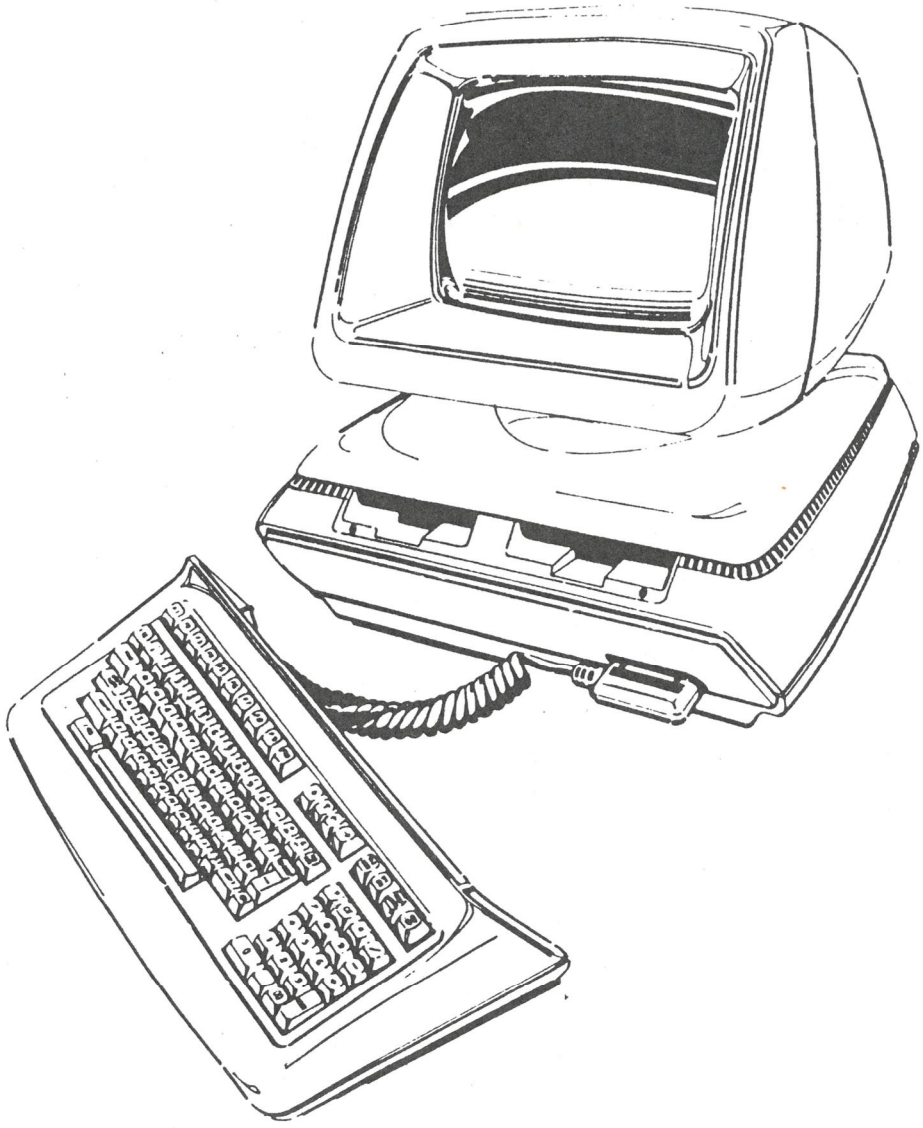
# BEDIENUNGSANLEITUNG



Dr. R. Meingast
Hasenweg 20
5253 Lindlar
(Schmitzhöhe)
Tel. 02207 / 6841

## 0. Inhaltsverzeichnis



Erwünscht: User port - Belegung der Stifte

# T E I L 1

## 1. Vorwort

Herzlichen Glückwunsch! Sie haben sich durch den Kauf des TURBO CBM 8700 für den schnellsten auf dem Markt befindlichen 8 BIT-Computer entschieden. Dieser Rechner wurde durch die Firma ultra electronic in Köln entwickelt auf der Basis bekannter Systemkomponenten. Grundlage des TURBO CBM 8700 ist der CBM 720 der Firma Commodore. Er wird ergänzt durch die PROXA-7000-Platine, die im wesentlichen alle Portbausteine, die Adressdekodierung und den Mikroprozessor des CBM 8032/8096 enthält. Dadurch wird die volle Kompatibilität zu diesem Rechner erreicht.

Gerade diese volle Kompatibilität zu den Rechnern CBM 700 und CBM 8032/8096 zeichnet diesen Rechner aus. Diese Bedienungsanleitung möchte deshalb nicht nochmals alle Leistungsmerkmale beider Rechner beschreiben, da diese in den entschsprechenden Handbüchern enthalten sind. Vielmehr sollen die zusätzlichen Leistungsmerkmale erläutert und durch Programmbeispiele veranschaulicht werden.

Da die Bedienung dieses Rechners nur unwesentlich von der Bedienung der Vorgängermodelle abweicht und die Unterschiede in erster Linie bei der Programmierung relevant sind, geriet diese Bedienungsanleitung eher zu einem Programmierhandbuch. Dabei wurde besonderer Wert auf die Beschreibung der zusätzlichen Leistungsmerkmale gelegt. Im Anhang A wird jedoch auch auf Literatur hingewiesen, die in die Bedienung und Programmierung der Rechner CBM 700 und CBM 8032/8096 einführt.

Im Lieferumfang des Rechners ist auch ein Programmpaket enthalten, das im Rahmen dieser Bedienungsanleitung beschrieben wird.
Die im jeweiligen Anhang befindliche Dokumentation soll es dem Anwender erleichtern, dieses Programm an seine eigenen Bedürfnisse anzupassen.

Diese Bedienungsanleitung wurde erstellt durch die Firma

BRILL-Software
Hans-Böckler-Str. 3
5190 Stolberg

Die Entwicklung des TURBO-CBM-8700 erfolgte durch Herrn Helmut Proxa, Geschäftsführer der Firma

ultra electronic GmbH & Co KG
Wormser Straße 45
5000 Köln 1

Telefon: (0221) 38 60 44

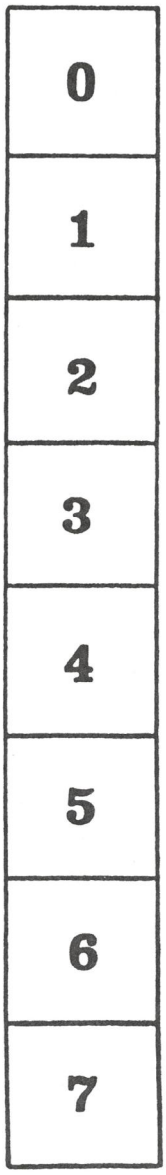
## 2.0 Aktivieren der PROXA-7000-Platine

Wie bereits im Vorwort erwähnt, basiert der TURBO CBM 8700 auf dem Rechner CBM 700 der Firma Commodore. Nach dem Einschalten ist auch kein Unterschied zu diesem Rechner erkennbar. Die einzige Ausnahme besteht darin, daß das BIT 7 des USER-Ports nicht benutzt werden darf, da hierüber die Umschaltung in die CBM 8032-kompatible Betriebsart erfolgt. Durch dieses BIT wird der Mikroprozessor 6509 des CBM 700 abgeschaltet und der Mikroprozessor 6512 des CBM 8032 aktiviert. Gleichzeitig ändert sich die Speicheraufteilung, da nun die Adreßdekodierung des CBM 8032 aktiv wird.

Der USER-Port des CBM 700 wird - wie bereits oben erwähnt - zur Steuerung der PROXA-7000-Platine genutzt. Bild 1 zeigt die Funktion dieses Registers:

**Wahl der aktiven Speicherbank**

| BIT 0 | BIT 1 | BIT 2 | BIT 3 | -> | Bank |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | 0 |
| 0 | 0 | 0 | 1 | | 1 |
| 0 | 0 | 1 | 0 | | 2 |
| 0 | 0 | 1 | 1 | | 3 |
| 0 | 1 | 0 | 0 | | 4 |
| 0 | 1 | 0 | 1 | | 5 |
| 0 | 1 | 1 | 0 | | 6 |
| 0 | 1 | 1 | 1 | | 7 |
| 1 | 0 | 0 | 0 | | 8 |
| 1 | 0 | 0 | 1 | | 9 |
| 1 | 0 | 1 | 0 | | 10 |
| 1 | 0 | 1 | 1 | | 11 |
| 1 | 1 | 0 | 0 | | 12 |
| 1 | 1 | 0 | 1 | | 13 |
| 1 | 1 | 1 | 0 | | 14 |
| 1 | 1 | 1 | 1 | | 15 |




**Aktivieren der Fenster**

**Abschalten der CBM-700-Portbausteine**

**Reserviert, Inhalt immer 0!**

**Aktivieren der PROXA-7000-Platine**

Bild 1 : Das Kontrollregister der PROXA-7000-Platine

Dieses Register befindet sich im CBM 700-Modus auf der Adresse $DC01. Werden in die vier niederwertigsten Bits die Banknummer geschrieben und die Bits 4 und 7 gesetzt, so wird die PROXA-7000-Platine zur Emulation eines CBM 8032 aktiviert. Zuvor müssen jedoch die Interrupts der Portbausteine des CBM 700 unterbunden werden, da der Interrupt des CBM 8032 auf diese nicht richtig reagieren kann. Bild 2 zeigt eine solche Umschaltroutine:

```
INIT6512   CLI                ;Interrupt erlauben
           LDA #$00
           STA $DE05          ;IRQ-Controller 6525 off
           SEI                ;Interrupts verbieten
           STA $DF03          ;Tastaturport 6525 off
           STA $DF04          ;(Port A + B auf INPUT)
           STA $DC03          ;6526 = INPUT
           LDA #%10010001
           STA $DC01          ;Kontrollregister initialisieren
           LDA #$FF           ;Kontrollregister auf OUTPUT
           STA $DC03          ;(dadurch erst wirksam)
           .BYTE $02          ;KILL 6509
```

Bild 2: Umschaltroutine 6509 -> 6512

Zunächst einmal müssen angefallene Interrupts des 6509 abgearbeitet werden, da diese vom IRQ des CBM 8032 nicht richtig verarbeitet werden können. Sodann muß verhindert werden, daß weitere Interrupts anfallen können. Acht Bit des Triports 6525 haben im CBM 700 die Funktion eines Interruptcontrollers. Durch Löschen des Registers $DE05 werden weitere Interrupts verhindert. Danach wird das INTERRUPT-ENABLE-Flag erneut gesetzt. Durch diese Maßnahmen wird sichergestellt, daß die Portbausteine des CBM 700 keine Interrupts mehr auslösen können.

Um Störungen bei der Dekodierung der Tastatur und des IEEE-Busses zu vermeiden, müssen die entsprechenden Portbausteine auf INPUT geschaltet werden. Die entsprechenden Ausgänge der Rechner CBM 700 und CBM 8032 sind parallel geschaltet und würden sich ohne diese Vorsichtsmaßnahme gegnseitig behindern.

Der USER-Port des CBM 700 wird beim RESET auf INPUT geschaltet. Deshalb kann in das Datenregister zunächst ein beliebiger Wert geschrieben werden, ohne daß dies irgendwelche Konsequenzen hätte. Erst durch das Umschalten des Datenrichtungsregisters auf OUTPUT werden diese Informationen aktiv.

Die CPU 6502 und damit auch die CPU 6509, die über die selbe Maske verfügt, hängt sich bei vielen unerlaubten OP-Codes auf. Ein solcher unerlaubter OP-Code ist das Byte $02. Ohne diese Vorsichtsmaßnahme verursacht der abgeschaltete Mikroprozessor 6509 noch einige Störungen, die den ordnungsgemäßen RESET des 6512 behindern würden.

Durch das Setzen des BITs 7 im Register $DC01/$8C01 wird die Stromversorgung des 6509 abgeschaltet und die des 6512 zugeschaltet. Dies hat zur Folge, daß eine einmal aktivierte PROXA-7000-Platine nicht wieder abschaltbarist. Nur das Ausschalten des Rechners bewirkt eine Rückkehr in die CBM 700-Betriebsart.

## 3. Beschreibung der verwendeten Hardware

Der Mikroprozessor 6509 des CBM 700 ist eine Weiterentwicklung des 6502. Er verfügt über den selben Befehlssatz und ist dadurch in der Lage, 6502-Programme abzuarbeiten. Lediglich die Befehle mit der Indirekt-Y-indizierten Adressierungsart verfügen über ein zusätzliches Leistungsmerkmal. Bei diesen Befehlen werden zusätzlich zu den sechzehn Adreßbits vier weitere auf den Bus ausgegeben. Diese wählen eine der sechzehn Speicherbänke an, wodurch insgesamt 1MByte Speicher adressierbar wird. Auf welche Bank dabei zugegriffen wird, legt dabei das DATA-INDIRECTION-Register fest. Dieses CPU-interne Register liegt an Stelle der ZERO-Page-Adresse 01 im Adreßraum der CPU. Ein weiteres Register, das DATA-Execution-Register, legt die Bank fest, aus der alle OP-Codes und Operanden gelesen werden.

Diese Eigenschaften erhöhen die Leistungsfähigkeit dieses Mikroprozessors erheblich, jedoch schränken sie die Kompatibilität in hohem Maße ein. Deshalb war es unumgänglich, dem System einen zweiten Mikroprozessor hinzuzufügen. Die Wahl fiel dabei auf den 6512, der ausnahmslos softwarekompatibel zum 6502 ist und lediglich über eine andere Pinbelegung und ein verbessertes Bustiming verfügt.

Der CBM 700 verfügte über einen fatalen Fehler: hing sich die CPU auf, so blieb der Refresh des Speichers aus, wodurch der Inhalt des dynamischen RAMs verloren ging. Durch eine entsprechende Schaltung auf der PROXA-7000-Platine konnte dieser Fehler behoben werden. Dies erleichtert insbesondere die Programmentwicklung in der Testphase.

Die Portbausteine des CBM 700 sind auch im 8032-Modus weiterhin erreichbar. Dies ermöglicht die Implementierung einer V.24-Schnittstelle (RS 232C), da der CBM 700 bereits über den entsprechenden Portbaustein verfügt. Außerdem wird dadurch der Zugriff auf das Kontrollregister $8C01 und den 6545 (Bildaufbereitung) freigegeben. Eine Beschreibung dieser Portbausteine enthält Anhang C.

Bei der Ansteuerung dieser Portbausteine wurden die Basisadressen beibehalten. Dadurch können selbst Maschinenprogramme weiterhin unverändert bertrieben werden. Lediglich im Falle des Bildschirmcontrollers 6545 und der Tastatur mußte von diesem Konzept abgewichen werden. Aus diesem Grunde sind auch diese Abweichungen und die entsprechenden Konsequenzen und mögliche Reaktionen im Anhang C beschrieben.

## 4. Bedienung des Speicher-Kontroll-Registers

Der Turbo-CBM-8700 ist nicht nur zum CBM 8032 voll kompatibel, sondern verfügt auch über alle Eigenschaften eines CBM 8096. Dazu wird die Bank 2 in vier Speicherbereiche mit je 16 kByte unterteilt, von denen jeweils 32 kByte im oberen Adreßbereich der CPU ($8000-$FFFF) adressiert werden können.

Sie sprechen diesen erweiterten Speicher über ein Register an, indem Sie zwischen den vier 16 kByte-Blöcken umschalten, die Platine einschalten und den Schreibschutz zuschalten. Weitere Bits in diesem Register ermöglichen den Zugriff auf Bildschirm- und Port-Bereiche selbst bei zugeschalteter Speichererweiterung.

Nach dem Einschalten des Rechners verfügen Sie wie bisher über Ihren Rechner - ohne jeden Unterschied. Erst durch Zuschalten der Speichererweiterung über das zuvor erwähnte Register wird der zusätzliche Speicher aktiv.

Der Zugriff auf diesen Speicher wird durch diverse Softwarepakete auf dem Markt unterstützt. Natürlich kann auch jeder Anwender die Speichererweiterung selbst ansteuern, wenn er mit Assembler und dem CBM-Betriebssystem vertraut ist. Eine Ansteuerung von BASIC aus ist ohne eine Betriebssystem-Erweiterung (z.B.: LOS-96) nicht möglich.

Im Folgenden soll erklärt werden, wie Sie selbst die Speichererweiterung ansteuern können. Dies geschieht im Wesentlichen über das Speicher-Kontroll-Register, welches sich auf der Speicherstelle $FFF0 (hex), bzw. 65520 (dez) befindet. In diesem Speicher-Kontroll-Register hat jedes der 8 Bit eine eigene, absolut selbständige Bedeutung.

Die Speichererweiterung befindet sich im Speicherbereich von $8000 (hex), bzw. 32768 (dez) bis $FFFF (hex), bzw. 65535 (dez). In diesem Bereich haben jeweils zwei der vorhandenen vier 16 kByte Blöcke der Speichererweiterung gleichzeitig Platz. Dies ermöglicht somit also den Zugriff auf zweimal 32 kByte Zusatzspeicher für den Anwender.

Die Zuschaltung des erweiterten Speichers erfolgt über die Bits 2 und 3 des Speicher-Kontroll-Registers. Näheres hierzu kann dem Diagramm 2 entnommen werden.

Die Adressen $8000 bis $BFFF können nur durch die Speicher-Bänke 0 (Blöcke) oder 1 belegt werden. Entsprechend können die beiden anderen Speicher-Bänke 2 und 3 ausschließlich im Speicherbereich von $C000 bis $FFFF liegen.

Diagramm 1 : Grafische Darstellung der Speicherbereiche

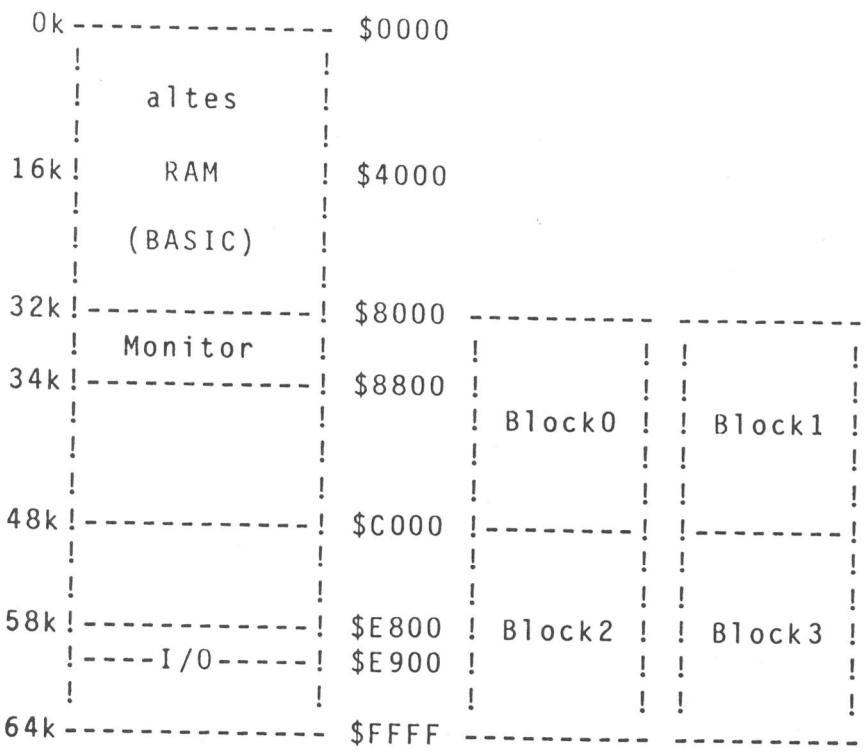


Diagramm 2 : Das Speicher-Kontroll-Register

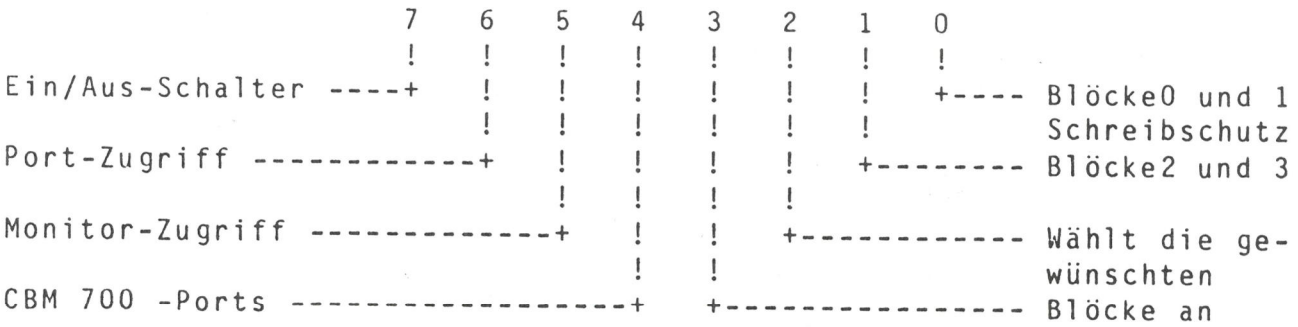



Diagramm 3 : Wahl der Speicher-Blöcke

| Bit 3 | Bit 2 | |
|---|---|---|
| 0 | 0 | Blöcke 0 und 2 aktiv |
| 0 | 1 | Blöcke 1 und 2 aktiv |
| 1 | 0 | Blöcke 0 und 3 aktiv |
| 1 | 1 | Blöcke 1 und 3 aktiv |

## Beschreibung des Speicher-Kontroll-Registers

**Bit 7**
Gesetzt: Speichererweiterung ist eingeschaltet
Gelöscht:Speichererweiterung ist ausgeschaltet
Beim Einschalten ist Bit7 immer gelöscht !

**Bit 6**
Gesetzt: Port-Bereich wird ausgeblendet
Gelöscht:Auch im Port-Bereich liegt RAM

**Bit 5**
Gesetzt: Bildschirm-Bereich wird ausgeblendet
Gelöscht:Auch im Bildschirm-Bereich liegt RAM

**Bit 4**
Mit diesem Bit können die Portbausteine des CBM 700 abgeschaltet (=0) oder zugeschaltet (=1) werden. Sind die Portbausteine abgeschaltet, so liegt in diesem Speicherbereich RAM und ermöglicht die Kompatibilität zu den Rechner CBM 8296/8296-D.

**Bit 3**
Gesetzt: Bank 3 ist angewählt
Gelöscht:Bank 2 ist angewählt

**Bit 2**
Gesetzt: Bank 1 ist angewählt
Gelöscht:Bank 0 ist angewählt

**Bit 1**
Gesetzt: Die Blöcke 0, bzw. 1 können nicht mehr beschrieben werden
Gelöscht:Die Blöcke 0, bzw. 1 können beschrieben und gelesen werden

**Bit 0**
Gesetzt: Die Blöcke 2, bzw. 3 können nicht mehr beschrieben werden
Gelöscht:Die Blöcke 2, bzw. 3 können beschrieben und gelesen werden

## Die Fenster im RAM-Bereich

Im Bereich von $E800 bis $E900 liegen die Portbausteine Ihres CBM-Computers:
1. 6520 PIA ab $E810
2. 6520 PIA ab $E820
3. 6522 PIA ab $E840

Für viele Anwendungen kann es sinnvoll sein, auf diese Bausteine weiterhin zugreifen zu können, obwohl eigentlich in diesem Bereich RAM liegt. Dafür ist Bit 6 vorgesehen. Zum Zugriff auf diese Bereiche muß Bit 6 gesetzt werden, falls die Speichererweiterung zugeschaltet wurde.

Im Bereich von $8000 bis $87FF liegt der Bildschirm-Wiederhol-Speicher. Um weiterhin auf den Bildschirm zugreifen zu können muß Bit 5 gesetzt werden, falls die Speichererweiterung zugeschaltet wurde.

## Literaturverzeichnis


Addison-Wesley — **"Basic and the Personal Computer"**, Dwyer and Critchfield

BRILL-Software — **"65C02 Assembler Programmierung auf CBM"**, Hans-Peter Brill

Commodore Büromaschinen Deutschland — **"Commodore CBM 600/700 Bedienungshandbuch"**
**"Commodore CBM 600/700 Programmierhandbuch"**
**"Commodore Computer Bedienungshandbuch für den CBM 8032"**

Compute — **"Compute´s First Book of PET/CBM"**

Cowbay Computing — **"Feed me, I´m Your PET Computer"**, C. Alexander
**"Looking good with your PET"**, Carol Alexander
**"Teacher´s PET - Plans, Quizzes, and Answers"**

Dilithium Press — **"Basic Basic-English-Dictionary for the PET"**, Larry Noonan
**"PET NASIC"**, Tom Rugg and Phil Feldman

Faulk Baker Associates — **"MOS Programming Manual"**, MOS Technology

Franzi´s Verlag — **"Erfolgreicher mit CBM arbeiten"** für alle CBM-Anwender eine verständliche Einführung in die Maschinensprache, Dipl.Ing. Franz Wunderlich
**"Der IEC-Bus"**, Piotrowsky

Hard & Soft — **"ROM-Listing für CBM 8032-ASCII"**
**"ROM-Listing für CBM 8032-DIN"**
**"ROM-Listing für CBM 600/700"**
**"ROM-Listing für CBM 720"**
O. Braun, Schineis

Hayden Bool Co. — **"BASIC From the Ground UP"**, David E. Simon
**"I Speak BASIC to My PET"**, Aubrey Jones, jr.
**"Library of PET Subroutines"**, Nick Hampshire
**"PET Graphics"**, Nick Hampshire
**"BASIC conversions Handbook Apple, TRS-80, PET"**, David A. Brain, Phillip R. Oviatt, Paul J. Paquin, and Chandler P. Stone

Howard W. Sams — **"PET Interfacing"**, James N. Downey and Steven M. Rodgers

| | |
|---|---|
| McGrow-Hill | **"Hands-On BASIC With a PET"**, Herbert D. Peckman |
| MOS-Technologie | **"MCS 6500 Microprozessor-Family - Programming Manual"**<br>**"MCS 6500 Microprozessor-Family - Hardware Manual"** |
| Osborne/McGraw-Hill | **"PET/CBM Personal Computer Guide"**, Carroll S. Donahue<br>**"PET Fun and Games"**, R. Jeffries and G. Fisher<br>**"PET and the IEEE"**, A. Osborne and C. Donahue<br>**"CBM Professional Computer Guide"**, Russell Rector and George Alexy<br>**"The PET Personal Guide"**, Russell Rector and George Alexy |
| Prentice-Hall | **"The PET Personal Computer for Beginners"**, S. Dunn and V. Morgan |
| Reston Publishing Co | **"PET and the IEEE 488 Bus (GPIB)"**, Eugene Fisher and C. W. Jensen<br>**"PET BASIC - Training Your PET Computer"**, Ramon Zamora, Wm. F. Carrie, and B. Allbrecht<br>**"PET Games and Recreation"**, M Ogelsby, L. Lindsey, and D. Kunkin<br>**"PET BASIC"**, Richard Huskell |
| SM-Software AG | **"ROM-Listing für den CBM 8032"** |
| tewi-Verlag | **"CBM-Computer-Handbuch"**, A. Osborn<br>**"Programme für CBM"**, L. Oswald |
| Total Information Services | **"Understanding Your PET/CBM, Vol. 1, BASIC Programming"** |

## Programmierbeispiele für den Turbo CBM 8700

Auf jedem Rechner gibt es einige Kniffe und Tricks, mit deren Hilfe Dinge leicht fallen, die sonst garnicht oder nur schwer lösbar sind. Die Entdeckung solcher Kniffe bleibt bei vielen Rechnern dem Zufall überlassen oder bedürfen tagelanger Experimente, die viel Zeit und Nerven kosten. Dieses Kapitel soll dies ersparen, indem Beispiele für die gängigsten Aufgabenstellungen im Zusammenhang mit der Proxa-7000--Platine aufgezeigt werden.

### Kompatibilität zum CBM 8296/8296D

Der TURBO CBM 8700 ist auch zum CBM 8296/8296-D weitestgehend kompatibel. Der 8296 verfügt über insgesamt 128 kByte RAM und 20 kByte ROM. 96 kByte dieses RAMs werden wie beim CBM 8096 genutzt, zu dem der Turbo CBM 8700 bekanntermaßen voll kompatibel ist. Die weiteren 32 kByte Speicher können anstelle der ROMs eingeblenet werden. Kopiert man zuvor das Betriebssystem in dieses RAM, so verfügt man über einen Rechner, dessen Betriebssystem durch POKE-Befehle veränderbar ist. In diesem Zustand befindet sich der TURBO CBM 8700 stets.

Viele Programme greifen jedoch auf das Register auf $FFF0 (65520) in unzulässiger Weise zu. So verändert das Programm ´BURNIN 8296´ beispielsweise BIT 4 dieses Registers. Werden sodann im Adreßbereich $8800-$8FFF Daten abgelegt, so kann dies zum Aufhängen des Rechners führen. Um dies zu vermeiden sollten die Portbausteine des CBM 700 ausgeblendet werden. Dies geschieht über BIT 5 des Registers auf $8C01 (35841):

```
POKE 35841 , PEEK (35841) OR 32
```

### Zugriff auf die zusätzlichen Speicherbänke

Der TURBO CBM 8700 verfügt über maximal 1 MByte RAM, wovon bis zu 512 kByte durch die mitgelieferte Software unterstützt wird. Zur Ablage von Daten durch Betriebssystemerweiterungen oder Maschinenprogramme kann sowohl der bereits unterstützte Speicher, als auch der zusätzliche Speicher genutzt werden.

Der Adreßraum der 6502-CPU ist auf 64 kByte beschränkt. Deshalb kann stets nur auf eine Speicherbank zugegriffen werden. Wird die Speicherbank umgeschaltet, so liest die CPU nicht nur die Daten aus dieser Bank, sondern erwartet auch das Programm an dieser Stelle. Dies beinhaltet jedoch eine Problematik: Beim Umschalten in eine andere Bank hängt sich die CPU auf, weil sie dort kein entsprechendes Programm findet.

In diesem Falle hilft ein kleiner Trick: Der Bildschirmspeicher liegt im Adreßraum jeder Bank, wenn das BIT 4 im Register $8C01 (35841) gesetzt bleibt. Außer den 2000 dargestellten Zeichen enthält das Bildschirm-RAM noch 48 weitere Bytes. Legt man in diesen Speicher das Programm, welches den Speicher umschaltet und kopiert, so treten keine Probleme in der oben genannten Art auf.

Werden diese 48 Byte hinter dem Bildschirminhalt eventuell noch durch andere Programme genutzt, so muß der Inhalt zunächst gerettet werden, bevor er durch ein Programm überschrieben wird und wiederhergestellt werden, nachdem der Zugriff auf die zusätzlichen Speicherbänke abgeschlossen ist. In dieser Art gehen die Programme RAM-Floppy und 8832--Modus vor. In der Dokumentation am Ende dieser Anleitung sind die entsprechenden Routinen ausführlich erläutert.

Beispielroutinen in der RAM-Floppy

| | |
|---|---|
| SC.NA.BU | Rettet benutzten Teil des Video-RAMs |
| BU.NA.SC | Zurückholen des geretteten Video-RAMs |
| LESEROUTINE | Lese angegebenen Block aus dem erweiterten RAM |
| SCHREIBROUT. | Schreibe angegebenen Block in erweitertes RAM |
| LESENKOPIEREN | Kopiere Leseroutine in das Video-RAM |
| SCHR.KOPIEREN | Kopiere Schreibroutine in das Video-RAM |
| BLOCKLESEN | Kopiert angegebenen Block in angegebene Bank |
| BLOCKSCHREIB | Kopiert angegebenen Block aus angegebener Bank |

Ein Zugriff auf den erweiterten Speicher sähe dann also folgendermaßen aus:

1.) JSR SC.NA.BU
2.) LDA §Nummer der Bank
3.) STA BANKNUMMER
4.) LDA §Nummer des Blocks
5.) STA BLOCKNUMMER
6.) JSR BLOCKLESEN
7.) JSR BU.NA.SC

Diese Routine liest 256 Bytes aus der angegebenen Bank. Die Blocknummer legt die Page fest, die gelesen werden soll. Anschließend liegen die Daten im Transferbuffer (TRANFERBUFF).

Sehr viel schneller kann man auf den Speicher zugreifen, wenn der sehr umständliche Umweg über den Bildschirmspeicher entfällt. Auch dies ist möglich, setzt jedoch voraus, daß die angesprochene Bank ein Programm enthält, welches parallel zu dem aktuellen Programm liegt und durch den Mikroprozessor nach dem Umschalten abgearbeitet werden kann. das dazu notwendige Programm kann über die Methode 1 (siehe oben) dort abgelegt worden sein.

Beispielprogramm für schnellen Bankwechsel

```
Programm in Bank 1                Programm in Bank 2
START           ...               START           ...
                LDA §$02                          ...
                SEI                               ...
                STA $8C01                         ...
                NOP                               NOP
                ...                               ...
                ...                               LDA §$11
                ...                               STA $8C01
                NOP                               NOP
                CLI                               ...
                ...                               ...
```

Die im Programm enthaltenen NOPs sind nicht unbedingt notwendig, wichtig jedoch ist, daß an dieser Stelle in beiden Speicherbänken der selbe OP-Code liegen muß. Anderenfalls kann sich die CPU aufhängen, da sie bedingt durch die Pipeline-technik den nächsten OP-Code noch aus der alten Bank liest.

Das Umschalten mit dieser zweiten Methode ist sehr viel schneller und kann moit geeigneter Software beispielsweise dazu genutzt werden, ein BASIC-Programm durch ein anderes anzuspringen, zwischen beiden Variablen auszutauschen u.v.a.m.

**Reservieren von Speicher**

Wird eine der Banks 2 bis 8 für eigene Anwendung genutzt und soll dennoch nicht auf die RAM-Floppy oder das 8832-Programm verzichtet werden, so muß der jeweilige Speicher geschützt werden. Dies gilt auch für den Fall, daß die RAM-Floppy und das 8832-Programm gleichzeitig betrieben werden sollen.

Das 8832-Programm prüft vor jedem Zugriff, ob der veränderte Interrupt-Vektor noch stimmt. Anderenfalls wird diese Bank nicht angesprungen und ein Drücken der entsprechenden Ziffer bleibt erfolglos. Diese Oberprüfung führt die Routine TEST02 durch. Geprüft werden die Adressen $DF18 und $DF1A, die die Werte $4C und $DF enthalten müssen. Entsprechend verhinderteine Zerstörung dieser Adressen den Zugriff des 8832-Programmes auf diese Bank. Selbstverständlich darf diese Zerstörung nur bei abgeschaltetem Interrupt erfolgen!

Die RAM-Floppy arbeitet nach einem gänzlich anderen Prinzip. Sie kann auf bis zu acht Banks zugreifen, deren Nummer in einer Tabelle ab dem Label FREIBANKS (siehe Seite 3 des Anhangs zur RAM-Floppy im Teil 2 der Bedienungsanleitung) enthalten sein muß.

# 6526 COMPLEX INTERFACE ADAPTER (CIA)

## BESCHREIBUNG

Der Baustein 6526 ist ein Interface-Adapter, mit dem 65XX-Bus kompatibel, mit flexiblem Timing und diversen Ein-/Ausgabemöglichkeiten.

## BESONDERHEITEN

- 16 einzeln programmierbare Ein-/Ausgabeleitungen
- 8- oder 16-Bit-Datentransport mit Handshaking-Betrieb beim Lesen oder Schreiben
- 2 unabhängige, verknüpfbare 16-Bit-Intervalltimer
- 24-Stunden-(AM/PM)-Zeituhr mit programmierbarem Alarm
- 8-Bit-Schieberegister für serielle Ein-/Ausgabe
- 2 TTL-Eingänge können gespeist werden
- CMOS-kompatibel
- 1- oder 2-MHz-Takt

### BESTELLUNGSHINWEISE

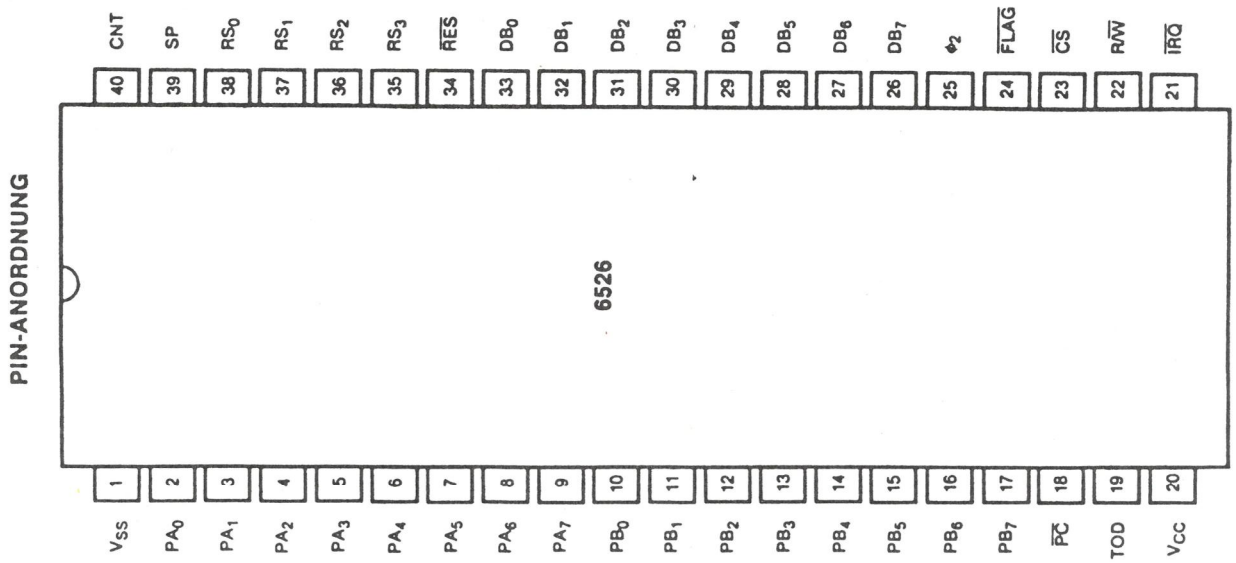
MXS 6526

MXS 6526 ——

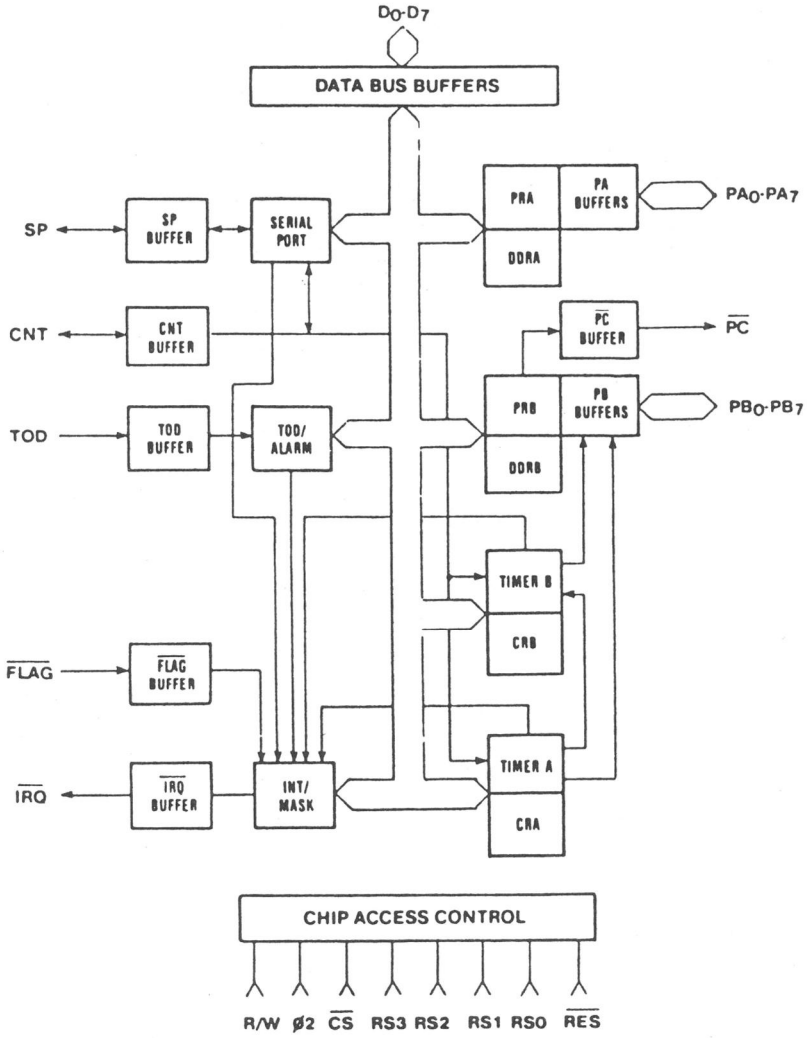
Frequenzbereich
Kein Suffix = 1 MHz
A = 2 MHz

Packet-Bezeichner
C = Keramik
P = Plastik

PIN-ANORDNUNG



## 6526
## BLOCKSCHALTBILD



# FUNKTIONSBESCHREIBUNG

### Ein-/Ausgangsports (PRA, PRB, DDRA, DDRB)

Jeder der beiden Ports A und B bestehen aus einem 8-Bit-Datenregister (PRA bzw. PRB) und einem Datenrichtungsregister (DDRA bzw. DDRB). Ist eines der Bits im DDR Eins gesetzt, wird das entsprechende Bit im PR ausgegeben; ist das Bit im DDR Null, wird das entsprechende Bit als Eingang geschaltet. Beim Lesen stellt das PR das am Ausgang (PA0-PA7, PB0-PB7) gültige Bit dar, unabhängig davon, ob der betreffende Pin als Ausgang oder Eingang geschaltet ist. Beide Ports sind sowohl TTL- als auch CMOS-kompatibel (durch aktive und passive Pullup-Elemente) und können zwei TTL-Einheiten treiben. Zusätzlich zur normalen Funktion übernehmen PB6 und PB7 die Funktion eines Intervalltimer-Ausgangs.

### Handshaking

Dieses Datenübertragungsverfahren kann durch Benutzung des Ausgangs $\overline{PC}$ und des Eingangs $\overline{FLAG}$ realisiert werden. $\overline{PC}$ wird für einen Taktzyklus Low geschaltet, wenn in PRB ein- oder ausgelesen wurde. Dieses Signal kann also als "data ready"- oder "data accepted"-Signal für PRB benutzt werden. (Bei 16-Bit-Datenübertragungen [mit PRA und PRB] würde also PRA zuerst gelesen werden). Der $\overline{FLAG}$-Eingang reagiert auf negative Flanken. Mit ihm kann das $\overline{PC}$-Signal von einem anderen 6526 empfangen werden, oder er wird als Interrupteingang benutzt. Jede negative Flanke an $\overline{FLAG}$ setzt das Flag-Interrupt-Bit 4.

| REG | NAME | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | PRA | $PA_7$ | $PA_6$ | $PA_5$ | $PA_4$ | $PA_3$ | $PA_2$ | $PA_1$ | $PA_0$ |
| 1 | PRB | $PB_7$ | $PB_6$ | $PB_5$ | $PB_4$ | $PB_3$ | $PB_2$ | $PB_1$ | $PB_0$ |
| 2 | DDRA | $DPA_7$ | $DPA_6$ | $DPA_5$ | $DPA_4$ | $DPA_3$ | $DPA_2$ | $DPA_1$ | $DPA_0$ |
| 3 | DDRB | $DPB_7$ | $DPB_6$ | $DPB_5$ | $DPB_4$ | $DPB_3$ | $DPB_2$ | $DPB_1$ | $DPB_0$ |

### INTERVALL-TIMER (TIMER A, TIMER B)

Beide bestehen aus je einem 16-Bit-Intervalltimer (nur Lesen) und einem 16-Bit-Latch (nur Schreiben). Beim Schreiben werden die Daten in das Latch geschrieben, während beim Lesen der Inhalt des Intervalltimers angezeigt wird. Die Timer können sowohl unabhängig voneinander als auch zusammen benutzt werden. Die verschiedenen Betriebsarten erlauben Zeitverzögerungen, variable Impulslängen, Impulsfolgen und Signale unterschiedlicher Frequenz.

Mit dem Eingang CNT können die Zähler externe Impulse zählen und Frequenzen, Impulslängen und Verzögerungszeiten messen. Jeder Zähler hat ein eigenes Kontrollregister zur unabhängigen Überwachung folgender Funktionen:

### START/STOP

Ein Kontrollbit (cb) ermöglicht dem Prozessor, den Zähler zu jeder Zeit zu starten und zu stoppen.

### PB ON/OFF

Ein Kontrollbit steuert die Ausgabe des Zählerüberlaufs an Port B (PB6 für Timer A, PB7 für Timer B). Dieses Bit überschreibt das DDRB-Kontrollbit und schaltet den entsprechenden Pin auf Ausgang.

### TOGGLE/PULSE

Ein Kontrollbit bestimmt die Art des Ausgangssignals, das an Port B erscheint. Am Ende jedes Zählerzyklus (underflow) kann der Ausgang entweder von Low nach High und umgekehrt wechseln, oder ein einzelner positiver Impuls (Länge: 1 Taktzyklus Ø2) erzeugt werden. Der Toggle-Ausgang wird auf High gesetzt, wenn der Zähler gestartet wird. Durch $\overline{RES}$ auf Low gesetzt.

### ONE SHOT/CONTINUOUS

Ein Kontrollbit wählt eine der beiden Betriebsarten. Im One-Shot-Modus wird von dem Wert im Latch bis Null gezählt, ein Interrupt erzeugt, der Wert erneut geladen und der Zähler gestoppt. Im Continuous-Modus wird nicht gestoppt, sondern dieser Vorgang kontinuierlich wiederholt.

### FORCE LOAD

Dieses Strobe-Bit erzwingt, daß der Inhalt des Latches in den Zeitzähler geladen wird, unabhängig davon, ob der Zähler läuft oder nicht.

### INPUT-MODE

Kontrollbits erlauben die Auswahl des Taktes, der zur Dekrementierung des Zählers benutzt wird. Timer A kann Ø2-Taktimpulse oder externe Impulse über CNT zählen. Timer B kann zusätzlich "underflow"-Impulse des Timers A zählen. Zusätzlich besteht eine Steuermöglichkeit über den Pin CNT. Der Inhalt des Latches wird bei jedem Zählerunterlauf, "force load" oder nach einem Schreiben des H-Byte des Latches (bei angehaltenem Zähler) in den Zähler übernommen. Wenn der Zähler läuft, bewirkt das Schreiben des H-Bytes nur ein Laden des Latches, kein Laden des Zählers.

### LESEN (TIMER)

| REG | NAME | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | TA LO | $TAL_7$ | $TAL_6$ | $TAL_5$ | $TAL_4$ | $TAL_3$ | $TAL_2$ | $TAL_1$ | $TAL_0$ |
| 5 | TA HI | $TAH_7$ | $TAH_6$ | $TAH_5$ | $TAH_4$ | $TAH_3$ | $TAH_2$ | $TAH_1$ | $TAH_0$ |
| 6 | TB LO | $TBL_7$ | $TBL_6$ | $TBL_5$ | $TBL_4$ | $TBL_3$ | $TBL_2$ | $TBL_1$ | $TBL_0$ |
| 7 | TB HI | $TBH_7$ | $TBH_6$ | $TBH_5$ | $TBH_4$ | $TBH_3$ | $TBH_2$ | $TBH_1$ | $TBH_0$ |

### SCHREIBEN (VORTEILER)

| REG | NAME | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | TA LO | $PAL_7$ | $PAL_6$ | $PAL_5$ | $PAL_4$ | $PAL_3$ | $PAL_2$ | $PAL_1$ | $PAL_0$ |
| 5 | TA HI | $PAH_7$ | $PAH_6$ | $PAH_5$ | $PAH_4$ | $PAH_3$ | $PAH_2$ | $PAH_1$ | $PAH_0$ |
| 6 | TB LO | $PBL_7$ | $PBL_6$ | $PBL_5$ | $PBL_4$ | $PBL_3$ | $PBL_2$ | $PBL_1$ | $PBL_0$ |
| 7 | TB HI | $PBH_7$ | $PBH_6$ | $PBH_5$ | $PBH_4$ | $PBH_3$ | $PBH_2$ | $PBH_1$ | $PBH_0$ |

**UHRZEIT (TOD – Time Of Day)**

Die TOD-Clock ist ein spezieller Zähler für Echtzeitanwendungen. Sie besteht aus einer 24-Stunden-Uhr mit einer Auflösung von 1/10-Sekunden. Sie ist in 4 Register aufgeteilt:
1/10-Sekunden,
Sekunden,
Minuten,
Stunden.
Das AM/PM-Flag ist das MSB des Stundenregisters, um das Lesen zu vereinfachen. Jedes Register wird im BCD-Code gelesen, damit die Konvertierung für das

Betreiben von Anzeigegeräten vereinfacht wird. Die Uhr benötigt einen Takt mit 50 Hz oder 60 Hz (programmierbar) mit TTL-Pegel. Ein programmierbarer ALARM ist dafür vorgesehen, einen Interrupt zu einer bestimmten Zeit auszulösen. Die zugehörigen Register liegen auf den gleichen Adressen wie die Register der Uhr, der Zugriff auf die ALARM-Register erfolgt über ein Bit im Kontrollregister. In die ALARM-Register kann nur geschrieben werden; ein Leseimpuls auf die Adressen der TOD-Register ergibt immer die Zeit, unabhängig vom Zustand des ALARM-Kontrollbits.
Um die Zeit zu lesen oder zu setzen, muß eine bestimmte Reihenfolge eingehalten werden. TOD wird automatisch gestoppt, wenn ein Schreibimpuls für die Stundenregister gültig wird, und wird erst wieder gestartet, nachdem in die 1/10-Sekunden-Register geschrieben wurde. Dies stellt sicher, daß TOD immer mit der gewünschten Zeit gestartet wird. Da ein Übertrag von einem Register zum nächsten sich auch während eines Lesezyklus ereignen könnte, werden während eines Lesezyklus alle Registerinhalte konstant gehalten (gelatcht). Alle vier Register werden gespeichert, sobald die Stunden gelesen werden, und bleiben gespeichert, bis die 1/10-Sekunden gelesen wurden. Erst danach zeigen die Register die aktuellen Werte. Wenn nur ein Register gelesen werden soll, gibt es kein Problem mit dem Übertrag. Das Register kann sofort gelesen werden. Nach dem Stunden-Register muß aber immer das 1/10-Sekunden-Register gelesen werden, um die Verriegelung aufzuheben.

**LESEN**

| REG | NAME | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 8 | TOD 10THS | 0 | 0 | 0 | 0 | $T_8$ | $T_4$ | $T_2$ | $T_1$ |
| 9 | TOD SEC | 0 | $SH_4$ | $SH_2$ | $SH_1$ | $SL_8$ | $SL_4$ | $SL_2$ | $SL_1$ |
| A | TOD MIN | 0 | $MH_4$ | $MH_2$ | $MH_1$ | $ML_8$ | $ML_4$ | $ML_2$ | $ML_1$ |
| B | TOD HR | PM | 0 | 0 | HH | $HL_8$ | $HL_4$ | $HL_2$ | $HL_1$ |

**SCHREIBEN**

$CRB_7$=0 TOD
$CRB_7$=1 ALARM
(Gleiches Format wie LESEN)

## SERIELLER PORT (SDR)

Dies ist ein gebuffertes, synchrones, 8 Bit breites Schieberegistersystem. Ein Kontrollbit wählt entweder Ein-/oder Ausgabemodus. Im Eingabemodus werden die Daten vom Pin SP, gesteuert durch positive Taktflanken am Pin CNT, in ein Schieberegister geschoben. Nach 8 Impulsen am Eingang CNT werden die Daten in das serielle Datenregister übernommen, und ein Interrupt wird erzeugt. Wird dieser Port als Ausgang benutzt, bestimmt Timer A die Baudrate. Die Daten werden mit 1/2 der Underflowrate von Timer A an Pin SP herausgeschoben. Die größte mögliche Baudrate ist Ø2/4; sie wird aber durch Kabelkapazitäten und der Geschwindigkeit, mit der der Empfänger auf den Dateneingang reagiert, begrenzt.
Die Übertragung beginnt, nachdem in das serielle Datenregister geschrieben wurde (vorausgesetzt, Timer A läuft im Modus CONTINUOUS). Das Taktsignal von Timer A erscheint als Ausgangssignal an Pin CNT. Die Daten aus dem seriellen Datenregister werden in das Schieberegister übernommen und ausgegeben, wenn an CNT ein Impuls erscheint. Die Ausgabe wird mit der negativen Flanke von CNT gültig und bleibt gültig bis zur nächsten negativen Flanke. Nach 8 Impulsen an Pin CNT wird ein Interrupt erzeugt, um anzuzeigen, daß die nächsten Daten übertragen werden können. Falls das serielle Datenregister vor diesem Interrupt mit neuen Daten geladen wurde, werden diese automatisch in das Schieberegister geladen, und die Übertragung wird fortgesetzt. Falls also der Prozessor das Schieberegister rechtzeitig nachlädt, ist die Übertragung kontinuierlich. Wenn keine Daten mehr übertragen werden sollen, erscheint nach 8 Impulsen an CNT an diesem Ausgang ein H-Pegel, und Pin SP bleibt auf dem Pegel, der dem zuletzt übertragenen Bit entspricht.
SDR gibt zuerst das MSB aus, diese Reihenfolge sollte auch bei der Eingabe verwendet werden. Weil die benutzten Pins bidirektional sind, können viele 6526-Bausteine auf einen seriellen Bus zusammengeschaltet werden, wobei einer der Bausteine als Master, der Daten und Takt ausgibt, und alle anderen als Slaves fungieren. Deshalb sind diese Pins "open-drain"-Schaltungen. Die Vorschrift für Verteilung von Master-/Slavefunktion kann über den seriellen Bus oder spezielle Leitungen übertragen werden.

| REG | NAME | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| C | SDR | $S_7$ | $S_6$ | $S_5$ | $S_4$ | $S_3$ | $S_2$ | $S_1$ | $S_0$ |

**INTERRUPT CONTROL (ICR)**

Es gibt 5 mögliche Quellen für einen Interrupt: Underflow von Timer A oder B; TOD ALARM; serieller Port voll/leer und $\overline{\text{FLAG}}$.
Die Maskier- und Interruptinformationen sind in einem Register zusammengefaßt. Das INTERRUPTKONTROLLREGISTER besteht aus einem Maskenregister, in das nur hineingeschrieben werden kann, und einem Datenregister, das nur gelesen werden kann. Jeder Interrupt setzt ein entsprechendes Bit im Datenregister. Wird der Interrupt durch das Maskenregister nicht gesperrt, wird das MSB des Datenregisters gesetzt (IR-Bit) und der Pin $\overline{\text{IRQ}}$ Low geschaltet. Sind mehrere 6526 zusammengeschaltet, können die IR-Bits abgefragt werden, um festzustellen, welcher Baustein den Interrupt ausgelöst hat. Nachdem das Datenregister gelesen wurde, wird es gelöscht und $\overline{\text{IRQ}}$ High gelegt. Da das Datenregister unabhängig vom Maskenregister gesetzt wird und jedes Interruptbit einzeln maskiert werden kann, um einen Interrupt zu verhindern, ist es möglich, Interruptanforderungen und ausgeführte Interrupts zu mischen.
Wenn das Bit IR abgefragt wird, wird das Datenregister gelöscht, die Informationen müssen also vom Benutzer gerettet werden.
Das Maskenregister ermöglicht eine einfache Steuerung der Maskierung. Wenn man in das Register schreibt und das 7. Bit der geschriebenen Daten (SET/$\overline{\text{CLEAR}}$) 0 ist, werden alle Bits, die 1 gesetzt sind, gelöscht, während die Bits, die 0 sind, nicht beeinflußt werden. Falls das 7. Bit der geschriebenen Daten 1 ist, wird jedes Maskierungsbit, das 1 ist, gesetzt, während diejenigen, die 0 sind, nicht berührt werden. Damit IR gesetzt werden und ein Interrupt ausgelöst werden kann, muß das korrespondierende Maskierungsbit gesetzt sein.

**LESEN (INT DATA)**

| REG | NAME | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| D | ICR | IR | 0 | 0 | FLG | SP | ALRM | TB | TA |

**SCHREIBEN (INT MASK)**

| REG | NAME | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| D | ICR | S/$\overline{\text{C}}$ | X | X | FLG | SP | ALRM | TB | TA |

# STEUERREGISTER

Der 6526 hat zwei Steuerregister: CRA und CRB. CRA ist mit TIMER A und CRB mit TIMER B verbunden. Es gilt folgendes Registerformat:

## CRA:

| Bit | Name | Funktion |
|---|---|---|
| 0 | START | 1=START TIMER A, 0=STOP TIMER A. Dieses Bit wird automatisch rückgestellt, wenn es im one-shot-mode zu einem Unterlauf kommt. |
| 1 | PBON | 1=TIMER AUSGABE A liegt an PB6 an, 0=PB6 Normalbetrieb. |
| 2 | OUTMODE | 1=TOGGLE, 0=PULS |
| 3 | RUNMODE | 1=ONE-SHOT, 0=KONTINUIERLICH |
| 4 | LOAD | 1=FORCE LOAD (dies ist eine STROBE-Eingabe. Es erfolgt keine Datenspeicherung, Bit 4 liest stets eine 0, und das Schreiben einer 0 hat keinen Einfluß). |
| 5 | INMODE | 1=TIMER A zählt positive CNT-Übergänge, 0=TIMER A zählt Ø2-Impulse. |
| 6 | SPMODE | 1=AUSGABE SERIELLER PORT, 0=SERIELLER PORT (externer Taktgeber erforderlich). |
| 7 | TODIN | 1=50-Hz-clock am TOD-Pin ergibt korrekte Uhrzeit.<br>0=60-Hz-clock am TOD-Pin ergibt korrekte Uhrzeit. |

## CRB:

| Bit | Name | Funktion |
|---|---|---|
| | | (Bits CRB0–CRB4 entsprechen CRA0–CRA4 von TIMER B. Bit 1 steuert jedoch die TIMER-Ausgabe B auf PB7.) |
| 5, 6 | INMODE | Bits CRB5 und CRB6 wählen eine der vier Eingabemodi von TIMER B: |

| CRB6 | CRB5 | |
|---|---|---|
| 0 | 0 | TIMER B zählt Ø2-Impulse. |
| 0 | 1 | TIMER B zählt positive CNT-Übergänge. |
| 1 | 0 | TIMER B zählt Unterlauf-Impulse von TIMER A. |
| 1 | 1 | TIMER B zählt Unterlauf-Impulse von TIMER A, während CNT H-Pegel hat. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7 | ALARM | 1=ALARM setzen durch Schreiben in TOD-Register, 0=TOD-clock setzen durch Schreiben in TOD-Register. | | | | | | | |

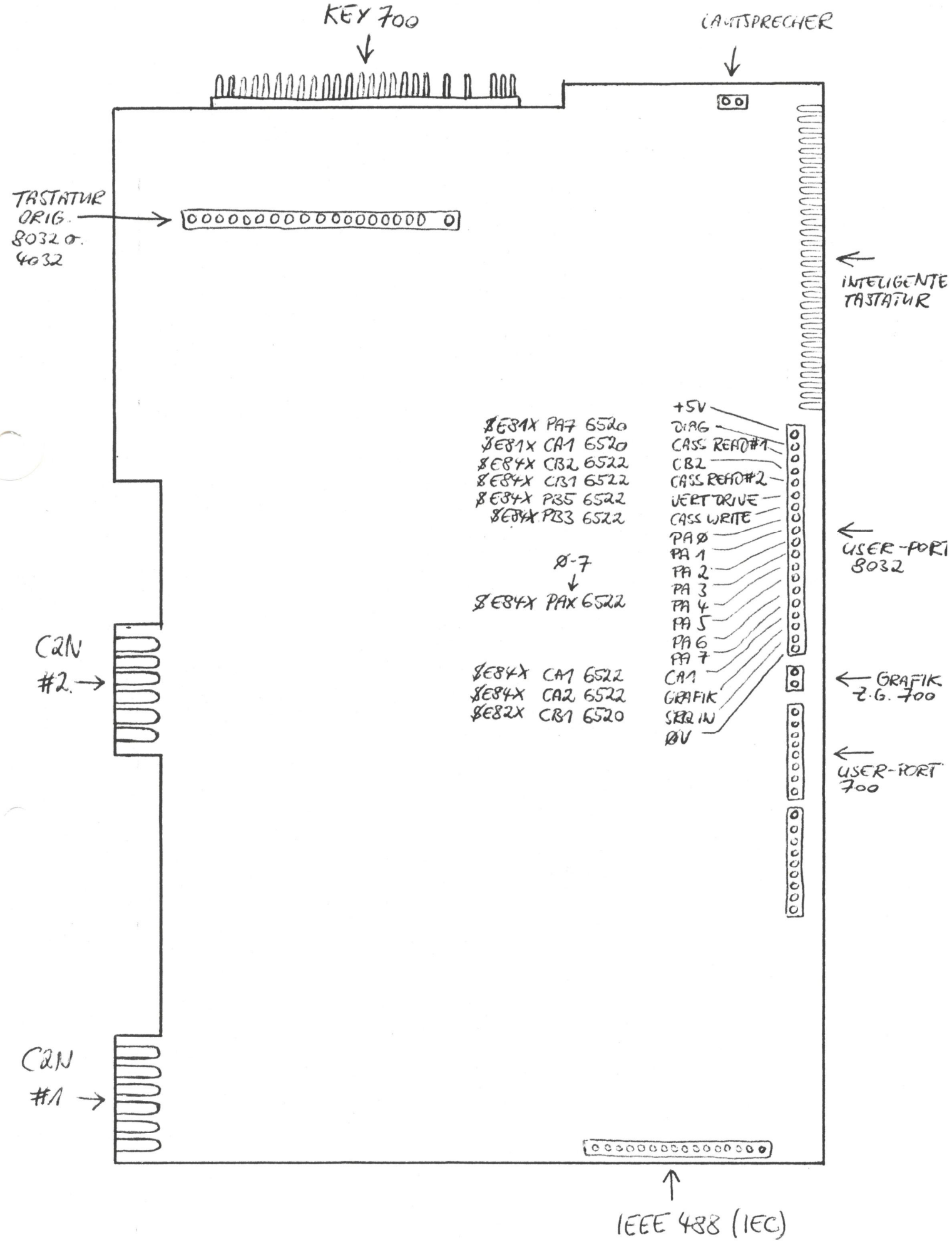
| REG | NAME | TOD IN | SP MODE | IN MODE | LOAD | RUN MODE | OUT MODE | PB ON | START |
|---|---|---|---|---|---|---|---|---|---|
| E | CRA | 0=60Hz<br>1=50Hz | 0=INPUT<br>1=OUTPUT | 0=ϕ2<br>1=CNT | 1=FORCE LOAD (STROBE) | 0=CONT.<br>1=O.S. | 0=PULSE<br>1=TOGGLE | 0=PB$_6$ OFF<br>1=PB$_6$ ON | 0=STOP<br>1=START |

TA

| REG | NAME | ALARM | IN | MODE | LOAD | RUN MODE | OUT MODE | PB ON | START |
|---|---|---|---|---|---|---|---|---|---|
| F | CRB | 0=TOD<br>1=ALARM | 0<br>1<br>1<br>1 | 0=ϕ2<br>1=CNT<br>0=TA<br>1=CNT·TA | 1=FORCE LOAD (STROBE) | 0=CONT.<br>1=O.S. | 0=PULSE<br>1=TOGGLE | 0=PB$_7$ OFF<br>1=PB$_7$ ON | 0=STOP<br>1=START |

TB

Alle nicht benutzten Register-Bits werden durch das Schreiben nicht beeinflußt und beim Lesen auf Null gesetzt


KEY 700
LAUTSPRECHER
TASTATUR ORIG. 8032 o. 4032
INTELIGENTE TASTATUR
USER-PORT 8032
GRAFIK z.B. 700
USER-PORT 700
C2N #2
C2N #1
IEEE 488 (IEC)
$E81X PA7 6520
$E81X CA1 6520
$E84X CB2 6522
$E84X CB1 6522
$E84X PB5 6522
$E84X PB3 6522
Ø-7
$E84X PAX 6522
$E84X CA1 6522
$E84X CA2 6522
$E82X CB1 6520
+5V
DRAG
CASS READ#1
CB2
CASS READ#2
VERT DRIVE
CASS WRITE
PA Ø
PA 1
PA 2
PA 3
PA 4
PA 5
PA 6
PA 7
CA1
GRAFIK
SERQ IN
ØV

F1 F2 F3 F4 F5 F6 F7 F8 F9 F10
INS DEL
CLR HOME
OFF RVS
RUN STOP
ESC
TAB
SHIFT LOCK
SHIFT
CTRL
RETURN
SHIFT
CE
ENTER

## Die Programmierung des 6545

| | CBM 8032 Text-Modus | CBM 8032 Grafik-Modus | CBM 8032 4k Text-Modus II | CBM 8032 4k Grafik-Modus II | CBM 600 Text/Grafik | CBM 700 Text/Grafik | CBM 600 Grafik | CBM 700 neu | TURBO-CBM 8700 DIN-ZG | CBM 700 RVS | CBM 700 (8k) DIN ZG | CBM 700 (8k) ASCII-ZG | CBM 700 40 Zeichen | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BILDSCHIRM HORIZONTAL | 00 | 31 | 31 | | | 7F | 6B | 7E | 6C | 6C | | | | 00 |
| BILDFENSTER HORIZONTAL | 01 | 28 | 28 | | | 50 | 50 | 50 | 50 | 50 | | | 28 | 01 |
| HORIZONTAL SYNCH POSITION | 02 | 29 | 29 | | | 60 | 50 | 62 | 52 | 52 | | | 40 | 02 |
| HORI/VERTI SYNCH WIDTHS | 03 | 0F | 0F | | | 0A | 0F | 0A | 0F | 0F | | | | 03 |
| BILDSCHIRM VERTIKAL | 04 | 27 | 31 | | | 26 | 19 | 1F | 19 | 19 | | | | 04 |
| FEINJUSTIERUNG VERTIKAL | 05 | 00 | 00 | | | 01 | 03 | 06 | 03 | 03 | | | | 05 |
| BILDFENSTER VERTIKAL | 06 | 19 | 19 | | | 19 | 19 | 19 | 19 | 19 | | | | 06 |
| VERTIKAL SYNCH POS. | 07 | 20 | 25 | | | 1E | 19 | 1C | 19 | 19 | | | | 07 |
| STEUERREG.: ALLE REG.BINÄR | 08 | 00 | 00 | | | 00 | 00 | 00 | 00 | 00 | | | | 08 |
| ABTASTZEILEN/ZEICHEN | 09 | 09 | 07 | | | 07 | 0D | 07 | 00 | 00 | | | | 09 |
| ABTASTZEILE CURSOR-OBERGRENZE (BLINKEN $6X) | 0A | 00 | 00 | | | 00 | 60 | 00 | 60 | 10 | | | | 0A |
| ABTASTZEILE CURSOR-UNTERGRENZE | 0B | 00 | 00 | | | 07 | 0D | 07 | 0D | 00 | | | | 0B |
| BILDSCHIRM START ADR. ADDR H | 0C | 10 | 10 | 30 | 30 | 00 | 00 | 00 | 00 | X | 20 | 00 | 10 | 0C |
| BILDSCHIRM START ADR. ADDR L | 0D | 00 | 00 | | | 00 | 00 | 00 | 00 | 00 | | | | 0D |
| CURSOR-POS.IM BILDSCHIRMRAM H | 0E | 00 | 00 | | | 00 | 00 | 00 | 00 | 00 | | | | 0E |
| CURSOR-POS.IM BILDSCHIRMRAM L | 0F | 00 | 00 | | | 00 | 00 | 00 | 00 | 00 | | | | 0F |
| LICHTGRIFFEL IM BILDSCH.RAM H | 10 | 00 | 00 | | | 00 | 00 | 00 | 00 | 00 | | | | 10 |
| LICHTGRIFFEL IM BILDSCH.RAM L | 11 | 00 | 00 | | | 00 | 00 | 00 | 00 | 00 | | | | 11 |

Adressen (a): e736 = 59190
e748 = 59208

(b) e793 = 59283
e7a5 = 59301

für rev. Darstellung:
poke 59190, 32
poke 59208, 32

## Die Tastaturmatrix des TURBO-CBM 8700

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | ; , | ( 8 | I | K | F8 | : . |
| 1 | M | / 7 | U | J | F7 | SPACE |
| 2 | H | % 5 | & 6 | Z | F6 | N |
| 3 | G | $ 4 | R | T | F5 | B |
| 4 | F | § 3 | E | D | F4 | V |
| 5 | X | " 2 | W | S | F3 | C |
| 6 | Y | ! 1 | Q | A | F2 | |
| 7 | SHIFT | ESC | TAB | | F1 | CTRL |
| 8 | ENTER | π / | - | + | STOP | |
| 9 | 3 | ↑ * | 9 | 6 | RVS | ØØ |
| A | 2 | CE | 8 | 5 | HOME | . |
| B | 1 | ? | 7 | 4 | INS DEL | Ø |
| C | ß | CRSR ← | CRSR → | ] > | CRSR ↑ | |
| D | RETURN | - - | [ < | + | CRSR ↓ | ' # |
| E | Ü | = Ø | ? ß | P | F10 | Ä |
| F | Ö | ) 9 | O | L | F9 | — _ |

# 6581 SOUND INTERFACE DEVICE (SID) CHIP SPECIFICATIONS

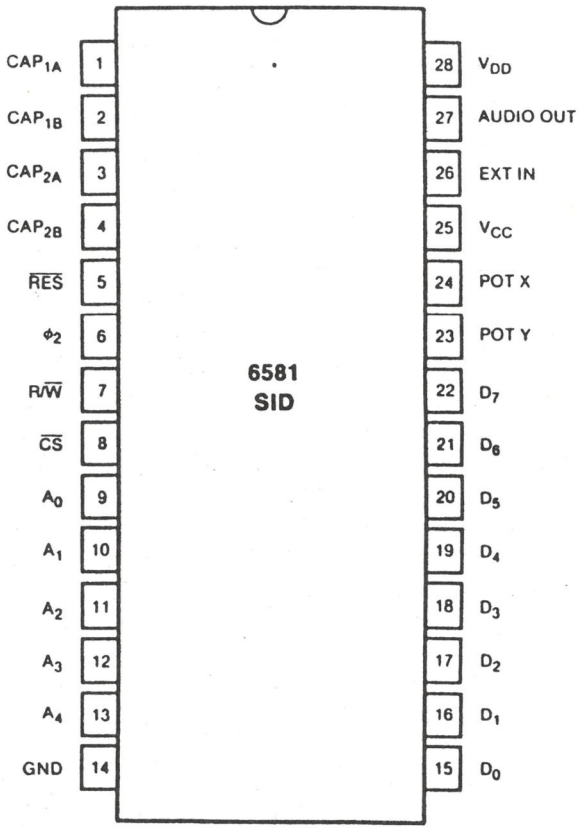
## KONZEPT

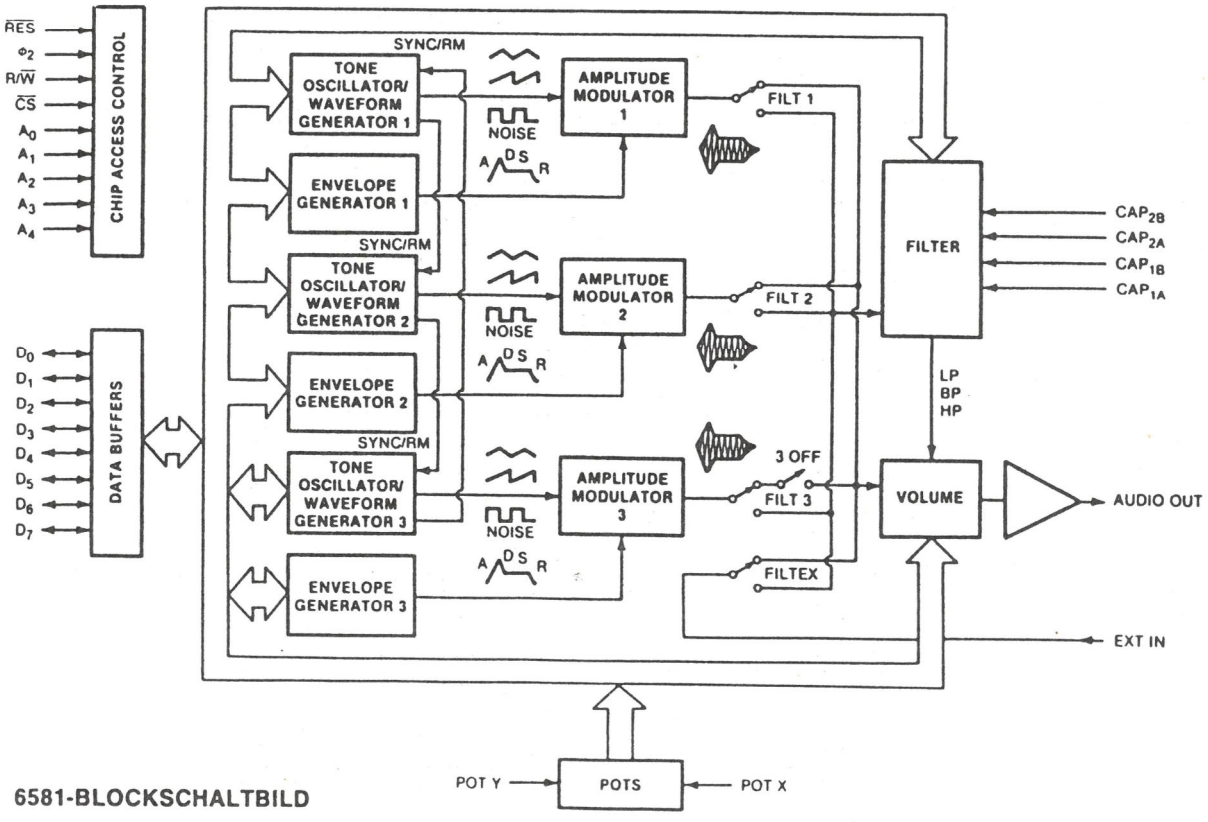
Der SID 6581 ist ein dreistimmiger, elektronischer Musik-/Geräuschgenerator, buskompatibel mit der Prozessorfamilie 65XX und ähnlichen Prozessoren. Die Tonfrequenz kann ebenso wie Klang und Lautstärke in einem weiten Bereich mit hoher Genauigkeit eingestellt werden. Spezielle Schaltkreise verringern die nötige Software, was den Einsatz in Heimcomputern und preiswerten Musikinstrumenten ermöglicht.

## BESONDERHEITEN

- 3 Tongeneratoren, 0–4 kHz
- 4 Kurvenformen pro Generator wählbar:
  Sinus, Dreieck, Rechteck (einstellbar) oder Rauschen
- 3 Amplitudenmodulatoren, jeweils 48 dB
- 3 Hüllkurvengeneratoren
  exponentieller Kurvenverlauf
  Anstiegszeit: 2 ms–8 s
  Abfallzeit: 6 ms–24 s
  Sustain-(Halte-)Pegel: 0–max. Lautstärke
  Ausklingzeit: 6 ms–24 s
- Synchronisierung der Oszillatoren
- Ringmodulation
- Programmierbare Filter
  Eck- bzw. Mittenfrequenz: 30 Hz–12 kHz
  Abfall: 12 dB/Oktave
  Tiefpaß, Bandpaß, Hochpaß oder Notchfilter
- Gesamtlautstärkeeinstellung
- Zufallsgenerator
- Anschlußmöglichkeit für 2 Potentiometer
- Audioeingang

## PIN-ANORDNUNG




**6581-BLOCKSCHALTBILD**

## Tabelle 1 SID-Registerbelegung

| | ADDRESS | | | | | REG # | DATA | | | | | | | | REG NAME | REG TYPE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | (HEX) | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | | |
| | | | | | | | | | | | | | | | **Voice 1** | |
| 0 | 0 | 0 | 0 | 0 | 0 | 00 | $F_7$ | $F_6$ | $F_5$ | $F_4$ | $F_3$ | $F_2$ | $F_1$ | $F_0$ | FREQ LO | WRITE-ONLY |
| 1 | 0 | 0 | 0 | 0 | 1 | 01 | $F_{15}$ | $F_{14}$ | $F_{13}$ | $F_{12}$ | $F_{11}$ | $F_{10}$ | $F_9$ | $F_8$ | FREQ HI | WRITE-ONLY |
| 2 | 0 | 0 | 0 | 1 | 0 | 02 | $PW_7$ | $PW_6$ | $PW_5$ | $PW_4$ | $PW_3$ | $PW_2$ | $PW_1$ | $PW_0$ | PW LO | WRITE-ONLY |
| 3 | 0 | 0 | 0 | 1 | 1 | 03 | — | — | — | — | $PW_{11}$ | $PW_{10}$ | $PW_9$ | $PW_8$ | PW HI | WRITE-ONLY |
| 4 | 0 | 0 | 1 | 0 | 0 | 04 | NOISE |  |  |  | TEST | RING MOD | SYNC | GATE | CONTROL REG | WRITE-ONLY |
| 5 | 0 | 0 | 1 | 0 | 1 | 05 | $ATK_3$ | $ATK_2$ | $ATK_1$ | $ATK_0$ | $DCY_3$ | $DCY_2$ | $DCY_1$ | $DCY_0$ | ATTACK/DECAY | WRITE-ONLY |
| 6 | 0 | 0 | 1 | 1 | 0 | 06 | $STN_3$ | $STN_2$ | $STN_1$ | $STN_0$ | $RLS_3$ | $RLS_2$ | $RLS_1$ | $RLS_0$ | SUSTAIN/RELEASE | WRITE-ONLY |
| | | | | | | | | | | | | | | | **Voice 2** | |
| 7 | 0 | 0 | 1 | 1 | 1 | 07 | $F_7$ | $F_6$ | $F_5$ | $F_4$ | $F_3$ | $F_2$ | $F_1$ | $F_0$ | FREQ LO | WRITE-ONLY |
| 8 | 0 | 1 | 0 | 0 | 0 | 08 | $F_{15}$ | $F_{14}$ | $F_{13}$ | $F_{12}$ | $F_{11}$ | $F_{10}$ | $F_9$ | $F_8$ | FREQ HI | WRITE-ONLY |
| 9 | 0 | 1 | 0 | 0 | 1 | 09 | $PW_7$ | $PW_6$ | $PW_5$ | $PW_4$ | $PW_3$ | $PW_2$ | $PW_1$ | $PW_0$ | PW LO | WRITE-ONLY |
| 10 | 0 | 1 | 0 | 1 | 0 | 0A | — | — | — | — | $PW_{11}$ | $PW_{10}$ | $PW_9$ | $PW_8$ | PW HI | WRITE-ONLY |
| 11 | 0 | 1 | 0 | 1 | 1 | 0B | NOISE |  |  |  | TEST | RING MOD | SYNC | GATE | CONTROL REG | WRITE-ONLY |
| 12 | 0 | 1 | 1 | 0 | 0 | 0C | $ATK_3$ | $ATK_2$ | $ATK_1$ | $ATK_0$ | $DCY_3$ | $DCY_2$ | $DCY_1$ | $DCY_0$ | ATTACK/DECAY | WRITE-ONLY |
| 13 | 0 | 1 | 1 | 0 | 1 | 0D | $STN_3$ | $STN_2$ | $STN_1$ | $STN_0$ | $RLS_3$ | $RLS_2$ | $RLS_1$ | $RLS_0$ | SUSTAIN/RELEASE | WRITE-ONLY |
| | | | | | | | | | | | | | | | **Voice 3** | |
| 14 | 0 | 1 | 1 | 1 | 0 | 0E | $F_7$ | $F_6$ | $F_5$ | $F_4$ | $F_3$ | $F_2$ | $F_1$ | $F_0$ | FREQ LO | WRITE-ONLY |
| 15 | 0 | 1 | 1 | 1 | 1 | 0F | $F_{15}$ | $F_{14}$ | $F_{13}$ | $F_{12}$ | $F_{11}$ | $F_{10}$ | $F_9$ | $F_8$ | FREQ HI | WRITE-ONLY |
| 16 | 1 | 0 | 0 | 0 | 0 | 10 | $PW_7$ | $PW_6$ | $PW_5$ | $PW_4$ | $PW_3$ | $PW_2$ | $PW_1$ | $PW_0$ | PW LO | WRITE-ONLY |
| 17 | 1 | 0 | 0 | 0 | 1 | 11 | — | — | — | — | $PW_{11}$ | $PW_{10}$ | $PW_9$ | $PW_8$ | PW HI | WRITE-ONLY |
| 18 | 1 | 0 | 0 | 1 | 0 | 12 | NOISE |  |  |  | TEST | RING MOD | SYNC | GATE | CONTROL REG | WRITE-ONLY |
| 19 | 1 | 0 | 0 | 1 | 1 | 13 | $ATK_3$ | $ATK_2$ | $ATK_1$ | $ATK_0$ | $DCY_3$ | $DCY_2$ | $DCY_1$ | $DCY_0$ | ATTACK/DECAY | WRITE-ONLY |
| 20 | 1 | 0 | 1 | 0 | 0 | 14 | $STN_3$ | $STN_2$ | $STN_1$ | $STN_0$ | $RLS_3$ | $RLS_2$ | $RLS_1$ | $RLS_0$ | SUSTAIN/RELEASE | WRITE-ONLY |
| | | | | | | | | | | | | | | | **Filter** | |
| 21 | 1 | 0 | 1 | 0 | 1 | 15 | — | — | — | — | — | $FC_2$ | $FC_1$ | $FC_0$ | FC LO | WRITE-ONLY |
| 22 | 1 | 0 | 1 | 1 | 0 | 16 | $FC_{10}$ | $FC_9$ | $FC_8$ | $FC_7$ | $FC_6$ | $FC_5$ | $FC_4$ | $FC_3$ | FC HI | WRITE-ONLY |
| 23 | 1 | 0 | 1 | 1 | 1 | 17 | $RES_3$ | $RES_2$ | $RES_1$ | $RES_0$ | FILTEX | FILT 3 | FILT 2 | FILT 1 | RES/FILT | WRITE-ONLY |
| 24 | 1 | 1 | 0 | 0 | 0 | 18 | 3 OFF | HP | BP | LP | $VOL_3$ | $VOL_2$ | $VOL_1$ | $VOL_0$ | MODE/VOL | WRITE-ONLY |
| | | | | | | | | | | | | | | | **Misc.** | |
| 25 | 1 | 1 | 0 | 0 | 1 | 19 | $PX_7$ | $PX_6$ | $PX_5$ | $PX_4$ | $PX_3$ | $PX_2$ | $PX_1$ | $PX_0$ | POT X | READ-ONLY |
| 26 | 1 | 1 | 0 | 1 | 0 | 1A | $PY_7$ | $PY_6$ | $PY_5$ | $PY_4$ | $PY_3$ | $PY_2$ | $PY_1$ | $PY_0$ | POT Y | READ-ONLY |
| 27 | 1 | 1 | 0 | 1 | 1 | 1B | $O_7$ | $O_6$ | $O_5$ | $O_4$ | $O_3$ | $O_2$ | $O_1$ | $O_0$ | $OSC_3$/RANDOM | READ-ONLY |
| 28 | 1 | 1 | 1 | 0 | 0 | 1C | $E_7$ | $E_6$ | $E_5$ | $E_4$ | $E_3$ | $E_2$ | $E_1$ | $E_0$ | $ENV_3$ | READ-ONLY |

# KONTROLLREGISTER

## GENERATOR 1

### FREQUENZ LOW/FREQUENZ HIGH (00,01)

Zusammen stellen diese Register ein 16-Bit-Wort dar, welches die Frequenz des 1. Oszillators nach folgender Gleichung festlegt:

$$F_{out} = (F_n * F_{clK}/16777216)Hz = (F_n * 0{,}0587214734)Hz$$

$F_n$ ist die 16-Bit-Zahl aus den Registern, $F_{clK}$ ist der Systemtakt, der am Eingang Ø2 anliegt.
Dadurch kann die Tonhöhe ohne wahrnehmbare Tonschritte durchgestimmt werden.

### PW LO/PW HI (02,03)

Diese Register bilden eine 12-Bit-Zahl (Bit 4–7 von PW HI werden nicht genutzt), welche das Tastverhältnis des Rechteckgenerators 1 bestimmt. Das Verhältnis errechnet sich wie folgt:

$$PW_{out} = (PW_n/40{,}95)\%$$

$PW_n$ ist hier die 12-Bit-Zahl in den PW-Registern. Das Tastverhältnis kann so ohne wahrnehmbare Schritte verändert werden. Diese Register haben nur dann einen hörbaren Effekt, wenn der Rechteckgenerator 1 eingeschaltet ist. Wenn in den Registern 0 oder 4095 steht, entsteht ein DC-Signal, 2048 ergibt dagegen ein Rechteck mit 50% Tastverhältnis.

### KONTROLLREGISTER (04)

Dieses Register enthält 8 Kontrollbits:

#### GATE (Bit 0)

Steuert den Hüllkurvengenerator. Wenn es 1 gesetzt ist, beginnt der Zyklus Attack/Decay/Sustain. Wenn es 0 gesetzt wird, beginnt der Zyklus Release (genauere Erklärung im Kapitel Hüllkurvengenerator).

#### SYNC (Bit 1)

Wenn dieses Bit 1 gesetzt ist, wird die Frequenz des Generators 1 mit der Frequenz des Generators 3 synchronisiert ("Hard-Sync"-Effekte).
Wenn die Frequenz des Generators 1 unter Berücksichtigung der Frequenz des Generators 3 variiert wird, entsteht eine große Zahl komplexer harmonischer Strukturen. Wenn Sync funktionieren soll, muß die Frequenz des dritten Generators kleiner als die des ersten Generators sein (nicht 0). Keine anderen Parameter der 3. Stimme beeinflussen Sync.

#### RING MOD (Bit 2)

Wenn dieses Bit 1 gesetzt ist, wird der Dreieckgenerator der 1. Stimme durch eine mit Frequenz 1 und 3 ringmodulierte Spannung ersetzt. Wenn jetzt die Frequenz 1 verändert wird, entstehen nichtharmonische Obertöne, welche für Klingel- oder Gonggeräusche und Spezialeffekte gebraucht werden. Hierfür muß bei Generator 1 Dreieck und bei Generator 3 eine Frequenz größer als Null eingestellt sein. Andere Parameter der 3. Stimme haben keine Wirkung.

**TEST (Bit 3)**

Wenn dieses Bit 1 gesetzt ist, wird der 1. Generator zurückgesetzt und auf 0 gehalten, bis das Testbit gelöscht ist. Der Rauschgenerator ist abgestellt, und der Rechteckgenerator wird auf DC gehalten. Zwar wird dieses Bit normalerweise für Testzwecke benutzt, es kann jedoch Generator 1 auch mit externen Ereignissen synchronisieren (kompliziertere Kurvenformen, Realzeit-Verarbeitung).

**BIT 4**

Wenn dieses Bit gesetzt ist, ist der Dreieckgenerator eingeschaltet. Diese Kurvenform ist arm an Obertönen und hat einen weichen, einer Flöte ähnlichen Charakter.

**BIT 5**

Wenn dieses Bit gesetzt ist, ist der Sägezahngenerator eingeschaltet. Dieser ist reich an geraden und ungeraden Obertönen und ergibt einen breiten, an Blechbläser erinnernden Klang.

**BIT 6**

Wenn dieses Bit gesetzt ist, ist der Rechteckgenerator ausgewählt. Der Obertonanteil kann durch das Tastverhältnis eingestellt werden, die Möglichkeiten reichen vom hellen, hohlen Rechteckklang bis zum nasalen, schrillen Klang kurzer Impulse. Wenn das Tastverhältnis beim Spielen verändert wird, entsteht ein "pashing"-Effekt, der den Eindruck einer Bewegung erweckt. Schnelles Hin- und Herschalten zwischen verschiedenen Tastverhältnissen kann interessante Sequenzen erzeugen.

**BIT 7**

Wenn dieses Bit gesetzt ist, ist der Rauschgenerator eingeschaltet.
Dieser produziert Rauschen, das die Klangfarbe vom tiefen Rumpeln bis zum zischenden weißen Rauschen durch die Frequenzeinstellung des Generators 1 verändern kann. Rauschen braucht man, um Explosionen, Gewehrschüsse, Düsenjäger, Wind und ähnliche Geräusche zu erzeugen, oder für Trommeln und Becken. Indem man die Frequenz beim Spielen verändert, kann man Stürme nachbilden. Obwohl einer dieser Generatoren eingeschaltet sein muß, um die 1. Stimme am Ausgang erklingen zu lassen, ist es nicht notwendig, die einzelnen Generatoren auszuschalten, um die Stimme abzustellen. Die Lautstärke wird nur durch den Hüllkurvengenerator bestimmt.

**Bemerkung:** Die Oszillatorausgänge können nicht addiert werden. Wenn mehr als ein Oszillator eingeschaltet ist, wird das Ergebnis eine logische "Und"-Verknüpfung der Kurvenform sein. Obwohl damit neue Kurvenformen erzeugt werden können, sollte dies vorsichtig benutzt werden. Wenn Rauschen eingeschaltet ist und zusätzlich eine Kurvenform eingeschaltet wird, verstummt das Rauschen, bis das Testbild zurückgesetzt oder der Pin 5 (RES) Low geschaltet wird.

## STIMME 2

Die Register $07–$0D kontrollieren die Stimme 2 und haben die gleiche Funktion wie die Register 00–06, mit folgenden Ausnahmen:

1) SYNC synchronisiert den Generator 2 mit Generator 1.
2) RING MOD ersetzt die Dreieckspannung durch die ringmodulierte Kombination der Generatoren 1 und 2.

## STIMME 3

Die Register $0E–$14 haben für die 3. Stimme die gleiche Funktion wie die Register 00–06, mit folgenden Ausnahmen:

1) SYNC synchronisiert Generator 3 mit Generator 2.
2) RING MOD ersetzt die Dreieckspannung durch ringmodulierte Kombination der Generatoren 2 und 3.

Wenn man einen Ton ansprechen will, muß man also Frequenz, Kurvenform, Effekte (SYNC, RING MOD) und Hüllkurve bestimmen. Dann kann man den Ton jederzeit mit dem Gatebit abrufen. Der Ton hält solange an, bis das Gatebit zurückgesetzt wird. Jede Stimme kann einzeln, mit unterschiedlichen Parametern oder mit anderen Stimmen zusammen benutzt werden, um eine einzelne, kräftige Stimme zu erhalten. Dabei kann eine leichte Verstimmung der Oszillatoren untereinander oder die Stimmung in musikalischen Intervallen einen wirkungsvollen Effekt ergeben.

### ATTACK/DECAY (05)

Bit 4–7 wählt eine von 16 möglichen Anstiegszeiten (Attack) für den Hüllkurvengenerator der 1. Stimme. Dies bestimmt, wie schnell der Ausgang auf volle Lautstärke anschwillt, wenn der Hüllkurvengenerator eingeschaltet wird (Gate).
Bit 0 bis 3 wählen eine von 16 möglichen Abschwellzeiten (Decay) aus. Diese Zeit gibt an, wie schnell die Lautstärke vom Spitzenwert auf den ausgewählten Haltepegel (Sustain) abfällt.

### SUSTAIN/RELEASE (06)

Bit 4–7 wählt einen von 16 möglichen Halte-(Sustain-)Pegeln des Hüllkurvengenerators aus. Diese Phase folgt dem Abfall, der Pegel wird gehalten, solange das Gatebit gesetzt ist. Der Pegel kann von Stille (0) bis zur Spitzenlautstärke (16) linear eingestellt werden. Ein Wert von 8 würde demnach der halben Lautstärke, die beim Anstieg (Attack) erreicht wird, entsprechen.
Mit Bit 0–3 kann eine der 16 Ausklingarten gewählt werden. Der Ausklingzyklus folgt der Haltezeit, wenn das Gatebit zurückgesetzt wird. Dann fällt die Lautstärke vom Haltepegel auf Null in der eingestellten Zeit. Die Ausklingzeiten mit den Werten 0–16 sind identisch mit den Abfallzeiten 0–16.

**Bemerkung:** Der geschilderte Ablauf kann ohne Einschränkung jederzeit durch das Gatebit verändert werden. Wenn das Gatebit z. B. zurückgesetzt wird, bevor die Anschlagszeit abgelaufen ist, beginnt sofort bei dem erreichten Pegel die Ausklingzeit. Wenn jetzt das Gatebit wieder gesetzt wird, beginnt sofort eine neue Anstiegszeit bei dem jetzt erreichten Pegel. Dadurch können komplizierte Amplitudenverläufe durch Realzeitprogrammierung erzeugt werden.

**Tabelle 2 Hüllkurvenraten**

| WERT | | ANSTIEGSRATE | ABKLING/ABFALLRATE |
|---|---|---|---|
| DEZIMAL | (HEX) | (Takt/Zyklus) | (Takt/Zyklus) |
| 0 | (0) | 2 ms | 6 ms |
| 1 | (1) | 8 ms | 24 ms |
| 2 | (2) | 16 ms | 48 ms |
| 3 | (3) | 24 ms | 72 ms |
| 4 | (4) | 38 ms | 114 ms |
| 5 | (5) | 56 ms | 168 ms |
| 6 | (6) | 68 ms | 204 ms |
| 7 | (7) | 80 ms | 240 ms |
| 8 | (8) | 100 ms | 300 ms |
| 9 | (9) | 250 ms | 750 ms |
| 10 | (A) | 500 ms | 1.5 s |
| 11 | (B) | 800 ms | 2.4 s |
| 12 | (C) | 1 s | 3 s |
| 13 | (D) | 3 s | 9 s |
| 14 | (E) | 5 s | 15 s |
| 15 | (F) | 8 s | 24 s |

**Bemerkung zur Tabelle:** Die angegebenen Werte beziehen sich auf eine Taktfrequenz von 1 MHz. Wenn die Taktfrequenz abweicht, müssen die Werte mit 1 MHz/F(clk) multipliziert werden.
Die angegebenen Zeiten beziehen sich auf die Zeit, die benötigt wird, um den Zyklus abzuschließen. Eine Anstiegszeit von 16 ms (Wert 2) bedeutet z. B., daß die Lautstärke nach 16 ms von Pegel 0 den Spitzenwert erreicht. Die Abfall-/Ausklingzeiten beziehen sich auf die Zeit, die benötigt wird, um vom Spitzenwert auf Null zu sinken.

## FILTERREGISTER

### FC LO/FC HI (Register $15,$16)

Diese Register bilden zusammen eine 11-Bit-Zahl (Bit 3–7 des Registers FC LO werden nicht genutzt). Diese bestimmt linear die Mitten- bzw. Eckfrequenz, sie kann von 30 Hz bis 12 kHz eingestellt werden.

### RES/FILT (Register $17)

Bit 4–7 dieses Registers bestimmen die Resonanz des Filters. Dieser Effekt hebt die Frequenzen in der Nähe der Eckfrequenz an, dadurch ergibt sich ein schärferer Klang. Es können 16 verschiedene Einstellungen vorgenommen werden (linear von 0 bis 16). Bit 0–3 legt fest, welches Signal gefiltert wird:

**FILT 1 (Bit 0):**

Eine 0 in diesem Register bedeutet, daß die Stimme 1 ohne Veränderung auf den Audioausgang geschaltet wird (Bypass). Wenn es gesetzt ist, wird die 1. Stimme gefiltert, ihr Obertonanteil verändert sich.

**FILT 2 (Bit 1):**

Gleiche Wirkung wie Bit 0 für die 2. Stimme.

**FILT 3 (Bit 2):**

Gleiche Wirkung wie Bit 0 für die 3. Stimme.

**FILTEX (Bit 3):**

Gleiche Wirkung wie Bit 0 für den Audioeingang.

### MODE/VOL (Register $18)

Bits 4–7 bestimmen verschiedene Filter- und Ausgabearten:

**LP (Bit 4):** Wenn dieses Bit gesetzt ist, ist der Tiefpaß eingeschaltet, d. h. alle Frequenzen unterhalb der Eckfrequenz bleiben unverändert, alle Frequenzen oberhalb werden mit 12 dB/Oktave abgeschwächt. Es entstehen volle Klänge.

**BP (Bit 5):**

Das gleiche für den Bandpaß. Alle Frequenzen unter und oberhalb der Mittenfrequenz werden mit 6 dB/Oktave abgeschwächt. Es entstehen offene, dünne Klänge.

**HP (Bit 6):**

Das gleiche für den Hochpaß. Alle Frequenzen oberhalb der Eckfrequenz bleiben unverändert, unterhalb werden sie mit 12 dB/Oktave abgeschwächt. Es entstehen summende und blecherne Klänge.

**3 OFF (Bit 7):**

Eins gesetzt, trennt dieses Bit die 3. Stimme vom Audioausgang ab. Wenn man Stimme 3 am Filter vorbei schaltet (mit FILT 3=0) und 3 OFF gesetzt ist, wird die 3. Stimme nicht auf den Ausgang geschaltet, kann aber zur Modulation der anderen Stimmen benutzt werden.

**Bemerkung:** Die Filter können zusammengeschaltet werden. Z. B. ergibt LP zusammen mit HP ein Notchfilter (Bandsperre). Damit der Filtereffekt hörbar wird, muß ein Filter eingeschaltet sein und eine Stimme durch das Filter geführt werden. Das Filter ist vielleicht das wichtigste Element im SID, da es durch die subtraktive Synthese viele Klangmöglichkeiten schafft (das Filter entzieht dem obertonreichen Eingangssignal bestimmte Frequenzen). Gute Ergebnisse erzielt man, wenn man die Eck- bzw. Mittenfrequenz während des Spielens variiert.

**VOL 0—VOL 3 (Bit 0—3):**
Hiermit wird die Gesamtlautstärke zwischen 0 (leise) und 15 (laut) in linearen Stufen eingestellt. Hiermit kann die Lautstärke beim Zusammenschalten mehrerer Chips abgestimmt oder Effekte wie Tremolo erzeugt werden. Bei VOL=0 ist der Ausgang stumm.

## WEITERE EIGENSCHAFTEN

### POTX (Register $19)

Dieses Register erlaubt dem Prozessor, die Position eines Potentiometers, das an Pin 24 angeschlossen ist, in Schritten von 0 bei kleinstem Widerstand bis 255 bei vollem Widerstand zu erkennen. Das Ergebnis liegt immer vor und wird alle 512 Takte erneuert.

### POTY (Register $1A)

Das gleiche für ein zweites Potentiometer (an Pin 23).

### OSC 3/RANDOM (Register $1B)

Dieses Register erlaubt dem Prozessor, die 8 oberen Bits des Ausgangs von Oszillator 3 zu lesen. Die Art der Ziffernfolge, die entsteht, ist direkt mit der Kurvenform verknüpft. Beim Sägezahn wächst die Zahlenfolge von 0 bis 255, um dann wieder bei 0 zu beginnen. Beim Dreieck wächst die Zahl von 0 bis 255, um dann von 255 bis 0 zu fallen. Beim Rechteck springt die Zahl zwischen 0 und 255 hin und her. Beim Rauschen wird eine Kette von Zufallszahlen erzeugt, deshalb kann dieses Register auch als Zufallszahlengenerator benutzt werden. Es gibt viele Anwendungsmöglichkeiten für dieses Register, die wichtigste ist vielleicht die Steuerung von Modulationen. Die Zahlen, die erzeugt werden, können per Software zum Inhalt der Oszillator- oder Filterfrequenzregister addiert werden etc. So können viele dynamische Effekte erzeugt werden: Sirenen, indem OSC3 (Sägezahn) zum Frequenzregister eines anderen Oszillators addiert wird. Vibrato entsteht, wenn OSC3 (Dreieck, 7 Hz) zum Frequenzregister einer anderen Stimme addiert wird. Dabei sollte der Audioausgang der 3. Stimme abgeschaltet sein (3OFF=1).

### ENV 3 (Register $1C)

Im Prinzip das gleiche wie OSC3, es wird jedoch der Ausgang des Hüllkurvengenerators 3 gelesen. Die Zahlen können z. B. zum Inhalt des Filterfrequenzregisters addiert werden, es entstehen sog. "Harmonische Hüllkurven" und Wahwah-Effekte. "Phasing" entsteht, wenn dieser Ausgang zum Frequenzregister eines Oszillators addiert wird. Um dieses Signal zu erzeugen, muß das Gatebit geschaltet werden. Der Ausgang OSC3 spiegelt immer die Veränderungen am Ausgang des 3. Oszillators wider, er wird nicht vom Hüllkurvengenerator beeinflußt.

# SID TONLEITER

Im Anhang sind alle Werte aufgelistet, die in die Frequenzregister eingeschrieben werden müssen, um die Töne einer "wohltemperierten" Tonleiter zu erhalten. Diese besteht aus einer Oktave mit 12 Halbschritten: C, D, E, F, G, A, H, C und C#, D#, F#, G#, A#. Die Frequenz jedes Halbtones läßt sich durch Multiplikation der Frequenz des vorigen Halbtones mit der 12. Wurzel aus 2 errechnen. Der Tabelle liegt ein Systemtakt von 1,02 MHz zugrunde. Für andere Taktfrequenzen muß man die bei den Frequenzregistern angegebene Umrechnung anwenden. Die angegebene Stimmung bezieht sich auf A4 = 440 Hz. Es ist möglich, eine andere Stimmung zu verwenden oder diese Tonfolge umzustellen.
Obwohl dies eine einfache und schnelle Methode ist, die Tonleiter zu programmieren, werden allein zur Speicherung dieser Tabelle 192 Bytes benötigt. Diese Verschwendung des Speicherplatzes kann durch einen Algorithmus umgangen werden, mit dem die Notenwerte berechnet werden können. Da eine Oktave die Verdoppelung der Frequenz bedeutet, brauchen nur die 12 Notenwerte einer Oktave gespeichert zu werden. Wenn diese 12 Eingaben (24 Bytes) aus den Werten für die 8. Oktave bestehen (C7–H7), kann der Wert für jede beliebige Note errechnet werden, indem die Frequenz des entsprechenden Tones der 8. Oktave für jede Oktave Unterschied einmal durch 2 geteilt wird. Eine Division durch 2 ist in binärer Darstellung eine Verschiebung um ein Bit nach rechts. Deshalb kann die Berechnung durch eine einfache Routine durchgeführt werden. Obwohl die Frequenz von H7 von dem Oszillator nicht gebildet werden kann, sollte sie zur Berechnung in die Tabelle aufgenommen werden.
Für jeden Ton muß nun festgelegt werden, um welchen Halbton es sich handelt und in welcher Oktave er erklingen soll. Da man 4 Bit braucht, um 1 von 12 Halbtönen zu wählen, und 3 Bit, um eine von 8 Oktaven zu bestimmen, reicht ein Byte aus. Die unteren 4 Bit bestimmen z. B. den Halbton (sie adressieren einen Platz der Tabelle) und die oberen 4 Bit, um wieviel Stellen der Tabellenwert nach rechts verschoben werden muß.

# MUSIKNOTENWERTE

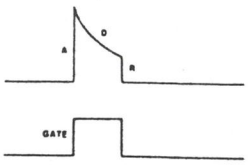
In diesem Anhang finden Sie eine vollständige Liste der Noten, zugehörigen Frequenzen und Frequenzparameter und der Werte, die in die Register FREQ HI und FREQ LO des Klangchips gePOKEt werden müssen, um den gewünschten Ton zu erzeugen.

| NOTE | OKTAVE | DEZIMAL | HI | LOW |
|---|---|---|---|---|
| 0 | C-0 | 278 | 1 | 22 |
| 1 | C#-0 | 295 | 1 | 39 |
| 2 | D-0 | 313 | 1 | 57 |
| 3 | D#-0 | 331 | 1 | 75 |
| 4 | E-0 | 351 | 1 | 95 |
| 5 | F-0 | 372 | 1 | 116 |
| 6 | F#-0 | 394 | 1 | 138 |
| 7 | G-0 | 417 | 1 | 161 |
| 8 | G#-0 | 442 | 1 | 186 |
| 9 | A-0 | 468 | 1 | 212 |
| 10 | A#-0 | 496 | 1 | 240 |
| 11 | H-0 | 526 | 2 | 14 |
| 12 | C-1 | 557 | 2 | 45 |
| 13 | C#-1 | 590 | 2 | 78 |
| 14 | D-1 | 625 | 2 | 113 |
| 15 | D#-1 | 662 | 2 | 150 |
| 16 | E-1 | 702 | 2 | 190 |
| 17 | F-1 | 743 | 2 | 231 |
| 18 | F#-1 | 788 | 3 | 20 |
| 19 | G-1 | 834 | 3 | 66 |
| 20 | G#-1 | 884 | 3 | 116 |
| 21 | A-1 | 937 | 3 | 169 |
| 22 | A#-1 | 992 | 3 | 224 |
| 23 | H-1 | 1051 | 4 | 27 |
| 24 | C-2 | 1114 | 4 | 90 |
| 25 | C#-2 | 1180 | 4 | 156 |
| 26 | D-2 | 1250 | 4 | 226 |
| 27 | D#-2 | 1325 | 5 | 45 |
| 28 | E-2 | 1403 | 5 | 123 |
| 29 | F-2 | 1487 | 5 | 207 |
| 30 | F#-2 | 1575 | 6 | 39 |
| 31 | G-2 | 1669 | 6 | 133 |
| 32 | G#-2 | 1768 | 6 | 232 |
| 33 | A-2 | 1873 | 7 | 81 |
| 34 | A#-2 | 1985 | 7 | 193 |
| 35 | H-2 | 2103 | 8 | 55 |
| 36 | C-3 | 2228 | 8 | 180 |
| 37 | C#-3 | 2360 | 9 | 56 |
| 38 | D-3 | 2500 | 9 | 196 |
| 39 | D#-3 | 2649 | 10 | 89 |
| 40 | E-3 | 2807 | 10 | 247 |
| 41 | F-3 | 2974 | 11 | 158 |
| 42 | F#-3 | 3150 | 12 | 78 |
| 43 | G-3 | 3338 | 13 | 10 |
| 44 | G#-3 | 3536 | 13 | 208 |
| 45 | A-3 | 3746 | 14 | 162 |
| 46 | A#-3 | 3969 | 15 | 129 |
| 47 | H-3 | 4205 | 16 | 109 |

| NOTE | OKTAVE | DEZIMAL | HI | LOW |
|---|---|---|---|---|
| 48 | C-4 | 4455 | 17 | 103 |
| 49 | C#-4 | 4720 | 18 | 112 |
| 50 | D-4 | 5001 | 19 | 137 |
| 51 | D#-4 | 5298 | 20 | 178 |
| 52 | E-4 | 5613 | 21 | 237 |
| 53 | F-4 | 5947 | 23 | 59 |
| 54 | F#-4 | 6301 | 24 | 157 |
| 55 | G-4 | 6676 | 26 | 20 |
| 56 | G#-4 | 7072 | 27 | 160 |
| 57 | A-4 | 7493 | 29 | 69 |
| 58 | A#-4 | 7939 | 31 | 3 |
| 59 | H-4 | 8411 | 32 | 219 |
| 60 | C-5 | 8911 | 34 | 207 |
| 61 | C#-5 | 9441 | 36 | 225 |
| 62 | D-5 | 10002 | 39 | 18 |
| 63 | D#-5 | 10597 | 41 | 101 |
| 64 | E-5 | 11227 | 43 | 219 |
| 65 | F-5 | 11894 | 46 | 118 |
| 66 | F#-5 | 12602 | 49 | 58 |
| 67 | G-5 | 13351 | 52 | 39 |
| 68 | G#-5 | 14145 | 55 | 65 |
| 69 | A-5 | 14986 | 58 | 138 |
| 70 | A#-5 | 15877 | 62 | 5 |
| 71 | H-5 | 16821 | 65 | 181 |
| 72 | C-6 | 17821 | 69 | 157 |
| 73 | C#-6 | 18881 | 73 | 193 |
| 74 | D-6 | 20004 | 78 | 36 |
| 75 | D#-6 | 21193 | 82 | 201 |
| 76 | E-6 | 22454 | 87 | 182 |
| 77 | F-6 | 23789 | 92 | 237 |
| 78 | F#-6 | 25203 | 98 | 115 |
| 79 | G-6 | 26702 | 104 | 78 |
| 80 | G#-6 | 28290 | 110 | 130 |
| 81 | A-6 | 29972 | 117 | 20 |
| 82 | A#-6 | 31754 | 124 | 10 |
| 83 | H-6 | 33642 | 131 | 106 |
| 84 | C-7 | 35643 | 139 | 59 |
| 85 | C#-7 | 37762 | 147 | 130 |
| 86 | D-7 | 40008 | 156 | 72 |
| 87 | D#-7 | 42387 | 165 | 147 |
| 88 | E-7 | 44907 | 175 | 107 |
| 89 | F-7 | 47578 | 185 | 218 |
| 90 | F#-7 | 50407 | 196 | 231 |
| 91 | G-7 | 53404 | 208 | 156 |
| 92 | G#-7 | 56580 | 221 | 4 |
| 93 | A-7 | 59944 | 234 | 40 |
| 94 | A#-7 | 63508 | 248 | 20 |

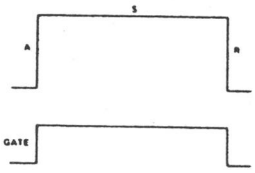
Der Amplitudenverlauf von Klavieren ist komplizierter, er kann aber mit dem ADSR leicht erzeugt werden. Der Ton erreicht seine volle Lautstärke, wenn die Taste angeschlagen wird, und beginnt dann abzuschwellen. Wenn die Taste losgelassen wird, wird der Ton durch die Mechanik abgedämpft. Diese Hüllkurve ist hier dargestellt:

| | | |
|---|---|---|
| **ATTACK:** | 0 | 2 ms |
| **DECAY:** | 9 | 750 ms |
| **SUSTAIN:** | 0 | |
| **RELEASE:** | 0 | 6 ms |



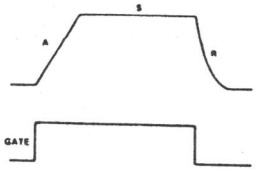

Man beachte, daß der Ton abklingt, bis das Gatebit zurückgesetzt und dann abgestellt wird.
Die einfachste Hüllkurve ist die einer Orgel: Solange eine Taste gedrückt ist, hat der Ton volle Lautstärke und wird sofort abgestellt, wenn die Taste wieder losgelassen wird.
Diese Hüllkurve ist hier dargestellt:

| | | |
|---|---|---|
| **ATTACK:** | 0 | 2 ms |
| **DECAY:** | 0 | 6 ms |
| **SUSTAIN:** | 15 ($F) | |
| **RELEASE:** | 0 | 6 ms |




Die wirkliche Stärke des SID liegt aber in der Erzeugung künstlicher Klänge. Der ADSR kann Hüllkurven erzeugen, die bei keinem Instrument vorkommen. Ein gutes Beispiel ist hierfür die "Rückwärts"-Hüllkurve. Sie wird von einem langsamen Anstieg und einem scharfen Abfall bestimmt, was so klingt, als hätte man das Instrument auf Tonband aufgenommen und würde die Aufnahme rückwärts abspielen. Sie sieht folgendermaßen aus:

| | | |
|---|---|---|
| **ATTACK:** | 10 ($A) | 500 ms |
| **DECAY:** | 0 | 6 ms |
| **SUSTAIN:** | 15 ($F) | |
| **RELEASE:** | 3 | 72 ms |




Viele bemerkenswerte Klänge entstehen, wenn der Hüllkurvenverlauf des einen Instrumentes mit dem Klang eines anderen kombiniert wird. Dadurch entstehen Klänge, die bekannten Instrumenten ähneln, aber irgendwie fremd klingen. Da Klänge im allgemeinen subjektiv empfunden werden, muß man mit verschiedenen Klangfarben und Hüllkurven experimentieren, bis man den gewünschten Klang erhält.

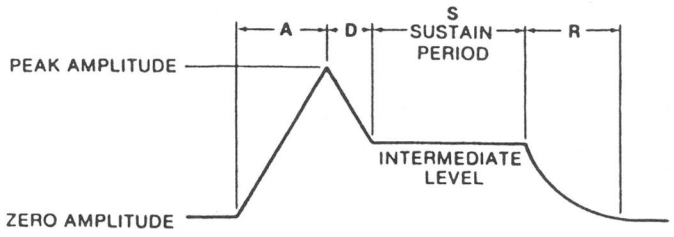
## FILTEREINSTELLUNGEN

| Adresse | Inhalt |
|---|---|
| | Grenzfrequenz, Low Byte (0–7)<br>Grenzfrequenz, High Byte (0–255)<br>Resonanz (Bits 4–7)<br>Filter, Stimme 3 (Bit 2)<br>Filter, Stimme 2 (Bit 1)<br>Filter, Stimme 1 (Bit 0)<br>Hochpaß (Bit 6)<br>Bandpaß (Bit 5)<br>Tiefpaß (Bit 4)<br>Lautstärke (Bits 0–3) |

# SID HÜLLKURVENGENERATOR

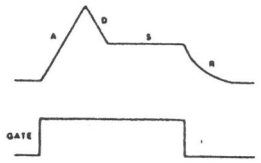
## HÜLLKURVENGENERATOR

Der vierteilige ADSR (Attack, Decay, Sustain, Release) hat sich in der elektronischen Musik als optimaler Kompromiß zwischen Flexibilität und einfacher Bedienung erwiesen. Passende Wahl der Parameter erlaubt es, eine Vielzahl von Instrumenten nachzuahmen.
Die Geige ist ein gutes Beispiel für ein Instrument mit lang anhaltendem Ton: Er schwillt langsam an, erreicht eine Spitzenlautstärke und fällt dann auf einen niedrigeren Wert ab. Der Geiger kann diesen Ton lange halten, um ihn dann langsam ausklingen zu lassen. Ein "Schnappschuß" dieser Hüllkurve zeigt dieses Bild:




Diese Hüllkurve kann folgendermaßen nachgebildet werden:

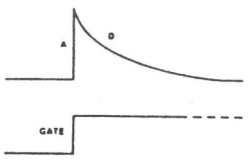
| | | |
|---|---|---|
| **ATTACK:** | 10 ($A) | 500 ms |
| **DECAY:** | 8 | 300 ms |
| **SUSTAIN:** | 10 ($A) | |
| **RELEASE:** | 9 | 750 ms |




Man beachte, daß der Ton solange anhält, bis das Gatebit zurückgesetzt wird. Mit wenigen Änderungen kann diese Hüllkurve für Blech- und Holzblasinstrumente und alle Streichinstrumente verwendet werden.
Eine ganz andere Hüllkurve besitzen Schlag- und Tasteninstrumente. Die Hüllkurve von Schlaginstrumenten wird von einem nahezu augenblicklichen Anstieg und einem darauf folgenden Abfall bestimmt, diese Instrumente können den Ton nicht auf einer konstanten Lautstärke halten. Eine Trommel erreicht in dem Moment, in dem sie angeschlagen wird, ihre volle Lautstärke, um dann schnell auszuklingen.

Die typische Hüllkurve eines Beckens wird hier gezeigt:

| | | |
|---|---|---|
| **ATTACK:** | 0 | 2ms |
| **DECAY:** | 9 | 750ms |
| **SUSTAIN:** | 0 | |
| **RELEASE:** | 9 | 750ms |




Man beachte, daß der Ton vollkommen ausklingt, obwohl das Gatebit nicht zurückgesetzt wird.

# T E I L 2

## Der 8832-Betriebsmodus der PROXA-7000-Platine

Viele Arbeitsvorgänge können durch einen Computer vereinfacht oder beschleunigt werden, doch erweist es sich zumeist als ausgesprochen nachteilig, daß der Computer durch eine Aufgabe bereits voll ausgelastet ist. Die Investition in ein zweites Computersystem steht in vielen Fällen in keinem Verhältnis zur Auslastung. In solchen Anwendungsfällen ist es oftmals wünschenswert, ein gerade laufendes Programm zu unterbrechen, ein anderes Programm zu verwenden, um danach in das erste Programm zurückzukehren, als hätte man es nie verlassen. Genau diese Möglichkeit eröffnet Ihnen das vorliegende Software-Paket; es macht aus Ihrem CBM 700 mit einer Proxa-7000-Platine zwei (CBM 710), vier (CBM 720) oder acht (nach Einbau einer Speichererweiterung) Rechner vom Typ CBM 8032. Anstelle der beiden ersten Rechner CBM 8032 ist auch der Betrieb eines CBM 8096 möglich.

Der Commodore 720 verfügt über insgesamt 256 kByte RAM, die beim Betrieb mit der Proxa-7000-Platine als CBM 8032 nur zu einem Viertel genutzt werden. Dieser Speicher ist aufgeteilt in vier Banks (CBM 710: zwei Banks), von denen stets nur eine aktiv sein kann. In dem so brachliegenden Speicher werden drei weitere Betriebssysteme installiert, die jeweils über einen eigenen Arbeitsspeicher, einen separaten Bildschirmspeicher, etc. verfügen. Diese insgesamt vier Betriebssysteme sind völlig unabhängig voneinander, so daß biespielsweise beim Betrieb der Bank 1 die Daten der anderen Banks unverändert bleiben. Per Tastendruck kann zwischen den Banks gewechselt werden, was durch ein auf dem Bildschirm erscheinendes Menu vereinfacht wird.

Es gibt zwei Versionen dieses Programmes. Die erste Version befindet sich in ROMs, die als Cartridge von hinten auf den Rechner aufgesteckt werden. Die zweite Version wird auf Diskette ausgeliefert und mit SHIFT-RUN in den Rechner geladen. Während sich die erste Version bereits nach dem Einschalten des Rechners mit einem Menu meldet, erfordert die Ladezeit der Diskettenversion etwas Geduld.

## Laden und Starten des Diskettenversion

Nachdem Sie die Diskette mittels SHIFT-RUN gestartet haben erscheint das Titelbild auf dem Schirm und die Meldung: 'Das system wird initialisiert'. Nun lädt der Rechner selbständig alle benötigten Programmmodule von Diskette und installiert bis zu acht Betriebssysteme im Rechner. Danach erscheint das Menue.

## Starten der ROM-Version

Die ROM-Version startet selbständig nach dem Einschalten des Rechners. Sie haben die Wahl zwischen 7 Betriebsarten:

1.) Original CBM 700-Betriebssystem
Sie werden keinen Unterschied zu Ihrem Original-Rechner feststellen. Alle CBM 700 Software ist weiterhin unverändert lauffähig.

2.) CBM 8032-ASCII-Betriebssystem
Wünschen Sie eine ASCII-Tastatur und lediglich einen CBM 8032, so wählen Sie die 2. Dies ist zum Beispiel sinnvoll bei der Verwendung einer RAM-Disk, die nicht gleichzeitig mit dem 8832-Modus betrieben werden darf.

3.) CBM 8032-DIN-Betriebssystem
Wie 2.), jedoch mit einer DIN-Tastatur.

4.) CBM 8832-ASCII-Betriebssystem
Initialisieren von maximal 8 ASCII-Betriebssystemen in allen verfügbaren Banks. Diese sind wie im nächsten Abschnitt dieser Bedienungsanleitung beschrieben umschaltbar.

5.) CBM 8832-DIN-Betriebssystem
Wie 4.), jedoch mit einer DIN-Tastatur.

6.) LOS-96-ASCII-Betriebssystem
Um den Rechner in diesem Modus betreiben zu können, muß das Laufwerk 0 Ihrer Floppy eine Diskette mit dem LOS-96-Betriebssystem enthalten. Dieses wird sodann geladen und gestartet.

7.) LOS-96-DIN-Betriebssystem
Wie 6.), jedoch mit einer DIN-Tastatur.

Eingaben dürfen nur über den Zehnerblock der Tastatur erfolgen. Eingaben über die Haupttastatur des Rechners werden ignoriert.

## Umschalten der Banks im 8832-Modus

Das Umschalten der Banks im 8832-Modus erfolgt über ein Menue. In das Menue gelangt man durch drücken der geshifteten Commodoretaste C= . Zur Auswahl einer Bank wählen Sie bitte ausschließlich den Zehnerblock der Tastatur und betätigen diese Eingabe mit ENTER. Waren in dieser Bank noch Files offen, so werden diese geschlossen, wenn Sie die Taste C betätigen. Auf offene Files weist der Rechner mit einer Meldung hin.

```
1000 rem*** 8832 - modus ***
1010 print"■■■" tab(20) "";
1020 print"  ■   ■       ■   ■       ■       ■     ■      ■"
1025 print" ■ ■   ■ ■     ■ ■   ■ ■           ■ ■    ■ ■     ■ ■"
1030 print"■ ■     ■ ■    ■ ■    ■ ■         ■ ■          ■ ■"
1035 print"■ ■     ■ ■    ■ ■    ■ ■        ■ ■           ■ ■"
1040 print"■ ■     ■ ■    ■ ■    ■ ■       ■ ■           ■ ■"
1045 print" ■ ■   ■ ■     ■ ■   ■ ■       ■ ■           ■ ■"
1050 print"  ■   ■       ■   ■       ■     ■       ■ ■"
1055 print" ■ ■   ■ ■     ■ ■   ■ ■         ■ ■     ■ ■"
1060 print"■ ■     ■ ■    ■ ■    ■ ■         ■ ■    ■ ■"
1062 print"■ ■     ■ ■    ■ ■    ■ ■         ■ ■    ■ ■"
1065 print"■ ■     ■ ■    ■ ■    ■ ■        ■ ■     ■ ■"
1070 print" ■ ■   ■ ■     ■ ■   ■ ■        ■ ■     ■ ■"
1075 print"  ■   ■       ■   ■       ■     ■       ■"
1080 print"■■■■■■■■■■■■■■■■■":fori=0to79:print"-";:next:printtab(20)"
1085 print"ultra electronic Helmut Proxa GmbH&Co.KG
1090 print"Postfach 45 12 46            5000 Koeln 1
1095 print"■■■     Das System wird initialisiert"
1100 rem*** menue ***
1200 rem*** initialisierung ***
1210 bload"bs8032",b1,p36864:rem*** betriebssystem 8032 ***
1220 bload"es-din",b1,p57344:rem***  e-sockel fuer din  ***
1230 bload"st6509",b15,p1024:rem*** 6509 start (bank15) ***
1240 bload"st6512",b1,p57112:rem*** 6512-irq   (bankxx) ***
1250 bank15:sys1027
```

```
 ;******************************
 ;***  8832 - startprogramm  ***
 ;******************************
 ;
 ;******************************
 ;***  Hans - Peter   Brill ***
 ;***  Hans-Boeckler-Str. 3 ***
 ;***  5190  Stolberg/Rhld. ***
 ;***  Telefon:(02402) 6148 ***
 ;******************************
 ;
 ;******************************
 ;*** Definition der verwen- ***
 ;***      deten Labels      ***
 ;******************************
i6509          = $01                    ;Data-Indirection-Register
ind            = $f0                    ;$f0/$f1 IND-Vektor
bank           = $f2                    ;zuletzt gewaehlte Bank
zwis           = $f3
files          = $f4
screen         = $8000                  ;6509 Bildschirmadresse
reset          = $fffc                  ;6512 RESET-Vektor
*              = $0400
 ;
```

```
;
;*******************************
;***      Bank-Wechsel       ***
;*******************************
                jmp menue
;
;*******************************
;***   8 Systeme initial.    ***
;*******************************
                sei
                .blck
                lda #$01
                sta i6509
                ldy #$00                ;Bereich $8800 - $87ff
                sty ind                 ;(Bildschirmspeicher)
                lda #$80                ;loeschen um Flackern
                sta ind+1               ;zu vermeiden !
                lda #' '
loop            sta (ind),y
                iny
                bne loop
                inc ind+1
                ldx ind+1
                cpx #$88
                bne loop
                .bend
;
;*******************************
;***   IRQ-Vektor aufbiegen  ***
;*******************************
                ldy #$00
                lda #<$e442             ;IRQ-Vektor verbiegen
                sta ind
                lda #>$e442
                sta ind+1
                lda #$4c                ;Op-Code JMP
                sta (ind),y
                iny
                lda #<$df18             ;neuer IRQ-Vektor
                sta (ind),y
                iny
                lda #>$df18
                sta (ind),y
;
```

```
;
;*******************************
;*** Betriebssytem kopieren ***
;*******************************
                .blck
                ldx #8
                ldy #$00                ;Bereich $8800 - $ffff
                sty ind                 ;von Bank 1 in die
                lda #$80                ;Bank 2,3 und 4
                sta ind+1               ;kopieren
loop            lda #$01                ;Bank 1
                sta i6509
                lda (ind),y             ;lesen
                .blck
loop            inc i6509               ;Bank 2 - 8
                sta (ind),y             ;schreiben
                cpx i6509
                bne loop
                .bend
                iny
                bne loop
                inc ind+1
                bne loop
                .bend
;
;*******************************
;***     8432 Kaltstart      ***
;*******************************
                lda #<menuekalt         ;Umschaltvektor setzen
                sta $03f8
                lda #>menuekalt
                sta $03f9
                lda #$0f                ;RESET 6512 Bank 15
                sta bank
;
;*******************************
;*** 6509 abschalten         ***
;*** 6512 RESET und Start    ***
;*******************************
start6512       .blck
                cli                     ;angefallene IRQ abarbeiten
                lda #$00
                sta $de05               ;IRQ 6525 off
                sei
                sta $df03               ;Tastatur 6525 off
                sta $df04               ;Port a+b input
                sta $dc03               ;6526 = input
                lda bank
                ora #$10                ;aktivieren des Fensters
                sta $dc01               ;Select Bank x
                lda #$ff
                sta $dc03               ;RESET 6512
                .byte $02               ;KILL 6509
                .bend
;
```

```
 ;
 ;*******************************
 ;***     Texte fuer Menu     ***
 ;*******************************
text1           .byte '1. 8032 oder 8096 Grundbank  . . . . (1)'
text2           .byte '2. 8032 (nicht bei 8096!)  . . . . . (2)'
text3           .byte '3. 8032  . . . . . . . . . . . . . . (3)'
text4           .byte '4. 8032  . . . . . . . . . . . . . . (4)'
text5           .byte '5.-8. vier weitere 8032  . . . . . . (.)'
text6           .byte 'Bitte Kennziffer eingeben  . . . . . ( )'
text7           .byte 'Achtung: Noch   Files offen !  (c=close)'
 ;
 ;*******************************
 ;***    Titelbild-Maske      ***
 ;*******************************
titelmaske      .byte %00111000,%00000111,%00000011,%11111000,%00111111
                .byte %01000100,%00001000,%10000000,%00001000,%01000001
                .byte %10000010,%00010000,%01000000,%00010000,%00000001
                .byte %10000010,%00010000,%01000000,%00100000,%00000001
                .byte %10000010,%00010000,%01000000,%01000000,%00000010
                .byte %01000100,%00001000,%10000000,%10000000,%00000100
                .byte %00111000,%00000111,%00000001,%11111000,%00001000
                .byte %01000100,%00001000,%10000000,%00001000,%00010000
                .byte %10000010,%00010000,%01000000,%00001000,%00100000
                .byte %10000010,%00010000,%01000000,%00001000,%01000000
                .byte %10000010,%00010000,%01000000,%00010000,%01000000
                .byte %01000100,%00001000,%10000000,%00100000,%01000000
                .byte %00111000,%00000111,%00000011,%11000000,%01111111
 ;
```

```
;
;*******************************
;***      Text ausgeben      ***
;*******************************
text            .blck                           ;Texte auf den Schirm bringen
                ldx #text2-text1-1
loop            lda text1,x
                jsr ascbsc
                sta screen+20+1280,x
                lda text2,x
                jsr ascbsc
                sta screen+20+1360,x
                lda text3,x
                jsr ascbsc
                sta screen+20+1440,x
                lda text4,x
                jsr ascbsc
                sta screen+20+1520,x
                lda text5,x
                jsr ascbsc
                sta screen+20+1600,x
                lda text6,x
                jsr ascbsc
                sta screen+20+1680,x
                lda files
                beq skip
                lda text7,x
                cpx #14
                bne skip0
                lda files
                ora #'0'
skip0           jsr ascbsc
                ora #$80
                sta screen+20+1840,x
skip            dex
                bpl loop
                rts
                .bend
;
;*******************************
;***  ASCII -> Bildschirm    ***
;*******************************
ascbsc          .blck
                sta zwis
                and #$3f
                bit zwis
                bpl skip
                ora #$40
skip            rts
                .bend
;
```

```
 ;
 ;*******************************
 ;***   Titelbild aufbauen    ***
 ;*******************************
titel           .blck
                lda #<screen+20                 ;Startadresse Titelbild
                sta ind
                lda #>screen
                sta ind+1
                ldx #0                          ;Anzahl der Zeilen
titelloopx      ldy #5
titelloopy      jsr titelbyte
                inx
                dey
                bne titelloopy
                lda ind
                clc
                adc #40
                sta ind
                bcc titelskip
                inc ind+1
titelskip       cpx #65
                bne titelloopx
                rts
titelbyte       tya                             ;Unterprogramm zur Ausgabe
                pha                             ;von 5 Zeichen aus der
                ldy #0                          ;Maske auf den Schirm
                lda titelmaske,x
loop            asl a
                pha
                lda #' '
                bcc skip0
                lda #160
skip0           sta (ind),y
                pla
                iny
                cpy #8
                bne loop
                lda ind
                clc
                adc #8
                sta ind
                bcc skip1
                inc ind+1
skip1           pla
                tay
                rts
                .bend
 ;
```

```
 ;
 ;*****************************
 ;***  Bildschirm loeschen   ***
 ;*****************************
cls             .blck
                ldx #250
                lda #' '
loop            sta screen-1,x
                sta screen-1+250,x
                sta screen-1+500,x
                sta screen-1+750,x
                sta screen-1+1000,x
                sta screen-1+1250,x
                sta screen-1+1500,x
                sta screen-1+1750,x
                dex
                bne loop
                rts
                .bend
 ;
 ;*****************************
 ;***      Linie ziehen     ***
 ;*****************************
line            .blck
                ldx #79
                lda #'-'
loop            sta screen+1120,x
                dex
                bpl loop
                rts
                .bend
 ;
 ;*****************************
 ;***     Portansteuerung   ***
 ;*****************************
getkey          .blck
                sty $e810
skip            lda $e812
                cmp $e812
                bne skip
                cmp #$fe
                rts
                .bend
 ;
```

```
 ;
 ;*****************************
 ;***          Menu          ***
 ;*****************************
menuekalt       lda #<start6512             ;Initialisierung des Menues
                sta $03f8
                lda #>start6512             ;Zeiger fuer spaeteren RESET
                sta $03f9
                lda #$01                    ;zuerst Bank 1 anwaehlen
                sta bank
                lda #$00                    ;Adressierung der Daten-
                sta $e811                   ;richtungsregister
                sta $e813                   ;des 6520 (Tastaturansteuerung)
                sta $e812                   ;8 Bit Input
                lda #$0f                    ;Bits 0-3 Output
                sta $e810                   ;Bits 4-7 Input
                lda #$04                    ;Adressierung der Daten-
                sta $e811                   ;register des
                sta $e813                   ;6520 (Tastaturansteuerung)
                ldx #$00
menue           cld
                stx files
                jsr cls                     ;Bildaufbau
                jsr titel
                jsr line
                jsr text
                lda bank                    ;Vorgabewert im menue
                ora #'0'
                sta screen+1680+58
 ;
```

```
 ;
 ;*******************************
 ;***     Tastaturabfrage     ***
 ;*******************************
tastatur        ldy #11
                ldx #1                          ;Taste 1 Zehnerblock
                jsr getkey
                beq keygot
                ldx #4                          ;Taste 4 Zehnerblock
                cmp #$f7
                beq keygot
                ldx #7                          ;Taste 7 Zehnerblock
                cmp #$fb
                beq keygot
                dey
                ldx #2                          ;Taste 2 Zehnerblock
                jsr getkey
                beq keygot
                ldx #5                          ;Taste 5 Zehnerblock
                cmp #$f7
                beq keygot
                ldx #8                          ;Taste 8 Zehnerblock
                cmp #$fb
                beq keygot
                dey
                ldx #3                          ;Taste 3 Zehnerblock
                jsr getkey
                beq keygot
                cmp #$f7
                bne return
                ldx #6                          ;Taste 6 Zehnerblock
;
keygot          stx zwis
                jsr test02
                bcs tastatur
                lda zwis
                ora #'0'
                sta screen+1680+58
                bne tastatur
return          dey
                jsr getkey                      ;Taste ENTER
                bne close
                lda screen+1680+58              ;RETURN detected
                and #%00001111
                sta bank
;
                .blck
                ldx #swend-switch-1
loop            lda switch,x                    ;6512-Startroutine kopieren
                sta $87e0,x
                dex
                bpl loop
                .bend
                jsr wait4ms
                .blck
loop            jsr getkey
                beq loop
                jmp $87e0                       ;Ansprung SWITCH-Routine
                .bend
 ;
```

```
 ;
 ;*******************************
 ;***   Aufruf CLOSE Files    ***
 ;*******************************
close           lda files                       ;Files geoeffnet ?
                beq tastatur
                sta $87ff
                ldy #$05                        ;Taste c gedrueckt ?
                jsr getkey
                cmp #$df
                bne tastatur
                .blck
                ldx #closeend-closeanf-1
loop            lda closeanf,x
                sta $87d0,x
                dex
                bpl loop
                .bend
                .blck
                ldx #40
                lda #' '
loop            sta screen+20+1840,x
                dex
                bpl loop
                .bend
                jmp $87d0
 ;
 ;*******************************
 ;***      CLOSE - Routine    ***
 ;*******************************
closeanf        .blck
                lda bank
                ora #$10
                sta $8c01
loop            lda $0251
                sta $d2                         ;lf
                jsr $f2e0                       ;close
                dec $87ff
                bne loop
                lda #$1f
                sta $8c01
                jmp tastatur
                .bend
closeend
 ;
 ;*******************************
 ;***      SWITCH-Routine     ***
 ;*******************************
 switch         lda bank
                ora #$10
                sta $8c01
                jmp $df1b
swend
 ;
```

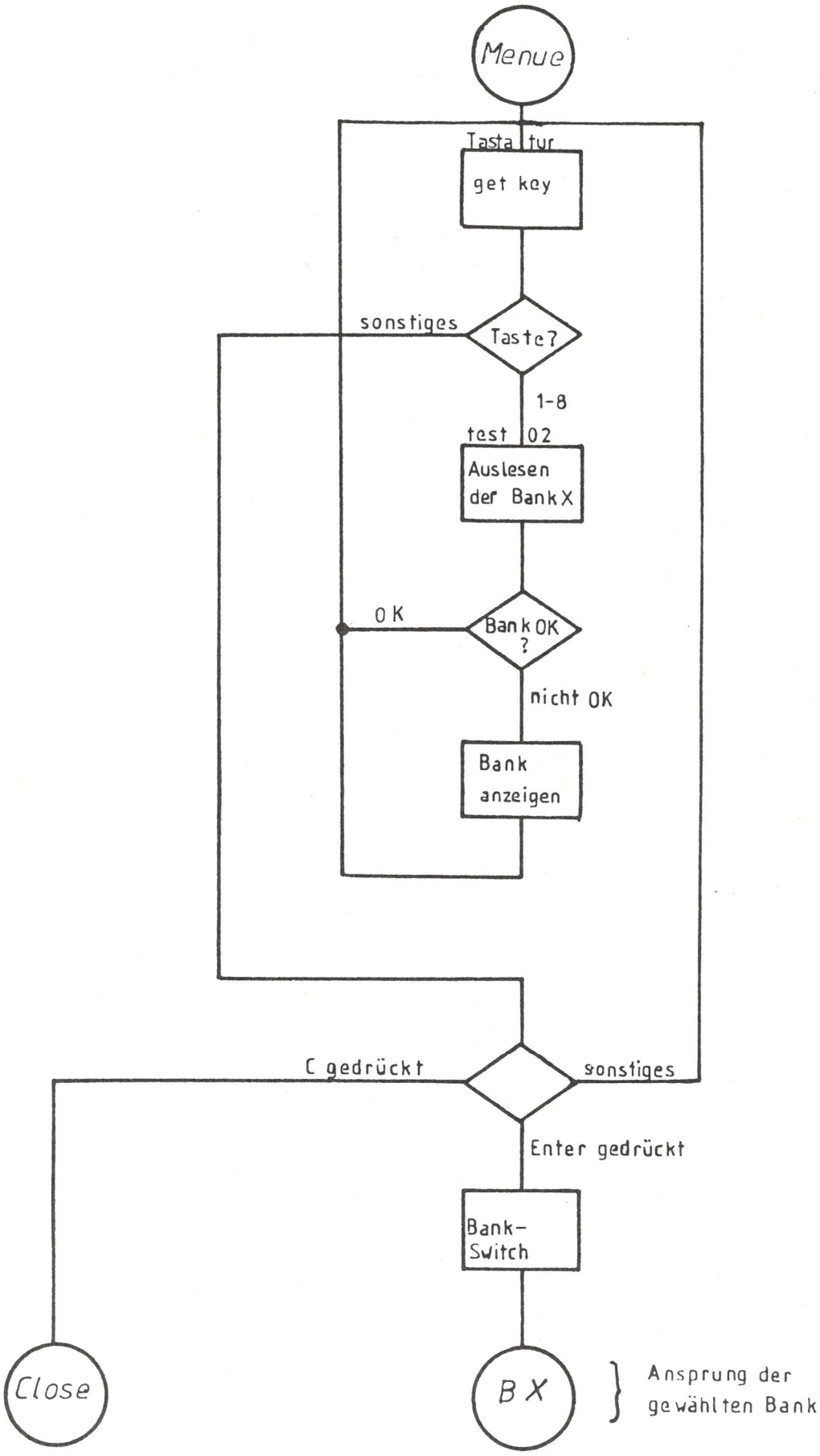
```
 ;
 ;*******************************
 ;***       4 ms warten       ***
 ;*******************************
wait4ms         .blck
                jsr waitsbr
waitsbr         pha
                lda #$00
loop            clc
                adc #$01
                bne loop
                pla
                rts
                .bend
 ;
 ;*******************************
 ;***     SWITCH-Routine      ***
 ;*******************************
test02          .blck
                ldx #look2end-look2-1
loop            lda look2,x
                sta $87e0,x
                dex
                bpl loop
                .bend
                lda zwis
                ora #$10
                jsr $87e0
                cpx #$4c
                bne looks
                cpy #$df
                bne looks
                clc
                rts
looks           sec
                rts
 ;
look2           sta $8c01
                ldx $df18
                ldy $df1a
                lda #$1f
                sta $8c01
                rts
look2end        .end
```

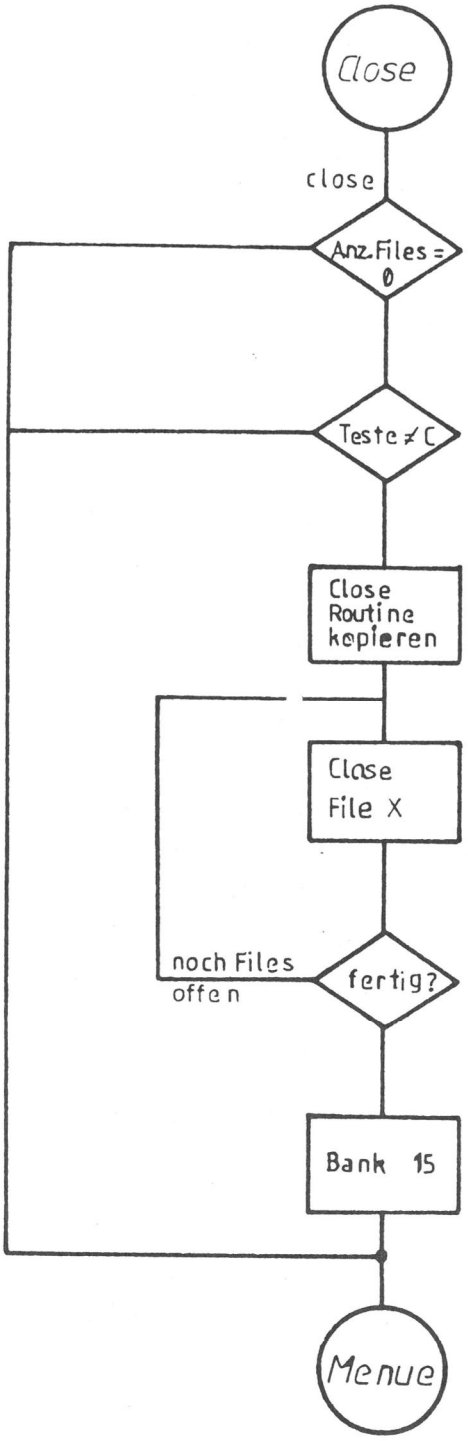
```
;*****************************
;***  Bank-Switching 6512  ***
;*****************************
;
;*****************************
;***  Hans - Peter  Brill  ***
;***  Hans-Boeckler-Str.3  ***
;***  5190 Stolberg/Rhld.  ***
;***  Tel.: (02402)  6148  ***
;*****************************
;
ind             = $28                           ;vektor in der zero-page
*               = $df18
;
;*****************************
;***   Makrodefinitionen   ***
;*****************************
move            .macro xxx,yyy
                lda &xxx&
                sta &yyy&
                lda &xxx&+1
                sta &yyy&+1
                .mend
;
```

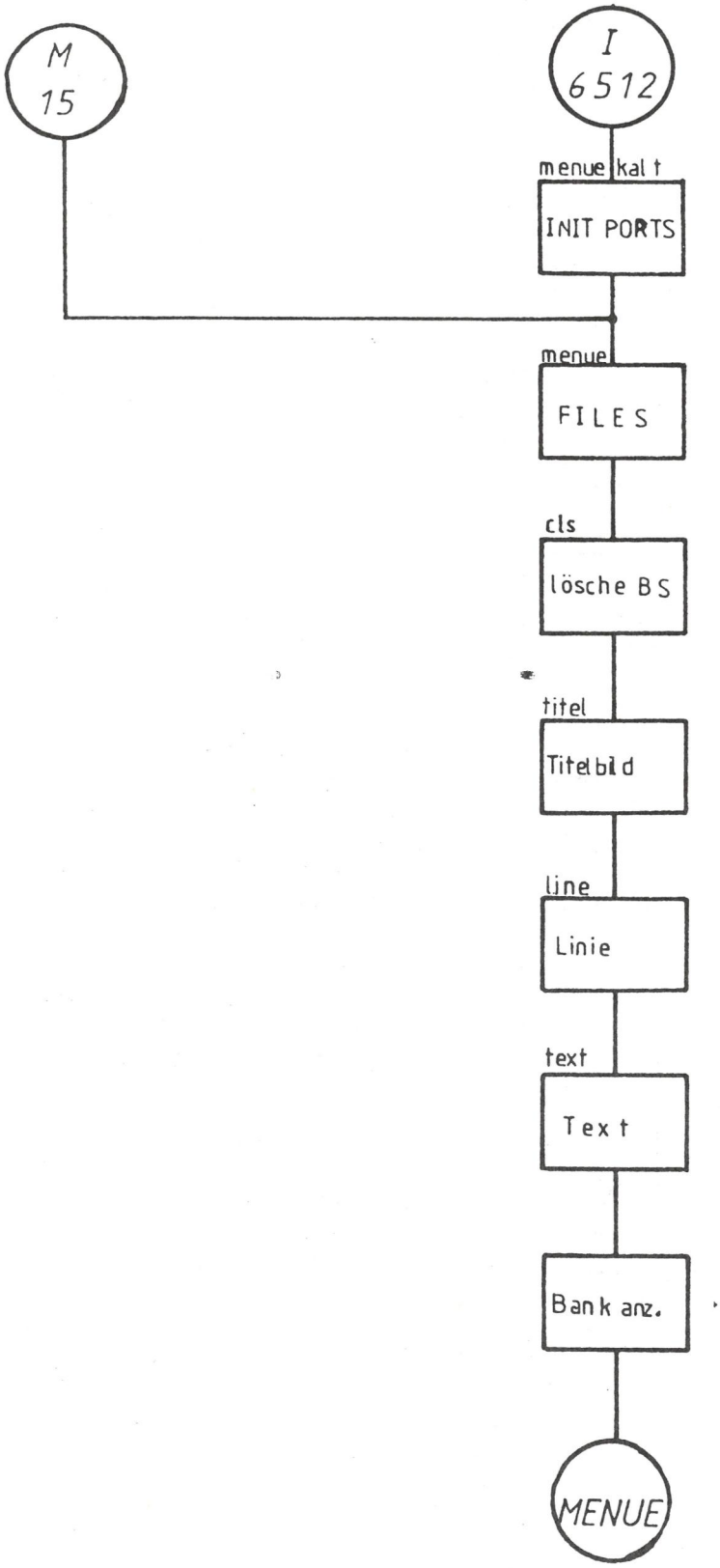
```
 ;
 ;*****************************
 ;***  IRQ-Tastaturabfrage  ***
 ;*****************************
               jmp irq                          ;abfrage commodore-
                                                ;taste
 ;
 ;*****************************
 ;***  Ansprung aus Bank 15 ***
 ;*****************************
bankxx         ldx regsp
               txs
               lda #%11010000                   ;bildschirm zurueckholen
               jsr copyscreen
               ldx regx
               ldy regy
               lda regst
               pha
               lda rega
               plp
               jmp (regpc)
irqend         lda rega
               pha
               txa
               pha
               jmp $e445
 ;
 ;*****************************
 ;***  Interruptergaenzung  ***
 ;*****************************
irq            sta rega
               lda #12
               jsr getkey
               bne irqend
               lda #7
               jsr getkey
               bne irqend
               stx regx
               sty regy
               pla
               sta regst
               pla
               sta regpc
               pla
               sta regpc+1
               tsx
               stx regsp
 ;
```

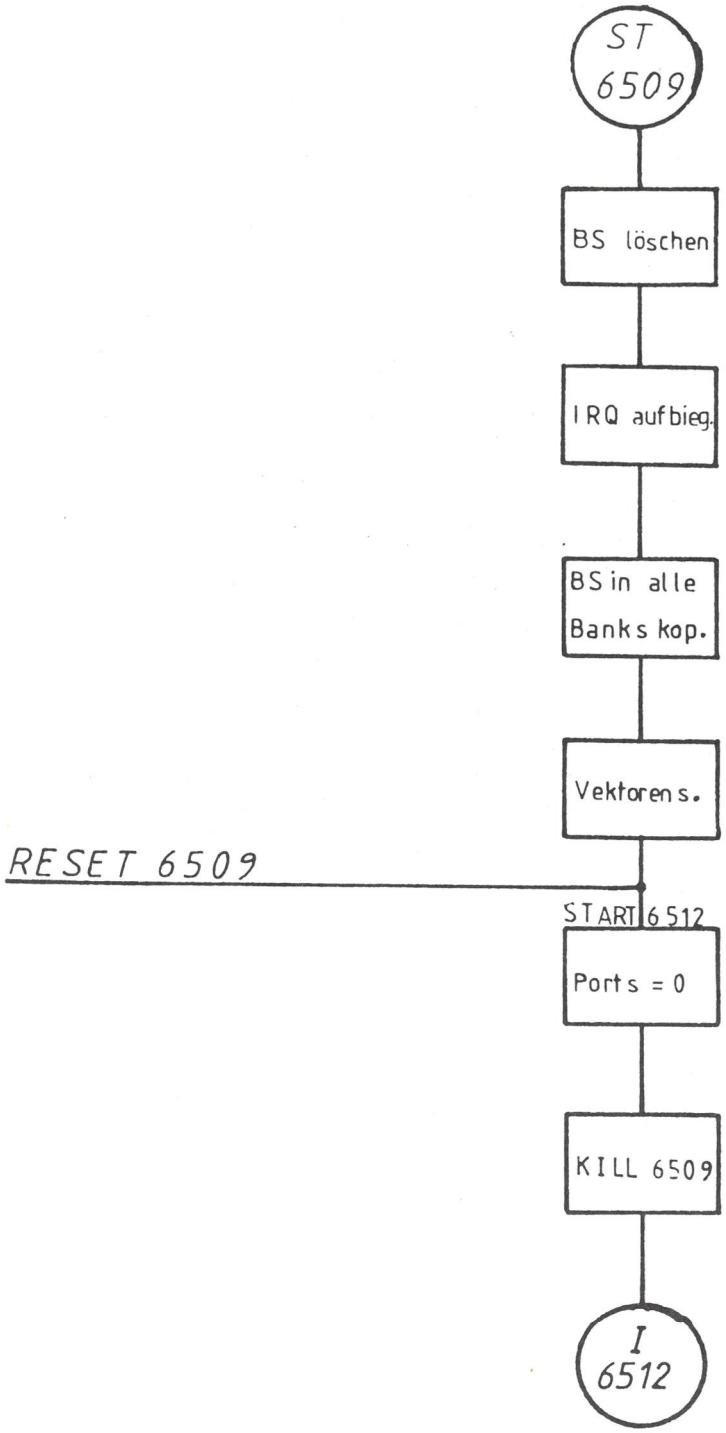
```
 ;
 ;*****************************
 ;*** Bildschirm-Subroutine ***
 ;*****************************
copyscreen      .blck
                sta wert1
                eor #%00100000
                sta wert2
                #move ind,zero                  ;bildschirm zurueckholen
                lda $8c01
                ora #$20                          ;8160-mode on
                sta $8c01
                ldy #$00
                sty ind
                lda #$80
                sta ind+1
loop            ldx wert1                         ;screen on/off
                stx $fff0
                lda (ind),y
                ldx wert2                         ;screen on/off
                stx $fff0
                sta (ind),y
                iny
                bne loop
                inc ind+1
                lda ind+1
                cmp #$88
                bne loop
                sty $fff0
                lda $8c01
                and #%11011111                    ;8160-mode off
                sta $8c01
                #move zero,ind
                .bend
                rts
 ;                                                ;in aktuelle bank
 ;*****************************
 ;***     Registerablage    ***
 ;*****************************
rega            .byte $00                         ;registerablage
regx            .byte $00
regy            .byte $00
regst           .byte $ff
regsp           .byte $ff
regpc           .word $fd16
zero            * = *+2
wert1           * = *+1
wert2           * = *+1
                .end
```

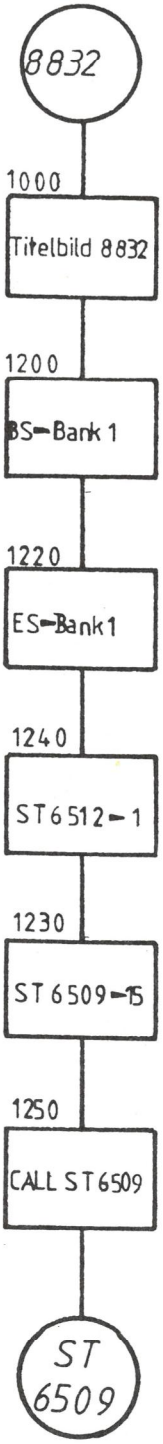
```
 ;
 ;*****************************
 ;***  Ansprung fuer Menue  ***
 ;*****************************
 ;
banksys         sei
 ;
 ;*****************************
 ;***   Bildschirm retten   ***
 ;*****************************
                lda #%11110000
                jsr copyscreen                  ;bildschirm retten
 ;
 ;*****************************
 ;***   Rueckkehr Systemb.  ***
 ;*****************************
                .blck
                ldx #7
loop            lda switch,x
                sta $87e0,x
                dex
                bpl loop
                .bend
                ldx $ae
                jmp $87e0
 ;
 ;*****************************
 ;***    SWITCH-Routine     ***
 ;*****************************
switch          lda #$1f                        ;bank 15
                sta $8c01
                jmp $0400
 ;
 ;*****************************
 ;***    Portansteuerung    ***
 ;*****************************
getkey          .blck
                sta $e810
loop            lda $e812
                cmp $e812
                bne loop
                cmp #$fe
                rts
                .bend
 ;
```


Menue
Tastatur
get key
sonstiges
Taste?
1-8
test 02
Auslesen der Bank X
OK
Bank OK ?
nicht OK
Bank anzeigen
C gedrückt
sonstiges
Enter gedrückt
Bank-Switch
Close
B X
Ansprung der gewählten Bank


Close
close
Anz.Files = 0
Teste ≠ C
Close Routine kopieren
Close File X
noch Files offen
fertig?
Bank 15
Menue

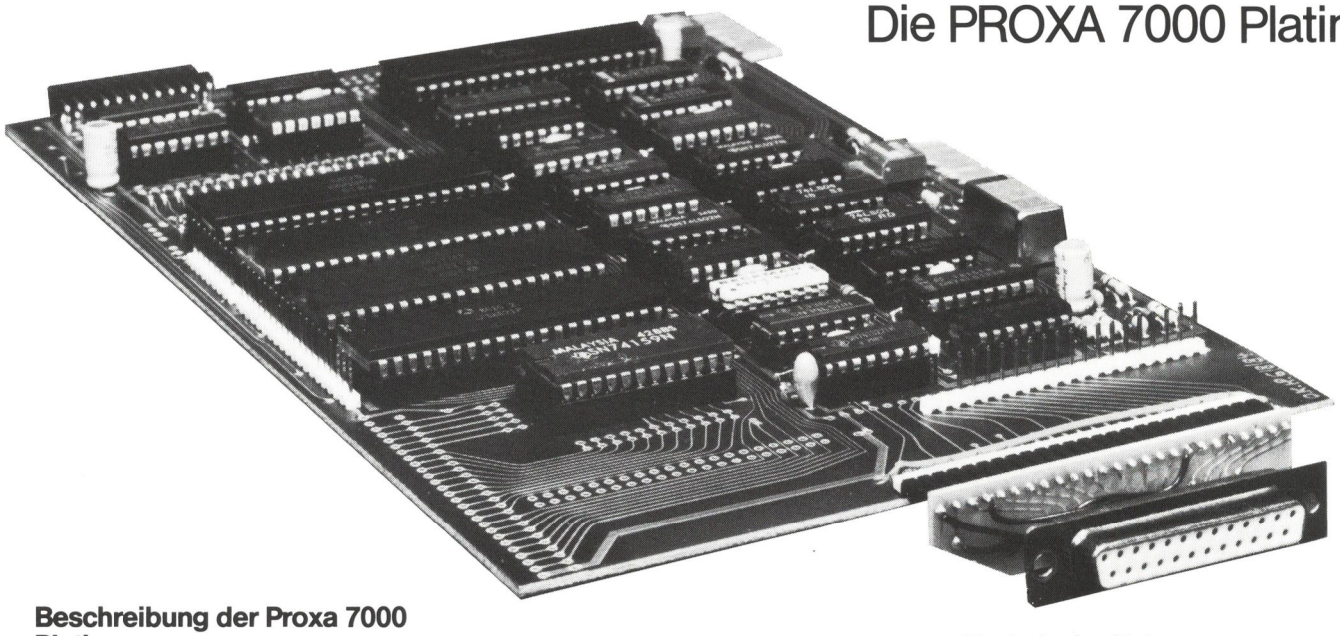

M
15
I
6512
menue kal t
INIT PORTS
menue
FILES
cls
lösche BS
titel
Titelbild
line
Linie
text
Text
Bank anz.
MENUE


ST
6509
BS löschen
IRQ aufbieg.
BS in alle
Banks kop.
Vektoren s.
RESET 6509
START 6512
Ports = 0
KILL 6509
I
6512


8832
1000
Titelbild 8832
1200
BS-Bank 1
1220
ES-Bank1
1240
ST6512-1
1230
ST6509-15
1250
CALL ST6509
ST
6509

69 Seiten

# Die PROXA 7000 Platine



**Beschreibung der Proxa 7000 Platine**

Die neue Platine der Firma Ultra electronic macht aus ihrem CBM 700 einen universellen Superrechner, der fast allen Anforderungen gewachsen ist.
Er ist kompatibel zu allen großen Commodore-Rechnern, wie CBM 2001 (PET), CBM 30er, 40er, 80er Serien, dem neuen 8296 und natürlich der 600er/700er Serie.
Die Platine wird einzeln oder in Verbindung mit einem CBM 720 ausgeliefert.
Vorteile:
Die Software der o. g. Systeme kann übernommen werden. Die Verarbeitungsgeschwindigkeit der Software im Rechner verdoppelt sich. Durch einen zusätzlichen Tastaturstecker kann sowohl die 8032er Tastatur als auch die 700er Tastatur übernommen werden. Dadurch ist ebenfalls volle hardwarekompatibilität gegeben.
Eine V-24 Schnittstelle zum Anschluß an Modems und Akustikkoppler ist bereits eingebaut und wird incl. Treibersoftware mitgeliefert. Der nachleuchtende Bildschirm ist vollkommen flimmerfrei und bietet durch seine 9 x 14 Matrix ein gestochen scharfes Schriftbild. Mit Hilfe einer Zusatzplatine ist ein eigener Zeichensatz programmierbar (z. B. mathematisch-griechischer Zeichensatz o. ä., aber auch fremdsprachliche Schriftzeichen).

2 Anschlüsse für Recorder oder Dongel sind vorhanden.
Zusätzlicher Musiksynthesizer (C-64 kompatibel).
Durch den großen Speicherplatz (256 K) ist es möglich eine sog. RAM-Disk zu realisieren, d. h. Daten und Programme werden im RAM zwischengespeichert. Dadurch erhöht sich die Zugriffszeit auf die Daten beträchtlich (1000fach).
Auf den ROM-Steckplätzen befindet sich RAM, so daß Umschaltplatine, RAM-Türmchen oder ROM-Boxen entfallen.
Dieser RAM ist softwaremäßig mit einem Schreibschutz zu versehen, so daß die gespeicherten Daten nicht mehr überschrieben werden können.

**Technische Daten:**

**Speicherplatz:**
128–896 K

Kompatibilität zu allen großen CBM-Rechnern

**Tastatur:**
alle CBM-Tastaturen anschließbar, für volle Hardwarekompatibilität.
Ebenfalls erhältlich intelligente Tastatur und Grafiktastatur.
2-Key roll over implementiert

**Bildschirm:**
stark nachleuchtend und flimmerfrei.
9 x 14 Punktmatrix mit Zusatzmodul frei programmierbar.

**Interfaces:**
RS 232 C, IEC, Cassettenrecorder-Interface, User-Port frei belegbar, 2 Tastaturschnittstellen.

**Taktfrequenz:**
2 MHz (für doppelte Geschwindigkeit)

**Besonderheiten:**
Refresh wird auch nach Ausstieg der CPU gewährleistet. Dadurch werden Programme und Daten erhalten. 20 Funktionstasten bei CBM-Tastatur frei belegbar.
Musiksynthesizer integriert.

proxa computer

1298,- (incl. Buch 7/86)

**Aachener Str. 29 · 5000 Köln 1 · Tel.: 0221/491091 · Telex: 8886627**