K D T R e v . 6

T C B / Z 8 0

Test-Debugger Handbuch

Eching, den 28.04.1983

Beschriebene Version : 6.20

KONTRON Mikrocomputer GmbH

INHALTSVERZEICHNIS: SEITE

1. Allgemeines:

1.1 Einführung

Der nachfolgend beschriebene Testdebugger ist aus der Notwendigkeit enstanden, die produzierten KDT-Platinen nicht nur einem GO-NOGO Test zu unterziehen, sondern um einerseits Testprozeduren ablaufen zu lassen, andererseits um einfache Maschinenprogramme direkt einzugeben und zu testen. Somit soll die Fehler-Diagnosezeit auf ein Minimum reduziert werden.

Der Testdebugger besteht aus 2 Proms. Prom 1 enthält eine Grundsoftware mit den Kommandos, Initialisierung, Verwalter, Kommandointerpreter, Bildschirmausgabe usw. und ist für sich alleine voll funktionsfähig. Prom 2 enhält die Disk-Software (siehe auch 6.3). Die Software kann das Vorhanden sein von Proms erkennen und meldet Zugriffe auf nicht vorhandene Software mit

* NOT IMPLEMENTED *

Der Testdebugger kann nach einem RESET mit CTRL-K aufgerufen werden. Am Bildschirm erscheint der Text :

TESTDEBUGGER VERSION : (Versionsnummer) (Datum)
TD>

Die derzeit (25.04.83) aktuelle Version ist 6.20.

Der Testdebugger ist in vier verschiedenen Ausführungen verfügbar:

- Video 60 Hz , SIO-Kanal B aktiv (für PSI 9XXX-Systeme)
- Video 50 Hz , SIO-Kanal B aktiv
- Video 60 Hz , SIO-Kanal A aktiv
- Video 50 Hz , SIO-Kanal A aktiv (für KLA)

SIO-Kanal A aktiv bedeutet, daß Ein/Ausgabe über die serielle Schnittstelle SIO A möglich ist und die Kommandos ON und OF auf diese Schnittstelle wirken (siehe 3.20 u. 3.21).

1.2 Sonderfunktion

Die Funktion von CTRL-Q, CTRL-R, CTRL-S, CTRL-T und CTRL-W siehe Tabelle 6.5.

1.2.1 ESC-Taste

Die ESC-Taste bewirkt einen sofortigen Abbruch des Programms und einen Warmstart. Eine Ausnahme bilden die Floppy-Disk Zugriffe, bei denen die ESC-Funktion unwirksam ist.

Hinweis: Da die ESC-Funktion einen sofortigen Abbruch des Programms zur Folge hat, können immer dann Probleme auftreten, wenn eine Interrupt-Service-Routine unterbrochen wird.

1.2.2 Abbruch durch beliebige Taste

Bei den Kommandos Nr. 2,3,18 (siehe Tabelle 6.1) kann die Ausführung durch Drücken einer beliebigen Taste (außer "ESC", siehe 1.2.1 und außer CTRL-R, -S, -T, -W siehe 6.5) abgebrochen werden. Es wird jedoch erst an einer für die Ausführung des jeweiligen Kommandos sinnvollen Stelle abgebrochen. Eine Unterbrechung ist mit "W" möglich. Danach führt die Eingabe einer beliebigen Taste (außer "ESC", siehe 1.2.1 und außer CTRL-R, -S, -T, -W siehe 6.5) zum Abbruch, eine Eingabe von "W" läßt das Kommando weiterarbeiten.

1.2.3 Erkennen von externer Hardware über serielle Schnittstelle

Nach einem Reset oder Kaltstart (siehe 4.1) wird das DTR-Signal der seriellen Schnittstelle (A oder B) zuerst auf Low (aktiv) und dann auf High (inaktiv) programmiert. Wenn auf diesen Low-High Übergang ein angeschlossenes Gerät mit "ACK" (ASCII 06H) antwortet, so wird im Testdebugger auf eine Laderoutine gesprungen und auf Eingaben über die serielle Schnittstelle (9600 BAUD) gewartet. Damit besteht die Möglichkeit, über die serielle Schnittstelle ein Testsystem für beliebige Funktionen anzuschließen.

1.3 Zeichenerklärung

<CR> bedeutet Eingabe von CARRIAGE RETURN (ASCII ODH). Dies geschieht mit Hilfe der "RETURN"-Taste, bei manchen Tastaturen auch als "NEW LINE" bezeichnet.

H Eine Hexadezimalzahl (auch Hexzahl genannt) wird durch ein "H" hinter der Zahl gekennzeichnet.

CTRL- Die Angabe "CTRL-" bedeutet, daß eine Taste zusammen mit der CONTROL-Taste zu drücken ist.

2. Kommandoeingaben

2.1 Aufbau eines Kommandos

Formal besteht jede Kommandoeingabe (jedes Kommandofeld) aus einem Identifikationsfeld (ID-FELD), einem Parameterfeld (P-Feld), sowie den dazwischenliegenden Trennzeichen (TZ).

Es ergibt sich das folgende Format für ein Kommando:

```
        <---ID-Feld--->
KOMMANDOAUFRUF TZ P1 TZ P2 TZ ...Pi TZ ...Pn <CR>
```

Das ID-Feld muß mit einem Großbuchstaben (A bis Z) beginnen und reicht formal bis zum ersten Trennzeichen (TZ1). Da der Kommandointerpreter in der Regel nur ein, höchstens aber zwei Zeichen zur Indentifikation eines Kommandos benötigt (siehe Punkt 3), bleiben eventuell vorhandene weitere Zeichen im ID-Feld bedeutungslos.

Beispiel:

Die Eingaben:
```
D 9000 100
DISPLAY 9000 100
DXYZ 9000 100
```

bewirken alle den Ausdruck von 100H Bytes ab Adresse 9000H.

Als Trennzeichen (TZ) dürfen nur Leerzeichen (ASCII-CODE 20H) verwendet werden, wobei deren Anzahl zwischen den Parametern keine Rolle spielt. Das Parameterfeld ist für verschiedene Kommandos optional. Es enthält mit Ausnahme des Register- und des Setkommandos (ASCII-Mode) nur hexadezimale Zahlenwerte (Ziffern 0...9 und die Zeichen A...F). Die Eingabe des Parameterfeldes ist formatfrei, d.h. führende Nullen brauchen nicht mit eingegeben zu werden.

Beispiel:

Die Eingaben
```
   9
  09
 009
0009
```

sind gleichbedeutend und werden als vierstellige Hexadezimalzahl 0009H angenommen.

Bei mehr als vier Zeichen werden lediglich die letzten vier berücksichtigt. Entsprechendes gilt auch dann, wenn das Monitorprogramm nur 2 hexadezimale Zeichen als Eingabe erwartet. Es können pro (logischer) Zeile mehrere Kommandos hintereinander eingegeben werden. Der Zeilenpuffer des Debuggers ist 255 Bytes groß, das entspricht mehr als drei Zeilen des Sichtschirmes.

Zur Trennung zweier Kommandos dient das Semikolon (ASCII-Code 3BH). Der ASCII-Code ODH (<CR>, Taste RETURN) schließt die Kommandozeile ab.

Es ergibt sich das folgende Format für eine Kommandozeile:

```
          <--------logische Zeile (255 Bytes max)-------->

    TD> KOMMANDO 1;KOMMANDO 2;...;KOMMANDO i;...;KOMMANDO n<CR>
```

2.2 Fehlermeldungen

Bei Kommandos, deren Format nicht richtig angegeben ist, erscheint:

```
* FORMAT ? *
```

Bei Eingaben, welche nicht als Kommandos erkannt werden:

```
* NOT IMPLEMENTED *
```

3. Kommandos

3.1 A = AGAIN

FORMAT: A <CR>

Beispiel: A <CR>

Das zuletzt abgearbeitete Kommando mit Ausnahme der Kommandos Nr. 1, 13, 14, 29, 30 (siehe Tabelle 6.1) wird wiederholt.

3.2 CP = COMPARE MEMORYBLOCKS

FORMAT: CP (Blockadresse1) (Blockadresse2) (Blocklänge) <CR>

Beispiel: CP 3000 C000 100 <CR>

Zwei Memoryblöcke können miteinander verglichen werden. Dabei wird der Inhalt von Adresse 3000H mit dem Inhalt von Adresse C000H verglichen, der Inhalt von Adresse 3001H mit dem Inhalt von Adresse C001H usw.
Die beiden letzten zu vergleichenden Speicherstellen sind in diesem Beispiel 30FFH und C0FFH (= 100H Länge).

Der gesamte Vergleich beider Blöcke wird 255 mal ausgeführt.

Bei der ersten Nichtübereinstimmung wird die Ausführung abgebrochen und eine Fehlermeldung mit folgenden Angaben ausgegeben:

- Soll- und Istwerte der nicht übereinstimmenden Speicherplätze
- Nummer des Durchlaufs, bei welchem der Fehler aufgetreten ist.

Dabei ist zu beachten, daß bei einem Fehler eine Abprüfung auf "Stop by Error" (siehe 3.7) durchgeführt wird.

Mit der Eingabe von "W" kann der Vergleich unterbrochen und ebenso wieder weitergeführt werden, mit einer beliebigen Taste wird die Ausgabe unterbrochen (siehe 1.2.2).

3.3 D = DISPLAY MEMORY

3.3.1 Display Mode

FORMAT: D (Startadresse) (Länge) <CR>
oder: D (Startadresse)-(Endadresse) <CR>

Beispiel: D 100 10 <CR>
oder: D 100-10F <CR>

Mit Display Memory werden die Adresseninhalte zeilenweise (10H je Zeile) mit ASCII-Äquivalent ausgegeben. Entspricht eine Hexzahl einem ASCII-Zeichen, wird dieses zwischen * * an der entsprechenden Stelle ausgegeben. Ein Text läßt sich somit leicht als solcher erkennen.

Mit der Eingabe von "W" kann die Ausgabe unterbrochen und ebenso wieder weitergeführt werden, mit einer beliebigen Taste wird die Ausgabe unterbrochen (siehe 1.2.2).

3.3.2 Display and Alter mode

FORMAT: D (Adresse) <CR>

Beispiel: D 5300 <CR>

Die Angabe der Adresse ist optional (Voreinstellung : Adresse=0). In diesem Modus arbeitet das D-Kommando interaktiv, d.h. nach dem Ausdruck des Speicherinhalts kann der Benutzer wahlweise den Inhalt des angezeigten Speicherplatzes verändern und/oder auf den nächsten/vorhergehenden Speicherplatz weiterschalten. Der Abbruch einer derartigen "Display and Alter Sequenz" erfolgt mit dem Zeichen "Q" (Quittierung).

Ausgabeformat: (Adresse) (Daten)

Fünf Funktionen können nun durch die Eingabe folgender Zeichen bzw. Zeichenfolgen veranlaßt werden:

	<CR>	Weiterschalten und Ausdruck des Inhalts der Speicherstelle Adresse +1
^	<CR>	Weiterschalten und Ausdruck des Inhalts der Speicherstelle -1
xx	<CR>	Ersetzen des Inhalts von Adresse durch den Wert xx mit anschließendem Weiterschalten auf Adresse +1
xx Q	<CR>	wie oben mit anschließendem Abbruch der Kommandos
Q	<CR>	Abbruch des Kommandos.

3.4 DO = DO-KOMMANDO

FORMAT: DO (Startadresse) <CR>

Beispiel: DO 3000 <CR>

Mit diesem Kommando wird die auf der Startadresse 3000H stehende Befehlsfolge abgearbeitet. Dabei können mehrere Befehle, mit Strichpunkt (;) getrennt, aneinander gereiht werden.
Rekursiv - Aufrufe sind gestattet.

Beispiel: S 3000 /CP 5000 6000 100;D 2000-2030;DO 3000 <CR>

Hierbei stellt "DO 3000" den Rekursivaufruf dar, d.h. den Aufruf auf sich selbst (Schleifenbildung).

3.5 DP = DISPLAY PORT

FORMAT: DP (Portadresse) <CR>

Beispiel: DP 2 <CR>

Der Port mit der Adresse 02H wird ausgelesen und angezeigt.

Hinweis: Der Statusport mit der Adresse 1CH darf nur beschrieben und nicht gelesen werden. Ein Lesevorgang auf diesem Port hat den Absturz der Hardware zur Folge.

3.6 EG = GO BY ERROR

FORMAT: EG <CR>

Beispiel: EG <CR>

Durch den Befehl "EG" (GO BY ERROR) wird die Funktion "ES" (STOP BY ERROR, siehe 3.7) wieder gelöscht.

GRUNDEINSTELLUNG nach Reset: STOP BY ERROR

3.7 ES = STOP BY ERROR

FORMAT: ES <CR>

Beispiel: ES <CR>

Durch den Befehl "ES" (STOP BY ERROR) wird die Funktion "GO BY ERROR" wieder gelöscht.

Nach Eingabe von ES (STOP BY ERROR) wird bei einem auftretenden Fehler beim Abarbeiten eines der Befehle 2, 17 oder 18 (siehe Tabelle 6.1) das Programm abgebrochen. Ebenso wird die Bearbeitung einer DO-Kommandofolge (siehe 3.4) abgebrochen.

GRUNDEINSTELLUNG nach Reset : STOP BY ERROR

3.8 F = FILL MEMORY

FORMAT: F (Startadr.) (Länge) (Daten) <CR>
oder : F (Startadr.)-(Endadr.) (Daten) <CR>

Beispiel: F 5000 100 AA <CR>
oder: F 5000-50FF AA <CR>

Mit dem Fill-Kommando können beliebige Speicherbereiche mit einer Konstanten gefüllt werden. In unserem Beispiel wird der Speicher von Adresse 5000H bis 50FFH mit dem Datum AAH gefüllt.

Der Debugger benützt den Speicherbereich von Adresse 4000H bis einschließlich 44FFH als Arbeitsspeicher. Auf diesen Bereich dürfen keine schreibenden Zugriffe gemacht werden.

Hinweis: Füllen des Speichers mit einer Datenfolge ist mit dem Move-Kommando möglich (siehe 3.16).

3.9 FC = FLOPPY CONTROL

FORMAT: FC (Disk-ID) (Spur in dezimal) <CR>

Beispiel: FC 1 20 <CR>

Die angegebene Spur 20 des Drive 0 (Mini double density ohne DMA, siehe 6.2) wird gelesen.

Dieses Kommando dient z.B. zur Reparatur von Laufwerken oder des Datenseparators. Folgende Unterkommandos sind hier möglich (Reaktionszeit liegt bei max. 2 Sekunden):

- I ---> gehe auf nächsthöhere Spur (IN)
- O ---> gehe auf nächstniedere Spur (OUT)
- Q ---> zurück in Testdebugger (QUIT)

Bei einer Überschreitung der Bereichsgrenzen (< Spur 0 oder > maximaler Spur) wird nichts getan.

Werden Laufwerke ohne eigenes READY-Signal verwendet (dieses wird dann auf der KDT-Platine auf Low gelegt) und kann der Floppy-controller bei einem solchen Laufwerk keine Daten lesen, so führt dies aus Gründen, die in der Hardware liegen, zum Absturz des Testdebuggers. Dies zeigt sich in einem Verweilen in der Disk-Leseroutine.

3.10 G = GO TO ADDRESS

FORMAT: G (Adresse) <CR>

Beispiel: G 2000 <CR>

Ein Maschinenprogramm ab Adresse 2000H wird gestartet. Dieses wird wie ein Unterprogramm behandelt, d.h. beim Finden eines RET Befehls (0C9H) wird ein Rücksprung zum Testdebugger ausgeführt.

3.11 I = PORT INPUT LOOP

FORMAT: I (Portadresse) <CR>

Beispiel: I 2 <CR>

Der Port mit der Adresse 02H wird laufend gelesen.

Dieses Programm dient für Messungen und erzeugt ein feststehendes Bild auf dem Oszilloskop.
Abbruch erfolgt mit der ESCAPE-Taste.

3.12 J = JUMP

FORMAT: J (Adresse) <CR>

Beispiel: J 6000 <CR>

Wie GO-Kommando (siehe 3.10).

3.13 K= BETRIEBSSYSTEM LADEN

FORMAT: K <CR>

Beispiel: K <CR>

Lädt das Betriebssystem (z.B. KOS, CPM).

3.14 L = LOCATE

FORMAT: L (Startadresse) (Länge) (Daten) <CR>
oder : L (Startadresse) - (Endadresse) (Daten) <CR>

Beispiel: L 5000 2001 33 22 <CR>
oder: L 5000-7000 33 22 <CR>

Im Speicherbereich 5000H bis 7000H wird die BYTE-Kombination 33H 22H gesucht. Wird diese Kombination gefunden, so wird am Monitor ein Adressbereich von 30H um die gefundenen Daten ausgegeben.

3.15 LL = LOCAL

FORMAT: LL <CR>

Beispiel: LL <CR>

Jedes eingegebene Zeichen wird am Monitor ausgegeben, ohne eine Kommando-Ausführung zu bewirken (nur Echo).

Anwendungsbeispiel:
Kommentar zu einem z.Zt. laufenden Dauertest, Mitteilung an einen z.Zt. abwesenden Terminalbenutzer, Austesten von Anzeigeformaten oder Keyboardfunktionen.

Hinweis: Beim Drücken der <CR>-Taste erfolgt ein CARRIAGE RETURN (ASCII ODH) ohne LINE FEED (ASCII OAH).

Abbruch des LOCAL-Zustandes erfolgt mit der ESCAPE-Taste.

3.16 MO = MOVE

FORMAT: MO (Quelladresse) (Zieladresse) (Länge) <CR>

Beispiel: MO 3000 C000 300 <CR>

Der Speicherinhalt von Adresse 3000H bis 32FFH wird in den Speicherbereich C000H bis C2FFH kopiert.

Der Debugger benützt den Speicherbereich von Adresse 4000H bis einschließlich 41FFH als Arbeitsspeicher. Auf diesen Bereich dürfen keine schreibenden Zugriffe gemacht werden.

Hinweis: Mit dem Move-Kommando ist es auch möglich, den Speicher mit einer Folge von Daten zu füllen.

Beispiel: S 5000 41 4C 4D;M0 5000 5003 102 <CR>

Der Speicherbereich von Adresse 5003H bis 5104H wird mit der Datenfolge 41H, 4CH, 40H gefüllt.

3.17 MA = Memory Mapper programmieren

FORMAT : MA (X) <CR> (X = 0,..,7)

Beispiel : MA 2 <CR>

Auf den Platinen KDT Rev. 6.x und TCB/Z80 befinden sich insgesamt 256 KByte Schreib-/Lese-Speicher (RAM), die in Form von vier 64K-Bänken realisiert sind.
Da die Z80-CPU nur 64KByte adressieren kann, ist ein sogenannter Memory-Mapper (LS 610/612) vorhanden, der diesen Speicher verwaltet. Die genaue Funktion des Mappers kann der jeweiligen Hardware-Beschreibung (z.B. TCB/Z80 2.2.1) entnommen werden.

Nach RESET oder Einschalten der Spannungsversorgung ist der Mapper so initialisiert, daß die Speicherbank 0 im 64K-Adreßbereich der CPU liegt. Es gilt dann folgende Speicherverteilung:

```
0000..0FFF : Testdebugger  (PROM)
1000..17FF : Boot-Software (PROM)
1800..1FFF : Disktest      (PROM)
2000..3FFF : frei  (RAM)
4000..41FF : Debugger RAM
4200..4EFF : frei (RAM)
4F00..4FFF : Boot RAM
5000..FFFF : frei (RAM)
```

Mit dem MA-Kommando kann nun entweder die erste oder die zweite Hälfte jeder 64K-Bank in den Bereich 8000..FFFF gemappt werden.

Im Bereich 0000..7FFF liegt immer die erste Hälfte von Bank 0.

```
MA oder MA 0 : 8000..FFFF = 2.Hälfte von Bank 0 (Standard)
        MA 1 : 8000..FFFF = 1.Hälfte von Bank 0
        MA 2 : 8000..FFFF = 1.Hälfte von Bank 1 (Beispiel)
        MA 3 : 8000..FFFF = 2.Hälfte von Bank 1
        MA 4 : 8000..FFFF = 1.Hälfte von Bank 2
        MA 5 : 8000..FFFF = 2.Hälfte von Bank 2
        MA 6 : 8000..FFFF = 1.Hälfte von Bank 3
        MA 7 : 8000..FFFF = 2.Hälfte von Bank 3
```

3.18 Memory-Test

Für den Speichertest stehen zwei Kommandos zur Verfügung, MT und MX . Mit MT kann Speicher nur innerhalb einer Bank getestet werden, MX testet in allen Bänken.

3.18.1 MT = MEMORY-TEST innerhalb einer Speicherbank

```
FORMAT 1:   MT (Startadresse) (Länge) <CR>
oder    :   MT (Startadresse)-(Endadresse) <CR>
FORMAT 2:   MT (Startadresse) (Länge) (Anzahl Loops) <CR>
oder    :   MT (Startadresse)-(Endadresse) (Anzahl Loops) <CR>

Beispiel 1: MT 6000 1000 <CR>
oder      : MT 6000-6FFF <CR>
Beispiel 2: MT 6000 1000 33 <CR>
oder      : MT 6000-6FFF 33 <CR>
```

Der Speicherbereich von Adresse 6000H bis 6FFFH wird getestet. Die Angabe der Anzahl der Loops ist optional (Beispiel 2). Der Test läuft in Beispiel 1 solange, bis ein Abbruch mit <ESC> (siehe 1.2.1) oder mit einer beliebigen Taste (siehe 1.2.2) erfolgt. In Beispiel 2 wird der Memorytest insgesamt 33 mal (= Anzahl der eingegebenen Loops) durchgeführt.
Das Testergebnis erscheint am Monitor entweder als:

"NO ERROR DETECTED !"

oder als "MEMORY-ERROR AT ADDRESS "

unter Angabe der fehlerhaften Speicherstelle und deren IST- und SOLL-Wert. Dabei ist zu beachten, daß der Memorytest im Fehlerfall dann abgebrochen wird, wenn "ES" (siehe 3.7) aktiviert ist.

Hinweis: Der Memorytest ist zerstörend und darf daher nicht im Speicherbereich von 4000H bis 41FFH durchgeführt werden, da dies der Arbeitsspeicher des Testdebuggers ist.

3.18.2 MX = MEMORY-TEST über alle Bänke

```
FORMAT 1 :  MX (Startadresse) (Länge) <CR>
  oder   :  MX (Startadresse)-(Endadresse) <CR>
FORMAT 2 :  MX (Startadresse) (Länge) (Anzahl Loops) <CR>
  oder   :  MX (Startadresse)-(Endadresse) (Anzahl Loops) <CR>
```

Beispiel : MX 4200-FFFF 2 <CR>

Der Bereich von 4200-FFFF wird getestet, wobei im Bereich 8000..FFFF über alle Bänke getestet wird.

Mit MX kann sowohl die Funktionsfähigkeit des Memory Mappers als auch aller Speicherbänke getestet werden. Der Ablauf dabei ist folgendermaßen :

a) Die Bänke 1..3 werden mit einem bestimmten Datenmuster gefüllt (1.Hälfte Bank 1 mit 02, 2.Hälfte mit 03 usw.).

b) In Bank 0 wird der angegebene Bereich getestet (genau wie bei MT-Kommando).

c) Falls Test in Bank 0 erfolgreich, wird geprüft ob die Datenmuster in den folgenden Bänken unverändert geblieben sind. Falls nicht erfolgt in der entsprechenden Bank eine Fehlermeldung : ERROR IN BANK X (X = 1..3). Der Mapper bleibt in diesem Fall auf diese Bank programmiert, so daß die betreffende Bank sofort überprüft werden kann .

d) Falls kein Fehler auftrat, wird die erste Hälfte von Bank 1 in den Bereich 8000..FFFF gemappt (entspricht dem Kommando MA 2)

e) Der angegebene Bereich wird getestet.

f) Das Datenmuster in den folgenden Bänken wird geprüft.

g) Falls kein Fehler, wird die zweite Hälfte von Bank 1 nach 8000..FFFF gemappt usw.

Während des Tests wird ausgegeben, welche Bank gerade getestet wird, bei erfolgreichem Ablauf ergibt sich (pro Durchlauf) der Ausdruck :

```
MEMORY TEST BANK 0 1 1 2 2 3 3   NO ERROR DETECTED !
```

Da die Bänke 1..3 in zwei Hälften getestet werden, erscheint die Bank-Nr. zweimal im Ausdruck.

Im Bereich 0000..7FFF liegt immer die 1.Hälfte von Bank 0, dies bedeutet, daß bei dem oben angegebenen Beispiel der Bereich 4200-7FFF von Bank 0 mehrfach (pro Durchlauf 7 mal) getestet wird.

Beispiele für mögliche Fehlermeldungen :

MEMORY TEST BANK 0 ERROR IN BANK 1

Bank 0 wurde erfolgreich getestet, in Bank 1 wurde das am Anfang eingetragene Datenmuster (02/03) nicht gefunden. Es liegt entweder ein Speicherfehler in Bank 1 vor oder der Mapper funktioniert nicht.

MEMORY TEST BANK 0 ERROR IN BANK 3

Bank 0 erfolgreich getestet, in Bank 3 wurde das eingetragene Datenmuster (06/07) nicht gefunden. Mapper vermutlich in Ordnung, wahrscheinlich Speicherfehler in Bank 3 .

Falls nach einer Fehlermeldung oder nach <ESC> abgebrochen wird, bleibt der Memory Mapper in dem Zustand indem er zuletzt war.

Bei normalem Ende (ohne Fehler) wird der Mapper wieder auf Bank 0 initialisiert.

3.19 O = PORT OUTPUT LOOP

FORMAT: O (Portadresse) (Daten) <CR>

Beispiel: O 2 55 <CR>

Auf den Port 02H wird laufend 55H geschrieben.

Dieser Test dient für Messungen und erzeugt ein feststehendes Bild auf dem Oszilloskop.

Abbruch erfolgt mit der ESCAPE-Taste.

3.20 OF = SERIELLE SCHNITTSTELLE AUSSCHALTEN

FORMAT: OF <CR>

Beispiel: OF <CR>

Mit diesem Kommando wird die serielle Schnittstelle abgeschaltet. An der Initialisierung wird nichts geändert.

Welche serielle Schnittstelle (A oder B) ausgeschaltet wird, hängt von der vorliegenden Debugger-Version ab (siehe 1.1).

3.21 ON = SERIELLE SCHNITTSTELLE EINSCHALTEN

FORMAT: ON <CR>

Beispiel: ON <CR>

Mit diesem Kommando wird die serielle Schnittstelle eingeschaltet, die Baudrate ist auf 9600 Baud gestellt.

Die Initialisierung des SIO geschieht sofort nach Reset oder Kaltstart und nicht erst nach der Ausführung des ON-Kommandos.

Welche serielle Schnittstelle (A oder B) eingeschaltet wird, hängt von der Debugger-Version ab (siehe 1.1).

3.22 OV = OVERLAY EINSCHALTEN

FORMAT: OV <CR>

Beispiel: OV <CR>

Nur bei entsprechender Hardware für Bildverarbeitung möglich.

Mit dem OV-Kommando ist es möglich, das Computerbild mit einem Videobild (TV-Kamera oder Videorecorder) zu überlagern.

3.23 P = PAUSE

FORMAT 1: P <CR>
FORMAT 2: P (Anzahl der Loops) <CR>

Beispiel 1: D 0 100;P;MV;P;MT 4500 2000 <CR>
Beispiel 2: S 5000 /D 0 50;P 5;D 5000 30;D0 5000 <CR>

Dies Kommando kann nur in einer Kommandofolge sinnvoll integriert sein. Es unterbricht die laufende Abarbeitung einer Kommandofolge, um auf eine Eingabe zu warten. Beim Drücken einer beliebigen Taste wird das nächste Kommando bearbeitet. Die ESC-Taste (siehe 1.2.1) bricht die Befehlsfolge ab. Damit ist eine Kontrolle über die einzelnen Schritte der Kommandofolge möglich.

Wird die optionelle Angabe der Anzahl der Loops benutzt (Beispiel 2), so reagiert das Pause-Kommando nur nach der angegebenen Anzahl der Durchläufe (hier 5 Durchläufe).

3.24 RA = PROM - RAM UMSCHALTUNG

FORMAT: RA <CR>

Beispiel: RA <CR>

Der Inhalt der E-Proms wird in die untere RAM-Bank kopiert, die Adresse 0000H-2000H als Prombereich abgeschaltet und auf RAM umgeschaltet.

3.25 RD = READ LOOP

FORMAT: RD (Adresse) <CR>

Beispiel: RD 3000 <CR>

Die Adresse 3000H wird laufend gelesen.

Dieses Kommando dient für Messungen am RAM und erzeugt ein feststehendes Bild am Oszilloskop.
Abbruch erfolgt mit der ESC-Taste.

3.26 RC = RECALIBRATE

FORMAT : RC (Drive-ID) <CR>

Der Schreib/Lesekopf des angegeben Laufwerks wird auf Spur 0 gefahren.

3.27 RS = READ SECTOR

FORMAT 1: RS (Drive-ID) (Track Sector) (Adresse) <CR>
FORMAT 2: RS <CR>

Beispiel 1: RS 1 2001 5000 <CR>
Beispiel 2: RS <CR>

Beispiel 1 liest Daten gemäß der Drive-ID (siehe 6.2) von Drive 0 (Mini double density ohne DMA) Spur 20 Sektor 1 auf die Adresse 5000H bis 50FFH (wegen double density). Anschließend wird der Speicherbereich 5000H bis 50FFH angezeigt.

Bei single density wird nur ein Speicherbereich von 80H beschrieben und angezeigt .

Bei fehlendem Parameterfeld (Beispiel 2) werden die zuletzt bei einem Disk-Kommando verwendeten Parameter benützt.

Werden Laufwerke ohne eigenes READY-Signal verwendet (dieses wird dann auf der KDT-Platine auf Low gelegt) und kann der Floppy-controller bei einem solchen Laufwerk keine Daten lesen, so führt dies aus Gründen, die in der Hardware liegen, zum Absturz des Testdebuggers. Dies zeigt sich in einem Verweilen in der Disk-Leseroutine.

Der Debugger benützt den Speicherbereich von Adresse 4000H bis einschließlich 41FFH als Arbeitsspeicher. Der Boot-PROM verwendet den Bereich 4F00..4FFF . Auf diese Bereiche dürfen keine schreibenden Zugriffe gemacht werden.

3.28 RT = READ TRACK

FORMAT 1: RT (Drive-ID) (Track) (Adresse) <CR>
FORMAT 2: RT <CR>

Beispiel 1: RT A 20 5000 <CR>
Beispiel 2: RT <CR>

Beispiel 1 liest Daten gemäß der Drive-ID (siehe 6.2) von Drive 0 (Mini double density mit DMA) Spur 20 auf die Adresse 5000H bis 5FFFH (wegen double density). Anschließend wird der Speicherbereich 5000H bis 515FH angezeigt. Der gesamte Datensatz kann mit dem Kommando D 5000 1000 angezeigt werden. Bei single density wird nur ein Block von 800H Bytes eingelesen.

Bei fehlendem Parameterfeld (Beispiel 2) werden die zuletzt bei einem Disk-Kommando verwendeten Parameter benützt.

Werden Laufwerke ohne eigenes READY-Signal verwendet (dieses wird dann auf der KDT-Platine auf Low gelegt) und kann der Floppy-controller bei einem solchen Laufwerk keine Daten lesen, so führt dies aus Gründen, die in der Hardware liegen, zum Absturz des Testdebuggers. Dies zeigt sich in einem Verweilen in der Disk-Leseroutine.

Der Debugger benützt den Speicherbereich von Adresse 4000H bis einschließlich 41FFH als Arbeitsspeicher. Der Boot-PROM verwendet den Bereich 4F00..4FFF . Auf diese Bereiche dürfen keine schreibenden Zugriffe gemacht werden.

3.29 S = SET MEMORY

FORMAT 1: S (Adresse) (Daten) (Daten).....(Daten) <CR>
FORMAT 2: S (Adresse) /(TEXT) <CR>

Beispiel 1: S 5000 34 56 78 90 <CR>
Beispiel 2: S 5000 /MT 3000 1000 9;CP 8000 C000 4000 <CR>

In Beispiel 1 wird auf die Adresse 5000H das Datum 34H, auf die Adresse 5001H das Datum 56H usw. geschrieben.

Beispiel 2 zeigt das Eintragen eines alphanumerischen Textes. Dies wird durch Eingabe eines SLASH (/) nach der Startadresse erreicht.

Der Debugger benützt den Speicherbereich von Adresse 4000H bis einschließlich 41FFH als Arbeitsspeicher. Der Boot-PROM verwendet den Bereich 4F00..4FFF . Auf diese Bereiche dürfen keine schreibenden Zugriffe gemacht werden.

3.30 SP = SET PORT

FORMAT: SP (Portadresse) (Daten) (Daten) ..... (Daten) <CR>

Beispiel: SP 80 55 <CR>

Dieser Befehl schreibt das Datum 55H auf den Port 80H.

Beispiel für die Initialisierung des SIO Kanal A (Datenport: 04H, Controllport: 06H) auf folgende Werte:

Baudrate 2400
CTS aktiv, 8 Bit/Char, kein Parity, Auto Enable, 2 Stoppbits

SP A 47 32 <CR> (für den CTC1 Kanal 2 Baudrate 2400)
SP 6 04 4C 05 EA 03 E1 <CR> (für den SIO Port A)

Es wird darauf hingewiesen, daß bei nicht bedienten CTS-Signal (z.B. bei 3-Draht-Leitung) im angeführten Beispiel keine Übertragung möglich ist (wegen Programmierung von CTS aktiv und Auto Enable) !

3.31 ST = SET REFRESH TIME

FORMAT 1: ST <CR>
FORMAT 2: ST (Konstante) <CR>

Beispiel 1: ST <CR>
Beispiel 2: ST 10 <CR>

Die Refresh Zyklen des Speichers können beim Memorytest (auch Video) unterdrückt werden. In diesem Beispiel werden sie 10H * 2 ms = 32 ms unterdrückt (10H = 16 Dezimal -> 32 ms).

Die Konstante kann bis maximal 20H erhöht werden, wobei der Grenzwert stark von den jeweils verwendeten RAM-Typen abhängt (teilweise ist auch 80H und mehr möglich).

VOREINSTELLUNG : 5H

3.32 TP = TESTPROGRAMM

FORMAT: TP (Drive-ID) <CR>

Beispiel: TP 2 <CR>

Mit diesem Kommando werden gemäß Drive-ID (siehe 6.2) Daten von Drive 1 (Mini double density ohne DMA) Spur 0 Sektor 1 in den Speicher ab Adresse 5000H eingelesen. Anschließend erfolgt automatisch das Kommando "D0 5000".

ANWENDUNG:
Auf Spur 0 Sektor 1 einer Diskette wird eine Kommandofolge geschrieben. Die Abarbeitung der Kommandos kann dann durch das TP-Kommando gestartet werden. Somit lassen sich Testprogramme sehr einfach implementieren.

3.33 W = WRITE LOOP

FORMAT: W (Adresse) (Daten) <CR>

Beispiel: W 5000 55 <CR>

Auf die Adresse 5000H wird laufend das Datum 55H geschrieben. Dieses Kommando dient Messungen am RAM und erzeugt ein feststehendes Bild am Oszilloskop.

Abbruch erfolgt mit der ESC-Taste.

Der Debugger benützt den Speicherbereich von Adresse 4000H bis einschließlich 41FFH als Arbeitsspeicher. Der Boot-PROM verwendet den Bereich 4F00..4FFF . Auf diese Bereiche dürfen keine schreibenden Zugriffe gemacht werden.

3.34 WR = WRITE / READ LOOP

FORMAT: WR (Adresse) (Daten) <CR>

Beispiel: WR 5000 55 <CR>

Auf die Adresse 5000H wird laufend das Datum 55H geschrieben und wieder gelesen.

Dieses Kommando dient Messungen am RAM und erzeugt ein feststehendes Bild am Oszilloskop.

Abbruch erfolgt mit der ESC-Taste.

Der Debugger benützt den Speicherbereich von Adresse 4000H bis einschließlich 41FFH als Arbeitsspeicher. Der Boot-PROM verwendet den Bereich 4F00..4FFF . Auf diese Bereiche dürfen keine schreibenden Zugriffe gemacht werden.

3.35 WS = WRITE SECTOR

FORMAT 1: WS (Drive-ID) (Track Sector) (Adresse) <CR>
FORMAT 2: WS <CR>

Beispiel 1: WS 2 2201 6000 <CR>
Beispiel 2: WS <CR>

Beispiel 1 schreibt Daten gemäß Drive-ID (siehe 6.2) von Adresse 6000H bis 60FFH auf Drive 1 (Mini double density ohne DMA) Spur 22 Sector 1.

Bei fehlendem Parameterfeld (Beispiel 2) werden die zuletzt bei einem Floppy-Befehl verwendeten Parameter verwendet.

Werden Laufwerke ohne eigenes READY-Signal verwendet (dieses wird dann auf der KDT-Platine auf Low gelegt) und kann der Floppycontroller bei einem solchen Laufwerk keine Daten lesen, so führt dies aus Gründen, die in der Hardware liegen, zum Absturz des Testdebuggers. Dies zeigt sich in einem Verweilen in der Disk-Leseroutine.

3.36 WT = WRITE TRACK

FORMAT 1: WT (Drive-ID) (Track) (Adresse) <CR>
FORMAT 2: WT <CR>

Beispiel 1: WT A 20 5000 <CR>
Beispiel 2: WT <CR>

Beispiel 1 schreibt Daten auf Drive 0 (Mini double density mit DMA) Spur 20 von der Adresse 5000H bis 5FFFH (wegen double density). Anschließend wird der Speicherbereich 5000H bis 515FH angezeigt. Bei single density wird nur ein Block von 800H Bytes geschrieben.

Bei fehlendem Parameterfeld (Beispiel 2) werden die zuletzt bei einem Disk-Kommando verwendeten Parameter benützt.

Werden Laufwerke ohne eigenes READY-Signal verwendet (dieses wird dann auf der KDT-Platine auf Low gelegt) und kann der Floppy-controller bei einem solchen Laufwerk keine Daten lesen, so führt dies aus Gründen, die in der Hardware liegen, zum Absturz des Testdebuggers. Dies zeigt sich in einem Verweilen in der Disk-Leseroutine.

4. Systemaufrufe und Einsprungpunkte

Mit dem Test-Debugger erhält der Benutzer die Möglichkeit, fertige Test- und Diagnoseprogramme abarbeiten zu lassen, sowie ein Paket von Unterprogrammen, die über speziell geschaffene Einsprungpunkte aufgerufen werden können.

Damit ist ein einfacher Weg geschaffen, schnell und problemlos Maschinenprogramme zu schreiben, die auch komplizierte Ein- und Ausgaben durchführen können. Die zur Verfügung stehenden Einsprungpunkte sind nachstehend mit Beispielen beschrieben.

4.1 Kaltstart

FORMAT:	C3 00 00	MNEMO-CODE:	JP 0000H
oder :	C7		RST 0H

Es erfolgt eine Hard- und Software-Neuinitialisierung. Der Kaltstart enspricht einem RESET.

Hinweis: Der PIO-Baustein hat keinen Reset-Eingang und bleibt daher auch nach einem Reset oder Kaltstart initialisiert.

4.2 Warmstart

FORMAT: C3 02 00 MNEMO-CODE: JP 0002H

Der Warmstart initialisiert die Software und setzt den Stackpointer neu.

Nach Abarbeitung dieser Routine meldet sich der Testdebugger mit der Ausgabe der vollen Überschrift.

4.3 Monitorausgabe

FORMAT: CD 08 00 MNEMO-CODE: CALL 0008H
oder : CF RST 8H

Eingaberegister: <A>
Ausgaberegister: keine
Zerstörte Register: keine

Der RST 8 gibt ein ASCII-Zeichen, das im Register <A> steht, auf dem Bildschirm an der aktuellen Cursor-Stelle aus.

Hinweis: Wenn die serielle Schnittstelle A eingeschaltet ist ("ON-Kommando" siehe 3.21), erfolgen sämtliche Ein- und Ausgaben auch zusätzlich über die serielle Schnittstelle.

Beispiel: S 5000 3E 55 CF C9;J 5000 <CR>

Erklärung:

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
5000	3E 55	LD A,55H	ASCII-CODE FÜR "U" IN <A>
5002	CF	RST 8H	AUSGABE EINES ASCII-ZEICHENS
5003	C9	RET	RÜCKSPRUNG IN DEN DEBUGGER

4.4 Eingabe eines ASCII-Zeichens mit ECHO

FORMAT: CD 0B 00 MNEMO-CODE: CALL 000BH

Eingaberegister: keine
Ausgaberegister: <A>
Zerstörte Register: <AF>

Jedes eingegebene Zeichen wird auf dem Bildschirm angezeigt (optional serielle Schnittstelle siehe 3.21).

BEISPIEL: S 5000 CD 0B 00 C9;J 5000 <CR>

Nun kann ein beliebiges Zeichen eingegeben werden, das sofort am Bildschirm an der aktuellen Cursor-Stelle erscheint.

4.5 Prüfung auf Eingabe

FORMAT: CD OE 00 MNEMO-CODE: CALL 000EH

Eingaberegister: keine
Ausgaberegister: <F> (Z-Flag)
Zerstörte Register: <AF>

Soll in einem Programm abgefragt werden, ob eine Eingabe (optional serielle Schnittstelle siehe 3.21) erfolgt ist, bietet sich dieses Unterprogramm als Hilfe an. Ist ein Zeichen eingegeben worden, so wird das Z-Flag zurückgesetzt. Wurde kein Zeichen eingegeben, so ist das Z-Flag gesetzt. Gleichzeitig kann im Register <A> das eingegebene Zeichen abgefragt werden.

Hinweis: Das eingelesene Zeichen bleibt weiterhin present. Es gilt erst als ausgelesen, wenn es über Einsprung 0BH (siehe 4.4) oder 10H (siehe 4.6) eingelesen wurde.

4.6 Eingabe eines ASCII-Zeichens

FORMAT: CD 10 00 MNEMO-CODE: CALL 0010H
oder : D7 RST 10H

Eingaberegister: keine
Ausgaberegister: <A>
Zerstörte Register: <AF>

Diese Subroutine wartet auf die Eingabe eines ASCII-Zeichens (optional serielle Schnittstelle, siehe 3.21) und liest dieses in das Register <A> ein.

Beispiel: S 5000 D7 32 50 50 C9;J 5000 <CR>

Erklärung:

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
5000	D7	RST 10H	ASCII-ZEICHEN IN AKKU
5001	32 50 50	LD (5050H),A	ASCII-ZEICHEN NACH ADRESSE 5050H SCHREIBEN
5004	C9	RET	RÜCKSPRUNG IN DEN DEBUGGER

Mit dem Display-Befehl kann leicht nachgeprüft werden, daß das eingebene Zeichen wirklich von Register <A> auf die Adresse 5050H geladen worden ist.

4.7 Ausgabe von "CURSOR AUF ANFANG DER NÄCHSTEN ZEILE" und Testausgabe

FORMAT: CD 13 00 MNEMO-CODE: CALL 0013H

Eingaberegister: <HL>
Ausgaberegister: keine
Zerstörte Register: <AF> (A=0),<HL>

Der Cursor wird auf den Anfang der nächsten Zeile gesetzt und anschließend erfolgt ein Sprung auf den Einsprungpunkt 18H, der einen Text ausgibt, der mit 00H als Ende-Markierung versehen sein muß. Die Adresse des Textes muß vorher in das Register <HL> geladen werden.

Beispiel: S 5050 /TEXTAUSGABE <CR>
S 505B 00 <CR> (Ende-Markierung)
S 5000 21 50 50 CD 13 00 C9;J 5000 <CR>

Ausgabe: TEXTAUSGABE

Erklärung:

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
5000	21 50 50	LD HL,5050H	ADRESSE DES TEXTANFANGS NACH <HL>
5003	CD 13 00	CALL 0013H	TEXT AM ANFANG DER NÄCHSTEN ZEILE AUSGEBEN
5006	C9	RET	RÜCKSPRUNG IN DEN DEBUGGER

4.8 Textausgabe

FORMAT: CD 18 00 MNEMO-CODE: CALL 0018H
oder : DF RST 18H

Eingaberegister: <HL>
Ausgaberegister: keine
Zerstörte Register: <AF> (A=0),<HL>

Der Text wird ab der aktuellen Cursorposition ausgegeben, und zwar bis 00H, was als Ende-Markierung vereinbart ist.

4.9 Ausgabe von 2 Byte als 4 ASCII-Zeichen

FORMAT: CD 1B 00 MNEMO-CODE: CALL 001BH

Eingaberegister: <HL>
Ausgaberegister: keine
Zerstörte Register: <A>

Der Inhalt des Registers <HL> wird als 4-stellige Hexzahl aufgefaßt, die Hexziffern werden in ASCII-Zeichen umcodiert und anschließend ausgegeben.

Beispiel: S 5000 21 34 12 CD 1B 00 C9;J 5000 <CR>

Erklärung:

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
5000	21 34 12	LD HL,1234H	HEXZAHL 1234 IN <HL> SCHREIBEN
5003	CD 1B 00	CALL 001BH	INHALT VON <HL> IN ASCII WANDELN UND AUSGEBEN
5006	C9	RET	RÜCKSPRUNG IN DEN DEBUGGER

4.10 Ausgabe eines Bytes als 2 ASCII-Zeichen

FORMAT: CD 20 00 MNEMO-CODE: CALL 0020H
oder : EF RST 20H

Eingaberegister: <A>
Ausgaberegister: keine
Zerstörte Register: keine

Das im Register <A> stehende Byte wird in 2 ASCII-Zeichen umgewandelt und ausgegeben.

Beispiel: S 5000 3E 55 E7 C9;J 5000 <CR>

Erklärung: (vergleiche dazu mit Beispiel in 4.3)

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
5000	3E 55	LD A,55H	ASCII-CODE FÜR "U" IN <A>
5002	E7	RST 20H	AUSGABE VON "55"
5003	C9	RET	RÜCKSPRUNG IN DEN DEBUGGER

4.11 Ausgabe von "CURSOR AUF ANFANG DER NÄCHSTEN ZEILE"

FORMAT: CD 23 00 MNEMO-CODE: CALL 23H

Eingaberegister: keine
Ausgaberegister: keine
Zerstöre Register: keine

Es erfolgt ein Sprung von der aktuellen Cursorposition auf den Anfang der nächsten Zeile.

4.12 Ausgabe von 3 Leerzeichen

FORMAT: CD 26 00 MNEMO-CODE: CALL 0026H

Eingaberegister: keine
Ausgaberegister: keine
Zerstörte Register: keine

Es werden 3 Leerzeichen ausgegeben. Ansonsten siehe 4.14.

4.13 Ausgabe von 2 Leerzeichen

FORMAT: CD 27 00 MNEMO-CODE: CALL 0027H

Eingaberegister: keine
Ausgaberegister: keine
Zerstörte Register: keine

Es werden 2 Leerzeichen ausgegeben. Ansonsten siehe 4.14.

4.14 Ausgabe von einem Leerzeichen

FORMAT: CD 00 28 MNEMO-CODE: CALL 0028H
oder : EF RST 28H

Eingaberegister: keine
Ausgaberegister: keine
Zerstörte Register: keine

Es wird ein Unterprogramm zur Ausgabe eines Leerzeichens ausgegeben.

Beispiel: S 5000 3E 3F CF CF EF C3 00 50;J 5000 <CR>

Ausgabe: ?? ?? ?? ?? ?? ?? ?? ?? ?? .....

Erklärung:

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
5000	3E 3F	LD A,3FH	ASCII-CODE VON "?" IN AKKU
5002	CF	RST 8H	AUSGABE DES 1. FRAGEZEICHENS
5003	CF	RST 8H	AUSGABE DES 2. FRAGEZEICHENS
5004	EF	RST 28H	AUSGABE EINES LEERZEICHENS
5005	C3 00 50	JP 5000H	SPRINGE ZUM PROGRAMMANFANG

4.15 Programmunterbrechung

FORMAT: CD 33 00 MNEMO-CODE: CALL 0033H

Eingaberegister: keine
Ausgaberegister: <F> (Z-Flag wird abgefragt)
Zerstörte Register: <AF>

Mit dieser Routine wird geprüft, ob das laufende Programm unterbrochen werden soll. Die Abfrage wird dazu an den gewünschten Stellen im Programm miteingebaut. Ist das Z-Flag gesetzt, so wird keine Unterbrechung gefordert.

Beispiel: S 5000 3E 24 CF CD 33 00 C0 C3 00 50;J 5000 <CR>

Ausgabe: $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$.....

Wird jetzt während des Programmablaufs ein 'W' eingegeben unterbricht der Debugger die Programmfolge, irgend ein weiteres eingegebenes Zeichen läßt das Programm fortfahren (siehe 1.2.2). Mit der ESC-Taste wird das Programm völlig abgebrochen (siehe 1.2.1).

Erklärung:

```
ADRESSE   OP-CODE   MNEMO-CODE     KOMMENTAR

5000      3E 24     LD A,24H       ASCII-CODE VON "$" IN AKKU
5002      CF        RST 8H         AUSGABE VON "$"
5003      CD 33 00  CALL 0033H     PROGRAMMUNTERBRECHUNG ?
5006      C0        RET NZ         RETURN, WENN UNTERBRECHUNG
5007      C3 00 50  JP 5000H       SPRUNG AN DEN PROGRAMMBEGINN
```

4.16 Programmabbruch mit Meldung

```
FORMAT: CD 38 00        MNEMO-CODE: CALL 0038H
oder:   FF                          RST 38H

Eingaberegister:    keine
Ausgaberegister:    keine
Zerstörte Register: alle
```

Wird in einem Programm der Befehl RST 38H (0FFH) erkannt, so wird ein Unterprogramm aufgerufen, welches das laufende Programm abbricht und folgende Meldung ausgibt:

BREAK AT (Adresse)

Als Adresse wird diejenige Speicherstelle angegeben, an der der nächste auszuführende Maschinenbefehl steht. Nach der Textausgabe erfolgt ein Warmstart (siehe 4.2).

Beispiel: S 5000 00 00 00 00 FF;J 5000 <CR>

Ausgabe: BREAK AT 5005

4.17 Einlesen eines Textes in einen Eingabepuffer

FORMAT: CD 3E 00 MNEMO-CODE: CALL 003EH

Eingaberegister: <A>,<HL>
Ausgaberegister: <B>
Zerstörte Register: <AF>,<B>,<HL>

Es wird in das Register <HL> die Startadresse und in das Register <A> die Länge des Eingabepuffers eingelesen. Diese Länge ist inklusive des abschließenden <CR> (ASCII ODH). Nach der Rückkehr aus der Eingaberoutine steht im Register <B> die tatsächliche Anzahl der eingegebenen Zeichen ohne abschließendes <CR>.

Beispiel:S 5000 21 50 50 3E 10 CD 3E 00 78 32 10 50 C9;J 5000 <CR>

Es können jetzt 15 (= 10H) Zeichen eingegeben werden. Wird das 16-te Zeichen eingegeben, ertönt ein akustisches Signal um anzuzeigen, daß der Eingabepuffer voll ist.

Erklärung:

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
5000	21 50 50	LD HL,5050H	ZEIGER IN <HL> AUF EINGABEPUFFER
5003	3E 10	LD A,10H	PUFFERLÄNGE 10H BYTES
5005	CD 3E 00	CALL 003EH	EINGABE
5008	78	LD A,B	LADE ANZAHL NACH <A>
5009	32 10 50	LD (5010H),A	LADE ANZAHL NACH ADRESSE 5010H
500C	C9	RET	RÜCKSPRUNG IN DEN DEBUGGER

Eingabe: 123456789ABCDEF <CR>

Mit dem Display-Kommando (D 5000 60) kann die Funktion des Programms überprüft werden. Man sieht auf Adresse 505FH das ODH (von <CR>) und auf Adresse 5010H die Anzahl der eingegebenen Bytes (OFH = 15 Zeichen).

4.18 Auslesen eines Zeichens aus Eingabepuffer

FORMAT: CD 41 00 MNEMO-CODE: CALL 0041H

Eingaberegister: keine
Ausgaberegister: <A>
Zerstörte Register: <AF>

Ein Zeichen, auf welches der Eingabepuffer-Zeiger zeigt (siehe 5.5), wird ins Register <A> eingelesen.

Hinweis: Ein Strichpunkt (;) wird als ODH (ASCII-Code für CR) interpretiert und eine Markierung gesetzt, daß noch weitere Kommandos folgen.

4.19 Pufferzeiger Hochzählen

FORMAT: CD 44 00 MNEMO-CODE: CALL 0044H

Eingaberegister: keine
Ausgaberegister: keine
Zerstörte Register: keine

Der Eingabepuffer-Zeiger (siehe 5.5) wird um eins erhöht. Damit können Manipulationen an der Reihenfolge des Einlesens vom Eingabepuffer (siehe 4.18) vorgenommen werden.

4.20 Pufferzeiger herunterzählen

FORMAT: CD 47 00 MNEMO-CODE: CALL 0047H

Eingaberegister: keine
Ausgaberegister: keine
Zerstörte Register: keine

Der Eingabepuffer-Zeiger (siehe 5.5) wird um eins erniedrigt. Damit können Manipulationen an der Reihenfolge des Einlesens vom Eingabepuffer (siehe 4.18) vorgenommen werden.

4.21 Einsprungpunkt für NON-MASKABLE INTERRUPT (NMI)

FORMAT: C3 66 00 MNEMO-CODE: JP 0066H

Eingaberegister: keine
Ausgaberegister: keine
Zerstörte Register: alle

Auf Adresse 0066H steht eine Interrupt-Service-Routine für den NMI, welche einen Sprungbefehl auf die Adresse 400CH enthält (siehe 5.8). Als Voreinstellung steht auf der Adresse 400CH ein Sprung auf die Adresse 007FH. Es wird folgender Text ausgegeben:

NMI AT (Adresse)

Nach Ausgabe des Textes erfolgt ein Warmstart (siehe 4.2).

Soll bei NMI nicht die Voreinstellung erwünscht sein, so hat der Benutzer die Möglichkeit, durch Manipulation der Adresse 400CH eine andere Interrupt-Service-Routine aufzurufen.

Beispiel für eine Änderung der Interrupt-Service-Routine :

S 400C FB ED 45 <CR>

Erklärung:

ADRESSE	OP-CODE	MNEMO-CODE	KOMMENTAR
4013	FB	EI	INTERRUPT FREIGEBEN
4014	ED 45	RETN	RÜCKSPRUNG AUS DER INTERRUPT-SERVICE-ROUTINE FÜR NMI

5. RAM-Speicheradressen

Folgende Speicheradressen stehen dem Anwender zur Verfügung:

```
4000H:      STATUS
4001H:      LETZTE EINGABE
4002H:      SCROLL-ADRESSE LOW  BYTE
4003H:      SCROLL-ADRESSE HIGH BYTE
4004H:      CURSOR-ADRESSE LOW  BYTE
4005H:      CURSOR-ADRESSE HIGH BYTE
4006H:      EINGABE PUFFERZEIGER LOW  BYTE
4007H:      EINGABE PUFFERZEIGER HIGH BYTE
4008H:      REFRESH ZEIT KONSTANTE
4009H:      WARTESCHLEIFE FÜR MEMORYTEST      (3 BYTES)
400CH:      NMI UNTERPROGRAMM FÜR MEMORYTEST (3 BYTES)
400FH:      WARTESCHLEIFE FÜR MONITORAUSGABE (3 BYTES)
4012H:      CURSOR ON-OFF FLAG
4013H:      INVERT SCREEN FLAG
4014H:      SIO-AUSGABE ROUTINE (3 BYTES)
4017H:      ESCAPE ROUTINE (3 BYTES)
401AH:      RECEIVE ROUTINE FÜR TESTSYSTEM (3 BYTES)
```

5.1 Status

Diese Speicherzelle beinhaltet den Status des Statusports 1CH. Da dieser Port nicht gelesen werden kann, muß der jeweilige Zustand zusätzlich auf der Adresse 4000H gespeichert werden. Dabei bedeuten (1/0) :

```
BIT 0:          WATCHDOG ENABLE/DISABLE
BIT 1:          4 MHZ / 2MHZ SYSTEM-TAKT
BIT 2:          AUDIO ENABLE/DISABLE
BIT 3:          NOT USED
BIT 4:          SIO-A und DMA / FDC und DMA
BIT 5:          PROM OFF / ON
BIT 6:          MINI-FLOPPY / STANDARD-FLOPPY
BIT 7:          FLOPPY MOTOR ON/OFF
```

Jede Änderung muß sowohl auf den Statusport 1CH als auch auf die Speicherzelle 4000H geschrieben werden.

5.2 Letzte Eingabe

Jedes durch Interrupt empfangene Zeichen (Keyboard, serielle Schnittstelle) wird auf der Speicherzelle 4001H abgelegt, bis es ausgelesen wird (siehe 4.4, 4.5, 4.6). Nach dem Auslesen wird die Speicherzelle auf 00H gesetzt um anzuzeigen, daß kein Zeichen präsent ist.

5.3 Scroll-Adresse

Die Scroll-Adresse ist die aktuelle Anfangsadresse der Bildschirm-Ausgabe. Die Adresse wird relativ zur Anfangsadresse des Bildspeichers (0000H) angegeben. Durch Umsetzen der Scroll-Adresse kann im Bildspeicher "geblättert" werden. Das Low-Byte der Scroll-Adresse steht auf Adresse 4002H, das High-Byte auf Adresse 4003H. Vond den vorhanden 64K Bildspeicher werden im Debugger nur 16K unterstützt, das entspricht 8 Bildschirmseiten. Die Scroll-Adresse bewegt sich somit immer im Bereich 0000..3FFFH .

5.4 Cursor-Adresse

Die Cursor-Adresse ist die aktuelle Position des Cursors relativ zur Anfangsadresse des Bildspeichers (8000H). Eine Umpositionierung des Cursors ist jederzeit möglich, besonders auch unter Einbeziehung der Scroll-Adresse. Das Low-Byte der Cursor-Adresse steht auf Adresse 4004H, das High-Byte auf Adresse 4005H.

5.5 Eingabe Pufferzeiger

Der Eingabe Pufferzeiger zeigt auf die aktuelle Speicherzelle des Eingabe-Puffers (siehe 4.18, 4.19, 4.20). Das Low-Byte steht auf Adresse 4006H, das High-Byte auf Adresse 4007H.

5.6 Refresh Zeit-Konstante

Die Speicherzelle 4008H enthält den Wert für die Zeitspanne, während der die Refresh-Zyklen beim Memorytest unterdrückt werden (siehe 3.30).

5.7 Warteschleife für Memorytest

Nach jedem Schreibvorgang wird eine Subroutine mit CALL 4009H aufgerufen. Normalerweise steht dort ein RET (0C9H) und dahinter die Adresse einer Warteroutine von 3.3 Sekunden Verzögerungszeit bei einer Refresh Zeit Konstante von 5H. Schreibt man nun Jump (0C3H) auf die Speicherzelle 4009H, so wird diese Subroutine in den Memorytest eingebunden und ein Testdurchlauf benötigt von da an wesentlich mehr Zeit und man hat eine Kontrolle, ob der Speicher die eingeschriebenen Daten auch längere Zeit behalten kann.

5.8 NMI Service Rountine

Um die NMI-Service Routine (Non-Maskable-Interrupt) verändern zu können wurde auf der Speicherstelle 400CH eine Änderungsmöglichkeit eingerichtet. Bei der Software-Initialisierung steht hier ein Sprung auf eine Meldung (JP 007FH = Jump auf Adresse 7FH). Es kann jedoch auch auf eine andere Routine durch entsprechendes Ändern der Adressen 400DH (Low-Byte) und 400EH (High-Byte) gesprungen werden (siehe auch 4.21).

5.9 Warteschleife für Monitorausgabe

Mit CTRL-S (siehe 6.5) kann die Geschwindigkeit der Ausgabe auf dem Bildschirm gesteuert werden. Dies geschieht durch ein Ändern der Speicherzelle 400FH. Normalerweise steht dort 0C9H (RET). Nach Eingabe von CTRL-S wird ein 0C3H (JP ....) eingetragen, was bei der Ausgabe einen Sprung auf eine Zeitschleife zur Folge hat. Bei der nächsten Eingabe von CTRL-S wird wieder auf 0C9H (RET) zurückgeschaltet.

5.10 Cursor On-Off

Auf dieser Speicherzelle wird die jeweilige Programmierung des Cursors im Videocontroller eingetragen. Diese kann mit CTRL-T (siehe 6.5) geändert werden.

5.11 Invert Screen Flag

Auf dieser Speicherzelle wird die jeweilige Programmierung des Bildschirms eingetragen. Diese kann mit CTRL-R (siehe 6.5) geändert werden.

5.12 Serielle Ausgabe

Auf Adresse 4014H steht je nach Status der seriellen Ausgabe (siehe 3.20 bzw. 3.21) ein Sprung zur Ausgabe über die serielle Schnittstelle (C3 54 08 = JP 0854H) oder ein RET (0C9H).

5.13 ESCAPE-Routine

Auf Adresse 4017H steht ein Sprung zum Unterprogramm für die ESC-Funktion. Soll die ESC-Funktion (sofortiger Programmabbruch) unwirksam sein (wie bei allen Disk-Zugriffen), so wird auf dieser Adresse Return (0C9H) eingetragen. Ansonsten steht dort ein Sprungbefehl (0C3H).

Es gibt die Möglichkeit, das bei Eingabe von ESC angesprochene Unterprogramm durch Eintragen eine Adresse auf Speicherzeller 4018H (Low-Byte) und 4019H (High-Byte) zu maskifizieren. Es muß allerdings darauf geachtet werden, daß so bald als möglich ein RETI-Befehl gegeben wird, da das Escape-Unterprogramm eine Interrupt-Service-Routine ist.

6. Erläuterung zu PROM 2 (Floppy-Routine)

Das Prom 2 (Adresse 1000H-1FFFH) enthält alle notwendigen Unterprogramme zum Ansprechen von Disk-Laufwerken. Hierzu kurz einige Hinweise für diejenigen, die etwas tiefer einsteigen wollen.

6.1 Autostart

Bei Reset wird normalerweise automatisch das Betriebsystem geladen. Dieser sogenannte Autostart wird durch einen Sprung auf Adresse 1000H erreicht. Auf welchen Laufwerken der BOOT (Zwischenprogramm zum Starten der Betriebssoftware mit Namen BOOT2.SYS) gesucht werden soll, steht in einer 20H langen Tabelle ab Adresse 1010H. Zwei Angaben sind dazu erforderlich, nämlich die Codierung für das angesprochene Laufwerk und die Spur, die als Inhaltsverzeichnis (Directory) angesprochen werden soll. Das Ende der Tabelle muß mit zweimal OFFH markiert sein.

Folgende Bitzuordnung für die Laufwerke gilt:

B0 + B1	2 Bits für Drive-Nummer	(0-3)
B2	Single/Double-Density	(0/1)
B3	Single/Double-Head	(0/1)
B4	Mini/Std.-Drive	(0/1)
B5	Non-DMA/DMA	(0/1)
B6	Floppy-Disk/Hard-Disk	(0/1)
B7	Implementiert (NO/YES)	(0/1)

Dazu ein Beispiel (nach RA-Kommando siehe 3.24 möglich)

```
S 1010 84 04 85 04 B4 26 B5 26 C4 03 80 05 FF FF <CR>
```

Hier würde der Reihe nach gesucht auf:

```
Minifloppy double-density Drive 0 non DMA auf Spur 4
Minifloppy double-density Drive 1 non DMA auf Spur 4
Standardfloppy double-density Drive 0 DMA auf Spur 26
Standardfloppy double-density Drive 1 DMA auf Spur 26
Mini-Hard-Disk DMA auf Spur 3
Minifloppy single-density Drive 0 non DMA auf Spur 5
```

Hinweis: Die Option Standard-Hard-Disk ist aus Platzgründen nicht implementiert.
Bei Standard-Drives und Hard-Disk ist aus Geschwindigkeitsgründen immer DMA zu verwenden.

6.2 Arbeiten mit Vektor

Für direktes Lesen von der Floppy kann über den Einsprungpunkt 1006H gearbeitet werden. Dazu ist mit dem Register <IX> ein Vektor aufzubauen. Es gilt folgende Vereinbarung:

(IX + 00) = Drive-Identifikation nach Tabelle in 6.3.1 ohne Bit 3 und Bit 7
(IX + 01) = Kommando Byte (00 = Recalibrate 02 = Read Track;
03 = Read Sektor; 04 = Write Sektor;
05 = Write Track)
(IX + 02) = Track Byte (Angabe des zu bearbeitenden Tracks)
(IX + 03) = Sektor Byte (Angabe des zu bearbeitenden Sektors)
(IX + 04) = Reserviert
(IX + 05) = Fehler Byte (Byte für Rückmeldung eines aufgetretenen Fehlers)
(IX + 06) = Buffer-Adresse (Angabe der Buffer-Adresse Low-Byte)
(IX + 07) = Buffer-Adresse (Angabe der Buffer-Adresse High-Byte)
(IX + 08) = Anzahl Versuche (Angabe der Anzahl der Schreib-Lese-Versuche)
(IX + 09) = Fehler Zähler (Zähler für die Anzahl der aufgetretenen Fehler)
(IX + 10) = Reserviert
bis
(IX + 15) = Reserviert

Alle Kommandos werden auf einen Vektor dieser Art umgesetzt. Als Anzahl der Versuche gilt 5 als Voreinstellung.

6.3 Fehlermeldungen Floppy

Bei Floppy-Disk-Zugriffen sind folgende Fehlermeldungen implememententiert:

6.3.1 Format

Bei falschen Parametern wird, soweit in einem Programm dieser Kürze eine Erkennung möglich ist, die Meldung

* FORMAT *

ausgegeben. Intern hat dieser Fehler den Error-Code 81H.

6.3.2 DRIVE NOT READY

Alle nicht näher spezifizierten Fehler werden mit dieser Fehlermeldung augezeigt. Intern hat dieser Fehler den Error-Code 82H.

6.3.3 READ NOT POSSIBLE

Wenn der Disk-Controller die Sektormarkierungen nicht lesen kann, wird diese Fehlermeldung ausgegeben. Intern hat dieser Fehler den Error-Code 83H.

6.3.4 CRC-ERROR

Auf jedem Sektor ist ein Schutz gegen Fehler in Form von einer Prüfsumme eingetragen. Wenn diese Prüfsumme nicht mit dem Inhalt des Sektors übereinstimmt, so wird eine Fehlermeldung ausgegeben. Intern hat dieser Fehler den Error-Code 84H.

6.3.5 DISK WRITE PROTECTED

Wird bei einem schreibenden Zugriff auf die Diskette festgestellt, daß diese durch Aufkleben des Schreibschutzes geschützt ist, so wird diese Fehlermeldung ausgegeben. Intern hat dieser Fehler den Error-Code 85H.

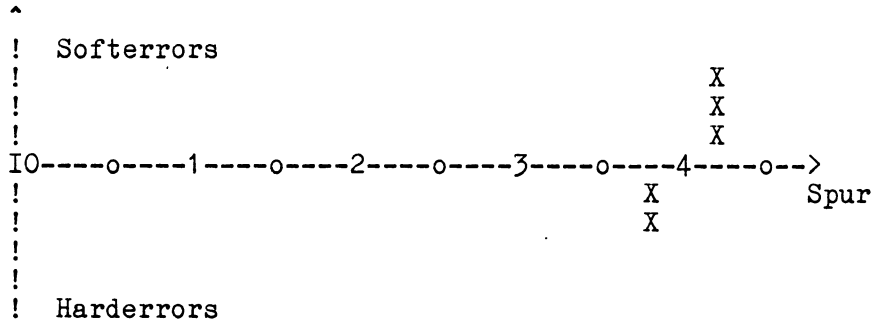
6.4 DISK-TEST

PROM 2 enthält auf Adresse 1800H ein Floppy-Disk-Testprogramm. Dieses Programm ist interaktiv und kann mit "J 1800 <CR>" angesprungen werden.
Der Test führt nur Lesezugriffe aus, er kann also mit jeder Scheibe durchgeführt werden. Der Ablauf sieht in etwa folgendermaßen aus :

Eingabe im Testdebugger : J 1800 <CR>

```
DISKTEST VERSION 1.51 VOM 30.11.1982  (derzeit aktuelle Version)
DRIVE ID? 1 <CR>
VERSUCHE: 5 <CR>     (Eingabe dezimal, 2-stellig)
LOOPS : 2 <CR>       (Eingabe hexadezimal, 2-stellig)
MODE 1/2 : 2 <CR>
```

Es erfolgt nun ein Test in Laufwerk 0, 5" Mini, double density, kein DMA. Dabei wird Spur für Spur gelesen, jeweils maximal 5 Versuche, dabei auftretende Hard- oder Soft-Errors werden aufgelistet. Der Test wird in dem oben angegebenen Beispiel zweimal durchlaufen.
Bei MODE 2 erfolgt die Darstellung der Fehler in einem sehr anschaulichen Histogramm das in etwa folgendes Aussehen hat :



Bei dem Test traten somit 2 Harderrors in Spur 38 und
3 Softerrors in Spur 42 auf.

Über einen sogenannten SCALE FACTOR wird der Wert eines Tabelleneintrags (X) variiert :

```
SCALE FACTOR 0010  :  X bedeutet : 1  Hard- oder Softerror
SCALE FACTOR 0020  :  X bedeutet : 2  Hard- oder Softerrors
SCALE FACTOR 0040  :  X bedeutet : 4  Hard- oder Softerrors
usw.
```

Bei MODE 1 erfolgt eine Fehlermeldung im folgenden Format :

```
PROBLEM AT TRACK XX WITH ......
```

Der Test kann mit W angehalten und mit jeder anderen Taste abgebrochen werden.

Wird bei VERSUCHE : nur <CR> eingegeben so werden 99 Versuche gemacht, wird bei LOOPS : nur <CR> eingegeben so werden 9999 Durchläufe gemacht.

7. Tabellen

7.1 KOMMANDOS

	Befehle	Erklärung	Anzahl der Eingaben	Format		
1)	A	Again	0			
2)	CP	Compare	3	Anf.-Adr.1	Anf.-Adr.2	Länge
3)	D	Display and alter	1	Adresse		
		Display Memory	2	Adresse	-Adr./Länge	
4)	DO	Kommandoausführung	1	Adresse		
5)	DP	Display Port	1	Port-Adresse		
6)	EG	Go by Error	0			
7)	ES	Stop by Error	0			
8)	F	Fill	3	Anf.-Adr.	-Adr./Länge	Wert
9)	FC	Floppy Controlmode	2	Disk-ID	Track	
10)	G	Go	1	Adresse		
11)	I	In-Loop Port	1	Port Adresse		
12)	J	identisch mit GO	1	Adresse		
13)	K	Betriebssystem laden	0			
14)	L	Locate	optional	Anf.-Adr.	-Adr./Länge	Wert.
15)	LL	Local-Mode	0			
16)	MO	Move	3	Quell-Adr.	Ziel-Adr.	Länge
17)	MA	Mapper Programmierung	1	Bank Nr.		
18a)	MT	Memory-Test (in 1 Bank)	2	Anf.-Adr.	-Adr./Länge	
		Memory-Test	3	Anf.-Adr.	-Adr./Länge	Anzahl
18b)	MX	Memory-Test (alle Bänke)	2	Anf.-Adr.	-Adr./Länge	
		Memory-Test	3	Anf.-Adr.	-Adr./Länge	Anzahl
19)	O	Output-Loop Port	2	Port-Adr.	Wert	
20)	OF	Serielle Schnittstelle ausschalten	0			
21)	ON	Serielle Schnittstelle einschalten (9600 Baud)	0			
22)	OV	Overlay einschalten (nur bei spezieller Hardware möglich)	0			
23)	P	Pause	0			
		Pause	1	Schleifenzähler		
24)	RA	RAM-Umschaltung	0			
25)	RD	Read-Loop Memory	1	Adresse		
26)	RC	Recalibrate Disk	1	Disk-ID		
27)	RS	Read Sector	0	alte Werte		
		Read Sector	3	Disk-ID	Track/Sector	Adr.
28)	RT	Read Track	0	alte Werte		
			3	Disk-ID	Track/Sector	Adr.
29)	S	Set Memory	optional	Adresse	Wert Wert....	
30)	SP	Set Port	optional	Port-Adr.	Wert Wert....	
31)	ST	Zeit-Konstante anzeigen	0			
		Zeit-Konstante setzen	1	Wert		
32)	TP	Testprogramm	1	Disk-ID		
33)	W	Write-Loop Memory	2	Adresse	Wert	
34)	WR	Write-Read-Loop Memory	2	Adresse	Wert	
35)	WS	Write Sector	0	alte Werte		
		Write Sector	3	Disk-ID	Track/Sector	Adr.
36)	WT	Write Track	0	alte Werte		
			3	Disk-ID	Track/Sector	Adr.

7.2 DRIVE-ID TABELLE

Drive-ID	Bedeutung	Laufwerk	
1	5" Mini	0	double density ohne DMA
2	5" Mini	1	double density ohne DMA
3	5" Mini	0	single density ohne DMA
4	5" Mini	1	single density ohne DMA
5	8" Standard	0	double density mit DMA
6	8" Standard	1	double density mit DMA
7	8" Standard	0	single density ohne DMA
8	8" Standard	1	single density ohne DMA
9	5" Winchester		double density mit DMA
A	5" Mini	0	double density mit DMA
B	5" Mini	1	double density mit DMA
C	5" Mini	0	single density mit DMA
D	5" Mini	1	single density mit DMA
E	8" Standard	0	single density mit DMA
F	8" Standard	1	single density mit DMA

Bei double-sided Laufwerken kann durch eine führende "1" bei der Drive-ID die Seite 2 angesprochen werden.

Beispiel: "16" bedeutet Seite 2 des linken 8" Standard Laufwerks, double density mit DMA.

Im Allgemeinen gilt die Zuordnung : Laufwerk 0 = rechts oder oben
Laufwerk 1 = links oder unten

7.3 Steuerzeichen

Taste:	ASCII-CODE:	Funktion:
CTRL-A	01H	Cursor left down
CTRL-F	06H	Cursor forward
CTRL-G	07H	Bell
CTRL-H	08H	Backspace
CTRL-I	09H	Tabulator
CTRL-J	0AH	Line feed
CTRL-L	0CH	Form feed
CTRL-Q	11H	Character invert
CTRL-R	12H	Invert screen
CTRL-S	13H	Invert speed
CTRL-T	14H	Cursor off-on
CTRL-W	17H	Blinking on/off
CTRL-Z	1AH	Cursor up
RETURN	0DH	Carriage return
RUBOUT	7FH	Clear input buffer
HOME	1CH	Cursor left top

Manche dieser Funktionen werden bei der Eingabe über Eingabe-Puffer unterdrückt (z.B. bei Kommandoeingaben).

Testsoftware für KDT6, TCB/Z80 und TCB/IOV

Diese Unterlage beinhaltet ein Assemblerquellprogramm, das eine zuverlässige Überprüfung des Hauptvideospeichers auf der KDT6 sowie der TCB/IOV ermöglicht.

Rel 1.0 Januar 1984

Videomemorytest für KDT6 und TCB/IOV

```
VMT6:

;VIDEOMEMORYTEST FÜR KDT6 REV 1.1. UND 1.2
;-----------------------------------------

;P.NAME:  VMT6
;AUTHOR:  Karl-Heinz Bauer
;DATE:            22.06.1982
;VERSION: 2.1
;LAST MOD.:       25.10.1982 by K.H.Bauer
;LAST MOD.:       24.01.1983 by K.H.Bauer


TITLE VMT6
PAGE 65


;Dieses Programm testet den Hauptvideospeicher
;der KDT6 sowie der TCB IOV.
;Es werden folgende Kombinationen eingeschrieben
;und überprüft:
;01, 02, 04, 08, 10, 20, 40, 80,
;FE, FD, FB, F7, EF, DF, BF, 7F, 00, FF,
;Ferner wird das LOW Adressbyte auf die jeweilige
;Adresse geschrieben und überprüft. Danach das HIGH
;Adressbyte.
;Im letzten Test wird alles auf 00 gesetzt und nacheinander
;nur eine der 16 Adressleitungen auf HIGH gesetzt, und die so
;adressierte Speicherzelle mit FF beschrieben und abgeprüft.
;Anschließend werden die Ergebnisse aller Tests am Bildschirm
;ausgegeben. Die Ausgabe erfolgt über die KOS Funktionen
;OUTPUT (86H) und STRING (87H).

;Zerstörte Register:-----> KEINE
;Zerstörter RAMinhalt:---> VIDEO RAM Inhalt wird zerstört:

;ERLIST:
;Zur Auswertung durch dieses oder eines anderen Programmes
;wird eine ERROR-LISTE (ERLIST) angelegt, mit je 5 BYTES für
;jeden Test:      1. BYTE ----> 00 = KEIN ERROR  FF = ERROR
;         2. BYTE ----> HIGH BYTE der ERROR ADRESSE
;         3. BYTE ----> LOW BYTE der ERROR ADRESSE
;         4. BYTE ----> Soll BYTE
;         5. BYTE ----> Ist BYTE
;Es sind 21 Tests implementiert.


;GLOBALS
;-------

  GLOBAL  VMT6
  GLOBAL  ERLIST
```

```
  JR START

STARTM:
  DEFB OCH
  DEFM "VIDEOMEMORYTEST FUER KDT6/ VERSION 2.1 VOM 24.01.83 VON"
  DEFM " K.H.BAUER"
  DEFW OAODH
  DEFB OOH

;ES WIRD NUR DER HAUPTVIDEOSPEICHER GETESTET

START:
  PUSH AF
  PUSH BC
  PUSH DE
  PUSH HL
  LD HL, STARTM
  CALL RST18H
  LD B,O          ;TEST COUNTER
  CALL CLEARERL   ;CLEAR ERRORLIST
  CALL BYTETEST   ;
  CALL LADRTEST   ;LOW ADDRESS TEST
  CALL HADRTEST   ;HIGH ADDRESS TEST
  CALL TEST17     ;ADDRESSBIT CHECK
  CALL ANZEIGE    ;AUSWERTEN UND ANZEIGEN
  POP HL          ;EXIT
  POP DE
  POP BC
  POP AF
  RET

RST18H:                   ;(HL)=TEXT-->MONI.
  LD IX, VECTOR
  LD (IX+1), STRING
  RST 8H
  RET


RST2OH:                   ;HEX->ASCII->OUT
  LD IX, VECTOR
  LD (IX+1), ACCOUT
  RST 8H
  RET


VMADR:            ;HL = VIDEOADRESSE
  PUSH AF
  LD A,H
  OUT (VAL.HIGH),A
  LD A,L
  OUT (VAL.LOW),A
  POP AF
  RET

MVFILL:           ;HL = MVADDRESS
  CALL VMADR
  PUSH BC
OUTLOOP:  ;DE = BLOCKLÄNGE
          ;A  = BYTE
          ;C  = PORT
  OUT (C),A
  DEC DE
  LD B,A
  LD A,D
  OR E
```

```
  LD A,B
  JR NZ,OUTLOOP
  POP BC
  RET

CLEARERL:
  PUSH HL
  PUSH AF
  PUSH BC
  LD HL,ERLIST
  LD B,ERLISTL
  XOR A
CLLOOP:
  LD (HL),A
  INC HL
  DJNZ CLLOOP
  POP BC
  POP AF
  POP HL
  RET


BYTETEST:
  LD HL,BYTELIST
  DEC B
BYTELOOP:
  INC B
  LD A,(HL)
  PUSH HL
  LD DE,0
  LD HL,0
  LD C,VMBO.AUTO.INC
  CALL MVFILL
  LD DE,0
  LD HL,0
  CALL BYTECHECK
  POP HL
  OR (HL)
  INC HL
  JR NZ,BYTELOOP
  RET


BYTECHECK:          ;HL = MVADDRESS
  CALL VMADR                ;LOAD STARTADDRESS
  PUSH BC
CHECKLOOP:
           ;DE = BLOCKLÄNGE
           ;A  = BYTE
           ;C  = PORT
           ;STACK TOP=TESTCOUNTER
  IN B,(C)
  CP B
  JR NZ,ERROR
  INC HL
  DEC DE
  LD B,A
  LD A,D
  OR E
  LD A,B
  JR NZ,CHECKLOOP
  POP BC
  RET
```

```
LADRTEST:           ;ADDRESSTEST
  LD HL,OH
  LD C,VMBO         ;VIDEO MEM. BANK
  INC B
  PUSH BC           ;SAVE TESTCOUNTER
LADRL:
  CALL VMADR        ;LOAD VMADDRESS
  OUT (C),L
  INC HL
  LD A,H
  OR L
  JR NZ,LADRL
LADRCHECK:
  CALL VMADR        ;LOAD HL TO THE
                    ;VIDEO MEM.ADR.LATCH
  LD A,L
  IN B,(C)
  CP B
  JR NZ,ERROR
  INC HL
  LD A,L
  OR H
  JR NZ,LADRCHECK
  POP BC
  RET


HADRTEST:
  INC B
  LD HL,O
  LD C,VMBO.AUTO.INC
  XOR A
HADRL:
  LD DE,0100H
  CALL MVFILL
  INC H
  LD A,H
  CP O
  JR NZ,HADRL
HADRCHECK:
  LD DE,0100H
  CALL BYTECHECK
  JR NZ,ERRET
  INC H
  LD A,H
  CP O
  JR NZ,HADRCHECK
ERRET:
  RET
```

```
ERROR:              ;HL = ERRORADDRESS
                    ;A  = BYTE SOLL
                    ;STACK TOP = TESTLOOP
  POP BC            ;TESTLOOP COUNTER
  PUSH DE
  PUSH AF
  PUSH HL
  LD HL,ERLIST      ;ERRORLISTE
  LD A,B
  RLCA
  RLCA
  ADD A,B           ;MULTP. MIT 5
  LD D,O
  LD E,A
  ADD HL,DE
  LD A,OFFH
  LD (HL),A
  INC HL
  POP DE            ;ERROR ADRESS
  LD (HL),D
  INC HL
  LD (HL),E
  INC HL
  POP AF
  LD (HL),A         ;SOLL BYTE
  PUSH AF
  INC HL
  EX DE,HL
  CALL VMADR
  IN A,(C)
  EX DE,HL
  LD (HL),A         ;IST BYTE
  POP AF
  LD D,A
  LD A,2H
  INC A             ;RESET Z-FLAG
  LD A,D
  POP DE
  RET
```

```
ANZEIGE:  ;
  PUSH AF
  PUSH BC
  PUSH HL
  LD B,1
  LD HL,STARTM
  CALL RST18H
  LD HL,TEXT
  CALL RST18H
  LD HL,ERLIST
ANLOOP:
  PUSH HL
  LD HL,TESTM      ;CR + TEST
  CALL RST18H
  LD A,B
  CALL RST20H      ;TEST NR
  POP HL           ;ZEIGER AUF ERLIST
  LD A,(HL)
  CP 0
  INC HL
  JR Z,NOER
  PUSH HL
  LD HL,NOKMSG
  CALL RST18H
  POP HL
  LD A,(HL)        ;HIGH ADDRESS
  CALL RST20H      ;HEX->ASCII->VIDEO
  INC HL
  LD A,(HL)        ;LOW ADDRESS
  CALL RST20H
  INC HL
  PUSH HL
  LD HL,BLANK6
  CALL RST18H
  POP HL
  LD A,(HL)        ;SOLL BYTE
  CALL RST20H
  INC HL
  PUSH HL
  LD HL,BLANK4
  CALL RST18H
  POP HL
  LD A,(HL)
  CALL RST20H
NLOOP:
  INC HL           ;5. INC = NEXT TEST
  LD A,B
  INC B
  CP TESTANZAHL
  JR NZ,ANLOOP
  POP HL
  POP BC
  POP AF
  RET


NOER:
  PUSH HL
  LD HL,OKMSG
  CALL RST18H
  POP HL
  INC HL
  INC HL
  INC HL
  JR NLOOP
```

```
TEST17:           ;SETZT ALLE 16 ADRESSLEITUNGEN
          ;NACHEINANDER AUF HIGH, BESCHREIBT
          ;DIE SO ADRESSIERTE SPEICHERZELLE UND
          ;ÜBERPRÜFT DIES
  INC B           ;SET TESTCOUNTER
  LD C,VMBO.AUTO.INC
  LD HL,01H       ;0000 0000 0000 0001B
  LD A,0FFH
T17LLOOP:         ;ADRESSBIT 0 BIS 7
  CALL CLEARVM    ;00 IN VIDEOMEMORY
  CALL VMADR      ;LOAD VIDEO MEM. ADDRESS
  OUT (C),A       ;LOAD FF TO ADDRESS
  CALL T17CHECK   ;ERROR ?
  RET NZ          ;NOT ZERO = ERROR
  RL L            ;NEUE ADRESSE
  JR NC,T17LLOOP  ;ENDE DES LOW LOOPS ?
  LD HL,0100H     ;LOAD H WITH 0000 0001 B
T17HLOOP:         ;ADRESSBIT 8 BIS 15
  CALL CLEARVM
  CALL VMADR
  OUT (C),A
  CALL T17CHECK
  RET NZ          ;ERROR!
  RL H
  JR NC,T17HLOOP  ;ENDE ?
  RET


T17CHECK:
  LD D,H          ;RICHTIGE ADRESSE
  LD E,L
  LD HL,0         ;CHECK STARTADDRESS
T17CHLOP:
  CALL VMADR      ;LOAD VIDEO MEM. ADR.
  IN A,(C)        ;READ BYTE.
  CP 0            ;
  PUSH BC         ;SAVE TESTCOUNTER
  JR NZ,T17BYTE
ADROK:            ;BYTE IS OK
  POP BC          ;TESTCOUNTER
  INC HL          ;ADDRESS + 1
  LD A,H
  OR L            ;BLOCK ENDE?
  LD A,0FFH
  JR NZ,T17CHLOP
  LD H,D          ;RICHTIGE ADR. ZURÜCK
  LD L,E
  RET
```

```
T17BYTE:  ;HL=AKTUELLE CHECKADRESSE
          ;A =BYTE (FF ?)
  CP OFFH          ;
  JP NZ,ERROR      ;NOT FF
  LD A,E           ;LOW ADDRESS
  CP L
  JR NZ,T17FEHLER
  LD A,D           ;HIGH ADDRESS BYTE
  CP H
  JR Z,ADROK
T17FEHLER:
  LD A,E           ;RIGHT LOW ADDRESS
  OR L             ;ERROR ADR.+RIGHT ADR.BIT
  LD L,A           ;LOW ADDRESS
  LD A,D           ;RIGHT HIGH ADDRESS
  OR H
  LD H,A           ;ERROR HIGH ADDRESS
  JP ERROR


CLEARVM:  ;C = PORT AUTO INC
  PUSH AF
  PUSH DE
  PUSH HL
  LD HL,0          ;STARTADR. 0
  LD DE,0          ;BLOCKLÄNGE
  XOR A
  CALL MVFILL      ;FILL VIDEOM.
  POP HL
  POP DE
  POP AF
  RET
```

```
NOKMSG:
  DEFM "    "
  DEFB 07H          ;BELL
  DEFB 011H
  DEFM "NOT OK"
  DEFB 011H
  DEFM "    "
  DEFB  00

OKMSG:
  DEFM "    OK"
  DEFB  00


TEXT:
  DEFM "            "        ;10 BLANKS
  DEFM "OK?       ADRESSE    SOLL  IST"
  DEFB 00H

TESTM:
  DEFW 0A0DH
  DEFM "TEST "
  DEFB 00H


BLANK6:
  DEFM "  "
BLANK4:
  DEFM "    "
  DEFB 00H


BYTELIST:
  DEFB  001H
  DEFB  002H
  DEFB  004H
  DEFB  008H
  DEFB  010H
  DEFB  020H
  DEFB  040H
  DEFB  080H
  DEFB  0FEH
  DEFB  0FDH
  DEFB  0FBH
  DEFB  0F7H
  DEFB  0EFH
  DEFB  0DFH
  DEFB  0BFH
  DEFB  07FH
  DEFB  0FFH
  DEFB  000H        ;00 = LETZTES BYTE
BLISTL    EQU  $-BYTELIST


  TESTANZAHL  EQU  021D


;READ - WRITE MEMORY
;===================

ERLIST:
  DEFS  TESTANZAHL*5
ERLISTL   EQU  $-ERLIST
```

```
VECTOR:
  DEFW  0
  DEFW  0
  DEFW  0
  DEFW  0

;EQUATES

  VMBO              EQU  30H
  VMBO.AUTO.DEC     EQU  36H
  VMBO.AUTO.INC     EQU  31H
  VAL.HIGH          EQU  40H
  VAL.LOW           EQU  41H
  STRING            EQU  87H
  OUTPUT            EQU  86H
  ACCOUT            EQU  88H

END VMT6
```