Betriebssystem ES4.0 für den JU+TE-Computer

Sonderdruck zum jugend+technik-Artikel in Heft 6/1990 Seite 82ff: "Die Krönung: Vollgrafik und Farbe für den JU+TE-Computer"

mit Bestückungsplan für die Bildschirmsteuerung

Redaktion jugend+technik
Text und Zeichnungen:
Harun Scheutzow

Verlag Junge Welt GmbH
Berlin 1990

Inhaltsverzeichnis

0. Vorwort

Der JUTE-Computer mit der neuen Bildschirmsteuerung, dem Betriebssystem ES4.0 (im folgenden abgekürzt OS) und einer guten Tastatur ist ein vollwertiger Homecomputer. Ein 8-Bit-Computer hat auch im Zeitalter der 16-, 32- und 64-Bit-Rechner seine Berechtigung. da er fast alle im Heimbereich anfallenden Aufgaben lösen kann. Mit diesem Heft wollen wir die umfangreiche Beschreibung des ES4.0 sofort verfügbar machen und einen Druck als 'Fortsetzungsroman' umgehen. Die Informationen zu einem Themenkomplex wurden jeweils zusammengefaßt, um eine effektive Nutzung als Nachschlagewerk bei der Programmierung zu gewährleisten. Deshalb wird man beim erstmaligen Lesen an einigen Stellen auf zunächst unverständliche Abschnitte stoßen. Hier hilft nur Weiterlesen und wieder von vorn anfangen. Die Schaltpläne befinden sich im Anhang.

0.1. Bezug des OS

Das ES4.0 und die dafür entwickelten Programme versendet die Redaktion "jugend und technik". Dazu ist eine Kassette, ein gelöschter EPROM 2764 und Rückporto in Briefmarken (90 Pfennig) an die Redaktion zu schicken, in einer postsicheren und für den Rückversand geeigneten Verpackung. Die Anschrift: Postfach 43, Berlin, 1026. Wir programmieren in den EPROM das OS und bespielen die Kassette mit den Programmen. Die Tastaturbelegung ist im Anhang angegebenen.
Auf der Kassette ist jedes Programm zweimal hintereinander abgespeichert, um eine möglichst hohe Erfolgsquote zu garantieren. Bei Ladeproblemen sollte man zuerst prüfen, ob der Tonkopf sauber ist und ihn gegebenenfalls reinigen, z.B. mit einem fusselfreien, spiritusgetränkten Lappen. Es kann auch an einer im Vergleich zu unseren Geräten anderen Einstellung des Tonkopfes liegen. In diesem Fall stellt man die Schraube am Tonkopf während der Wiedergabe des betreffenden Programms so ein, daß die Wiedergabe mit maximaler Lautstärke erfolgt. Wir empfehlen, vorher die alte Einstellung zu markieren, damit man seine alten Aufzeichnungen später wieder laden kann.

0.2. Kassetteninhalt

-Betriebssystem ES4.0 zum Selbstprogrammieren. Es ist im KC-Format gespeichert, also mit dem alten 4K-System ladbar. (Wer einen EPROM mitschickt, erhält das OS trotzdem noch zusätzlich auf Kassette)
Die folgenden Programme sind nur mit ES4.0 ladbar.
-Monitorerweiterung KCTRANS, siehe 7.2.
-Monitorerweiterung ASDIS, siehe 6.4.
-Monitorerweiterung BITMAN, siehe 6.4.
-Steuerprogramm für S 3004, siehe 8.
-Ein Basicprogramm, das mit L geladen wird und nach dem Start mit R mitteilt, was sich noch auf dieser Kassette befindet. Das werden wahrscheinlich sein: Assembler mit Markenverarbeitung, Textverarbeitungsprogramm, Forth mit Screenarbeit. Routinen zum Rechnen mit Gleitkommazahlen und weitere Einfälle, die wir noch haben.

1. Installation des OS

Das OS ist 6 KByte lang und wird auf einen EPROM 2764 programmiert, der im Computer auf Modul 1 gesteckt wird und den Bereich %0800-1FFF belegt. Das OS belegt demzufolge im EPROM ebenfalls diesen Adreßbereich.

1.1. RAM-Speicher

Es sind mindestens 8 KByte RAM im Bereich %E000-FFFF erforderlich. Davon nutzt ES4.O %F500-FFFF für sich. Viele Programme brauchen noch RAM im Bereich %C000-DFFF und einige lohnen sich erst mit 32 KByte RAM.

1.2. Tastatur

Es lassen sich beliebige Tastaturen mit einer Matrix bis 16 * 8 verwenden. Bei mehr als vier Zeilenleitungen muß dazu natürlich der in JU+TE 5/90 vorgestellte Modul eingesetzt werden. Die Tastaturbelegung kann völlig frei programmiert werden, siehe Abschnitt 2. Damit mehrere Tasten gleichzeitig gedrückt werden können, wie es bei diesem OS notwendig ist, muß in jede Spaltenleitung eine Diode (z.B. GA104) mit der Kathode zum Tastaturdekoder eingefügt werden. Die Widerstände am V40098 sollten max. 4 kOhm betragen. Bei größeren Werten kann es Probleme mit der Erkennung einiger Tasten geben. Die bisher ungenutzte Tastaturdekoder-Spaltenleitung (pin 1 des MH 74154 bzw. pin 15 des unteren DS 8205 bei der Schaltung aus JU+TE 2/88) kann ebenfalls verwendet werden.

1.3. Kassetteninterface

Wer mit der Schaltung für das 4K-System bisher nur Ärger hatte, sollte es mit der Schaltung "Kassetteninterface" im Anhang versuchen. Sie ist einfacher, funktioniert aber mindestens genauso gut. Die Buchse ist für ein Überspielkabel beschaltet. Bei Nutzung eines Diodenkabels müssen pin 1 und pin 3 der Buchse vertauscht werden.
Die 4K-System-Interfaceschaltung muß geringfügig verändert (vereinfacht) werden, damit sie mit ES4.0 funktioniert. Die Widerstands-Kondensatorkombination (22 k, 2 * 2.7 nF, 100 k, 22 k) von P36 zu pin 1 der Buchse (Buchse für Diodenkabel) ist durch eine 20 k, 1 k Widerstandskombination zu ersetzen, wie sie im "Kassetteninterface" im Anhang vorhanden ist.

2. Tastatur

Wird eine Taste länger als 0.5 s gedrückt, dann setzt der Autorepeat ein, d.h. die Taste wird bis zum Loslassen automatisch schnell hintereinander gedrückt. Die SHIFT-Tasten (SHT) sind wie bei einer Schreibmaschine gemeinsam mit der jeweiligen anderen Taste zu bedienen. Die Tasten SHT2 und SHT3 sind Shift-Tasten, die jeweils noch eine andere Tastenbelegung (Tastaturebene) aktivieren. Die Tasten F1 bis F8 sind Funktionstasten.
Die abgedruckte Tastaturbelegung (Anhang) wird von uns in den EPROM programmiert und ist als Vorschlag für 12 * 4 Matrix-Tastaturen (TT-Tastenpulte) gedacht. Sie entspricht weitgehend der alten Tastatur.

2.1. Eigene Tastaturbelegung

Das OS benötigt die Zeilenzahl der Tastatur auf der Adresse %1B22 im EPROM. Die normalen Tasten haben die Codes %01 bis %7F und die Funktionstasten F1-F8 haben %80-87. Die SHT-Tastencodes sind der Tabelle zu entnehmen. Der Bereich %1D00-1E7F im EPROM ist für die Tastaturebenen vorgesehen. Die Tastencodes werden, beginnend mit der obersten Zeile, zeilenweise, innerhalb einer Zeile von links nach rechts, abgelegt. Die SHT-Tasten sind in der Grundebene (Tastenbelegung ohne SHT) unterzubringen. Die Kleinbuchstaben sollten ebenfalls in der Grundebene liegen, da die Tastaturabfrageroutine nach RESET so eingestellt ist, daß Kleinbuchstaben automatisch in große umgewandelt werden (Caps aktiv).

```
4-Zeilen-Tastatur:
auf %1B22: %04
max. 6 Ebenen zu je 64 Byte
Grundebene %1D00-1D3F
SHT1  %C1   %1D40-1D7F
SHT2  %C2   %1D80-1DBF
SHT3  %C3   %1DC0-1DFF
SHT4  %C4   %1E00-1E3F
SHT5  %C5   %1E40-1E7F

5- bis 8-Zeilen-Tastatur:
auf %1B22: %05 bis %08 (Zeilenzahl)
max. 3 Ebenen zu je 128 Byte
Grundebene %1D00-1D7F
SHT1 %C2   %1D80-1DFF
SHT2 %C4   %1E00-1E7F
```

2.2. Funktionstasten

Die Funktionstasten F1 bis F8 rufen jeweils ein Maschinenprogramm aus der Tastaturabfrageroutine heraus auf. Sie sollten deshalb nur betätigt werden, wenn man sicher ist, daß dieses Maschinenprogramm auch existiert. Die Funktionstasten sind nicht vom Autorepeat betroffen. Ihre Betätigung wird nach der Ausführung des Funktionstastenprogramms mit einem 0.5 s langen Piep quittiert. Die Startadressen sind %F7E0, %F7E4, ... bis %F7FC im 4 Byte-Abstand. Das reicht aus, um kurze Programme direkt bzw. einen Sprung zu einem Programm einzutragen. Die Programme dürfen den Registerpointer %FD nicht verändern. Eine Standardbelegung für die Funktionstasten ist bereits im ES4.0 im EPROM gespeichert. Diese kann man vom MON aus mit M1EB0 F7E0 0020 in den RAM kopieren und enthält folgende Funktionen:

F1 Tastaturpiep umschalten (an/aus)

F2 Caps umschalten (klein-)groß Buchstabenwandlung an/aus bei Tastatureingabe)

F3 Drucker an

F4 Drucker aus

F5 Bildschirm wechseln. Danach ist der auf HOM-Position stehende Kursor bis zum nächsten Tastendruck unsichtbar. Der Wechsel erfolgt zwischen zwei Textbildschirmen durch Speicherverschiebung.

F6 Tastencodeverschiebung umschalten (an/aus). Diese ist

normalerweise aus und man erhält die normalen Zeichen von der Tastatur. Ist sie eingeschaltet, dann wird zum ASCII-Kode noch %80 addiert und man erhält, solange der Zeichengenerator nicht erweitert ist, sehr merkwürdige Zeichen, die nicht in Basicprogramme eingebaut werden dürfen.
F7 und *F8* sind nicht belegt.

3. Computer einschalten

Nach dem Einschalten oder RESET muß auf dem gelöschten Bildschirm die Meldung "Willkommen am JTC : ES4.0 by SWB 1990" erscheinen. Darunter steht noch "Edi" und etwas tiefer der Kursor, der jeweils das Zeichen unter ihm invertiert. Der Computer befindet sich also im Basic-Editor, abgekürzt EDI. Außerdem gibt es noch den Maschinenmonitor MON. Ein Programm, das nie direkt in Erscheinung tritt, aber den Bedienkomfort ganz wesentlich bestimmt, ist der FSE.

4. FSE

Der FSE (Full Screen Editor) wird von den anderen Programmen genutzt, um den Dialog mit dem Bediener zu führen. Über den FSE geben die Programme Texte auf den Bildschirm aus. Die Eingaben über die Tastatur werden ebenfalls durch den FSE verarbeitet. Bei einem Full Screen Editor (deutsch: Ganzbildschirmeditor) kann man mit dem Kursor auf jede beliebige Stelle des Bildschirms fahren und dort schreiben bzw. korrigieren. Dabei gibt es keine Einschränkungen. Erst wenn man die RET-Taste drückt, wird die Zeile verarbeitet, in der der Kursor stand. Man kann also mehrmals mit dem Kursor auf eine Zeile fahren und RET drücken, jedesmal wird die Zeile wieder verarbeitet.

4.1. Betriebsarten

Der FSE kennt zwei Betriebsarten, den 40- und den 80-Zeichenmodus. Im 40-Zeichenmodus hat man 24 Zeilen mit je 40 Zeichen auf dem Bildschirm, so wie man es auch sieht. Im 80-Zeichenmodus gibt es 12 Zeilen zu je 80 Zeichen, wobei sich die Darstellung auf dem Bildschirm nicht ändert, aber intern werden immer zwei Bildschirmzeilen als zusammengehörig betrachtet (erste und zweite, dritte und vierte Zeile usw.) Eine Bildschirmzeile (immer 40 Zeichen lang) wird als physische Zeile bezeichnet. Eine logische Zeile (künftig nur als Zeile bezeichnet) enthält beim 40-Zeichenmodus eine physische Zeile, aber im 80-Zeichenmodus zwei physische Zeilen. Diese Unterscheidung ist bei den Editierfunktionen von Bedeutung.

4.2. Editierfunktionen

Als erstes steht die Abkürzung, die auch auf der Tastaturbelegung zu finden ist. Dann folgt die ausgeschriebene englische Bezeichnung in Klammern und die Funktionsbeschreibung.
Die *Pfeiltasten* bewegen den Kursor um eine Position nach links/rechts oder eine physische Zeile nach oben/unten.
DBS (delete back space) löscht das Zeichen links vom Kursor und bewegt den Kursor nach links.

DEL (delete) löscht das Zeichen unter dem Kursor und rückt den rechts stehenden Zeilenrest heran.
INS (insert) fügt an der Kursorposition ein Leerzeichen ein und rückt dabei den Rest nach rechts.
SOL (start of line) bewegt den Kursor auf die erste Position der Zeile, in der der Kursor stand.
HOM (home) bewegt den Kursor in die linke obere Bildschirmecke.
CLS (clear screen) löscht den Bildschirm und Kursor HOM.
LDE (line delete) löscht die Zeile unter dem Kursor und rückt die anderen Zeilen von unten heran sowie Kursor SOL.
LIN (line insert) fügt unter dem Kursor eine Leerzeile ein und rückt die anderen Zeilen nach unten sowie Kursor SOL.
RET (return) schließt eine Eingabe ab und bewegt den Kursor eine Zeile nach unten.
ESC (escape) ist ein Spezialzeichen. das die unbedingte Darstellung des als nächstes eingegebenen Zeichens auf dem Bildschirm bewirkt. Beispiel: CLS löscht das Bild, aber ESC und danach CLS zeichnet ein Quadrat auf das Bild als Symbol für einen sauberen Bildschirm. ESC A schreibt natürlich nur ein A.

4.3. Scrollen

Erfolgt in der untersten Zeile ein RET oder wird die letzte Position (rechts unten) des Bildschirms beschrieben, dann wird das Bild normalerweise um vier physische Zeilen nach oben gerollt. Wenn Bit 4 von Register %55 null ist wird nicht gescrollt, und der Kursor springt in die obere linke Ecke. Diese Betriebsart hat aber einen Fehler: Bei einer Eingabe in der untersten Zeile liefert die Routine CHARIN völligen Unsinn zurück.

5. EDI und Basic

Der EDI dient zur Arbeit mit dem Basic des EMR und nutzt den 80-Zeichenmodus des FSE. Ein Kommando wird immer mit der RET-Taste abgeschlossen. Es gibt keine unterschiedlichen Kursordarstellungen mehr. Die Eingabe einer Basiczeile wird ebenfalls mit RET abgeschlossen, nachdem die Zeile vollständig notiert wurde.

5.1. Kommandos des EDI

N löscht das Basic-Programm
L lädt ein Programm von Kassette, danach erscheint eine 0 für fehlerfrei oder 255 für Ladefehler.
S speichert ein Programm auf Kassette, danach wird eine 0 ausgegeben.
?X druckt den Wert der Variablen X aus
M ruft den MON auf
E listet das Programm, beginnend mit der ersten Zeile, zeilenweise auf, zur Fortsetzung nach jeder Zeile Leertaste drücken, sonst eine andere Taste
E125 beginnt mit dem Auflisten bei Zeile 125, siehe E
R startet das Basicprogramm
C setzt das Programm nach STOP-Befehl fort.
Wenn ein Basisprogramm läuft, wird der 40-Zeichenmodus

verwendet.
10 oder eine andere ein- bis fünfstellige Zahl wird als Zeilennummer angesehen. Folgen ihr keine weiteren Zeichen, so wird eine eventuell vorhanden Zeile mit dieser Nummer gelöscht, sonst wird diese Zeile in das Programm eingefügt. Es kann jede auf dem Bild stehende Basiczeile, z.B. durch den E-Befehl aufgelistet, erneut mit RET abgeschlossen werden, und sie wird dann ins Programm eingefügt.
Der EDI arbeitet korrekt mit allen Zeilennummern, also auch mit solchen, die im Lowbyte ein %00 oder %0D haben.
Durch die Ausführung der Kommandos L oder S werden die Basic-Variablen A, B, C verändert. Nach den Kommandos L, S, R, C, M und N meldet sich der EDI jeweils mit "Edi".

5.2. Basicbefehle

Für die Basic-Befehle werden keine Schlüsselwörter mehr verwendet, sondern nur Buchstaben. Damit kommt man aber nach kurzer Gewöhnung gut zurecht. Hinter dem Buchstaben (also Basic-Befehl) darf kein Leerzeichen stehen.

Schlüsselwort (alt)	Buchstabe (neu)
WAIT	W
END	E
RETURN	R
STOP	T
INPUT	I
PROC	O
PRINT	P
PTH	H
G0SUB	S
GOTO	G
LET	L
CALL	C
REM	M
TOFF	/
TRAP...TO...	!...,...
IF...THEN...	F...;...
ELSE	>;

Als Fehlermeldungen werden ein O : Ende mit END, ein S : Ende mit STOP oder ein E : Error ausgegeben, jeweils gefolgt von einer Zahl, dem in JU+TE 1/88, S.74 erläuterten Fehlerkode.

5.3. INPUT-Besonderheit

Bei der Verwendung des Basic-Befehls INPUT (Buchstabe I) ist folgendes zu beachten: Die einzugebende Zahl kann mit allen Funktionen des FSE editiert werden; bevor die Eingabe mit RET abgeschlossen wird. Die Zahl muß aber auf der ersten Position der Zeile (ganz links) beginnen und nach der Zahl dürfen in derselben Zeile nur noch Leerzeichen stehen. Dafür kann man notfalls "von Hand" durch Wegfressen unerwünschter Zeichen mit DEL oder DBS sorgen. Programmbeispiel:

I"X-Wert="X	falsch, da vor der Zahl etwas steht
P"X-Wert:";IX	richtig
HX,;OPTC[6];IX	richtig: Der Kursor steht auf dem Prozentzeichen

der Hexzahl. Wird nur RET gedrückt, dann wird die Zahl gleich als neuer Eingabewert genommen. Das geht nicht so

einfach für Dezimalzahlen, da diese mit einem Leerzeichen am Anfang gedruckt werden, wenn sie kein negatives Vorzeichen haben. Siehe auch 5.6.

5.4. Programmanpassungen

Am günstigsten ist natürlich die Kenntnis der Eigenschaften beider Betriebssysteme. Dann kann man alles erfolgreich anpassen. Für weniger Erfahrene möchten wir einige häufige Änderungen aufführen.
Basicprogramme
Eine Abfrage auf Shift+Enter (Kode %7F) ist durch Betätigen der ß-Taste zu befriedigen. Zugriffe auf das alte Kursorregister %5B oder den alten ASCII-Textspeicher %F000-FD7F sind durch die Routine SCRFUN zu ersetzen. Die Kursortasten haben jetzt andere Kodes, die Auflistung erfolgt als alt:neu
%0A:%04 %1A:%03 %0B:%01 %1B:%02
Weitere Änderungen betreffen folgende oft auftretende Befehlsfolgen: (x ist irgendeine Variable)

alt	neu
CALL%8DD	OPTC[12]
CALL%C56	C%81B
CALL%824;LET x=GETR[%5A]	C%17F4;Lx=GETR[%5A],OPTC[x]
CALL%C1D	C%17F4
LET x=GTC	C%17F4;Lx=GETR[%5A]=OPTC[x]

Wenn mit GTC nur auf irgendeinen Tastendruck gewartet wurde, ist keine Änderung erforderlich, aber anstelle einer beliebigen Taste die RET-Taste zu drücken. Wurde GTC verwendet, um eine komplette Zeile einschließlich des %0D-Kodes zu lesen, ist auch nichts zu ändern.
Im Zusammenhang mit GTC oder INPUT siehe auch 5.3. und 5.6. Das Basicprogramm des EPROMERs ist entsprechend der Basic-Hinweise abzuändern, für den Maschinenteil gilt folgende Änderung:
,E1DB 2B ,E1E3 0E E0 ,E1E7 1E ,E1EB 13
,E211 15 ,E215 18
%17F4 BWKEY arbeitet genau wie WKEY (Beschreibung bei 13.), liefert das Zeichen aber in Register %5A

5.5. Tips

Nachdem ein Basicprogramm mit L geladen wurde, kann mit ?B seine Länge ermittelt werden.
Für die Verwaltung mehrerer Basicprogramme ist kein RAM-MANAGER mehr erforderlich. Man nutzt den MON, um z.B. die neue Anfangsadresse %E480 einzugeben: .

M	vom EDI in den MON
!06E4	Anfangsadresse H-Byte
!0780	Anfangsadresse L-Byte
Q	zurück zum EDI

Die eingetragene Anfangsadresse bleibt erhalten bis eine neue eingetragen oder RESET betätigt wird.

5.6. Simulationsmodus

Ist Bit 2 von Register %55 null, dann arbeitet die Routine CHARIN nicht wie unter 13. beschrieben, sondern wie die beim 4K-System bei %0815 beginnende Routine: Jeder Tastendruck

wird auf dem Bildschirm ausgeführt und sofort auch an den CHARIN-Aufrufer weitergegeben. Für Basicprogramme hat dies folgende Konsequenzen: Bei Eingaben mit INPUT dürfen vor und nach der Zahl wieder beliebige Zeichen stehen. Eine Korrektur der Eingabe mit den Editierfunktionen des FSE ist nicht möglich (sie wird zwar auf dem Bild korrekt ausgeführt, aber vom Basic nicht verstanden). Die Funktion GTC arbeitet wie beim 4K-System. Dieser Modus sollte nach Möglichkeit nicht genutzt werden. Beim Ein- und Ausschalten muß man an die anderen Bits von %55 denken, die ebenfalls Funktionen haben und nicht geändert werden sollten, falls man sie nicht definitiv auf einen bestimmten Wert setzen möchte. Endet ein Basicprogramm mit eingeschaltetem Simulationsmodus oder wird der MON so aufgerufen, dann hilft nur noch die Reset-Taste.

6. MON

Der MON ermöglicht die Nutzung und Bearbeitung von Maschinen-programmen. Er verwendet nur Hexadezimalzahlen, die ohne % Zeichen geschrieben werden. Dabei müssen A-F große Buchstaben sein.

6.1. Befehle

Ein Befehl wird durch nur ein Zeichen dargestellt.
Erklärung der in der Beschreibung verwendeten "Variablen":

aaaa	2 Byte Hexzahl (vierstellig)
bc	1 Byte Hexzahl (zweistellig)
d	ASCII-Zeichen

Falls mehrmals aaaa auftaucht, dann steht es im Befehl in der gleichen Reihenfolge wie in der Erklärung.
Die in der Erklärung mit "spc" gekennzeichneten Befehle warten nach Ausgabe einer Zeile auf einen Tastendruck. Wird die Leertaste gedrückt, dann wird die nächste Zeile ausgegeben. Jede andere Taste beendet den Befehl und wird gleich auf den Bildschirm ausgegeben.
Mit "mon" gekennzeichneten Befehle drucken nach ihrer Ausführung "Mon" aus.

Haaaa Hexanzeige 8 aufeinanderfolgender Speicherstellen, spc

,aaaa bc bc bc bc bc bc bc bc Hexschreiben des Speichers, auch bei weniger als 8 Byte bc werden Daten eingeschrieben, aber keine neue Adresse ausgegeben

Aaaaa ASCII-Anzeige von 16 aufeinanderfolgenden Speicherstellen, spc

;aaaa ddddddddddddddd ASCII-Schreiben, es werden die Zeichen bis einschließlich eines RET-Zeichens, aber maximal 16, in den Speicher geschrieben. Wenn 16 Zeichen geschrieben wurden und auch das 16. kein RET-Zeichen war, dann wird die nächste Adresse automatisch ausgegeben.

Q Rückkehr zum aufrufenden Programm (meist EDI)

?aaaa aaaa gibt aaaa plus aaaa.und aaaa minus aaaa aus

Saaaa aaaa speichert ab Adresse aaaa auf Kassette aaaa Bytes, gibt danach 00 als in Ordnung-Meldung aus, mon
Im Unterschied zum 4K-Betriebssystem ist als zweite Zahl nicht die Endadresse, sondern die Anzahl der abzuspeichernden Bytes anzugeben.

Laaaa lädt von Kassette in Speicher ab Adresse aaaa, druckt dann aaaa bc aus, aaaa ist geladene Byteanzahl und bc Fehlermeldung (00: fehlerfrei), mon

Maaaa aaaa aaaa verschiebt von Adresse aaaa nach aaaa aaaa Bytes im Speicher, auch bei Überschneidung des Quell- und Zielbereichs erfolgt immer korrekte Verschiebung, also es wird automatisch festgestellt, ob das erste oder das letzte Byte zuerst verschoben werden muß, mon

Jaaaa startet Maschinenprogramm ab Adresse aaaa mit CALL, das Programm kann mit RET zum MON zurückkehren, das während des Programmlaufes verwendete Nutzer-Flagregister ist sonst in Register %16 gespeichert und der Nutzer-Registerpointer in %17, mon

Faaaa aaaa bc füllt Speicher ab Adresse aaaa mit aaaa Bytes bc, mon

#eeeee gibt die vorzeichenlose Dezimalzahl eeeee (O bis 65635) umgewandelt in hex aus

%aaaa gibt die Hexzahl dezimal aus

Rbc zeigt das Register bc an, spc

!bcbc Register bc wird mit Byte bc beschrieben, wenn Byte bc korrekt war (also eine Hexzahl), dann wird nächste Registeradresse ausgegeben. (zwischen den beiden bc steht wirklich kein Leerzeichen)

Anstelle des hier in der Befehlsbeschreibung zwischen zwei Werten verwendeten Leerzeichens kann auch jedes beliebige andere Zeichen stehen. Darunter leidet aber die Übersichtlichkeit.

Wenn die eingegebenen Zahlen bei den Befehlen nicht korrekt sind, dann wird der Befehl nicht ausgeführt, es erfolgt aber keine Fehlermeldung.

6.2. Routinen des MON

Die im MON vorhandenen Routinen werden hier erklärt, um ihre Nutzung in eigenen Programmen und insbesondere Monitorerweiterungen zu ermöglichen.
Vom MON benutzte Register

%10-12 Hilfsregister
%16 Flagregister des Nutzers
%17 Registerpointer des Nutzers
%18/19 Zahl zum Ausdrucken
%1A/1B Zeiger auf das bearbeitete Byte im Speicher
%1C/1D Ergebnis der ASCII->Zahl Wandlung
%1E/1F Zeiger auf das nächste Byte im ASCII-Zeilenpuffer

Routinen
In all diesen Routinen wird vorausgesetzt, daß der Registerpointer auf %10 steht

%0AF7 PMON druckt "Mon" und einen Returncode aus. Wenn ein Befehl nach erfolgreicher Ausführung "Mon" melden soll, dann ist er statt mit RET mit JP %0AF7 zu beenden.

%0C69 HTA16 druckt die 16 Bit aus %18/19 als 4-Ziffer-Hexzahl aus, %18/19 zerstört

%0C72 HTA8 druckt die 8 Bit aus %19 als 2-Ziffer-Hexzahl aus, %19 zerstört

%0C7B HTA4 druckt die niederen 4 Bit aus %19 als Hexziffer aus, %19 zerstört

%0C8D PRRET druckt einen Return-Kode aus (%0D)

%0C91 RWCONT druckt einen Return-Kode aus und wartet dann auf Tastendruck, wenn die Leertaste gedrückt wurde ist das Z-Flag gesetzt

%0C9B PCAS druckt das Zeichen aus %15 aus, dann die 16 Bit aus %1A/1B (%18/19 zerstört) als Hexzahl und noch ein Leerzeichen

%0CA9 ADRE wandelt die 4 Ziffern ab Adresse (%1E/1F) nach %1C/1D und %lA/1B, bei Wandlungsfehler wird POP POP RET ausgeführt, sonst Rückkehr mit um fünf erhöhtem %1F

Fehlermeldung von den folgenden 4 Routinen: C-Flag gesetzt, sonst gelöscht
Die Erhöhung von %1F stimmt nur bei gelöschtem C-Flag.

%0CB8 ATH4 wandelt die Ziffer von (%1E/1F) in niedere 4 Bit von %1D, %1F um eins erhöht

%0CD5 ATH16 wandelt 4 Ziffern ab (%1E/1F) nach %1C/1D, %12 zerstört, %1F um vier erhöht

%OCDC ATH8 wandelt 2 Ziffern ab (%1E/1F) nach %1D, %12 zerstört, %1F um zwei erhöht

%0A52 DAXTH16 wandelt eine 1 bis 5 Ziffern lange Dezimalzahl (0 bis 65535) ab (%1E/1F) nach %14/15, %12/13 und %1C/1D zerstört, %1F um Anzahl der gewandelten Ziffern erhöht

Ab %0C39 steht eine 48 Byte lange Tabelle, die max. 16 Befehlseinträge umfaßt. Ein Eintrag ist noch frei.
Aufbau der Befehlseinträge: 1.Byte ASCII-Befehlszeichen, 2./3.Byte Startadresse.

Alle Befehle enden mit RET oder einem Sprung nach PMON. Beim Aufruf der Befehle zeigt (%1E/1F) auf das Byte nach dem Befehlsbuchstaben im ASCII-Zeilenpuffer. In den Befehlen wird vorausgesetzt, daß der Registerpointer auf %10 steht.

6.3. MON-Erweiterungen

Der Befehlssatz das MON kann durch Programme erweitert werden, die im RAM stehen. Wird das Befehlszeichen in der Tabelle im EPROM nicht gefunden, dann sucht der MON den Speicher %8000-FFFF nach folgender Datenkonstruktion ab, die auf einer %XXX0 Adresse beginnen muß.
0.Byte : Befehlszeichen
1.Byte : %95
2.Byte : %95
3.Byte : %95
4/5.Byte : Byteanzahl bei der Prüfsummenberechnung (H/L)
6/7.Byte : Ergänzungswert für die Prüfsumme bis %0000
8.Byte : hier beginnt die Prüfsummenberechnung und hier wird mit CALL hingesprungen, wenn alles übereinstimmte (Beginn des Befehls).
Die Prüfsumme wird als 16-Bit-Summe aller Byte gebildet, Überträge über 16 Bit werden nicht berücksichtigt. Zu der Prüfsumme wird noch der Ergänzungswert addiert und wenn das 0000 ergibt, der Befehl als korrekt angesehen. Durch die Prüfsumme kann man sich fast 100%ig darauf verlassen, daß dieser Befehl noch nicht durch irgendwelche Programmabstürze defekt ist (mit menschlicher Intelligenz kann man den Befehl trotz richtiger Summe zerstören, ein abgestürztes Programm besitzt keine solche).
Einigen Erweiterungsbefehlen ist es egal, ab welcher %XXX0 Adresse sie im Speicher stehen, da sie verschiebbar sind. Anderen Befehlen (nichtverschiebbaren) ist das nicht egal. Sie könnten anhand des in %1C/1D als Nebenprodukt übergebenen Zeigers auf das 8.Byte bei einer falschen Adresse ihre Arbeit sofort mit RET beenden oder sich darauf einstellen.

6.4. MON-Erweiterungsbefehle

Programm BITMAN
Die Befehle B und < sind nützlich zur Erstellung von Zeichengeneratoren und laufen ab jeder %XXX0 Adresse im Bereich %8000-FFFF. Sie können z.B. mit LC000 geladen werden und sind 0060 Byte lang.

Baaaa listet Speicherinhalt binär auf, ein 0-Bit ist ein . (Punkt) und ein 1-Bit ein * (Mal). Das Bit 7 steht links. spc

*<aaaa. *.*.*.*.* beschreibt den Speicher binär, siehe B, gibt die nächste Adresse aus, wenn alle 8 Bit korrekt [also . oder *) sind.

Programm ASDIS

Die Befehle D und (Punkt) laufen nur ab %C100, müssen also mit LC100 geladen werden und sind 0900 Byte lang. Sie sind der Monitordisassembler/assembler. Weil keine Verarbeitung von Marken möglich ist, ist der Assembler insbesondere zum Schreiben von kurzen Routinen geeignet.

Daaaa disassembliert den Maschinencode ab aaaa und gibt ihn in symbolischer Darstellung aus. Bei Relativsprüngen (JR... und DJNZ...) wird als Ziel die Absolutadresse (das eigentliche Sprungziel) angegeben. Die Arbeitsregister werden anstelle der Dezimalzahlen 0-15 durch Hexziffern 0-F bezeichnet (z.B. nicht r12 sondern RC). spc

.aaaa befehl assembliert den Befehl befehl auf Adresse aaaa. War der Befehl korrekt, dann wird die nächste Adresse ausgegeben. Die Korrektheit wird hauptsächlich anhand der Daten erkannt. Das Komma wird nicht überprüft, so daß dort auch ein anderes Zeichen stehen kann. Ist bei Relativsprüngen die Sprungweite überschritten, so werden sie als nicht richtig erkannt. Die Befehle müssen genau in dem durch Daaaa gelieferten Format geschrieben werden.

Ein kleines Assemblerprogramm, das 192mal ein A auf den Bildschirm ausgibt, als Beispiel für korrekte Befehle:

```
.E400 PUSH %FD
.E402 SRP #%20
.E404 LD R0,#%C0
.E406 LD %15,#%41
.E409 CALL %0818
.E40C DJNZ R0,%E406
.E40E POP %FD
.E410 RET
```

7. Kassetteninterface

Das Aufzeichnungsverfahren wurde etwas verändert und ist jetzt doppelt so schnell. Das Bild wird nicht abgeschaltet. Wenn beim Laden ein Fehler auftritt, fängt der Computer an zu piepen. Dann die Kassette bis etwas vor die Fehlerstelle zurückspulen und solange die Taste C drücken, bis das Piepen aufhört. Sollte dies auch nach mehreren Versuchen nicht der Fall sein, ist der Fehler nicht korrigierbar und mit Taste B kann das Laden abgebrochen werden. Programmnamen oder Startadressen beim Abspeichern gibt es nicht mehr.
Die alten Aufzeichnungen im KC-Format können nur mit einem extra Programm geladen werden (siehe 7.2.).

7.1. Aufzeichnungsverfahren

Fileaufbau: 5s Vorton, Blöcke mit Pause von 40 ms
Blocksendung: Vorimpuls (eine Phase mit 0.5 ms), %85 Bytes
Blockaufbau: (Adresse im Kassettenpuffer als erste Zahl)

%F57B	Blocknummer %00 bis %FF, evt. wieder von vorn
%F57C	Kopie von %F57B
%F57D	Prüfsumme über Formatbyte und Datenbytes (Byte-Summe mit dazuaddierten Carrys)
%F57E	Kopie von %F57D

%F57F Formatbyte: %FA: weniger als 128 Datenbyte. Anzahl der gültigen Datenbyte auf %F5FF
%FC: 128 Datenbyte
%FE: End of File Block, ohne sinnvolle Datenbytes
%F580-F5FF Datenbytes
Byteaufbau: Bit 7 6 5 4 3 2 1 0
Frequenzen:
Vorton und in den Pausen: 5 kHz
Bitsendung: eine Schwingung mit 2 kHz für 1-Bit, mit 4 kHz für 0-Bit

7.2. KC-Laderoutinen

Mit dem Programm KCTRANS kann man Programme im KC-Kassettenformat laden und speichern. Es ist eine Befehlserweiterung für den MON, wird mit LF100 geladen und stellt die Befehle G und P bereit. Achtung! Wenn man das Programm mal ab irgendeiner anderen Adresse geladen hat, sollte man es dort wieder löschen (mit dem F-Befehl), da ansonsten die Befehle P und G nicht funktionieren, auch wenn das Programm noch ab der richtigen Adresse (F100) geladen wird (mit Batteriestütze bleibt ja alles erhalten).
Bedienung:
Gaaaa Laden eines Programms im KC-Format von Kassette in den Speicher ab Adresse aaaa. Meldung wie bei Befehl L. Danach kann das Programm mit dem S-Befehl abgespeichert werden. Ab %F580 steht im Speicher der Name des Programms (Anzeigen mit AF580) und auf %F591/2 steht die alte Startadresse im L/H-Format (L-Byte, dann H-Byte). Diese Werte stehen im Kassettenpuffer des ES4.0, werden also beim nächsten S- oder L-Befehl überschrieben.
Paaaa aaaa name Speichern eines Programms im KC-Format ab Adresse aaaa mit aaaa Byte Länge auf Kassette und Namen name.

Beide Befehle verändern die Register %40-4F. Die Bedienung bei Ladefehlern ist identisch zum normalen Laden mit einer Ausnahme: Man darf nach einem Ladefehler nicht gleich den letzten Block laden lassen, da dann trotz Fehler das Laden beendet wird.

Im ES4.0 wird die Startadresse eines Programms beim Abspeichern nicht(!) in diesem vermerkt (siehe MON-Befehl S, dort gibt es keine Startadressenangabe). Beim MON-Befehl L muß man immer eine Ladeadresse angeben. Diese sollte auf der Kassette notiert werden.

8. Druckeranschluß

In diesem OS gibt es eine sehr einfache Möglichkeit für den Druckeranschluß. Der Drucker druckt jedes auf den Bildschirm ausgegebene Zeichen (außer Tastatureingaben) ebenfalls aus. Wer eine Schreibmaschine S 3004 verwendet, kann das alte Modul (JU+TE 5/89) benutzen. Zusätzlich muß mit LF512 das Druckprogramm für die S 3004 geladen werden.

8.1. Centronics-Interface

Der Anschluß eines Druckers mit Centronics-Interface (8-Bit-Parallelinterface) erfolgt über die Schaltung "Centronicsinterface". Diese arbeitet etwas ungewöhnlich: Um Leitungen und Portbits zu sparen, werden die Daten vom Computer seriell ausgegeben. Im Schieberegister DL 299 werden jeweils die 8 Bit eines Bytes gesammelt und gleichzeitig an den Drucker übergeben. ES4.0 besitzt bereits eine Routine zur Ausgabe über diese Schnittstelle, die durch die Eintragung eines Sprunges mit dem MON: ,F512 8D 17 A6 zur Ausgabe genutzt wird. Ein am Drucker evtl. vorhandener Schalter muß so eingestellt sein, daß das Zeichen %0D (Carriage Return) auch einen Zeilenvorschub auslöst.

8.2. Eigene Druckerroutine

ES4.0 ruft jedesmal, wenn die Druckfunktion angeschaltet ist und ein Zeichen auf den Bildschirm ausgegeben werden soll, mit CALL %F512 die bei %F512 liegende selbstdefinierte Routine auf und übergibt ihr in %15 das zu druckende Zeichen (Register %15 und Registerpointer %FD nicht ändern). Die Routine darf die Register %60-66 nutzen. Der RAM %F512 bis %F579 ist für die Druckerroutine reserviert.

Die folgende kleine Routine druckt nichts, sondern sammelt alle Zeichen, die auf den Bildschirm ausgegeben werden, im Speicher. Die Anfangsadresse, ab der die Zeichen abgelegt werden sollen, ist auf %F530/F531 einzutragen. Das Sammeln ist nützlich, wenn man die Ausgaben irgendeines Programms später in einen Text einfügen möchte, ohne sie extra wieder einzugeben.

```
,F512 70 FD 31 60 0C F5 1C 30
,F51A 2C 64 83 20 83 20 2C 15
,F522 93 24 1C 30 2C 64 93 20
,F52A 93 20 50 FD AF FF FF FF
```

8.3. Drucken

Nun ist das Drucken z.B. eines Basicprogramms kein Problem mehr: Die Funktionstasten F3 und F4 müssen belegt sein, ab %F512 muß das Druckprogramm stehen. Dann Druckfunktion an mit F3, Programm auflisten, und Druckfunktion mit F4 aus. Auf diese Weise können die Textausgaben beliebiger Programme ausgedruckt werden.

9. Zeichengenerator (ZG)

Im EPROM befindet sich ab %1000 ein 1 KByte langer ZG für die ASCII-Kodes %00 bis %7F. Für jedes Zeichen stehen 8 * 8 Bit = 8 Byte. Das H-Byte der Anfangsadresse steht in Register %67. Ein eigener ZG kann prinzipiell auf jeder %XX00 Adresse beginnen. Dazu ist der jeweilige Wert für XX in %67 einzutragen. Beim Einschalten eines anderen ZG ändern sich wegen der Grafikdarstellung die auf dem Bild schon vorhandenen Zeichen nicht, aber die neugedruckten stammen aus dem neuen ZG. Die Kodes %80-%FF sind im EPROM-ZG nicht belegt. In einem eigenen können dort noch Zeichen stehen, so das dieser bis zu 2 KByte lang werden kann. Das seltsame Zeichen mit dem Kode %00 ist noch ein Programmstück (nicht ändern im EPROM!).

In Verbindung mit einem eigenen ZG werden auch die Zeichenkodes %81-FF interessant, die die Tastatur nach der Umschaltung mit F6 liefern kann (nicht im Basic nutzen!).

10. Speicherbelegung

Hier und bei der Registerbelegung bedeutet:
in Klammern: nach Möglichkeit keine direkten Zugriffe
OS: vom ES4.0 in einigen Routinen genutzt
OSD: vom ES4.0 ständig genutzt

%0000-07FF	interner ROM des U883
%0800-1FFF	Betriebssystem ES4.0 auf EPROM 2764
%2000-3FFF	frei
%4000-5FFF	Video-RAM-bänke
%6000-63FF	Steuerregister für Videorambankauswahl
%6400-7BFF	frei (für IO-Geräte, Ramdisk, ...)
%7000-7FFF	Tastaturabfrage
%8000-	RAM-Beginn bei 32 KByte Vollausbau
%C000-	RAM-Beginn bei 16 KByte
%E000-	RAM-Beginn bei 8 KByte
bis F4FF	RAM, frei zur Nutzung
%F500-F511	IRQ-Sprünge
%F512-F579	RAM für Druckroutine, siehe 8.
(%F57B-F5FF	Kassettenpuffer)
%F600-F6FF	Stack
%F700-F74F	Zeilenspeicher bei Rückkehr aus Routine CHARIN
(%F750-F76F	RAM für Gleitkommaroutinen)
%F770-F77F	noch frei
(%F780-F78F	Registerspeicher des MON)
%F790-F79F	noch frei
%F7A0	Bitmaske für Textzeichen, siehe 11.
%F7A1	Bitmaske für Kursor, siehe 11.
(%F7A2-F7AB	OSD)
(%F7AC-F7DF	evtl. später OSD)
%F7E0-F7FF	Funktionstastenprogramme
(%F800-FBFF	zweiter ASCII-Textspeicher)
(%FC00-FFBF	ASCII-Textspeicher)
%FFC0-FFFF	IO-Vektortabelle, siehe 16.

IRQ-Sprünge, hierhin springen die IRQ:
%F500, %F503, %F506, %F509, %F50C ist nicht genutzt;
dieser IRQ springt über Register %56/57, %F50F

11. Bitmasken zur Textdarstellung

Die Bitmaske für Textzeichen / Kursor legt die Farbdarstellung eines Textzeichens / des Kursors fest. Dabei ist das H-nibble (Bit 7-4) für die normale (so wie im Zeichengenerator) und das L-nibble (Bit 3-0) für die inverse (alle Bits gegenüber Zeichengenerator negiert) Darstellung zuständig. Ein auf 0 gesetztes Bit bewirkt ein Einschreiben der Information ins Bild, ein 1-Bit keine Änderung im Bild. Bit 7 und 3 sind der ersten Video-RAM-ebene zugeordnet. Die Standardeinstellung ist 2D C3. Es sollten in einem Byte immer 4 Bit auf 0 sein, falls keine Spezialeffekte erwünscht sind, und

zwar jeweils Bit 7 oder 3, 6 oder 2, 5 oder 1, 4 oder 0. Ansonsten wird in die Video-RAM-ebene, für die kein entsprechendes Bit null ist, auch nichts geschrieben.

12. Registerbelegung

```
%00-03     Ports
%04-0F     Basic, darf nicht für Interrupts genutzt werden,
           selbst wenn kein Basic genutzt
%10-1F     Basic, MON, EDI
%20-25     Basicvariable A, B, C und SAVE/LOAD
%26-53     Basicvariable D-Z, [V-Z: siehe Grafikroutinen]
%54        noch frei
%55        Bit 7: 1= ESC-Sequenz läuft     0= normal
           Bit 6: 1= 80-Zeichenmodus       0= 40-Z
           Bit 4: 1= Scrolling erlaubt     0= kein S.
           Bit 3: 1= Tastencode+%80        0= normal
           Bit 2: 1= CHARIN normal         0= wie 4K-System
           Bit 1-0: noch frei (nicht ändern!)
%56/57     Sprungvektor für IRQ
%58        OSD (Zeilenlänge)
%59-5A     OS
(%5B/5C    Kursor)
%5D        noch frei
%5E-66     OS
%67        H-Byte Startadresse des Zeichengenerators
%68-6B     OS
%6C        Bit 7: 1= Autorepeat läuft      0= nicht
           Bit 6: 1= Tastenpiep an         0= aus
           Bit 5: 1= Caps aktiv       0= aus
           Bit 4-0: Autorepeatrate in 0.004 Sekunden
%6D        OSD (gedrückte Taste, 00: keine)
%6E        noch frei
%6F        durch IRQ alle 0.004 s decrementiert bis auf %00
%70-7F     von einigen Grafikroutinen und PRISTRI genutzt
```

13. Die Sprungtabelle

Sie wurde angelegt, um auch bei Weiterentwicklungen des OS zu sichern, das die Anwenderprogramme noch laufen, da sie mit Sicherheit unverändert bleibt. Hier findet man die wichtigsten Betriebssystemroutinen. Alle Routinen verändern den Registerpointer nicht. Jede Routine bekommt einen Namen, der ihre Funktion widerspiegelt. Das ist einprägsamer, als wenn man immer nur von irgendwelchen Adressen spricht.

```
%0812   KOMMAND
%0815   CHARIN
%0818   CHAROUT
%081B   KEY
%081E   WKEY
%0821   SAVE
%0824   LOAD
%0827   SCRFUN
```

%082A MONITOR
%082D PRISTRI
%0830 SHOWPLAYER
%0833 HIDEPLAYER
%0836 RND

KOMMAND ist der Einsprung nach RESET. Hierhin sollten Anwenderprogramme springen, wenn kein ordentlicher Programmabschluß mehr möglich ist.

CHARIN liefert in %13 nacheinander die Zeichen einer vom FSE editierten Zeile, als letztes noch %0D (returncode). Wenn nichts mehr zu liefern ist, dann läßt der FSE den Nutzer eine neue Zeile eingeben. Das kann man auch erreichen, indem man %58 auf %FF setzt, bevor man die Zeile vollständig geholt hat. Dies ist oft sinnvoll, da der FSE die editierte Zeile auch ab %F700 im Speicher ablegt, aber ohne %0D am Ende sondern bis zur maximalen Zeilenlänge (40/80 Zeichen) mit Leerzeichen aufgefüllt. Die Länge dieser Zeile (ohne die hinten angehängten Spaces) steht bei der ersten Rückkehr aus CHARIN um eins vermindert in %58

CHAROUT führt das Zeichen aus %15 auf dem Bildschirm aus und gibt es evt. noch auf den Drucker (siehe Register %55) aus. Führt aus heißt, daß ein normales Zeichen (%10-FF) ausgedruckt und für ein Steuerzeichen (%00-0F) die entsprechende Funktion ausgeführt wird, falls nicht gerade eine ESC-Sequenz läuft, denn dann wird ein Steuerzeichen auch dargestellt wie ein normales. Die ESC-Sequenz läßt sich jeweils für das als nächstem ausgegebene Zeichen durch Setzen von Bit 7 in %55 oder Auegabe des Kodes %0E über CHAROUT erreichen.

KEY liefert den ASCII-Kode der momentan gedrückten Taste in %6D, wenn keine gedrückt, dann %00. Wird hier die Betätigung einer Funktionstaste festgestellt, dann wird die entsprechende Routine aufgerufen, danach mit einem Piep etwa 0.5 s gewartet und %00 zurückgegeben als wäre nichts gewesen. Auch alle Tastaturabfragen im OS laufen über KEY, so daß man zum Anschluß einer total anderen Tastatur (z.B. mit eigenem Prozessor) nur den Sprung bei KEY auf eine eigene Routine, die die gleiche Funktion aufweist, ändern muß.

WKEY wartet mit Kursordarstellung auf einen Tastendruck und liefert den ASCII-Kode dann in %13. Zur Tastaturabfrage wird KEY genutzt. Hier wird der normale Tastenpiep erzeugt.

SAVE speichert den Speicherbereich auf Kassette, dessen Anfangsadresse in %20/21 steht mit der in %22/23 angegebenen Anzahl von Bytes ab. Fehlerrückmeldung wie LOAD.

LOAD lädt ab der in %20/21 stehenden Adresse die Daten von Kassette in den Speicher und gibt in %22/23 die Anzahl der fehlerfrei geladenen Byte zurück. Zusätzlich steht in %24, ob ein nichtkorrigierbarer Fehler auftrat (%FF) (das weiß man eigentlich sowieso, da man ja die B-Taste zum Abbruch drückte), oder alles ok war (%00). Die in %22/23 angegebene

Bytezahl ist auch bei nichtkorrigierbaren Ladefehlern noch korrekt, SAVE und LOAD verändern die Register %13 und %15.
Bei beiden besteht die Möglichkeit, einen Namen zu übergeben. Das wird durch die derzeit installierten Kassettenroutinen nicht genutzt, ist aber vielleicht später für einen Diskettenanschluß interessant. Dazu muß %60/61 auf das erste Byte des irgendwo im Speicher stehenden Namens zeigen.

SCRFUN sind drei Spezialfunktionen für den Textbildschirm, die zur Datenübergabe die Basicvariable X, Y, Z nutzen. Es wird aber nur das Lowbyte beachtet und gesetzt, also Variable X entspricht Register %4F. Y:%51, Z:%53. Die Funktionen werden anhand des in %53 zu übergebenen Wertes (der bei Rückkehr auf 00 steht wenn nicht anders vermerkt) unterschieden. Der Textbildschirm hat ein Koordinatensystem, dessen Ursprung (0,0) in der linken oberen Ecke liegt. X läuft nach rechts bis 39 und Y nach unten bis 23.

%00 liefert in %4F (Spalte) und in %51 (Zeile) die Position des Textkursors.

%01 setzt den Textkursor auf die in %4F (Spalte) und %51 (Zeile) übergebene Position, falls diese im Bereich des Machbaren liegt.

%02 liefert in %53 den ASCII-Kode des Zeichens, das sich auf der in %4F und %51 übergebenen Position befindet. Wer hier unsinnige Positionen angibt, bekommt auch Müll zurück.

MONITOR ruft den Maschinenmonitor MON auf. Dabei wird der Registersatz %10-1F im RAM gespeichert und bei der Rückkehr wiederhergestellt, so das selbst Basicprogramme den MON mit C%82A aufrufen können und nach Q richtig weiterlaufen.

PRISTRI druckt eine Zeichenkette über CHAROUT aus, die nach dem CALL-Aufruf beginnt und mit einem %00-Kode endet. Die Programmbearbeitung wird hinter dem %00 fortgesetzt. (verändert Register %70-72)

SHOWPLAYER siehe 14.2.

HIDEPLAYER siehe 14.2.

RND liefert nach jedem Aufruf in %74/75 eine 16-Bit-Zufallszahl. Diese kann man auch aus %F7A8/9 nehmen. Nutzt %70-77

14. Grafikroutinen

Die Programme zum Setzen und Testen von Punkten und Zeichnen von Linien wurden mit in das OS aufgenommen. Sie arbeiten ohne irgendwelche Störungen auf dem Bildschirm zu erzeugen, was sich bei der Textausgabe im Interesse einer angemessenen Geschwindigkeit leider nicht vermeiden ließ. Der Bildschirm ist als ein Koordinatensystem, dessen Ursprung (0,0) in der linken oberen Ecke ist. Die X-Achse erstreckt sich nach rechts bis zu einem Wert von 319. Die Y-Achse geht nach unten

bis 191. Jeder Punkt kann 16 verschieden Farben oder Graustufen (0 bis 15) annehmen, wenn alle RAM-Module auf der Videoplatine bestückt sind. Dabei wird ein Bit im ersten RAM-Modul durch das Bit 3 dargestellt, d.h. wenn nur ein Modul vorhanden ist, sind nur die Farben 0 und 8 sinnvoll, wobei 0 schwarz und 8 weiß ergibt.

14.1. Grafiksprungtabelle

V,W,X,Y,Z sind die Basic-Variablen. Die Farbe wird immer in Z, und zwar nur in Bit 3-0, übergeben. Die Grafikroutinen verändern nicht den Registerpointer %FD, aber die in Klammern dahinter angegebenen Register.

```
%17F7 DRAW      (%5E-66, %68-6B, %70-7F)
%17FA PTEST     (%60-65, %69-6B)
%17FD PLOT      (%60-66, %68-6B)
```

DRAW verbindet die Punkte (V,W) und (X,Y) durch eine Gerade der Farbe Z

PTEST liefert die Farbe des Punktes (X,Y) in Z

PLOT setzt den Punkt (X,Y) mit der Farbe aus Z

Wenn die X-Koordinate >319 oder die Y-Koordinate >191 wird, erscheint bei DRAW und PLOT nichts auf dem Bild und PTEST liefert Müll.

Drei kleine Beispiele:
DRAW:

```
10OPTC[12]
20P"DIESER TEXT WIRD MIT EINEM"
30P"KREUZ DURCHGESTRICHEN"
40LZ=15,V=0,W=0,X=200,Y=16;C%17F7
50LW=16,Y=0;C%17F7;E
```

PLOT:

```
10OPTC[12]
20P"UNTERSTREICHUNG MIT PUNKTEN"
30LX=0,Y=10
40LZ=X$A1*15;C%17FD
50LX=X+1;FX<216;G40
60E
```

PTEST und PLOT:

```
10OPTC[12]
20P"UMKEHRUNG DER FARBWERTE IN HALBER"
30P"ZEICHENHÖHE IN DER ERSTEN ZEILE"
40LX=0,Y=4
50C%17FA;LZ=NOT[Z];C%17FD
60LX=X+1;FX<270;G50
70E
```

14.2. Bewegte Grafik

Ein großes Problem ist insbesondere bei Spielen die Programmierung von bewegten Objekten (Playern), wenn es dafür keine

Unterstützung durch den Computer gibt. Eine Hardware zur Playerdarstellung ermöglicht die größte Geschwindigkeit, ist aber im Selbstbau zu aufwendig. Deshalb wurde für den J+T-Computer eine Softwarevariante in das Betriebssystem integriert. Dabei war ein Kompromiß zwischen Größe und Geschwindigkeit notwendig. Es wurde eine Playergröße von 16 * 16 Pixeln und die Darstellung in nur einer Ebene des Video-RAM gewählt. Damit erreicht man brauchbare Ergebnisse. Durch die Laufzeit dar Routinen läßt sich bei punktweiser Bewegung eine bestimmte Geschwindigkeit nicht überschreiten, zumal ein Player mindestens 0.04 s dargestellt sein sollte, damit er überhaupt auf der Bildröhre erscheint. Bei schneller Bewegung fällt es aber nicht auf, wenn der Player mit einem Mal mehrere Punkte weit bewegt wird.

14.2.1. Playeraufbau

Das Aussehen eines Players wird durch einen Block von 48 Byte definiert. Darin ist der Player zeilenweise abgelegt. Nach jeder Zeile (2 Byte) folgt ein Byte %00. 16 Zeilen zu je 3 Byte ergeben genau 48 Byte. Das Bit 7 des ersten Bytes ist die obere linke Ecke des Players, auf die sich die Koordinatenangaben beziehen. Ein auf 1 stehendes Bit wird im Video-RAM gesetzt (OR-Modus) oder das entsprechende Bit des Video-RAM wird negiert (XOR-Modus). Ein 0-Bit beeinflußt den Video-RAM nicht.

14.2.2. Playerdatenübergabe

Den Playerroutinen werden die Daten in den Basicvariablen V bis Z übergeben. V legt die Video-RAM-ebene, in der der Player dargestellt wird, und die Art der Darstellung fest: Im Lowbyte darf nur ein Bit auf 1 stehen, sonst treten Buskurzschlüsse der DS8286 bei SHOWPLAYER auf. Die Bits 3 bis 0 entsprechen den Video-RAM-ebenen und ihren zugehörigen Farbwerten. Ist z.B. Bit 3 auf 1 gesetzt (also %08 im Lowbyte), dann wird der Player in der Ebene dargestellt, in der auch das Bit bei PLOT mit dem Farbwert Z=8 gesetzt würde. Im Highbyte ist nur das Bit 0 von Bedeutung. Ist es 0, dann erfolgt die Einblendung des Players mit XOR, sonst mit OR. W enthält die Adresse des Speicherblocks, der das Aussehen des Players definiert.
X,Y ist die Playerposition (obere linke Ecke des 16 * 16 Punkte-Feldes). Dabei darf X aber nur Werte von 0 bis 296 und Y von 0 bis 176 annehmen.
Z enthält eine Adresse, ab der 48 Byte RAM frei sind. Dort wird der unter dem Player befindliche Bildinhalt zwischengespeichert.

14.2.3. Playerroutinen

SHOWPLAYER stellt einen Player auf dem Bild dar und speichert gleichzeitig den alten Bildinhalt dieser Stelle. Die Routine läuft durchschnittlich 10 ms und benutzt die Register %7X und %6X.

HIDEPLAYER stellt den alten Bildinhalt wieder her, löscht also den Player vom Bild. Die Routine läuft durchschnittlich 5 ms, benutzt nur %6X

Bei SHOW/HIDE-PLAYER sind Durchschnittswerte der Laufzeit angegeben, da sie im ungünstigsten Fall etwa 14 ms warten müssen, bis der Videoprozessor den oberen oder unteren Bildrand darstellt und ein störungsfreier Zugriff möglich ist.

Die Anzahl der Player auf dem Bild ist nicht begrenzt. Sofern diese in verschiedenen Ebenen dargestellt werden, ist die Reihenfolge des Ein- und Ausblendens egal. Dies gilt auch, wenn der Abstand zwischen den Koordinatenpunkten horizontal >23 und vertikal >15 ist. Andernfalls muß das Ausblenden in der umgekehrten Reihenfolge des Einblendens erfolgen.

15. Interrupt (IRQ)

Das OS nutzt, wie schon die vorigen Betriebssysteme, den Zähler T0 zur IRQ-Erzeugung. Durch den IRQ wird das Register %6F alle 0.004 s heruntergezählt, bis es auf 0 steht. Dies dient in der WKEY-Routine zur Zeitmessung. Ohne laufenden IRQ bleibt WKEY hängen.

16. Ein-/Ausgabeumlenkung

Der IO-Vektorram (%FFC0-FFFF) beinhaltet eigentlich nur Sprünge zu Ein- und Ausgaberoutinen für verschiedene Geräte, realisiert aber gerade dadurch eine große Flexibilität. Der Begriff Gerät wird hier sowohl für Hardware-Geräte (z.B. Kassettenrekorder) als auch für mehr softwaremäßige Gebilde verwendet (z.B. eine RAM-Disk). Aufgrund der Gleichheit der Aufrufe der Routinen verschiedener Geräte ist es für Anwenderprogramme einfach, verschiedene Geräte zu nutzen. Andererseits kann man wenige Bytes an dieser Stelle ändern, und die Ein- und Ausgaben eines Programms erfolgen über ein anderes Gerät, ohne daß das ausgebende Programm davon etwas merkt.

16.1. Kanäle

Es existieren vier Kanäle, wobei ein Kanal jeweils für ein Gerät da ist, sozusagen die Daten von/zu einem Gerät leitet. Die Anfangsadressen der jeweils 16 Byte umfassenden Sprünge eines Kanals sind:
%FFC0 #0, %FFD0 #1, %FFE0 #2, %FFF0 #3
ES4.0 belegt standardmäßig nur #2 und #3 mit Sprüngen. #2 ist der Kanal zum Terminal (Bildschirm und Tastatur im Zusammenwirken) und #3 geht zum Kassettenrekorder. Die Routinen SAVE und LOAD nutzen #3, CHARIN und CHAROUT nutzen #2, jeweils unabhängig vom dort konkret eingetragenen Gerät. Wenn man sich z.B. eine RAM-Disk programmiert, kann man diese in #3 eintragen, indem man die 16 Byte austauscht, und die Befehle L und S des MON und des EDI arbeiten mit der RAM-Disk anstelle des Kassettenrekorders. Dazu sind diese Befehle bereits so vorbereitet, daß man im EDI direkt hinter L oder S und im MON mit einem Leerzeichen hinter der letzten Hexzahl einen Namen für das Programm angeben kann. Die Kanäle #1 und #0 muß man derzeit noch selbst belegen, wenn man sie nutzen möchte. Dies sollte am Anfang eines Programms erfolgen.

16.2. Aufrufbedingungen

%XXX0 OPEN öffnet den Kanal für nachfolgende PUT/GET. Der Zeiger auf einen eventuell notwandigen Dateinamen muß in Register %60/61 stehen. In %15 ist der Modus zu übergeben:
Bit 0 =1: öffnen zum Lesen (über GET)
Bit 1 =1: öffnen zum Schreiben (über PUT)
Bit 2-7 sollten 0 sein (evtl. später genutzt)

%XXX3 CLOSE schließt den Kanal wieder.

%XXX6 GET liest ein Byte von dem Kanal und liefert es in %13

%XXX9 PUT schreibt ein Byte aus %15 auf den Kanal

%XXXC SPECIAL ist eine gerätespezifische Spezialfunktion. Nur die Fehlermeldung und der Zeiger auf einen evtl. notwendigen Dateinamen sind standardisiert.

Alle Routinen ändern den Registerpointer nicht. Wenn sie erfolgreich abgelaufen sind, liefern sie ein gelöschtes Carryflag. Ansonsten ist das Carryflag gesetzt und der Fehlerkode steht in %13. Dabei sind zwei Fehlerkodes bereits mit einem konkreten Sinn belegt.
%FF kennzeichnet einen Fehler in der Datenübertragung.
%88 ist die Fileendemeldung, also die Fehlermeldung, das man mit GET versucht hat, mehr Bytes zu lesen als (beispielsweise in einem Datensatz auf Kassette) vorhanden sind.

Die Terminalroutinen, die auf #2 eingetragen sind, erfordern kein OPEN oder CLOSE, liefern nie einen Fehler und man kann PUT und GET beliebig nutzen.

Die Kassettenroutinen auf #3 erfordern OPEN und CLOSE. PUT darf man nur nach einem Schreib-Open und GET nach einem Lese-Open anwenden. Bei OPEN wird der IRQ gesperrt und erst bei CLOSE wieder erlaubt. Zwischendurch darf man den IRQ nicht erlauben. Es wird normales Zeitverhalten eingestellt. Falls man für Bildzugriffe erweitertes Zeitverhalten braucht, muß man vor dem nächsten GET oder PUT bzw. vor CLOSE wieder normales einschalten.

Ein anderer Vorschlag für Reset und Stützspannungsversorgung statischer RAM

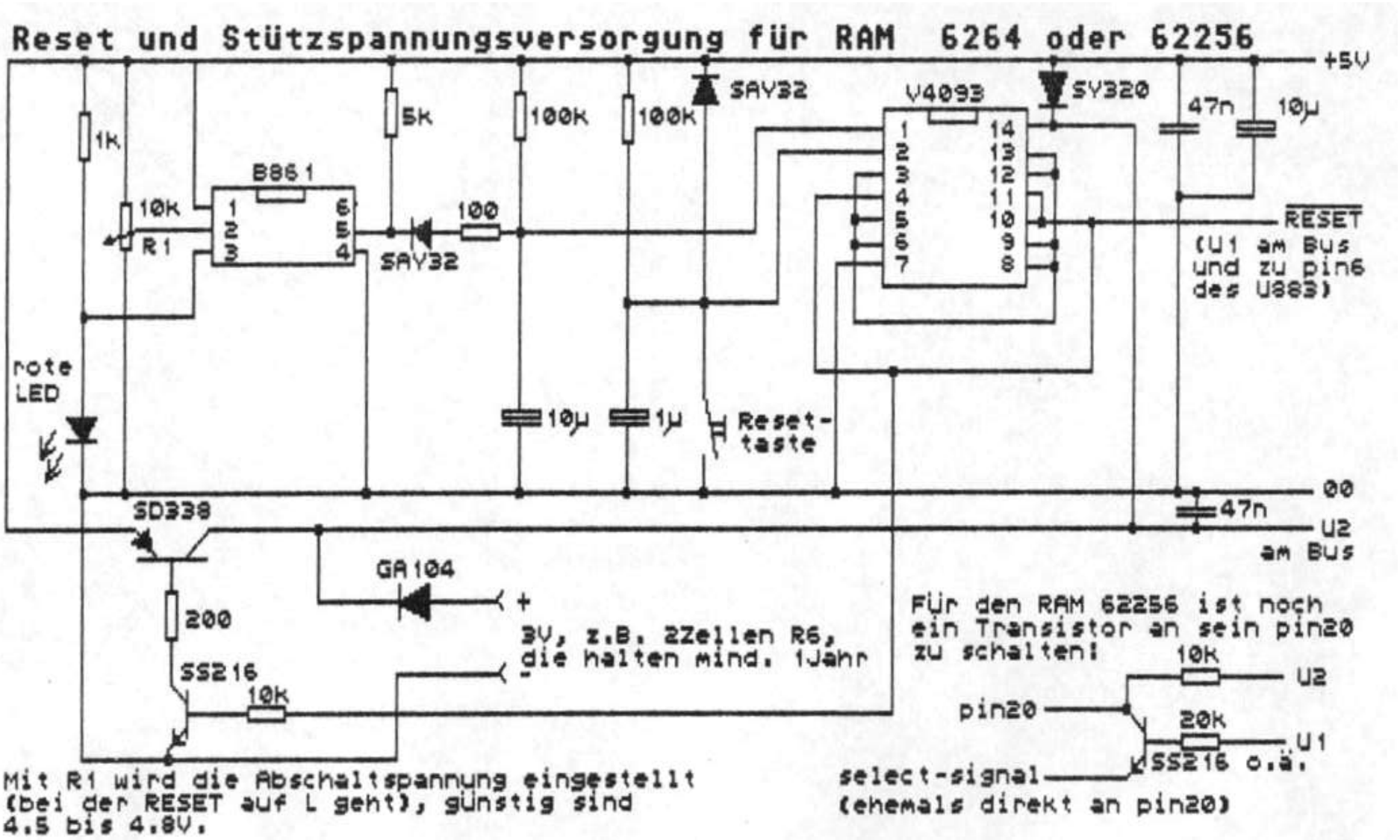
Die in [1] angegebene Schaltung für die Reset- und Stützspannungserzeugung erzeugt zwar sicher einen Resetimpuls, schützt aber oft den Speicherinhalt nicht gegen Veränderung. Dies trifft leider auch für die in [2] und [3] für den U 6516 angegebenen Ergänzungen zu. Deshalb möchte ich hier meine Erfahrungen mit der Reset- und Stützspannungserzeugung darlegen.

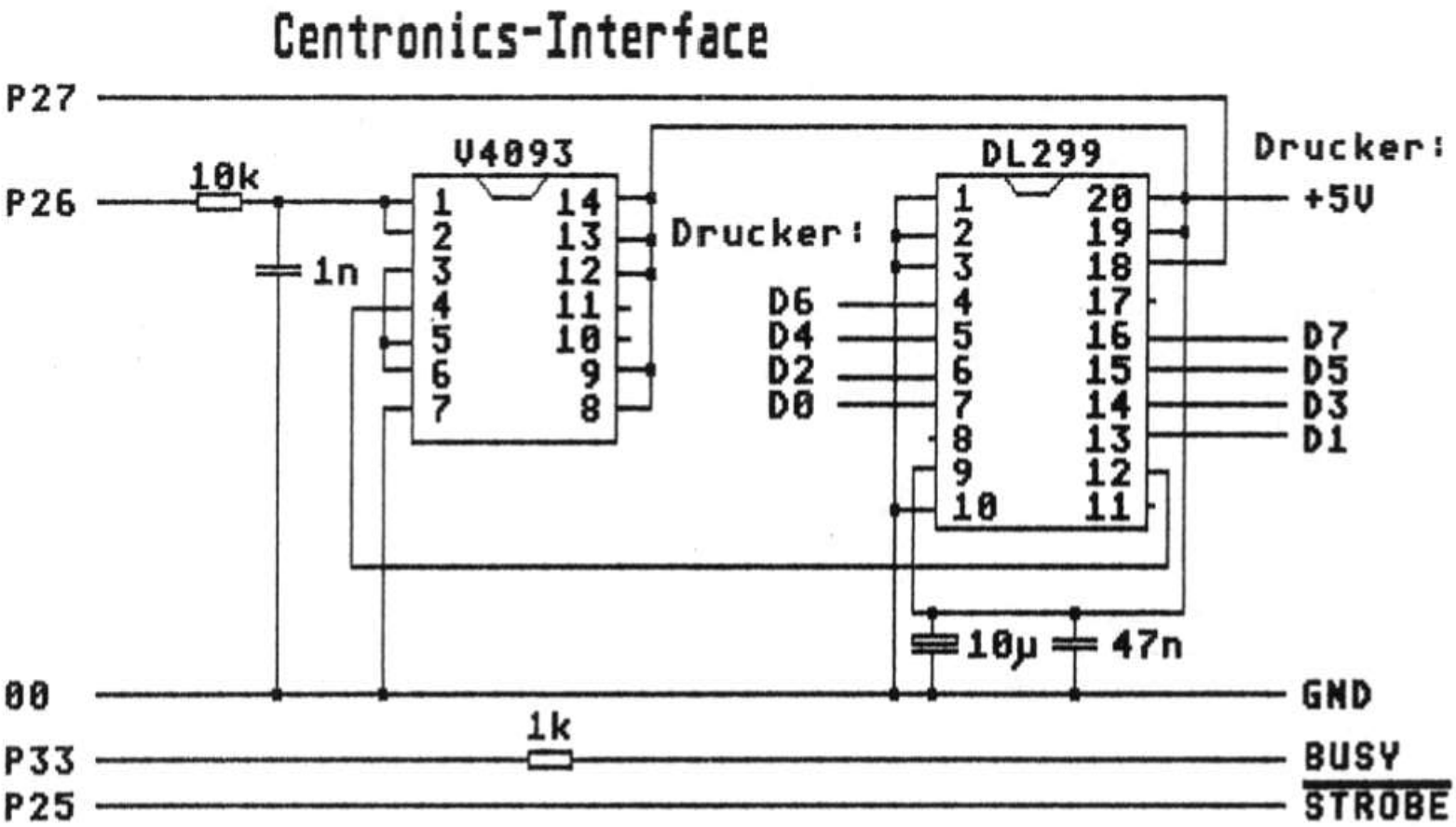
Anstelle der Schaltung in [1] kann die im Anhang angegebene genutzt werden. Sie kommt ohne eine Verbindung zur unstabilisierten Spannung am Ladekondensator aus und überwacht allein die stabilisierten +5 V. Sinken die +5 V unter den an R1 eingestellten Wert, dann wird zuverlässig ein L-Pegel an RESET erzeugt. Dieser bleibt auch nach dem vollständigen Abschalten der +5 V erhalten, da er durch den mit Stützspannung versorgten V 4093 erzeugt wird. Die Stützspannung kann anstelle der 2 * R6 auch durch drei NiCd-Akkus erzeugt werden. Deren Nachladewiderstand ist in diesem Fall über die Diode GA 104 zu legen. Sind im System nur 6264 als RAM vorhanden, müßte jetzt der RAM-Inhalt gesichert sein.

Bei einigen Schaltkreisen U 224 und U 6516 funktioniert zum Teil der Datenerhalt mit Stützspannung nicht bzw. die RAM nehmen Ströme bis über 1 mA pro Schaltkreis auf. Für U 6516, die vom Schaltkreis her in Ordnung sind, ist die Variante 1 in [3] nutzbar, allerdings mit einer Änderung: pin 16 des 74 HCT 138 darf nicht mit +5 V, sondern muß mit der Stützspannung V2 verbunden sein. Die Variante 2 verhindert zwar oft einen Datenverlust, hat aber häufig eine unangenehme Nebenerscheinung: Des Schreibsignal ist inaktiv, also sind die RAM auf Lesen geschaltet. Gleichzeitig werden durch den DS 8205 die CE- und OE-Eingänge auf L-Pegel gehalten und damit die Ausgangstreiber der RAM aktiviert. Dies führt zu einem hohen Stromverbrauch an der Stützspannung.

Für den RAM 62256 ist zusätzlich die im Anhang angegebene Schaltung mit dem einen Transistor erforderlich, um das CE-Signal inaktiv zu schalten.

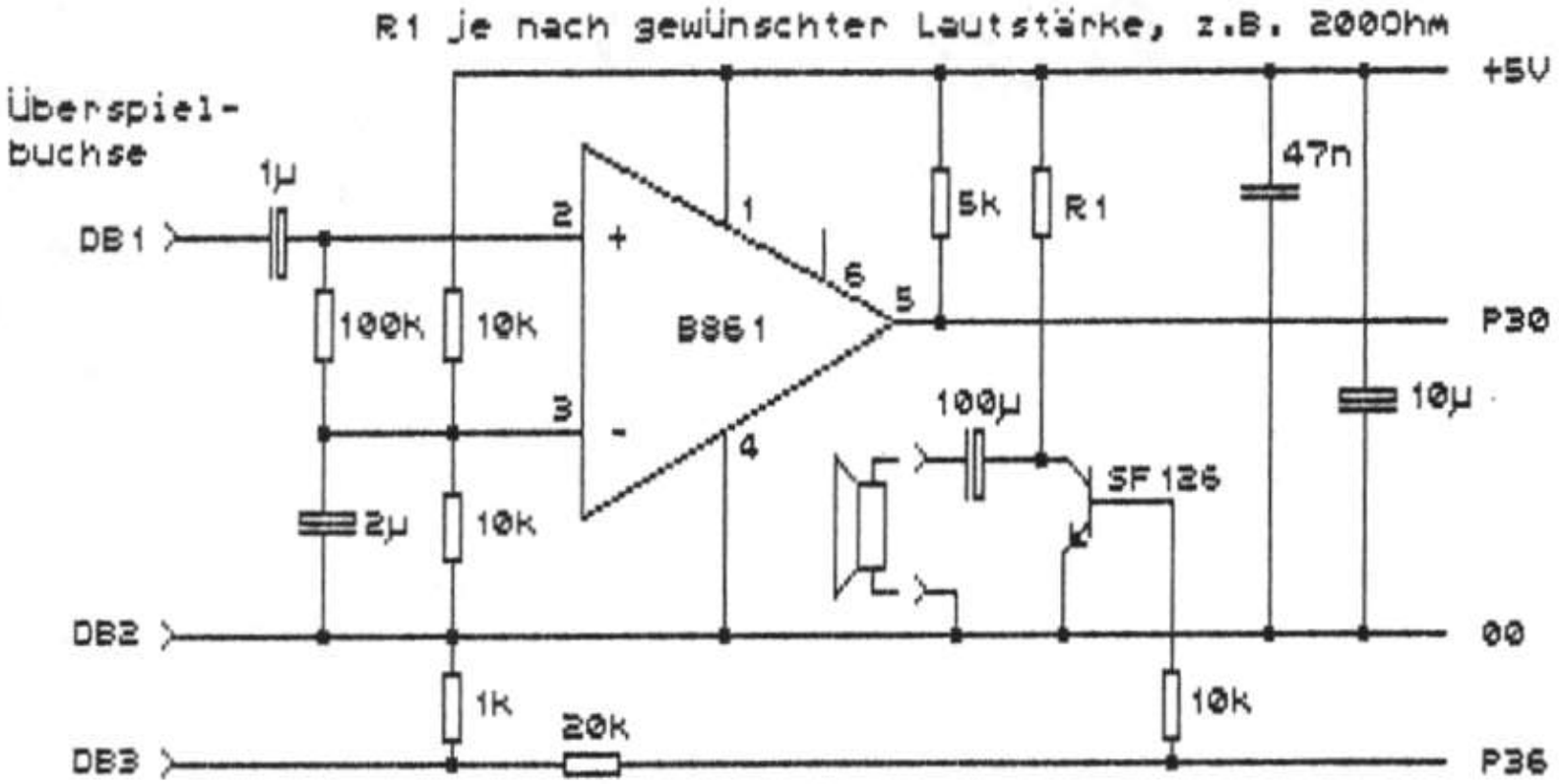
[1] JU+TE 4/1988, S.287-288
[2] JU+TE 6/1988, S.442-444
[3] JU+TE 9/1989, S.713


Reset und Stützspannungsversorgung für RAM 6264 oder 62256
+5V
1K
5K
100K
100K
SAY32
V4093
SY320
47n
10µ
B861
10K
R1
100
SAY32
RESET
(U1 am Bus
und zu pin6
des U883)
rote
LED
10µ
1µ
Reset-
taste
00
47n
SD338
U2
am Bus
GA104
+
200
3V, z.B. 2Zellen R6,
die halten mind. 1Jahr
-
SS216
10K
Für den RAM 62256 ist noch
ein Transistor an sein pin20
zu schalten!
10K
U2
pin20
20K
U1
SS216 o.ä.
select-signal
(ehemals direkt an pin20)
Mit R1 wird die Abschaltspannung eingestellt
(bei der RESET auf L geht), günstig sind
4.5 bis 4.8V.



Wenn der Drucker keine +5V liefern kann, die +5V vom Computer nutzen. Die 5 Leitungen zum Computer sollten max. 2m lang sein. P25, P26 und P27 im Computer mit je 20k an +5V legen, damit immer korrekter H-Pegel erzeugt wird.

Kassetteninterface



ASCII-Tabelle

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		←	→	↑	↓	↖					▲	▼		↵		
1											ä	ö	ü	Ä	Ö	Ü
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	¦	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	♦	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	ß

Steuerzeichencodes

00 nicht belegt
01 Kursor links
02 Kursor rechts
03 Kursor hoch
04 Kursor runter
05 HOM
06 SOL
07 DEL
08 DBS
09 INS
0A LDE
0B LIN
0C CLS
0D RET
0E ESC
0F nicht belegt

Bild auf der Nebenseite: Bestückungsplan der Bildschirmsteuerung
Layout: Hoyer

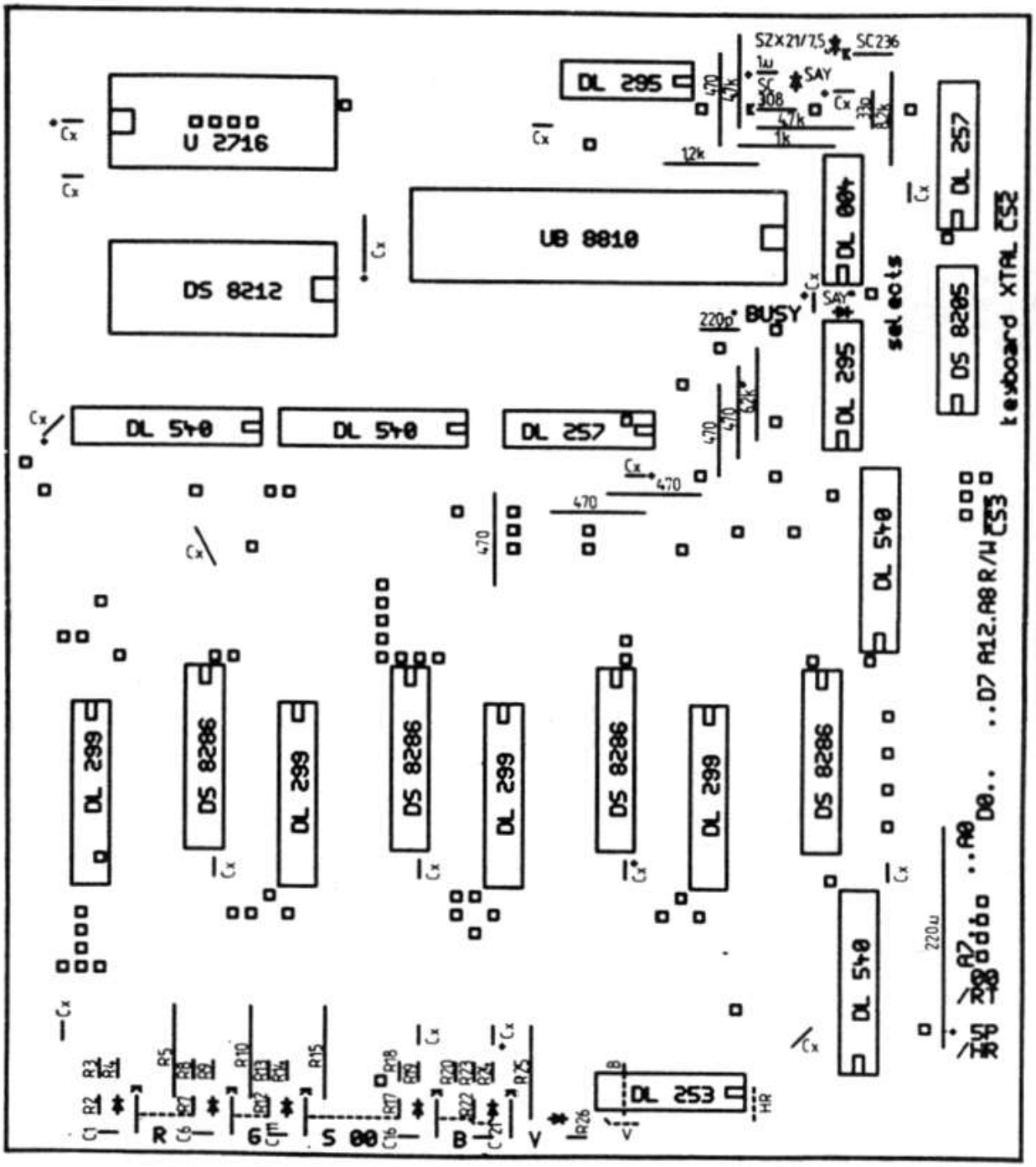
Die kleinen Quadrate sind Durchkontakte, Die Cx sind Kondensatoren mit 22 nF bis 100 nF.

Tastaturbelegung

Taste															
Diese Spalte war beim 2K/4K-System nicht nutzbar.	noch nicht belegt	!	2	3	4	5	6	7	8	9	0	ß			
		DBS	INS	CLS		ä	ö	ü	Ä	Ö	Ü	^			
		!	"	#	$	%	&	'	@	(	)	>			
		1	2	3	4	5	6	7	8	9	0	<			
	SHT3	F1	F2	F3	F4	F5	F6	F7	F8			♦			
		DEL	↑	SOL	LDE							\			
		Q	W	E	R	T	Z	U	I	O	P	-			
		q	w	e	r	t	z	u	i	o	p	+			
	SHT2	←	HOM	→	LIN										
		A	S	D	F	G	H	J	K	L	:	/			
		a	s	d	f	g	h	j	k	l	;	*			
	SHT1		↓		ESC				{	}	_	~			
		Y	X	C	V	B	N	M	[	]	=	?			
		y	x	c	v	b	n	m	,	.	Space	RET			

← mit SHT3
← mit SHT2
← mit SHT1
← allein

Space ist die Leertaste


U 2716
DS 8212
UB 8810
DL 295
DL 257
DL 004
DS 8205
DL 540
DL 540
DL 257
DL 295
DL 540
DL 299
DS 8286
DL 299
DS 8286
DL 299
DS 8286
DL 299
DS 8286
DL 540
DL 253
SZX21/7.5
SC236
SAY
BUSY
keyboard XTAL CS2
..D7 A12.A8 R/W CS3
..D0 ..A0
R
G
S 00
B
V