Interface-System ist darüber hinaus weitgehend kompatibel zu den C-64 Befehlen (die DAI-Disk Befehle bilden eine Obermenge der C-64 Disk Befehle), sodaß durchaus auch C-64 Dateiverwaltungsprogramme, die mit relativen Dateien arbeiten, für den DAI übernommen werden können (nur leicht Änderungen notwendig) !

Die Verwaltung einer relativen Datei läuft nach folgendem Muster ab (grobe Übersicht) :

Einrichten einer relativen Datei :

1. Die Datei wird geöffnet. Dabei wird die Länge eines Records festgelegt.
2. Der letzte Record wird gekennzeichnet.
3. Die Datei wird wieder geschlossen.

Schreiben eines Records :

1. Die Datei wird geöffnet.
2. Es wird auf den zu schreibenden Record positioniert.
3. Der Record wird geschrieben.
4. Die Datei wird geschlossen.

Lesen eines Records :

1. Die Datei wird geöffnet.
2. Es wird auf den zu lesenden Record positioniert.
3. Der Record wird gelesen.
4. Die Datei wird geschlossen.

Einrichten einer relativen Datei :

Wichtig ist, daß immer nur eine relative Datei geöffnet sein kann. Wollen Sie mit zwei relativen Dateien arbeiten, so muß immer die erste geschlossen werden, bevor die zweite geöffnet wird. Zusätzlich zu der relativen Datei kann eine sequentielle Datei geöffnet werden.

☞ Zum erstmaligen Einrichten einer relativen Datei ist es sinnvoll, den letzten Record freizugeben, da dann sämtliche vor diesem Record liegende Datensätze auch freigegeben werden. Freigeben bedeutet, den Record mit dem Byte CHR\$(255) zu beschreiben. Versucht man, einen Record zu lesen, dessen Nummer über die des letzten Records der Datei liegt, so verursacht dies den Fehler „RECORD NOT PRESENT“. Beschreibt man jedoch einen Record, der über dem bisher höchsten Record

liegt, so werden gleichzeitig alle Records, die unterhalb dieses neuen Records liegen, mit CHR\$(255) beschrieben. Ein späterer Lesezugriff auf einen Record dieses Bereichs erfolgt dann fehlerlos. Das Beschreiben dieser „freigegebenen“ Records erfolgt dann wesentlich schneller, weil alle Records, die unter diesem liegen, nicht mehr freigegeben werden müssen.

Beispiel zum Einrichten einer Adreßdatei mit 1000 Records mit jeweils 55 Zeichen (für Name 20, Straße 20 und Stadt 15 Zeichen):

```
*IMPINT:LIST
100 RECLEN=55
110 RN=1000
120 HB=RN/256
130 LB=RN-HB*256
140 OPEN 1 8 3 "MAILING LIST,L,"+CHR$(RECLEN)
150 OPEN 2 8 15
160 PUT# 2;"P",3,LB,HB,0 →Byte 0 innerhalb des Records
                        ↓
                        Kanal-Nr.
170 PUT# 1;255
180 CLOSE 0 ←hiermit werden beide Kanäle auf einmal geschlossen
```

HB ≙ High Byte
LB ≙ Low Byte
RN ≙ Recordnummer

Das Freigeben der 1000 Records nimmt einige Zeit in Anspruch. So kann das Einrichten dieser Datei ca. 10 Minuten dauern.

Übertragung eines Records :

Im Prinzip unterscheidet sich die Datenübertragung nicht von der bei der sequentiellen Speicherung. Sätze werden mit PRINT#, PUT# oder MPUT# geschrieben und mit INPUT#, GET# oder MGET# wieder gelesen (ab ComDOS V2.6 kein INPUT# mehr).

Beispiel für das Schreiben eines Records :

```
200 REPT 55,SP$=" " ←erzeugt einen String mit 55 Leerzeichen (XBASIC-Befehl)
210 INPUT "Nr., Text";RN,T$:PRINT:HB=RN/256:LB=RN-HB*256
220 T$=LEFT$(T$+SP$,55) ←Text auf 55 Zeichen Länge bringen
230 OPEN 1 8 3 "MAILING LIST,L,"+CHR$(RECLEN)
240 OPEN 2 8 15
250 PUT# 2;"P",3,LB,HB,0 ←positionieren
260 PRINT# 1,T$; ←Record schreiben
270 CLOSE 0
```

Beispiel für das Lesen eines Records :

```
300 REPT 55,T$=" ":ZEIG=VARPTR(T$)
310 ANFADR=PEEK(ZEIG)+256*PEEK(ZEIG+1)+1 ←Anfang von T$

320 INPUT "Nr. ";RN:PRINT:HB=RN/256:LB=RN-HB*256
330 OPEN 1 8 3 "MAILING LIST,L,"+CHR$(RECLEN)
340 OPEN 2 8 15
350 PUT# 2;"P",3,LB,HB,0
360 MGET# 1,ANFADR 55 ←in T$ lesen
370 CLOSE 0
380 PRINT T$ ←in T$ stehen nun die Daten des Records RN
```

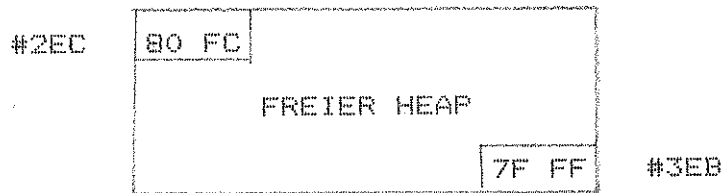
Wenn das erste Zeichen des Records gleich CHR\$(255) ist, so ist, wie oben schon erwähnt, dieser Record gemäß der Commodore Struktur leer. Dies kann z.B. durch „GET# 1;IN\$:IF ASC(IN\$)=255 GOTO leer“ überprüft werden. Danach muß allerdings neu positioniert werden, da das erste Zeichen ja schon gelesen wurde. Da der String T\$ nach dem obigen Beispiel die Information Name, Straße und Stadt enthalten soll, muß dieser nur noch in geeigneter Form „zerschnitten“ werden [z.B. NAME\$=LEFT\$(T\$,20), STR\$=MID\$(T\$,20,20) und STADT\$=RIGHT\$(T\$,15)].

DER GEBRAUCH DES HEAP

1. Der HEAP ist der Teil des RAM, der von einem Basicprogramm gebraucht wird, um Strings und Arrays zu speichern. Der HEAP beginnt auf Adresse #2EC.

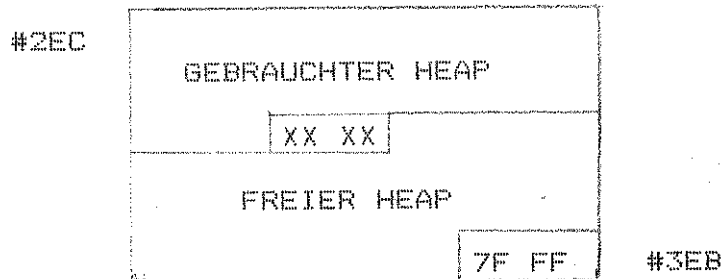
Durch einen RESET (oder Power on) werden für den HEAP 256 BYTES reserviert. Die Größe des HEAP steht in dem HEAP SIZE Pointer #29D-#29E. Von den für den HEAP reservierten Bytes braucht der HEAP selbst vierzwei um das Ende des HEAP anzugeben (und somit die maximale Größe) und zwei um die noch freien HEAP-Plätze zu berechnen.

Nach dem Einschalten des DAI sieht der HEAP folgendermaßen aus:



Der freie HEAP-Platz wird folgendermaßen berechnet: $7FFF + 80FC = 00FB$. Somit stehen 252 freie Bytes zur Verfügung.

Wenn (z.B. nach RUN eines Programmes) ein Teil des HEAP verbraucht ist, ist der Pointer zur Berechnung des freien Platzes angepaßt und hinter das letzte gebrauchte Byte hochgeschoben:



2. Der CLEAR-Befehl:

Die Größe des HEAP kann mittels des CLEAR-Befehls an die Anzahl und Größe der in dem Programm benötigten Arrays und Strings angepaßt werden.

z.B. CLEAR 1000 : 1000 Bytes werden für den HEAP reserviert. Es stehen somit 996 Bytes zur Verfügung.

Die Zahl 1000 im CLEAR-Befehl ist ein Dezimalwert!

Nach Ausführung dieses CLEAR-Befehls ist der HEAP-SIZE-Pointer #29D-#29E geändert (#3E8). Der Textbuffer - in dem das Basicprogramm steht - und die Symboltable sind auf einen höheren RAM-Bereich verschoben.

Die maximale Größe des HEAP ist 32767 Bytes (#7FFF). Es ist gut jedes Programm mit CLEARxxxx zu beginnen. Die HEAP-Größe kann schlecht innerhalb eines Programmes geändert werden.

Nur ein RESET oder POWER ON setzen den HEAP auf 256 Bytes zurück. Das Kommando NEW tut das nicht !

3. ARRAYS:

Wenn in einem Programm Arrays gebraucht werden, dann muß dafür zuerst Speicherplatz im HEAP reserviert werden. Mittels eines CLEAR-Kommandos muß dafür gesorgt werden, daß der HEAP groß genug ist.

Die Dimensionierung des Speicherplatzes für Arrays erfolgt mit dem DIM-Befehl.

Dies soll anhand des untenstehenden Programmbeispiels veranschaulicht werden.

```

10 DIM A%(2)
20 DIM B!(3)
30 DIM C$(5)
40 DIM D%(1,1)

```

Nach einem RUN dieses Programms sieht der HEAP folgendermaßen aus:

	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
02E0												00	0E	01	02
02F0	00	00	00	00	00	00	00	00	00	00	00	00	12	01	03
0300	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0310	00	0E	01	05	00	00	00	00	00	00	00	00	00	00	00
0320	00	13	02	01	01	00	00	00	00	00	00	00	00	00	00
0330	00	00	00	00	00	80	B3	00	00	00	00	00	00	00	00
....															
....															
03E0	00	00	00	00	00	00	00	00	00	7F	FF				

In der zu dem Programm gehörenden Symboltable steht:

```

51 41 52 EE 02 41 42 42 FE 02 61 43 62 12 03
51 44 52 22 03

```

(die Anfangsadresse der Symboltable steht in dem Pointer #2A1-#2A2).

Die Symboltable gibt an welche Variablen gebraucht sind und wo im HEAP der Platz für die Arrays reserviert ist:

```

AZ - 51 41 52 EE 02 - A (41) , % (51..52) #02EE
B! - 41 42 42 FE 02 - B (42) , ! (41..42) #02FE
C$ - 61 43 62 12 03 - C (43) , $ (61..62) #0312
D% - 51 44 52 22 03 - D (44) , % (51..52) #0322

```

Im HEAP ist nun der Platz von #2EC bis #334 benutzt. Auf #335-#336 steht nun der Pointer #80B3. Indem hierzu #7FFF addiert wird, erhält man den freien HEAP-Platz.

Die Reservierung der verschiedenen Variablen im HEAP ist wie folgt:

AZ(2)	-	00	0E	01	02	+ 12 Bytes	
B!(3)	-	00	12	01	03	+ 16 Bytes	
C\$(5)	-	00	0E	01	05	+ 12 Bytes	
D%(1,1)	-	00	13	02	01	01	+ 16 Bytes

↳ reservierte Bytes
 ↳ Anzahl Elemente des Array
 ↳ Dimension des Array
 ↳ gesamte Länge

3.1 Für Integer und Floatingpoint ARRAYS werden pro Element 4 Bytes reserviert. Erfolgt im Programm eine Wertzuweisung auf ein Arrayelement, so wird der Wert direkt in die zugehörigen Bytes gespeichert.

```
A%(2) - 00 0E 01 02 00 00 00 00 00 00 00 00 00 00 00
           ^-----^-----^
           A%(0)   A%(1)   A%(2)
```

eindimensionales Array mit 3 Elementen, insgesamt 14 Bytes.

Bei Bedarf kann die Adresse eines Arrayelementes z.B. mit PRINT HEX\$(VARPTR(A%(1))) abgefragt werden.

Wird im Programm A%(1) auf 10 gesetzt, so steht im Array

```
A%(2) - 00 0E 01 02 00 00 00 00 00 00 0A 00 00 00 00
```

(n.B. es wird MSB(most significant Byte)-first abgespeichert, im Gegensatz zur üblichen 8080-Darstellung)

3.2 STRING ARRAYS werden ganz anders behandelt. Beim Dimensionieren eines Stringarrays wird pro Element ein 2-Byte-Pointer reserviert:

```
C$(5) - 00 0E 01 05 00 00 00 00 00 00 00 00 00 00 00
           ^-----^-----^-----^-----^
           C$(0) C$(1) C$(2) C$(3) C$(4) C$(5)
```

eindimensionales Array mit 6 Elementen, Gesamtlänge 14 Bytes.

Die zwei pro Element reservierten Bytes werden als Adreßpointers gebraucht. Beim Dimensionieren wird hier noch nichts eingetragen (d.h. 0, ist ein Verweis auf einen leeren String, da in der Zelle 0 der Wert 0 steht). Erfolgt im Programm eine Wertzuweisung auf ein Arrayelement, so geschieht dies folgendermaßen:

C\$(1)="DAI"
ergibt die folgende Veränderung im HEAP

```
C$(5) - 00 0E 01 05 00 00 36 03 00 00 00 00 00 00 00
```

| Pointer auf den Platz im
|--- HEAP, wo der Sting abge-
| speichert ist

Der String "DAI" befindet sich nun auf der Adress #336:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0330	00	00	00	00	00	00	03	44	41	49	80	AE	00	00	00	00
							D	A	I							
							3 BYTE				Pointer angepaßt					

Für die Stringarrayelemente werden keine überflüssigen Speicherplätze reserviert, sie werden in der Reihenfolge der Wertzuweisung nacheinander in den freien Heapplätzen gespeichert. Im Array wird durch die Adreßpointer auf den String verwiesen.

Wird in einem Programm die Länge des Strings verändert, so werden alle hinter dem String verschoben und die zugehörigen Pointer upgedated.

4. STRINGS

Außer für Arrays wird der Heap auch für gewöhnliche Strings verwendet. Dies erfolgt analog zur Behandlung von Stringarrays, d.h. die Symboltable enthält statt des Wertes einen Verweis auf den String. Dies wird im folgenden verdeutlicht:

```
A$="DAI"
```

Nach dem RUN steht in der Symboltable:

```
21 41 22 ED 02 - A (41) , $ (21..22) #02ED
```

Im Heap steht dann:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
02E0													00	02	44	41
02F0	49	80	F7													

5. ANMERKUNGEN:

- Wenn mehr Heapplatz benötigt wird als durch CLEAR vereinbart wurde, erfolgt die Fehlermeldung: OUT OF STRING SPACE
- Im Prinzip werden durch ein CLEAR alle Arrayelemente auf 0 gesetzt. Trotzdem ist es zweckmäßig, am Programmbeginn alle Elemente mit einer Schleife zu initialisieren (zwecks Transportabilität auf andere Rechner).
- Wenn ein ein Array nicht dimensioniert wird, so resultiert die Fehlermeldung UNDEFINED ARRAY IN LINE xxxx .
- Während des Programmlaufes kann die Dimension eines Arrays durch den Befehl DIM geändert werden; dabei werden alle Elemente auf 0 gesetzt.

 * DAI-PC SPEICHERBELEGUNG *

Werte in Klammern sind die "Normal"-Werte

Interrupt-Vektoren: 0000-003F

0000-0007 V0
 0008-000F V1
 0010-0017 V2
 0018-001F V3
 0020-0027 V4
 0028-002F V5
 0030-0037 V6
 0037-003F V7

Aufbau der Routinen: 00 NOP
 E5 PUSH H
 2A LHLD
 XX Adresse des Vektors (62-71)
 XX
 E9 PCHL Sprung zur Routine
 00 NOP
 00 NOP

0040 Kopie von F006 (Bit-Benutzung siehe dort)
 0041, 0042 merkt beim Bank-switch PSW
 0043, 0044 ----- " ----- HL

'LOOK'-Speicher: (Anmerkung: 'LOOK' funktioniert erst,
 nachdem der Interrupt-Vektor 0
 durch 'Z2' oder 'Z3' initialisiert
 wurde)

0048, 0049 obere Grenze für LOOK
 004A, 004B untere Grenze
 004C enthält FB (enable interrupt), wenn LOOK schon
 benutzt
 004D-004F momentaner Befehl
 0051, 0052 I Adresse des momentanen Befehls
 0053, 0054 A/F PSW
 0055, 0056 B/C
 0057, 0058 D/E
 0059, 005A H/L
 005B, 005C Stackpointer
 005D, 005E Programmzähler

005F M Interrupt-Maske (Kopie von FF08)
 0060 T
 0061 G

Interrupt-Vektoren:

0062,0063	RST 0	Timer 1 'LOOK' und 'GO'-Routinen; wird erst nach 'Z2' oder 'Z3' initialisiert (EB5D)
0064,0065	RST 1	Timer 2 (C70E) Utility und Basic-Compilierung
0066,0067	RST 2	Externer Interrupt (D9E2) wird bei Stack-Overflow angesprungen
0068,0069	RST 3	Timer 3 (D755) Envelope-Interrupt
006A,006B	RST 4	'Receive buffer full' (C6C0) Rechen-Routinen
006C,006D	RST 5	'Transmit buffer empty' (C6FD) Bildschirm-Routinen
006E,006F	RST 6	Timer 4 (D578) Tastatur-Abfrage
0070,0071	RST 7	Timer 5 und 'auxiliary interrupt' (D9A9) Cursor-Blinken und 'WAIT TIME'

Für eigene Zwecke können die Vektoren 'umgeleitet' werden, wie es z.B. auch bei der 'REAL TIME CLOCK' aus dem englischen Handbuch geschieht. Nach Abarbeiten der eigenen Routine erfolgt dann ein Sprung zur 'normalen'.

Bildschirm-Variable (72-CF)

0072,0073	Cursor-Position
0074	(01) Cursor-Art wenn <>0 dann wechselt das Zeichen an der Cursor-Position mit dem Inhalt von 0075 wenn =0 dann wird das Hintergrund-Byte mit dem Inhalt von 0075 exklusiv-odiert z.B.: POKE #74,0;POKE #75,#81;COLORT B 0 10 0
0075	Cursor-Information
0076	(00) Hintergrund-Information für das nächste zu druckende Zeichen
0077	Zeichen an der Cursor-Position
0078,0079	Kontrollbyte-Adresse der momentanen Zeile
007A	Low-Byte der Adresse des letzten verwendbaren Zeichens in der momentanen Zeile
007B	Anzahl der momentan angehängten Zeile (0-3)
007C-007F	COLORT-Register COLORT a b c d: B0+a,90+b,A0+c,B0+d
0080,0081	(BFFF) Zeiger auf letzte Screen-Adresse
0082,0083	(BFEE) letzte Mode-Byte-Adresse
0084,0085	(B34F bei MODE0) letzte freie Adresse vor Screen-RAM
0086,0087	Zeiger auf das Mode-Byte der Zeile, in der das Bild im A-Modus abgeschnitten wurde (Zeile, die oben am Rand nicht mehr zu sehen ist)
0088,0089	Zeiger auf letztes COLORT-Byte
008A,008B	Zeiger auf letztes Byte für Texte
008C,008D	A-Modus: Zeiger auf letztes COLORG-Byte
008E,008F	Zeiger auf letztes Mode-Byte des gesplitteten Bereiches

0090,0091 Zeiger auf Anfangs-Adresse des Zwischenspeichers bei Umschaltung von Grafik auf A-Modus

0092,0093 A-Modus: Zeiger auf letztes COLDRG-Byte
Grafik: Zeiger auf letztes Mode-Byte des gespl. Bereiches

0094,0095 Anzahl der horizontalen Bildpunkte

0096 Anzahl der vertikalen Punkte (=Linien)

0097 Anzahl der unsichtbaren Linien im A-Modus

0098 Zahl der Bytes pro Linie im Mode

0099,009A Zeiger auf voriges letztes Grafik-Byte

009B,009C Zeiger auf voriges erstes Schrift-Byte

009D momentaner Bildschirm-Modus:
00=MODE1 01=MODE1A 02=MODE2 03=MODE2A
04=MODE3 05=MODE3A 06=MODE4 07=MODE4A
08=MODE5 09=MODE5A 0A=MODE6 0B=MODE6A
FF=MODE0 10=?????

009E-00A1 COLDRG-Register (80+a,90+b,A0+c,B0+d)

00A2,00A3 Beginn des Edit-Buffers

00A4,00A5 momentanes Ende des Edit-Buffers
(zeigt auf die zweite Null hinter dem letzten Zeichen)

00A6,00A7 obere Grenze des Edit-Buffers, die nicht überschritten werden darf; bei Erreichen der Grenze ist der Buffer voll, d.h. es werden keine Zeichen mehr angenommen

00AB Anzahl der Zeichen, die links nicht sichtbar sind

00A9,00AA Anzahl der Zeilen, die oben unsichtbar sind

00AB X-Position des Cursors

00AC,00AD Y-Position

00AE,00AF enthält die RAM-Adresse des Zeichens, das gerade gesetzt wurde, wenn die Zeilenlänge von 255 nicht überschritten wurde, enthält #AF 0

00B0,00B1 Anfang der Bildschirmzeile, in der editiert wurde

00B2,00B3 Anfang der Zeile im Edit-Buffer

00B4,00B5 wenn beide <>0, dann zeigen sie auf eine Tabulator-Tabelle, z.B. B4,B5=2EC ; 2EC.=07,14,3F,0 geht nur über Maschinenprogramme, bei 'EDIT' werden sie auf 0 gesetzt

00B4-00C3 Variable für Grafik-Berechnungen

00C4,00C5 (CA01) Sprung zur Routine, die bei Moduswechsel prüft, ob genügend freier Speicher vorhanden ist und FRE berechnet

00C6,00C7 (CA25) Sprung zur 'OUT OF SPACE FOR MODE'-Routine

Rechen-Variable:

00D0,00D1 Zeiger auf eine Sprungtabelle für Fehlermeldungen in mathematischen Berechnungen

00D2,00D3 (DDE0) Zahlen-Eingabe-Routine

00D4 Math-Chip Flag: 00=kein, 7B=Math-Chip vorhanden

00D5-00D8 hier steht immer einer der Operanden bei math. Operationen

00D9-00FF Zwischenspeicher

Basic-Variable (0100-02EB)

0100,0101	Zeiger auf Anfang der momentanen Basic-Zeile
0102,0103	Zeiger auf Adresse des momentanen Befehls
0104,0105	zeigt auf die mom. Schleifenvariable, wenn eine Schleife läuft, sonst 0000
0106	Schleifenvariablen- u. impl./expl.- Flag Bit7=0: FPT, Bit7=1: INT Bit0=0: impl., Bit0=1: expl. (Schrittweite<>1)
0107-010A	Schrittweite, wenn explizit
010B-010E	Anzahl der Schleifendurchläufe, wird vor dem Start der Schleife hier abgelegt und beim Lauf jeweils um 1 erniedrigt; aus diesem Grunde ist es unmöglich, durch Änderung der Schleifenvariablen in der Schleife die Anzahl der Durchläufe zu beeinflussen.
010F,0110	Zeiger auf Anfang der Schleife im Programm
0111,0112	Zeiger auf Anfang der Zeile, in der die Schleife beginnt
0113,0114	Stack-Pointer vor dem letzten GOSUB, 0 wenn kein GOSUB
0115	Trace-Flag; 00=TROFF, FF=TRON
0116	Step-Flag; 00=off, FF=STEP
0117	Input-Flag; 00=kein, FF=INPUT läuft
0118	Run-Flag; 00=off, FF= ein Programm läuft
0119,011A	Zeiger auf Ziel des letzten GOSUB; Anfangsadresse des zuletzt gelisteten Bereiches
011B,011C	Endadresse des zuletzt gel. Bereiches
0119-011C	letzte gewählte COLORG/COLORT-Werte
011D,011E	Stack-Pointer (bei Programm-Abbruch durch Fehler)
011F-0121	unbenutzt (?)
0122	Codier-Flag; 00=-, FF=es wird gerade eine Zeile kompiliert
0123	Offset des nächsten einzulesenden Zeichens in der momentanen DATA-Zeile
0124,0125	Zeiger auf Ende der mom. DATA-Zeile

0126 CONT=Flag; 00=-, 01=unterbrochenes Basic-Programm,
 kann mit 'CONT' oder 'STEP' wieder starten

0127,0128 momentaner Stack-Level
0129-012C Argument X des letzten RND(X)

012D-0130 momentaner Wert W der Zufallssequenz ($1 < W < 2$)
 $RND(X) = X * (W - 1)$

0131 Output-Schalter:
 00 = Ausgabe auf RS-232 und Bildschirm
 01 = nur auf Bildschirm
 02 = in Edit-Buffer; das auszugebende Zeichen
 wird an die Stelle gelegt, auf die (00A3,
 00A4) zeigt
 03 und größer = die Ausgabe-Routine springt mit
 dem Zeichen im Akkumulator nach 02DD. Dort
 steht normalerweise RET (C9). Hier kann ein
 Sprung zu einer eigenen Routine stehen, die
 beliebige Ausgaben erlaubt und auch 0131
 noch weiter aufschlüsselt (z.B. 3,4,5 etc.)
 Wird auch für die Ausgabe auf Floppy benutzt

0132,0133 Zeiger auf Anfang der Zeile, die gerade eingele-
 sen und kompiliert wird (Bildschirm oder Edit-
 Buffer)

0134 Offset des mom. Zeichens in dieser Zeile

0135 Input-Schalter
 00 = Eingabe von der Tastatur
 01 = Eingabe von einem String (z.B. bei READ)
 02 = Eingabe vom Edit-Buffer

0136 Typ der Zahl oder Variablen, die gerade compi-
 liert wird
 00 = FPT, 10 = INT, 20 = STR

0137 Operation, die gerade übersetzt wird

0138 vorige Operation

0139,013A Zeiger auf die Stelle im Eingabebuffer, in die
 die Operation eingeschrieben wird

013B,013C Position der letzten eingeschriebenen Operation

013D Auswahl der benutzten Kassetten
 Bit4: 0 = Recorder 1 aus, 1 = Rec.1 wird bei
 LOAD und SAVE eingeschaltet
 Bit5: Recorder 2 entsprechend

013E-01BD 128-Bytes Input-Buffer
 hier werden die kompilierten Basic-Zeilen
 zwischengespeichert; außerdem benutzt als
 Speicher für File-Namen

01BE,01BF diese Speicherstelle wird durch den Timer 5 (RST 7) jedesmal um 1 erniedrigt, wenn <>0 (alle 20 msec) wird bei WAIT TIME benutzt, läßt sich jedoch auch durch direktes ein'poken' verwenden

01C0 Zähler für Cursor-Blinken; zählt immer von #F auf 0, wenn 0 erreicht, dann wechselt der Cursor

01C1 Zähler für Tastaturabfrage; zählt von 2 auf 1

01C2-01CF SOUND0-Kontrollblock

1C2 bisherige Dauer der mom. Lautstärke

1C3,1C4 Zeiger auf mom. Werte in der ENVELOPE-Tabelle

1C5,1C6 Zeiger auf Anfang der benutzten ENVELOPE-Tabelle

1C7 SOUND-Lautstärke * 8

1C8 mom. Lautstärke; ergibt sich aus der SOUND-L. und dem mom. ENVELOPE-Wert

1C9 Zähler für Tremolo

1CA Endlautstärke; ergibt sich aus mom. Lautstärke und Tremolo

1CB Glissando-Ende-Flag; 0 = Ende, 2 = nicht erreicht

1CC,1CD mom. Periode (2.000.000/Frequenz)

1CE,1CF zu erreichende Periode (Glissando)

01D0-01DD SOUND1-Kontrollblock

01DE-01EB SOUND2-Kontrollblock

01EC-01F4 NOISE-Kontrollblock
gleich wie SOUND bis auf Frequenzen u. Tremolo

01F5-0234 ENVELOPE0-Tabelle
enthält abwechselnd Lautst., Dauer-1; Ende=FF kann bis in ENVELOPE1 hineinreichen, dadurch bis zu 63 Wertepaare möglich (nur durch POKE)

0235-0274 ENVELOPE1-Tabelle

0275-028E IMP-Belegungen der Variablen
00 = FPT, 10 = INT, 20 = STR
0275 = A, 028E = Z

028F durch IMP gewählter Zahlentyp

0290 Zahlentyp der mom. Operation

0291,0292 Zeiger auf Anfang der mom. DATA-Zeile

0293 Verzögerungszähler bei Hardware-random

0294 Duplikat von FD04 (siehe dort)

0295 Duplikat von FD05

0296 Input-Schalter; 0 = von Tastatur,
 <>0 = von RS-232
 wenn <>0, wird 02E0 angesprungen (DINC)

0297-029A unbenutzt (?)

029B,029C (2E0) Zeiger auf Beginn des Heap
 Der Heap läßt sich durch ändern dieser Speicher
 beliebig nach oben verlagern (*UT,>S29B,>B,*NEW),
 dadurch erhält man Platz für Maschinenprogramme,
 die auch vor NEW geschützt sind (hier liegt z.B.
 auch das DOS)

029D,029E Größe des Heap

029F,02A0 Zeiger auf Beginn des Basic-Programms

02A1,02A2 Zeiger auf Beginn der Variablen-Tabelle
 (EndeProgramm + 1)

02A3,02A4 Ende der Variablen-Tabelle + 1
 (Beginn des freien RAM)

02A5,02A6 Beginn des Video-RAM (je nach MODE)
 02A7,02A8 (EBC5 ROM-Bank 3) Zeiger auf die
 Tastatur-Tabelle
 Durch verlegen der Tabelle ins RAM lassen
 sich die einzelnen Tasten beliebig belegen
 (z.B. auch Shift-0)

02A9,02B0 letzte Tasten-Abfrage (Reihe 0 in 2A9,
 Reihe 7 in 2B0)

02AF REPEAT-Flag; Bit 5: 0 = kein,
 1 = REPT gedrückt

02B0 SHIFT-Flag; Bit 6: 0 = kein,
 1 = SHIFT gedrückt

02B1-02B8 vorige Tastatur-Abfrage

02B9 wenn <>0, wird nur die BREAK-Taste abgefragt

02BA-02BD zirkulärer Buffer für die letzten vier
 gedrückten Tasten

02BE,02BF Pos. für die nächste Eingabe in den Buffer

02C0,02C1 Position des nächsten auszulesenden Zeichens

02C2 Zähler für REPEAT

02C3 CTRL-Flag
 00 = normale Belegung
 FF = Kleinbuchstaben

02C4 BREAK-Flag
 00 = kein BREAK, FF = BREAK gedrückt
 In Basic-Programmen wird sofort unterbrochen,
 in Maschinen-Programmen mit Verzögerungszeit.

0205-02EB Floppy/Kassetten-Vektoren
 (Kopie des ROM's von D7A4-D7CA)

0205	JMP	D2B8	WOPEN	Schreib-Routinen
0208	JMP	D2F1	WBLK	
020B	JMP	D427	WCLOSE	
020E	JMP	D325	ROPEN	Lese-Routinen
02D1	JMP	D340	RBLK	
02D4	JMP	D445	RCLO	
02D7	JMP	D3A2	MBLK	
02DA	RET	0 0	RESET	
02DD	RET	0 0	DOUTC	Ausgabe hierher, wenn (#131)>2
02E0	JMP	0DB4	DINC	Eingabe von RS-232
02E3	RET	0 0	?	
02E6	24	24	TAPSL	Frequenzdaten für Kassetten-Leader
02E8	24	30	TAPSD	----- " ----- Programm
02EA	24	18	TAPST	----- " ----- Ende

02EC-BFFF freier RAM-Bereich:
 Heap (Strings und Arrays)
 Basic-Programm
 Symbol-Tabelle (= Variablen)
 freies RAM
 Bildschirm-RAM

C000-DFFF 8 KByte ROM

E000-EFFF 16 KByte ROM
 vier schaltbare Bank's zu je 4 KByte

F000-F7FF nicht benutzter Bereich, kann mit einem EPROM
 belegt werden; wenn in diesen Bereich einge-
 geschrieben wird, entsteht ein 'STACK OVERFLOW'-
 Interrupt

F800-FBFF Stack; wird auch bei Initialisierung des DOS
 verwendet

Input/Output-Adressen F900-FFFF

F900-FAFF nicht benutzt; frei für Erweiterungen

Achtung: für die Adressen FB00-FFFF sind die Adressleitungen A4 bis A7 nicht dekodiert;
d.h. z.B. FB00=FBF0

FB00-FBFF Arithmetik-Prozessor 9511

FBx0 Daten

FBx2 Kontrollwort und Status

FC00-FCFF Programmierbarer Intervall-Timer 8253

FCx0, FCx1 Zähler 0 (SOUND 0)

(auch als Zähler für Paddle-Operationen benutzt)

FCx2, FCx3 Zähler 1 (SOUND 1)

FCx4, FCx5 Zähler 2 (SOUND 2)

FCx6 Kontrollwort-Register

Vor dem Setzen der Frequenzen in die SOUND-Register muß hier nacheinander #36, #76 und #B6 eingeschrieben werden.

Kontrollwort-Format:

Bit0	0:	binärer Zähler 16 Bit
	1:	BCD-Zähler vier Ziffern
3,2,1	000 MODE0:	Interrupt bei Zählende
	001 MODE1:	progr. 'one-shot'
	x10 MODE2:	Oszillator (Modulo-Zähler)
	x11 MODE3:	Rechteck-Oszillator
	100 MODE4:	Software-getriggertes STROBE
	101 MODE5:	Hardware-grtriggerter STROBE
5,4	00	Zähler mit 'latch'
	01	lade nur MSB
	10	lade nur LSB
	11	lade erst LSB, dann MSB
7,6	00	wähle Zähler 0
	01	Zähler 1
	10	Zähler 2
	11	illegal

FD00-FDFF einzelne I/O-Adressen

FD00 (4) INPUT-PORT

Bit0: -

1: -

2: Page signal (20 msec)

3: serielle Ausgabe bereit

4: Taste an Paddle 1 (1 = geschlossen)

5: Taste an Paddle 2

6: Zufallsdaten (für RND(0))

7: Kassetten-Eingang

FD01 (5) Paddle-Trigge
 Wenn hier gelesen wird, wird der Paddle-Timer
 gestartet.

FD04 OUT (2) Bit 0-3: Lautstärke Kanal 0
 4-7: Lautstärke Kanal 1

FD05 OUT (2) Bit 0-3: Lautstärke Kanal 2
 4-7: Lautstärke NOISE

FD06 OUT (3) BIT 0: Kassetten-Eingang
 1,2: Paddle-Anwahl
 3: Paddle 'ein'
 4: Kassetten-Motor 1 (0 = ein)
 5: Kassetten-Motor 2
 7,6: ROM-Bank-Anwahl (00 bis 11)

FE00-FE0F 3 programmierbare Parallel-Ports 8255
 benutzt für DCE-Bus

FE00 IN/OUT PORT A
 FE01 IN/OUT PORT B
 FE02 IN/OUT PORT C

FE03 OUT (3) Kontroll-Wort
 Aufbau des Kontrollworts:
 Bit 7 0 = Setzen/Rücksetzen eines Bit
 in PORT C
 1 = Setzen des MODE

Setzen des MODE:
 Bit 6,5: 00 = MODE 0 für PORT A
 01 = MODE 1 "
 1x = MODE 2 "
 4: 0 = Output PORT A
 1 = Input
 3: 0 = Output PORT C, Bits 4-7
 1 = Input
 2: 0 = MODE 0 für PORT B
 1 = MODE 1
 1: 0 = Output PORT B
 1 = Input
 0: 0 = Output PORT C, Bits 0-3
 1 = Input

Setzen/Rücksetzen eines Bit in PORT C:
 Bit 6,5,4: ohne Belang
 3,2,1: Nummer des Bit (z.B. 110 = Bit 6)
 0: 1 = Setzen; 0 = Rücksetzen

FF00-FFFF Timer u. Interrupt-Controller 5501

FF00 (4) enthält das letzte von RS-232 empfangene
 Zeichen

FF01 (4) Tastatur-Eingang (Bits 0-6)
 Bit 7 ist das Signal IN7+ vom DCE-Bus und
 ist auch mit dem 20msec-Pagesignal gekoppelt

- FF02 (5) Interrupt-Register
 Bits 5,4,3: Nummer des momentanen Interrupts
 alle anderen Bits sind immer 1
- FF03 (4) Status-Register
 Bit 0: wird gesetzt bei BREAK von der
 RS-232-Schnittstelle
 1: wird gesetzt, wenn ein Zeichen
 empfangen, aber von der CPU noch
 nicht gelesen wurde
 2: gesetzt, wenn keine Daten
 empfangen werden
 3: gesetzt, wenn ein Zeichen
 empfangen wurde
 4: gesetzt, wenn der RS-232-Ausgang
 bereit ist, ein neues Zeichen zu
 verarbeiten
 5: gesetzt, wenn ein oder mehrere
 der nichtgesperrten Interrupts
 anstehen
 6: gesetzt, wenn das Start-Bit eines
 Zeichens empfangen wurde
 7: gesetzt, wenn das erste Data-Bit
 eines Zeichens empfangen wurde
- FF04 (5) Kontrollwort-Register
 Bit 0: 1 = Reset des Chips
 1: 1 = sende BREAK auf RS-232
 2: wähle Quelle für Interrupt 7
 1 = Timer 5
 0 = IN 7+ vom DCE-Bus
 3: 1 erlaubt dem Chip ein 'Interrupt
 acknowledge' von der CPU zu empfan-
 gen
 4-7: immer 0
- FF05 (3) Baudrate-Register
 Bit 0: 110 Baud
 1: 150 Baud
 2: 300 Baud
 3: 1200 Baud
 4: 2400 Baud
 5: 4800 Baud
 6: 9600 Baud (wird bei RESET gewählt)
 7: 1 = ein Stopp-Bit
 0 = zwei Stopp-Bits
- FF06 (3) RS-232-Ausgang
 In diese Speicherstelle die auszugebenden
 Zeichen einschreiben, allerdings nur, wenn
 FF03 Bit4 = 1
- FF07 (3) Tastaturabfrage-Ausgang
- FF08 (5) Interrupt-Maske
 1 = Int. erlaubt; 0 = gesperrt
 Bit 0 = Int. 0 Bit 7 = Int. 7
- FF09 (5) Timer 1 (RST 0) benutzt für LOOK

FF09	(5) Timer 1	(RST 0)	benutzt für LOOK
FF0A	(5) Timer 2	(RST 1)	unbenutzt
FF0B	(5) Timer 3	(RST 3)	benutzt für SOUND (ENVELOPE)
FF0C	(5) Timer 4	(RST 6)	benutzt für Tastatur- Abfrage
FF0D	(5) Timer 5	(RST 7)	20 msec; benutzt für Cursor-Blinken

In die Timer-Register können Werte eingeschrieben werden. Sofort danach zählt der Timer im 16msec-Rythmus rückwärts bis auf 0, danach erzeugt er einen ihm zugehörigen Int., wenn dieser nicht gesperrt ist.

FF0E, FF0F nicht benutzt

Bedeutung der Zahlen in Klammern:

- (1) Schreiben u. Lesen erlaubt
- (2) auch; kann aber vom Basic überschrieben werden
- (3) Lesen verboten
- (4) Schreiben verboten
- (5) sollten nicht angesprochen werden