



**BASIS**  
**108 E**

**Betriebsanleitung**

HINWEIS:

Dieses Handbuch ist urheberrechtlich geschützt. Alle Rechte sind vorbehalten.  
Dieses Schriftstück darf weder im Ganzen noch als Teil kopiert, fotokopiert,  
reproduziert, übersetzt oder auf ein elektronisches Medium überführt oder in eine  
maschinenlesbare Form gebracht werden, ohne daß eine vorherige schriftliche  
Zustimmung der BASIS MICROCOMPUTER GmbH vorliegt.

(C) 1982 BASIS MICROCOMPUTER GmbH  
Postfach 1603  
D-4400 Münster

Änderungen bedingt durch technischen Fortschritt bleiben vorbehalten.

Eingetragene Warenzeichen:  
Apple/Apple II: Apple Computer Corp.  
CP/M : Digital Research Inc.  
UCSD-Pascal : University San Diego California

## INHALTSVERZEICHNIS

### Kapitel 1

#### Allgemeine Beschreibung und Inbetriebnahme

- 6 Das BASIS 108 Computersystem und Datensichtgeräte
- 6 Anschluß der Kabel
- 8 Belegung der Pins bei den Steckleisten der Rückseite
- 8 Öffnen des BASIS 108
- 11 Die Hauptplatine
- 13 Der Handregleranschluß
- 14 Die Stromversorgung
- 15 Pinbelegung der Slots
- 18 Die Diskettenlaufwerke
- 19 Pflege der Diskettenlaufwerke und der Disketten
- 20 Einlegen und Herausnehmen von Disketten

### Kapitel 2

#### Software

- 23 Einleitung
- 24 UCSD p-System IV.0
- 27 Das CP/M-System
- 28 Das DOS3.3-System

### Kapitel 3

#### Zugriff zur Hardware

- 31 Logischer Schaltplan
- 32 Text- und Graphikdarstellung
- 32 Der Textbildschirm
- 32 80/40 Zeichendarstellung
- 33 Das Prinzip der 80-Zeichendarstellung
- 33 Softwareschalter für die Textdarstellung
- 33 Softwareschalter für die Graphik
- 33 LO-RES-Graphik
- 34 MI-RES-Graphik
- 34 HI-RES-Graphik
- 34 Farbdarstellung der HI-RES-Graphik
- 35 Zeichengenerator
- 36 Tastatur

## Kapitel 4

### Der Monitor

- 39 Einleitung
- 39 Einweisung
- 40 Daten und Adressen
- 40 Inhaltsüberprüfung einer Speicherstelle
- 41 Überprüfen mehrerer Speicherstellen
- 42 Änderung einer Speicherstelle
- 42 Änderung von aufeinanderfolgenden Speicherstellen
- 43 Übertragen eines Speicherbereiches
- 44 Vergleich von zwei Speicherbereichen
- 44 Programmieren und Starten von Maschinenprogrammen
- 46 Prüfen und Ändern von Registerinhalten des 6502
- 46 Weitere Monitor-Kommandos
- 47 Kleine Hilfen für den Umgang mit dem Monitor
- 48 Erzeugen eigener Kommandos
- 49 Übersicht über die Monitorkommandos
- 52 Liste ausgewählter Monitor-Unterprogramme
- 56 Spezialadressen des Monitors

## Kapitel 5

### Der Speicher

- 58 Speicherorganisation
- 58 Aufteilung des Adreßraumes
- 59 BANK 0/BANK 1 Umschalten
- 60 ROM und RAM Umschaltung
- 61 Das Statik-RAM für die 80 Z-Darstellung

## Kapitel 6

### Ein-/Ausgabe

- 63 Eingebaute Ein-/Ausgabemöglichkeiten
- 63 Dateneingänge, Status Eingänge, Strobe
- 64 Kippschalter, Drucker Interface
- 64 Serielles RS 232c Interface
- 65 Kontrollregister
- 66 Kommando Register
- 67 Status Register
- 68 Kassettenrekorder Interface
- 68 Handregleranschluß und TTL Ein- und Ausgänge
- 68 Lautsprecher
- 68 Erweiterungs-ROM

## A N H A N G

- A 73 Hinweise zur Softwarekompatibilität mit Apple II
- B 81 Volume UT 108
- C 85 BASIS 108 System Monitor
- D 87 Hinweise zu Applesoft Basic FP40 und FP80
- E 88 V24 Parameter
- F 90 Anschluß eines Fernsehgerätes ohne Videoeingang
- G 91 Arbeiten mit dem Kassettenrekorder
- H 93 Hexadezimalzahlen
- I 94 Tabelle der Tastenbelegung
- J 97 Zusammenstellung der Ein-/Ausgaberegister
- K 99 Der Z-80-Teil
- L 102 Datenblatt und Befehlsregister des Z-80
- M Datenblatt und Befehlsregister des 6502
- N Auflistung der Monitor ROM Programmbeehle
- O Stichwortverzeichnis
- P Schaltung der Tastaturplatine
- Q Schaltung der Hauptplatine

## **Vorwort**

In diesem Handbuch finden Sie neben einer Reihe sehr einfacher Hinweise für den Umgang mit Ihrem Computer eine Vielzahl von Hinweisen, die vor allem für den fortgeschrittenen Programmierer von Interesse sind.

Für den Anfänger ist dieses Buch in weiten Passagen wohl kaum verständlich. Deswegen sollte er sich auch zunächst mit Einführungen in die Programmierung und Arbeitsweise eines Computers beschäftigen, ehe er intensiver mit diesem Handbuch arbeitet. Er sollte aber die Kapitel 1 und 2, sowie Teile des Anhangs, die ihn evtl. betreffen, auch wenn Ihm andere Programmierhandbücher zur Verfügung stehen, zunächst lesen.

Zum Teil werden hier auch Möglichkeiten aufgezeigt, die aus der Kompatibilität des BASIS 108 mit dem Apple II resultieren. Möglichkeiten also, die z.B. Anwender des UCSD p-Systems IV.0 kaum interessieren.

Ein Handbuch wird geschrieben für den Anwender, deshalb hier zum Schluß die Bitte an Sie: Wenn Sie Kritik und Anregungen haben, so teilen Sie uns diese mit, damit wir sie bei der nächsten Auflage berücksichtigen können.

Wir wünschen Ihnen erfolgreiche Arbeit mit Ihrem BASIS 108.

## **KAPITEL 1**

### **INHALTSVERZEICHNIS**

#### Allgemeine Beschreibung und Inbetriebnahme

- 6 Das BASIS 108 Computersystem und Datensichtgeräte
- 6 Anschluß der Kabel
- 8 Belegung der Pins bei den Steckleisten der Rückseite
- 8 Öffnen des BASIS 108
- 11 Die Hauptplatine
- 13 Der Handregleranschluß
- 14 Die Stromversorgung
- 15 Pinbelegung der Slots
- 18 Die Diskettenlaufwerke
- 19 Pflege der Diskettenlaufwerke und der Disketten
- 20 Einlegen und Herausnehmen von Disketten

## **Das BASIS 108 Computersystem und Datensichtgeräte**

Ihr BASIS 108 Computersystem besteht aus folgenden Teilen:

1. Der Zentraleinheit mit oder ohne eingebauten Diskettenlaufwerken,
2. der Tastatur,
3. dem Netzanschlußkabel,
4. der Diskette ZAP:, auf der Rückseite befindet sich Volume UT108:,
5. und diesem Handbuch.

Bewahren Sie das Verpackungsmaterial bitte auf, falls Sie das System einmal transportieren wollen, bietet es guten Schutz vor Beschädigung des Computers. Zum Betrieb des Systems benötigen Sie noch einen Bildschirm (Datensichtgerät) oder, falls Ihnen 40 Zeichen/ Zeile genügen, ein Fernsehgerät mit Video-Eingang. (Mehr als 40 Zeichen/ Zeile kann ein normales Fernsehgerät nicht sauber darstellen). Für den Anschluß eines Fernsehgerätes ohne Videoeingang s. Anhang F. Sollten Sie großen Wert auf gute Farbausgabe legen, dann benötigen Sie einen hochauflösenden RGB-Monitor. Ihr BASIS Vertriebspartner wird Sie auch in dieser Angelegenheit beraten.

### **Anschluß der Kabel**

Wenn Sie ein BASIS 108 System ohne Diskettenlaufwerke erworben haben und die ersten Schritte mit Ihrem eigenen Computer per Kassettenrekorder zurücklegen wollen, dann schließen Sie Ihren Kassettenrekorder an die dafür vorgesehene DIN-Buchse auf der Rückseite des BASIS 108 an, weiteres s. Anhang G.

**Wichtig:** Fragen Sie Ihren BASIS Vertriebspartner nach dem richtigen Monitor-ROM zum Laden des Betriebssystems mit Kassettenrekorder.

Haben Sie Ihr BASIS 108 Computersystem mit Diskettenlaufwerken erworben, um damit eine Arbeitserleichterung bei Ihren täglichen Routinearbeiten zu erzielen, so haben Sie keinerlei Anschlußarbeiten.

Ein eigenes Laufwerk sollten Sie entsprechend der Anleitung Seite 11 einbauen. Die Steckdosen auf der Rückseite sind für den Bildschirm und Drucker vorgesehen. Verbinden Sie also alle Einheiten miteinander, indem Sie das Netzkabel des Bildschirms und des Druckers in die dafür vorgesehenen Steckdosen auf der Rückseite des BASIS 108 einstecken.

Diese beiden Steckdosen werden über den zentralen Netzschalter des Systems geschaltet.

**Wichtig:** Bitte die Steckdosen nur für Drucker und Bildschirm benutzen, nicht für Staubsauger etc.



### Rückseite

Verbinden Sie den Bildschirm oder das Fernsehgerät durch ein Video-Kabel mit dem RGB, S/W-Video oder PAL-Video Ausgang des Systems.

Stecken Sie den Stecker der Tastatur in den dafür vorgesehenen Buchsenstecker auf der Rückseite des Gerätes.

In der Betriebsanweisung Ihres Druckers finden Sie Angaben darüber, ob er über eine serielle oder parallele Schnittstelle verfügt. Entsprechend können Sie die Verbindung zum BASIS 108 herstellen, indem Sie das Datenkabel zur Rückseite führen und es in die infrage kommende Steckleiste stecken.

Verbinden Sie nun das System über das Netzkabel mit der nächsten Steckdose und vergewissern Sie Sich noch einmal, ob alle Geräte richtig verbunden sind. Jetzt schalten Sie den Netzschalter an der unteren linken Seite der Front des BASIS 108 ein.

Die rote Lampe leuchtet auf, der eingebaute Lautsprecher piept kurz und das linke Diskettenlaufwerk läuft an.

Auf dem Bildschirm erscheint die Meldung:

### B A S I S   1 0 8

Da Sie mehrere Betriebssysteme und Zusatzgeräte verwenden können, ist es notwendig, die grundsätzliche Arbeit mit Ihrem Computer in einem gesonderten Kapitel zu besprechen.

Wenn Sie nicht mehr über Ihren BASIS 108 wissen möchten, dann lesen Sie bitte Kapitel 2.

Falls Sie aber Ihren persönlichen Computer näher kennenlernen möchten, dann lesen Sie weiter.

## **Belegung der Pins bei den Steckleisten der Rückseite**

Auf der nächsten Seite finden Sie die Zeichnung mit der Rückseite. Hier sind die entsprechenden Pins der Steckleisten bezeichnet. Die Bedeutung der Zeichen ergibt sich zum Teil aus der Beschriftung.

Die Bezeichnungen D0 - D7 sind von der Tastatur her Dateneingänge, bei der parallelen Schnittstelle die Ausgänge der Druckzeichen.

Die Bezeichnung GND bedeutet Gerätemasse.

SM ist dagegen die Signalmasse.

Ausgang sind die Signale: RTS, DTR, R, G, B.

Eingang sind die Signale: CTS, DSR, DCD, PC, DI, AC.

Die Abkürzungen der Signale bei der seriellen Schnittstelle entnehmen Sie bitte im Anhang dem Datenblatt des 6551.

ST Strobe ist ein negatives Signal mit 1 Mikrosekunde Dauer .

AC Ist ein negatives Antwortsignal mit 1 Mikrosekunde Dauer (Acknowledge).

PC (Printer Connect) ist auf 0 gezogen, wenn der Drucker eingeschaltet ist.

Die beiden 12 V Anschlüsse der seriellen Schnittstelle sind durch Widerstände von 1 kOhm geschützt.

Ist der Eingang CTS inaktiv, dann erfolgt keine Sendung.

## **Öffnen des BASIS 108**

**Wichtig:** Bevor Sie das System öffnen, ziehen Sie bitte den Netzstecker aus der Steckdose

Das BASIS 108 System besteht aus einem Aluminium-Gußgehäuse mit dem eingebauten Netzteil und der Hauptplatine. In der Front des Gehäuses sind Öffnungen zum Einbau von zwei Diskettenlaufwerken, die durch Blindabdeckungen verschlossen sind, wenn keine Laufwerke eingebaut wurden. Montagebleche und Befestigungsschrauben für Diskettenlaufwerke sind aber in jedem Fall vorhanden, siehe S. 12.

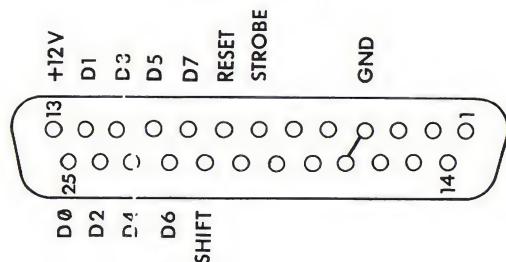
In der Mitte finden Sie neben den schon belegten Buchsensteckern für die Tastatur und die serielle sowie parallele Schnittstelle noch drei weitere Montageplätze für DP-25 Buchsenstecker.

Darunter befindet sich neben den Anschlußbuchsen für einen RGB-Monitor, PAL-Video Fernseher und S/W Bildschirm (BNC-Buchse) ein Durchbruch zum direkten Herausführen von Flachbandkabeln bis zu einer Breite von 50 Adern.

Das Gehäuse besteht aus zwei Teilen: dem Unterteil mit der hochgezogenen Rückwand und dem Deckel. Der Deckel wird an der Rückwand des Unterteils von zwei Metallstiften gehalten und durch zwei Schrauben, die sich im hinteren Bereich des Unterteils befinden, gesichert.

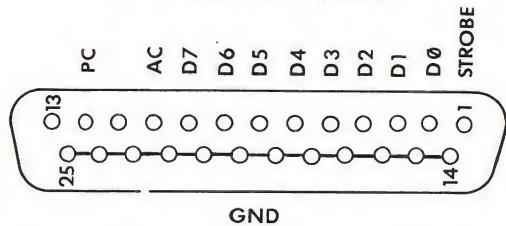
Heben Sie das System an und lösen Sie die Schrauben mit einem stabilen Schraubenzieher. Ziehen Sie nun das Oberteil nach vorne ab.

## Tastatur



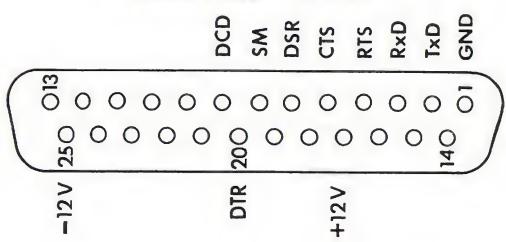
1

## Parallel · E/A

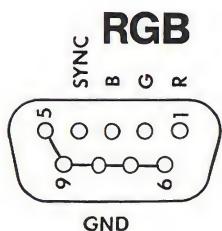


2

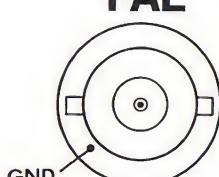
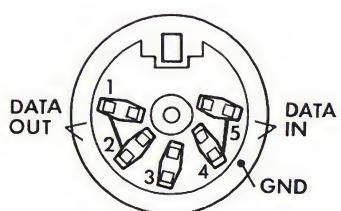
## Seriell · E/A



3

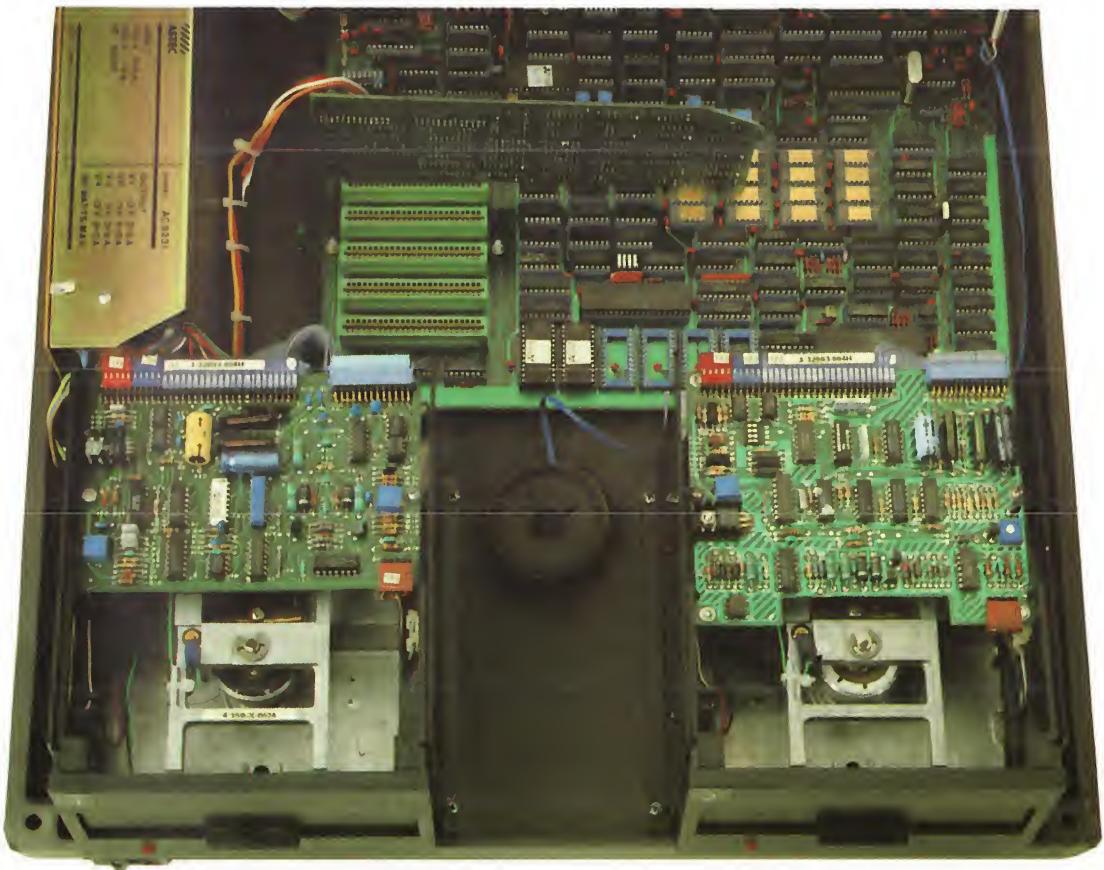


V  
D  
I  
O



Kassette

4



Innenansicht

## Die Hauptplatine

Die große Leiterplatte ist der eigentliche Computer. Auf ihr sind ca. 130 hochintegrierte Schaltkreise, ICs, untergebracht, die die elektrische Verbindung zwischen den zwei Mikroprozessoren (6502 und Z-80), den Speicherbausteinen RAM (Random Access Memory) und ROM (Read only Memory) und den Ein-/Ausgabebausteinen herstellen.

Auf der linken Seite der Platine befinden sich sechs 50-polige Slots (Buchsenleisten), von 2 bis 7 nummeriert, in die Systemerweiterungen wie z.B. Steuereinheiten für Diskettenlaufwerke, serielle und parallele Schnittstellenkarten für weitere Drucker oder Hauptspeichererweiterungen eingesetzt werden können. Wenn Ihr System mit Diskettenlaufwerken ausgestattet ist, dann steckt in dem Steckplatz 6 die Steuereinheit, der Controller. Dieser Controller kann bis zu zwei Diskettenlaufwerke kontrollieren.

Hinter rechts befinden sich drei Stiftleisten mit je 20 Stiften, von denen aus Flachbandkabel zu den Buchsensteckern auf der Rückwand des Systems führen, für die Tastatur, sowie für einen parallel und einen seriell anzusteuernden Drucker.

Hinter in der Mitte der linken Seite ist eine Stiftleiste mit 10 Stiften. Hierüber wird das RGB-Signal über ein Flachbandkabel auf den entsprechenden Stecker auf der Rückseite gegeben. Rechts daneben befindet sich der schwarz/weiß Video-Ausgang (S/W-Video). Der Ausgang für PAL-Video bzw. den Anschluß eines UHF-Modulators ist die Steckleiste mit den vier Stiften in der linken oberen Ecke der Platine.

Die Farbqualität bei Farbausgabe läßt sich über den Trimmkondensator, links oben, mit Hilfe eines kleinen Schraubenziehers einstellen. Die Intensität des S/W-Videosignales läßt sich über das rechts in der Nähe des Trimmkondensators stehende Potentiometer regeln.

Der auf der rechten Seite der Platine angebrachte Stecker führt ein Verbindungskabel zum Lautsprecher und zum Kassettenrekorder-Anschluß.

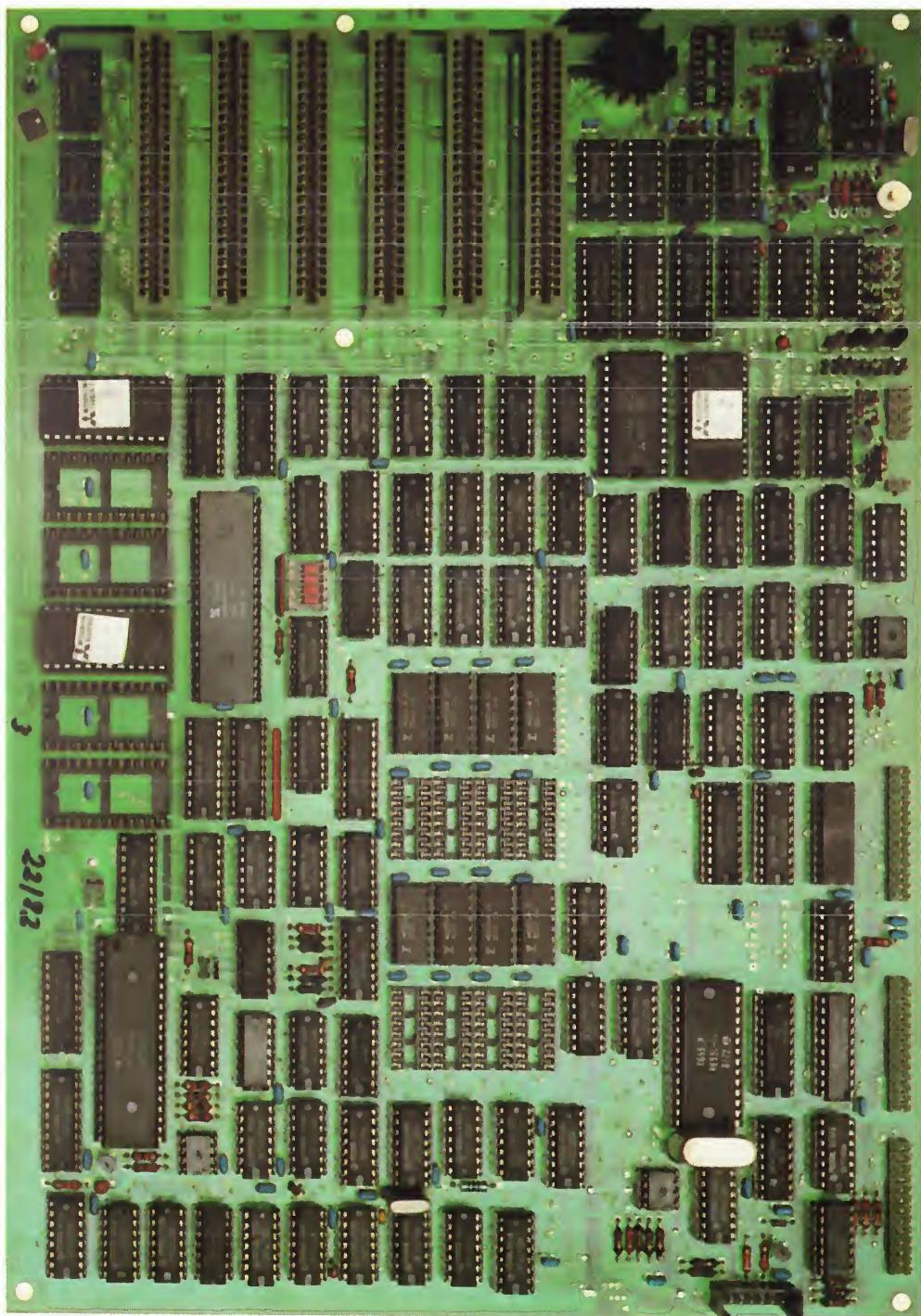
Der große Stecker direkt hinter der Buchsenleiste 7 verbindet über ein Anschlußkabel das Netzteil mit der Hauptplatine.

Etwa in der Mitte der Platine sind die Hauptspeicherbausteine (RAMs) angeordnet. In der Grundausstattung des BASIS 108 befinden sich 8 IC's mit je 64 KBit in den eingelöteten Sockeln. Weitere 8 Bausteine können durch einfaches Einsetzen in die dafür vorgesehenen Steckplätze nachgerüstet werden und erweitern dann den Hauptspeicher auf eine Kapazität von insgesamt 128 KByte.

Da die verwendeten 8 Bit Mikroprozessoren 6502 und Z-80 nur einen Speicheradressraum von 65 536 Bytes (64 KBytes) ansprechen können, benötigen Sie zum Adressieren des Gesamtspeicherraumes von 2x 65 536 Bytes ein spezielles Programm, das Sie von Ihrem BASIS Vertriebspartner beziehen können.

In der ersten Reihe auf der Platine sind sechs Sockel angeordnet, von denen zwei Sockel durch integrierte Bausteine belegt sind. Diese Sockel sind für Festwertspeicher (ROMs) reserviert. Sie können Programme oder Programmiersprachen aufnehmen, die im Augenblick des Einschaltens des BASIS 108 verfügbar werden. Eines dieser Programme ist schon in dem linken Baustein vorhanden; der BASIS 108 System-Monitor. Mit Hilfe dieses Monitors (Programmes) wird nach dem Einschalten des Systems das linke Diskettenlaufwerk (Laufwerk 1) angesteuert, hierzu weiteres in Kapitel 2 und 4.

Ist kein Laufwerk eingebaut, können Sie Programme vom Kassettenrekorder einlesen, wenn in Ihrem BASIS 108 ein 40 Spalten Monitor-ROM eingebaut ist. Weiteres hierzu siehe Anhang G.



Hauptplatine

Die beiden Schaltungsbrücken in der Nähe des 6502 sind zur Umschaltung zwischen ROM- und EPROM-Bestückung. Im Lieferzustand befinden sich die beiden Jumper (Kurzschlußbrücken) in der Position EPROM. In diesem Zustand sind das eingesetzte BASIS-Monitor-EPROM und das "Dummy"-EPROM aktiv geschaltet. Soll ein kompletter Satz EPROMs vom Typ 2716 installiert werden, wird die Jumper-Stellung nicht verändert.

Bei Einsatz der ROM-Bestückung (original Applesoft- oder Integer-ROMs) müssen beide Jumper in die entgegengesetzte Position.

Die Beschreibung der Stellung des Dip-Schalters über dem Z-80 finden Sie im Anhang bei der Beschreibung des Z-80 Teiles.

### **Der Handregleranschluß**

Links hinter der Buchsenleiste 7 befindet sich ein nicht mit einem IC bestückter Sockel. Dieser Sockel dient der Aufnahme eines Steckers von Handreglern (Game Paddle oder Joystick). Die Kabel müssen nach links aussen zeigen. Entsprechende Spielprogramme fordern Sie auf, die Handregler anzuschließen.

Im folgenden sind die Handregleranschlußbelegung und die Beschreibung der Spielanschlußsignale wiedergegeben.

#### Handregleranschlußbelegung

+5V	1	16	NC
SW0	2	15	AN0
SW1	3	14	AN1
SW2	4	13	AN2
C040 STB	5	12	AN3
PDL0	6	11	PDL3
PDL2	7	10	PDL1
GND	8	9	NC

#### Beschreibung der Handregleranschlußsignale

Anschluß	Name	Beschreibung
1	+5V	+5 V Stromversorgung, max. 100 mA.
2 - 4	SW0 - SW2	Ein-Bit-Eingänge (Drucktasten). Es sind Standard-TTL-Eingänge der 74LS-Serie.
5	C040 STB	Der Impulsausgang ist ein Standard-TTL 74LS-Ausgang. Dieser Anschluß liegt normalerweise an +5 V und geht beim Zugriff auf eine Adresse von \$C040 bis \$C04F für die Dauer von 0.4 Mikrosekunden in Phase $\Phi_0$ auf logisch 0.

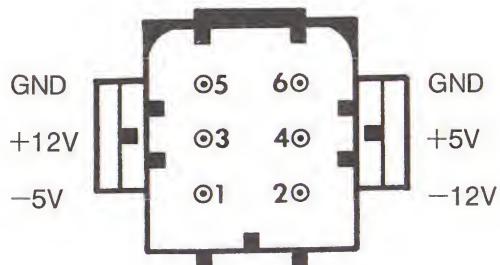
Anschluß	Name	Beschreibung (Forts.)
6, 7, 10, 11	PDL0-PDL3	Spielsteuereingänge. Diese Analogeingänge sollten mit 150 kOhm-Regelwiderständen an +5 V angeschlossen werden.
8	GND	Elektrische Masse des Systems: 0 V.
12-15	AN0-AN3	Signal-Ausgänge (Annunciator). Diese Standardausgänge der TTL 74LS-Serie sollten gepuffert werden, falls sie andere als TTL-Eingänge treiben sollen.
9, 16	NC	Kein Anschluß.

### Die Stromversorgung

Das Metallgehäuse auf der linken Seite neben der Hauptplatine ist das Netzteil. Es liefert vier Spannungen:

+5 Volt,  
 -5 Volt,  
 +12 Volt,  
 -12 Volt.

Die Pinbelegung entnehmen Sie der Abbildung:



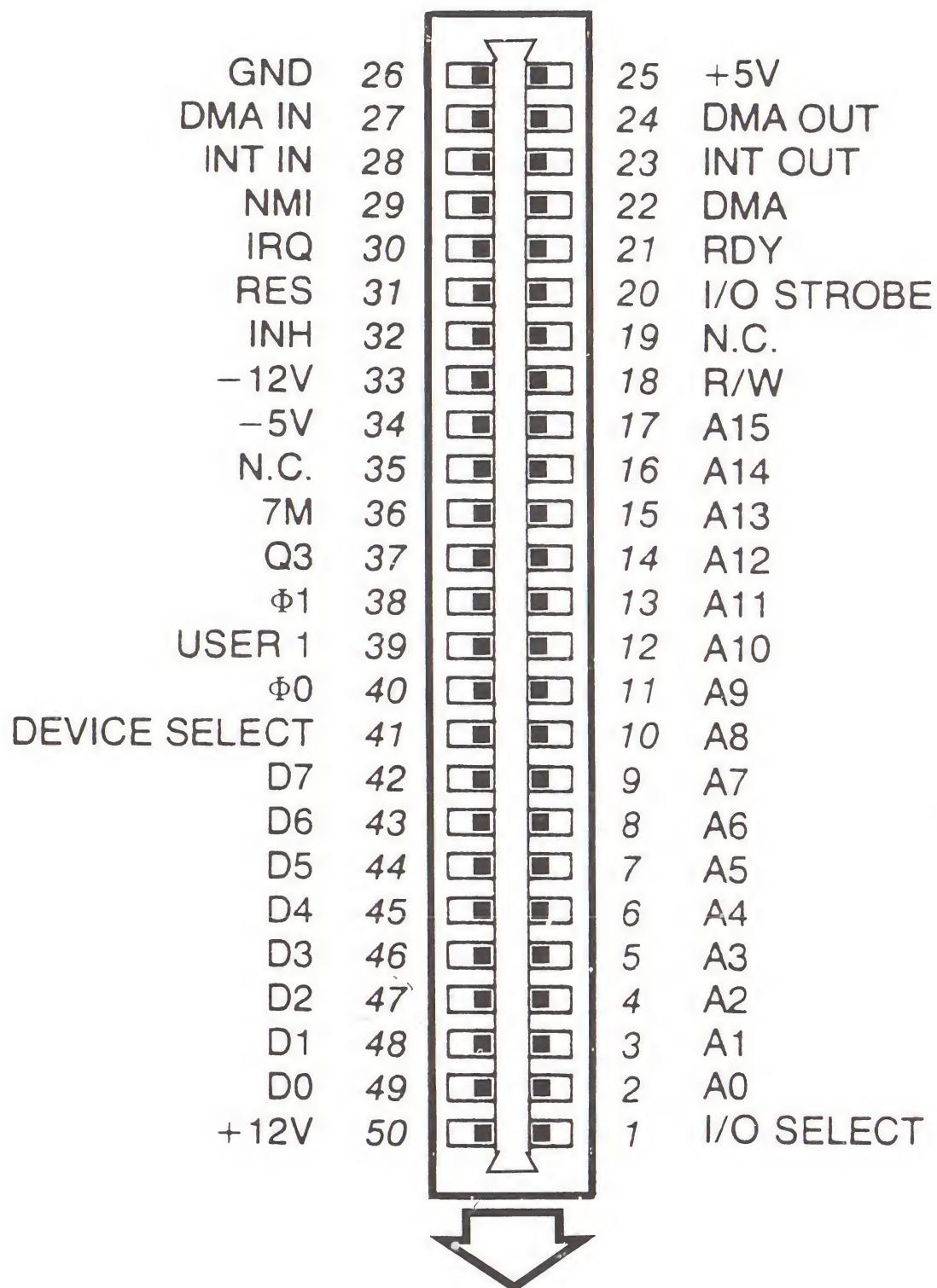
Das getaktete Netzteil wurde mit einer Schutzeinrichtung versehen, damit keine Überlastung auftreten kann. Die Eingangsseite kann an 110 Volt bis 250 Volt angeschlossen werden, bei 110 Volt muß im Netzteil ein Stecker umgesteckt werden, und ist über ein Kabel mit dem an der Rückseite des Systems angebrachten Netzfilter verbunden.

**Wichtig:** Das Netzteil nicht öffnen! Lebensgefährliche Spannungen!

## Pinbelegung der Slots

Im folgenden ist die Pinbelegung der Slots aufgeführt. Die Zeichnung finden Sie auf der nächsten Seite. Die aufgeführten Zahlen mit einem \$-Zeichen sind Hexadezimalzahlen. Bitte sehen Sie hierzu in den Anhang H und in die Kapitel Monitor ff.

An-schluß	Name	Beschreibung
1	I/O SELECT	Diese Leitung liegt normalerweise auf +5 V. Wenn der Mikroprozessor auf Seite \$Cn zugreift (wobei n die Slotnummer ist), sinkt die Spannung auf logisch 0 ab. Dieses Signal wird während $\Phi_1$ aktiv und treibt 10 LS-TTL-Lasten.
2-17	A0-A15	Der gepufferte Adressbus. Die Adressen werden in $\Phi_1$ gültig und bleiben es in $\Phi_0$ . Jede dieser Leitungen treibt 5 LS-TTL-Lasten.
18	R/W	Gepuffertes Lese-/Schreib-Signal (Read/Write). Dieses Signal ist zur selben Zeit gültig wie der Adressbus und geht auf +5 V in einem Lese- und auf logisch 0 in einem Schreibvorgang. Diese Leitung kann 2 LS-TTL-Lasten versorgen.
20	I/O STROBE	Diese Leitung treibt 4 LS-TTL-Lasten und geht während $\Phi_0$ auf 0, wenn der Adressbus eine Adresse zwischen \$C800 und \$CFFF enthält.
21	RDY	Der RDY-Eingang des 6502-Mikroprozessors. Wird diese Leitung während $\Phi_1$ auf 0 gezogen, so stoppt der Mikroprozessor und hält die aktuelle Adresse im Adressbus fest.
22	$\overline{DMA}$	Wird dieser Anschluß auf logisch 0 gelegt, so wird der Adressbus gesperrt und der Mikroprozessor gestopt. Diese Leitung wird durch einen 1 KOhm Widerstand auf +5 V gehalten.
23	INT OUT	Daisy-Chain Interrupt-Ausgang zu Geräten niedriger Priorität. Dieser Anschluß wird normalerweise mit Pin 28 (INT IN) verbunden. INT OUT 7 führt zum Z-80-Teil.
24	DMA OUT	Daisy-Chain DMA-Ausgang zu Geräten niedrigerer Priorität. Dieser Anschluß wird normalerweise mit Pin 22 (DMA IN) verbunden. DMA OUT 7 führt zum Z-80-Teil.



Pinbelegung der Slots

Anschluß	Name	Beschreibung
25	+5 V	+5 V Stromversorgung. Für alle Peripheriekarten stehen insgesamt 3 A zur Verfügung.
26	GND	Elektrische Masse des Systems.
27	DMA IN	Daisy-Chain DMA-Eingang von Geräten höherer Priorität. Gewöhnlich mit Anschluß 24 (DMA OUT) verbunden.
28	INT IN	Daisy-Chain Interrupt-Eingang von Geräten höherer Priorität. Gewöhnlich mit Anschluß 23 (INT OUT) verbunden. INT IN von Slot 2 kommt von der seriellen Schnittstelle der Tastatur.
29	<u>NMI</u>	Nicht maskierbarer Interrupt (hardwaremäßiges Einschieben eines speziellen Unterprogrammes). Wenn diese Leitung auf 0 gezogen wird, beginnt der BASIS 108 einen Interrupt-Ablauf und springt dann zu einem Interrupt-Behandlungs-Programm auf Adresse \$3FB.
30	<u>IRQ</u>	Maskierbarer Interrupt (Interrupt ReQuest). Wenn diese Leitung auf logisch 0 liegt und das I-Bit des 6502-Mikroprozessors (Interrupt Sperre) nicht gesetzt ist, beginnt der BASIS 108 einen Interrupt Ablauf und springt zu dem Interrupt-Behandlungsprogramm, dessen Adresse in den Speicherzellen \$3FE und \$ 3FF zu finden sind.
31	<u>RES</u>	Wird dieser Anschluß auf logisch 0 gelegt, so beginnt der Mikroprozessor einen (RESET)-Ablauf.
32	<u>INH</u>	Wenn diese Leitung auf 0 gezogen wird, wird der obere 12 K Adressraum auf der Platine abgeschaltet. Diese Leitung wird durch einen 1 kOhm Widerstand auf +5 V gehalten.
33	-12 V	-12 V Spannungsversorgung. Der Maximalstrom beträgt 0,5 A für alle Peripheriekarten zusammen.
34	-5 V	-5 V Spannungsversorgung. Der maximal zulässige Strom beträgt für alle Peripheriekarten zusammen 0,5 A.

An-schluß	Name	Beschreibung
35		darf nicht beschaltet werden.
36	7M	7 MHz Takt. Diese Leitung treibt zwei LS-TTL-Lasten.
37	Q3	Asymmetrischer 2 MHz Takt. Dieser Anschluß treibt zwei LS-TTL-Lasten.
38	$\Phi_1$	Phase 1-Takt des Mikroprozessors. Dieser Anschluß kann zwei LS-TTL-Lasten versorgen.
39	USER 1	Wenn diese Leitung auf 0 gezogen wird, ist der \$Cxxx-Bereich unterbrochen.
40	$\Phi_0$	Phase 0-Takt des Mikroprozessors. Dieser Anschluß kann zwei LS-TTL-Lasten versorgen.
41	<u>DEVICE SELECT</u>	Leitung wird auf jedem Peripherieanschluß aktiv (logisch 0), wenn der Adressbus eine Adresse zwischen \$C0n0 und \$C0nF gespeichert hat, wobei n die um \$8 erhöhte Slotnummer angibt. Diese Leitung treibt 10 LS-TTL-Lasten.
42-49	D0-D7	In zwei Richtungen gepufferter Datenbus. Die Dateninformation auf dieser Leitung liegt mindestens 300 ns in Phase 0 beim Schreiben und sollte beim Lesen nicht länger als 100 ns vor dem Ende von $\Phi_0$ erhalten bleiben.
50	+12 V	+12 V Stromversorgung. Bis zu 2,5 A können insgesamt an alle Peripheriekarten abgegeben werden.

### Die Diskettenlaufwerke

Die Verwendung von Diskettenlaufwerken in Verbindung mit dem BASIS 108 System ist weitaus schneller und einfacher als die Verwendung eines Kassettenrekorders. Jedes BASIS 108 System ist mit Halteblechen für zwei Diskettenlaufwerke ausgerüstet. Wenn keine Laufwerke eingebaut sind, befinden sich die Befestigungsschrauben für die Laufwerke in einer kleinen Plastiktüte an den Halteblechen.

Falls Sie Diskettenlaufwerke nachträglich montieren wollen, dann schrauben Sie nach Abnehmen des Gehäusedeckels die Haltebleche von dem Gehäuseboden ab.

**Wichtig:** Vergewissern Sie sich, ob auch der Netzstecker gezogen ist und die kleine rote Kontroll-Lampe auf der Hauptplatine aus ist.

Montieren Sie jetzt mit Hilfe der mitgelieferten Schrauben die Halbleche an die Laufwerke und setzen anschließend die komplett montierten Einheiten wieder an ihren Platz zurück. Bevor Sie die Bleche am Gehäuseboden fest montieren, legen Sie einmal den Gehäusedeckel auf das System und kontrollieren Sie, ob die Laufwerke genau in den dafür vorgesehenen Ausschnitt im Gehäusedeckel passen. Zentrieren Sie die Diskettenlaufwerke und schrauben Sie diese dann fest. Die Flachbandkabel von den Laufwerken verbinden Sie mit der Laufwerkssteuerkarte (Controller), wobei das linke Laufwerk das Laufwerk 1 oder A und das rechte Laufwerk 2 oder B sein sollte. Eine entsprechende Beschriftung finden Sie an den Steckerleisten des Controllers. Wenn das Kabel von den Laufwerken zur Steuerkarte nicht richtig aufgesteckt wird, können an den Diskettenlaufwerken und am Controller erhebliche Schäden auftreten.

**Wichtig:** Achten Sie darauf, daß der Stecker richtig auf der Stifteleiste des Controllers sitzt. Das Kabel zeigt am Controller nach unten.

Setzen Sie nun die Steuerkarte in den Erweiterungssteckplatz 6 ein. Die Flachbandkabel-Anschlüsse zeigen zur Rückwand.

Je nach eingesetztem Betriebssystem sind die üblichen Plätze für weitere Diskettenlaufwerke die Slots (Stekkleisten) 4, 5 und/oder 7. Achten Sie hier bitte auf die Angaben in den entsprechenden Betriebshandbüchern. Da die weiteren Laufwerke nicht eingebaut werden, müssen die Flachbandkabel durch den Durchbruch auf der Rückseite von den Laufwerken zu den Steckkarten geführt werden.

### Pflege der Diskettenlaufwerke und der Disketten

Diskettenlaufwerke sind mechanische Geräte mit Motoren und anderen, sehr empfindlichen beweglichen Teilen. Daher sind sie etwas anfälliger als der BASIS 108 ohne Laufwerke. Rauhe Behandlung, wie Stöße, können zu Beschädigungen führen.

Die Diskette ist eine Plasticscheibe mit einer Beschichtung ähnlich der eines Tonbandes. Auf der Oberfläche können Informationen gespeichert oder von dort wieder abgerufen werden.

Die Diskette ist zum Schutz vor Staub und Kratzern in einer schwarzen Plastikhülle eingeschweißt. Innerhalb dieser Hülle kann sich die Diskette frei drehen.

Obwohl die Diskette relativ flexibel ist, vermeiden Sie bitte Verbiegen oder Knicke. Behandeln Sie auch die Hülle sorgfältig und stecken Sie sie sofort nach Gebrauch wieder in die zu jeder Diskette gehörende Papiertasche.

**Vermeiden Sie jegliche Berührung der Oberfläche der Diskette.**

**Fassen Sie die Diskette nur an ihrer Hülle an.**

Ein unsichtbarer Kratzer an der Oberfläche der Diskette oder lediglich ein Fingerabdruck können schon Fehler hervorrufen.

Legen Sie Disketten niemals auf schmutzige oder fettige Oberflächen und lassen Sie sie nicht verstauben.

Verwenden Sie einen Filzstift zum Beschriften der Diskettenaufkleber, wobei der Aufkleber erst nach dem Beschriften auf die Diskette geklebt werden sollte.

**Halten Sie Disketten von Magnetfeldern fern, legen Sie sie nicht auf Bildschirmgeräte.**

Disketten sind sehr empfindlich gegen extreme Temperaturen. Legen Sie sie nie in die Sonne oder in unmittelbare Nähe anderer Heizquellen, da sich die Disketten sonst wellen und nicht mehr gelesen werden können.

Bei sorgfältiger Pflege haben Disketten eine lange Lebensdauer.

#### **Einlegen und Herausnehmen von Disketten**

Das Laufwerk wird geöffnet und die Diskette mit dem Aufkleber nach oben hineingeschoben, wie es auf der Abbildung zu sehen ist. Die Kante mit dem ovalen Ausschnitt in der Hülle muß dabei zuerst hineingeschoben werden.

Schieben Sie die Diskette langsam hinein, bis sie vollständig im Laufwerk steckt. Biegen Sie sie dabei auf keinen Fall und schieben Sie nicht zu fest. Schließen Sie die Laufwerksklappe.



**Einlegen der Diskette**

Die Diskette wird herausgenommen, indem Sie das Laufwerk öffnen und die Diskette

vorsichtig herausziehen. Beim Öffnen der Laufwerksklappe wird auch gleichzeitig der Andruck für den Lese-/Schreibkopf gelöst. Es kann aber evtl. noch weiter geschrieben werden, was zu Datenverlust führen kann.

**Wichtig:** Nehmen Sie niemals eine Diskette aus dem Laufwerk, solange die rote Lampe des Laufwerks leuchtet, das kann die abgespeicherten Informationen zerstören.

Wenn Sie eine Diskette im Laufwerk lassen wollen, ohne mit dem System zu arbeiten, so empfiehlt es sich, die Laufwerksklappe zu öffnen, so daß der Kopf nicht auf der Diskette aufliegt.

## **KAPITEL 2**

### **INHALTSVERZEICHNIS**

#### **Software**

- 23 Einleitung
- 24 UCSD p-System IV.0
- 27 Das CP/M-System
- 28 Das DOS3.3-System

## SOFTWARE

### Einleitung

Das BASIS 108 System ist mit einem Monitor ROM ausgestattet, der das System automatisch startet, s. auch Kapitel 4. Damit haben Sie Zugriff zu den in diesem Kapitel beschriebenen Möglichkeiten des Monitor ROMs und können Ihre eigenen Betriebssysteme aufbauen.

Wahrscheinlich wird es allerdings so sein, daß Sie auf ein vorhandenes oder beim Kauf des BASIS 108 gleichzeitig erworbenes Betriebssystem zurückgreifen, um in einer der herkömmlichen Programmiersprachen auf Ihrem BASIS 108 arbeiten zu können.

Diese Betriebssysteme sind in der Regel auf Disketten abgelegt. Die Arbeit mit einem Kassettenrekorder ist möglich, aber sehr zeitaufwendig.

Die Betriebssysteme stellen im Prinzip nichts anderes dar als Arbeitshilfen, die es Ihnen ermöglichen auf einfachere und zugänglichere Weise mit Ihrem Computer zu sprechen. D.h. es hat Ihnen schon jemand die Arbeit des Umarbeitens Ihrer Programme in eine dem Computer verständliche Sprache abgenommen.

Grundsätzlich ist es so, daß diese Betriebssysteme in entsprechender Weise geladen werden müssen.

Hier sollen nicht alle möglichen Betriebssysteme angesprochen werden, sondern nur die nach unserer Erfahrung gebräuchlichsten:

UCSD p-System IV.0,  
CP/M,  
DOS3.3.

Es sei nochmals darauf hingewiesen, daß eine ganze Reihe anderer Betriebssysteme auf dem BASIS 108 möglich sind, eventuell ist eine vorherige Anpassung notwendig. Spezialfälle sollten Sie über Ihren Händler erfragen.

So sind alle Betriebssysteme, die für den Apple II angeboten werden oder die Sie von diesem Gerät noch besitzen, kompatibel. Das Apple Pascal ist das UCSD p-System II.1 und damit eine Teilmenge aus dem hier besprochenen UCSD p-System IV.0. Unterschiede im Betrieb werden kurz angesprochen. Die Firma Apple entwickelte speziell für die Umgebung von Basic das DOS-System, das nach Erstellen der ZAP-Diskette, s. Anhang A, vollständig kompatibel ist.

Auf die über diese Betriebssysteme möglichen Programmiersprachen können wir im Rahmen dieses Handbuches nicht eingehen, die gängigsten Sprachen in den einzelnen Systemen werden aber entsprechend erwähnt.

Andererseits soll dieses Kapitel nicht die Betriebshandbücher ersetzen, sondern Ihnen die Möglichkeiten aufzeigen und Ihnen eventuell die Entscheidung für das eine oder andere System erleichtern.

## **UCSD p-System IV.0**

Das UCSD IV.0 Betriebssystem ist ein Programmentwicklungs werkzeug für Microcomputersysteme, erstellt von der University of California San Diego.

Für den BASIS 108 steht Ihnen eine Interpreter Implementation des UCSD IV.0 Pascal zur Verfügung. Das bedeutet, daß ein Compiler Ihre Programme in einen Pseudo-Code (P-Code) übersetzt. Dieser Code ist unabhängig vom jeweiligen Mikroprozessor. Während der Ausführung des Programmes wird der P-Code durch ein Assemblerprogramm interpretiert und auf dem 6502 Prozessor des BASIS 108 ausgeführt. Auch die Module des Betriebssystems sind Pascalprogramme und werden in der gleichen Weise wie die Benutzerprogramme ausgeführt.

Es besteht aus den Programm-Modulen Editor, Compiler, Linker, Assembler, Filer und einem Debugger.

Wenn Sie Ihr System starten, erscheinen in der oberen Bildschirmzeile die System Kommandos, mit denen Sie durch Drücken des Anfangsbuchstaben die obigen Programm-Module anwählen können.

Kommando-Zeile:

Command: E(dit,R(un,C(omp,L(ink,X(ecute,A(ssem, D(ebug,? [IV.0 B3n]

Beschreibung der Kommandos:

E

ruft den bildschirmorientierten Texteditor auf, der eine recht komfortable Textverarbeitung zuläßt. Der bearbeitete Text wird vom Betriebssystem nach Abschluß der Textbearbeitung unter dem Namen SYSTEM.WRK.TEXT auf der Diskette gesichert und wird im folgenden mit Workfile bezeichnet.

R

übersetzt den Workfile, sofern es ein Programm in einer höheren Sprache ist, durch den Compiler in den P-Code und führt das Programm anschließend aus. Entspricht der Text nicht der Syntax, so erfolgt eine Fehlermeldung. Ist die Übersetzung des Workfile in den P-Code erfolgreich, so wird dieser Codefile unter dem Namen SYSTEM.WRK.CODE abgespeichert. Dieser Codefile kann jederzeit über R ausgeführt werden.

F

Startet das Programm-Modul Filer und es erscheint eine neue Kommandozeile:

Filer: G(et,S(ave,W(hat,N(ew,L(dir,R(em,C(hng,T(rans,D(ate,? [C.12a]

Mit den Filerkommandos verwalten Sie das aktuelle Datum, ihren Arbeitsfile (sichern, löschen, bestehende Files bearbeiten) und ihre Programme. Sie können Programme transferieren, Programmnamen ändern und sich den Inhalt der Disketten ansehen (näheres siehe Betriebshandbuch).

C

Startet das Programm-Modul Compiler, das den anzugebenden Programmtext xxx.TEXT einer höheren Programmiersprache in den P-Code übersetzt und bei erfolgreicher Compilierung unter xxx.CODE sichert. xxx ist der Name den der Benutzer selbst festlegt.

L

Ruft das Programm-Modul Linker auf, welches den P-Code mit dem echten Maschinen-Code verbindet. Es wird vornehmlich zum Verbinden von Assemblerroutinen mit Hauptprogrammen höherer Programmiersprachen benötigt.

X

Durch dieses Kommando werden übersetzte Programme, die unter dem Namen xxx.CODE auf der Diskette verfügbar sind, ausgeführt.

A

Assemblerprogramme, die mit dem Texteditor erstellt worden sind, werden in einen echten Maschinen-Code übersetzt und können mit dem Linker in Hauptprogramme höherer Programmiersprachen eingebunden werden.

D

Der Debugger ist eine zusätzliche Hilfe bei der Fehlersuche in bereits compilierten Programmen. Er kann von der Kommandozeile aus und auch während der Programmausführung aufgerufen werden und erleichtert das Auffinden von Fehlern, die der Compiler nicht berücksichtigt (z.B. logische Fehler im Programmablauf).

Die Leistungsfähigkeit des Betriebssystems UCSD IV.0 wird durch die Verfügbarkeit von Bibliotheksprogrammen unterstrichen.

Prozeduren und Functions, die häufig benötigt werden, können in der System-Bibliothek abgelegt werden (SYSTEM LIBRARY). Programme höherer Programmiersprachen können nun diese Routinen benutzen.

#### Inhaltsverzeichnis der vier notwendigen Disketten

##### 108.1:

SYSTEM.BOOT	10	31-May-82
SYSTEM.SBIOS	7	31-May-82
SYSTEM.INTERP	28	28-May-82
SYSTEM.MISCINFO	1	27-May-82
SYSTEM.FILER	33	19-Oct-81
SYSTEM.LIBRARY	11	28-Jan-82
SYSTEM.SYNTAX	14	4-Dec-80
SYSTEM.PASCAL	103	3-Jun-82
SYSTEM.WRK.TEXT	4	3-Jun-82
SYSTEM.WRK.CODE	2	3-Jun-82

**108.2:**

SYSTEM.COMPILER	96	5-Jan-82
SYSTEM.SYNTAX	14	4-Dec-80
SYSTEM.EDITOR	49	7-Dec-81
LIBRARY.CODE	13	7-Dec-81
SUNITS.LIBRARY	52	31-May-82
ID.TEXT	4	31-May-82
KEYWORDS.TEXT	4	31-May-82
WINDOW.CODE	2	25-May-82
DISPLAY.CODE	2	25-May-82
WINDOW.TEXT	4	25-May-82
DISPLAY.TEXT	4	25-May-82

**108.3:**

SYSTEM.ASSEMBLER	46	7-Dec-81
6500.OPCODES	2	20-Dec-78
6500.ERRORS	7	23-Sep-80
SYSTEM.LINKER	26	7-Dec-81
SYSTEM.EDITOR	49	7-Dec-81
LIBRARY.CODE	13	7-Dec-81
COMPRESS.CODE	10	7-Dec-81

**108.4:**

SETUP.CODE	27	7-Dec-81
BOOTER.CODE	3	7-Dec-81
DISKCHANGE.CODE	8	7-Dec-81
DISKSIZE.CODE	3	7-Dec-81
FINPARAMS.CODE	9	7-Dec-81
ABSWRITE.CODE	4	7-Dec-81
YALOE.CODE	12	7-Dec-81
SCREENTEST.CODE	13	7-Dec-81
DECODE.CODE	28	5-Mar-81
COPYDUPDIR.CODE	3	7-Dec-81
MARKDUPDIR.CODE	4	7-Dec-81
PATCH.CODE	34	7-Dec-81
COMPRESS.CODE	10	7-Dec-81
XREF.CODE	28	7-Dec-81
RECOVER.G.CODE	8	7-Dec-81
FORMATTER.CODE	14	7-Jan-82

Dies ist der Stand vom 18.6.1982. Sollten Sie neuere Versionen besitzen, so sind Abweichungen im Interesse des Fortschrittes möglich.

## **Das CP/M-System**

CP/M (Control Program for Microprocessors) der Firma Digital Research, USA, ist ein Steuerprogramm für Mikrocomputersysteme mit Disketten- und/oder Festplattenlaufwerken, speziell für Computer, die einen 8080/8085 oder Z-80 als Zentraleinheit haben und über mindestens 16 KByte Hauptspeicher verfügen. Beides trifft für den BASIS 108 zu.

Während Sie bei den UCSD-Systemen über das Drücken der jeweiligen Buchstabentaste den Befehlsablauf steuern, rufen Sie beim CP/M-System die jeweilige gewählte Funktion über das zusätzliche (RETURN) ab.

Die spezielle Sammlung von CP/M-Programmen machen durch einfache Systembefehle dem Benutzer alle vom Computer gesteuerten Hardwarekomponenten zugänglich. CP/M verwaltet darüber hinaus alle internen und externen Einheiten, unter anderen auch alle verfügbaren Speicherkapazitäten der Disketten und des Arbeitsspeichers, vollkommen selbstständig.

In den Arbeitsspeicher des Systems geladen, bildet CP/M einen integrierten Bestandteil des gesamten Systems. Der Benutzer kann mit CP/M in Dialog treten und beliebige Anwendungsprogramme starten.

CP/M ist in drei Funktionsmodule aufgeteilt:

CCP (Console Command Processor),  
BDOS (Basic Disk Operating System),  
BIOS (Basic Input/Output System).

CCP liest die Tastaturkommandos und erzeugt BDOS-System-Aufrufe.

Zum Lesen und Arbeiten von Programmiersprachen benötigt CP/M wie auch das oben besprochene UCSD p-System IV.0 einen entsprechenden Compiler oder Interpreter. Damit ist es dann möglich, praktisch in allen gängigen Programmiersprachen zu arbeiten, wobei das CP/M-System die Organisation übernimmt.

Ferner besitzt CP/M die Möglichkeit zum Assemblieren von Programmen und zum Einordnen von Asemblerprogrammen in die jeweils laufenden Programme.

Die Zahl der möglichen höheren Programmiersprachen ist sehr groß. Es gibt ausgezeichnete Textsysteme und andere Anwenderprogramme, so daß man hier ebenfalls ein umfassendes Betriebssystem zur Verfügung hat.

Im folgenden werden einige häufig vorkommende Kommandos aufgeführt und kurz beschrieben:

<b>ASM</b>	Assemblieren (8080) einer Datei.
<b>DDT</b>	Testen und Ändern von 8080-Maschinenprogrammen.
<b>DIR</b>	Anzeigen einer Liste aller auf der Diskette des selektierten Laufwerks verzeichneten Dateien.
<b>ERA</b>	Löschen einer oder mehrerer Dateien auf der Diskette.
<b>PIP</b>	Kopieroperationen von Dateien.

**SAVE** Sichern eines Speicherinhaltes als Disk-Datei.

**REN** Umbenennen einer Datei.

**SUBMIT** Ausführen einer Befehlsfolge.

Die Anwendung dieser und weiterer Programme entnehmen Sie bitte einem CP/M-Betriebshandbuch.

Es folgt der Inhalt der Diskette, die das CP/M-Betriebssystem enthält:

A: FORMAT	COM : DEUTSCH	COM
A: ASCII	COM : APL	COM
A: SYSWRT	BAS : PIP	COM
A: STAT	COM : ED	COM
A: ASM	COM : DDT	COM
A: LOAD	COM : SUBMIT	COM
A: XSUB	COM : DUMP	ASM
A: XSUB	COM	

Auch hier können sich Änderungen ergeben, Version vom 18.6.1982.

### Das DOS3.3-System

Um im DOS3.3 arbeiten zu können, muß es zunächst auf den BASIS 108 angepaßt werden. Das geschieht entsprechend Anhang A einmal. Dann geben Sie zunächst die ZAP-Diskette in Ihr Laufwerk 1, wählen die entsprechende Basic-Art und können dann nach Eingabe Ihrer DOS-Diskette arbeiten wie z.B. auf einem Apple, wenn Sie einige kleine Änderungen berücksichtigen.

Wie schon erwähnt, handelt es sich beim DOS3.3 eigentlich nicht um ein echtes Betriebssystem, sondern eher um eine Umgebung für Basic. D.h., hiermit lassen sich praktisch nur die entsprechenden Basic-Arten bearbeiten. Andererseits haben Sie hier die Möglichkeit, über entsprechende Befehle das Monitor ROM anzusteuern und in ihm zu arbeiten, s. Kapitel 4.

Da es aber eine Vielzahl von Anwenderprogrammen in Basic gibt, die speziell auf das DOS-System ausgelegt sind, ist auch dieses System attraktiv.

Die häufigsten Befehle mit einer kurzen Beschreibung:

**BRUN X** Lädt Maschinen-Programm X in Speicher und läßt es ablaufen.

**CATALOG** Gibt den Inhalt der im aktuellen Laufwerk liegenden Diskette an.

- DELETE X** Entfernt Programm X von der Diskette.
- IN # n** Steuert Slot n für Eingabe an.
- LOAD X** Lädt Basic-Programm X in den Speicher.
- PR # n** Steuert Slot n für Ausgabe an.
- RUN X** Lädt Basic-Programm X in Speicher und läßt es ablaufen.
- SAVE X** Speichert Basic-Programm X auf Diskette.

In dem entsprechenden Betriebshandbuch für DOS finden Sie diese und weitere Befehle und Funktionen und Ihre Anwendung. Hier ist also wie bei den anderen beiden Betriebssystemen nur ein kleiner Ausschnitt aus den Möglichkeiten aufgeführt.

Die hier im folgenden abgedruckte Inhaltsliste der DOS3.3 SYSTEM MASTER Diskette enthält nicht die möglichen Spiele oder Demonstrationsprogramme:

A	006	DOS3.3
B	010	BOOT13
I	009	COPY
B	003	COPYA.OBJ0
A	009	COPYA
B	020	FID
B	050	FPBASIC
B	050	INTBASIC
B	009	MASTER CREATE
B	027	MUFFIN
A	010	RANDOM
A	013	RENUMBER
A	002	DISPLAY
B	002	DISPLAY SPEC
A	006	BAUD
A	006	PRINTER/V24

Auch hier können sich je nach der verwendeten Version Änderungen ergeben.

## KAPITEL 3

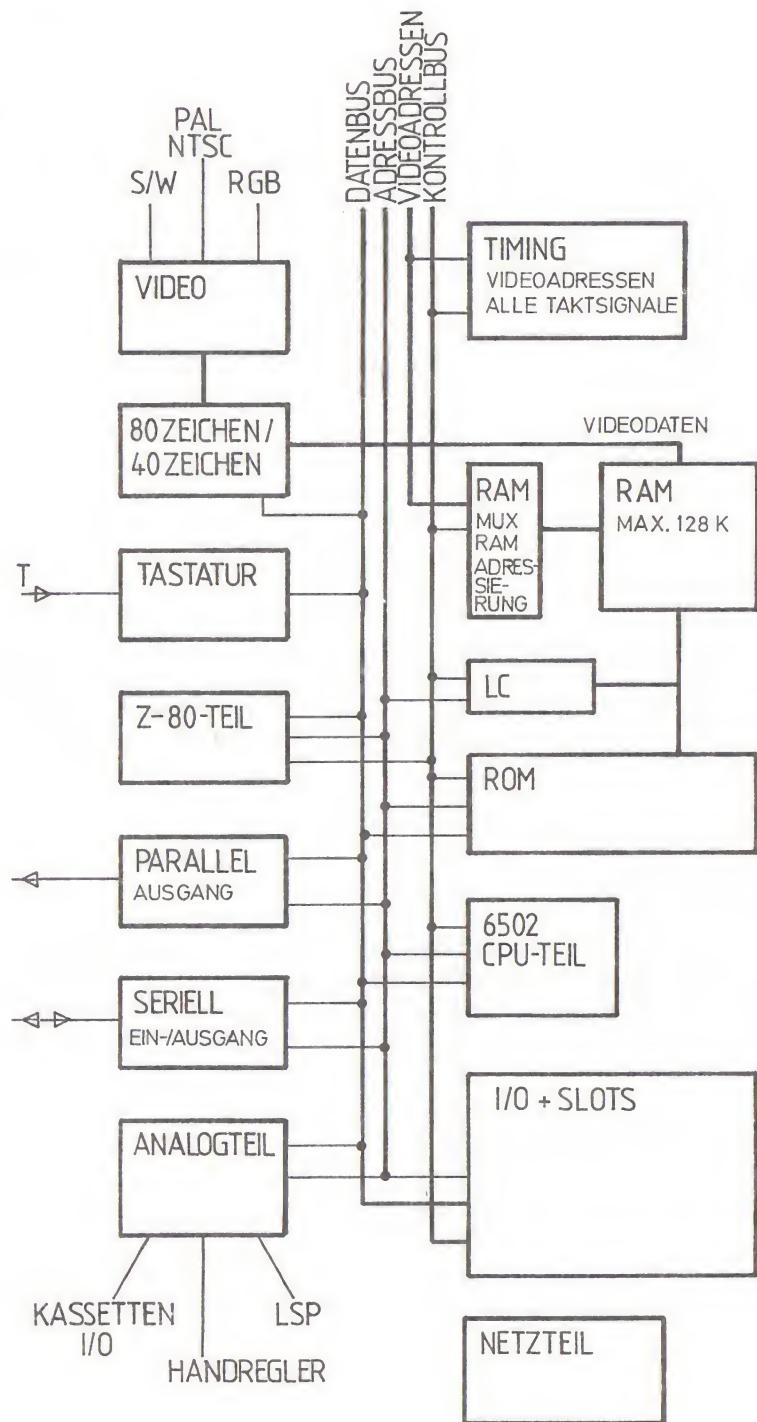
### INHALTSVERZEICHNIS

#### Zugriff zur Hardware

- 31 Logischer Schaltplan
- 32 Text- und Graphikdarstellung
- 32 Der Textbildschirm
- 32 80/40 Zeichendarstellung
- 33 Das Prinzip der 80-Zeichendarstellung
- 33 Softwareschalter für die Textdarstellung
- 33 Softwareschalter für die Graphik
- 33 LO-RES-Graphik
- 34 MI-RES-Graphik
- 34 HI-RES-Graphik
- 34 Farbdarstellung der HI-RES-Graphik
- 35 Zeichengenerator
- 36 Tastatur

## Logischer Schaltplan

Zum besseren Verständnis der folgenden Kapitel wird hier zunächst der logische Schaltplan aufgeführt.



## Text- und Graphikdarstellung

Das BASIS 108 Computersystem kann sowohl Text als auch Graphik darstellen. Zur Darstellung von Text oder LO-RES-Graphik (niedrige Auflösung) und MI-RES-Graphik (mittlere Auflösung) stehen 2 Bereiche (Seiten) und für die HI-RES-Graphik (hohe Auflösung) zwei weitere Bereiche zur Verfügung. Diese Bereiche sind direkt im Adressraum der Mikroprozessoren untergebracht.

Der Textbildschirm kann entweder 40 oder 80 Zeichen in 24 Zeilen - je nach ausgewähltem Mode - darstellen. Die gleichen Seiten werden auch für die niedrig auflösende Graphik genutzt, so daß sich im Graphik Mode entweder 40 x 48 Blöcke oder 80 x 48 Blöcke in 16 Farben darstellen lassen.

Ein weiterer Bereich des Speichers wird für 2 Seiten der HI-RES-Graphik mit einer Auflösung von 280 x 192 Punkten in 6 Farben genutzt.

1. Textseite \$0400-\$07FF (Text oder LO-, MI-RES-Graphik)  
2. Textseite \$0800-\$BFFF

1. Graphikseite \$2000-\$3FFF (HI-RES-Graphik).  
2. Graphikseite \$4000-\$5FFF

## Der Textbildschirm

Die erste Seite des Textbildschirmes liegt auf der Adresse \$0400 und reicht bis zur Adresse \$07FF, die zweite Seite schließt direkt mit der Adresse \$0800 an und reicht bis zu Adresse \$OBFF. Über die Softwareschalter \$C054 (Seite 1) und \$C055 (Seite 2) kann die jeweils auf dem Bildschirm darzustellende Seite ausgewählt werden.

## 80/40-Zeichendarstellung

Für die 80-Zeichendarstellung wurde dem Adressbereich der beiden Textseiten ein 2 KByte statisches RAM parallel geschaltet. Dieses statische RAM wird mit den gleichen Adressen angesprochen wie auch die normalen Textseiten. Beim Schreiben in die Textseiten wird über einen Softwareschalter die entsprechende Seite ausgewählt.

\$C00Dw aktiviert das statische RAM,  
\$C00Cw aktiviert den Standard-Bereich.

Durch diesen Schalter werden immer beide Textseiten umgeschaltet.

Die Adresse \$C00Bw schaltet die 80-Zeichen Darstellung ein und \$C00Aw wieder aus. Das statische RAM kann aber unabhängig von diesem Schalter beschrieben oder gelesen werden.

## **Das Prinzip der 80-Zeichendarstellung**

Der Bildschirmwiederholungsspeicher kann nur in den Augenblicken ausgelesen werden, in denen der Mikroprozessor keine Speicherzugriffe durchführt. Dieses ist immer der Fall, wenn der Takt des Prozessors auf logisch 0 liegt. Dieses wird nun genutzt, um ein Zeichen für den Bildschirm aus dem Speicher zu lesen. Die Darstellung von 80 Zeichen in einer Zeile würde aber verlangen, daß auch während der anderen Taktphasen ein Zeichen gelesen werden muß. Damit die Kompatibilität zum Apple erhalten bleibt, ist dies aber ohne wesentliche Veränderung nicht möglich. Im BASIS 108 werden deshalb 2 Zeichen gleichzeitig ausgelesen. Ein Zeichen aus dem Standard RAM und ein Zeichen aus dem statischen RAM. Diese Zeichen werden zwischengespeichert und können dann unabhängig vom Mikroprozessortakt weiter verarbeitet werden.

Diese Technik bedingt, daß sich alle Zeichen mit einer geraden Platznummer im Standard RAM und alle mit einer ungeraden im statischen RAM befinden. Das statische RAM kann, wenn es selektiert wurde, vom Mikroprozessor ausgelesen werden.

## **Softwareschalter für die Textdarstellung**

\$C054	Seite 1 aktiv,
\$C055	Seite 2 aktiv,
\$C00Aw	80 Zeichendarstellung aus,
\$C00Bw	80 Zeichendarstellung ein,
\$C00Dw	statisches RAM selektiert,
\$C00Cw	Standard RAM selektiert .

## **Softwareschalter für die Graphik**

### LO-RES-Graphik

Die LO-RES-Graphik benutzt die gleichen Bereichen, wie die Textseiten und ist daher ebenfalls auf 2 Seiten vorhanden. In dieser Graphikart können entweder 40 x 48 Blöcke in 16 Farben (Vollgraphik) oder 40 x 40 Blöcke mit 4 Zeilen Text am unteren Bildschirmrand (mixed Graphik) dargestellt werden. Die Anwahl geschieht mit Hilfe von Softwareschaltern.

Schalter für die LO-RES-Graphik:

\$C050	Graphik einschalten,
\$C051	Graphik ausschalten,
\$C056	LO- + MI-RES-Graphik,
\$C053	mixed (4 Zeilen Text werden eingeblendet),
\$C052	Vollgraphik (die Textzeilen werden ausgeblendet),
\$C00Aw	80 Spalten aus.

## MI-RES-Graphik

Die MI-RES-Graphik stellt  $80 \times 48$  Blöcke oder  $80 \times 40$  Blöcke in 6 Farben dar. Sie besitzt die selben Möglichkeiten, wie die LO-RES-Graphik, nur wird das statische RAM zur 80 Zeichendarstellung mitverwendet. Es gelten die gleichen Bedingungen für das statische RAM wie bei der 80 Zeichen Textdarstellung.

Schalter für die MI-RES-Graphik:

```
$C050 Graphik einschalten,  
$C051 Graphik ausschalten (nur Text),  
$C056 LO- + MI-RES-Graphik ein, HI-RES aus,  
$C053 mixed (4 Zeilen Text werden eingeblendet),  
$C052 Vollgraphik,  
$C00Bw 80 Spalten ein.
```

Weiterhin sind für die Programmierung noch die Schalter \$C00D und \$C00C für das Beschreiben des statischen RAMs notwendig.

## HI-RES-Graphik

Die HI-RES-Graphik ist eine hochauflösende Farbgraphik mit  $280 \times 192$  Punkten in 6 Farben. Auch diese Graphikart hat 2 Seiten im Speicher; Seite 1 im Adressbereich \$2000 bis \$3FFF und Seite 2 von \$4000 bis \$5FFF. Die HI-RES-Graphik kann als Vollgraphik ( $280 \times 192$  Punkte) oder als mixed Graphik ( $280 \times 160$  Punkte) mit 4 Zeilen Text am unteren Bildrand betrieben werden. In diesem Mode wird als Text der Inhalt der entsprechenden Textseite mit 40 oder 80 Zeichen pro Zeile eingeblendet.

Schalter für die HI-RES-Graphik:

```
$C050 Graphik einschalten,  
$C051 Graphik ausschalten (Text ein),  
$C057 HI-RES-Graphik ein,  
$C053 mixed HI-RES-Graphik,  
$C052 Vollgraphik.
```

## **Farbdarstellung der HI-RES-Graphik**

Jeder Punkt auf dem Bildschirm repräsentiert ein Bit aus dem Bildspeicher. Von den 8 Bit eines jeden Bytes werden die Bits 0 . . . 6 auf dem Bildschirm dargestellt, das Bit 7 bestimmt die Farben der Punkte in diesem Byte. Auf einem S/W Bildschirm erscheint ein Punkt, wenn das Bit logisch 1 ist, und kein Punkt, wenn es logisch 0 ist.

Auf einem Farbbildschirm ist dies nicht ganz so einfach. Hier ist die Bit-Position für die dargestellte Farbe wichtig. Ist ein Bit auf einer ungeraden Position an, stellt es entweder grün oder hellblau dar. Ist ein Bit auf einer geraden Position an, ergeben sich die Mischfarben aus rot und grün oder aus hellblau und violett. Die zweite Kombination (hellblau, violett) ist nur dann gültig, wenn das 8 Bit des

entsprechenden Bytes an ist. Innerhalb eines Bytes ist es nicht möglich, die Farbgruppe zu wechseln. Die hier genannten Farben können je nach Bildschirmtyp und Einstellung voneinander abweichen.

### Zeichengenerator

Im BASIS 108 Computersystem ist der Zeichengenerator in einem 4 KByte EPROM (2732 Typ) untergebracht. In diesem EPROM können bis zu 5 Zeichensätze untergebracht werden. Durch 4 Softwareschalter kann der gewünschte Zeichensatz ausgewählt werden. Wenn der Schalter SW 3 (\$C006) auf logisch 1 steht, ist ein Zeichensatz mit 128 Zeichen normal, 64 Zeichen invertiert und 64 Zeichen blinkend ausgewählt. Ist dieser Schalter auf logisch 0, können 4 weitere Zeichensätze angewählt werden.

### Zeichengenerator

			SW0	SW1	SW2	SW3
Satz 0	Standard Apple II	64 Zeichen	0	0	0	0
Satz 1	Standart ASCII	128 Zeichen	x	1	0	0
Satz 2	Deutsch	128 Zeichen	x	0	1	0
Satz 3	APL	128 Zeichen	x	1	1	0
(In Ländern außerhalb des deutschen Sprachraumes kann Satz 2 und 3 vertauscht sein.)						

Adresse	Schalter
\$C000w	SW0 aus
\$C001w	SW0 ein
\$C002w	SW1 aus
\$C003w	SW1 ein
\$C004w	SW2 aus
\$C005w	SW2 ein
\$C006w	SW3 aus
\$C007w	SW3 ein,

EIN entspricht logisch 1 und AUS logisch 0.

Der Schalter SW0 kann den Zeichensätzen 1-3 entweder INVERSE oder FLASHING (Blinken) als Sonderdarstellung zuordnen:

\$C000w Inverse,  
\$C001w Flash.

## Die Tastatur

Die Tastatur besteht aus einer erweiterten Schreibmaschinentastatur, einem numerischen Zehnerblock mit Tasten für die Grundrechenoperationen, einem Cursorsteuerfeld und 15 Zusatztasten. Sie ist in einem sehr flachen Kunststoffgehäuse untergebracht und mit dem BASIS 108 über ein 16-adriges Kabel verbunden. Die Steckerbelegung des Tastatkabels finden Sie auf Seite 8.

Die 15 Zusatztasten sind vierfach belegt. Sie werden wie normale Tasten verwandt, gehen aber über den ASCII-Zeichensatz hinaus. Unter CP/M 3.0 können sie softwaremäßig mit einigen Funktionen belegt werden.

Groß-/Kleinschreibung wird durch die SHIFT-Taste erreicht, die durch Drücken der Taste LOCK festgesetzt wird, SHIFT-LOCK. Im alphanumerischen Tastenfeld sind Umlaute und Sonderzeichen vorhanden. Sollen nur die Buchstaben großgeschrieben werden, die übrigen Tasten aber mit ihrer unteren Belegung erscheinen, so drückt man gleichzeitig CTRL-LOCK. Alle Tasten sind mit Autowiederholung ausgerüstet, das bedeutet, daß sich bei längerem Druck auf die Taste das gedrückte Zeichen automatisch wiederholt.

### BASIS 108 Tastatur

Anzahl der Tasten	:	100
Codierung	:	ASCII mit Sonderzeichen
Anzahl der Tastencodes	:	128 ASCII und 70 Funktionen
Ausgang	:	TTL
Betriebsspannung	:	+12 Volt .

Die Dekodierung der Tastenbelegung erfolgt auf der Hauptplatine des BASIS 108 in einem ROM. Hierdurch ist eine flexible Tastenbelegung durch Austauschen des ROMs gegeben. Außerdem besteht die Möglichkeit, über einen Softwareschalter (\$C009 ein, \$C008 aus) die Abfrage der Tastatur über einen Interrupt zu steuern.

Auf einer zweiten Eingabeadresse kann der geübte Programmierer den Status bestimmter Funktionen der Tastatur abfragen (siehe auch nächste Seite):

- CONTROL-Taste gedrückt,
- SHIFT-Taste gedrückt,
- Funktionstaste gedrückt.

Außerdem können auf dieser Adresse noch die folgenden Statusinformationen, die nicht mit der Tastatur zusammenhängen, abgefragt werden:

- Printer Return,
- HBL (Horizontal Austastsignal),
- Sync (Video Synchronisationssignal).

Die RESET-Funktion wird durch das gleichzeitige Drücken der beiden SHIFT- und der CRTL-Taste ausgelöst. Der Programmablauf wird unterbrochen und beim Loslassen der Tasten startet der Computer den RESET-Ablauf.

### Adressen der Tastatur

\$C000-\$C007	Lesen des ASCII-Code der Taste
\$C008-\$C00F	Lesen des Status der Tastatur
\$C009w	Tastaturinterrupt ein
\$C008w	Tastaturinterrupt aus

Das auf einer der Adressen \$C008-\$C00F gelesene Byte besitzt folgende Zuordnung:

Bit 7	Zusatztaste
Bit 6	Shifttaste
Bit 5	Controltaste
Bit 4	z.Z. nicht definiert
Bit 3	z.Z. nicht definiert
Bit 2	HBL (Horizontal Austastsignal)
Bit 1	Video Synchronisationssignal
Bit 0	Drucker aktiv

Im Anhang I finden Sie die Belegung der Tasten mit den einzelnen Zeichen und eine Tabelle in der die Tasten, der Code (hexadezimal) und das ASCII Zeichen aufgelistet sind.

## KAPITEL 4

### INHALTSVERZEICHNIS

#### Der Monitor

- 39 Einleitung
- 39 Einweisung
- 40 Daten und Adressen
- 40 Inhaltsüberprüfung einer Speicherstelle
- 41 Überprüfen mehrerer Speicherstellen
- 42 Änderung einer Speicherstelle
- 42 Änderung von aufeinanderfolgenden Speicherstellen
- 43 Übertragen eines Speicherbereichs
- 44 Vergleich von zwei Speicherbereichen
- 44 Programmieren und Starten von Maschinenprogrammen
- 46 Prüfen und Ändern von Registerinhalten des 6502
- 46 Weitere Monitor-Kommandos
- 47 Kleine Hilfen für den Umgang mit dem Monitor
- 48 Erzeugen eigener Kommandos
- 49 Übersicht über die Monitorkommandos
- 52 Liste ausgewählter Monitor-Unterprogramme
- 56 Spezialadressen des Monitors

## Einleitung

Der System-Monitor, ein kleines aber leistungsstarkes Programm, befindet sich auf der Hauptplatine in einem ROM (Read Only Memory). Das Monitor-ROM hat eine Speicherkapazität von 2 KByte und sitzt auf dem IC-Platz 25, in der vorderen Reihe auf der Platine. Mit Hilfe dieses Programmes werden Abläufe im System kontrolliert und gesteuert.

Sie hatten beim Kauf Ihres BASIS 108 die Wahl zwischen zwei unterschiedlichen Monitoren. Das eine Monitor-ROM ist für den Einsatz des BASIS 108 mit Laufwerk vorgesehen und der andere Monitor-ROM beinhaltet die Schreib-/Leseroutinen für einen Kassettenrekorder. Eine Tabelle mit den Unterschieden der beiden Monitor-ROMs finden Sie im Anhang N. Beschreibung des Monitor-ROM für Kassettenrekorder (40 Zeichen/Zeile) mit den entsprechenden Routinen finden Sie im Anhang G.

Das Monitor-ROM erfüllt in den verschiedenen, möglichen Betriebssystemen unterschiedliche Aufgaben. Während es für das Betriebssystem UCSD IV.0 bzw. Apple Pascal nur für den Ladevorgang benötigt wird, wird es von den beiden weiteren üblichen Systemen CP/M und DOS ständig benötigt. Siehe auch die entsprechenden Betriebshandbücher.

## Einweisung

Das Programm des BASIS 108 Monitor-ROMs beginnt ab der Adresse \$FF69 (dezimal 65385 oder -151).

Aus einem BASIC-Programm können Sie den Monitor-ROM mit dem Befehl CALL -151 aufrufen. Haben Sie FP40 oder FP80 geladen, so kann das Monitor-ROM auch über SYS angesprochen werden.

Aus dem Betriebssystem CP/M kann man das Monitor-ROM nicht so einfach ansprechen. Bitte lesen Sie für diesen Fall das entsprechende CP/M-Handbuch.

Haben Sie kein System geladen so können Sie durch Abstellen des Laufwerkcontrollers mit (RESET) ebenfalls in den Monitor-ROM gelangen.

Der Monitor meldet sich auf dem Bildschirm mit einem Stern \* und rechts daneben der Cursor. Damit wird angezeigt, daß das Monitor-Programm bereit ist, von Ihnen einen Befehl zu empfangen.

Ihre Eingaben über die Tastatur dürfen bis zu 255 Zeichen lang sein und müssen mit (RETURN) beendet werden. Wenn Sie den Monitor verlassen wollen und zu der Sprache zurückkehren wollen, mit der Sie eben gearbeitet haben, dann drücken Sie Q oder System-(RESET) (gleichzeitig SHIFT-SHIFT-CTRL).

## Daten und Adressen

Die Kommunikation mit dem Monitor erfolgt über die Tastatur oder Ihr Programm. Sie tippen eine Zeile auf der Tastatur und geben danach (RETURN) ein. Der Monitor wird das verarbeiten, was Sie ihm eingegeben haben. Er kann folgende Arten an Informationen verwerten: Adressen, Werte und Befehle (Kommandos).

Adressen und Werte nimmt er nur in hexadezimaler Schreibweise an. Diese hexadezimale Schreibweise wird im Anhang H näher dargestellt.

Jede Adresse im BASIS 108 wird durch vier Hexadezimalziffern dargestellt und jeder Wert, Inhalt einer Speicherstelle, durch zwei Hexadezimalziffern. Wenn der Monitor die Eingabe einer Adresse erwartet (Stern mit danebenstehendem Cursor), akzeptiert er jede Gruppe von Hexadezimalziffern. Sind weniger als vier Ziffern in dieser Gruppe, so wird er führende Nullen ergänzen, gibt es mehr als vier Ziffern, so werden nur die letzten vier Ziffern ausgewertet. Entsprechend behandelt der Monitor die Eingabe der zweiziffrigen Datenwerte.

Der Monitor erkennt 22 verschiedene Kommandos. Einige sind Satzzeichen, andere sind Buchstaben oder Steuerzeichen. Das Monitor-ROM benötigt, wie Sie es von den verschiedenen Betriebssystemen her kennen, nur den ersten Buchstaben eines Kommandos, ein Kommando wird durch Steuerzeichen aufgerufen.

! Obwohl der Monitor das Steuerzeichen CTRL-B erkennt und richtig  
! interpretiert, wird es nicht auf dem Bildschirm sichtbar gemacht.

## Inhaltsüberprüfung einer Speicherstelle

In den folgenden Abschnitten werden die von Ihnen einzugebenden Werte fett gedruckt, wobei die Antworten, die der Monitor auf dem Bildschirm darstellt, normal gedruckt, aber groß geschrieben sind.

Wenn Sie die Adresse einer Speicherstelle eingeben, wird der Monitor wie folgt antworten:

- Wiederholung der eingegebene Adresse,
- ein Doppelpunkt,
- ein Leerzeichen,
- den Wert dieser Speicherstelle.

Beim Überprüfen der folgenden Beispiele können die auf dem Bildschirm ausgegebenen Speicherinhalte, solange Sie sie nicht in der vorgeschriebenen Form geändert haben, von den hier gedruckten Speicherinhalten abweichen.

Beispiel:

```
*20(RETURN)  
0020: 00  
*
```

## **Überprüfen mehrerer Speicherstellen**

Wenn Sie dem Monitor in einer Eingabezeile einen Punkt . und darauffolgend eine Adresse angeben, erhalten Sie einen Speicherauszug von der zuletzt angezeigten Adresse bis zu der eingegebenen Adresse. Die letzte der dabei angezeigten Adressen ist die Startadresse für weitere Anzeigen.

Beispiel:

```
*0 (RETURN)
0000: 04
*.11 (RETURN)
0001: C6 00 0A 1B 18 18 00 00 FF 4C FF FF 22 00 6B
0010: 00 00
*
```

Nachfolgend noch einige Bemerkungen zum Format eines Speicherauszugs:

1. Der Speicherauszug beginnt mit der auf die zuletzt angezeigten bzw. geöffneten Speicheradresse, im Beispiel oben also mit 0001.
2. Die anderen Zeilen beginnen mit Adressen, die mit einer Null enden.  
Bei dem Monitor-ROM, der mit 40 Zeichen/Zeile startet, sind die Zeilen aufgeteilt und beginnen mit Adressen, die mit einer 8 oder mit einer 0 enden.
3. Es werden entsprechend dem Monitor-ROM 8 bzw. 16 Werte in einer Zeile angezeigt.

Sie können die zwei Kommandos auf einmal eingeben: Tippen Sie die Anfangsadresse, dann einen Punkt und die Endadresse. Diese beiden Adressen, die durch einen Punkt getrennt wurden, nennt man Speicherbereich.

Beispiel:

```
*30.40 (RETURN)
0030: FF 00 FF AA 05 00 BD 9E 81 9E FF FF 36 00 41 00
0040: 30 00
*
```

Ein Druck auf (RETURN) veranlaßt den Monitor, eine Zeile mit dem Speicherauszug anzuzeigen. Der Speicherauszug beginnt bei der Adresse, die der zuletzt angezeigten oder geöffneten Adresse folgt, und endet bei der Adresse, die mit einem F endet. Wieder wird die zuletzt angezeigte Adresse als die zuletzt geöffnete und als nächste veränderbare Adresse betrachtet.

Beispiel:

```
*5 (RETURN)
0005: 18
*(RETURN)
18 00 00 FF 4C FF FF 22 00 6B
*(RETURN)
0010: 00 00 00 00 04 00 FF 00 FF FF FF FF FF FF FF
*
```

### Änderung einer Speicherstelle

In dem letzten Abschnitt haben Sie einiges über die nächste veränderbare Speicherstelle erfahren. Wenn Sie das folgende Beispiel durchführen, können Sie sehen, was wirklich passiert.

Tippen Sie einen Doppelpunkt und dann einen Wert.

Beispiel:

```
*0 (RETURN)
0000: 04
*:3C (RETURN)
*0 (RETURN)
0000: 3C
*
```

Sie sehen, daß der Wert des Speichers 0 den neuen Wert 3C hat.

Sie können das Öffnen und Ändern zu einer Anweisung zusammenfassen:

Beispiel:

```
*10:33 (RETURN)
*10 (RETURN)
0010: 33
*
```

Wenn Sie den Inhalt einer Speicherstelle verändern, verliert sie den alten Wert. Der neue Wert bleibt solange erhalten, bis er wiederum von einem anderen Wert überschrieben wird.

### Änderung von aufeinanderfolgenden Speicherstellen

Wenn Sie mehrere aufeinanderfolgende Speicherstellen verändern wollen, brauchen Sie nicht jede einzelne Speicherstelle so einzutippen, wie es im vorigen Abschnitt beschrieben wurde. Der Monitor ermöglicht es Ihnen, maximal 58 Speicherstellen auf einmal zu ändern. Dazu geben Sie die Anfangsadresse, einen Doppelpunkt und dann alle Werte ein.

Die Werte müssen durch Leerstellen voneinander getrennt sein.

Der Monitor trägt nun ab der angegebenen Anfangsadresse alle Werte der Reihe nach in die Speicherstellen ein. Wollen Sie noch mehr Speicherstellen ändern, brauchen Sie nicht die Adresse neu eingeben, sondern nur einen Doppelpunkt und die neuen Werte, sofern die Startadresse mit der nächsten, auf die zuletzt geänderten Adresse übereinstimmt.

Beispiel:

```
*0.7 (RETURN)
0000: 5F C6 00 0A 1B 18 18 00
*0: 6F 3A 1 B 1A 16 11 07 (RETURN)
*0.7 (RETURN)
0000: 6F 3A 01 0B 1A 16 11 07
*
```

### Übertragen eines Speicherbereichs

Der Inhalt eines Speicherbereichs (eingegrenzt durch zwei mit einem Punkt voneinander getrennte Speicheradressen) kann als ein Ganzes aufgefaßt werden und mit einem MOVE-Kommando des Monitors von einer Speicherstelle zu einer anderen gebracht werden. Dazu müssen Sie dem Monitor angeben, wo der Speicherbereich liegt und wo er hin soll.

Diese Information besteht aus folgenden Teilen:

- Der Zieladresse,
- einer linken spitzen Klammer (kleiner als),
- der Anfangs- und der Endadresse des Bereichs,
- einem M , damit der Monitor einen Transport (Move) durchführt.

Die Anfangs- und Endadresse geben Sie in gewohnter Weise an (durch einen Punkt getrennt).

Als Beispiel übertragen wir die Speicher 0 - 7 auf 100 - 107, zunächst lassen wir uns diese neuen Speicher ausdrucken:

```
*100.107 (RETURN)
0100: FF FF FF FF FF FF FF FF
*100<0.7M (RETURN)
*100.107 (RETURN)
0100: 6F 3A 01 0B 1A 16 11 07
*
```

Der Monitor kopiert die Werte des angegebenen Bereichs und überträgt sie an den Bestimmungsort. Der Original-Bereich bleibt unverändert. Die Endadresse des Originalbereichs wird jetzt die zuletzt geöffnete Adresse und die nächste veränderbare Adresse ergibt sich aus der Anfangsadresse des Originalbereichs. Ist die zweite Adresse des angegebenen Bereichs kleiner als die erste, so wird nur

ein Wert (nämlich der Wert der ersten Speicherstelle des Bereichs) übertragen. Liegt die Zieladresse des MOVE-Kommandos innerhalb des Originalbereichs oder überschneiden sich die beiden Bereiche, so werden die Bereiche speicherweise überschrieben und die Originalwerte der Zieladressen gehen verloren.

### Vergleich von zwei Speicherbereichen

Zwei Speicherbereiche können miteinander verglichen werden. Dazu verwenden Sie das selbe Format, wie Sie es soeben beim MOVE-Kommando kennengelernt haben. Mit dem Vergleichs-Kommando VERIFY lässt sich nach dem MOVE-Kommando feststellen, ob die Übertragung ordnungsgemäß abgelaufen ist.

Das VERIFY-Kommando benötigt wie das MOVE-Kommando eine Zieladresse und einen Bereich.

Der Monitor vergleicht den Inhalt des angegebenen Bereichs mit dem Inhalt des Bereichs ab der Zieladresse.

! Sind die Bereiche gleich, so erfolgt keine Ausgabe.

Sollten Unterschiede erkannt werden, so gibt der Monitor die Adresse mit den jeweiligen unterschiedlichen Inhalten aus.

Beispiel:

*100<0.7V(RETURN)	c(1)
*100<0.8V(RETURN)	c(2)
0008: 00 FF	c(3)
*	

Herrscht Übereinstimmung wie in (1) (c hier als Kommentar), so erfolgt kein Ausdruck. Im Fall (2) besteht keine Übereinstimmung, es sei denn rein zufällig, deshalb hier der Ausdruck (3).

Beide Adressen bleiben unverändert. Die letzte geöffnete und die nächste veränderbare Adresse ergibt sich jeweils wie im MOVE-Kommando. Wenn die Endadresse kleiner ist als die Anfangsadresse, wird nur die Anfangsadresse verglichen. Auch bei VERIFY kommt es zu Schwierigkeiten, wenn die Zieladresse im Originalbereich liegt.

### Programmieren und Starten von Maschinenprogrammen

Viele Programme, die in einer höheren Programmiersprache, wie BASIC oder CP/M geschrieben sind, greifen auf Unterprogramme zu, die in der Maschinensprache eines der auf der Hauptplatine des BASIS 108 untergebrachten Mikroprozessors, des 6502, geschrieben wurden.

Der Monitor hat spezielle Befehle, um den Programmierern, die sich mit der Maschinensprache des 6502 befassen, beim Erstellen von Unterprogrammen zu helfen.

Sie können ein Maschinenprogramm schreiben und die hexadezimalen Werte der

Befehlsteile und der zugehörigen Adresssteile mit den oben beschriebenen Kommandos in die Speicherstellen eintragen. Mit Hilfe des Monitor-ROMs können Sie einen hexadezimalen Speicherauszug Ihres Programms erhalten, es überall im Speicher verschieben oder es auf Band schreiben und wieder einlesen. Das wichtigste Kommando im Zusammenhang mit der Maschinensprache ist aber das GO -Kommando (gehen). Wenn Sie eine Speicherstelle öffnen und G tippen, veranlaßt der Monitor den Mikroprozessor an der geöffneten Adresse dieses Programm wie ein Unterprogramm zu behandeln; am Ende der Ausführungen sollte ein RTS -Befehl (Rücksprung aus dem Unterprogramm) stehen, um die Kontrolle wieder dem Monitor zu übergeben.

Die von Ihnen erstellten Programme in Maschinensprache können viele Unterprogramme des Monitors aufrufen. Hier wird ein Programm, das die Zahlen 0 bis 9 auf dem Bildschirm ausgibt, eingegeben und gestartet.

Beispiel:

```
*0:A9 B0 20 ED FD 18 69 1 C9 BA D0 F6 60 (RETURN)
*0.9 (RETURN)
 0000: A9 B0 20 ED FD 18 69 01 C9 BA D0 F6 60 00
*0G (RETURN)
 0123456789
*
```

(Den Befehlssatz des 6502 Mikroprozessors finden Sie im Anhang dieses Handbuchs.)

Ein hexadezimaler Speicherauszug des Programmes in Maschinensprache ist nicht einfach zu lesen und die Suche nach Fehlern dadurch erschwert. Darum gibt es im Monitor-ROM ein Kommando, das Maschinenprogramme in Assemblersprache ausgibt. Das bedeutet, daß eine unformatierte Menge von Hexadezimalziffern in einzelne Befehle von 1, 2 oder 3 Byte zerlegt wird. Mit L wird das LIST-Programm des Monitor-ROMs aufgerufen.

Beispiel:

```
*0.DL (RETURN)
 0000: A9 B0      LDA    #$B0
 0002: 20 ED FD   JSR    $FDED
 0005: 18          CLC
 0006: 69 01       ADC    #$01
 0008: C9 BA       CMP    #$BA
 000A: D0 F6       BNE    $0002
 000C: 60          RTS
*
```

Das Maschinenprogramm wurde jetzt in Assemblerform ausgegeben. Vereinfacht läßt sich sagen, daß in der ersten Spalte die Befehle und in der zweiten bzw. dritten die Operanden stehen, die dann in der vierten bzw. fünften Spalte in der Assemblerform ausgegeben werden. Näheres über das Schreiben und Auswerten von Maschinenprogrammen finden Sie in den entsprechenden Handbüchern über Assembler.

### Prüfen und Ändern von Registerinhalten des 6502

Beschäftigen Sie sich intensiver mit dem Monitor ROM und dem 6502 Mikroprozessor, dann wollen Sie sicher einmal eins der internen Register des Prozessors ansehen oder es verändern. Das Monitor ROM reserviert fünf Speicherstellen für die fünf 6502-Register: A, X, Y, P (Prozessorzustand) und S (Stackpointer). Das EXAMINE-Kommando des Monitor ROM's wird durch das Fragezeichen ? ausgelöst und zeigt die Inhalte dieser Adressen an. Die Speicherstelle für das 6502-A-Register ist dann die nächste veränderbare Adresse. Wollen Sie die Werte dieser Speicherstelle ändern, so brauchen Sie nur einen Doppelpunkt und dann die Werte, durch Leerzeichen getrennt, eingeben. Beim nächsten G wird das Monitor ROM erst diese Werte in die echten Register des 6502 laden, bevor es den ersten Befehl Ihres Programms ausführen wird.

Beispiel:

```
*? (RETURN)  
A=88 X=13 Y=D8 P=00 S=B7  
*:A B (RETURN)  
*? (RETURN)  
A=0A X=0B Y=D8 P=B0 S=F8  
*
```

### Weitere Monitor-Kommandos

Sie können den Zustand der NORMAL-/INVERSE-Darstellung auf dem Bildschirm durch COUT vom Monitor aus bestimmen. Das INVERSE -Kommando des Monitor ROMs stellt durch Tippen von I auf inverse Ausgabe um, allerdings bleiben die Eingabezeilen in der Normaldarstellung.

Der NORMAL-Zustand wird dann durch das Kommando N wieder hergestellt.

Wenn Sie die von der Firma Apple Computer Inc. verfügbaren Applesoft BASIC ROMs oder Integer BASIC ROMs (siehe dazu in Kapitel 1 -Hauptplatine-) eingesetzt haben, können Sie mit Druck auf die Tasten CTRL und gleichzeitig B den Monitor verlassen, um in die BASIC-Sprache zu gelangen. Auf diesem Wege gehen Ihnen aber alle vorhandenen Programme und Variablen verloren. Sind Sie von BASIC in den Monitor gegangen und wollen Sie wieder zurück ins BASIC, ohne Programm und Variablen zu verlieren, so können Sie das mit Q.

Ein weiteres Kommando ist das PRINTER-Kommando. Mit der Eingabe von nP lenken Sie alle Ausgaben, die normalerweise auf dem Bildschirm erscheinen sollen, auf einen Drucker. n gibt an, in welcher Erweiterungsbuchsenleiste Sie die Steuerkarte für Ihren Drucker eingesetzt haben oder ob Sie eine der auf der Hauptplatine eingebauten Steuerungen für Ihren Drucker benutzen, in der Regel 1.

Beispiel:

```
*1P(RETURN)  
*
```

Das Kommando OP bringt die dann folgende Ausgabe des BASIS 108 wieder auf den Bildschirm.

Das KEYBOARD-Kommando K ersetzt die Tastatur des BASIS 108 durch ein entsprechendes anderes Eingabegerät, das über einen der Erweiterungssteckplätze angeschlossen ist, analoger Gebrauch wie beim P. Entsprechend schaltet OK wieder auf die Tastatur zurück.

**Wichtig:** Obwohl nur Erweiterungsbuchsenleisten von 2 bis 7 auf der Hauptplatine eingebaut sind, schaltet das Kommando 9P die eingebaute serielle Schnittstelle auf 'Ausgabe' und das Kommando 9K auf 'Eingabe' um.

### Kleine Hilfen für den Umgang mit dem Monitor

Sie können mehrere Kommandos in einer Eingabe zusammenfassen, solange Sie sie durch Leerzeichen voneinander trennen und nicht mehr als 253 Zeichen eingeben. Die Leerzeichen zählen mit.

Sie können außer dem STORE-Kommando, dem Doppelpunkt :, alle Kommandos in beliebiger Reihenfolge angeben.

Da der Monitor alle Werte nach dem Doppelpunkt in aufeinanderfolgende Speicherstellen ablegt, muß dem letzten Wert des STORE-Kommandos ein Buchstabenkommando folgen. Das NORMAL-Kommando N ist dafür ein gutes Trennzeichen, da es meist keine Veränderung bewirkt und überall verwendet werden kann.

Kommandos mit einem Buchstaben, wie L, R, I und N brauchen nicht mit Leerzeichen von anderen Kommandos getrennt werden.

Erreicht der Monitor bei der Bearbeitung einer Eingabezeile ein Zeichen, das er weder als Hexadezimalzahl noch als gültiges Kommandozeichen erkennen kann, führt er alle Kommandos bis zu diesem Zeichen aus. Dann meldet er über den Lautsprecher den Fehler und ignoriert den Rest der Eingabezeile.

Das MOVE-Kommando kann dazu benutzt werden, eine beliebige Folge von Werten in einen Speicherbereich zu übertragen. Dazu wird diese Folge von Werten an den Anfang des Bereichs geschrieben:

Beispiel:

```
*0:1 2 3(RETURN)  
*
```

Dabei kommt es auf die Anzahl der zu wiederholenden Werte an (in diesem Fall sind es drei).

Das MOVE-Kommando bekommt dann eine andere Einteilung:

(Anfangsadresse+Anzahl) (Anfangsadresse).(Endadresse-Anzahl)M

Dieses MOVE-Kommando kopiert die Folge von Werten hinter die Originalfolge, überträgt diese Kopie in die anschließenden Speicherzellen und wiederholt diesen Vorgang, bis der gesamte Bereich gefüllt ist.

Beispiel:

```
*3<0.0M(RETURN)
*0.10(RETURN)
 0000: 01 02 03 01 02 03 01 02 03 01 02 03 01
 0010: 00
*
```

Sie können eine Kommandozeile schreiben, die sich selbst oder einen Teil der Zeile unaufhörlich wiederholt. Dazu fängt der Teil, der sich wiederholt, mit einem Buchstabenkommando ,z.B. N , an und endet mit 34:n , wobei n die hexadezimale Position des Zeichens ist, an dem die Schleife anfängt (für das erste Zeichen ist n=0). Damit diese Schleife funktioniert, muß nach dem Wert für n ein Leerzeichen folgen.

Beispiel:

```
*N 0 2 34:0 N (RETURN)
 0000: 01
 0002: 03
 0000: 01
 0002: 03
 0000: 01
 0002: 03
 .
 .
 .
 (RESET)           c(SHIFT,SHIFT,CRTL)
*
```

Eine derartige Schleife läßt sich nur durch (RESET) stoppen.

### Erzeugen eigener Kommandos

Das USER-Kommando wird durch ein U eingegeben und läßt den Monitor zur Adresse \$3F8 springen. In diese Adresse können Sie einen JMP-Befehl einsetzen, der zu dem von Ihnen erstellten Programm oder der gewünschten Adresse springt. Ihr Programm kann so z. B. die Register, die Spezialadressen des Monitors oder die Eingabezeile prüfen. Beispielsweise kann durch U der Lautsprecher angesprochen werden, wenn in \$3F8 das Kommando \$FF3A steht.

Beispiel:

```
* 3F 8 (RETURN)
  03F8: 4C
* 3F 8: 4C 3A FF (RETURN)
* 3F 8L (RETURN)
  03F8: 4C 3A FF      JMP    $FF3A
*U (RETURN)
*
c (der lautsprecher erklingt).
```

Eventuell werden auch einige Speicher ausgedruckt.

## Übersicht über die Monitor-Kommandos

### Speicherstellen ansehen

(Adresse)

Gibt den Inhalt einer Speicherstelle aus.

(Anfang).(Ende)

Gibt alle Inhalte zwischen (Anfang) und (Ende) aus.

(RETURN)

Zeigt die Werte von max. 16 Speicherstellen nach der zuletzt geöffneten Adresse an.

### Speicherinhalte verändern

(Adresse):(Wert)

Speichert (Wert) unter (Adresse) ab.

:(Wert) (Wert)...

Speichert ab der nächsten veränderbaren Adresse die Werte in aufeinanderfolgende Speicherstellen.

### Übertragen und Vergleichen

(Ziel) (Anfang).(Ende)M

Kopiert die Werte des Bereichs (Anfang).(Ende) in den Bereich (Ziel) ab.

(Ziel) (Anfang).(Ende)V

Vergleicht die Werte des Bereichs (Anfang).(Ende) mit dem Bereich (Ziel).

Schreiben und Lesen auf Band (nur bei Arbeiten mit 40 Zeichen/  
Zeile, siehe auch Anhang G)  
**(Anfang).(Ende)W**

Schreibt die Werte des Bereichs nach einer 10 s-Vorinformation auf Band.

**(Anfang).(Ende)R**

Liest Werte vom Band in den Speicherbereich (Anfang).(Ende). Druckt ERR im Fehlerfall.

Starten und Ausdrucken von Programmen

**(Adresse)G**

Läßt den Mikroprozessor 6502 ab (Adresse) das Maschinenprogramm ausführen.

**(Anfang).(Ende)L**

Läßt ab Anfangsadresse bis Endadresse das Maschinenprogramm in Assemblersprache ausgeben. .(Ende) L läßt weitere Befehle ausgeben.

Verschiedenes

**?**

Zeigt die Inhalte der 6502-Register an.

**I**

Setzt INVERSE-Modus.

**N**

Setzt NORMAL-Modus.

**CTRL-B**

Startet die Sprache, die im ROM des BASIS 108 verfügbar ist.

**Q**

Setzt die Sprache fort, die im ROM des BASIS 108 verfügbar ist. Genauer gesagt, das Programm springt auf die Adresse, die in den Speicherstellen (3F2,3F3) angegeben ist.

**nP**

Bestimmt die Ausgabe zu dem Gerät, dessen Steuerkarte in dem durch n angegebenen Erweiterungssteckplatz sitzt. n=0: dann kommt die Ausgabe auf den Bildschirm zurück. n=1: parallele Schnittstelle, Nummer=9: serielle Schnittstelle).

**nK**

Nimmt die Eingabe von dem Gerät an, dessen Steuerkarte in dem durch n angegebene Steckplatz sitzt. n=0: dann wird die Eingabe von der Tastatur erwartet. n=9: serielle Schnittstelle.

**U**

Springt zu dem Maschinenprogramm ab Adresse \$3F8.

## Liste ausgewählter Monitor-Unterprogramme

Diese Liste enthält einige nützliche Unterprogramme im Monitor-ROM des BASIS 108. Vor dem Aufruf der Unterprogramme laden Sie die nötigen Speicheradressen oder 6502-Registerinhalte. Der Aufruf erfolgt durch einen JSR-Befehl (Sprung ins Unterprogramm) zu der angegebenen Startadresse des Unterprogramms. Es wird die beschriebene Funktion ausführen und die Register so hinterlassen, wie es jeweils angegeben ist. Der Prozessorstatus (C, Z, N, V) wird im allgemeinen geändert.

- \$FDED** COUT Ausgabe eines Zeichens (Character OUTput).  
COUT ist das Standard-Unterprogramm für Zeichenausgabe. Das Zeichen, das ausgegeben werden soll, steht im Akkumulator. COUT ruft das aktuelle Unterprogramm zur Zeichenausgabe auf, dessen Adresse in CSW (Adressen \$36 und \$37) steht.
- \$FDF0** COUT1 Ausgabe auf den Bildschirm.  
COUT 1 bringt das Zeichen im Akkumulator auf den Bildschirm des BASIS 108. Es wird auf die Ausgabeposition gesetzt und bewegt dann diese Position weiter. Das Zeichen wird mit dem Inhalt der NORMAL-/INVERSE-Speicherstelle modifiziert. Die Steuerzeichen RETURN, Zeilenvorschub und Klingelzeichen werden von COUT 1 ebenfalls behandelt. Das Unterprogramm lässt alle Register intakt.
- \$FE80** SETINV Setzt den INVERSE-Modus (SET INVerse).  
Der INVERSE-Modus für COUT 1 wird gesetzt. Dadurch erscheinen alle Zeichen als schwarze Punkte auf weißem Hintergrund, die dann von COUT 1 ausgegeben werden. Das Y-Register wird auf \$7F gesetzt, alle anderen Register bleiben unverändert.
- \$FE84** SETNORM Setzt den NORMAL-Modus (SET NORMAL).  
Setzt den NORMAL-Modus für COUT 1. So werden alle Zeichen als weiße Punkte auf schwarzem Hintergrund ausgegeben. Das Y-Register erhält den Wert \$FF, alle anderen Register bleiben unverändert.
- \$FD8E** CROUT Gibt ein RETURN aus (Carriage Return OUTput).  
CROUT sendet ein RETURN zu dem aktuellen Ausgabegerät.
- \$FDDA** PRBYTE Druckt ein Byte als Hexadezimalzahl.  
Dieses Unterprogramm gibt den Inhalt des Akkumulators als Hexadezimalzahl auf das aktuelle Ausgabegerät. Der Inhalt des Akkumulators wird verändert.
- \$FDE3** PRHEX Druckt eine Hexadezimalziffer (PRInt HEXadecimal digit).  
Dieses Unterprogramm gibt die unteren vier Bits (Bit 3 bis Bit 0) des Akkumulators als eine Hexadezimalziffer aus. Der Inhalt des Akkumulators wird verändert.

- \$F941** PRNTAX Druckt A und X als eine Hexadezimalzahl  
 (PRINT A und X in hexadecimal).  
 Dieses Unterprogramm gibt die Inhalte des Akkumulators und des X-Registers als vierziffrige Hexadezimalzahl aus. Der Akkumulator enthält die linken zwei Ziffern, das X-Register bestimmt die rechten zwei Ziffern. Der Inhalt des Akkumulators wird verändert.
- \$F948** PRBLNK Druckt drei Leerzeichen (PRint 3 BlaNK spaces).  
 Gibt drei Leerzeichen über das Standard-Ausgabegerät aus. Der Akkumulator bekommt den Wert \$AO und das X-Register den Wert 0.
- \$F94A** PRBL2 Druckt viele Leerzeichen.  
 Dieses Unterprogramm gibt 1 bis 256 Leerzeichen zur Standardausgabe. Beim Aufruf bestimmt der Inhalt des X-Registers die Anzahl der Leerzeichen. Ist X=0, so werden 256 Leerzeichen ausgegeben. Beim Ausgang hat der Akkumulator den Inhalt \$A0 und das X-Register den Inhalt 0.
- \$FF3A** BELL Ausgabe eines Klingel-Zeichens (BELL).  
 Dieses Unterprogramm sendet ein Klingel-Zeichen (CTRL G) zu dem aktuellen Ausgabegerät. Der Akkumulator bekommt den Wert \$87.
- \$FBDD** BELL1 Abgabe eines Tonsignals aus dem Lautsprecher des BASIS 108.  
 Dieses Unterprogramm erzeugt ein kurzes 2-Ton Signal. Die Inhalte des Akkumulators und des Y-Registers werden verändert.
- \$FD0C** RDKEY Eingabe eines einzelnen Zeichens.  
 Dies ist das Unterprogramm für Standard-Zeicheneingabe. Ein blinkender Eingabezeiger erscheint auf dem Bildschirm an der Position des Ausgabezeigers und das Unterprogramm springt zu dem aktuellen Eingabe-Unterprogramm, dessen Adresse in KSW (Adressen \$38 und \$39).
- \$FD35** RDCHAR Eingabe eines einzelnen Zeichens oder einer Steuer-Anweisung.  
 RDCHAR ist ein weiteres Eingabe-Unterprogramm, das Zeichen von der Standardeingabe erhält, aber auch die Tasten des Cursorblockes bis auf die beiden Tasten links und rechts unten.
- \$FD1B** KEYIN Lesen eines Zeichens von der Tastatur.  
 Dies ist das Unterprogramm für die Eingabe über die Tastatur. Nach Abfrage wartet der BASIS 108 auf einen Tastendruck, eine Zufallszahl wird gebildet. Erfolgt ein Tastendruck, so wird der blinkende Zeiger entfernt und der Tastencode in den Akkumulator gegeben. Falls Zusatztaste oder Zeigertaste gedrückt wurde, so ist im Akkumulator Bit 7=0, sonst 1.

- \$FD6A** GETLN Anforderung einer Eingabezeile mit Bereitschaftszeichen.  
 Das Unterprogramm GETLN sammelt aus einzelnen Zeichen eine Eingabezeile. Ihre Programme können das Bereitschaftszeichen für GETLN in der Speicherzelle \$33 bestimmen. Das Unterprogramm GETLN kehrt mit der Eingabezeile im Eingabepuffer (ab Adresse \$200) und mit der Länge der Eingabezeile im X-Register zurück. Die Tasten des Cursorblockes werden ausgeführt, die Zusatztasten dagegen nicht.
- \$FD67** GETLNZ Anforderung einer Eingabezeile.  
 Das Unterprogramm GETLNZ schickt erst einen Zeilenvorschub zum Standardausgabegerät, bevor GETLN ausgeführt wird (s. oben).
- \$FD6F** Anforderung einer Eingabezeile ohne Bereitschaftszeichen.  
 Dieser Einsprung beginnt in GETLN erst an der Stelle, an der die Eingabezeile gebildet wird, so daß kein Bereitschaftszeichen erscheint. Löschen Sie jedoch mehr Zeichen als in der Eingabezeile vorhanden waren oder betätigen Sie CE, so wird der Inhalt der Speicherzelle \$33 als Bereitschaftszeichen einer neuen Eingabezeile ausgegeben.
- \$FCA8** WAIT Warten.  
 Dieses Unterprogramm wartet eine bestimmte Zeit und kehrt dann wieder zu dem Programm zurück, das es aufgerufen hat. Der Akkumulator bestimmt diese Zeit. Wenn A der Inhalt des Akkumulators ist, ergibt sich eine Verzögerung von  $(13 + 12A + 5A^2)$  Zyklen. Das ist ca. 1 Mikrosekunde. Bei A = 0 zählt es wie 256. WAIT läßt X und Y unverändert, nur das A-Register wird 0.
- \$F864** SETCOL Setzt die Farbe für die Ausgabe von Lo-Res Graphik (SET COLOR).  
 Der Akkumulator bestimmt die Farbe, die bei der Lo-Res Graphik-Ausgabe auf den Bildschirm verwendet werden soll. Der Akkumulator wird verändert, sonst ändern sich die Register nicht.
- \$F85F** NEXTCOL Die Farbnummer wird um 3 erhöht (NEXT COLOR).  
 Die aktuelle Farbe für die Ausgabe von Lo-Res Graphik wird um 3 erhöht. Nur das A-Register wird verändert.
- \$F800** PLOT Überträgt einen Block auf den Lo-Res Bildschirm.  
 Dieses Unterprogramm druckt einen einzelnen Block in der vorher eingestellten Farbe auf den Bildschirm, beim 80 Spalten Monitor-ROM bis zu 79 Zeichen. Die vertikale Position wird im Akkumulator übergeben und die horizontale Position wird dem Y-Register entnommen. PLOT verändert nur den Akkumulator.
- \$F819** HLINE Zeichnet eine waagrechte Linie von Blöcken.  
 Es wird eine Zeile von Blöcken in der vorher festgelegten Farbe auf den Lo-Res-Bildschirm gezeichnet (s. auch PLOT). Folgende Angaben müssen beim Aufruf vorhanden sein: Die senkrechte Koordinate steht im Akkumulator, die waagrechte Koordinate des linken Endes im Y-Register, die des rechten Endes in \$2C. HLINE verändert A und Y, läßt aber X intakt.

- \$F828** VLINE Zeichnet eine senkrechte Linie von Blöcken.  
 Dieses Unterprogramm zeichnet eine senkrechte Linie von Blöcken der vorher festgelegten Farbe auf den Lo-Res-Bildschirm. Folgende Werte müssen beim Aufruf vorliegen:  
 Die oberste vertikale Position im Akkumulator, die unterste vertikale Koordinate in \$2D und die horizontale Koordinate der Linie im Y-Register. VLINE verändert den Akkumulator.
- \$F832** CLRSCR Löscht den gesamten Lo-Res Bildschirm.  
 CLRSCR löscht den gesamten Bildschirm der Blockgraphik. Wird CLRSCR im TEXT-Modus aufgerufen, so wird der Bildschirm mit inversen §-Zeichen gefüllt. CLRSCR verändert die Inhalte von A und X.
- \$F836** CLRTOP Löscht den oberen Teil der Lo-Res Graphik.  
 CLRTOP arbeitet wie CLRSCR (s. oben), aber es werden nur die oberen 40 Reihen des Bildschirms gelöscht.
- \$F871** SCRN Liest ein Zeichen auf dem Lo-Res Bildschirm.  
 Dieses Unterprogramm kehrt mit der Farbe eines bestimmten Blocks auf dem Bildschirm in das Programm zurück, das SCRN aufgerufen hat. Den Anruf gestalten Sie wie bei PLOT (s. oben). Die Nummer der Farbe des Blocks steht nach dem Aufruf im Akkumulator. Andere Register werden nicht verändert.
- \$FB1E** PREAD Liest die Stellung einer Spielsteuerung.  
 PREAD braucht zum Aufruf die Nummer der Spielsteuerung im X-Register. Diese Zahl muß 0, 1, 2 oder 3 sein, sonst werden Sie sich wundern. Die Stellung der Spielsteuerung wird als Zahl zwischen \$00 und \$FF im Y-Register übergeben. Der Akkumulator wird verändert.
- \$FF4A** SAVE Rettet alle Register.  
 Die Inhalte aller internen Register des 6502-Mikroprozessors werden in der Reihenfolge A-X-Y-P-S in die Speicherstellen \$45 bis \$49 geschrieben. Die Inhalte von A und X werden verändert und der Dezimalmodus des Mikroprozessors wird gelöscht.
- \$FF3F** RESTORE Register werden wiederhergestellt.  
 Die Inhalte der internen Register des 6502-Mikroprozessors werden von den Speicherstellen \$45 bis \$48 geladen. S (stack) Register wird nicht geändert, damit Restore zurückkehren kann.

## SPEZIALADRESSEN DES MONITORS

Adresse Dezimal	Verwendung im BASIS 108 Monitor Hexa	
1008	\$3F0	Enthält die Adresse des Unterprogramms, das "BRK"-Befehle behandelt
1009	\$3F1	(normal: \$FA59).
1010	\$3F2	Warmstart in die benutzte Sprache.
1011	\$3F3	Monitor "Q" springt auf die Adresse.
1012	\$3F3	Einschalt-Byte
1013	\$3F5	Enthält einen JMP (Sprung)-Befehl zu
1014	\$3F6	dem Unterprogramm, das FPBASIC -Kom-
1015	\$3F7	mando behandelt . (Normal: \$4C \$58 \$FF)
1016	\$3F8	Enthält einen JMP-Befehl zu dem Unter-
1017	\$3F9	programm, das "USER" (U)-Kommandos be-
1018	\$3FA	handelt.
1019	\$3FB	Enthält einen JMP-Befehl zu dem Unter-
1020	\$3FC	programm, das nichtmaskierbare Inter-
1021	\$3FD	rupts behandelt.
1022	\$3FE	Enthält die Adresse des Unterprogramms,
1023	\$3FF	das maskierbare Interrupts (IRQ) behan- det.
1273	\$4F9	Wenn 0, dann 40 Zeichen, wenn ≠ 0, dann 80 Zeichen.

## KAPITEL 5

### INHALTSVERZEICHNIS

#### Der Speicher

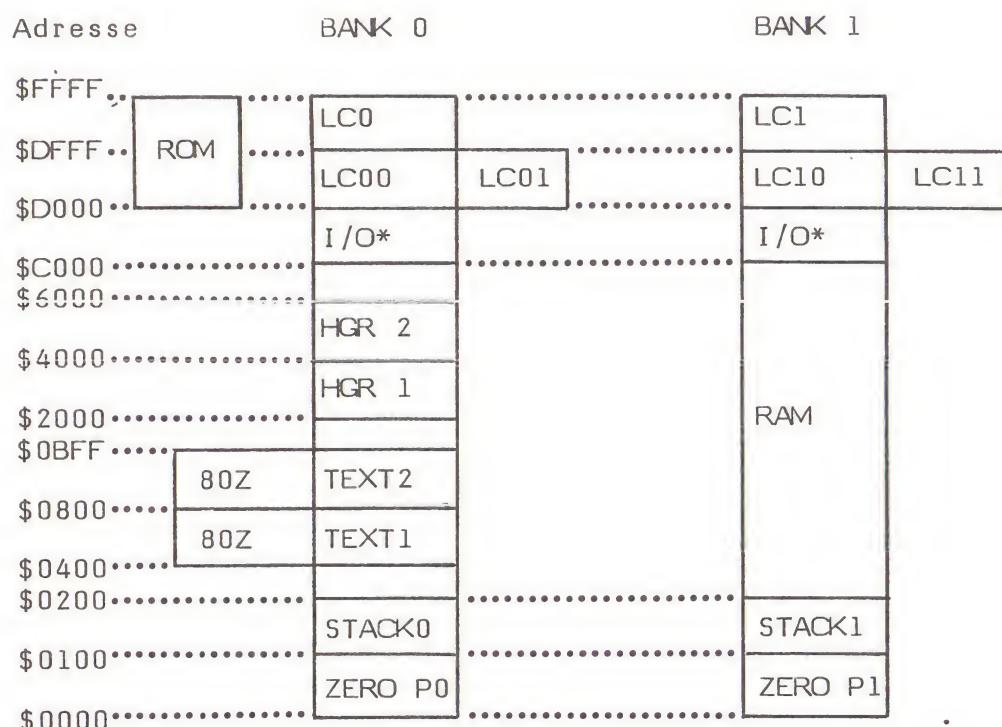
- 58 Speicherorganisation
- 58 Aufteilung des Adreßraumes
- 59 BANK 0/BANK 1 Umschaltung
- 60 ROM und RAM Umschaltung
- 61 Das Statik-RAM für die 80 Z-Darstellung

## Speicherorganisation

Das BASIS 108 Computersystem kann mit einem RAM-Speicher bis zu 128 kByte ausgerüstet werden. Der 6502 Mikroprozessor (wie auch der Z-80 Mikroprozessor) kann allerdings mit seinen 16 Adressleitungen nur einen Speicherraum von 64 kByte verwalten. Zusätzlich zu dem RAM-Speicher ist ein ROM-Bereich von 12kByte und der Ein-/Ausgabebereich, der einen Adressraum von 4 kByte belegt, zu adressieren. Da sich somit ein Adressraum von 144 kByte ergibt, den es zu adressieren gilt, wurde die Möglichkeit geschaffen, nur bestimmte Teile des ROM- und RAM-Bereiches zur gleichen Zeit zu aktivieren.

Um dies zu erreichen, wurde der RAM-Bereich zunächst in 2 Seiten, Banks genannt, von je 64 kByte Größe eingeteilt, dann jeder Bereich nochmals in 8 kByte Blöcke. Dadurch besteht die Möglichkeit, zwischen den Banks in Schritten von 8 kByte umzuschalten. Der nächste Schritt war nun, den ROM-Bereich in den Adressraum zu integrieren. Da der 6502 Mikroprozessor nach einem Reset die Adresse \$FFFC ausgibt und auf dieser eine ausführbare Operation ständig gespeichert sein muß, ist der ROM-Bereich am Ende des Adressraumes angesiedelt, dem sich direkt der Ein-/Ausgabebereich anschließt.

### Aufteilung des Adreßraumes



\* I/O-Ein-/Ausgabe

Damit haben wir die oberen 16 kByte des Adressraumes einmal mit ROM und Ein-/Ausgabebereich belegt und zum anderen existiert auch noch der RAM-Speicher für diesen Bereich. Dieser 16 kByte große Speicher wird noch einmal in 4 kByte Blöcke aufgeteilt. Da der 4 kByte Ein-/Ausgabebereich dem Prozessor ständig zur

Verfügung stehen muß, wird der für diesen Adressraum vorgesehene RAM-Speicher dem nächsten 4 kByte Block parallel geschaltet. Die Wahl, welcher dieser beiden Blöcke nun aktiv sein soll, kann dann über einen Software-Schalter getroffen werden (s. unten). Da dieser RAM-Speicher parallel zum ROM-Speicher liegt und nur ein Bereich aktiv sein darf, wird auch hier der aktive Bereich durch einen Softwareschalter ausgewählt.

Um diesen RAM-Bereich für besondere Aufgaben einsetzen zu können (z. B. Speicherung eines Basic Interpreters o. ä.) ist es möglich, diesen Bereich vor unbeabsichtigtem Schreiben zu schützen. Auch ist eine Kombination von ROM-Lesen und RAM-Schreiben möglich.

All diese oben genannten Möglichkeiten werden über Softwareschalter erreicht und gelten sowohl für die BANK 0 als auch für die BANK 1.

Im RAM-Bereich der BANK 0 sind außerdem die verschiedenen Bereiche der Bildwiederholungsspeicher angesiedelt. Eine Darstellung der Bildwiederholungsspeicher in der BANK 1 ist nicht möglich, da bei einem Speicherzugriff der Bildwiederholungslogik immer BANK 0 durch die Hardware verwendet wird.

Den beiden Textseiten des Bildwiederholungsspeichers ist ein 2kByte statisches RAM parallel geschaltet, um die 80 Zeichen pro Zeile Darstellung zu ermöglichen. Wenn nun in den Bildwiederholungsspeicher Nr. 0 Zeichen geschrieben werden sollen, wird je nach Position dieses Zeichens, entweder der RAM-Bereich des normalen RAM's oder das statische RAM aktiviert.

### BANK 0/BANK 1 - Umschaltung

Die nachfolgenden Adressen schalten zwischen BANK 0 und BANK 1 um.

Die Umschaltung erfolgt aber nur, wenn ein Schreibbefehl auf diese Adresse ausgeführt wird. Ein Lesebefehl dieser Adressen liest den Zustand der entsprechenden TTL- und Analogeingänge.

Nach dem Einschalten des BASIS 108 Computersystems oder einem RESET ist grundssätzlich die BANK 0 aktiv.

Bank 0 aktiv	Bank 1 aktiv	Adressraum
\$C060w	\$C061w	\$0000 - \$1FFF
\$C062w	\$C063w	\$2000 - \$3FFF
\$C064w	\$C065w	\$4000 - \$5FFF
\$C066w	\$C067w	\$6000 - \$7FFF
\$C068w	\$C069w	\$8000 - \$9FFF
\$C06Aw	\$C06Bw	\$A000 - \$BFFF
\$C06Cw	\$C06Dw	\$D000 - \$DFFF
\$C06Ew	\$C06Fw	\$E000 - \$FFFF

Der Schalter \$C06C/\$C06D schaltet nur den 4 kByte Adressraum von \$D000 bis \$DFFF, der Adressraum \$C000 bis \$CFFF ist der Ein-/Ausgabebereich und kann daher nicht geschaltet werden.

### ROM und RAM Umschaltung

Die nachfolgend beschriebenen Schalter erlauben die Umschaltung zwischen ROM und RAM der jeweils aktivierte BANK im Adressbereich \$E000-\$FFFF, sowie das Umschalten des mit RAM-Speicher doppelt belegten Adressbereichs \$D000 bis \$DFFF und das Schützen dieser Bereiche vor versehentlichem Beschreiben. Die Schaltergruppe \$C080 bis \$C083 bezieht sich auf den Block LCx0 und die Gruppe \$C088 bis \$C08B auf die Blöcke LCx1, wobei x durch die jeweils aktivierte Bank dargestellt wird, (Bank 0 x=0; Bank 1 x=1).

Die nachfolgenden Schalteradressen sollen nur durch Leseoperationen angesteuert werden.

RAM-Auswahl		RAM/ROM-Auswahl
\$D000 - \$DFFF	Seite 0/Seite 1	
\$C080    \$C088		RAM ist schreibgeschützt, Lesen erlaubt, ROM ist abgeschaltet.
\$C081    \$C089		ROM Lesen erlaubt, RAM schreibgeschützt. Wird der Befehl zwei oder mehrmal gegeben, ist es möglich im RAM zu schreiben.
\$C082    \$C08A		RAM schreibgeschützt, es wird aus ROM gelesen.
\$C083    \$C08B		erlaubt den RAM zu lesen, schreibgeschützt. Wird der Befehl zwei- oder mehrmal gegeben, so kann auch geschrieben werden.

Einige Erklärungen zu den Schaltern:

\$C080/\$C088 Der RAM-Bereich wird nur für Leseoperationen aktiviert und der ROM-Bereich abgeschaltet.

\$C081/\$C089 Der ROM-Bereich wird für Leseoperationen aktiviert und der RAM-Bereich hierfür abgeschaltet. Bei zwei- oder mehrmaligem Ansprechen wird der RAM-Bereich für Schreiboperationen aktiv, so daß zum Beispiel das Kopieren der ROMs in den RAM-Bereich möglich ist.

- \$C082/\$C08A Schaltet das RAM Lesen ab und aktiviert den ROM-Bereich. Der RAM-Bereich bleibt aber schreibgeschützt.
- \$C083/\$C08B Der RAM-Bereich wird für Leseoperationen aktiviert. Bei zwei- oder mehrmaligem Ansprechen wird der RAM auch schreibfähig. Das bedeutet, daß dieser Bereich nun ein normales RAM-Memory darstellt.

### Das Statik-RAM für die 80 Z-Darstellung

Dieses 2K Statik-RAM ist dem Adressbereich \$0400-\$0BFF parallel geschaltet. Dies ermöglicht 2 Seiten Bildschirmwiederholungsspeicher mit je 80 Zeichen pro Zeile bei 24 Zeilen. Da auch dieser Bereich parallel zum normalen RAM-Bereich liegt, wird über einen Softwareschalter der jeweils aktive Bereich ausgewählt.  
\$C00Dw Zusatz RAM eingeschaltet, Normal RAM abgeschaltet  
\$C00Cw Zusatz RAM abgeschaltet, Normal RAM eingeschaltet.

Diese Softwareschalter sind nur mit einem Schreibbefehl zu betätigen.

## **Kapitel 6**

### **INHALTSVERZEICHNIS**

#### **Ein-/Ausgabe**

- 63 Eingebaute Ein-/Ausgabemöglichkeiten
- 63 Dateneingänge, Status Eingänge, Strobe
- 64 Kippschalter, Drucker Interface, serielles RS 232c  
Interface
- 65 Kontrollregister
- 66 Kommandoregister
- 67 Statusregister
- 68 Kassettenrekorder Interface
- 68 Handregleranschluß und TTL Ein- und Ausgänge
- 68 Lautsprecher
- 68 Erweiterungs-ROM

## **Eingebaute Ein-/Ausgabemöglichkeiten**

Auf der Hauptplatine des BASIS 108 sind folgende Ein- und Ausgabemöglichkeiten integriert:

- Paralleles Drucker Interface (Centronics kompatibel),
- Serielles RS 232c Interface,
- Kassettenrekorder Interface,
- Anschluß für 4 Handregler,
- 3 Eingänge für TTL-Signale,
- 4 TTL-Ausgänge,
- Lautsprecherausgang,
- Tastatur,
- Video.

Man kann diese Ein- und Ausgabemöglichkeiten in mehrere Gruppen einteilen; Dateneingänge, Strobes, Softwareschalter, Kippschalter und Statuseingänge.

### **Dateneingänge**

Als Dateneingänge des BASIS 108 Systems kann neben der parallelen und seriellen Schnittstelle auch der Tastatureingang gewertet werden. Das höchstwertige Bit dieses Einganges ist ein Statusbit und die niederwertigen 7 Bits der entsprechenden ASCII-Code der gedrückten Taste. Ist das höchstwertige Bit 1, wurde auf der Tastatur eine Taste gedrückt.

### **Status Eingänge**

Diese Eingänge können nur die Zustände EIN oder AUS annehmen. Angezeigt wird dieses im höchstwertigsten Bit der angesprochenen Adresse. Das Erkennen des entsprechenden Zustandes kann von einer höheren Programmiersprache durch Testen des gelesenen Bytes, ob größer oder gleich 128 für EIN und kleiner als 128 für AUS durchgeführt werden. Solche Eingänge sind die 3 TTL-Eingänge, der Kassettenrekorder Eingang und die Handreglereingänge.

### **Strobe**

Signale dieses Typs werden ebenfalls über Speicheradressen erzeugt und dienen zum definierten Setzen oder Rücksetzen einiger Statuseingänge. Im BASIS 108 Computersystem existieren 3 Strobe Signale.

1. Tastatur Strobe (\$C010), dieses Strobe Signal setzt das höchstwertigste Bit des Tastatureinganges (\$C000) auf NULL zurück.

2. Der Handregler Strobe (\$C070) setzt alle vier Mono-Flops der Handreglereingänge zurück und startet die Zeitschleife neu.
3. Der Utilitie Strobe (\$C040) ist auf Pin 5 des Handregleranschlusses zu finden. Wenn diese Adresse angesprochen wird, geht diese Leitung für 0.4 Mikrosekunden von TTL-high auf TTL-low. Wenn mit einem Schreibbefehl der Form absolut-indiziert oder indirekt-indiziert diese Adresse angesprochen wird, werden 2 Pulse erzeugt. Wenn der 6502 Mikroprozessor einen Schreibbefehl ausführt, liest er zuerst die angesprochene Adresse, bevor sie überschrieben wird. Dadurch erfolgen bei einem Schreibbefehl zwei Zugriffe zu der entsprechenden Adresse.

### Kippschalter

Der Lautsprecher, wie auch der Kassettenrekorder-Ausgang werden über einen Kippschalter angesprochen.

Ein Lesen der entsprechenden Adresse veranlaßt ein Flip-Flop in den anderen Zustand zu fallen. Das bedeutet; der Ausgang des Flip-Flops geht von logisch 0 auf logisch 1 und bleibt solange in diesem Zustand, bis das Flip-Flop erneut angesprochen wird.

### Drucker Interface

Das parallele Drucker Interface generiert alle notwendigen Signale zur Steuerung eines Druckers mit Centronics kompatibler Schnittstelle. Die Ausgabedaten werden in die Ausgabeadresse \$C090-C097 geschrieben, wodurch automatisch die Generierung eines Strobe Signals ausgelöst wird. Im höchstwertigen Bit der Adresse \$C1C1 kann die Übernahmebestätigung (Acknowledge) des Druckers abgefragt werden. Eine Standard Treiber Routine ist in einem 256x8 ROM auf der Adresse \$C100 abgelegt.

### Serielles RS 232c Interface

Das serielle Interface besteht aus dem Baustein 6551 mit nachgeschalteten Leitungsempfängern und Treibern. Dieser Baustein hat 2 Handshakeleitungen. Das Datenregister dieses Bausteins ist auf der Adresse \$C098, das Statusregister auf \$C099, das Command Register auf \$C09A und das Mode Register auf der Adresse \$C09B. Die Übertragungsgeschwindigkeit kann zwischen 50 und 19200 Baud gewählt werden. Eine Standard Treiber Routine befindet sich ebenfalls in dem ROM auf der Adresse \$C108. Diese Treiber Routine initialisiert das serielle Port auf folgende Werte:

9600 Baud, Wortlänge 8 Bit und 2 Stopbit, keine Parität.

Wollen Sie die V24 Treibersoftware oder andere Parameter benutzen, schlagen Sie bitte im Anhang E nach. Auf den nachfolgenden Seiten finden Sie hierfür die wichtigsten Parameter dieses Bausteines.

Adressen	Schreiben	Lesen
\$C098	Transmit Data Register	Receiver Data Register
\$C099	Programm Reset *	Statusregister
\$C09A		Comm. Register
\$C09B		Contr. Register .

\* Ein Schreiben auf die Adresse des Statusregisters bewirkt ein Setzen des ACIA in einen bestimmten Status. Hiervon werden alle Register betroffen (für weitere Informationen s. Datenblatt im Anhang).

### Kontrollregister

Mit dem Kontrollregister wird die Wortlänge, die Anzahl der Stopbits und die Übertragungsrate festgelegt.

Bit 7	STOP BITS 0 = 1 Stopbit 1 = ? Stopbits 1 Stopbit, wenn die Wortlänge 8 und Parität gesetzt ist. 1,5 Stopbits, wenn die Wortlänge 5 und keine Parität gesetzt ist.
-------	---

Bit 6 u. 5	Wortlänge 0 0 8 Bit 0 1 7 Bit 1 0 6 Bit 1 1 5 Bit
------------	---

Bit 4	Empfänger Takt Frequenz 1 = Interner Baud Rate Generator ! muß immer 1 sein !
-------	---

Baud Rate Generator  
 -mit diesen Bits wird die Baud Rate ausgewählt-

Bit	3 2 1 0	Baud Rate
	0 0 0 0	illegal
	0 0 0 1	50 Baud
	0 0 1 0	75
	0 0 1 1	110
	0 1 0 0	134,5
	0 1 0 1	150
	0 1 1 0	300
	0 1 1 1	600
	1 0 0 0	1200
	1 0 0 1	1800
	1 0 1 0	2400
	1 0 1 1	3600
	1 1 0 0	4800
	1 1 0 1	7200
	1 1 1 0	9600
	1 1 1 1	19200

## Kommandoregister

Das Kommandoregister steuert spezielle Sende- und Empfangsfunktionen.

### Überprüfung der Paritäten

Bit	7 6 5	
	x x 0	keine Parität bei Sendung und Empfang
	0 0 1	ungerade Sender und Empfänger
	0 1 1	gerade Sender und Empfänger
	1 0 1	Sendet 1 statt Parität Parität Test abgeschaltet
	1 1 1	Sendet 0 statt Parität Parität Test abgeschaltet.

Bit 4 Normal/Echo Mode Empfänger

0=Normal  
 1=Echo .

### Transmitter Kontrolle

Bit	3 2	Transmitter Unterbrechung	RTS Pegel
	0 0	abgestellt	inaktiv
	0 1	eingeschaltet	aktiv
	1 0	abgestellt	aktiv
	1 1	abgestellt	aktiv, es wird BREAK gesendet.

Bit 1 Empfangsunterbrechung  
0 = eingeschaltet  
1 = ausgeschaltet.

Bit 0 Data Terminal Ready (DTR)  
0 = Empfang aus / Baustein (DTR inaktiv)  
1 = Empfang an / Baustein (DTR aktiv) .

### Statusregister

Im Statusregister wird der aktuelle Zustand des Bausteins angezeigt.

Bit 7 Interrupt (IRQ)  
0 = kein Interrupt  
1 = Interrupt ist aufgetreten

Bit 6 Data Set Ready (DSR)  
0 = DSR bereit  
1 = DSR nicht bereit

Bit 5 Data Carrier Detect (DCD)  
0 = DCD erkannt  
1 = DCD nicht erkannt

Bit 4 Datensenderegister  
0 = nicht leer  
1 = leer

Bit 3 Datenempfangsregister  
0 = nicht voll  
1 = voll

Bit 2 Überlauf  
0 = kein Fehler  
1 = Fehler, Datenverlust, da nicht schnell genug gelesen.

Bit 1	Taktfehler 0 = kein Fehler 1 = Fehler, wahrscheinlich falsche Baudrate
Bit 0	Paritätsfehler 0 = kein Fehler 1 = Fehler wurde erkannt

### Kassettenrekorder Interface

Das Einlesen einer Information vom Kassettenrekorder geschieht auf der Adresse \$C060, die Ausgabe auf \$C02x. Eine entsprechende Treiberroutine ist im speziellen Monitor-ROM für 40 Zeichen/Zeile untergebracht. Dieses Monitor-ROM muß gesondert erworben werden, s. Anhang G.

### Handregleranschluß und TTL Ein- und Ausgänge

Der Handregleranschluß und die TTL Ein- und Ausgänge sind gemeinsam auf einem 16-poligen DIL-Sockel verfügbar.

Über den Regelwiderstand des Handreglers wird die Rücksetzzeit eines monostabilen Flip-Flops gesteuert. Das Setzen oder Starten aller 4 Flip-Flops wird über die Adresse \$C07x gesteuert, die Abfrage des Status der einzelnen Flip-Flops auf den Adressen \$C064 bis \$C067.

Die 4 TTL-Ausgänge sind auf den Adressen \$C058 bis \$C05F und die 3 TTL-Eingänge auf den Adressen \$C061 bis \$C063.

Auf dem DIL-Sockel befindet sich noch ein weiteres Signal, welches über die Adresse \$C04x angesprochen wird und dem Benutzer zur freien Verfügung steht.

### Lautsprecher

Durch Ansprechen der Adresse \$C03x wird ein Flip-Flop geschaltet und der Lautsprecher erzeugt ein einmaliges Klick-Geräusch. Durch ein entsprechendes Programm lassen sich Töne verschiedenster Frequenzen und Dauer produzieren.

### Erweiterungs-ROM

Das BASIS 108 Computersystem besitzt 6 Erweiterungssteckplätze für Interfacekarten oder andere Erweiterungskarten. Um diese Steckplätze vorteilhaft ausnutzen zu können, sind jedem Steckplatz 2 direkte Adressbereiche und allen gemeinsam zusätzlich noch ein 2 KByte großer Adressraum zugeordnet. Im einzelnen gleichen sich diese Bereiche wie folgt:

1. Peripheriekarten I/O Adressen.

Dies sind 16 Adressen für jeden Steckplatz. Die Signalleitung DEVICE SELECT (PIN 41 jedes Steckplatzes) signalisiert, daß der Prozessor eine Adresse innerhalb dieses Bereiches anspricht. Diese Adressen sollten bevorzugt für Ein-/Ausgabe Operationen verwendet werden.

Peripheriekarte I/O Zuweisung  
x = \$0 . . . \$F

\$C0Ax	2	
\$C0Bx	3	
\$C0Cx	Ein/Ausgabe für Steck-	4
\$C0Dx	platznummer	5
\$C0Ex		6
\$C0Fx		7

2. Peripheriekarten ROM Adressraum.

Ein weiterer Adressraum von 256 Byte ist jedem Steckplatz für die Aufnahme von Treiberroutinen oder ähnlichem direkt zugeordnet.

Die I/O SELECT Leitung (Pin 1 jedes Steckplatzes) zeigt, wenn sie auf logisch 0 geht, daß eine Adresse in diesem Bereich angesprochen wird.

Die Startadresse eines jeden Steckplatzes ergibt sich direkt aus der Nummer des Platzes. Steckplatz 3 hat die Startadresse \$C300 (im hexadezimalen Format).

Peripheriekarte PROM Zuweisung  
xx = 00 . . . FF

\$C2xx	2	
\$C3xx	3	
\$C4xx	PROM Raum für Steck-	4
\$C5xx	platznummer	5
\$C6xx		6
\$C7xx		7

Der Adressraum von \$C800 bis \$CFFF ist einem 2 KByte Erweiterungs-ROM oder EPROM vorbehalten. Dieser Bereich ist nur einmal vorhanden und das ROM sollte über eine Selektionslogik auf den Peripheriekarten aktiviert werden.

Das Signal I/O STROBE (PIN 20 eines jeden Steckplatzes) zeigt an, daß der Prozessor auf eine Adresse dieses Bereiches zugreifen möchte.

Auf jeder eingesetzten Peripheriekarte kann ein ROM für diesen Adressraum installiert sein, aber nur jeweils ein ROM darf aktiv sein. Um dies zu erreichen, sollte die Aktivierung des ROMs über ein R-S Flip-Flop gesteuert werden. Der Setzeingang des Flip-Flops sollte durch eine definierte Adresse des I/O SELECT angesteuert und mit der Adresse \$CFFF zurückgesetzt werden. Die Adresse \$CFFF sollte zur Deaktivierung des ROMs oder EPROMs immer benutzt werden. Nach Benutzung dieses Bereiches sollte durch \$CFFF ein eventuell aktives ROM oder EPROM abgeschaltet und anschließend gezielt das neue ROM oder EPROM aktiviert werden. Eine entsprechende Routine kann in dem 256 Byte Adressraum des entsprechenden Steckplatzes abgelegt sein. Ein großer Vorteil dieses Adressbereiches ist, daß bei der Erstellung der Software für diesen Bereich nicht auf Verschiebbarkeit der Software geachtet werden muß, da das ROM unabhängig vom

Steckplatz immer auf den Adressen \$C800 bis \$CFFF liegt.

Beispiel:

```
BIT  $CFFF ; Abschalten aller C8-ROMs,  
BIT  $C300 ; Einschalten des C8-ROM von Slot 3,  
LDA  #$C3  
STA  $7F8  
JSR  $C800 ; Benutzung der C8-ROMs.
```

Da es für viele Aufgaben zweckmäßig ist, neben dem ROM auch einen RAM-Bereich für die Peripheriekarte zur Verfügung zu haben, werden RAM-Adressen, die durch den Bildwiederholungsspeicher nicht benutzt werden, den einzelnen Steckplätzen zugeordnet.

Basis- adressen	I/O RAM Zwischenspeicher						
	Steckplatznummer						
	1*	2	3	4	5	6	7
\$0478	\$0479	\$047A	\$047B	\$047C	\$047D	\$047E	\$047F
\$04F8	\$04F9	\$04FA	\$04FB	\$04FC	\$04FD	\$04FE	\$04FF
\$0578	\$0579	\$057A	\$057B	\$057C	\$057D	\$057E	\$057F
\$05F8	\$05F9	\$05FA	\$05FB	\$05FC	\$05FD	\$05FE	\$05FF
\$0678	\$0679	\$067A	\$067B	\$067C	\$067D	\$067E	\$067F
\$06F8	\$06F9	\$06FA	\$06FB	\$06FC	\$06FD	\$06FE	\$06FF
\$0778	\$0779	\$077A	\$077B	\$077C	\$077D	\$077E	\$077F
\$07F8	\$07F9	\$07FA	\$07FB	\$07FC	\$07FD	\$07FE	\$07FF

\*Diese Adressen werden von den eingebauten seriellen und parallelen Treibern schon benutzt.

## A N H A N G

### INHALTSVERZEICHNIS

#### Anhang A

- 73 Hinweise zur Software-Kompatibilität mit Apple II
- 73 Anpassung des Apple-Pascal 1.1 Systems
- 75 Anpassung von Apple CP/M-Disketten an den BASIS 108
- 75 Durchführung der Anpassung
- 76 Eigenschaften der CP/M-Diskette nach der Anpassung
- 77 Anpassung des Applesoft oder Integer Basics von Apple
- 78 Laden des Basics
- 79 Beschreibung der Basicversionen

#### Anhang B

- 81 Volume UT108

#### Anhang C

- 85 BASIS 108 System Monitor

#### Anhang D

- 87 Hinweise zu Applesoft Basic FP40 und FP80

#### Anhang E

- 88 V24 Parameter

#### Anhang F

- 90 Anschluß eines Fernsehgerätes ohne Videoeingang

#### Anhang G

- 91 Arbeiten mit dem Kassettenrekorder
- 91 Schreiben eines Speicherbereiches auf Kassette
- 92 Lesen eines Speicherbereiches von der Kassette

#### Anhang H

- 93 Hexadezimalzahlen

#### Anhang I

- 94 Tabelle der Tastenbelegung

Anhang J  
97 Zusammenstellung der Ein-/Ausgabeadressen

Anhang K  
99 Der Z-80-Teil  
99 Einleitung  
99 Taktgenerierung  
99 Kontrolle des Z-80-Teiles  
100 Anpassung des Adress Bus  
100 DMA Daisy Chain  
101 Interrupts

102 Anhang L  
Datenblatt und Befehlsregister des Z-80

Anhang M  
Datenblatt und Befehlsregister des 6502

Anhang N  
Auflistung der Monitor-ROM Programmbefehle

Anhang O  
Stichwortverzeichnis

Anhang P  
Schaltung der Tastaturplatine

Anhang Q  
Schaltung der Hauptplatine

## A N H A N G   A

### HINWEISE ZUR SOFTWARE-KOMPATIBILITÄT MIT APPLE II

Die ZAP:-Diskette erfüllt drei verschiedene Funktionen:

1. Modifizierung des Apple-Pascal 1.1-Systems, so daß die 80-Zeichendarstellung und die eingebaute Parallel- und Seriellschnittstelle verfügbar sind.
2. Modifizierung des Microsoft CP/M-Systems, um ebenfalls die 80-Zeichendarstellung und die Schnittstellen verfügbar zu machen.
3. Laden der gewünschten BASIC-Version.

Das Herstellen dieser Modifizierungen brauchen Sie nur einmal durchzuführen, mit den geänderten Disketten können Sie dann arbeiten, wie in anderen Systemen auch üblich. Siehe auch Kapitel 2 und die entsprechenden Betriebshandbücher.

Die Beschreibung für diese Operationen setzt zwei Laufwerke voraus. Bei nur einem Laufwerk bitten Sie Ihren Händler um Hilfe beim Anpassen der Disketten.

#### Zu 1. Hinweise zur Anpassung des Apple-Pascal 1.1 Systems

Um die gewünschte Pascalversion zu erhalten, müssen die Files SYSTEM.APPLE und SYSTEM.MISCINFO, die sich auf der Diskette APPLE1: befinden, verändert werden.

Verwenden Sie für die Anpassung eine Kopie Ihrer Apple-Pascal Diskette, nicht das Original.

Im folgenden werden im Text die Abläufe intern und extern beschrieben. Dann folgen die Ein- und Ausgaben auf dem Bildschirm. Dabei sind Ihre Eingabebefehle gesperrt gedruckt und die Ausgaben in Großschreibung ausgeführt. Nur die zu drückende Returntaste ist bei Ihren Eingaben als (RETURN) angegeben.

Transferieren Sie zunächst wie folgt das File SYSTEM.APPLE von der Diskette APPLE1: auf die Diskette ZAP:. Stecken Sie die Diskette APPLE1: in Laufwerk 1, die Diskette ZAP: in Laufwerk 2 und schalten Sie den Rechner ein. Durch Drücken der Taste F gelangen Sie in den Filer. Rufen Sie nun die Transferroutine durch Drücken der Taste T auf:

F  
FILER: G, S, W, N, L, R, C, T, D, Q  
T  
WHAT FILE ? APPLE1:SYSTEM.APPLE (RETURN)  
TO WHERE ? ZAP:\$ (RETURN)  
Q

Durch das Drücken von Q gelangen Sie wieder zur Kommandozeile. Es geht weiter mit dem Drücken der Taste X :

X  
EXECUTE WHAT FILE ? ZAP:ZAP (RETURN)  
VERSION 2.0 ZAP, 29-MARCH-82 (das datum muß nicht  
(C) SANDOR SCARI 1982 identisch sein)  
BUFFER SIZE: 54 BLOCKS  
COMMAND CONSOLE: COMMAND 'ZAP:PASCAL' (RETURN)

Das Programm ZAP legt jetzt auf der Diskette Zap: eine modifizierte Version des Files SYSTEM.APPLE unter dem Namen NEW.APPLE ab. Während das Programm arbeitet, läuft das Laufwerk, in dem sich die ZAP:-Diskette befindet. Außerdem erscheinen verschiedene Texte auf dem Bildschirm. Nach Beendigung des Programms erscheint die Kommandozeile auf dem Bildschirm. Die Files NEW.APPLE und 108.MISCINFO müssen nun von der Diskette ZAP: auf die Diskette APPLE1: mit folgenden Kommandos transferiert werden:

F  
FILER: G, S, W, N, L, C, T, D, Q  
T  
TRANSFER WHAT FILE ? ZAP:NEW.APPLE (RETURN)  
TO WHERE ? APPLE1:SYSTEM.APPLE (RETURN)  
REMOVE OLD SYSTEM.APPLE ? Y  
T  
TRANSFER WHAT FILE ? ZAP:108.MISCINFO (RETURN)  
TO WHERE ? APPLE1:SYSTEM.MISCINFO (RETURN)  
REMOVE OLD SYSTEM.MISCINFO ? Y

Hiermit ist die Prozedur der Änderung der Diskette APPLE1 für das Apple-Pascal beendet.

Laden Sie Ihr System neu. Wenn Sie die Reihenfolge eingehalten haben und alle Operationen richtig ausgeführt haben, arbeitet Ihr Apple Pascal 1.1 jetzt mit 80 Zeichen/Zeile.

Im folgenden sind einige Zeichen aufgeführt, die durch die Änderung der Diskette anders sind.

1. Editor-Accept ist die 'HOME' Taste des Cursorblocks, bei Apple CRTL-C.
2. Die Pfeiltaste ' $\leftarrow$ ' entspricht der Apple-Taste 'Pfeil links', die Pfeiltaste ' $\rightarrow$ ' der Apple-Taste 'Pfeil rechts'.  
Die Pfeiltasten 'rechts, links, oben und unten' werden vom Editor richtig gedeutet und ausgeführt.
3. Die Zusatztaste Shift-CTRL-F15 ist mit BREAK belegt.
4. Die Zusatztaste Shift F1 ist mit Stop belegt. (Hält die Ausgabe an).
5. Die Zusatztaste Shift F2 ist mit Flush belegt.  
(Bildschirmausgabe wird unterdrückt).

Alle anderen Zusatztasten können Sie frei verwenden (Zusatztasten sind daran zu erkennen, daß Bit 7 gesetzt ist, d. h. ASCII über 127).

Hier ein Auszug aus einem entsprechenden Abfrageprogramm:

```
Read(Keyboard,ch);
if ord(ch) ≥ 128 then
writeln('Funktion' ,ord(ch):4).
```

## zu 2) Anpassung von Apple CP/M-Disketten an den BASIS 108

Ziel der Anpassung ist es, die 80-Zeichen-Darstellung, das Parallelinterface und die V24-Schnittstelle des BASIS 108 unter CP/M nutzen zu können.

Wie im vorigen Abschnitt sind die Ausgaben des Computers großgeschrieben, Ihre Befehle dagegen fett gedruckt.

Verwenden Sie für die Anpassung eine Kopie Ihrer CP/M-Diskette, nicht das Original.

### Durchführung der Anpassung

Sie benötigen zur Anpassung eine Pascal-Diskette. Sollten Sie kein Pascal-System haben, so bitten Sie Ihren Händler, für Sie die folgende Procedur auszuführen. Im folgenden ist die Version beschrieben, wenn Sie das System UCSD IV.0 verwenden. Haben Sie das System APPLE1, so lassen Sie jeweils das .IV hinter dem ZAP fort.

1. Laden Sie nun als erstes Ihr Pascalsystem in Laufwerk 4 und dann die Diskette ZAP: in Laufwerk 5.

2. Starten Sie das Programm ZAP.IV auf der Diskette ZAP durch den Befehl X und antworten Sie entsprechend dem Fettdruck im folgenden:

X  
EXECUTE WHAT FILE ? **ZAP:ZAP.IV** (RETURN)  
VERSION IV.O ZAP, 27-MAY-1982 (das datum muß nicht  
(c) SANDOR SCARI 1982 identisch sein.)  
COMMAND CONSOLE: **COMMAND 'ZAP:CPM'**

ACHTUNG: bevor Sie Return drücken, müssen Sie nun Ihre CP/M-Diskette in Drive 4 stecken. Es findet keine Prüfung, ob die CP/M-Diskette wirklich in Laufwerk 4 steckt, statt.

(RETURN) .

Erst nach dem Drücken der (RETURN)-Taste wird das CP/M-System angepaßt.

#### Eigenschaften der CP/M Diskette nach der Anpassung

Dem logischen Drucker LST: kann mit Hilfe des Stat-Programms

entweder PLT: (Parallelprinter)  
oder UL1: (serieller Printer)  
zugeordnet werden.

PUN: kann UP1:  
und  
RDR: kann URL:  
zugeordnet werden.

Die serielle Schnittstelle (UP1: und UL1:)  
hat die voreingestellte Baudrate von 9600 Bits/s.  
Übertragen werden: 8 Datenbits, 2 Stopppbits, kein Paritätsbit.

Die Baudrate kann durch Beschreiben der Adresse \$F280 eingestellt werden,  
siehe nächste Seite.

Wie Sie den entsprechenden Handbücher über CP/M entnehmen können, haben Sie hier Änderungsmöglichkeiten über DDT.

Befehl in \$F280	Baudrate
\$91	50
\$92	75
\$93	110
\$94	134,5
\$95	150
\$96	300
\$97	600
\$98	1200
\$99	1800
\$9A	2400
\$9B	3600
\$9C	4800
\$9D	7200
\$9E	9600
\$9F	19200

### Zu 3. Anpassung des Applesoft oder Integer Basics von Apple

Bevor Sie von der ZAP:-Diskette die gewünschte Basicversion laden können, müssen die Files INTBAS.DATA und FPBAS.DATA von der BASICS:-Diskette, die mit den Floppydisklaufwerken mitgeliefert wird, mit Hilfe des Pascalsystems auf die ZAP:-Diskette kopiert werden.

Stecken Sie zu diesem Zweck die Diskette UCSD IV.0 (oder APPLE1, dann entfällt jeweils das .IV in den Kommandos) in Laufwerk 1, die Diskette ZAP: in Laufwerk 2 und schalten Sie nun den Rechner ein.

Sollte zuvor die Modifizierung des Pascalsystems vorgenommen worden sein, so befinden sich auf der Zap:-Diskette noch die Files SYSTEM.APPLE und NEW.APPLE, die aus Platzgründen wieder gelöscht werden müssen.

Um ein File löschen zu können, muß die Taste R (für Remove) gedrückt werden.

Wie bislang werden Ihre Befehlseingaben fett gedruckt und die Ausgaben groß geschrieben:

```
F  
FILER: G, S, N, L, R, C, T, D, Q, W, B, E, K, M, P, V, X, Z  
R  
REMOVE WHAT FILE ? ZAP:=.APPLE (RETURN)  
ZAP:SYSTEM.APPLE - REMOVED  
ZAP:NEW.APPLE - REMOVED  
  
UPDATE DIRECTORY ? Y
```

Sollten beide Files schon nicht mehr auf der Diskette sein, so erscheint auf dem Bildschirm anstelle der Bestätigung die Meldung:

FILE NOT FOUND .

Um den freien Speicherplatz auf der Diskette voll nutzen zu können ist es nötig, durch Drücken der Taste K die Crunch-Routine zu starten.

K  
CRUNCH WHAT VOL ? ZAP: (RETURN)  
FROM END OF DISK; BLOCK 280 ? (Y/N) Y  
....  
ZAP: CRUNCHED ,

werden Files verschoben, so wird dies auf dem Bildschirm angezeigt . . . .  
Tauschen Sie nun die Diskette APPLE 1: in Laufwerk 1 gegen die BASICS-Diskette aus.

Machen Sie weiter mit Drücken der Taste T (für Transfer):

T  
TRANSFER WHAT FILE ? BASICS:=BAS.DATA (RETURN)  
TO WHERE? ZAP:\$ (RETURN) .

In Laufwerk 1 muß nun die BASICS:-Diskette wieder gegen die APPLE1:-Diskette ausgetauscht werden. Drücken der Taste Q läßt wieder die Kommandozeile auf dem Bildschirm erscheinen.  
Um aus den transferierten Files die verschiedene Basicversion zu erzeugen, muß das auf der ZAP:-Diskette befindliche Programm ZAP gestartet werden.  
Drücken Sie zu diesem Zweck die Taste X (für Execute), zunächst jedoch:

Q  
X  
EXECUTE WHAT FILE ? ZAP:ZAP.IV (RETURN)  
VERSION 2.0 ZAP, 29-MARCH-82 c(datum kann anders sein)  
(C) SANDOR SCARI 1982  
BUFFER SIZE: 56 BLOCKS  
  
COMMAND 'CONSOLE:'  
COMMAND 'ZAP: BASIC' (RETURN) .

Nach Ablauf des Programms können die verschiedenen Basicversionen von der Zap:-Diskette geladen werden.

## **Laden des Basics**

Da der BASIS 108 kein Basic in ROMs hat, muß bei Verwendung von Basicprogrammen nach dem Einschalten einmal die gewünschte Basicversion geladen werden.

Legen Sie die ZAP:-Diskette in Laufwerk 1 und schalten Sie den Rechner ein. Auf dem Bildschirm erscheint nun:

INTERPRETER FILES:	(die reihenfolge kann auch vertauscht sein.)
A: FPBAS.DATA	
B: INTBAS.DATA	
C: VC.16	
D: FP 40	
E: FP 80	
F: INT 40	.

Sie können nun die gewünschte Version mit einem der Buchstaben A ... F wählen.

! Sollte auf dem Bildschirm keine derartige Auflistung zu sehen sein, sind die am Anfang dieses Punktes beschriebenen Tätigkeiten noch nicht, oder nicht richtig ausgeführt worden.

## **Beschreibung der Basicversionen**

### **FPBAS.DATA**

Original Applesoft mit Apple-Autostart-Monitor  
(Der BASIS 108 verhält sich wie ein Apple II mit Applesoft).

### **INTBAS.DATA**

Apple Integer Basic mit Apple-Autostart-Monitor  
(Der BASIS 108 verhält sich wie ein Apple II mit Integerbasic).

### **VC.16**

Muß vorgeladen werden, bevor **Visicalc** geladen wird.

### **FP40**

Floatingpointbasic mit 40-Zeichendarstellung,  
Cursorblock ist aktiv, Groß/Kleinschreibung,  
Funktionstasten liefern ASCII-Zeichen 128.

**FP80**

Floatingpointbasic mit 80-Zeichendarstellung,  
Cursorblock ist aktiv, Groß/Kleinschreibung,  
Funktionstasten liefern ASCII-Zeichen 128

**INT40**

Integerbasic mit 40-Zeichendarstellung,  
Cursorblock ist aktiv, Groß/Kleinschreibung,  
Funktionstasten liefern ASCII-Zeichen 128.

Die FP-Versionen sind verbessertes Applesoft, die Verbesserungen bzw. Zusatzmöglichkeiten entnehmen Sie bitte Anhang D.

Sie arbeiten nun mit der entsprechenden Version des Basics, die Sie gewählt haben, indem Sie die entsprechende DOS-System-Diskette in das Laufwerk 1 einlegen und (RETURN) drücken.

## A N H A N G    B

### **Volume UT108:**

Auf der Rückseite der ZAP:-Diskette befinden sich einige nützliche Programme, die unter den Betriebssystemen Pascal, CP/M und DOS eingesetzt werden können. Folgende Möglichkeiten sind gegeben:

Anpassung an verschiedene Drucker,  
Veränderung des Bildschirm-Zeichensatzes,  
Erhöhung der Diskettenkapazität ( nur unter Pascal ),  
Serielle Schnittstelle und Kleinschreibung unter DOS,  
Demonstrations-Programme.

### **Benutzung der Diskette unter Apple Pascal Version 1.1**

#### **DISPLAY.TEXT und DISPLAY.CODE, DISPLAY.A2.TEXT und DISPLAY.A2.CODE**

Stellen Sie zunächst fest, welche Revisionsnummer Ihr Computersystem hat. Für Systeme mit der Revisionsnummer A2, die vor Sommer 1982 ausgeliefert wurden, wählen Sie die Programme DISPLAY.A2.TEXT und DISPLAY.A2.CODE. Sie finden diese Nummer auf der Hauptplatine. Mit dem Programm DISPLAY.CODE lassen sich die verfügbaren Zeichensätze des BASIS 108 darstellen und durch die entsprechende Eingabe umstellen. Die Umstellung ist aber nur temporär und lässt sich mit diesem Programm nicht auf der Boot-Diskette festhalten. ( Wenn Sie eine Änderung auf der Diskette vornehmen wollen, so können Sie dies mit dem Programm PRNT/V24.CODE erreichen. ) DISPLAY.TEXT ist das dazugehörige Textfile.

X Execute what file? **UT108:DISPLAY** (RETURN) .

#### **FORMAT40.CODE**

Mit diesem Programm können Sie die Speicherkapazität von 5 1/4" Disketten auf 160 KByte erhöhen, sofern Sie die entsprechenden Laufwerke besitzen. Dies geschieht durch Formattierung von 40 Spuren.

X Execute what file? **UT108:FORMAT40** (RETURN) .

## **PRNT/V24.CODE**

Mit diesem Programm können Sie den BASIS 108 an die Erfordernisse ihres Druckers anpassen. Dabei lassen sich folgende Parameter ändern:

Baudrate	( 50..19200 )
Databits	( 5,6,7,8 )
Parity	( j/n )
Stopbits	( 1,2 )
Printer: an V24-Schnittstelle	( j/n )
Bildschirm-Zeichensatz	.

Die Änderung des Bildschirm-Zeichensatzes läßt sich auf der Bootdiskette eintragen, so daß der angewählte Zeichensatz beim erneuten Booten automatisch eingestellt wird.

X Execute what file? **UT108:PRNT/V24 (RETURN)** .

## **6551.TEXT**

Dieses Textfile ist der modifizierte Treiber für die serielle Schnittstelle.

## **Benutzung der Diskette unter CP/M**

### **DEUTSCH, ASCII, APL**

Die auf der Diskette verfügbaren Files APL, ASCII, DEUTSCH ermöglichen eine Veränderung des Bildschirm-Zeichensatzes, die durch Aufruf des entsprechenden Programmes realisiert wird. Beispiel:

**DEUTSCH (RETURN) .**

Hiermit stellen Sie den BASIS 108 auf den deutschen Zeichensatz um.

### **REBOOT**

Wenn Sie dieses Programm ausführen, haben Sie die Möglichkeit, das System durch Eingabe von SHIFT SHIFT CONTROL von der Tastatur aus neu zu booten.

**REBOOT (RETURN) .**

### **SYSWRT**

Mit diesem Programm können Sie Boot-Disketten für den BASIS 108 herstellen. Die Disketten müssen formatiert sein.

**SYSWRT (RETURN) .**

### **V24**

Mit diesem Programm können Sie den BASIS 108 an die Erfordernisse ihres Druckers anpassen. Dabei lassen sich folgende Parameter ändern:

Baudrate	( 50..19200 )
Databits	( 5,6,7,8 )
Parity	( j/n )
Stopbits	( 1,2 )
Printer: an V24-Schnittstelle	( j/n )
Bildschirm-Zeichensatz	.

Auch die Änderung des Bildschirm-Zeichensatzes läßt sich auf der Boot-Diskette eintragen, so daß der angewählte Zeichensatz beim erneuten Booten automatisch eingestellt wird.

**Wichtig:** Da beim erneuten Booten die V24-Schnittstelle nicht automatisch angesprochen wird, müssen Sie folgende Zuweisung unter CP/M tätigen.

**STAT LST:=UL1: .**

## **Benutzung der Diskette unter DOS**

Die deutsche Programmversion wird durch ein D hinter dem Programmnamen gekennzeichnet.

### **PRINTER/V24 und PRINTER/V24 D**

#### **PRINTER/V24 V2.1 und PRINTER/V24 V2.1 D**

Mit diesen Programmen können Sie den BASIS 108 an die Erfordernisse ihres Druckers anpassen. Für die Anpassung brauchen Sie nur eines der Programme aufzurufen, die für Ihr Computersystem richtige Version wird automatisch ausgeführt. Es lassen sich folgende Parameter ändern:

Drucker und V24 CR- CR/LF Übersetzung	( n,j )
Drucker und V24 Bildschirmecho	( n,j )
V24 Baudrate	( 50..19200 )
V24 Databits	( 5,6,7,8 )
V24 Paritätsbit	( j/n )
V24 Stopbits	( 1,2 ) .

### **DOS PATCH und DOS PATCH D**

Nach der Ausführung dieses Programms lässt sich die Kleinschreibung auch für DOS-Kommandos verwenden. Außerdem kann Kleinschrift aus Textfiles gelesen werden. Unter PR 9 lässt sich die serielle Schnittstelle ansprechen.

### **RENUMBER UPDATE und CHAIN UPDATE**

Wenn Sie eine überarbeitete Version des Programms RENUMBER erhalten wollen, gehen Sie am besten wie folgt vor:

Laden Sie das Programm RENUMBER UPDATE von der Diskette UT108:

**LOAD RENUMBER UPDATE,S6,D1 .**

Dann legen Sie eine nicht schreibgeschützte Diskette mit dem File RENUMBER in das Laufwerk D1 und starten das Programm RENUMBER UPDATE.

**RUN .**

Wenn keine Fehlermeldungen erscheinen, war die Überarbeitung erfolgreich. Die überarbeitete Version des Programms CHAIN erhalten sie in der gleichen Weise. Ersetzen Sie bei den oben angegebenen Befehlen RENUMBER durch CHAIN.

### **NEW FP DEMO , CHRGEN und COLOR DEMO108**

Diese Programme werden als Demonstrationsbeispiele zum Bildschirm-Zeichensatz und zur Farbdarstellung mitgeliefert. Weiterhin sei darauf hingewiesen, daß das FP80 BASIC einige Vorteile gegenüber dem Applesoft enthält.

## A N H A N G C

### BASIS 108 Monitor-ROM

Bildschirm:

Apple 24x40	BASIS 108 24x40*	24x80
ESC-		HOME
ESC-E	(Pfeil Ecke oben links)	
ESC-F	(Pfeil Ecke oben rechts)	
ESC-I	(Pfeil oben)	
ESC-M	(Pfeil unten)	
ESC-J	(Pfeil links)	
ESC-K	(Pfeil rechts)	
(Pfeil links)	(Pfeil Ecke unten links)	
(Pfeil rechts)	(Pfeil Ecke unten rechts)	

\*Monitor-ROM mit 40 Zeichen/Zeile  
oder entsprechende Version aus ZAP.

Kassette:

xxxx.yyyy R	xxxx.yyyy R	-----
xxxx.yyyy W	xxxx.yyyy W	-----

BASIC Kaltstart (nur ohne Disk):

CTRL-B	CTRL-B	CTRL-B
--------	--------	--------

BASIC Warmstart:

ohne Disk.	CTRL-C	Q	Q
mit Disk.	3D0G	Q	Q
LO-RES	40x40	40x40	40x80
	48x40	48x40	48x80

Disas	xxxx.yyyyL	xxxx.yyyyL	xxxx.yyyyL
-------	------------	------------	------------

Apple  
24x40

BASIS 108  
24x40\*          24x80

Eingabe-Vector:

nCTRL-K              nK              nK

Ausgabe-Vector:

nCTRL-P              nP              nP

6502-Register zeigen:

CTRL-E              ?              ?

User-Programm:

CTRL-Y              U              U

Eingabe:

nur Groß-  
buchstaben

Groß-/Kleinbuchstaben

6502 Programm starten:

xxxxG

xxxxG

xxxxG

Move              xxxx<yyyy.zzzzM  
Verify            xxxx<yyyy.zzzzV  
Display           xxxx.yyyy

(unverändert)  
(unverändert)  
(unverändert, zeigt jedoch  
16 Bytes/Zeile).

## A N H A N G D

### Hinweise zu Applesoft BASIC FP40 und FP80

#### 1. Folgende Fehler wurden beseitigt:

FOR I=S TO P ist nicht mehr FOR I=STOP  
Da hier Blanks beachtet werden, müssen Befehle wie COLOR=,  
TAB( ohne Blank vor dem Sonderzeichen geschrieben werden.

TAB(..), SPC(..), HTAB,  
(bleibt immer im eingestellten Bildfenster).  
S. Applesoft Ref. Manual, Seite 129.

LEFT\$(A\$,0) ergibt String der Länge 0 ohne Fehlermeldung  
RIGHT\$(A\$,0) entsprechend.

#### 2. Erweiterungen

Bei der Version 80 Zeichen/Zeile können im Grafik Modus LORES  
80x40 oder 80x48 Bildpunkte gesetzt werden.

Der INPUT-Befehl kann kleine und große Buchstaben annehmen, allerdings keine Zusatztasten.

Der GET-Befehl unterstützt auch die Zusatztasten:

GET A\$: IF ASC(A\$) > 127 THEN PRINT "Zusatztaste"; ASC(A\$)-160:

Schlüsselwörter und Variable dürfen kleingeschrieben werden.

Es gibt drei Möglichkeiten auszugeben:

normal,  
flash und  
inverse.

Das bedeutet aber, daß  $3 * 96 = 256$  Zeichen belegt sind, deshalb gibt es nicht gleichzeitig INVERSE und FLASH.

## A N H A N G E

### V24 Parameter

	DOS BASIC	UCSD II.1.1 ( 6502 Adressen )	UCSD IV.0	CP/M ( Z-80 Adr. )
Baudrate, Wortlänge, Stopbits: 6551 Control Reg.	\$06F9	\$FFCE	\$0271	\$F280
Parität: RTS, DTR 6551 Command Reg.	\$0779	\$FFCF	\$0270	\$F281
Gerätename Eingabe:	IN #9 *)	remin: #7:	remin: #7:	UR1:
Ausgabe:	PR #9 *)	remout: #8:	remout: #8:	UL1: UP1:

Paralleler Druckerausgang:

Gerätename Ausgabe:	PR #1	printer: #6:	printer: #6:	LPT:
------------------------	-------	-----------------	-----------------	------

\*) Bemerkung:

DOS 3.3 erlaubt IN # und PR # nur im Bereich 0..7. Damit IN#9 und PR#9 in DOS auch möglich sind, muß POKE 41153,10 geändert werden..

Disketten, die mit geändertem DOS angelegt werden, erlauben IN#9 und PR#9 ohne weitere POKE-Befehle.

CR - CR/LF Übersetzung (gilt nur für DOS/BASIC)

Übersetzung.	keine	ein	keine	ein
Bildschirmschirmecho	kein	kein	ein	ein
\$0679:	\$00	\$80	\$40	\$C0
\$05F9:	\$A5	\$25	\$E5	\$65.

Wenn die 2 Bytes bei \$0679 und \$05F9 nicht zusammenpassen, werden alle Drucker und V24-Parameter auf die Standardeinstellung gesetzt:

Standardeinstellung des V24: 9600 Baud, 2 Stopbits, keine Parität  
CR - CR/LF Übersetzung ein, Bildschirmschirmecho ein.

Paralleler Druckerausgang: CR- CR/LF Übersetzung ein, Bildschirmschirmecho ein.

## **A N H A N G   F**

### **Anschluß eines Fernsehgerätes ohne Videoeingang**

Besorgen Sie sich bei Ihrem BASIS Vertriebspartner einen UHF-Modulator, der das Video-Signal in ein HF-Signal umwandelt.

Bitte lesen Sie zunächst S. 8 "Öffnen des Systems" und dann auch entsprechend auf S. 10 "Hauptplatine".

Ziehen Sie den Stecker auf der linken oberen Seite der Platine Verbindungskabel zum Außenstecker für Video) und befestigen Sie das lose Kabel mit einem Klebstreifen an der Gehäuserückwand. Stecken Sie nun den entsprechenden Stecker des Modulators auf die Stifteleiste. Den Modulator befestigen Sie am besten ebenfalls mit Klebstreifen an der Rückwand. Das Anschlußkabel für das Fernsehgerät wird vom Modulator durch den Durchbruch auf der Gehäuserückseite nach außen geführt. Auf Kanal 36 (beachten Sie aber bitte hierzu die Angaben beim Modulator) können Sie die Datenausgabe Ihres BASIS 108 empfangen. Bitte bedenken Sie aber, daß die Qualität der Zeichendarstellung durch den Umweg über den Modulator leidet und nicht mit einem guten Monitor vergleichbar ist.

Es sei nochmals darauf hingewiesen, daß ein normales Fernsehgerät mehr als 40 Zeichen/Zeile nicht sauber darstellen kann.

Sollten Sie großen Wert auf gute Farbausgabe legen, dann benötigen Sie einen hochauflösenden RGB-Monitor. Ihr BASIS-Vertriebspartner wird Sie auch in dieser Angelegenheit beraten.

## A N H A N G   G

### Arbeiten mit dem Kassettenrekorder

#### Schreiben eines Speicherbereichs auf Kassette

Dieses Monitor-Kommando kann nur ausgeführt werden, wenn der Monitor ROM in Ihren BASIS 108 mit 40 Zeichen/ Zeile arbeitet. D.h., Sie können hiermit arbeiten, wenn Sie FPBAS.DATA, INTBAS.DATA, FP40 oder INT40 geladen haben. Wollen Sie allerdings von der Kassette Basic laden, arbeiten Sie also ohne Diskettenlaufwerk, dann benötigen Sie den Monitor ROM für 40 Zeichen/ Zeile. Die Unterschiede der beiden Monitor ROMs sind in Anhang M aufgelistet.

Zwei spezielle Kommandos ermöglichen es Ihnen Speicherbereiche auf die Kassette Ihres Kassettenrekorders zu schreiben und bei späterem Gebrauch wieder einzulesen. Das erste dieser beiden Kommandos, das WRITE-Kommando, schreibt den Inhalt von einer oder bis zu 65536 Speicherstellen auf die Kassette.

Um einen solchen Speicherbereich auf Kassette zu schreiben, geben Sie dem Monitor die Anfangs- und Endadresse des Speicherbereichs, gefolgt von einem W (für WRITE=Schreiben) ein.

Um fehlerfrei aufnehmen zu können, muß der Kassettenrekorder auf "Aufnahme" stehen, bevor Sie (RETURN) nach Ihrer Eingabe tippen. Lassen Sie das Band ein paar Sekunden laufen, bevor Sie (RETURN) tippen. Der Monitor schreibt eine 10 Sekunden lange Vorinformation (HEADER) auf das Band und dann erst die Daten. Sobald der Vorgang beendet ist, meldet der Monitor sich mit einem Ton aus dem Lautsprecher und wartet auf weitere Anweisungen. Sie können dann das Band zurückspulen, es aus dem Rekorder nehmen und mit einer Inhaltsangabe versehen.

Beispiel:

```
*0.14(RETURN)
0000: FF FF AD 30 C0 88 D0 04 C6 01 F0 08 CA D0 F6 A6
0010: 00 4C 02 00 60
*0.14W          c(kassettenrekorder auf aufnahme
                  schalten und zehn sekunden
                  laufen lassen)
                  (RETURN)
*
```

Es dauert ca. 20 Sekunden (einschl. der 10 Sekunden für die Vorinformation), um die Werte von 4096 Speicherstellen auf Band zu schreiben. Dabei werden ca. 3000 Bit pro Sekunde übertragen. Wenn alle Daten übertragen sind, schreibt der Monitor noch einen zusätzlichen Wert auf das Band; die "Prüfsumme", die aus allen übertragenen Werten des Speicherbereichs gebildet wird. Das READ-Kommando (siehe unten)

benutzt diesen Wert, um Übertragungsfehler festzustellen. Die Prüfsumme ist anfangs \$FF und wird durch Exclusive-OR von jedem Wert des übertragenen Bereichs verändert.

#### Lesen eines Speicherbereichs von der Kassette

Den mit Hilfe des WRITE-Kommandos auf Band geschriebenen Speicherbereich können Sie mit dem READ-Kommando (Lesen) R wieder in einen von Ihnen zu bestimmenden Bereich einlesen.

Geben Sie auch hier nicht sofort das (RETURN), sondern stellen Sie den Kassettenrekorder auf "Wiedergabe" und warten Sie, bis das Vorspannband durchgelaufen ist. Obwohl das WRITE-Kommando eine 10 Sekunden lange Vorinformation geschrieben hat, braucht das READ-Kommando nur drei Sekunden, um sich auf die Frequenz einzustellen. Sie sollten also ein paar Sekunden vergehen lassen, bis Sie die (RETURN)-Taste tippen.

Beispiel:

```
*0:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (RETURN)
*0.14 (RETURN)
0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00
*0.14R           c(kassettenrekorder einschalten
                  einige sekunden warten)
                  (RETURN)
*0.14 (RETURN)
0000: FF FF AD 30 C0 88 D0 04 C6 01 F0 08 CA D0 F6 A6
0010: 00 4C 02 00 60
*
```

Nachdem der Monitor alle Werte gelesen und gespeichert hat, liest er die auf Band gespeicherte Prüfsumme und vergleicht sie mit der soeben beim Lesen erstellten Prüfsumme. Weichen beide Werte voneinander ab, gibt der Monitor ein Signal zum Lautsprecher und schreibt ERR (Fehler) auf den Bildschirm. Sie erhalten also eine Warnung, daß beim Lesen der Daten ein Fehler aufgetreten ist und die im Speicher befindlichen Werte nicht mit den aufgezeichneten Werten übereinstimmen. Wenn die Prüfsumme stimmt, erwartet der Monitor weitere Anweisungen von Ihnen.

#### Wichtig

Es sei hier nochmals darauf hingewiesen, daß die soeben behandelten Kommandos W und R nur in dem Monitor ROM für 40 Zeichen/ Zeile vorhanden sind.  
Siehe auch Anhang M.

## A N H A N G   H

### Hexadezimalzahlen

Eine Vielzahl von Adressen und Werten, vor allem im Monitor ROM oder bei Arbeiten mit anderen Speichern, benötigt man die Angaben in hexadezimaler Schreibweise.

Diese Schreibweise verwendet neben den Ziffern 0 bis 9 zusätzlich die Buchstaben A bis F, um die Werte 10 bis 15 darzustellen. Eine Hexadezimalziffer kann deshalb die Werte von 0 bis 15 annehmen. Damit stellen also zwei Hexadezimalziffern die Dezimalzahlen von 0 bis 255 und eine Gruppe von vier Ziffern den Bereich von 0 bis 65535 dar.

Eine Adresse wird im BASIS 108 also durch vier Hexadezimalziffern und jeder Wert (Inhalt einer Speicherstelle) durch zwei Hexadezimalziffern dargestellt. Um die Umrechnung Hexadezimalziffern in Dezimalzahlen zu erleichtern und zu veranschaulichen dient die folgende Tabelle.

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

## A N H A N G I

### Tabelle der Tastenbelegung

In der folgenden Tabelle wird der ASCII-Zeichensatz mit der Tastenbelegung und den zugehörigen Hexadezimalzahlen aufgeführt.

Da die Zifferntastatur nur immer entsprechend einfach belegt ist, wird hier nur das Haupttastenfeld und der Cursorblock behandelt.

Es gelten folgende Abkürzungen: CT - CTRL, SH - SHIFT.

Werden Zeichen bei den Tasten durch einen Bindestrich verbunden, so bedeutet das, daß diese Tasten gleichzeitig gedrückt werden müssen.

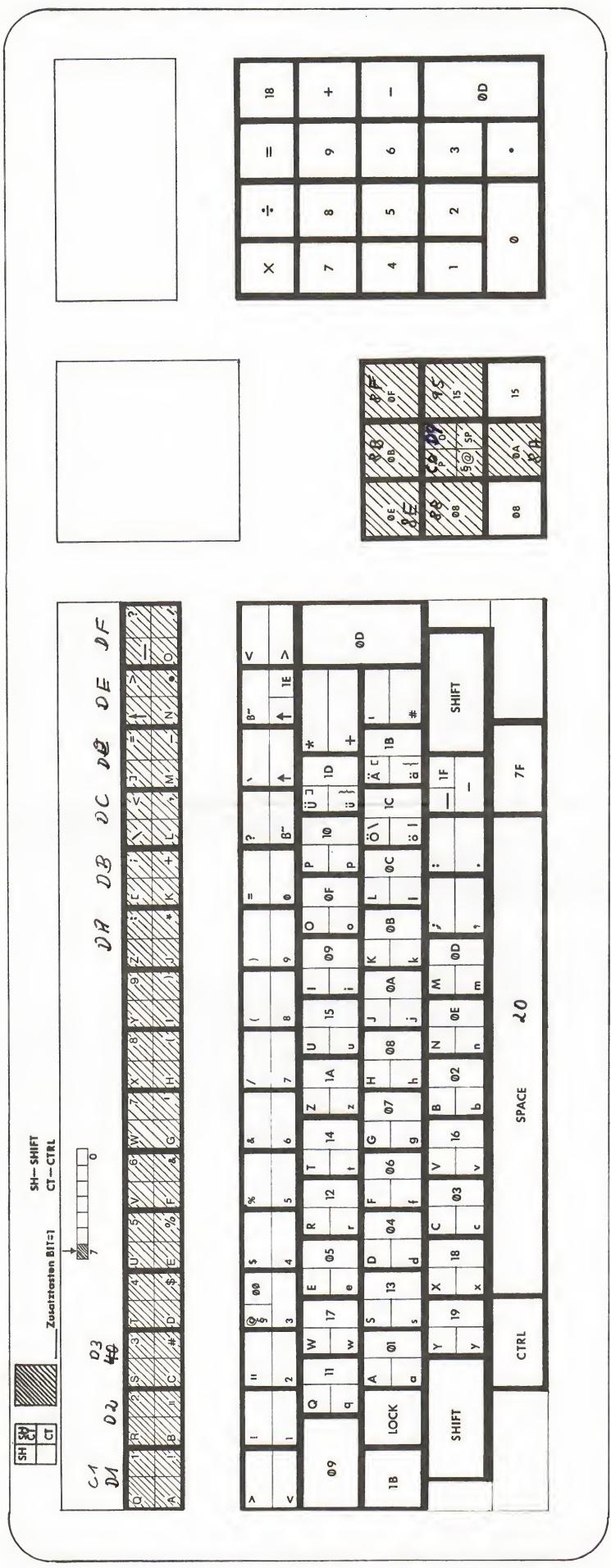
Hex.	ASCII	Taste	Hex.	ASCII	Taste
\$00	nul	CT-SH-3	\$20	space	Space
\$01	soh	CT-a	\$21	!	SH-1
\$02	stx	CT-b	\$22	"	SH-2
\$03	etx	CT-c	\$23	#	#
\$04	eot	CT-d	\$24	\$	SH-4
\$05	enq	CT-e	\$25	%	SH-5
\$06	ack	CT-f	\$26	&	SH-6
\$07	bel	CT-g	\$27	'	SH-#
\$08	bs	=	\$28	(	SH-8
\$09	ht	TAB	\$29	)	SH-9
\$0A	lf	CT-j	\$2A	*	SH-+
\$0B	vt	CT-k	\$2B	+	+
\$0C	ff	CT-l	\$2C	,	,
\$0D	cr	CT-m	\$2D	-	-
\$0E	so	CT-n	\$2E	.	.
\$0F	si	CT-o	\$2F	/	SH-7
\$10	dle	CT-p	\$30	0	0
\$11	dcl	CT-q	\$31	1	1
\$12	dc2	CT-r	\$32	2	2
\$13	dc3	CT-s	\$33	3	3
\$14	dc4	CT-t	\$34	4	4
\$15	nak	=	\$35	5	5
\$16	syn	CT-v	\$36	6	6
\$17	etb	CT-w	\$37	7	7
\$18	can	CT-x	\$38	8	8
\$19	em	CT-y	\$39	9	9
\$1A	sub	CT-z	\$3A	:	SH-.
\$1B	esc	ESC	\$3B	;	SH-,
\$1C	fs	CT-ö =	\$3C	< = μ	SH->
\$1D	gs	CT-ü = }	\$3D	=	SH-0
\$1E	rs	CT-↑	\$3E	> = °	>
\$1F	us	CT-SH--	\$3F	?	SH-B

Hex.	ASCII	Taste	Hex.	ASCII	Taste
\$40	§ = @	SH- 3	\$60	'	SH- '
\$41	A	SH- a	\$61	a	a
\$42	B	SH- b	\$62	b	b
\$43	C	SH- c	\$63	c	c
\$44	D	SH- d	\$64	d	d
\$45	E	SH- e	\$65	e	e
\$46	F	SH- f	\$66	f	f
\$47	G	SH- g	\$67	g	g
\$48	H	SH- h	\$68	h	h
\$49	I	SH- i	\$69	i	i
\$4A	J	SH- j	\$6A	j	j
\$4B	K	SH- k	\$6B	k	k
\$4C	L	SH- l	\$6C	l	l
\$4D	M	SH- m	\$6D	m	m
\$4E	N	SH- n	\$6E	n	n
\$4F	O	SH- o	\$6F	o	o
\$50	P	SH- p	\$70	p	p
\$51	Q	SH- q	\$71	q	q
\$52	R	SH- r	\$72	r	r
\$53	S	SH- s	\$73	s	s
\$54	T	SH- t	\$74	t	t
\$55	U	SH- u	\$75	u	u
\$56	V	SH- v	\$76	v	v
\$57	W	SH- w	\$77	w	w
\$58	X	SH- x	\$78	x	x
\$59	Y	SH- y	\$79	y	y
\$5A	Z	SH- z	\$7A	z	z
\$5B	Ä= [	SH- ä	\$7B	ä= {	ä
\$5C	Ö= \	SH- ö	\$7C	ö=	ö
\$5D	Ü= ]	SH- ü	\$7D	ü= }	ü
\$5E	^= '	^	\$7E	ß= @	ß
\$5F	-	SH--	\$7F	de l	DELETE

Da es die ASCII-Zeichen in unterschiedlichen Versionen (z.B. US- oder deutsch) gibt und die Tastatur diese Zeichen wiedergibt, kommen manche Zeichen mehrfach vor (z.B. Ä und ß) bzw. unterschiedliche Belegung (z.B. ö und ).

Die Bedeutung der Cursorblocktasten können Sie entsprechend den Eintragungen im Tastenfeld entnehmen. Siehe nächste Seite.

Die Zusatztasten gehen mit Ihren Zeichen, die in dem Tastaturschema eingetragen sind, über den üblichen ASCII-Zeichensatz hinaus. Diese Zeichen sind aber im Vergleich zum normalen ASCII-Zeichensatz um 128 nach oben verschoben, d.h. Bit 7 ist 1 bei den ASCII-Werten dieser Tasten.



## A N H A N G J

### Zusammenstellung der Ein-/Ausgabeadressen

Adresse	Lesen	Schreiben
\$C000	Tastatur	Inverse
\$C001		Flash
\$C002		SW1 aus
\$C003		SW1 ein
\$C004		SW2 aus
\$C005		SW2 ein
\$C006		2 x 128 Zeichen
\$C007		2 x 64 + 128 Zeichen
\$C008	Tastaturerweiterung	Tastaturunterbrechung aus
\$C009		Tastaturunterbrechung ein
\$C00A		40 Zeichen/Zeile
\$C00B		80 Zeichen/Zeile
\$C00C		Statik RAM aus
\$C00D		Statik RAM ein
\$C00E		\$C08x aktiv
\$C00F		\$C08x blockiert
\$C010	Tastaturstrobe	
\$C020	Kassettenausgang	
\$C030	Lautsprecher	
\$C04x	Utility Strobe	Utility Strobe
\$C050	Graphik ein	
\$C051	Graphik aus	
\$C052	Vollgraphik	
\$C053	mixed Graphik	
\$C054	Seite 1 aktiv	
\$C055	Seite 2 aktiv	
\$C056	LO-RES-Graphik	
\$C057	HI-RES-Graphik	
\$C058	TTL-0 low	
\$C059	TTL-0 high	
\$C05A	TTL-1 low	
\$C05B	TTL-1 high	
\$C05C	TTL-2 low	
\$C05D	TTL-2 high	
\$C05E	TTL-3 low	
\$C05F	TTL-3 high	

Adresse	Lesen	Schreiben
\$C060	Kassette Eingang	\$0000 - \$1FFF Bank 0
\$C061	TTL-Eingang 1	\$0000 - \$1FFF Bank 1
\$C062	TTL-Eingang 2	\$2000 - \$3FFF Bank 0
\$C063	TTL-Eingang 3	\$2000 - \$3FFF Bank 1
\$C064	Handregler 0	\$4000 - \$5FFF Bank 0
\$C065	Handregler 1	\$4000 - \$5FFF Bank 1
\$C066	Handregler 2	\$6000 - \$7FFF Bank 0
\$C067	Handregler 3	\$6000 - \$7FFF Bank 1
\$C068		\$8000 - \$9FFF Bank 0
\$C069		\$8000 - \$9FFF Bank 1
\$C06A		\$A000 - \$BFFF Bank 0
\$C06B		\$A000 - \$BFFF Bank 1
\$C06C		\$D000 - \$DFFF Bank 0
\$C06D		\$D000 - \$DFFF Bank 1
\$C06E		\$E000 - \$FFFF Bank 0
\$C06F		\$E000 - \$FFFF Bank 1
\$C070	Handreglerstrobe	
\$C08x	LC-Steuerung	
\$C090		Drucker parallel Ausgang
\$C098	seriell Eingang	seriell Ausgang
\$C099	seriell Status	seriell RESET
\$C09A	seriell Command	seriell Command
\$C09B	seriell Control	seriell Control
\$C0Ax	Slot 2 DEVICE Select	Slot 2 DEVICE Select
.	.	.
.	.	.
.	.	.
\$C0Fx	Slot 7 DEVICE Select	Slot 7 DEVICE Select
\$C100		Z80 ein/aus
\$C1C1	Drucker Acknowledge	.

## A N H A N G   K

### Der Z-80-Teil

#### Einleitung

Der Z-80-Teil beinhaltet die notwendige Hardware, um einen Z-80 Mikroprozessor an den BUS anzupassen. Dadurch ist die direkte Ausführung des 8080 und Z-80 Programms einschließlich des CP/M-Betriebssystems möglich.  
In das System ist die Language Card für das 56k CP/M oder ein anderes Programm, das unter CP/M arbeitet, integriert.

#### Taktgenerierung

Der Z-80 Mikroprozessor ist synchronisiert und mit dem 6502 Takt phasengekoppelt. Während jeder Video Refresh Periode  $\Phi_1$ , wird der 7 Mhz Takt unterteilt, um 3 halbe Perioden von 135 ns zu ermöglichen.

Der erste halbe Takt ist immer höher, der zweite immer niedriger und der dritte wieder hoch.

Nach dem Ende des dritten halben Taktes geht das Signal auf logisch 0 und bleibt dort bis zum Start des nächsten  $\Phi_1$ . Das bedeutet, daß der Z-80 Takt während des Systemtaktes  $\Phi_0$  und einem geringen Teil von  $\Phi_1$  logisch 0 ist. Der vierte Halbtakt ist 563 ns lang. (Diese Zeit wird um 69 ns am Ende eines jeden Videolaufes verlängert). Der effektive Z-80 Takt ist 2.041 MHz.

Jede Art von Maschinentakt beinhaltet eine Speicherzugangszeit  $\Phi_0$ . Das Lese-/Schreibsignal wird durch Synchronisieren der ansteigenden Flanke des Schreibübergangs zum Z-80-Teil-Takt erzeugt und garantiert, daß das Schreiben während dieser Zeit nach logisch 0 geht und der Z-80-Teil nach logisch 1.

Da alle Adressübergänge vom Z-80 ausgehen, wenn deren Takte logisch 1 sind, müssen sie alle während  $\Phi_1$  mit den Videoerneuerungszugriffen erscheinen. Deswegen haben alle  $\Phi_0$  Takte feste Adressen für die ganze Dauer des Taktes.

#### Kontrolle des Z-80-Teiles

Der Z-80-Teil wird durch Schreibkommandos in den Speicherraum, der normalerweise periphere ROMs beinhaltet, kontrolliert. Es ist sehr wichtig, mit Schreibbefehlen zu arbeiten, um sicherzustellen, daß der 6502 nicht 2 Zugriffe hintereinander ausführt (dieses würde ein Zurückschalten auf den 6502 verhindern).

Wenn der BASIS 108 eingeschaltet ist, schaltet das (RESET)-Signal den Z-80-Teil aus. Das (RESET)-Signal ist mit dem internen Takt synchronisiert, um sicherzustellen, daß eine Schreiboperation nicht unterbrochen werden kann. Der Z-80 geht sofort in einen Wartemodus über und bleibt dort bis der Z-80-Teil aktiviert wird.

Nach Empfang eines Schreibbefehles im richtigen Speicherbereich ist der Z-80-Teil eingeschaltet. Der Z-80 bleibt in einem Wartemodus bis ein Speichertakt mit Adressinformationen für den Z-80-Teil erscheint. Jetzt wird der Z-80 vom

Wartemodus befreit und läuft nun ohne weitere Wartetakte.

Mit Empfang eines anderen Schreibbefehles im gleichen Speicherbereich (dieses Mal aus dem Z-80-Teil selbst) wird der Z-80-Teil ausgeschaltet.

Die Speicheradressen für die Kontrolle des Z-80-Teiles sind:

\$C100 - \$C1FF.

### Anpassung des Adress Bus

Der Adress Bus des Z-80-Teiles ist an den BASIS 108 I/O Bus durch eine Adressübersetzung angepaßt. Diese Übersetzung beseitigt die Speicherprobleme, die zwischen der 6502 Architektur und den CP/M- und Z-80-Konventionen bestehen. Diese Logik addiert \$1000 in allen Adressen, wenn er eingeschaltet ist. Der Dip-Schalter S1-1 ist dann aus. Dies verschiebt die Z-80 Interrupt-Adressen und die CP/M Startadressen aus der 0 Bank des 6502-Speichers.

Zusätzlich werden Adressen in den \$C000-\$FFFF-Bereichen verschoben, um dem CP/M angrenzende Speicher zu öffnen.

Die aufgeföhrte Tabelle zeigt genau, wie der Übersetzer funktioniert:

Z-80 Adressen	6502 Adressen
\$0000 - \$0FFF	\$1000 - \$1FFF
\$1000 - \$1FFF	\$2000 - \$2FFF
.	.
.	.
\$A000 - \$AFFF	\$B000 - \$BFFF
\$B000 - \$BFFF	\$D000 - \$DFFF
\$C000 - \$CFFF	\$E000 - \$EFFF
\$D000 - \$DFFF	\$F000 - \$FFFF
\$E000 - \$EFFF	\$C000 - \$CFFF
\$F000 - \$FFFF	\$0000 - \$0FFF.

Der Z-80 kann zusammenhängende Speicher von \$0000-\$DFFF adressieren, ohne die 0 Page des 6502 Prozessors und den I/O Bereich zu stören.

Wenn der Übersetzer ausgeschaltet ist (S1-1 eingeschaltet) erscheinen die gepufferten Z-80 Adressen unverändert auf dem I/O Bus.

Alle Puffer sind immer im hochohmigen Zustand, wenn der Z-80-Teil die Kontrolle über den Bus aufgibt. Die Zeitsteuerung beim Ein- und Ausschalten soll den Z-80-Teil daran hindern, auf den Adressenbus zuzugreifen, wenn andere Elemente die Bus-Kontrolle übernommen haben.

Die Zeitsteuerung des Z-80-Teiles zwingt alle Adressübergänge während der Zeit zu erscheinen, in der der Bildschirm durch den BASIS 108 aufgefrischt wird. Da für jeden Speicherzugriff die Adressen bereits bei Beginn des Zyklus stabil sind, ist kein Wartezyklus erforderlich.

### DMA Daisy Chain

Der DMA Daisy Chain wird so lange durchgeführt, bis eine höher privilegierte DMA Device die Übernahme der Kontrolle des Bus vom Z-80 anfordert. Der eingeschaltete Dip-Schalter S1-2 ermöglicht es dem DMA, den Z-80-Teil zu unterbrechen. Wenn dieser Schalter eingeschaltet ist und die DMA Daisy Chain Leitung (Pin 24) nach 0 geht, wird der laufende Z-80 Maschinenzyklus beendet. Der

Z-80 zeigt die Freigabe des Bus durch die DMA-Leitung an. DMA geht auf logisch 0.

Zu diesem Zeitpunkt kann ein anderes Gerät die Kontrolle übernehmen, indem die DMA-Leitung logisch 0 gesetzt wird. Die Kontrolle darf durch das andere Gerät nicht früher übernommen werden, da bis zu diesem Zeitpunkt der Z-80 den Bus immer noch kontrolliert.

Der Z-80 hat die niedrigste DMA-Priorität.

### Interrupts

Damit sowohl der Z-80 als auch der 6502 Mikroprozessor Interrupts erkennen können, wurde entsprechende Hardware integriert. Wenn der Dip-Schalter S1-4 eingeschaltet ist, erkennt der Z-80 Interrupts. Das Interruptprogramm sollte die Kontrolle an den 6502 für den weiteren Betrieb zurückgeben. So hat der 6502, der auch den Interrupt feststellte, die Möglichkeit sich vom Interruptstatus zu befreien.

Der Z-80 wird im Interruptmode 1 betrieben.

Der Dip-Schalter S1-3 hat die gleichen Funktionen für den nicht maskierbaren Interrupt.

## Z8400 Z80® CPU Central Processing Unit



### Product Specification

Z80 CPU

March 1981

**Features**

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Six MHz, 4 MHz and 2.5 MHz clocks for the Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers together with indexed and relative addressing result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system

may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.

- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high speed interrupt processing: 8080 compatible, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

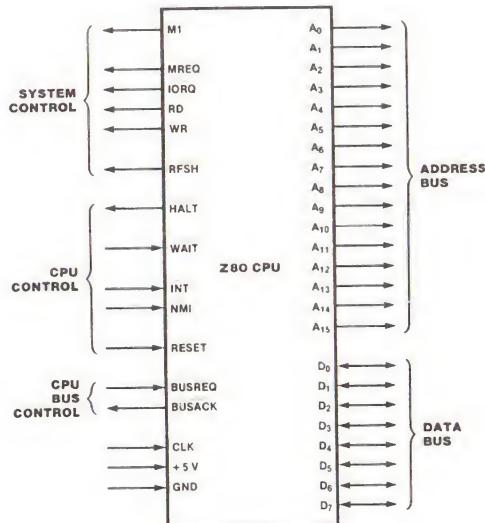


Figure 1. Pin Functions

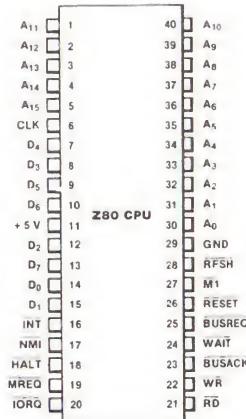


Figure 2. Pin Assignments

### General Description

The Z80, Z80A, and Z80B CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may

be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source, all output signals are fully decoded and timed to control standard memory or peripheral circuits, and is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

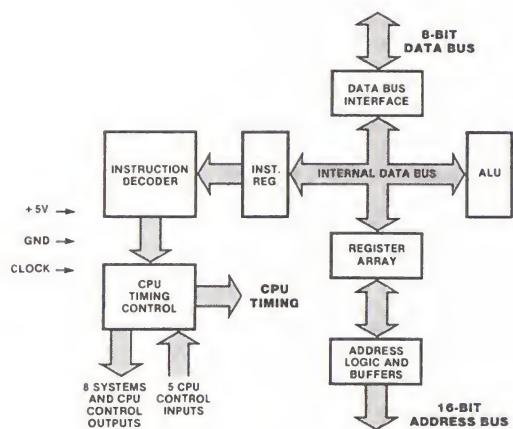


Figure 3. Z80 CPU Block Diagram

## Z80 Micro-processor Family

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers,

## Z80 CPU Registers

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-

each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.

- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Synch and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

**Z80 CPU**

foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

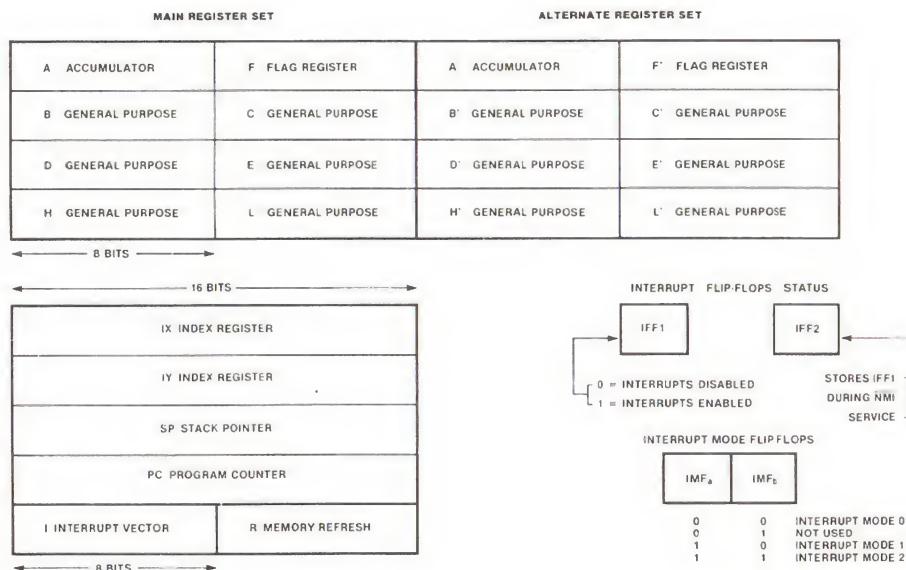


Figure 4. CPU Registers

Z80 CPU Registers (Continued)	Register	Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte    C — Low byte D — High byte    E — Low byte H — High byte    L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Stores addresses or data temporarily. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF <sub>1</sub> -IFF <sub>2</sub>	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80 CPU Registers

**Interrupts:  
General  
Operation**

The CPU accepts two interrupt input signals: NMI and INT. The NMI is a non-maskable interrupt and has the highest priority. INT is a lower priority interrupt since it requires that interrupts be enabled in software in order to operate. Either NMI or INT can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, INT, has three programmable response modes available. These are:

- Mode 0 — compatible with the 8080 microprocessor.

- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the NMI and INT signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

**Interrupts:  
General  
Operation**  
(Continued)

**Non-Maskable Interrupt (NMI).** The non-maskable interrupt cannot be disabled by program control and therefore will be accepted at all times by the CPU. NMI is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the NMI signal (providing BUSREQ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routine.

**Maskable Interrupt (INT).** Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and BUSREQ is not active) a special interrupt processing cycle begins. This is a special fetch (M1) cycle in which IORQ becomes active rather than MREQ, as in a normal M1 cycle. In addition, this special M1 cycle is automatically extended by two WAIT states, to allow for the time required to acknowledge the interrupt request and to place the interrupt vector on the bus.

**Mode 0 Interrupt Operation.** This mode is compatible with the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus, which is then acted on six times by the CPU. This is normally a Restart Instruction, which will initiate an unconditional jump to the selected one of eight restart locations in page zero of memory.

**Mode 1 Interrupt Operation.** Mode 1 operation is very similar to that for the NMI. The principal difference is that the Mode 1 interrupt has a vector address of 0038H only.

**Mode 2 Interrupt Operation.** This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit address vector on the data bus during the interrupt acknowledge cycle. The high-order byte of the interrupt service routine address is supplied by the I (Interrupt) register. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available

location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 ( $A_0$ ) must be a zero.

**Interrupt Priority (Daisy Chaining and Nested Interrupts).**

The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardware to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

**Interrupt Enable/Disable Operation.** Two flip-flops, IFF<sub>1</sub> and IFF<sub>2</sub>, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual*.

Action	IFF <sub>1</sub>	IFF <sub>2</sub>	Comments
CPU Reset	0	0	Maskable interrupt INT disabled
D1 instruction execution	0	0	Maskable interrupt INT disabled
EI instruction execution	1	1	Maskable interrupt INT enabled
LD A,I instruction execution	•	•	IFF <sub>2</sub> → Parity flag
LD A,R instruction execution	•	•	IFF <sub>2</sub> → Parity flag
Accept NMI	0	IFF <sub>1</sub>	IFF <sub>1</sub> → IFF <sub>2</sub> (Maskable interrupt INT disabled)
RETN instruction execution	IFF <sub>2</sub>	•	IFF <sub>2</sub> → IFF <sub>1</sub> at completion of an NMI service routine.

Table 2. State of Flip-Flops

<b>Instruction Set</b>	The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.	<input type="checkbox"/> 16-bit arithmetic operations <input type="checkbox"/> Rotates and shifts <input type="checkbox"/> Bit set, reset, and test operations <input type="checkbox"/> Jumps <input type="checkbox"/> Calls, returns, and restarts <input type="checkbox"/> Input and output operations
	The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The <i>Z80 CPU Technical Manual</i> (03-0029-01) and <i>Assembly Language Programming Manual</i> (03-0002-01) contain significantly more details for programming use.	A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:
	The instructions are divided into the following categories:	<input type="checkbox"/> Immediate <input type="checkbox"/> Immediate extended <input type="checkbox"/> Modified page zero <input type="checkbox"/> Relative <input type="checkbox"/> Extended <input type="checkbox"/> Indexed <input type="checkbox"/> Register <input type="checkbox"/> Register indirect <input type="checkbox"/> Implied <input type="checkbox"/> Bit
	<input type="checkbox"/> 8-bit loads <input type="checkbox"/> 16-bit loads <input type="checkbox"/> Exchanges, block transfers, and searches <input type="checkbox"/> 8-bit arithmetic and logic operations <input type="checkbox"/> General-purpose arithmetic and CPU control	

8-Bit Load Group	Mnemonic	Symbolic Operation	Flags			Opcode 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
			S	Z	H					
	LD r, r'	r ← r'	•	•	X	• X	• • •	01 110	1	1 4
	LD r, n	r ← n	•	•	X	• X	• • •	00 r 110 → n →	2	2 7
	LD r, (HL)	r ← (HL)	•	•	X	• X	• • •	01 r 110	1	2 7
	LD r, (IX+d)	r ← (IX+d)	•	•	X	• X	• • •	11 011 101 DD 01 r 101 → d →	3	5 19
	LD r, (IY+d)	r ← (IY+d)	•	•	X	• X	• • •	11 111 101 FD 01 r 110 → d →	3	5 19
	LD (HL), r	(HL) ← r	•	•	X	• X	• • •	01 110 r 11 011 101 DD 01 110 r → d →	1	2 7
	LD (IX+d), r	(IX+d) ← r	•	•	X	• X	• • •	11 011 101 DD 01 110 r → d →	3	5 19
	LD (IY+d), r	(IY+d) ← r	•	•	X	• X	• • •	11 111 101 FD 01 110 r → d →	3	5 19
	LD (HL), n	(HL) ← n	•	•	X	• X	• • •	00 110 110 36 → n →	2	3 10
	LD (IX+d), n	(IX+d) ← n	•	•	X	• X	• • •	11 011 101 DD 00 110 110 36 → d →	4	5 19
	LD (IY+d), n	(IY+d) ← n	•	•	X	• X	• • •	11 111 101 FD 00 110 110 36 → d → → n →	4	5 19
	LD A, (BC)	A ← (BC)	•	•	X	• X	• • •	00 001 010 0A → n →	1	2 7
	LD A, (DE)	A ← (DE)	•	•	X	• X	• • •	00 011 010 1A → n →	1	2 7
	LD A, (nn)	A ← (nn)	•	•	X	• X	• • •	00 111 010 3A → n →	3	4 13
	LD (BC), A	(BC) ← A	•	•	X	• X	• • •	00 000 010 02 → n →	1	2 7
	LD (DE), A	(DE) ← A	•	•	X	• X	• • •	00 010 010 12 → n →	1	2 7
	LD (nn), A	(nn) ← A	•	•	X	• X	• • •	00 110 010 32 → n →	3	4 13
	LD A, I	A ← I	I	I	X	0 X	IFF 0	• 11 101 101 ED 01 010 111 57	2	2 9
	LD A, R	A ← R	I	I	X	0 X	IFF 0	• 11 101 101 ED 01 011 111 5F	2	2 9
	LD I, A	I ← A	•	•	X	• X	• • •	11 101 101 ED 01 000 111 47	2	2 9
	LD R, A	R ← A	•	•	X	• X	• • •	11 101 101 ED 01 001 111 4F	2	2 9

NOTES: r, r' means any of the registers A, B, C, D, E, H, L.  
IFF the content of the interrupt enable flag (Intp., IFF) is copied into the P/V flag.  
For an explanation of flag notation and symbols for mnemonics tables, see Symbolic Notation section, following tables.

16-Bit Load Group	Mnemonic	Symbolic Operation	S	Z	Flags H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments		
	LD dd, nn	dd ← nn	*	*	X	*	X	*	*	00 dd0 001	3	3	10	dd	Pair
					→ n →								00	BC	
	LD IX, nn	IX ← nn	*	*	X	*	X	*	*	11 011 101 DD 00 100 001 21	4	4	14	01	DE
					→ n →								10	HL	
	LD IY, nn	IY ← nn	*	*	X	*	X	*	*	11 111 101 FD 00 100 001 21	4	4	14	11	SP
					→ n →										
	LD HL, (nn)	H ← (nn+1) L ← (nn)	*	*	X	*	X	*	*	00 101 010 2A	3	5	16		
					→ n →										
	LD dd, (nn)	ddH ← (nn+1) ddL ← (nn)	*	*	X	*	X	*	*	11 101 101 ED 01 dd0 011	4	6	20		
					→ n →										
	LD IX, (nn)	IXH ← (nn+1) IXL ← (nn)	*	*	X	*	X	*	*	11 011 101 DD 00 101 010 2A	4	6	20		
					→ n →										
	LD IY, (nn)	IYH ← (nn+1) IYL ← (nn)	*	*	X	*	X	*	*	11 111 101 FD 00 101 010 2A	4	6	20		
					→ n →										
	LD (nn), HL	(nn+1) ← H (nn) ← L	*	*	X	*	X	*	*	00 100 010 22	3	5	16		
					→ n →										
	LD (nn), dd	(nn+1) ← ddH (nn) ← ddL	*	*	X	*	X	*	*	11 101 101 ED 01 dd0 011	4	6	20		
					→ n →										
	LD (nn), IX	(nn+1) ← IXH (nn) ← IXL	*	*	X	*	X	*	*	11 011 101 DD 00 100 010 22	4	6	20		
					→ n →										
	LD (nn), IY	(nn+1) ← IYH (nn) ← IYL	*	*	X	*	X	*	*	11 111 101 FD 00 100 010 22	4	6	20		
					→ n →										
	LD SP, HL	SP ← HL	*	*	X	*	X	*	*	11 111 001 F9	1	1	6		
	LD SP, IX	SP ← IX	*	*	X	*	X	*	*	11 011 101 DD 11 111 001 F9	2	2	10		
	LD SP, IY	SP ← IY	*	*	X	*	X	*	*	11 111 101 FD 11 111 001 F9	2	2	10		
	PUSH qq	(SP-2) ← qqL (SP-1) ← qqH	*	*	X	*	X	*	*	11 qq0 101	1	3	11	qq	Pair
					→ n →								00	BC	
	PUSH IX	(SP-2) ← IXL (SP-1) ← IXH	*	*	X	*	X	*	*	11 011 101 DD 11 100 101 E5	2	4	15	10	HL
					→ n →								11	AF	
	PUSH IY	(SP-2) ← IYL (SP-1) ← IYH	*	*	X	*	X	*	*	11 111 101 FD 11 100 101 E5	2	4	15		
					→ n →										
	POP qq	qqH ← (SP+1) qqL ← (SP)	*	*	X	*	X	*	*	11 qq0 001	1	3	10		
					→ n →										
	POP IX	IXH ← (SP+1) IXL ← (SP)	*	*	X	*	X	*	*	11 011 101 DD 11 100 001 E1	2	4	14		
					→ n →										
	POP IY	IYH ← (SP+1) IYL ← (SP)	*	*	X	*	X	*	*	11 111 101 FD 11 100 001 E1	2	4	14		
					→ n →										

NOTE: dd is any of the register pairs BC, DE, HL, SP.  
qq is any of the register pairs AF, BC, DE, HL.  
(AFH, L) refer to high order and low order eight bits of the register pair respectively.  
e.g. BC = C AFH = A

Exchange, Block Transfer, Block Search Groups	EX DE, HL	DE ← HL	*	*	X	*	X	*	*	11 101 011 EB	1	1	4		
		DE ← AF	*	*	X	*	X	*	*	00 001 000 08	1	1	4		
	EXX	BC ← BC'	*	*	X	*	X	*	*	11 011 001 D9	1	1	4	Register bank and auxiliary register bank exchange	
		DE ← DE'													
		HL ← HL'													
	EX (SP), HL	H ← (SP+1) L ← (SP)	*	*	X	*	X	*	*	11 100 011 E3	1	5	19		
	EX (SP), IX	IXH ← (SP+1) IXL ← (SP)	*	*	X	*	X	*	*	11 011 101 DD 11 100 011 E3	2	6	23		
	EX (SP), IY	IYH ← (SP+1) IYL ← (SP)	*	*	X	*	X	*	*	11 111 101 FD 11 100 011 E3	2	6	23		
					①										
	LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	*	*	X	0	X	1	0	11 101 101 ED 10 100 000 A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)	
	LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Repeat until BC = 0	*	*	X	0	X	0	0	11 101 101 ED 10 110 000 B0	2	5	21	If BC ≠ 0	
													4	16	If BC = 0

NOTE: (P) V flag is set if the result of BC=1 = 0; otherwise P V = 1.

Exchange, Block Transfer, Block Search Groups (Continued)	Mnemonic	Symbolic Operation	S	Z	Flags	H	P/V	N	C	Opcode	76 543 210 Hex	No. of Bytes	No. of M	No. of T	Comments
LDD		DE ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1	•	•	X	0	X	1	•	11 101 101 ED	2	4	16		
LDDR		(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1 Repeat until BC = 0	•	•	X	0	X	0	•	11 101 101 ED 10 111 000 BB	2	5	21	If BC ≠ 0 If BC = 0	
CPI		A ← (HL) HL ← HL + 1 BC ← BC - 1	1	1	X	1	X	1	1	11 101 101 ED 10 100 001 A1	2	4	16		
CPIR		A ← (HL) HL ← HL + 1 BC ← BC - 1 Repeat until A = (HL) or BC = 0	1	1	X	1	X	1	1	11 101 001 ED 10 111 001 B1	2	5	21	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)	
CPD		A ← (HL) HL ← HL + 1 BC ← BC - 1	1	1	X	1	X	1	1	11 101 101 ED 10 101 001 A9	2	4	16		
CPDR		A ← (HL) HL ← HL + 1 BC ← BC - 1 Repeat until A = (HL) or BC = 0	1	1	X	1	X	1	1	11 101 101 ED 10 111 001 B9	2	5	21	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)	

NOTES: ① P/V flag is 0 if the result of BC = 0, otherwise P/V = 1  
 ② Z flag is 1 if A = HL, otherwise Z = 0

8-Bit Arithmetic and Logical Group															
ADD A, r	A ← A + r	1	1	X	1	X	V	0	1	10 000 r	1	1	4	r	Req
ADD A, n	A ← A + n	1	1	X	1	X	V	0	1	11 000 110	2	2	7	000 B	
										• n →				001 C	
														010 D	
ADD A, (HL)	A ← A + (HL)	1	1	X	1	X	V	0	1	10 000 110	1	2	7	011 E	
ADD A, (IX+d)	A ← A + (IX+d)	1	1	X	1	X	V	0	1	11 011 101 DD	3	5	19	100 H	
										10 000 110				101 L	
										• d →				111 A	
ADD A, (IY+d)	A ← A + (IY+d)	1	1	X	1	X	V	0	1	11 111 101 FD	3	5	19	10 000 110	
										• d →				s is any of r, n (HL), (IX+d) (IY+d) as shown for ADD instruction The indicated bits replace the 000 in the ADD set above	
ADC A, s	A ← A + s + CY	1	1	X	1	X	V	0	1	00 000					
SUB s	A ← A - s	1	1	X	1	X	V	1	1	01 000					
SBC A, s	A ← A - s - CY	1	1	X	1	X	V	1	1	01 001					
AND s	A ← A & s	1	1	X	1	X	P	0	0	10 000					
OR s	A ← A ∨ s	1	1	X	0	X	P	0	0	10 001					
XOR s	A ← A ⊕ s	1	1	X	0	X	P	0	0	10 010					
CP s	A - s	1	1	X	1	X	V	1	1	10 100					
INC r	r ← r + 1	1	1	X	1	X	V	0	•	00 r 000	1	1	4		
INC (HL)	(HL) ← (HL) + 1	1	1	X	1	X	V	0	•	00 110 000	1	3	11		
INC (IX+d)	(IX+d) ← (IX+d) + 1	1	1	X	1	X	V	0	•	11 011 101 DD	3	6	23	00 110 000	
										• d →					
INC (IY+d)	(IY+d) ← (IY+d) + 1	1	1	X	1	X	V	0	•	11 111 101 FD	3	6	23	00 110 000	
										• d →					
DEC m	m ← m - 1	1	1	X	1	X	V	1	1	10 100				m is any of r, (HL), (IX+d), (IY+d) as shown for INC DEC same format and states as INC Replace 100 with 101 in opcode	

General-Purpose Arithmetic and CPU Control Groups	Mnemonic	Symbolic Operation	Flags			Opcode 76 543 210 Hex	No. of Bytes	No. of Cycles	No. of M States	Comments
			S	Z	H					
	DAA	Converts acc content into packed BCD following add or subtract with packed BCD operands	1	1	X	1	X	P	0	1
	CPL	A = ~A	•	•	X	1	X	•	1	00 101 111 2F
	NEG	A = -A	1	1	X	1	X	V	1	11 101 101 ED 01 000 100 44
	CCF	CY = ~CY	•	•	X	X	X	•	0	1
	SCF	CY = 1	•	•	X	0	X	•	0	00 110 111 37 00 000 000 00
	NOF	No operation	•	•	X	•	X	•	•	00 000 000 00
	HALT	CPU halted	•	•	X	•	X	•	•	01 110 110 76
	DI *	IFP = 0	•	•	X	•	X	•	•	11 110 011 F3
	EI *	IFP = 1	•	•	X	•	X	•	•	12 111 011 FB
	IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	11 101 101 ED 01 000 110 46
	IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	11 101 101 ED 01 010 110 56
	IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	11 101 101 ED 01 011 110 5F

NOTE: IFP indicates the interrupt enable flag.  
 \* These instructions are not guaranteed to work in the event of EI or DI.

16-Bit Arithmetic Group	Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of Cycles	No. of M States	Comments
	ADD HL ss	HL = HL + ss	•	•	X	X	X	•	0	1	3	11	ss Req 00 BC 01 DE 10 HL 11 SP
	ADC HL ss	HL = HL + ss + CY	1	1	X	X	X	V	0	1	4	15	
	SBC HL ss	HL = HL - ss - CY	1	1	X	X	X	V	1	1	4	15	
	ADD IX pp	IX = IX + pp	•	•	X	X	X	•	0	1	4	15	pp Req 00 BC 01 DE 10 IX 11 SP
	ADD IY rr	IY = IY + rr	•	•	X	X	X	•	0	1	4	15	rr Req 00 BC 01 DE 10 IY 11 SP
	INC ss	ss = ss + 1	•	•	X	•	X	•	•	00 ss0 01	1	1	6
	INC IX	IX = IX + 1	•	•	X	•	X	•	•	11 011 101 00	2	2	10
	INC IY	IY = IY + 1	•	•	X	•	X	•	•	11 111 101 ED 00 110 011 23	2	2	10
	DEC ss	ss = ss - 1	•	•	X	•	X	•	•	00 ss0 01	1	1	6
	DEC IX	IX = IX - 1	•	•	X	•	X	•	•	11 011 101 DD 00 011 011 2B	2	2	10
	DEC IY	IY = IY - 1	•	•	X	•	X	•	•	11 111 101 FD 00 101 011 2B	2	2	10

Note: If CY = 1, then the result is 1 less than the true sum or difference.

Rotate and Shift Group	Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of Cycles	No. of M States	Comments
	RLC A	$\boxed{CY \leftarrow [7 \rightarrow 0] \rightarrow A}$	•	•	X	•	X	•	0	1	1	4	Rotate left circular accumulator
	RLC	$\boxed{\{CY\} \leftarrow \{[7 \rightarrow 0]\} \rightarrow A}$	•	•	X	0	X	•	0	1	1	4	Rotate left accumulator
	RRC A	$\boxed{[7 \rightarrow 0] \leftarrow \{CY\} \rightarrow A}$	•	•	X	•	X	•	1	1	1	4	Rotate right circular accumulator
	RRC	$\boxed{\{[7 \rightarrow 0]\} \leftarrow \{CY\} \rightarrow A}$	•	•	X	•	X	•	1	1	1	4	Rotate right accumulator
	RLC r	$r = r \ll 1$	1	1	X	•	X	F	0	1	2	6	Rotate left register
	RLC HL	$HL = HL \ll 1$	1	1	X	•	X	F	0	1	4	6	Register HL
	RLC IX+d	$IX+d = IX+d \ll 1$	1	1	X	•	X	F	1	4	6	IX+d	
	RLC IY+d	$IY+d = IY+d \ll 1$	1	1	X	•	X	F	1	4	6	IY+d	
	RL m	$m = r (HL \ll (IX+d) \ll d)$	1	1	X	0	X	P	0	1	4	6	Instruction format and states do not write PC's. To form new code replace 0000 by shown code.
	RRC m	$m = r (HL \ll (IX+d) \ll d)$	1	1	X	0	X	P	0	1	4	6	With shown code

Rotate and Shift Group (Continued)	Mnemonic	Symbolic Operation	S	Z	Flags	H	P/V	N	C	Opcode	76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
	RR m		t	t	X	0	X	P	0	t		E11				
	SLA m		t	t	X	0	X	P	0	t		E03				
	SRA m		t	t	X	0	X	P	0	t		E01				
	SRL m		0	t	X	0	X	P	0	t		E11				
	RLD		t	t	X	0	X	P	0	t	11 101 101 ED	ED	2	5	18	Rotate digit left and right between the accumulator and location (HL)
	RRD		t	t	X	0	X	P	0	t	11 101 101 ED	ED	2	5	18	The content of the upper half of the accumulator is unaffected.
<b>Bit Set, Reset and Test Group</b>	BIT b, r	Z = r <sub>b</sub>	X	t	X	1	X	X	0	t	11 001 011 CB	2	2	8	r <sub>b</sub> Reg.	
	BIT b, (HL)	Z = (HL) <sub>b</sub>	X	t	X	1	X	X	0	t	11 001 011 CB	2	3	12	000 B	
	BIT b, (IX+d) <sub>b</sub>	Z = (IX+d) <sub>b</sub>	X	t	X	1	X	X	0	t	11 011 101 DD	4	5	20	001 C	
											11 001 011 CB				010 D	
											— d —				011 E	
											01 110				100 H	
											01 110				101 L	
											01 110				111 A	
															b Bit Tested	
	BIT b, (IY+d) <sub>b</sub>	Z = (IY+d) <sub>b</sub>	X	t	X	1	X	X	0	t	11 111 101 FD	4	5	20	000 0	
											11 001 011 CB				001 1	
											— d —				010 2	
											01 110				011 3	
															100 4	
															101 5	
															110 6	
															111 7	
	SET b, r	r <sub>b</sub> = 1	*	*	X	*	X	*	*	*	11 001 011 CB	2	2	8		
	SET b, (HL)	(HL) <sub>b</sub> = 1	*	*	X	*	X	*	*	*	11 001 011 CB	2	4	15		
	SET b, (IX+d) <sub>b</sub>	(IX+d) <sub>b</sub> = 1	*	*	X	*	X	*	*	*	11 011 101 DD	4	6	23		
	SET b, (IY+d) <sub>b</sub>	(IY+d) <sub>b</sub> = 1	*	*	X	*	X	*	*	*	11 111 101 FD	4	6	23		
	RES b, m	m <sub>b</sub> = 0 m = r (HL), (IX+d), (IY+d)	*	*	X	*	X	*	*	*	10				To form new opcode replace	
															Flags and time states for SET instruction.	

Notes: The notation nn indicates that the value of nn is determined by the condition.

Jump Group	JP nn	PC = nn	*	*	X	*	X	*	*	*	11 000 011 C3	3	3	10	
	JP cc, nn	If condition cc is true PC = nn, otherwise continue	*	*	X	*	X	*	*	*	11 cc 010	3	3	10	cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
	JR e	PC = PC + e	*	*	X	*	X	*	*	*	00 011 000 18	2	3	12	
	JR C, e	If C = 0, continue If C = 1, PC = PC + e	*	*	X	*	X	*	*	*	00 111 000 38	2	2	7	If condition not met
	JR NC, e	If C = 1, continue If C = 0, PC = PC + e	*	*	X	*	X	*	*	*	00 110 000 30	2	2	7	If condition is met
	JR Z, e	If Z = 0, continue If Z = 1, PC = PC + e	*	*	X	*	X	*	*	*	00 101 000 28	2	2	7	If condition not met
	JR NZ, e	If Z = 1, continue If Z = 0, PC = PC + e	*	*	X	*	X	*	*	*	00 100 000 20	2	2	7	If condition not met
	JP (HL)	PC = PC + e PC = HL	*	*	X	*	X	*	*	*	11 101 001 E9	1	1	4	
	JP (IX)	PC = IX	*	*	X	*	X	*	*	*	11 011 101 DD	2	2	8	

Jump Group (Continued)	Mnemonic	Symbolic Operation	S	Z	Flags	H	P/V	N	C	Opcode	76 543 210 Hex	No. of Bytes	No. of Cycles	No. of T States	Comments
	JP (IY)	PC = IY	•	•	X	•	X	•	•	11 111 101 FD		2	2	8	
	DJNZ, e	B = B - 1 If B = 0, continue If B ≠ 0, PC = PC + e	•	•	X	•	X	•	•	00 010 000 10		2	2	8	If B = 0.

NOTES e represents the extension in the relative addressing mode.  
e is a signed two's complement number in the range < -128, 129 >  
e - 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Call and Return Group	CALL nn	(SP - 1) - PC <sub>H</sub> (SP - 2) - PC <sub>L</sub> PC - nn	•	•	X	•	X	•	•	11 001 101 CD		3	5	17	
	CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	•	X	•	•	11 cc 100		3	3	10	If cc is false.
	RET	PC <sub>L</sub> = (SP) PC <sub>H</sub> = (SP + 1)	•	•	X	•	X	•	•	11 001 001 C9		1	3	10	
	RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	•	X	•	•	11 cc 000		1	1	5	If cc is true.
	RETI	Return from interrupt	•	•	X	•	X	•	•	11 101 101 ED		2	4	14	
	RETN <sup>1</sup>	Return from non maskable interrupt	•	•	X	•	X	•	•	11 101 101 ED	01 000 101 45	2	4	14	
	RST p	(SP - 1) - PC <sub>H</sub> (SP - 2) - PC <sub>L</sub> PC <sub>H</sub> = 0 PC <sub>L</sub> = p	•	•	X	•	X	•	•	11 t 111		1	3	11	t = 000 00H 001 08H 010 10H 011 18H 100 20H 101 28H 110 30H 111 38H

NOTE <sup>1</sup>RETN loads IFF<sub>2</sub> = IFF<sub>1</sub>

Input and Output Group	IN A, (n)	A - (n)	•	•	X	•	X	•	•	11 011 011 DB		2	3	11	n to A <sub>0</sub> ~ A <sub>7</sub> Acc to A <sub>8</sub> ~ A <sub>15</sub> C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	IN r, (C)	r - (C)	1	1	X	1	X	P	O	11 101 101 ED	01 r 000	2	3	12	
	INI	(HL) - (C) B = B - 1 HL = HL + 1	X	1	X	X	X	X	1	11 101 101 ED	10 100 010 A2	2	4	16	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	INIR	(HL) - (C) B = B - 1 HL = HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	11 101 101 ED	10 110 010 B2	2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	IND	(HL) - (C) B = B - 1 HL = HL - 1	X	1	X	X	X	X	1	11 101 101 ED	10 101 010 AA	2	4	16	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	INDR	(HL) - (C) B = B - 1 HL = HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	11 101 101 ED	10 111 010 BA	2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	OUT (n), A	(n) - A	•	•	X	•	X	•	•	11 010 011 D3		2	3	11	n to A <sub>0</sub> ~ A <sub>7</sub> Acc to A <sub>8</sub> ~ A <sub>15</sub> C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	OUT (C), r	(C) - r	•	•	X	•	X	•	•	11 101 101 ED	01 r 001	2	3	12	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	OUTI	(C) - (HL) B = B - 1 HL = HL + 1	X	1	X	X	X	X	1	11 101 101 ED	10 100 011 A3	2	4	16	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	OTIR	(C) - (HL) B = B - 1 HL = HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	11 101 101 ED	10 110 011 B3	2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>
	OUTD	(C) - (HL) B = B - 1 HL = HL - 1	X	1	X	X	X	X	1	11 101 101 ED	10 101 011 AB	2	4	16	C to A <sub>0</sub> ~ A <sub>7</sub> B to A <sub>8</sub> ~ A <sub>15</sub>

NOTE ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset

Input and Output Group (Continued)	Mnemonic	Symbolic Operation	Flags			Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
			S	Z	H					
	OTDR	(C) ← (HL) B ← B ↓ HL ← HL ↓ Repeat until B = 0	X	1	X	X	X	1	•	101 101 ED 1e 111 011
									2	If B ≠ 0; If B = 0;
								4	16	B to A8 – A15

Summary of Flag Operation	Instruction	D7			D0	Comments				
		S	Z	H						
	ADD A, s. ADC A, s	1	1	X	1	X	V	0	1	8 bit add or add with carry
	SUB s. SBC A, s. CP s. NEG	1	1	X	1	X	V	1	1	8 bit subtract with carry, compare and negate accumulator
	AND s	1	1	X	1	X	P	0	0	
	OR s. XOR s	1	1	X	0	X	P	0	0	Logical operations
	INC s	1	1	X	1	X	V	0	•	8 bit increment
	DEC s	1	1	X	1	X	V	1	•	8 bit decrement
	ADD DD, ss	•	•	X	X	X	•	0	1	8 bit add
	ADC HL, ss	1	1	X	X	X	V	0	1	16 bit add with carry
	SBC HL, ss	1	1	X	X	X	V	1	1	16 bit subtract with carry
	RLA, RLCA, RRA, RRCA	•	•	X	0	X	•	0	1	Rotate accumulator
	RL m, RLX m, RH m,	1	1	X	0	X	P	0	1	Rotate and shift locations
	RRC m, SLA m,									
	SRA m, SKL m									
	RLD, RRD	1	1	X	0	X	P	•	•	Rotate digit left and right
	DAA	1	1	X	1	X	P	•	1	Decimal adjust accumulator
	CPL	•	•	X	1	X	•	•	•	Complement accumulator
	SCF	•	•	X	0	X	•	•	1	Set carry
	CCF	•	•	X	0	X	•	0	1	Complement carry
	IN r (C)	1	1	X	0	X	P	0	1	Input required indicator
	INI, IND, OUTI, OUTD	X	1	X	X	X	X	1	•	
	INIR, INDR, OTIF, OTDR	X	1	X	X	X	X	1	•	
	LDI, LDD	X	X	X	1	X	I	C	•	
	LDIR, LDDR	X	X	X	1	X	G	0	•	
	CPI, CPIR, CPD, CPDR	X	1	Y	X	X	I	C	•	
	LDA, LLD, A, R	1	1	X	1	X	IFF	0	•	
	BIT b, s	X	1	X	1	X	X	0	•	The content of the interrupt enable flip flop, IFF, is copied into the P/V flag. The state of bit b of memory is copied into the Z flag.

Symbolic Notation	Symbol	Operation	Symbol	Operation
S	S	Sign flag. S = 1 if the MSB of the result is 1.	†	The flag is affected according to the result of the operation.
Z	Z	Zero flag. Z = 1 if the result of the operation is 0.	•	The flag is unchanged by the operation.
P/V	P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.	0	The flag is reset by the operation.
H	H	Halt carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.	1	The flag is set by the operation.
N	N	Add/Subtract flag. N = 1 if the previous operation was a subtract.	X	The flag is a "don't care."
H & N	H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.	V	P/V flag affected according to the overflow result of the operation.
C	C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.	P	P/V flag affected according to the parity result of the operation.
	r		r	Any one of the CPU registers A, B, C, D, E, H, L.
	s		s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
	ss		ss	Any 16-bit location for all the addressing modes allowed for that instruction.
	ii		ii	Any one of the two index registers IX or IY.
	R		R	Refresh counter.
	n		n	8-bit value in range < 0, 255 >.
	nn		nn	16-bit value in range < 0, 65535 >.

Pin Descriptions	
<b>A<sub>0</sub>-A<sub>15</sub>.</b> Address Bus (output, active High, 3-state). A <sub>0</sub> -A <sub>15</sub> form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.	placed on the data bus.
<b>BUSACK.</b> Bus Acknowledge (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals MREQ, IORQ, RD, and WR have entered their high-impedance states. The external circuitry can now control these lines.	<b>M<sub>1</sub>.</b> Machine Cycle One (output, active Low). M <sub>1</sub> , together with MREQ, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. M <sub>1</sub> , together with IORQ, indicates an interrupt acknowledge cycle.
<b>BUSREQ.</b> Bus Request (input, active Low). Bus Request has a higher priority than NMI and is always recognized at the end of the current machine cycle. BUSREQ forces the CPU address bus, data bus, and control signals MREQ, IORQ, RD, and WR to go to a high-impedance state so that other devices can control these lines. BUSREQ is normally wire-ORed and requires an external pullup for these applications. Extended BUSREQ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.	<b>MREQ.</b> Memory Request (output, active Low, 3-state). MREQ indicates that the address bus holds a valid address for a memory read or memory write operation.
<b>D<sub>0</sub>-D<sub>7</sub>.</b> Data Bus (input/output, active High, 3-state). D <sub>0</sub> -D <sub>7</sub> constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.	<b>NMI.</b> Non-Maskable Interrupt (input, active Low). NMI has a higher priority than INT. NMI is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.
<b>HALT.</b> Halt State (output, active Low). HALT indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.	<b>RD.</b> Memory Read (output, active Low, 3-state). RD indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.
<b>INT.</b> Interrupt Request (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. INT is normally wire-ORed and requires an external pullup for these applications.	<b>RESET.</b> Reset (input, active Low). RESET initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that RESET must be active for a minimum of three full clock cycles before the reset operation is complete.
<b>IORQ.</b> Input/Output Request (output, active Low, 3-state). IORQ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. IORQ is also generated concurrently with M <sub>1</sub> during an interrupt acknowledge cycle to indicate that an interrupt response vector can be	<b>RFSH.</b> Refresh (output, active Low). RFSH, together with MREQ, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.
	<b>WAIT.</b> Wait (input, active Low). WAIT indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended WAIT periods can prevent the CPU from refreshing dynamic memory properly.
	<b>WR.</b> Memory Write (output, active Low, 3-state). WR indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.



## 8-Bit Microprocessor Family

### SY6500

### MICROPROCESSOR PRODUCTS

- Single 5 V  $\pm 5\%$  power supply
- N channel, silicon gate, depletion load technology
- Eight bit parallel processing
- 56 Instructions
- Decimal and binary arithmetic
- Thirteen addressing modes
- True indexing capability
- Programmable stack pointer
- Variable length stack
- Interrupt capability
- Non-maskable interrupt
- Use with any type or speed memory
- Bi-directional Data Bus

- Instruction decoding and control
- Addressable memory range of up to 65 K bytes
- "Ready" input
- Direct memory access capability
- Bus compatible with MC6800
- Choice of external or on-board clocks
- 1 MHz, 2 MHz, 3 MHz and 4 MHz operation
- On-chip clock options
  - \* External single clock input
  - \* Crystal time base input
- 40 and 28 pin package versions
- Pipeline architecture

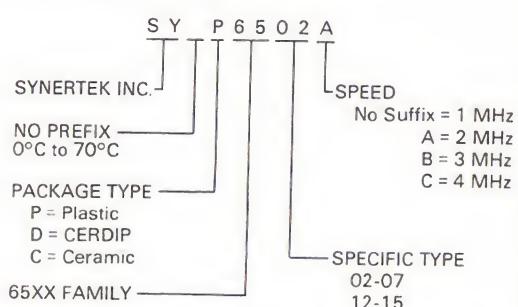
The SY6500 Series Microprocessors represent the first totally software compatible microprocessor family. This family of products includes a range of software compatible microprocessors which provide a selection of addressable memory range, interrupt input options and on-chip clock oscillators and drivers. All of the microprocessors in the SY6500 family are software compatible within the group and are bus compatible with the MC6800 product offering.

The family includes six microprocessors with on-board clock oscillators and drivers and four microprocessors driven by external clocks. The on-chip clock versions are aimed at high performance, low cost applications where single phase inputs or crystals provide the time base. The external clock versions are geared for the multi-processor system applications where maximum timing control is mandatory. All versions of the microprocessors are available in 1 MHz, 2 MHz, 3 MHz and 4 MHz maximum operating frequencies.

#### MEMBERS OF THE FAMILY

PART NUMBERS	CLOCKS	PINS	IRQ	NMI	RYD	ADDRESSING
SY6502	On-Chip	40	✓	✓	✓	64 K
"		28	✓	✓		4 K
"		28	✓			8 K
"		28	✓		✓	4 K
SY6506	"	28	✓			4 K
SY6507	"	28			✓	8 K
SY6512	External	40	✓	✓	✓	64 K
SY6513	"	28	✓	✓		4 K
SY6514	"	28	✓			8 K
SY6515	"	28	✓		✓	4 K

#### ORDERING INFORMATION

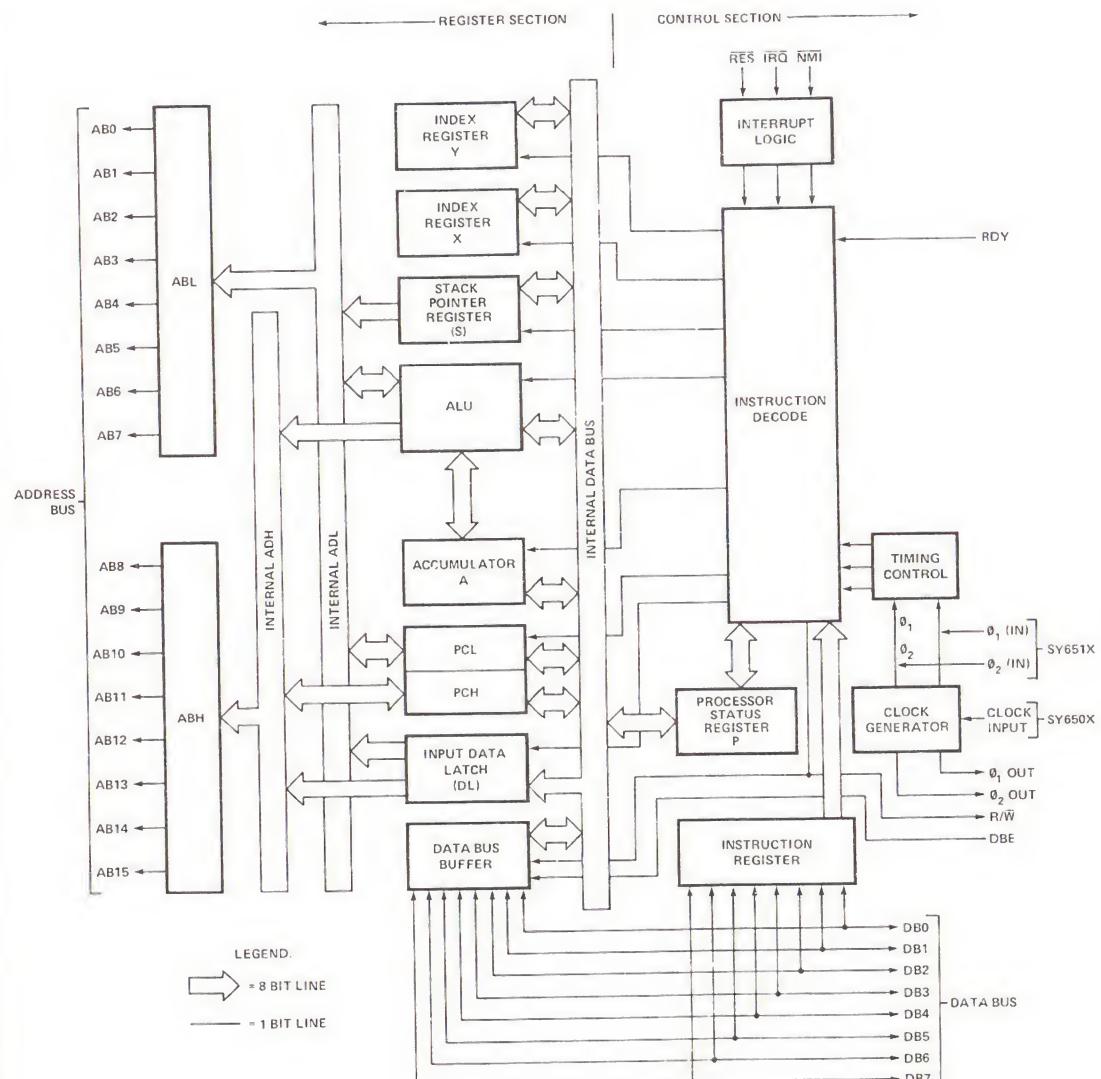


Only 6502 and 6512 are available in 3 and 4 MHz

## COMMENTS ON THE DATA SHEET

The data sheet is constructed to review first the basic "Common Characteristics" — those features which are common to the general family of microprocessors. Subsequent to a review of the family characteristics will be sections devoted to each member of the group with specific features of each.

## SY6500 INTERNAL ARCHITECTURE



NOTE  
 1. CLOCK GENERATOR IS NOT INCLUDED ON SY651X  
 2. ADDRESSING CAPABILITY AND CONTROL OPTIONS VARY WITH EACH OF THE SY6500 PRODUCTS.



## MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 to +7.0	V
Input Voltage	$V_{in}$	-0.3 to +7.0	V
Operating Temperature	$T_A$	0 to +70	°C
Storage Temperature	$T_{STG}$	-55 to +150	°C

## COMMENT

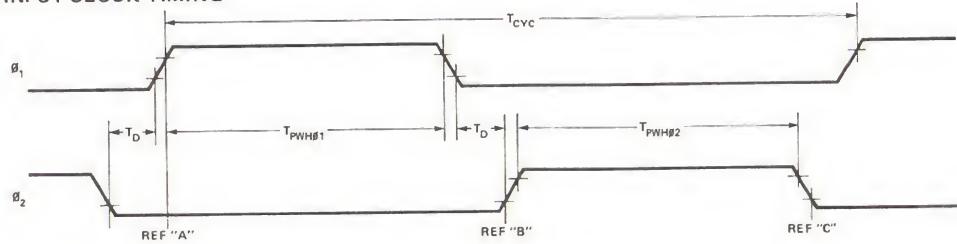
This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

D.C. CHARACTERISTICS ( $V_{CC} = 5.0V \pm 5\%$ ,  $T_A = 0-70^\circ C$ )

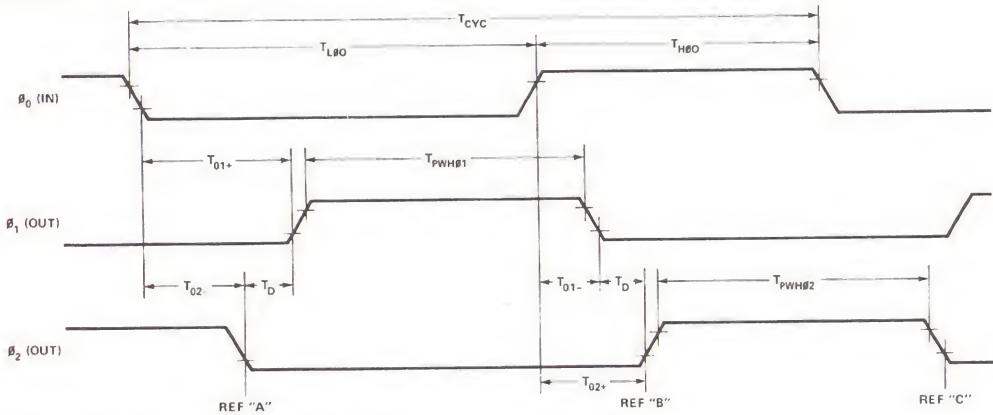
( $\theta_1, \theta_2$  applies to SY651X,  $\theta_o(\text{in})$  applies to SY650X)

Symbol	Characteristic	Min.	Max.	Unit
$V_{IH}$	Input High Voltage Logic and $\theta_o(\text{in})$ for all 650X devices }      } 1,2,3 MHz } 4 MHz	+2.4 +3.3	$V_{CC}$ $V_{CC}$	V V
	$\theta_1$ and $\theta_2$ only for all 651X devices. Logic as 650X }	All Speeds	$V_{CC} - 0.5$	$V_{CC} + 0.25$
$V_{IL}$	Input Low Voltage Logic, $\theta_o(\text{in})$ (650X) $\theta_1, \theta_2$ (651X)	-0.3 -0.3	+0.4 +0.2	V
$I_{IL}$	Input Loading ( $V_{in} = 0 V$ , $V_{CC} = 5.25 V$ ) RDY, S.O.	-10	-300	$\mu A$
$I_{in}$	Input Leakage Current ( $V_{in} = 0$ to $5.25 V$ , $V_{CC} = 0$ ) Logic (Excl. RDY, S.O.) $\theta_1, \theta_2$ (651X) $\theta_o(\text{in})$ (650X)	- -- -	2.5 100 10.0	$\mu A$ $\mu A$ $\mu A$
$I_{TSI}$	Three-State (Off State) Input Current ( $V_{in} = 0.4$ to $2.4 V$ , $V_{CC} = 5.25 V$ ) DB0-DB7	-	$\pm 10$	$\mu A$
$V_{OH}$	Output High Voltage ( $I_{LOAD} = -100\mu A$ , $V_{CC} = 4.75 V$ ) SYNC, DB0-DB7, A0-A15, R/W	1,2,3 MHz 4 MHz	2.4 2.0	V V
$V_{OL}$	Output Low Voltage ( $I_{LOAD} = 1.6mA$ , $V_{CC} = 4.75 V$ ) SYNC, DB0-DB7, A0-A15, R/W	1,2,3 MHz 4 MHz	- --	0.4 0.8
$P_D$	Power Dissipation ( $V_{CC} = 5.25V$ )	1 MHz and 2 MHz 3 MHz 4 MHz	- - -	700 800 900
C	Capacitance ( $V_{in} = 0$ , $T_A = 25^\circ C$ , $f = 1 MHz$ ) RES, NMI, RDY, IRQ, S.O., DBE DB0-DB7			
$C_{in}$	A0-A15, R/W, SYNC	-	10	
$C_{out}$	$\theta_o(\text{in})$ (650X)	-	15	
$C_{\theta_1}$	$\theta_1$ (651X)	-	12	pF
$C_{\theta_2}$	$\theta_2$ (651X)	-	15 50 80	

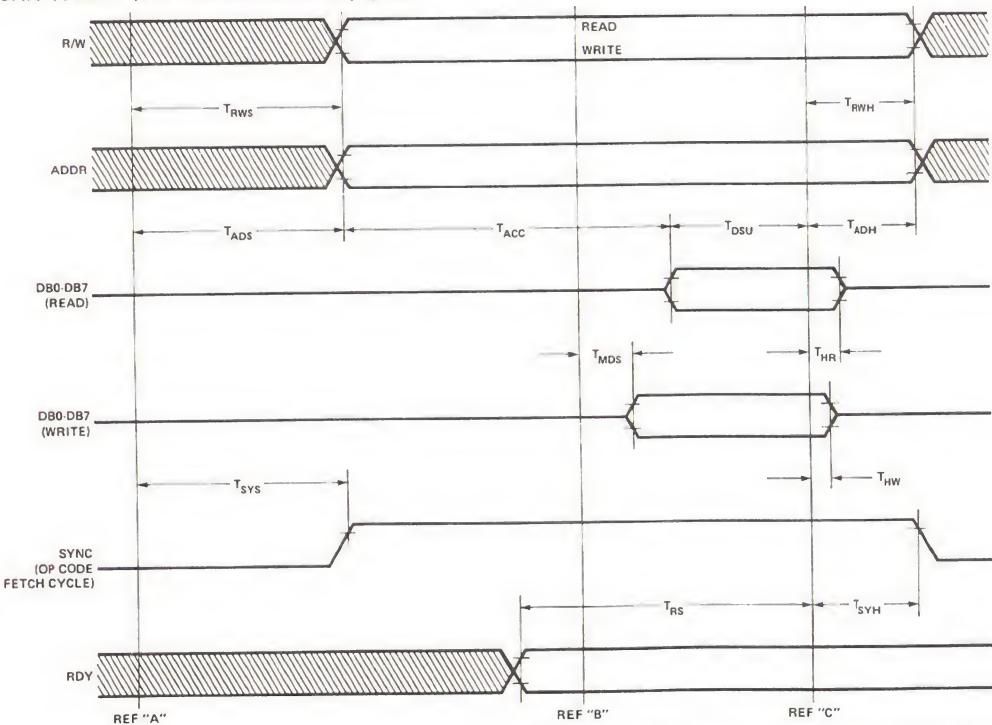
**TIMING DEFINITIONS, COMPOSITE TIMING DIAGRAM (See note at bottom of page 5)**  
**SY651X INPUT CLOCK TIMING**



**SY650X INPUT CLOCK TIMING**



**SY65XX TIMING (See note at bottom of page 5)**



## DYNAMIC OPERATING CHARACTERISTICS

(V<sub>CC</sub> = 5.0 ± 5%, T<sub>A</sub> = 0° to 70°C)

Parameter	Symbol	1 MHz		2 MHz		3 MHz		4 MHz		Units
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
<b>651X</b>										
Cycle Time	T <sub>CYC</sub>	1.00	40	0.50	40	0.33	40	0.25	40	μs
θ <sub>1</sub> Pulse Width	T <sub>PWHθ1</sub>	430	—	215	—	150	—	—	—	ns
θ <sub>2</sub> Pulse Width	T <sub>PWHθ2</sub>	470	—	235	—	160	—	—	—	ns
Delay Between θ <sub>1</sub> and θ <sub>2</sub>	T <sub>D</sub>	0	—	0	—	0	—	—	—	ns
θ <sub>1</sub> and θ <sub>2</sub> Rise and Fall Times <sup>[1]</sup>	T <sub>R</sub> , T <sub>F</sub>	0	25	0	20	0	15	—	—	ns
<b>650X, 651X</b>										
Cycle Time	T <sub>CYC</sub>	1.00	40	0.50	40	0.33	40	0.25	40	μs
θ <sub>0(IN)</sub> Low Time <sup>[2]</sup>	T <sub>Lθ0</sub>	480	—	240	—	160	—	110	—	ns
θ <sub>0(IN)</sub> High Time <sup>[2]</sup>	T <sub>Hθ0</sub>	460	—	240	—	160	—	115	—	ns
θ <sub>0</sub> Neg to θ <sub>1</sub> Pos Delay <sup>[5]</sup>	T <sub>01+</sub>	10	70	10	70	10	70	10	70	ns
θ <sub>0</sub> Neg to θ <sub>2</sub> Neg Delay <sup>[5]</sup>	T <sub>02-</sub>	5	65	5	65	5	65	5	65	ns
θ <sub>0</sub> Pos to θ <sub>1</sub> Neg Delay <sup>[5]</sup>	T <sub>01-</sub>	5	65	5	65	5	65	5	65	ns
θ <sub>0</sub> Pos to θ <sub>2</sub> Pos Delay <sup>[5]</sup>	T <sub>02+</sub>	15	75	15	75	15	75	15	75	ns
θ <sub>0(IN)</sub> Rise and Fall Time <sup>[1]</sup>	T <sub>RO</sub> , T <sub>FO</sub>	0	30	0	20	0	15	0	10	ns
θ <sub>1(OUT)</sub> Pulse Width	T <sub>PWHθ1</sub>	T <sub>Lθ0</sub> -20	T <sub>Lθ0</sub>	ns						
θ <sub>2(OUT)</sub> Pulse Width	T <sub>PWHθ2</sub>	T <sub>Lθ0</sub> -40	T <sub>Lθ0</sub> -10	ns						
Delay Between θ <sub>1</sub> and θ <sub>2</sub>	T <sub>D</sub>	5	—	5	—	5	—	5	—	ns
θ <sub>1</sub> and θ <sub>2</sub> Rise and Fall Times <sup>[1,3]</sup>	T <sub>R</sub> , T <sub>F</sub>	—	25	—	25	—	15	—	15	ns
<b>650X, 651X</b>										
R/W Setup Time	T <sub>RWS</sub>	—	225	—	140	—	110	—	90	ns
R/W Hold Time	T <sub>RWH</sub>	30	—	30	—	15	—	10	—	ns
Address Setup Time	T <sub>ADS</sub>	—	225	—	140	—	110	—	90	ns
Address Hold Time	T <sub>ADH</sub>	30	—	30	—	15	—	10	—	ns
Read Access Time	T <sub>ACC</sub>	—	650	—	310	—	170	—	110	ns
Read Data Setup Time	T <sub>DSU</sub>	100	—	50	—	50	—	50	—	ns
Read Data Hold Time	T <sub>HR</sub>	10	—	10	—	10	—	10	—	ns
Write Data Setup Time	T <sub>MDS</sub>	20	175	20	100	20	75	—	70	ns
Write Data Hold Time	T <sub>HW</sub>	60	150	60	150	30	130	20	—	ns
Sync Setup Time	T <sub>SYS</sub>	—	350	—	175	—	100	—	90	ns
Sync Hold Time	T <sub>SYH</sub>	30	—	30	—	15	—	15	—	ns
RDY Setup Time <sup>[4]</sup>	T <sub>RS</sub>	200	—	200	—	150	—	120	—	ns

## NOTES:

- Measured between 10% and 90% points on waveform.
- Measured at 50% points.
- Load = 1 TTL load +30 pF.
- RDY must never switch states within T<sub>RS</sub> to end of θ<sub>2</sub>.
- Load = 100 pF.
- The 2 MHz devices are identified by an "A" suffix.
- The 3 MHz devices are identified by a "B" suffix.
- The 4 MHz devices are identified by a "C" suffix.

## TIMING DIAGRAM NOTE:

Because the clock generation for the SY650X and SY651X is different, the two clock timing sections are referenced to the main timing diagram by three reference lines marked REF 'A', REF 'B' and REF 'C'. Reference between the two sets of clock timings is without meaning. Timing parameters are referred to these lines and scale variations in the diagrams are of no consequence.

## PIN FUNCTIONS

### Clocks ( $\emptyset_1, \emptyset_2$ )

The SY651X requires a two phase non-overlapping clock that runs at the  $V_{cc}$  voltage level.

The SY650X clocks are supplied with an internal clock generator. The frequency of these clocks is externally controlled. Clock generator circuits are shown elsewhere in this data sheet.

**Address Bus ( $A_0-A_{15}$ )** (See sections on each micro for respective address lines on those devices.)

These outputs are TTL compatible, capable of driving one standard TTL load and 130 pF.

### Data Bus ( $DB_0-DB_7$ )

Eight pins are used for the data bus. This is a bi-directional bus, transferring data to and from the device and peripherals. The outputs are three-state buffers, capable of driving one standard TTL load and 130 pF.

### Data Bus Enable (DBE)

This TTL compatible input allows external control of the three-state data output buffers and will enable the microprocessor bus driver when in the high state. In normal operation DBE would be driven by the phase two ( $\emptyset_2$ ) clock, thus allowing data output from microprocessor only during  $\emptyset_2$ . During the read cycle, the data bus drivers are internally disabled, becoming essentially an open circuit. To disable data bus drivers externally, DBE should be held low. This signal is available on the SY6512, only.

### Ready (RDY)

This input signal allows the user to halt the microprocessor on all cycles except write cycles. A negative transition to the low state during or coincident with phase one ( $\emptyset_1$ ) will halt the microprocessor with the output address lines reflecting the current address being fetched. This condition will remain through a subsequent phase two ( $\emptyset_2$ ) in which the Ready signal is low. This feature allows microprocessor interfacing with low speed PROMS as well as fast (max. 2 cycle) Direct Memory Access (DMA). If ready is low during a write cycle, it is ignored until the following read operation. Ready transitions must not be permitted during  $\emptyset_2$  time.

### Interrupt Request ( $\overline{IRQ}$ )

This TTL level input requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the current instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, therefore transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A  $3K\Omega$  external resistor should be used for proper wire-OR operation.

### Non-Maskable Interrupt ( $\overline{NMI}$ )

A negative going transition on this input requests that a non-maskable interrupt sequence be generated within the microprocessor.

$\overline{NMI}$  is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for  $\overline{IRQ}$  will be performed, regardless of the state interrupt mask flag. The vector address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, thereby transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$  also requires an external  $3K\Omega$  resistor to  $V_{cc}$  for proper wire-OR operations.

Inputs  $\overline{IRQ}$  and  $\overline{NMI}$  are hardware interrupt lines that are sampled during  $\emptyset_2$  (phase 2) and will begin the appropriate interrupt routine on the  $\emptyset_1$  (phase 1) following the completion of the current instruction.

### Set Overflow Flag (S.O.)

A NEGATIVE going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of  $\emptyset_1$ .

### SYNC

This output line is provided to identify those cycles in which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during  $\emptyset_1$  of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the  $\emptyset_1$  clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

### Reset ( $\overline{RES}$ )

This input is used to reset or start the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from the memory vector locations FFFC and FFFD. This is the start location for program control.

After  $V_{cc}$  reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles. At this time the R/W and SYNC signal will become valid.

When the reset signal goes high following these two clock cycles, the microprocessor will proceed with the normal reset procedure detailed above.

### Read/Write (R/W)

This output signal is used to control the direction of data transfers between the processor and other circuits on the data bus. A high level on R/W signifies data into the processor; a low is for data transfer out of the processor.

## PROGRAMMING CHARACTERISTICS

### INSTRUCTION SET – ALPHABETIC SEQUENCE

ADC	Add Memory to Accumulator with Carry	DEC	Decrement Memory by One	PHA	Push Accumulator on Stack
AND	"AND" Memory with Accumulator	DEX	Decrement Index X by One	PHP	Push Processor Status on Stack
ASL	Shift left One Bit (Memory or Accumulator)	DEY	Decrement Index Y by One	PLA	Pull Accumulator from Stack
BCC	Branch on Carry Clear	EOR	"Exclusive or" Memory with Accumulator	PLP	Pull Processor Status from Stack
BCS	Branch on Carry Set	INC	Increment Memory by One	ROL	Rotate One Bit Left (Memory or Accumulator)
BEQ	Branch on Result Zero	INX	Increment Index X by One	ROR	Rotate One Bit Right (Memory or Accumulator)
BIT	Test Bits in Memory with Accumulator	INY	Increment Index Y by One	RTI	Return from Interrupt
BMI	Branch on Result Minus	JMP	Jump to New Location	RTS	Return from Subroutine
BNE	Branch on Result not Zero	JSR	Jump to New Location Saving Return Address	SBC	Subtract Memory from Accumulator with Borrow
BPL	Branch on Result Plus	LDA	Load Accumulator with Memory	SEC	Set Carry Flag
BRK	Force Break	LDX	Load Index X with Memory	SED	Set Decimal Mode
BVC	Branch on Overflow Clear	LDY	Load Index Y with Memory	SEI	Set Interrupt Disable Status
BVS	Branch on Overflow Set	LSR	Shift One Bit Right (Memory or Accumulator)	STA	Store Accumulator in Memory
CLC	Clear Carry Flag	NOP	No Operation	STX	Store Index X in Memory
CLD	Clear Decimal Mode	ORA	"OR" Memory with Accumulator	STY	Store Index Y in Memory
CLI	Clear Interrupt Disable Bit			TAX	Transfer Accumulator to Index X
CLV	Clear Overflow Flag			TAY	Transfer Accumulator to Index Y
CMP	Compare Memory and Accumulator			TSX	Transfer Stack Pointer to Index X
CPX	Compare Memory and Index X			TXA	Transfer Index X to Accumulator
CPY	Compare Memory and index Y			TXS	Transfer Index X to Stack Pointer
				TYA	Transfer Index Y to Accumulator

### ADDRESSING MODES

#### Accumulator Addressing

This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

#### Immediate Addressing

In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

#### Absolute Addressing

In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

#### Zero Page Addressing

The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

#### Indexed Zero Page Addressing – (X, Y indexing)

This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X," or "Zero Page, Y." The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

#### Indexed Absolute Addressing – (X, Y indexing)

This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X," and "Absolute, Y." The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

#### Implied Addressing

In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

#### Relative Addressing

Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

#### Indexed Indirect Addressing

In indexed indirect addressing (referred to as (Indirect,X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

#### Indirect Indexed Addressing

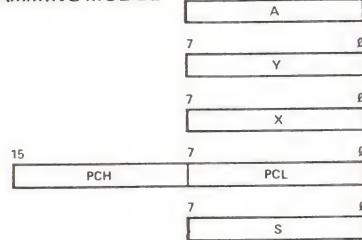
In indirect indexed addressing (referred to as (Indirect,Y)), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

#### Absolute Indirect

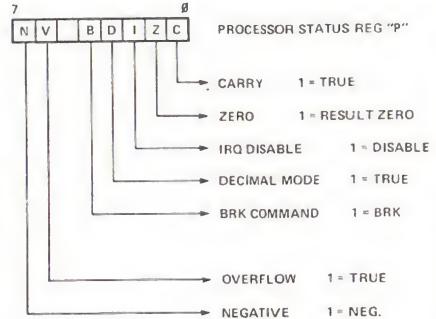
The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter.

## PROGRAMMING CHARACTERISTICS

## PROGRAMMING MODEL



ACCUMULATOR      A  
INDEX REGISTER    Y  
INDEX REGISTER    X  
PROGRAM COUNTER   "PC"  
STACK POINTER     "S"



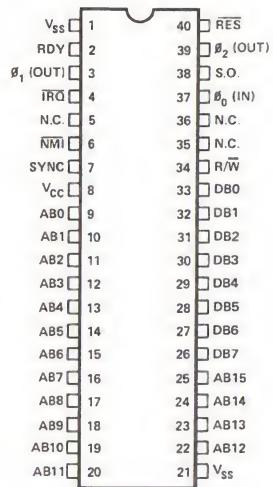
## INSTRUCTION SET – OP CODES, EXECUTION TIME, MEMORY REQUIREMENTS

INSTRUCTIONS	IMMEDIATE	ABSOLUTE	ZEROPAGE	ACCUM	IMPLD	IND X	IND Y	Z PAGE	ABS X	ABS Y	RELATIVE	INDIRECT	Z PAGE	CONDITION CODES
Mnemonic	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	N Z C I D V
ADC A+M+C-A	(4)(1)69 2 2	60 4 3	65 3 2			61 6 2	71 5 2	75 4 2	70 4 3	79 4 3				
AND A#M-A	(1)29 2 2	20 4 3	25 3 2			21 6 2	31 5 2	35 4 2	30 4 3	39 4 3				
ASL C<7>0		0E 16 3	0E 5 2	0A 2 1							90 2 2			
BCC BRANCH ON C=0	(2)										80 2 2			
BCS BRANCH ON C=1	(2)										F0 2 2			
BEQ BRANCH ON Z=1	(2)													
BIT A#M		2C 4 3	24 3 2											M <sub>2</sub> / M <sub>1</sub>
BMI BRANCH ON N=1	(2)										30 2 2			
BNE BRANCH ON Z=0	(2)										00 2 2			
BPL BRANCH ON N=0	(2)										10 2 2			
BRK (See Fig. 1)							90 ? 1							
BVC BRANCH ON V=0	(2)										50 2 2			
BVS BRANCH ON V=1	(2)										70 2 2			
CLC 0-C							1B 2 1							0
CLD 0-D							DB 2 1							0
CLI 0-I							5B 2 1							0
CLV 0-V							BB 2 1							0
CMP A#M	(1)C9 2 2	CD 4 3	C5 3 2					11 6 2	01 5 2	05 4 2	00 4 3	09 4 3		
CPX X#M	E9 2 2	EC 4 3	E4 3 2											
CPY Y#M	C9 2 2	CC 4 3	C4 3 2											
DEC M-1-M	(1)E1 6 2	E1 6 2	E1 6 2					06 6 2	0E 7 3					
DEX X-1-X							CA 2 1							
DEY Y-1-Y							BB 2 1							
EDR A#M-A	(1)49 2 2	4D 4 3	46 3 2					41 6 2	51 5 2	55 4 2	50 4 3	59 4 3		
INC M+1-M	(1)EE 6 2	EE 6 2	EE 6 2					4F 6 2	FE 7 2					
INX X+1-X							EB 2 1							
INY Y+1-Y							CB 2 1							
JMP JUMP TO NEW LOC		4C 3 3												
JSR (See Fig. 2) JUMP SUB	20 6 3										60 5 3			
LDA M-A	(1)A9 2 2	AD 4 3	A5 3 2					A1 6 2	B1 5 2	85 4 2	BD 4 3	89 4 1		
INSTRUCTIONS	IMMEDIATE	ABSOLUTE	ZEROPAGE	ACCUM	IMPLD	IND X	IND Y	Z PAGE	IND X	IND Y	RELATIVE	INDIRECT	Z PAGE	CONDITION CODES
Mnemonic	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	Op N	N Z C I D V
LDX M-X	(1)A2 2 2	AE 4 3	AG 3 2						84 4 2	8C 4 3				
LDY M-Y	(1)A9 2 2	AC 4 3	AA 3 2						56 6 2	5E 7 3				
LSR 0<7>0		AE 6 3	46 5 2	4A 2 1										0 ✓ ✓
NOP NO OPERATION							EA 2 1							
ORA A#M-A	09 2 2	9D 4 3	05 3 2					01 6 2	11 5 2	15 4 2	1D 4 3	19 4 3		
PHA A-Ms	S 1-S						48 3 1							
PHP P-Ms	S 1-S						88 3 1							
PLA S+1-S	Ms-A						68 4 1							
PLP S+1-S	Ms-P						28 4 1							(RESTORED)
ROL 0<7>0		2E 6 3	26 5 2	2A 2 1					36 6 2	3E 7 3				
ROR 0<7>0		76 6 3	66 5 2	6A 2 1					76 6 2	E7 3				RESTORED
RTI							40 6 1							
RTN (See Fig. 2) RTN INT							60 6 1							
RTS (See Fig. 2) RTN SUB								E1 6 2	F1 5 2	F5 4 2	FD 4 3	FF 4 3		
SBC A-M-C-A	(1)E9 2 2	ED 4 3	E5 3 2					38 2 1						
SEC	I-C							FB 2 1						1
SED	I-D							78 2 1						
SEI	I-I													
STA A-M		BD 4 3	85 3 2					81 6 2	91 6 2	95 4 2	9D 5 3	99 5 3		
STX X-M		BF 4 3	86 3 2											
STY Y-M		BC 4 3	84 3 2											
TAX A-X							AA 2 1							
TAY A-Y							AB 2 1							
TSX S-X							BA 2 1							
TXA X-A							BC 2 1							
TXS X-S							9A 2 1							
TYA Y-A							9B 2 1							
X INDEX X														
Y INDEX Y														
A ACCUMULATOR														
M MEMORY PER EFFECTIVE ADDRESS														
MS MEMORY PER STACK POINTER														
X INDEX X														
Y INDEX Y														
A AND														
V OR														
V EXCLUSIVE OR														
Z MODIFIED														

- (1) ADD 1 TO N IF PAGE BOUNDARY IS CROSSED  
 (2) ADD 1 TO N IF BRANCH OCCURS TO SAME PAGE  
 (3) ADD 2 TO N IF BRANCH OCCURS TO DIFFERENT PAGE  
 (4) CARRY NOT = BELOW  
 (5) IF IN DECIMAL MODE Z FLAG IS INVALID  
 ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

- X INDEX X  
 Y INDEX Y  
 A ACCUMULATOR  
 M MEMORY PER EFFECTIVE ADDRESS  
 MS MEMORY PER STACK POINTER

- ADD  
 SUBTRACT  
 AND  
 OR  
 EXCLUSIVE OR  
 MODIFIED
- M<sub>1</sub> MEMORY BIT  
 M<sub>2</sub> MEMORY BTG  
 N NO CYCLES  
 V NO BYTES

**SY6502 – 40 Pin Package****Features**

- 65K Addressable Bytes of Memory
- IRQ Interrupt      • NMI Interrupt
- On-the-chip Clock
  - ✓ TTL Level Single Phase Input
  - ✓ Crystal Time Base Input
- SYNC Signal
  - (can be used for single instruction execution)
- RDY Signal
  - (can be used for single cycle execution)
- Two Phase Output Clock for Timing of Support Chips



# Asynchronous Communication Interface Adapter

SYNERTEK  
A SUBSIDIARY OF HONEYWELL

## SY6551

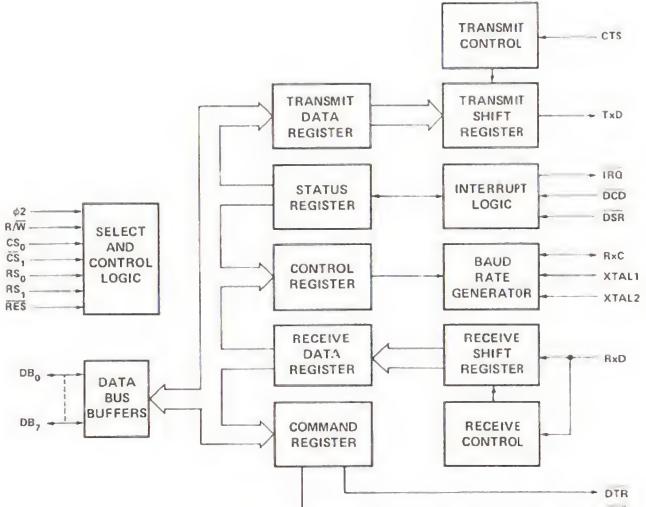
### MICROPROCESSOR PRODUCTS

- On-chip baud rate generator: 15 programmable baud rates derived from a standard 1.8432 MHz external crystal (50 to 19,200 baud).
- Programmable interrupt and status register to simplify software design.
- Single +5 volt power supply.
- Serial echo mode.
- False start bit detection.
- 8-bit bi-directional data bus for direct communication with the microprocessor.
- External 16x clock input for non-standard baud rates (up to 125 Kbaud).
- Programmable: word lengths; number of stop bits; and parity bit generation and detection.
- Data set and modem control signals provided.
- Parity: (odd, even, none, mark, space).
- Full-duplex or half-duplex operation.
- 5, 6, 7, 8 and 9 bit transmission.

The SY6551 is an Asynchronous Communication Adapter (ACIA) intended to provide for interfacing the 6500/6800 microprocessor families to serial communication

data sets and modems. A unique feature is the inclusion of an on-chip programmable baud rate generator, with a crystal being the only external component required.

#### PIN CONFIGURATION



#### ORDERING INFORMATION

Part No.	Package	Clock Rate
SYC6551	Ceramic	1 MHz
SYD6551	Cerdip	1 MHz
SYP6551	Plastic	1 MHz
SYC6551A	Ceramic	2 MHz
SYD6551A	Ceramic	2 MHz
SYP6551A	Plastic	2 MHz

Figure 1. Block Diagram

## ABSOLUTE MAXIMUM RATINGS

Rating	Symbol	Allowable Range
Supply Voltage	$V_{CC}$	-0.3V to +7.0V
Input/Output Voltage	$V_{IN}$	-0.3V to +7.0V
Operating Temperature	$T_{OP}$	0°C to 70°C
Storage Temperature	$T_{STG}$	-55°C to 150°C

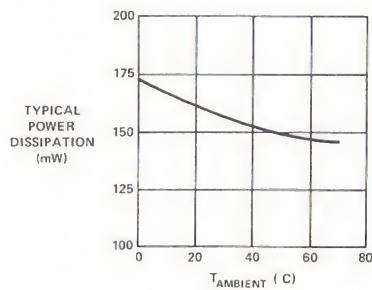
All inputs contain protection circuitry to prevent damage to high static charges. Care should be exercised to prevent unnecessary application of voltages in excess of the allowable limits.

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS ( $V_{CC} = 5.0V \pm 5\%$ ,  $T_A = 0-70^\circ C$ , unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage	$V_{IH}$	2.0	—	$V_{CC}$	V
Input Low Voltage	$V_{IL}$	-0.3	—	0.8	V
Input Leakage Current: $V_{IN} = 0$ to 5V ( $\phi 2$ , R/W, $\overline{RES}$ , $\overline{CS}_0$ , $\overline{CS}_1$ , $\overline{RS}_0$ , $\overline{RS}_1$ , $\overline{CTS}$ , Rx $D$ , $\overline{DCD}$ , $\overline{DSR}$ )	$I_{IN}$	—	$\pm 1.0$	$\pm 2.5$	$\mu A$
Input Leakage Current for High Impedance State (Three State)	$I_{TSI}$	—	$\pm 2.0$	$\pm 10.0$	$\mu A$
Output High Voltage: $I_{LOAD} = -100\mu A$ (DB <sub>0</sub> - DB <sub>7</sub> , Tx $D$ , Rx $C$ , RTS, DTR)	$V_{OH}$	2.4	—	—	V
Output Low Voltage: $I_{LGAD} = 1.6mA$ (DB <sub>0</sub> - DB <sub>7</sub> , Tx $D$ , Rx $C$ , RTS, DTR, IRQ)	$V_{OL}$	—	—	0.4	V
Output High Current (Sourcing): $V_{OH} = 2.4V$ (DB <sub>0</sub> - DB <sub>7</sub> , Tx $D$ , Rx $C$ , RTS, DTR)	$I_{OH}$	-100	—	—	$\mu A$
Output Low Current (Sinking): $V_{OL} = 0.4V$ (DB <sub>0</sub> - DB <sub>7</sub> , Tx $D$ , Rx $C$ , RTS, DTR, IRQ)	$I_{OL}$	1.6	—	—	mA
Output Leakage Current (Off State): $V_{OUT} = 5V$ (IRQ)	$I_{OFF}$	—	1.0	10.0	$\mu A$
Clock Capacitance ( $\phi 2$ )	$C_{CLK}$	—	—	20	pF
Input Capacitance (Except XTAL1 and XTAL2)	$C_{IN}$	—	—	10	pF
Output Capacitance	$C_{OUT}$	—	—	10	pF
Power Dissipation (See Graph) ( $T_A = 0^\circ C$ ) $V_{CC} = 5.25V$	$P_D$	—	170	300	mW

## POWER DISSIPATION vs TEMPERATURE



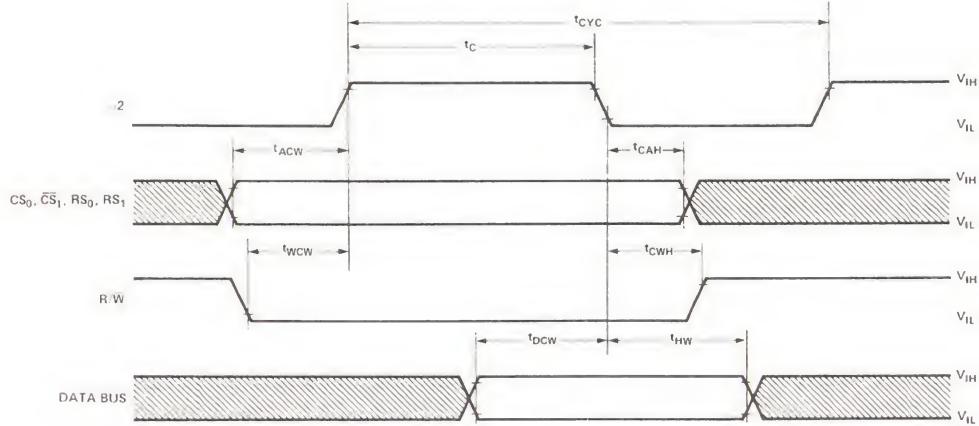


Figure 2. Write Timing Characteristics

WRITE CYCLE ( $V_{CC} = 5.0V \pm 5\%$ ,  $T_A = 0$  to  $70^\circ C$ , unless otherwise noted)

Characteristic	Symbol	SY6551		SY6551A		Unit
		Min	Max	Min	Max	
Cycle Time	$t_{Cyc}$	1.0	—	0.5	—	$\mu s$
$\phi_2$ Pulse Width	$t_C$	400	—	200	—	ns
Address Set-Up Time	$t_{ACW}$	120	—	70	—	ns
Address Hold Time	$t_{CAH}$	0	—	0	—	ns
R/W Set-Up Time	$t_{WCW}$	120	—	70	—	ns
R/W Hold Time	$t_{CWH}$	0	—	0	—	ns
Data Bus Set-Up Time	$t_{DCW}$	150	—	60	—	ns
Data Bus Hold Time	$t_{HW}$	20	—	20	—	ns

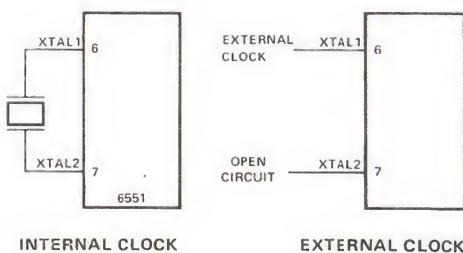
(tr and tf = 10 to 30 ns)

## CRYSTAL SPECIFICATION

1. Temperature stability  $\pm 0.01\%$  ( $0^\circ$  to  $70^\circ C$ )
2. Characteristics at  $25^\circ C \pm 2^\circ C$ 
  - a. Frequency (MHz) 1.8432
  - b. Frequency tolerance ( $\pm\%$ ) 0.02
  - c. Resonance mode Series
  - d. Equivalent resistance (ohm) 400 max.
  - e. Drive level mW 2
  - f. Shunt capacitance pF 7 max.
  - g. Oscillation mode Fundamental

No other external components should be in the crystal circuit

## CLOCK GENERATION



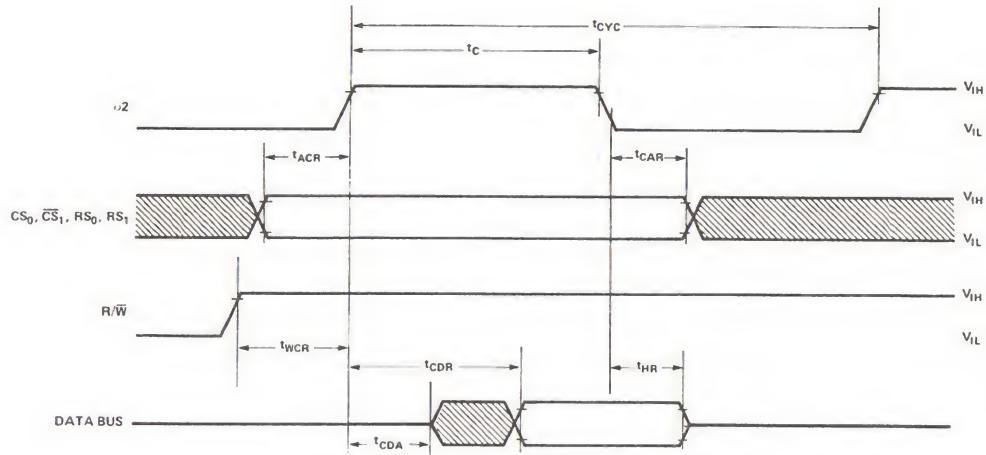
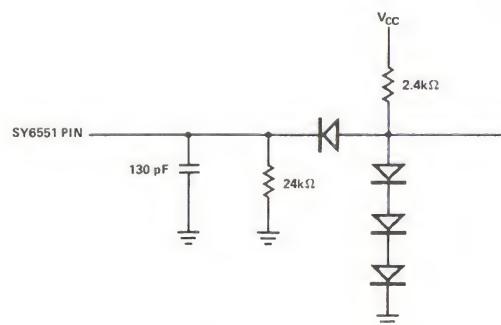


Figure 3. Read Timing Characteristics

**READ CYCLE** ( $V_{CC} = 5.0V \pm 5\%$ ,  $T_A = 0$  to  $70^\circ C$ , unless otherwise noted)

Characteristic	Symbol	SY6551		SY6551A		Unit
		Min	Max	Min	Max	
Cycle Time	t <sub>CYC</sub>	1.0	—	0.5	—	μs
Pulse Width ( $\phi_2$ )	t <sub>C</sub>	400	—	200	—	ns
Address Set-Up Time	t <sub>ACR</sub>	120	—	70	—	ns
Address Hold Time	t <sub>CAR</sub>	0	—	0	—	ns
R/W Set-Up Time	t <sub>WCR</sub>	120	—	70	—	ns
Read Access Time (Valid Data)	t <sub>CDR</sub>	—	200	—	150	ns
Read Data Hold Time	t <sub>HR</sub>	20	—	20	—	ns
Bus Active Time (Invalid Data)	t <sub>CDA</sub>	40	—	40	—	ns



TEST LOAD FOR DATA BUS (DB<sub>0</sub> – DB<sub>7</sub>, TxD, DTR, RTS OUTPUTS)

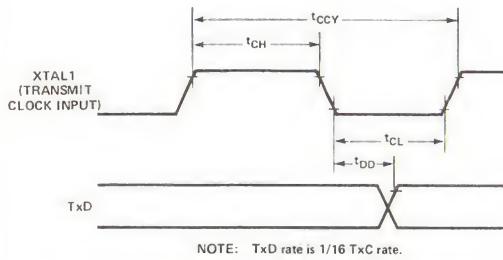


Figure 4a. Transmit Timing with External Clock

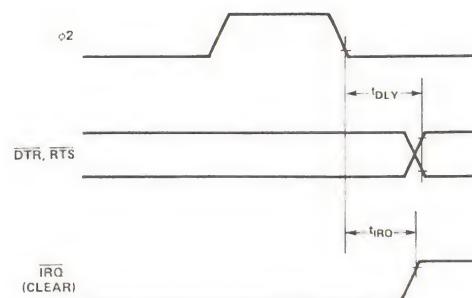


Figure 4b. Interrupt and Output Timing

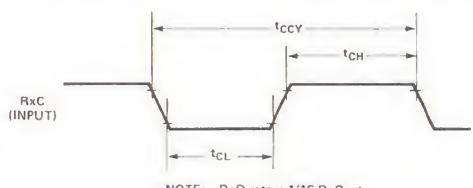


Figure 4c. Receive External Clock Timing

## TRANSMIT/RECEIVE CHARACTERISTICS

Characteristic	Symbol	SY6551		SY6551A		Unit
		Min	Max	Min	Max	
Transmit/Receive Clock Rate	$t_{CCY}$	400*	—	400*	—	ns
Transmit/Receive Clock High Time	$t_{CH}$	175	—	175	—	ns
Transmit/Receive Clock Low Time	$t_{CL}$	175	—	175	—	ns
XTAL1 to TxD Propagation Delay	$t_{DD}$	—	500	—	500	ns
Propagation Delay (RTS, DTR)	$t_{DLY}$	—	500	—	500	ns
IRQ Propagation Delay (Clear)	$t_{IRQ}$	—	500	—	500	ns

(tr, tf = 10 to 30 nsec)

\*The baud rate with external clocking is: Baud Rate =  $\frac{1}{16 \times T_{CCY}}$ 

## INTERFACE SIGNAL DESCRIPTION

**RES** (Reset)

During system initialization a low on the RES input will cause internal registers to be cleared.

**φ2** (Input Clock)

The input clock is the system φ2 clock and is used to trigger all data transfers between the system microprocessor and the SY6551.

**R/W** (Read/Write)

The R/W is generated by the microprocessor and is used to control the direction of data transfers. A high on the R/W pin allows the processor to read the data supplied by the SY6551. A low on the R/W pin allows a write to the SY6551.

**IRQ** (Interrupt Request)

The IRQ pin is an interrupt signal from the interrupt control logic. It is an open drain output, permitting

several devices to be connected to the common IRQ microprocessor input. Normally a high level, IRQ goes low when an interrupt occurs.

**DB<sub>0</sub> - DB<sub>7</sub>** (Data Bus)

The DB<sub>0</sub>-DB<sub>7</sub> pins are the eight data lines used for transfer of data between the processor and the SY6551. These lines are bi-directional and are normally high-impedance except during Read cycles when selected.

**CS<sub>0</sub>, CS<sub>1</sub>** (Chip Selects)

The two chip select inputs are normally connected to the processor address lines either directly or through decoders. The SY6551 is selected when CS<sub>0</sub> is high and CS<sub>1</sub> is low.

**RS<sub>0</sub>, RS<sub>1</sub>** (Register Selects)

The two register select lines are normally connected to the processor address lines to allow the processor to select the various SY6551 internal registers. The following table indicates the internal register select coding:

RS <sub>1</sub>	RS <sub>0</sub>	Write	Read
0	0	Transmit Data Register	Receiver Data Register
0	1	Programmed Reset (Data is "Don't Care")	Status Register
1	0	Command Register	
1	1	Control Register	

The table shows that only the Command and Control registers are read/write. The Programmed Reset operation does not cause any data transfer, but is used to clear the SY6551 registers. The Programmed Reset is slightly different from the Hardware Reset ( $\overline{RES}$ ) and these differences are described in the individual register definitions.

### ACIA/MODEM INTERFACE SIGNAL DESCRIPTION

#### XTAL1, XTAL2 (Crystal Pins)

These pins are normally directly connected to the external crystal (1.8432 MHz) used to derive the various baud rates. Alternatively, an externally generated clock may be used to drive the XTAL1 pin, in which case the XTAL2 pin must float.

#### TxD (Transmit Data)

The TxD output line is used to transfer serial NRZ (non-return-to-zero) data to the modem. The LSB (least significant bit) of the Transmit Data Register is the first data bit transmitted and the rate of data transmission is determined by the baud rate selected.

#### RxD (Receive Data)

The RxD input line is used to transfer serial NRZ data into the ACIA from the modem, LSB first. The receiver data rate is either the programmed baud rate or the rate of an externally generated receiver clock. This selection is made by programming the Control Register.

#### RxC (Receive Clock)

The RxC is a bi-directional pin which serves as either the receiver 16x clock input or the receiver 16x clock output. The latter mode results if the internal baud rate generator is selected for receiver data clocking.

#### RTS (Request to Send)

The RTS output pin is used to control the modem from the processor. The state of the RTS pin is determined by the contents of the Command Register.

#### CTS (Clear to Send)

The CTS input pin is used to control the transmitter operation. The enable state is with CTS low. The transmitter is automatically disabled if CTS is high.

#### DTR (Data Terminal Ready)

This output pin is used to indicate the status of the SY6551 to the modem. A low on DTR indicates the SY6551 is enabled and a high indicates it is disabled. The processor controls this pin via bit 0 of the Command Register.

#### DSR (Data Set Ready)

The DSR input pin is used to indicate to the SY6551 the status of the modem. A low indicates the "ready" state and a high, "not-ready." DSR is a high-impedance input and must not be a no-connect. If unused, it should be driven high or low, but not switched.

Note: If Command Register Bit 0 = 1 and a change of state on DSR occurs, IRQ will be set, and Status Register Bit 6 will reflect the new level. The state of DSR does not affect either Transmitter or Receiver operation.

#### DCD (Data Carrier Detect)

The DCD input pin is used to indicate to the SY6551 the status of the carrier-detect output of the modem. A low indicates that the modem carrier signal is present and a high, that it is not. DCD, like DSR, is a high-impedance input and must not be a no-connect.

Note: If Command Register Bit 0 = 1 and a change of state on DCD occurs, IRQ will be set, and Status Register Bit 5 will reflect the new level. The state of DCD does not affect Transmitter operation, but must be low for the Receiver to operate.

### INTERNAL ORGANIZATION

The Transmitter/Receiver sections of the SY6551 are depicted by the block diagram in Figure 5.

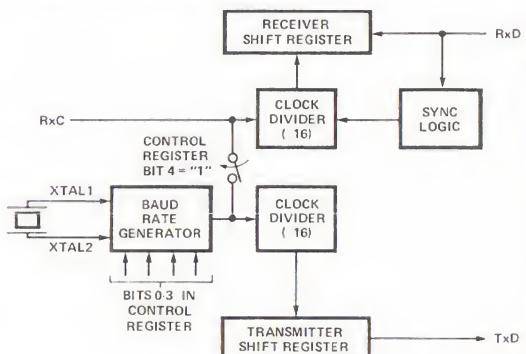


Figure 5. Transmitter/Receiver Clock Circuits

Bits 0-3 of the Control Register select the divisor used to generate the baud rate for the Transmitter. If the Receiver clock is to use the same baud rate as the Transmitter, then RxC becomes an output pin and can be used to slave other circuits to the SY6551.

### CONTROL REGISTER

The Control Register is used to select the desired mode for the SY6551. The word length, number of stop bits, and clock controls are all determined by the Control Register, which is depicted in Figure 6.

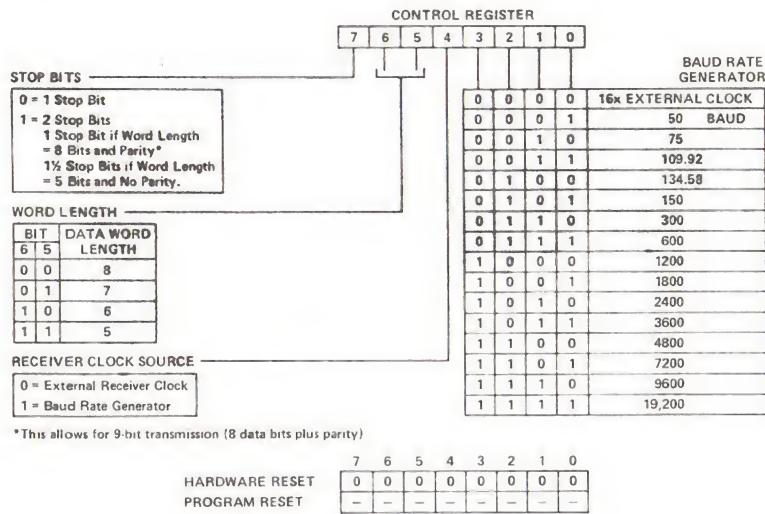


Figure 6. Control Register Format

### COMMAND REGISTER

The Command Register is used to control Specific Transmit/Receive functions and is shown in Figure 7.

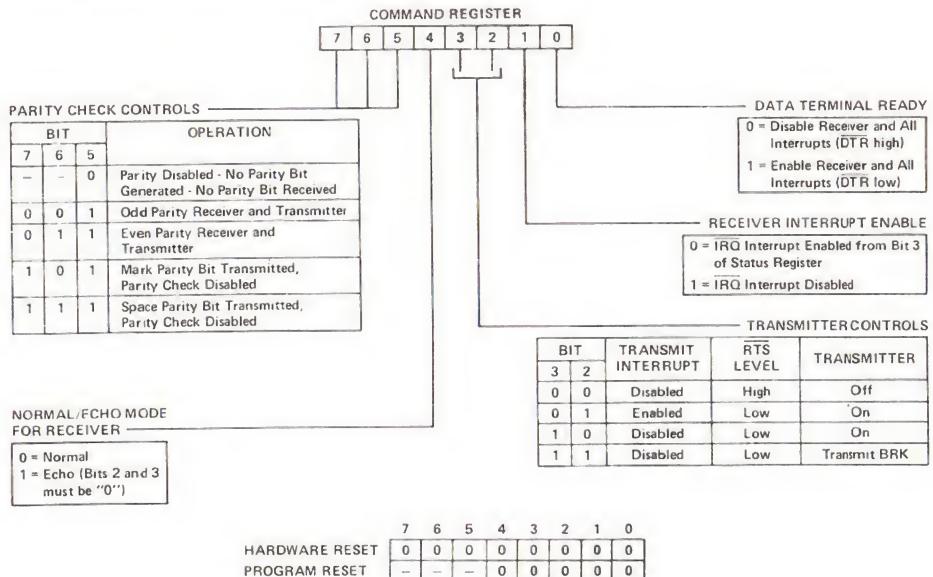


Figure 7. Command Register Format

### STATUS REGISTER

The Status Register is used to indicate to the processor the status of various SY6551 functions and is outlined in Figure 8.

7	6	5	4	3	2	1	0
STATUS							
Parity Error*		0 = No Error 1 = Error		Self Clearing**			
Framing Error*		0 = No Error 1 = Error		Self Clearing**			
Overrun*		0 = No Error 1 = Error		Self Clearing**			
Receive Data Register Full		0 = Not Full 1 = Full		Read Receive Data Register			
Transmit Data Register Empty		0 = Not Empty 1 = Empty		Write Transmit Data Register			
DCD		0 = DCD Low 1 = DCD High		Not Resettable Reflects DCD State			
DSR		0 = DSR Low 1 = DSR High		Not Resettable Reflects DSR State			
IRO		0 = No Interrupt 1 = Interrupt		Read Status Register			

\*NO INTERRUPT GENERATED FOR THESE CONDITIONS.  
\*\*CLEARED AUTOMATICALLY AFTER A READ OF RDR AND THE NEXT ERROR FREE RECEIPT OF DATA.

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	-	-	1	0	0	0	0
PROGRAM RESET	-	-	-	-	-	0	-	-

Figure 8. Status Register Format

### TRANSMIT AND RECEIVE DATA REGISTERS

These registers are used as temporary data storage for the 6551 Transmit and Receive circuits. The Transmit Data Register is characterized as follows:

- Bit 0 is the leading bit to be transmitted.
- Unused data bits are the high-order bits and are "don't care" for transmission.

The Receive Data Register is characterized in a similar fashion:

- Bit 0 is the leading bit received.
- Unused data bits are the high-order bits and are "0" for the receiver.
- Parity bits are not contained in the Receive Data Register, but are stripped-off after being used for external parity checking. Parity and all unused high-order bits are "0".

Figure 9 illustrates a single transmitted or received data word, for the example of 8 data bits, parity, and 1 stop bit.

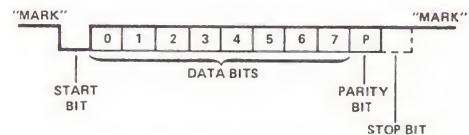
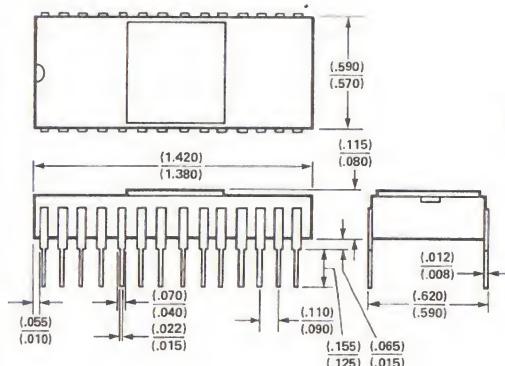


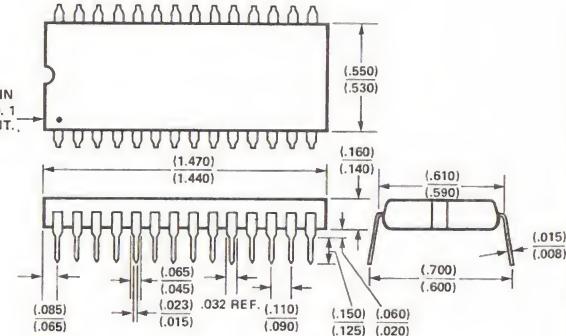
Figure 9. Serial Data Stream Example

### PACKAGE OUTLINES

#### 28 LEAD CERAMIC



#### 28 LEAD PLASTIC



O

C

O

C

## ANHANG N

```
Current memory available: 9169
0000: 0001      chrfont .equ    1
0000:
0000:           .include rom0
0000:           .absolute
0000:           ;
0000:           ;      revision A3 Board
0000:           ;
0000:           ;
0000:           ; 24*80 Monitor without tape IO
0000:           ;
2 blocks for procedure code 8685 words left
```

## MONITOR FILE:ROM0.TEXT

```

0000:          .proc    monitor
Current memory available: 8636
0000:
0000:          .org    0F800
F800: 0000      LOC0    .equ    0
F800: 0001      LOC1    .equ    1
F800: 0020      WNDLFT  .equ    20
F800: 0021      WNDWDTH .equ    21
F800: 0022      WNDTOP  .equ    22
F800: 0023      WNDBTM  .equ    23
F800: 0050      width   .equ    80.
F800: 0024      CH      .equ    24
F800: 0025      CV      .equ    25
F800: 0026      GBASL   .equ    26
F800: 0027      GBASH   .equ    27
F800: 0028      BASL    .equ    28
F800: 0029      BASH    .equ    29
F800: 002A      BAS2L   .equ    2A
F800: 002B      BAS2H   .equ    2B
F800: 002C      H2      .equ    2C
F800: 002C      LMNEM   .equ    2C
F800: 002D      V2      .equ    02D
F800: 002D      RMNEM   .equ    02D
F800: 002E      MASK    .equ    02E
F800: 002E      FORMAT  .equ    02E      ; dism
F800: 002F      LASTIN  .equ    02F      ; tape in
F800: 002F      LENGTH  .equ    02F      ; dism
F800: 0030      COLOR   .equ    030      ; LoRes color
F800: 0031      MODE    .equ    031      ; dism
F800: 0032      INVFLG  .equ    032      ; prompt char
F800: 0033      PROMPT  .equ    033
F800: 0034      YSAV    .equ    034
F800: 0035      YSAV1   .equ    035
F800: 0036      CSWL    .equ    036      ; output vector
F800: 0037      CSHW    .equ    037
F800: 0038      KSWL    .equ    038      ; input vector
F800: 0039      KSHW    .equ    039
F800: 003A      PCL     .equ    03A      ; go-, list-command
F800: 003B      PCH     .equ    03B
F800: 003C      A1L     .equ    03C
F800: 003D      A1H     .equ    03D
F800: 003E      A2L     .equ    03E
F800: 003F      A2H     .equ    03F
F800: 0040      A3L     .equ    040      ; memory set
F800: 0041      A3H     .equ    041
F800: 0042      A4L     .equ    042
F800: 0043      A4H     .equ    043
F800: 0045      ACC     .equ    45      ; 6502 register
F800: 0046      XREG   .equ    ACC+1
F800: 0047      YREG   .equ    ACC+2
F800: 0048      STATUS  .equ    ACC+3
F800: 0049      SPNT   .equ    ACC+4
F800: 004E      RNDL   .equ    04E      ; random number
F800: 004F      RNDH   .equ    04F
F800:

```

## MONITOR FILE:ROM0.TEXT

F800: 0200	IN	.equ 0200	; keyboard buffer
F800: 03F0	BRKV	.equ 03F0	; brk vector
F800: 03F2	SOFTEV	.equ 03F2	; soft reset vector
F800: 03F4	PWREDUP	.equ 03F4	
F800: 03F5	AMPERV	.equ 03F5	; Applesoft &
F800: 03F8	USRADR	.equ 03F8	; U-command
F800: 03FB	NMI	.equ 03FB	; jmp nmi
F800: 03FE	IRQLOC	.equ 03FE	; jmp \$irqloc
F800: 0400	LINE1	.equ 0400	; first screen line
F800: 0479	chy	.equ 479	; 80-col video driver
F800: 04F9	switch	.equ 4f9	; 40/80-col switch
F800: 07F8	MSLOT	.equ 07F8	; active slot ID
F800: C000	IOARD	.equ 0C000	
F800: 08C0	iopage	.equ 0C0	
F800: C000	KBD	.equ 0C000	; ASCII input
F800: C008	Kbdextn	.equ 0C008	; functions key input
F800: C006	chrBas	.equ 0C006	; 64+64+128 set (inverse, flash, normal)
F800: C002	chrgen0	.equ 0C002	; char gen A10
F800: C004	chrgen1	.equ 0C004	; char gen A11
F800: C000	chrinv	.equ 0C000	; invers/flash switch
F800: C00A	vid40	.equ 0C00A	; 40/80 col switch
F800: C00B	vid80	.equ 0C00B	
F800: C00C	vidbnk	.equ 0C00C	; video RAM switch
F800: C010	KBDSTRB	.equ 0C010	
F800: C020	TAPEOUT	.equ 0C020	
F800: C030	SPKR	.equ 0C030	
F800: C050	TXTCLR	.equ 0C050	
F800: C052	MIXCLR	.equ 0C052	
F800: C054	LOWSCR	.equ 0C054	
F800: C056	LORES	.equ 0C056	
F800: C058	TTLout0	.equ 0C058	; even: off, low <= 0.4V
F800: C05A	TTLout1	.equ 0C05A	; odd : on, high >= 2.4 V
F800: C05C	TTLout2	.equ 0C05C	
F800: C05E	TTLout3	.equ 0C05E	
F800: C060	TAPEIN	.equ 0C060	
F800: C064	PADDL0	.equ 0C064	
F800: C070	PTRIG	.equ 0C070	
F800: CFFF	CLRROM	.equ 0CFFF	
F800: E000	BASIC	.equ 0E000	
F800: E003	BASIC2	.equ 0E003	
F800:			
F800: 0080	bit7	.equ 80	
F800:			
F800: 4A	PLOT	LSR A	.include rom1
F800: 08		PHP	
F802: 20 ****		JSR GBASCALC	
F805: 28		PLP	
F806: A9 0F		LDA #0F	
F808: 90**		BCC \$1	
F80A: 69 E0		ADC #0E0	
F808: 02			
F80C: 85 2E	\$1	STA MASK	

MONITOR FILE:ROM1.TEXT

F80E1 08	PLOT1	php	
F80F1 20 ****		jsr	selbnk
F8121 4C ****		jmp	plot80
F8151 00 00 00 00		.org	0F819
F8191 20 00F8	HLINE	JSR	PLOT ; Basic HLINE
F81C1 C4 2C	\$1	CPY	H2
F81E1 B0**		BCS	RTS1
F8201 C8		INY	
F8211 20 0EF8		JSR	PLOT1
F8241 90F6		BCC	\$1
F8261 69 01	VLINE2	ADC	#1
F8281 48	VLINE	PHA	; Basic VLINE
F8291 20 00F8		JSR	PLOT
F82C1 68		PLA	
F82D1 C5 2D		CMP	V2
F82F1 90F5		BCC	VLINEZ
F81E* 11			
F8311 60	RTS1	RTS	
F8321			
F8321 A0 2F	CLRSCR	LDY	#82F ; Y-Max
F8341 D0**		BNE	CLRSC2
F8361 A0 27	CLRTOP	LDY	#27 ; Y-max, Basic GR
F834* 02			
F8381 20 ****	CLRSC2	jsr	clrsc3
F83B1 EA		nop	
F83C1 A9 00	\$1	LDA	#0
F83E1 85 30		STA	COLOR
F8401 20 28F8		JSR	VLINE
F8431 88		DEY	
F8441 10F6		BPL	\$1
F8461 60		RTS	
F8471			
F8471		.org	0F847
F803* 47F8			
F8471 48	GBASCALC	PHA	
F8481 4A		LSR	A
F8491 29 03		AND	#3
F84B1 09 04		ORA	#4 ; for LoRes Page 1
F84D1 85 27		STA	GBASH
F84F1 68		PLA	
F8501 29 18		AND	#18
F8521 90**		BCC	\$1
F8541 09 00		ora	#80
F852* 02			
F8561 85 26	\$1	STA	GBASL
F8581 0A		ASL	A
F8591 0A		ASL	A
F85A1 05 26		ORA	GBASL
F85C1 85 26		STA	GBASL
F85E1 60		RTS	
F85F1			
F85F1 A5 30	nxcol	LDA	COLOR
F8611 18		CLC	
F8621 69 03		ADC	#3
F8641 29 0F	SETCOL	AND	#0F ; Basic COLOR=

MONITOR FILE:ROM1.TEXT

F866: 85 30		STA	COLOR
F868: 0A		ASL	A
F869: 0A		ASL	A
F86A: 0A		ASL	A
F86B: 0A		ASL	A
F86C: 05 30		ORA	COLOR
F86E: 85 30		STA	COLOR
F870: 60		RTS	
F871:			
F871: 4A	SCRN	LSR	A ; Basic SCRNX,Y) function
F872: 08		PHP	
F873: 20 ****		jsr	scrn80
F876: EA		nop	
F877: EA		nop	
F878: 28		PLP	
F879: 90**	scrn2	BCC	\$1
F87B: 4A		LSR	A
F87C: 4A		LSR	A
F87D: 4A		LSR	A
F87E: 4A		LSR	A
F879* 04			
F87F: 29 0F	\$1	AND	#0F
F881: 60		RTS	
F882:			

## MONITOR FILE:ROM1.TEXT

F8821		.page	
F8821		.ORG	0F882
F8821 A6 3A	INSDS1	LDX	PCL
F8841 A4 3B		LDY	PCH
F8861 20 ****		JSR	PRYX2
F8891 20 ****		JSR	PRBLNK
F88C1 A1 3A	INSDS2	LDA	2PCL,X
F88E1 A8		TAY	
F88F1 4A		LSR	A
F8901 90**		BCC	IEVEN
F8921 6A		ROR	A
F8931 B0**		BCS	ERR ; all xxxxxxx1 opcodes are illegal
F8951 C9 A2		CMP	#0A2 ; no STA # operation
F8971 F0**		BEQ	ERR
F8991 29 87		AND	#87
F890* 09			
F89B1 4A	IEVEN	LSR	A
F89C1 AA		TAX	
F89D1 BD ****		LDA	FMT1,X
F8A01 20 79F8		JSR	SCRN2
F8A31 D0**		BNE	GETFMT
F897* 0C			
F893* 10			
F8A51 A8 80	ERR	LDY	#80
F8A71 A9 00		LDA	#0
F8A3* 04			
F8A91 AA	GETFMT	TAX	
F8AA1 BD ****		LDA	FMT2,X
F8AD1 85 2E		STA	FORMAT
F8AF1 29 03		AND	#3
F8B11 85 2F		STA	LENGTH
F8B31 98		TYA	
F8B41 29 8F		AND	#8F
F8B61 AA		TAX	
F8B71 98		TYA	
F8B81 A8 03		LDY	#3
F8B91 E0 8A		CPX	#8A
F8BC1 F0**		BEQ	MNNDX3
F8BE1 4A	MNNDX1	LSR	A
F8BF1 90**		BCC	MNNDX3
F8C11 4A		LSR	A
F8C21 4A	MNNDX2	LSR	A
F8C31 09 20		ORA	#20
F8C51 88		DEY	
F8C61 D0FA		BNE	MNNDX2
F8C81 C8		INY	
F8BF* 08			
F8BC* 0B			
F8C91 88	MNNDX3	DEY	
F8CA1 D0F2		BNE	MNNDX1
F8CC1 60		RTS	

MONITOR FILE:ROM1.TEXT

F8CD:		.page
F8CD: 00 00 00		.org 0F8D0
F8D0: 20 82F8	INSTDSP	JSR INSDS1
F8D3: 48		PHA
F8D4: B1 3A	PRNTOP	LDA #PCL, Y
F8D6: 20 ****		JSR PRBYTE
F8D9: A2 01		LDX #1
F8DB: 20 ****	PRNTBL	JSR PRBL2
F8DE: C4 2F		CPY LENGTH
F8E0: C8		INY
F8E1: 90F1		BCC PRNTOP
F8E3: A2 03		LDX #3
F8E5: C0 04		CPY #4
F8E7: 90F2		BCC PRNTBL
F8E9: 68		PLA
F8EA: A8		TAY
F8EB: B9 ****		LDA MNEML,Y ; print 3 characters, packed in 2 bytes
F8EE: 85 2C		STA MNEM
F8F0: B9 ****		LDA MNEMR,Y
F8F3: 85 2D		STA MNEM
F8F5:		
F8F5: A9 00	\$0	LDA #0
F8F7: A0 05		LDY #5 ; shift 5 bits
F8F9: 06 2D	\$1	ASL RMNEM
F8FB: 26 2C		ROL LMNEM
F8FD: 2A		ROL A
F8FE: 88		DEY
F8FF: D0F8		BNE \$1
F901: 69 BF		ADC #0BF ; "?"
F903: 20 ****		JSR COUT
F906: CA		DEX
F907: D0EC		BNE \$0
F909:		
F909: 20 ****		JSR PRBLNK
F90C: A4 2F		LDY LENGTH
F90E: A2 06		LDX #6
F910: E0 03	PRADR1	CPX #3
F912: F0**		BEQ PRADR5
F914: 06 2E	PRADR2	ASL FORMAT
F916: 90**		BCC \$0
F918: BD ****		LDA CHAR1-1,X
F91B: 20 ****		JSR COUT
F91E: BD ****		LDA CHAR2-1,X
F921: F0**		BEQ \$0 ; no 2nd char
F923: 20 ****		JSR COUT
F921*: 03		
F916*: 0E		
F926: CA	\$0	DEX
F927: D0E7		BNE PRADR1 ; next format bit
F929: 60		RTS
F92A: 88	PRADR4	DEY
F92B: 30E7		BMI PRADR2
F92D: 20 ****		JSR PRBYTE
F912*: 1C		

MONITOR FILE:ROM1.TEXT

F938: A5 2E	PRADR5	LDA	FORMAT
F932: C9 E8		CMP	#0E8
F934: B1 3A		LDA	#PCL,Y
F936: 90F2		BCC	PRADR4
F938: 20 ****	RELADR	JSR	PCADJ3
F93B: AA		TAX	
F93C: E8		INX	
F93D: D8**		BNE	PRNTYX
F93F: C8		INY	
F93D* 01			
F940: 98	PRNTYX	TYA	
F941: 20 ****	PRNTAX	JSR	PRBYTE
F944: 8A	PRNTX	TXA	
F945: 4C ****		JMP	PRBYTE
F948:			
F98A*: 48F9			
F88A*: 48F9			
F948: A2 03	PRBLNK	LDX	#3
F8DC*: 4AF9			
F94A: A9 A8	PRBL2	LDA	#8A8 ; "
F94C: 20 ****	PRBL3	JSR	COUT
F94F: CA		DEX	
F950: D8F8		BNE	PRBL2
F952: 68		RTS	
F953:			
F953: 38	PCADJ	SEC	
F954: A5 2F	PCADJ2	LDA	LENGTH
F939*: 56F9			
F956: A4 3B	PCADJ3	LDY	PCH
F958: AA		TAX	
F959: 10**		BPL	PCADJ4
F95B: 88		DEY	
F959*: 01			
F95C: 65 3A	PCADJ4	ADC	PCL
F95E: 98**		BCC	RTS2
F960: C8		INY	
F95E*: 01			
F961: 68	RTS2	RTS	
F962:			

## MONITOR FILE:ROM1.TEXT

```

F962:          .page
F962:      ; FMT1: 128 (dec) 4-bit pointer to the FMT2 table for xxxx xx00 opcodes
F962:      ;       16 (dec) 4-bit pointer to the FMT2 table for xxxx xx01 opcodes
F962:
F89E* 62F9
F962:          FMT1
F962: 04 20 54 30 0D 80 04    .byte 004,028,054,030,00D,080,004,090
F969: 90
F96A: 03 22 54 33 0D 80 04    .byte 003,022,054,033,00D,080,004,090
F971: 90
F972: 04 20 54 33 0D 80 04    .byte 004,028,054,033,00D,080,004,090
F979: 90
F97A: 04 20 54 3B 0D 80 04    .byte 004,028,054,03B,00D,080,004,090
F981: 90
F982: 00 22 44 33 0D C8 44    .byte 000,022,044,033,00D,0C8,044,000
F989: 00
F98A: 11 22 44 33 0D C8 44    .byte 011,022,044,033,00D,0C8,044,0A9
F991: A9
F992: 01 22 44 33 0D 80 04    .byte 001,022,044,033,00D,080,004,090
F999: 90
F99A: 01 22 44 33 0D 80 04    .byte 001,022,044,033,00D,080,004,090
F9A1: 90
F9A2:          ; xxxx xx01 class:
F9A2: 26 31 87 9A    .byte 026,031,087,09A ; ORA,AND,EOR,ADC,STA,LDA,CMD,SBC
F9A6:
F9A6:          ; FMT2 bit 0..1 : instruction length-1
F9A6:          ; FMT2 bit 7..2 : if bit[i] then <print chr1[i-2],chr2[i-2]>
F9A6:
F8AB* A6F9
F9A6: 00          FMT2  .byte 00      ; illegal opcode
F9A7: 21          .byte 21      ; #$hh
F9A8: 81          .byte 81      ; $hh
F9A9: 82          .byte 82      ; $dddd
F9AA: 00          .byte 00      ;
F9AB: 00          .byte 00      ;
F9AC: 59          .byte 59      ; ($hh,X)
F9AD: 4D          .byte 4D      ; ($hh),Y
F9AE: 91          .byte 91      ; $hh,X
F9AF: 92          .byte 92      ; $hhhh,X
F9B0: 86          .byte 86      ; $hhhh,Y
F9B1: 4A          .byte 4A      ; ($hhhh)
F9B2: 85          .byte 85      ; $hh,Y
F9B3: 9D          .byte 9D      ; $hhhh special case: relative
F9B4:
F9B4:          .org 0F9b4      ; char1/char2 used by mini assembler
F919* B3F9
F9B4: AC A9 AC A3 A8 A4    CHAR1 .byte 0AC,0A9,0AC,0A3,0A8,0A4      ; ",),#($"
F91F* B9F9
F9B4: D9 00 D8 A4 A4 00    CHAR2 .byte 0D9,000,0D8,0A4,0A4,000      ; "Y X$$ "
F9C0:
F8EC* C0F9
F9C0:          MNML
F9C0:          ;
F9C0:          ; IIIII000:

```

## MONITOR FILE:ROM1.TEXT

```

F9C0: 1C          .byte 01C      ; BRK
F9C1: 8A          .byte 08A      ; PHP
F9C2: 1C          .byte 01C      ; BPL
F9C3: 23          .byte 023      ; CLC
F9C4: 5D          .byte 05D      ; JSR
F9C5: 8B          .byte 08B      ; PLP
F9C6: 1B          .byte 01B      ; BMI
F9C7: A1          .byte 0A1      ; SEC
F9C8: 9D          .byte 09D      ; RTI
F9C9: 8A          .byte 08A      ; PHA
F9CA: 1D          .byte 01D      ; BVC
F9CB: 23          .byte 023      ; CLI
F9CC: 9D          .byte 09D      ; RTS
F9CD: 8B          .byte 08B      ; PLA
F9CE: 1D          .byte 01D      ; BVS
F9CF: A1          .byte 0A1      ; SEI
F9D0: 00          .byte 000      ; ?
F9D1: 29          .byte 029      ; DEY
F9D2: 19          .byte 019      ; BCC
F9D3: AE          .byte 0AE      ; TYA
F9D4: 69          .byte 069      ; LDY
F9D5: A8          .byte 0A8      ; TAY
F9D6: 19          .byte 019      ; BCS
F9D7: 23          .byte 023      ; CLV
F9D8: 24          .byte 024      ; CPY
F9D9: 53          .byte 053      ; IBY
F9DA: 1B          .byte 01B      ; BNE
F9DB: 23          .byte 023      ; CLD
F9DC: 24          .byte 024      ; CPX
F9DD: 53          .byte 053      ; INX
F9DE: 19          .byte 019      ; BEQ
F9DF: A1          .byte 0A1      ; SED
F9E0:
F9E0: ; IIIxxx100:
F9E0: 00          .byte 000      ; ?
F9E1: 1A          .byte 01A      ; BIT
F9E2: 5B          .byte 05B      ; JMP
F9E3: 5B          .byte 05B      ; JMP
F9E4: A5          .byte 0A5      ; STY
F9E5: 69          .byte 069      ; LDY
F9E6: 24          .byte 024      ; CPY
F9E7: 24          .byte 024      ; CPX
F9E8:
F9E8: ; IIII1010:
F9E8: AE          .byte 0AE      ; TXA
F9E9: AE          .byte 0AE      ; TXS
F9EA: A8          .byte 0A8      ; TAX
F9EB: AD          .byte 0AD      ; TSX
F9EC: 29          .byte 029      ; DEX
F9ED: 00          .byte 000      ; ?
F9EE: 7C          .byte 07C      ; NOP
F9EF: 00          .byte 000      ; ?
F9F0:
F9F0: ; 0IIxxx10:
F9F0: 15          .byte 015      ; ASL

```

## MONITOR FILE:ROM1.TEXT

```

F9F1: 9C          .byte 09C      ; ROL
F9F2: 6D          .byte 06D      ; LSR
F9F3: 9C          .byte 09C      ; ROR
F9F4:
F9F4:             ; IIIx0010, IIIx0110, IIIx1110:
F9F4: A5          .byte 0A5      ; STX
F9F5: 69          .byte 069      ; LDX
F9F6: 29          .byte 029      ; DEC
F9F7: 53          .byte 053      ; INC
F9F8:
F9F8:             ; IIIxxx01:
F9F8: 84          .byte 084      ; ORA
F9F9: 13          .byte 013      ; AND
F9FA: 34          .byte 034      ; EOR
F9FB: 11          .byte 011      ; ADC
F9FC: A5          .byte 0A5      ; STA
F9FD: 69          .byte 069      ; LDA
F9FE: 23          .byte 023      ; CMP
F9FF: A0          .byte 0A0      ; SBC
FA00:
FA01* 00FA
FA00: D8 62 5A 48 26 62 94 MNEMR .byte 0D8, 062, 05A, 048, 026, 062, 094, 088
FA07: 88
FA08: 54 44 C8 54 68 44 E8     .byte 054, 044, 0C8, 054, 068, 044, 0E8, 094
FA0F: 94
FA10: 00 B4 88 84 74 B4 28     .byte 000, 0B4, 008, 084, 074, 0B4, 028, 06E
FA17: 6E
FA18: 74 F4 CC 4A 72 F2 A4     .byte 074, 0F4, 0CC, 04A, 072, 0F2, 0A4, 08A
FA1F: 8A
FA20:
FA20: 00 AA A2 A2 74 74 74     .byte 000, 0aa, 0a2, 0a2, 074, 074, 074, 72
FA27: 72
FA28:
FA28: 44 68 B2 32 B2 00 22     .byte 044, 068, 0b2, 032, 0b2, 0, 022, 0
FA2F: 00
FA30:
FA30: 1A 1A 26 26              .byte 1a, 1a, 26, 26
FA34: 72 72 88 C8              .byte 72, 72, 88, 0c8
FA38:
FA38: C4 CA 26 48 44 44 A2     .byte 0c4, 0ca, 026, 048, 044, 044, 0a2, 0c8
FA3F: C8
FA40:
FA40:
FA40:
FA40:             .include rom2

```

## MONITOR FILE:ROM2.TEXT

```

FA40:           .page      ; filer ROM2.text
FA40:           .org       0FA40
FA40: 85 45     irq       sta acc
FA42: 68        pla
FA43: 48        pha
FA44: 29 10     and #10    ; test break flag, bit 4
FA46: D0**     bne break
FA48: 6C FE03   jmp #irqloc
FA4B:
FA4B: 00        .org       0FA4C
FA46* 84
FA4C1 28       break    plp
FA4D: 20 ****   jsr      sav1
FA50: 68        pla
FA51: 85 3A     sta pcl
FA53: 68        pla
FA54: 85 3B     sta pch
FA56: 6C F003   jmp #brkv
FA59:
FA59: 28 82F8   oldbrk  jsr insds1
FA5C: 20 ****   jsr rgdsp1
FA5F: 4C ****   jmp mon
FA62:
FA62:           .org       0FA62
FA62: D8        reset    cld
FA63: 20 ****   jsr setnorm
FA66: 20 ****   jsr init
FA69: 20 ****   jsr setvid
FA6C: 20 ****   jsr setkbd
FA6F:
FA6F: D8        newmon  cld
FA70: 20 ****   jsr bell
FA73:           .if      chrfont=1      ; national
FA73: 8D 03C0   sta chrgen0+1
FA76: 8D 04C0   sta chrgen1
FA79:
FA79:           .endc
FA79:           .if      chrfont=2      ; ASCII
FA79:           .endc
FA79:           .if      chrfont=3      ; APL
FA79:           .endc
FA79:
FA79: 8D 00C0   sta chrinv
FA7C: 2C FFCF   bit clrROM
FA7F: 2C 10C0   bit Kbdstrb
FA82: AD F303   lda softev+1
FA85: 49 A5     eor #0a5
FA87: CD F403   cmp pwredup
FA8A: D0**     bne pwrunp
FA8C: AD F203   lda softev
FA8F: D0**     bne nofix
FA91: A9 E0     lda #0e0
FA93: CD F303   cmp softev+1
FA96: D0**     bne nofix
FA98:

```

## MONITOR FILE:ROM2.TEXT

FA981 A8 03	fixsev	ldy	#3
FA9A1 8C F203		STY	SOFTEV
FA9D1 4C 00E0		JMP	BASIC
FA96* 08			
FA8F* 0F			
FAA01 6C F203	NOFIX	JMP	@SOFTEV
FAA31			
FA8A* 17			
FAA31 38	PWRUP	sec	
FAA41 6E F904		ror	switch
FAA71 20 ****		JSR	LOGO1
FAAA1 A2 05	SETPG3	LDX	#5
FAAC1 BD ****	SETPLP	LDA	PWRCON-1,X
FAAF1 9D EF03		STA	BRKV-1,X
FAB21 CA		DEX	
FAB31 D0F7		BNE	SETPLP
FAB51 A9 C8		LDA	#0C8 ; last slot+1
FAB71 85 01		STA	LOC1 ; SET PTR H
FAB91 86 00		STX	LOC0 ; Xreg=0
FABB1 A0 07	SLOOP	LDY	#7 ; Y is byte offset into the \$lot ROM
FABD1 C6 01		DEC	LOC1
FABF1 A5 01		LDA	LOC1
FAC11 C9 C1		CMP	#0C1 ; slot=1?
FAC31 F0D3		BEQ	FIXSEV ; yes, slot 1 is the builtin printer
FAC51 8D F807		STA	MSLOT
FAC81 B1 00	\$0	LDA	@LOC0,Y ; read slot ROM
FACA1 D9 ****		CMP	DISKID,Y ; is it a boot device (floppy, harddisk...) ??
FACD1 D0EC		BNE	SLOOP ; no, test next slot
FACF1 88		DEY	
FAD01 88		DEY	; yes so check next odd byte
FAD11 10F5		BPL	\$0
FAD31 6C 0000		jmp	@LOC0 ; it is a disk! jump to boot

MONITOR FILE:ROM2.TEXT

```

FAD61
FAD61 00 .page
FAD71 20 **** .ORG 0FAD7
FA5D* DAFA REGDSP JSR CROUT
FADA1 A9 45 RGDSP1 LDA #ACC
FADC1 85 40 STA A3L
FADE1 A9 00 LDA #0
FAE01 85 41 STA A3H
FAE21 A2 FB LDX #0FB ; -5
FAE41 A9 A0 $1 LDA #0A8
FAE61 20 **** JSR COUT
FAE91 BD **** LDA RTBL-251., X
FAEC1 20 **** JSR COUT
FAEF1 A9 BD LDA #0BD ; "="
FAF11 20 **** JSR COUT
FAF41 B5 4A LDA ACC+5, X
FAF61 20 **** JSR PRBYTE
FAF91 E8 INX
FAFA1 30E8 BMI $1
FAFC1 60 RTS
FAFD1
FAFD2
FAAD* FCFA
FAFD1 59FA pwrcon .word OLDBRK
FAFF1 00E0 .word basic
FB011
FACB* 01FB
FB011 45 20 diskid eor 20 ; opcode (0E0^0A5=45) used for mask!!
FB031 A8 00 ldy #0 ; code never executed,
FB051 A2 03 idx #3 ; only for disk ID
FB071 86 3C stx 3c
FB091
FB091 08 15 0A 0B 40 0E 0F locchr .byte 08,15,0a,0b,40,0e,0f
FB101
FB101 D8** sw1 bne sw2
FB121 0A asl a
FB10* 01
FB131 8D 7904 sw2 sta chy
FB161 4C **** jmp scr180
FB191
FB191 .ORG 0FB19
FAE4* 1EFA
FB191 C1 D8 D9 D8 D3 RTBL .byte 0C1,0D8,0D9,0D8,0D3 ; "AXYPS"
FB1E1
FB1E1 AD 70C0 PREAD LDA PTRIG ; Basic PDL(n) function
FB211 A0 00 LDY #0
FB231 EA NOP
FB241 EA NOP
FB251 BD 64C8 $1 LDA PADDL0,X
FB281 10** BPL $2
FB2A1 C8 INY
FB2B1 D0F8 BNE $1
FB2D1 88 DEY
FB28* 04

```

## MONITOR FILE:ROM2.TEXT

```

FB2E: 60          $2      RTS
FB2F:
FA67* 2FFB
FB2F: A9 04      INIT    LDA    #4      ; set I flag!
FB31: 85 48      STA     STATUS
FB33: 2C 56C0    BIT     LORES
FB36: 2C 54C0    bit    lowscr
FB39: 2C 51C0    SETTXT  BIT    TxtClr+1   ; set text mode, Basic TEXT
FB3C: A9 00      LDA    #0
FB3E: F0**      BEQ    SETWND
FB40: 2C 50C0    SETGR   BIT    TXTCLR    ; set graphic, Basic GR
FB43: 2C 53C0    BIT    MIXclr+1   ; set mixed mode
FB46: 20 36F8    JSR    CLRTOP
FB49: A9 14      LDA    #14
FB3E* 0B
FB4B: 85 22      SETWND  STA    WNDTOP
FB4D: A9 00      lda    #0
FB4F: 85 28      sta    wndlft
FB51: A9 50      LDA    #width
FB53: 85 21      STA    WNDWDTH
FB55: A9 18      LDA    #18
FB57: 85 23      STA    WNDBTM
FB59: A9 17      LDA    #17
FB5B: 85 25      TABV   STA    CV
FB5D: 4C ****    JMP    VTAB
FB60:
FB60: 20 ****    LOGO   JSR    HOME    ; CLEAR THE SCRN
FB63: A8 08      LDY    #8
FB65: B9 ****    $1     LDA    TITLE, Y ; GET A CHAR
FB68: 99 0004    STA    LINE1, Y
FB6B: 88          DEY
FB6C: 10F7          bpl   $1
FB6E: 60          RTS
FB6F:
FB6F:           .org  0FB6F
FB6F: AD F303    SETPWRC LDA    SOFTEV+1
FB72: 49 A5      EOR    #0A5
FB74: 8D F403    STA    PWREDUP
FB77: 60          RTS
FB78:
FB78: AC 00C0    VIDWAIT LDY    KBD
FB7B: C0 93      CPY    #93    ; ctrl-S pressed?
FB7D: D0**      BNE    $2      ; no so continue
FB7F: 2C 10C0    BIT    KBDSTRB ; clear keyboard strobe
FB82: AC 00C0    $1     LDY    KBD    ; wait until next key pressed
FB85: 10FB          BPL   $1
FB87: C0 83      CPY    #83    ; ctrl-C?
FB89: F0**      BEQ    vidout ; yes, it is for Basic
FB8B: 8D 10C0    sta    KBDSTRB ; clear strobe
FB7D* 0F
FB8E: D0**      $2     bne    VIDOUT ; display char in accu
FB90:
FB66* 90FB
FB90: C2 E1 F3 E9 F3 A0 B1 TITLE .byte  0C2,0E1,0F3,0E9,0F3,0A0,0B1,0B0,0B8 ; "Basis 108"
FB97: B0 B8

```

## MONITOR FILE:ROM2.TEXT

```

FB99I
FB99I 0F 3E 65 19 57 9B 41 locjmp .byte 0F,3E,65,19,57,9b,41
FBA0I A0 07 local ldy #7
FBA2I D9 09FB $1 cmp locchr,y
FBA5I D0** bne $2
FBA7I A9 FC lda #0fc
FBA9I 48 pha
FBAAI B9 99FB lda locjmp,y
FBADl 48 pha
FBAEI A0 18 ldy #18
FBB0I D0** bne pip ; echo for legal keys
FBA5* 0B
FBB2I 88 $2 dey
FBB3I 10ED bpl $1
FBB5I 60 rts
FBB6I 20 A0FB jlocal jsr local
FBB9I 20 **** rdchar1 jsr rdKey
FBBCI 29 FF and #0ff ; test bit 7
FBBEI 10F6 bpl jlocal
FBC0I 60 rts
FBC1I
FBC1I .ORG 0FBC1
FBC1I 48 BASCALC PHA
FBC2I 4A LSR A
FBC3I 29 03 AND #3
FBC5I 09 04 ORA #4 ; for text page 1
FBC7I 85 29 STA BASH
FBC9I 68 PLA
FBCAI 29 18 AND #18
FBCCI 90** BCC $1
FBCEI 09 80 ora #80
FBCC* 02
FBD0I 85 28 $1 STA BASL
FBD2I 0A ASL A
FBD3I 0A ASL A
FBD4I 05 28 DRA BASL
FBD6I 85 28 STA BASL
FBD8I 60 RTS
FBD9I
FBD9I C9 87 BELL1 CMP #87
FBDBI D0** BNE noctrl
FBDDI A0 78 LDY #070 ; new sound
FBB0* 2D
FBDFI 98 pip tya ; another sound
FBE0I 4A lsr A
FBE1I 4A lsr A
FBE2I 09 07 ora #7 ; set minimum time
FBE4I 20 **** jsr WAIT
FBE7I 2C 30C0 bit SPKR
FBEAI 88 dey
FBEBI D0F2 bne pip
FBDB* 10
FBEDI 60 noctrl rts
FBEEI .org 0FBEE

```

## MONITOR FILE:ROM2.TEXT

FBEE: 25 32		storing and	invflg
FBF0: 20 ****		STORADV jsr	stor80
FBF3: EA		nop	
FBF4: E6 24		ADVANCE INC	CH
FBF6: A4 24		1dy	CH
FBF8: C4 21		cpy	WNDWDTH
FBFA: B0**		BCS	CR
FBFC: 60		RTS	
FBFD:		.org	0FBFD
FB8E* 60			
FB89* 72			
FBFD: C9 A8	VIDOUT	cmp	#8A0 ; ctrl?
FBFF: B0ED		bcs	storing ; no, display it normal or inverse
FC01: A8		tay	
FC02: 10EC		bp1	STORADV
FC04: C9 8D		CMP	#8D
FC06: F0**		BEQ	CR
FC08: C9 8A		CMP	#8A
FC0A: F0**		BEQ	LF
FC0C: C9 88		CMP	#88
FC0E: D0C9		BNE	BELL1
FC10: C6 24	BS	DEC	CH
FC12: 10**		BPL	RTS4
FC14: A5 21		LDA	WNDWDTH
FC16: 85 24		STA	CH
FC18: C6 24		DEC	CH
FC1A: A5 22	UP	LDA	WNDTOP
FC1C: C5 25		CMP	CV
FC1E: B0**		BCS	RTS4
FC20: C6 25		DEC	CV
FB8E* 22FC			
FC22: A5 25	VTAB	LDA	CV
FC24: 20 C1FB	VTABZ	JSR	BASCALC
FC27: 4C ****		jmp	vtab80
FC1E* 8A			
FC12* 16			
FC2A: 60	RTS4	RTS	
FC2B:			
FC2B: B9 0002	getupcs	lda	in,y ; read uppercase char from input buffer
FC2E: C8		iny	
FC2F: C9 E0	upper	cmp	#0E0 ;
FC31: 90**		bcc	\$1
FC33: 29 DF		and	#0DF ; shift to uppercase
FC31* 02			
FC35: 60	\$1	rts	
FC36:			
FC36: 48	sw5	pha	
FC37: 98	sw0	tya	
FC38: 44		lsr	a
FC39: 8D 08C0		sta	vid80
FC3C: 4C ****		jmp	selbnk2
FC3F:		.org	0FC3F
FC3F: 4C F4FB		jmp	advance ; cursor right jmp

MONITOR FILE:ROM2.TEXT

FC42:	.ORG	0FC42
FC42: A4 24	CLREOP	LDY CH
FC44: A5 25		LDA CV
FC46: 48	CLEOP1	PHA
FC47: 20 24FC		JSR VTABZ
FC4A: 20 ****		JSR CLEOLZ
FC4D: A0 00		LDY #0
FC4F: 68		PLA
FC50: 69 00		ADC #0 ; carry=1 from cleolz
FC52: C5 23		CMP WNDBTM
FC54: 90F0		BCC CLEOP1
FC56: B0CA		BCS VTAB
FB61* 58FC		
FC58: A5 22	HOME	LDA WNDTOP
FC5A: 85 25		STA CV
FC5C: A0 00		LDY #0
FC5E: 84 24		STY CH
FC60: F0E4		BEQ CLEOP1
FC62:		

## MONITOR FILE:ROM2.TEXT

		.page
FC62I		
FC06* 5A		
FBFA* 66		
FC62I A9 00	CR	LDA #0
FC64I 85 24		STA CH
FC0A* 5A		
FC66I E6 25	LF	INC CV
FC68I A5 25		lda cv
FC6AI C5 23		cmp wndbtm
FC6CI 90B6		bcc vtabz
FC6EI C6 25		dec cv
FC70I A5 22	scroll	lda wndtop
FC72I 48		PHA
FC73I 20 24FC		JSR VTABZ
FC76I A5 28	\$1	LDA BASL
FC78I 85 2A		STA BAS2L
FC7AI A5 29		LDA BASH
FC7CI 85 2B		STA BAS2H
FC7EI A4 21		LDY WNDWDTH
FC80I 88		DEY
FC81I 68		PLA
FC82I 69 01		ADC #1 ; carry=0 from scroll line
FC84I C5 23		CMP WNDBTM
FC86I B0**		BCS \$3
FC88I 48		PHA
FC89I 20 24FC		JSR VTABZ
FC8CI 98		tya
FC8DI AC F904		ldy switch
FC90I 20 10FB		jsr sw1 ; on return carry=0
FC93I 90E1		bcc \$1 ; bra \$1
FC95I		
FC95I		.org 0FC95
FC86* 0D		
FC95I A0 00	\$3	LDY #0
FC97I 20 ****		JSR CLEOLZ
FC9AI B086		BCS VTAB
FC9CI A4 24	CLREOL	LDY CH
FC98* 9EFC		
FC4B* 9EFC		
FC9EI 38	CLEOLZ	sec ; carry=1 after plp
FC9FI 08		php
FCA0I 4C ****		jmp cleol80

MONITOR FILE:ROM2.TEXT

```
FCA3:                                .page
FCA3: 00 00 00 00 00                  .org  0FCAB
FBE5* A8FC
FCA8: 38          WAIT   SEC      ; wait for ord(Accu^2) time
FCA9: 48          $1      PHA
FCAA: E9 01        $2      SBC    #1
FCAC: D0FC        BNE    $2
FCAE: 68          PLA
FCAF: E9 01        SBC    #1
FCB1: D0F6        BNE    $1
FCB3: 68          RTS
FCB4:
FCB4: E6 42        NXTA4  INC     A4L
FCB6: D0**        BNE    NXTA1
FCB8: E6 43        INC     A4H
FCB6* 02
FCBA: A5 3C        NXTA1  LDA     A1L
FCBC: C5 3E        CMP     A2L
FCBE: A5 3D        LDA     A1H
FCC0: E5 3F        SBC     A2H
FCC2: E6 3C        INC     A1L
FCC4: D0**        BNE    $2
FCC6: E6 3D        INC     A1H
FCC4* 02
FCC8: 60          $2      RTS
```

## MONITOR FILE:ROM2.TEXT

```

FCC9I .page
FCC9I ;
FCC9I ; 80-col screen driver
FCC9I ;
FCC9I 4C **** selbnk jmp sw3
FC3D* CCFC
FCCC1 8D 0CC0 selbnk2 sta vidbnk ; 400..BFF: dynamic RAM
FCCF1 90** bcc $1
FCD1I 78 sei
FCD2I 8D 0DC0 sta vidbnk+1; 400..BFF: static RAM
FCCF* 84
FCD5I 8C 7904 $1 sty chy ; save Yreg in active bank!
FCD8I A8 tay ; for lda/sta @basl,y
FCD9I 68 pla
FCDAI 60 rts
FCDBI
FCA1* DBFC
FCDBI 20 C9FC cleol80 jsr selbnk ; clear to end of line
FCDEI A9 A0 lda #0A0
FCE0I 91 28 sta @basl,Y
FCE2I AC 7904 ldy chy
FCE5I C8 iny
FCE6I C4 21 cpy wndwdth
FCE8I 90F1 bcc cleol80
FCEAI 4C **** jmp vidplp
FCEDI
FCEDI B1 26 plot80 lda @gbasl,y ; MiRes plot
FCEF1 45 30 eor color
FCF1I 25 2E and mask
FCF3I 51 26 eor @gbasl,y
FCF5I 91 26 sta @gbasl,y
FCF7I 4C **** jmp vidrts
FCFAI
FCCA* FAFC
FCFAI 48 sw3 pha
FCFB1 AD F904 lda switch
FCFEI F0** beq sw4
FD00I 4C 37FC jmp sw6
FCFE* 03
FD03I 68 sw4 pla
FD04I 8C 7904 sty chy
FD07I 8D BAC0 sta vid40
FD0AI 68 rts
FD0BI

```

## MONITOR FILE:ROM2.TEXT

```
FD0B:          .page
FD0B: 00          .org  0FD0C
FBB0* 0CFD
FD0C: 4C ****      RDKEY  jmp   rdkey2
FAA8* 0FFD
FD0F: 20 2FFB      logo1  jsr    init
FD12: 4C 60FB      jmp    logo
FD0D* 15FD
FD15: 20 ****      rdkey2 jsr    curs80
FD18:
FD18: 6C 3800      .org  0FD18
FD18: jmp    0ksw1
FD1B:
FD1B:          .org  0FD1B
FD1B: E6 4E          KEYIN INC    RNDL ; slow human is the random generator
FD1D: D0**          BNE   $1
FD1F: E6 4F          INC    RNDH
FD1D* 02
FD21: 2C 00C0      $1    BIT    KBD   ; key pressed?
FD24: 10F5          BPL    KEYIN
FD26: 20 ****      jsr    curs80 ; remove cursor
FD29: AD 08C0      lda    Kbdextn ; read function key bit
FD2C: 29 88          and   #bit7
FD2E: 4D 00C0      eor    KBD   ; merge with ASCII code
FD31: 8D 10C0      sta    KBDSTRB
FD34: 60          rts
FD35:
FD35:          .org  0FD35
FD35: 4C B9FB      RDCHAR jmp   rdchar1
FD38:
```

## MONITOR FILE:ROM2.TEXT

FD38!		.page	
FD38: 00 00 00 00 00		.org	0FD3D
FD3D: A5 32	NOTCR	LDA	INVFLG
FD3F: 48		PHA	
FD40: A9 FF		LDA	#0FF
FD42: 85 32		STA	INVFLG
FD44: BD 0002		LDA	IN,X
FD47: 20 ****		JSR	COUT
FD4A: 68		PLA	
FD4B: 85 32		STA	INVFLG
FD4D: BD 0002		LDA	IN,X
FD50: C9 08		CMP	#088 ; ctrl-H
FD52: F8**		BEQ	BCKSPC
FD54: C9 98		CMP	#098 ; ctrl-X
FD56: F0**		BEQ	CANCEL
FD58: E0 F8		CPX	#0F8
FD5A: 90**		BCC	NOTCR1
FD5C: 20 ****		JSR	BELL
FD5A*: 03			
FD5F: E8	NOTCR1	INX	
FD60: D0**		BNE	NXTCHAR
FD56*: 8A			
FD62: A9 A3	CANCEL	LDA	#0A3 ; "#" like MBasic 5.2
FD64: 20 ****		JSR	COUT
FD67: 20 ****	GETLN2	JSR	CRDUT
FD6A: A5 33	GETLN	LDA	PROMPT
FD6C: 20 ****		JSR	COUT
FD6F: A2 01		LDX	#1
FD52*: 1D			
FD71: 8A	BCKSPC	TXA	
FD72: F0F3		BEQ	GETLN2
FD74: CA		DEX	
FD60*: 13			
FD75: 20 35FD	NXTCHAR	JSR	RDCHAR
FD78: C9 95		CMP	#95 ; ctrl-U
FD7A: D0**		BNE	ADDINP
FD7C: 20 ****		jsr	get80
FD7F: EA		nop	
FD80: EA		nop	
FD81: EA		nop	
FD82: EA		nop	
FD83: EA		nop	
FD84:		.org	0FD84
FD7A*: 08			
FD84: 9D 0002	ADDINP	STA	IN, X
FD87: C9 8D		CMP	#8D
FD89: D0B2		BNE	NOTCR
FD8B:		.ORG	0FD8B
FD8B: 20 9FCF	\$1	JSR	CLREOL ; entry by DOS 3.3 toolkit asmb!
FD68*: 8EFD			
FAD8*: BEFD			
FD8E: A9 8D	CRDUT	1da	#8d
FD90: D0**		BNE	COUT
FD92:			

## MONITOR FILE:ROM2.TEXT

FD92:		.page	
FD92:		.org	0FD92
FD92: A4 3D	pra1	ldy	a1h
FD94: A6 3C		idx	a1l
FD96: 20 ****	pryx2	jsr	newln
FD99: 20 40F9		jsr	prntyx
FD9C: A8 00		ldy	#0
FD9E: A9 BA		lda	#0BA ; :'
FDA0: 4C ****		jmp	cout
FDA3:			
FDA3: A5 3C	XAM8	LDA	A1L
FDA5: 09 0F		ora	#0f
FDA7: 85 3E		STA	A2L
FDA9: A5 3D		LDA	A1H
FDAB: 85 3F		STA	A2H
FDAD: A5 3C	MOD8CHK	LDA	A1L
FDAF: 29 0F		and	#0F
FDB1: D0**		BNE	DATAOUT
FDB3: 20 92FD	XAM	JSR	PRA1
FDB1* 03			
FDB6: A9 A0		DATAOUT	LDA #0A0
FDB8: 20 ****		JSR	COUT
FDBB: B1 3C		lda	#a1l,y
FDBD: 20 ****		jsr	prbyte
FDC8: 20 BAFC		jsr	nxtal
FDC3: 90E8		bcc	mod8chk
FDC5: 60		RTS	
FDC6:			
FDC6: AD F904	sw7	lda	switch
FDC9: F0**		beq	sw740
FDCB: A5 20		lda	wndlft
FDCD: 4A		tsr	a
FDCE: 60		rts	
FDC9* 04			
FDCF: A9 28	sw740	lda	#028
FDD1: C5 21		cmp	wndwdth
FDD3: B0**		bcs	wdthok
FDD5: 85 21		sta	wndwdth
FDD3* 02			
FDD7: A5 20	wdthok	lda	wndlft
FDD9: 60		rts	
FDDA:			
FDBE* DA0D			
FAF7* DA0D			
FDDA: 48	PRBYTE	PHA	
FDDB: 4A		LSR	A
FDDC: 4A		LSR	A
FDDD: 4A		LSR	A
FDEE: 4A		LSR	A
FDDF: 20 ****		JSR	PRHEXZ
FDE2: 68		PLA	
FDE3: 29 0F	PRHEX	AND	#0F
FDE8* E5FD			
FDE5: 09 B0	PRHEXZ	ORA	#0B0

MONITOR FILE:ROM2.TEXT

```
FDE7: C9 BA      CMP    #0BA   ; ":"  
FDE9: 90**      BCC    COUT  
FDEB: 69 86      ADC    #6     ; ":".."? -> "A".."F"  
FDED:  
FDED:          .org    0FDED  
FDE9* 02  
FDB9* EDFD  
FDA1* EDFD  
FD90* 5B  
FD6D* EDFD  
FD65* EDFD  
FD48* EDFD  
FAF2* EDFD  
FAED* EDFD  
FAE7* EDFD  
FDED: 6C 3680    COUT   JMP    @CSWL  
FDF0: 48        COUT1  PHA  
FDF1: 84 35      STY    YSAV1  
FDF3: 20 78FB    JSR    VIDWAIT  
FDF6: A4 35      LDY    YSAV1  
FDF8: 68        PLA  
FDF9: 60        RTS  
FDFA:  
FD97* FAFD  
FDFA: 20 8EFD    newln jsr    crout  
FDFD: A9 A8      lda    #0A8  
FDFF: D0EC      bne    cout  
FE01:  
FE01:  
FE01:          .include rom3
```

## MONITOR FILE:ROM3.TEXT

```

FE01:          .page
FE01:          ;
FE01:          ; monitor command page
FE01:          ;
FE01:          .org 0FE01
FE01: C6 34    BL1   DEC    YSAV
FE03: F09E    BEQ    XAM8
FE05: CA      BLANK  DEX
FE06: D0**    BNE    SETMDZ
FE08: C9 BA    CMP    #0BA   ; ":" 
FE0A: D0A7    BNE    XAM
FE0C: 85 31    STOR   STA    MODE
FE0E: A5 3E    LDA    A2L
FE10: 91 40    STA    #A3L, Y
FE12: E6 40    INC    A3L
FE14: D0**    BNE    $1
FE16: E6 41    INC    A3H
FE14* 02
FE18: 60      $1     RTS
FE19:
FE19: A4 34    SETMODE LDY    YSAV
FE1B: B9 FF01  LDA    IN-1, Y
FE06* 16
FE1E: 85 31    SETMDZ STA    MODE
FE20: 60      RTS
FE21:
FE21: A2 01    LT     LDX    #1
FE23: B5 3E    $1     LDA    A2L, X
FE25: 95 42    STA    A4L, X
FE27: CA      DEX
FE28: 10F9    BPL    $1
FE2A: 60      RTS
FE2B:
FE2B: 00
FE2C: B1 3C    MOVE   LDA    #A1L, Y
FE2E: 91 42    STA    #A4L, Y
FE30: 20 B4FC  JSR    NXTA4
FE33: 90F7    BCC    MOVE
FE35: 60      RTS
FE36:
FE36: B1 3C    verify LDA    #A1L, Y
FE38: D1 42    CMP    #A4L, Y
FE3A: F0**    BEQ    $1
FE3C: 20 92FD  JSR    PRA1
FE3F: B1 3C    LDA    #A1L, Y
FE41: 20 DAFD  JSR    PRBYTE
FE44: A9 BC    LDA    #0BC   ; "<"
FE46: 20 EDFD  JSR    COUT
FE49: A9 BE    LDA    #0BE   ; ">"
FE4B: 20 EDFD  JSR    COUT
FE4E: B1 42    LDA    #A4L, Y
FE50: 20 DAFD  JSR    PRBYTE
FE3A* 17
FE53: 20 B4FC  $1     JSR    NXTA4

```

## MONITOR FILE:ROM3.TEXT

FE56	I 90DE	BCC	verify
FE58	I 60	RTS	
FE59			
FE59		.org	0FE59
FE59	I 6C F203	BASCONT	JMP
FE5C	I 4C 00E0	XBASIC	JMP
FE5F			BASIC
FE5F	I 80	.org	0FE60
FE60	I 20 ****	LIST	jsr
		\$1	jsr
FE63	I 20 D0F8		instdsp
FE66	I 20 53F9		jsr
FE69	I 85 3A		pcaj
FE6B	I 84 3B		sta
FE6D	I C5 3E		pcl
FE6F	I 98		sty
FE70	I E5 3F		pcn
FE72	I 90EF		cmp
FE74	I 60		a21
FE75			tya
FE75			sbc
FE75*	I 75FE		a2h
FE75	I 8A	A1PC	txa
FE76	I F0**		BEQ
FE78	I B5 3C	\$1	LDA
FE7A	I 95 3A		A1L, X
FE7C	I CA		STA
FE7D	I 10F9		PCL, X
FE76*	I 07		DEX
FE7F	I 60		BPL
FE80		\$2	RTS
FE80	I A0 7F	SETINV	LDY
FE82	I D0**		#7F
FE84	I A0 FF	SETNORM	LDY
FE82*	I 02		#OFF
FE86	I 84 32	SETIFLG	STY
FE88	I 60		INVLG
FE89		RTS	
FE89	I A9 00	SETKBD	LDA
FE8B	I 85 3E	INPUT	STA
FE8D	I A2 38	INPR	LDX
FE8F	I A0 1B		#KSWL
FE91	I D0**		LDY
FE93	I A9 00	SETVID	LDA
FE95	I 85 3E	OUTPORT	STA
FE97	I A2 36	OUTPRT	LDX
FE99	I A0 F0		#CSWL
FE91*	I 08		LDY
FE9B	I A5 3E	IOPRT	LDA
FE9D	I 29 07		A2L
FE9F	I F0**	AND	#07
FEA1	I 09 C0	BEQ	; only slots 1..7 are legal
FEA3	I A0 00	ORA	IOPRT1
FEA5	I F0**		; slot 0 has no I/O ROM space
FE9F*	I 06	LDY	#iopage
FEA7	I A9 FD	BEQ	IOPRT2
FEA5*	I 02	IOPRT1	LDA
			#0FD

MONITOR FILE:ROM3.TEXT

FEA9: 94 00	IOPRT2	STY	loc0, X
FEAB: 95 01		STA	loc1, X
FEAD: A5 3E		lda	a21 ; if slot in [8..15.] then entry:=Cs08
FEAF: 29 08		and	#8 ; else entry:=Cs00
FEB1: 15 00		ora	loc0,x
FEB3: 95 00		sta	loc0,x
FEB5: 60		rts	
FEB6:			
FEB6: 20 75FE	GO	.org	0FEB6
FEB9: 20 ****		JSR	A1PC
FEBC: 6C 3A00		JSR	RESTORE
FEBF: 4C D7FA	REGZ	JMP	3PCL
FEC2:			
FEC2: 20 47F8	scrn80	jsr	gbascalc
FEC5: 4C ****		jmp	scrn802
FEC8:			
FEC8: 00 00	USR	.org	0FECA
FECA: 4C F803		JMP	USRADR

## MONITOR FILE:ROM3.TEXT

```

FEC0! .page
FEC0!
FEC0! 60 write rts ; no tape out!
FECE!
FECE! 88 stor80 php
FECF! A4 24 ldy ch
FED1! 20 C9FC jsr selbnk
FED4! 4C **** jmp strts
FED7!
FD27* D7FE
FD16* D7FE
FED7! 08 curs80 php
FED8! A4 24 ldy ch
FEDA! 20 C9FC jsr selbnk
FEDD! B1 28 lda #bas1,y
FEDF! 49 80 eor #bit7
FED5* E1FE
FEE1! 91 28 strts sta #bas1,y ; write char,
FCF8* E3FE
FEE3! AC 7904 vidrts ldy chy ; restore Yreg,
FCEB* E6FE
FEE6! 8D 0CC0 vidplp sta vidbnk ; restore memory bank,
FEE9! 28 plp ; restore Iflag
FEEA! 60 rts
FEEB!
FC28* EBFE
FEEB! 20 C6FD vtab80 jsr sw7
FEEE! 18 clc
FEEF! 65 28 adc bas1
FEF1! 85 28 sta bas1
FEF3! 60 rts
FEF4!
FEF4!
FEF4! 00 00 .org 0FEF6
FEF6! 20 01FE CRMON JSR BL1
FEF9! 68 PLA
FEFA! 68 PLA
FEFB! D8** BNE MONZ
FEFD!
FEFD! 60 read rts ; no tape input!
FEFE!
FD7D* FEFE
FEFE! 08 get80 php
FEFF! A4 24 ldy ch
FF01! 20 C9FC jsr selbnk
FF04! B1 28 lda #bas1,y
FF06! 4C E3FE jmp vidrts
FF09! ;
FF09! ; fast scroll 1 line without jsr selbnk
FF09!
FF09! 08 scr180 php
FF0A! 78 sei ; DANGER: 400..BFF is switched!
FF0B! 4A lsr A
FF0C! AB tay

```

## MONITOR FILE:ROM3.TEXT

FF0D: 90**	bcc	evenchr	; first time odd or even?
FF0F: 8D 0DC0	oddchr	sta	vidbnk+1 ; static RAM on
FF12: B1 28		lda	#bas1,y ; copy in static RAM
FF14: 91 2A		sta	#bas21,y
FF16: 8D 0CC0		sta	vidbnk ; static RAM off
FF19: CE 7904		dec	chy
FF1C: 30**		bmi	scrlex ; ready?
FF0D* 0F			
FF1E: B1 28	evenchr	lda	#bas1,y ; copy in dynamic RAM
FF20: 91 2A		sta	#bas21,y
FF22: 88		dey	
FF23: CE 7904		dec	chy
FF26: 10E7		bpl	oddchr ; more to scroll?
FF1C* 0A			
FF28: 28	scrlex	plp	
FF29: 18		clc	
FF2A: 60		rts	
FF2B:			
FF2B: 00 00		.org	0FF2D
FF2D: 60	PRERR	rts	
FF2E:			
FEC6* 2EFF			
FF2E: 20 C9FC	scrn802	jsr	selbnk
FF31: B1 26		lda	#gbas1,y
FF33: 8D 0CC0		sta	vidbnk
FF36: AC 7904		ldy	chy
FF39: 60		rts	
FF3A:			
FD5D* 3AFF			
FF3A: A9 87	BELL	LDA	#87
FF3C: 4C EDFD		JMP	COUT

## MONITOR FILE:ROM3.TEXT

		.page	
FF3F:			
FEBA* 3FFF			
FF3F: A5 48	RESTORE	LDA	STATUS
FF41: 48		PHA	
FF42: A5 45		LDA	acc
FF44: A6 46	RESTR1	LDX	Xreg
FF46: A4 47		LDY	Yreg
FF48: 28		PLP	
FF49: 60		RTS	
FF4A:			
FF4A: 85 45	SAVE	STA	acc
FF4C: 86 46	SAV1	STX	Xreg
FF4E: 84 47		STY	Yreg
FF50: 08		PHP	
FF51: 68		PLA	
FF52: 05 48		STA	status
FF54: B8		TSX	
FF55: 86 49		STX	spnt ; save the wrong stack pointer value!
FF57: D8		CLD	
FF58:		.org	0FF58
FF58: 60	iorts	RTS	; used by slot ROM
FF59:			

## MONITOR FILE:ROM3.TEXT

			.page
FF59:		OLDRST	JSR SETNORM
FF59: 20 84FE			JSR INIT
FF5C: 20 2FF8			JSR SETVID
FF5F: 20 93FE			JSR SETKBD
FF62: 20 89FE			
FF65:			
FF65: D8	MON	CLD	
FF66: 20 3AFF		JSR	BELL
FEFB* 6C			
FF69: A9 AA	MON2	LDA #0AA	; **
FF6B: 85 33		STA	PROMPT
FF6D: 20 67FD		JSR	GETLNZ
FF70: 20 ****		JSR	ZMODE
FF73: 20 ****	NXTITM	JSR	GETNUM
FF76: 84 34		STY	YSAV
FF78: A0 11		LDY	#011
FF7A: 88	CHRSRCH	DEY	
FF7B: 30E8		BMI	MON
FF7D: D9 ****		CMP	CHRTBL, Y
FF80: D0F8		BNE	CHRSRCH
FF82: 20 ****		JSR	TOSUB
FF85: A4 34		LDY	YSAV
FF87: 4C 73FF		JMP	NXTITM
FF8A: A2 03	DIG	LDX	#3
FF8C: 0A		ASL	A
FF8D: 0A		ASL	A
FF8E: 0A		ASL	A
FF8F: 0A		ASL	A
FF90: 0A	NXTBIT	ASL	A
FF91: 26 3E		ROL	A2L
FF93: 26 3F		ROL	A2H
FF95: CA		DEX	
FF96: 10F8		BPL	NXTBIT
FF98: A5 31	NXTBAS	LDA	MODE
FF9A: D0**		BNE	NXTBS2
FF9C: B5 3F		LDA	A2H, X
FF9E: 95 3D		STA	A1H, X
FFA0: 95 41		STA	A3H, X
FF9A* 06			
FFA2: E8	NXTBS2	INX	
FFA3: F0F3		BEQ	NXTBAS
FFA5: D0**		BNE	NXTCHR
FF74* A7FF			
FFA7: A2 00	GETNUM	LDX	#0
FFA9: 86 3E		STX	A2L
FFAB: 86 3F		STX	A2H
FFA5* 06			
FFAD: 20 2BFC	NXTCHR	jsr	getupcs
FFB0: 49 B0		EOR	#0B0
FFB2: C9 0A		CMP	#0A
FFB4: 90D4		BCC	DIG
FFB6: 69 88		ADC	#88
FFB8: C9 FA		CMP	#0FA
FFBA: B0CE		BCS	DIG

MONITOR FILE:ROM3.TEXT

FFBC1 60		RTS
FFBD1		
FFBD1 00		.org 0FFBE
FFB3* BEFF		
FFBE1 A9 FE	TOSUB	LDA #0FE ; command page
FFC01 48		PHA
FFC11 B9 ***		LDA SUBTBL, Y
FFC41 48		PHA ; JMP by RTS
FFC51 A5 31		LDA MODE
FF71* C7FF		
FFC71 A0 00	ZMODE	LDY #0
FFC91 84 31		STY MODE
FFCB1 60		RTS

MONITOR FILE:ROM3.TEXT

FFCC1	.page	
FFCC1	.org	0FFCC
FF7E* CFFF		
FFCC1 EA	CHRTBL .byte	0EA ; Q
FFCD1 BB		0BB ; ctrl-B
FFCE1 EE		0EE ; U
FFCF1 98		098 ; ?
FFD01 EF		0EF ; V
FFD11 06		006 ; M
FFD21 04		004 ; K
FFD31 E9		0E9 ; P
FFD41 07		007 ; N
FFD51 02		002 ; I
FFD61 05		005 ; L
FFD71 00		000 ; G
FFD81 93		093 ; :
FFD91 A7		0A7 ; .
FFDA1 95		095 ; <
FFDB1 C6		0C6 ; ctrl-M
FFDC1 99		099 ; blank
FFDD1		
FFC2* DFFF		
FFDD1 58	SUBTBL .byte	058 ; Basic warm jmp 03F2 is moved
FFDE1 5B		05B ; Basic cold jmp 0E000 is moved
FFDF1 C9		0C9 ; user jmp 03F8
FFE01 BE		0BE ; register display
FFE11 35		035 ; verify
FFE21 2B		02B ; move
FFE31 8C		08C ; input vector
FFE41 96		096 ; output vector
FFE51 83		083 ; normal
FFE61 7F		07F ; inverse
FFE71 5F		05F ; list is moved!
FFE81 B5		0B5 ; go
FFE91 18		018 ; :
FFEA1 18		018 ; .
FFEB1 20		020 ; <
FFEC1 F5		0F5 ; <cr>
FFED1 04		004 ; <space>
FFEE1		
FFEE1 84 2D	clrsc3 sty	v2
FFF01 A0 4F	1dy	#04f ; 80-col -1
FFF21 AD F904	1da	switch
FFF51 D8**	bne	clr80
FFF71 A0 27	1dy	#027 ; 40-col -1
FFF5* 02		
FFF91 60	clr80 rts	
FFFF1		
FFFF1	.org	0FFFFA
FFFF1 FB03	.word	NMI
FFFC1 62FA	.word	RESET
FFFE1 40FA	.word	IRQ
00001		
00001		.end

MONITOR FILE:ROM3.TEXT

SYMBOLTABLE DUMP

AB - Absolute	LB - Label	UD - Undefined	MC - Macro
RF - Ref	DF - Def	PR - Proc	FC - Func
PB - Public	PV - Private	CS - Consts	

A1H	AB 003D	A1L	AB 003C	A1PC	LB FE75
A2H	AB 003F	A2L	AB 003E	A3H	AB 0041
A3L	AB 0040	A4H	AB 0043	A4L	AB 0042
ACC	AB 0045	ADDINP	LB FD84	ADVANCE	LB FBF4
AMPERV	AB 03F5	BAS2H	AB 002B	BAS2L	AB 002A
BASCALC	LB FBC1	BASCONT	LB FE59	BASH	AB 0029
BASIC	AB E000	BASIC2	AB E003	BASL	AB 0028
BCKSPC	LB FD71	BELL	LB FF3A	BELL1	LB FBD9
BIT7	AB 0080	BL1	LB FE01	BLANK	LB FE05
BREAK	LB FA4C	BRKV	AB 03F0	BS	LB FC10
CANCEL	LB FD62	CH	AB 0024	CHAR1	LB F9B4
CHAR2	LB F9BA	CHR BAS	AB C006	CHR FONT	AB 0001
CHRCEN0	AB C002	CHRCEN1	AB C004	CHR INV	AB C000
CHRSRCH	LB FF7A	CHRTBL	LB FFCC	CHY	AB 0479
CLEOLB0	LB FCDB	CLEOLZ	LB FC9E	CLEOP1	LB FC46
CLR80	LB FFF9	CLREOL	LB FC9C	CLREOP	LB FC42
CLRROM	AB CFFF	CLRSC2	LB FB38	CLRSC3	LB FFEE
CLRSCR	LB F832	CLRTOP	LB FB36	COLOR	AB 0030
COUT	LB FDE0	COUT1	LB FDF0	CR	LB FC62
CRMON	LB FEF6	CROUT	LB FD8E	CSWH	AB 0037
CSWL	AB 0036	CURS80	LB FED7	CV	AB 0025
DATAOUT	LB FDB6	DIG	LB FF8A	DISKID	LB FB01
ERR	LB F8A5	EVENCHR	LB FF1E	FIXSEV	LB FA98
FMT1	LB F962	FMT2	LB F9A6	FORMAT	AB 002E
GBASCALC	LB F847	GBASH	AB 0027	GBASL	AB 0026
GET80	LB FFEF	GETFMT	LB F8A9	GETLN	LB FD6A
GETLNZ	LB FD67	GETNUM	LB FFA7	GETUPCS	LB FC2B
GO	LB FEB6	H2	AB 002C	HLINE	LB F819
HOME	LB FC58	I EVEN	LB F89B	IN	AB 0200
INIT	LB FB2F	IMPORT	LB FE8B	INPRT	LB FE8D
INSDS1	LB F882	INSDS2	LB F88C	INSTDSP	LB F8D0
INVFLG	AB 0032	IOARD	AB C000	IOPAGE	AB 00C0
IOPRT	LB FE9B	IOPRT1	LB FEA7	IOPRT2	LB FEA9
IORTS	LB FF58	IRQ	LB FA40	IRQLOC	AB 03FE
JLOCAL	LB FBB6	KBD	AB C000	KBDEXTN	AB C008
KBDSTRB	AB C010	KEYIN	LB FD1B	KSWH	AB 0039
KSWL	AB 0038	LASTIN	AB 002F	LENGTH	AB 002F
LF	LB FC66	LINE1	AB 0400	LIST	LB FE60
LMNEM	AB 002C	LOC0	AB 0000	LOC1	AB 0001
LOCAL	LB FBA0	LOCCHR	LB FB09	LOCJMP	LB FB99

## MONITOR FILE:ROM3.TEXT

LOGO	LB FB60:	LOG01	LB FD0F:	LORES	AB C056:
LOWSCR	AB C054:	LT	LB FE21:	MASK	AB 002E:
MIXCLR	AB C052:	MNEML	LB F9C0:	MNEMR	LB FA00:
MNNDX1	LB F8BE:	MNNDX2	LB F8C2:	MNNDX3	LB F8C9:
MOD8CHK	LB FDAD:	MODE	AB 0031:	MON	LB FF65:
MONZ	LB FF69:	MOVE	LB FE2C:	MSLOT	AB 07F8:
NEWLN	LB FDFA:	NEWMON	LB FA6F:	NMI	AB 03FB:
NOCTRL	LB FBED:	NOFIX	LB FAA0:	NOTCR	LB FD3D:
NOTCR1	LB FD5F:	NXTA1	LB FCBA:	NXTA4	LB FCB4:
NXTBAS	LB FF98:	NXTBIT	LB FF90:	NXTBS2	LB FFA2:
NXTCHAR	LB FD75:	NXTCHR	LB FFAD:	NXTCOL	LB F85F:
NXTITM	LB FF73:	ODDCHR	LB FF0F:	OLDBRK	LB FA59:
OLDRST	LB FF59:	OUTPORT	LB FE95:	OUTPRT	LB FE97:
PADDL0	AB C064:	PCADJ	LB F953:	PCADJ2	LB F954:
PCADJ3	LB F956:	PCADJ4	LB F95C:	PCH	AB 003B:
PCL	AB 003A:	PIP	LB FB0F:	PLOT	LB F800:
PL0T1	LB F80E:	PL0T80	LB FCED:	PRA1	LB FD92:
PRADR1	LB F910:	PRADR2	LB F914:	PRADR4	LB F92A:
PRADR5	LB F930:	PRBL2	LB F94A:	PRBL3	LB F94C:
PRBLNK	LB F948:	PRBYTE	LB FDDA:	PREAD	LB FB1E:
PRERR	LB FF20:	PRHEX	LB FDE3:	PRHEXZ	LB FDE5:
PRNTAX	LB F941:	PRNTBL	LB F8DB:	PRNTOP	LB F8D4:
PRNTX	LB F944:	PRNTYX	LB F940:	PROMPT	AB 0033:
PRYX2	LB FD96:	PTRIG	AB C070:	PWRCON	LB FAFD:
PWREDUP	AB 03F4:	PWRUP	LB FAA3:	RDCHAR	LB FD35:
RDCHAR1	LB FBB9:	RKEY	LB FD0C:	RDKEY2	LB FD15:
READ	LB FEFD:	REGDSP	LB FAD7:	REGZ	LB FEBF:
RELADR	LB F938:	RESET	LB FA62:	RESTORE	LB FF3F:
RESTR1	LB FF44:	RGDSP1	LB FADA:	RMNEM	AB 002D:
RNDH	AB 004F:	RNDL	AB 004E:	RTBL	LB FB19:
RTS1	LB F831:	RTS2	LB F961:	RTS4	LB FC2A:
SAV1	LB FF4C:	SAVE	LB FF4A:	SCRL80	LB FF09:
SCRLEX	LB FF28:	SCRN	LB F871:	SCRN2	LB F879:
SCRN80	LB FEC2:	SCRN802	LB FF2E:	SCROLL	LB FC70:
SELBNK	LB FCC9:	SELBNK2	LB FCCC:	SETCOL	LB F864:
SETGR	LB FB40:	SETIFLG	LB FE86:	SETINV	LB FE88:
SETKBD	LB FE89:	SETMD2	LB FE1E:	SETMODE	LB FE19:
SETNORM	LB FE84:	SETPG3	LB FAAA:	SETPLP	LB FAAC:
SETPWRC	LB FB6F:	SETTXT	LB FB39:	SETVID	LB FE93:
SETWND	LB FB4B:	SLOOP	LB FABB:	SOFTEV	AB 03F2:
SPKR	AB C030:	SPNT	AB 0049:	STATUS	AB 0048:
STOR	LB FE0C:	STOR80	LB FECE:	STORADV	LB FBF0:
STORINV	LB FBEE:	STRTS	LB FEE1:	SUBTBL	LB FFDD:
SW1	LB FB10:	SW2	LB FB13:	SW3	LB FCFA:
SW4	LB FD03:	SW5	LB FC36:	SW6	LB FC37:
SW7	LB FDC6:	SW748	LB FDCF:	SWITCH	AB 04F9:
TABV	LB FB5B:	TAPEIN	AB C060:	TAPEOUT	AB C028:
TITLE	LB FB90:	TOSUB	LB FFBF:	TTLOUT0	AB C058:
TTLOUT1	AB C05A:	TTLOUT2	AB C05C:	TTLOUT3	AB C05E:
TXTCLR	AB C050:	UP	LB FC1A:	UPPER	LB FC2F:

MONITOR FILE:ROM3.TEXT

USR	LB FECA:	USRADR	AB 03F8:	V2	AB 002D:
VERIFY	LB FE36:	VID40	AB C00A:	VID80	AB C00B:
VIDBNK	AB C00C:	VIDOUT	LB FBFD:	VIDPLP	LB FEE6:
VIDRTS	LB FEE3:	VIDWAIT	LB FB78:	VLINE	LB F828:
VLINEZ	LB F826:	VTAB	LB FC22:	VTAB80	LB FEEB:
VTABZ	LB FC24:	WAIT	LB FCA8:	WDTHOK	LB FDD7:
WIDTH	AB 0050:	WNDBTM	AB 0023:	WNDLFT	AB 0020:
WNDTOP	AB 0022:	WNDWDTH	AB 0021:	WRITE	LB FEC0:
XAM	LB FDB3:	XAM8	LB FDA3:	XBASIC	LB FE5C:
XREG	AB 0046:	YREG	AB 0047:	YSAV	AB 0034:
YSAV1	AB 0035:	ZMODE	LB FFC7:		:

MONITOR FILE:ROM3.TEXT

Current minimum space is 6838 words

F810\* C9FC  
F813\* EDFC  
F887\* 96FD  
F946\* DAFD  
F942\* DAFD  
F92E\* DAFD  
F8D7\* DAFD  
F94D\* EDFD  
F924\* EDFD  
F91C\* EDFD  
F904\* EDFD  
FA64\* 84FE  
FA6D\* 89FE  
FA6A\* 93FE  
F874\* C2FE  
FBF1\* CEFE  
FB17\* 09FF  
FA71\* 3AFF  
FA4E\* 4cff  
FA68\* 65FF  
F839\* EEFF

Assembly complete: 1335 lines  
8 Errors flagged on this Assembly

## PRINTER FILE:PRINTER.21

```

0000:           .absolute
0000:           .proc printer
Current memory available: 8644
0000:
0000: 0021      version .equ 21      ; version 2.1
0000:
0000: C100      rom    .equ 0C100
0000: 00C1      rompage .equ 0C1
0000:           .org rom
C100:
C100: C090      devsel .equ 0C090
C100: C1C1      pready .equ 0C1C1
C100:
C100: C098      preg   .equ devsel
C100: C098      acia   .equ devsel+8
C100:
C100: C098      inreg  .equ acia+0 ; 7 6 5 4 3 2 1 0
C100: C098      outreg .equ acia+0 ;
C100:
C100: C099      stsreg .equ acia+1 ; IRQ DSR DCD tran rec ovr- frm- par-
C100:           ; occur inact inact empty full error
C100:
C100: C09A      cmdreg .equ acia+2 ; parity par rec transmit rec- DTR
C100:           ; mode-ctrl enabl echo IRQ,RTS,brk IRQ activ
C100:
C100: C09B      ctrlreg .equ acia+3 ; 2 stop word- clock baud rate
C100:           ; bits length intr
C100:
C100: 0478      Accu   .equ 478   ; save char
C100: 04F8      chanel .equ 4F8   ; par/ser out switch : if channel<80 then par else ser
C100:
C100: 0479      vid0   .equ 479   ; used in the 80-col screen driver
C100: 04F9      vid1   .equ 4F9   ; reserved
C100: 0579      vid2   .equ 579   ; reserved
C100: 05F9      modechk .equ 5F9   ; warmstart byte
C100: 8679      mode   .equ 679   ; CR->CR/LF video echo
C100:
C100: 86F9      ctrl   .equ 6F9   ; value for ACIA ctrl-reg
C100: 0779      cmd    .equ 779   ; value for ACIA cmd-reg
C100:
C100: 07F9      hCount .equ 7F9
C100:
C100: 0024      ch     .equ 24
C100: 0036      csrw  .equ 36
C100: 0038      ksw   .equ 38
C100: FDF0      coutl .equ 0FDF0
C100:
C100: 002C      bit_a  .equ 2C
C100: 20 ****   jsr    init
C100: 90**      bcc   pwrite2
C105:
C105:           .org rom+5
C105: 48        pha    ; tested by Pascal
C106: 21        .byte  version
C107: 48        pha    ; tested by Pascal
C108:

```

## PRINTER FILE:PRINTER.21

```

C108I .org rom+8
C108I 48 v24 pha ; first entry for IN#9 or PR#9
C109I A5 39 lda ksw+1
C10BI C9 C1 cmp #rompage
C10DI D8** bne swrite ; no
C10FI 68 pla
C110I A9 14 sread lda #14 ; yes, first entry
C112I 85 38 sta ksw ; zap entry to sread2
C114I
C114I .org rom+14
C114I 20 **** sread2 jsr init
C117I A9 08 lda #8
C119I 2C 99C0 $8 bit stsreg
C11CI F0FB beq $8
C11EI AD 98C0 lda inreg
C121I 49 88 eor #80
C123I 60 rts
C124I
C124I
C10D* 15
C124I A9 29 swrite lda #29 ; first PR#9 entry
C126I 85 36 sta csw ; zap entry vector
C128I 68 pla
C129I
C129I .org rom+29
C129I 20 **** swrite2 jsr init ; setup the 6551
C12CI 38 sec
C12DI
C103* 28
C12DI 6E F804 pwrite2 ror chanel
C130I output
C130I EE F907 $1 inc hCount
C133I A5 24 lda ch
C135I CD F907 cmp hCount
C138I 98** bcc notab
C13AI A9 A0 lda #8A0
C13CI 20 **** jsr out1
C13FI 4C 30C1 jmp $1
C138* 08
C142I 20 **** notab jsr out
C145I C9 0D cmp #8D
C147I D8** bne nocr
C149I 20 **** jsr cCount
C14CI 2C 7906 bit mode
C14FI 10** bpl nocr
C151I A9 8A lda #8A
C159I 20 **** jsr out1
C14F* 05
C147* 0D
C156I 2C 7906 nocr bit mode
C159I AD 7804 lda accu
C15CI 50** bvc ret
C15EI 4C F0FD jmp cout1
C161I
C161I 00 brk

```

## PRINTER FILE:PRINTER.21

```

C1621
C12A* 62C1
C115* 62C1
C101* 62C1
C1621 80 7804      init  sta   Accu    ; low(addr)=Fx
C1651 AD F905      lda    modechk
C1681 49 A5        eor    #0A5    ; printer/v24 warmstart?
C16A1 CD 7906      cmp    mode
C16D1 F0**        beq    warm   ; yes
C16F1
C16F1 A9 9E        lda    #0E    ; no, set default values: 8 data+2 stop bits,
C1711 80 F906      sta    ctrl   ;                                9600 baud
C1741
C1741 A9 0B        lda    #0B    ; no parity, DTR=low,   RTS=low
C1761 80 7907      sta    cmd
C1791
C1791 A9 C0        lda    #0C0   ; mode bit 7: CR->CR/LF translation on
C17B1 80 7906      sta    mode   ;       bit 6: output echo to video
C17E1
C17E1 49 A5        eor    #0A5
C1801 80 F905      sta    modechk ; set warmstart flag
C1831
C14A* 83C1
C1831 A9 00        cCount lda   #0
C1851 80 F907      sta    hCount ; init Tabulator count
C1881
C16D* 19
C1881 AD 7907      warm   lda   cmd
C1881 CD 9AC0      cmp    cmdreg ; is the 6551 cmd register ok ?
C18E1 F0**        beq    $1
C1901 80 9AC0      sta    cmdreg ; no
C18E* 03
C1931 AD F906      $1    lda   ctrl
C1961 CD 9BC0      cmp    ctrlreg ; is the 6551 ctrl register ok ?
C1991 F0**        beq    $2
C19B1 80 9BC0      sta    ctrlreg ; no
C199* 03
C19E1 18          $2    cic
C15C* 41
C19F1 60          ret    rts
C1A01
C143* ABC1
C1A01 AD 7804      out   lda   Accu
C154* A3C1
C13D* A3C1
C1A31 49 80        outi  eor   #00
C1A51 20 F804      bit    channel
C1A81 10**        bpl    pout
C1AA1
C1AA1 48          sout  pha   ; save char
C1AB1 A9 10        lda    #10
C1AD1 20 99C0      $0    bit    stsreg ; ready for next char ?
C1B01 F0FB        beq    $0    ; no, wait
C1B21 60          pla   ; yes
C1B31 80 98C0      sta    outreg ; send it

```

PRINTER FILE:PRINTER.ZI

```
C1B6I 60          rts
C1B7I
C1A8* 0D
C1B7I 2C C1C1      pout   bit    pready
C1BAI 30FB      bmi    pout
C1BCI 8D 90C0      sta    devsel
C1BFI 60          rts
C1C0I
C1C0I           .org   rom+0C0
C1C0I           .end
```

PRINTER FILE:PRINTER.21 SYMBOLTABLE DUMP

AB - Absolute      LB - Label      UD - Undefined      MC - Macro  
RF - Ref           DF - Def          PR - Proc           FC - Func  
PB - Public        PV - Private     CS - Consts

ACCU	AB 04781	ACIA	AB C0981	BITA	AB 002C1	BYTE5	LB C1051
BYTE7	LB C1071	CCOUNT	LB C1831	CH	AB 00241	CHANNEL	AB 04F81
CMD	AB 07791	CMDREG	AB C09A1	COUT1	AB FDF01	CSW	AB 00361
CTRL	AB 06F91	CTRLREG	AB C09B1	DEVSEL	AB C0901	HCOUNT	AB 07F91
INIT	LB C1621	INREG	AB C0981	KSW	AB 00381	MODE	AB 06791
MODECHK	AB 05F91	NOCR	LB C1561	NOTAB	LB C1421	OUT	LB C1A81
OUT1	LB C1A31	OUTPUT	LB C1301	OUTREG	AB C0981	POUT	LB C1B71
PREADY	AB C1C11	PREG	AB C0901	PRINTER	PR ---1	PWRITE2	LB C12D1
RET	LB C19F1	ROM	AB C1001	ROMPAGE	AB 00C11	SOUT	LB C1AA1
SREAD	LB C1101	SREAD2	LB C1141	STSREG	AB C0991	SWRITE	LB C1241
SWRITE2	LB C1291	V24	LB C1081	VERSION	AB 00211	VID0	AB 04791
VID1	AB 04F91	VID2	AB 05791	WARM	LB C1881		

Current minimum space is 8231 words

Assembly complete: 158 lines  
0 Errors flagged on this Assembly

## ANHANG O

### Stichwortverzeichnis

#### A

Acknowledge	8, 98
Adress-	
bus	15, 18, 100
raum, aufteilung	58
Adressen-	
der Tastatur	37
Zeichengenerator	35
Ein-/Ausgabe	97
Apple	
CP/M	75, 88, 100
Pascal	73
Applesoft	77, 87
ASCII-Zeichen	35, 94
Anschluß-	
Betriebsspannung	14
Drucker	7
Fernsehgerät	7, 90
Handregler	13, 68
Kassettenrekorder	11
Tastatur	8
Autostart-ROM	
= Monitor ROM	11, 39, 85

#### B

Bank	59
Basicversionen	79
Baud Rate	66, 88
Betriebssystem	6, 7, 23, 39
Bildmodus-Schalter	32
Bildschirm	6, 7

#### C

Controller	11, 39
CONTROL-Taste	36
CP/M	23, 27, 39, 88
CRTL - CONTROL	36

#### D

Daisy Chain	15, 100
Interrupt	15
DMA-Ausgang	15
Datenbus	18
Dateneingänge	63
Datensichtgerät	6
Diskette	
ZAP:	6, 73
Disketten	19
Diskettenlaufwerk	6, 11, 18
Einbau	19
Pflege	19
DOS3.3	23, 28, 39, 88
Druckzeichen	8

#### E

Ein-/Ausgabe	62
Bausteine	11
Adressen	97
Ein-/Ausgang	
Handregler	68
Erweiterungs ROM	68

#### F

Farbausgabe	
Einstellung	11
Fernsehgerät	6
Festspeicher = ROM	
Flash	35, 87

#### G

Gerätemasse	8
GND	8, 17
Graphik	33, 97
HI-RES	34
LO-RES	33
MI-RES	34
mixed	33

## H

Handregler	
adressen	68, 98
anschluß	13, 68
-signal	13
-belegung	13
Hauptplatine	8, 11, 12
Hexadezimalziffern	39, 93
HI -RES-Graphik	34
Hochauflösende Graphik	34

## I

Impulsausgang	
Interface	
Drucker	64
serielles RS 232c	64
Kassettenrekorder	68
Integer Basic	77
Interrupt	15
Tastatur	37
Invers	35, 87
I/O RAM Zwischenspeicher	
70	
I/O SELECT	15

## K

Kabelanschluß	6
Kaltstart	85
Kassettenrekorder	11, 39
Adressen	68, 98
Anschluß	11
Arbeiten mit dem	91
Schreiben/Lesen	50
Kommandoregister	66
Kompatibilität mit Apple	
73	
Kontrollregister	65

## L

Language Card	99
Lautsprecher	68
Lese-/Schreibsignal	99
Linkspfeiltaste	94
Logischer Schaltplan	31
LO-RES-Graphik	33

## M

Maschinenprogramme	44, 50
Mikroprozessoren	
6502	11, 15, 46, 50, 86
	99
Z-80	11, 99
MI -RES-Graphik	33
mixed Graphik	33
Monitor ROM	11, 39
Kommandos	46, 49
Unterprogramme	53
Spezialadressen	56
Move	86

## N

Netzteil	11, 14
Pinbelegung	14
Page	
Pascal	23, 24, 39
Peripheriekarten	69
Pinbelegung	
Steckleiste, Rücks.	8
Slots	15
Printer Connect	8
Programmschalter	
= Softwareschalter	33

## R

RAM	11, 60
Rechtspfeiltaste	94
Register	46
Kommando	66, 88
Kontroll	65, 88
Status	67
RESET	17, 36, 39
RETURN	39
ROM	11, 39, 60
RGB-Monitor	6, 11
Rücksetztaste	
= Linkspfeiltaste	

**S**

Schaltplan	31
logischer	
Schaltungsbrücke	13
Schnittstellen	
parallel u. seriell	64
Schreib-/Lesespeicher	
= ROM	11
SHIFT-Taste	36
Signalmasse	8
Softwareschalter	
Bankumschaltung	59
Graphik	33
ROM und RAM Umsch.	60
Tastatur	37
Text	33
Zeichengenerator	35
Speicherorganisation	58
Speicherstelle	40
Änderung	41, 42, 49
Überprüfen	40, 49
Übertragen	43, 49, 86
Vergleich	44, 49, 86
Spieleanschluß	13
s. Handregler	13
Steuerung	13
Statik-RAM	61
Statusregister	67
Steckdosen	6, 8
Steckleisten	7, 8
Strobe	8, 63
Stromversorgung	14, 17, 18

**T**

Takt-	18
7MHz	
2MHz	18
Steuerung	99
Generierung	99
Tastatur	6, 8, 36, 96
Anschluß	7, 9
Tastenbelegung	94
Text	
Darstellung	32
Bildschirm	32'
Text-Fenster	32

**U**

UCSD p-System IV.0	23, 24
	39, 88
Umschaltung	
Bank	59
ROM und RAM	59
USER	48, 86
UT 108, Volume	6, 81

**V**

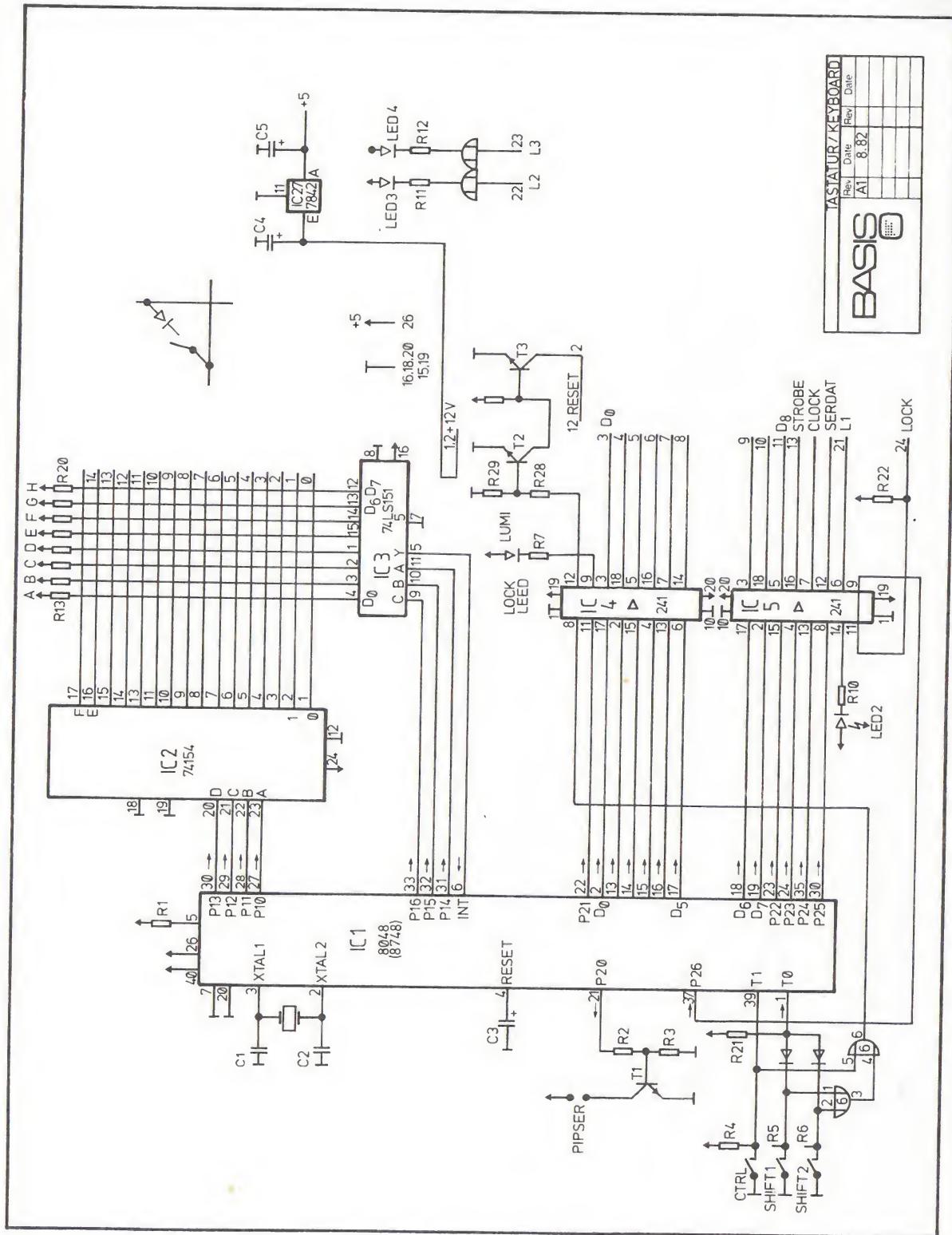
Vergleichen von Bereichen	
Video-Anschluß	6, 11
Vollgraphik	33
Volume UT 108	6
V24 Parameter	88

**W**

Warmstart	85
-----------	----

**Z**

ZAP: -Diskette	6, 73
Zeichen/ Zeile	
40	6, 11, 32, 50, 85
80	32, 33, 61, 85
Zeichengenerator	35
Zeichensatz, ändern	81
Zentraleinheit	6
Zusatztasten	37
Z-80	11
-Teil	99





C

O

C

O



MICROCOMPUTER  
GMBH

D-4400 Münster  
Postfach 1603  
Telex 892 643 basis d  
BTX 244