CPC464


Schneider
Das komplette CPC464
Betriebssystem
Firmware Handbuch
SOFT 258

CPC464 Firmware

ROM Routinen und Erläuterungen

Bruce Godden, Locomotive Software

CAS CATALOG CAS CHECK CAS IN ABANDON CAS IN CHAR CAS IN CLOSE CAS IN DIRECT CAS IN OPEN CAS INITIALISE CAS NOISY CAS OUT ABANDON CAS OUT CHAR CAS OUT CLOSE CAS READ CAS RESTORE MOTOR CAS RETURN CAS SET SPEED CAS START MOTOR CAS STOP MOTOR CAS TEST EOF CAS WRITE GRA ASK CURSOR GRA CLEAR WINDOW GRA GET ORIGIN GRA GET PAPER GRA GET PEN GRA GET W HEIGHT GRA GET W WIDTH GRA INITIALISE GRA LINE ABSOLUTE GRA LINE RELATIVE GRA MOVE ABSOLUTE GRA MOVE RELATIVE GRA PLOT ABSOLUTE GRA PLOT RELATIVE GRA RESET GRA SET ORIGIN GRA SET PAPER GRA SET PEN GRA TEST ABSOLUTE GRA TEST RELATIVE GRA WIN HEIGHT GRA WIN WIDTH GRA WR CHAR HI KL CURR SELECTION HI KL L ROM DISABLE HI KL L ROM ENBLE HI KL LDDR HI KL LDIR HI KL POLL SYNCHRONOUS HI KL PROBE ROM HI KL ROM DESELECT HI KL ROM RESTORE HI KL ROM SELECT HI KL U ROM DISABLE HI KL U ROM ENABLE IND GRA LINE IND GRA PLOT IND GRA TEST IND KM TEST BREAK IND MC WAIT PRINTER IND SCR READ IND SCR WRITE IND TXT DRAW CURSOR IND TXT OUT ACTION IND TXT UNDRAW CURSOR IND TXT UNWRITE IND TXT WRITE CHAR KL ADD FAST TICKER KL ADD FRAME FLY KL ADD TICKER KL CHOKE OFF KL DEL FAST TICKER KL DEL FRAME FLY KL DEL SYNCHRONOUS KL DEL TICKER KL DISARM EVENT KL DO SYNC KL DONE SYNC KL EVENT DISABLE KL EVENT ENABLE KL FIND COMMAND KL INIT BACK KL INIT EVENT KL LOG EXT KL NEW FAST TICKER KL NEW FRAME FLY KL NEXT SYNC KL ROM WALK KL SYNCH RESET KL TIME PLEASE KL TIME SET KM ARM BREAKS KM BREAK EVENT KM CHAR RETURN KM DISARM BREAK KM EXP BUFFER KM GET CONTROL KM GET DELAY KM GET EXPAND KM GET JOYSTICK KM GET REPEAT KM GET SHIFT KM GET STATE KM GET TRANSLATE KM INITIALISE KM READ CHAR KM READ KEY KM RESET KM SET CONTROL KM SET DELAY KM SET EXPAND KM SET REPEAT KM SET SHIFT KM SET TRANSLATE KM TEST KEY KM WAIT CHAR KM WAIT KEY LOW EXT INTERRUPT LOW FAR CALL LOW FRM JUMP LOW INTERRUPT ENTRY LOW KL FAR ICALL LOW KL FAR PCHL LOW KL LOW PCHL LOW KL SIDE PCHL LOW LOW JUMP LOW PCDE INSTRUCTION LOW PCHL INSTRUCTION LOW RAM LAM LOW RESET ENTRY LOW SIDE CALL LOW USER RESTART MC BOOT PROGRAM MC BUSY PRINTER MC CLEAR INKS MC JUMP RESTORE MC PRINT CHAR MC RESET PRINTER MC SCREEN OFFSET MC SEND PRINTER MC SET INKS MC SET MODE MC SOUND REGISTER MC START PROGRAM MC WAIT FLYBACK SCR ACCESS SCR CHAR INVERT SCR CHAR LIMITS SCR CHAR POSITION SCR CLEAR SCR DOT POSITION SCR FILL BOX SCR FLOOD BOX SCR GET BORDER SCR GET FLASHING SCR GET INK SCR GET LOCATION SCR HORIZONTAL SCR INITIALISE SCR INK DECODE SCR INK ENCODE SCR NEXT BYTE SCR NEXT LINE SCR PIXELS SCR PREV BYTE SCR PREV LINE SCR REPACK SCR RESET SCR SET BASE SCR SET BORDER SCR SET FLASHING SCR SET INK SCR SET MODE SCR SET OFFSET SCR SW ROLL SCR UNPACK SCR VERTICAL SOUND A ADDRESS SOUND AMPL ENVELOPE SOUND ARM EVENT SOUND CHECK SOUND CONTINUE SOUND HOLD SOUND QUEUE SOUND RELEASE SOUND RESET SOUND T ADDRESS SOUND TONE ENVELOPE TXT CLEAR WINDOW TXT CUR DISABLE TXT CUR ENABLE TXT CUR OFF TXT CUR ON TXT GET BACK TXT GET CONTROLS TXT GET CURSOR TXT GET M TABLE TXT GET MATRIX TXT GET PAPER TXT GET PEN TXT GET WINDOW TXT INITIALISE TXT INVERSE TXT OUTPUT TXT PLACE CURSOR TXT RD CHAR TXT REMOVE CURSOR TXT RESET TXT SET BACK TXT SET COLUMN TXT SET CURSOR TXT SET GRAPHIC TXT SET M TABLE TXT SET MATRIX TXT SET PAPER TXT SET PEN TXT SET ROW TXT STR SELECT TXT SWAP STREAMS TXT VALIDATE TXT VDU DISABLE TXT VDU ENABLE TXT WIN ENABLE TXT WR CHAR

Schneider Computer Division
Silvastraße 1
8939 Türkheim 1

SW 158, erste Ausgabe 1984
Herausgegeben von der Schneider Computer Division
Originalausgabe in Englisch

CPC464 Firmware

Vorwort

Das im ROM integrierte Betriebssystem des Schneider CPC464 setzt sich aus dem BASIC-Interpreter und der Firmware zusammen. Die Firmware, die in diesem Buch behandelt wird, besteht aus einer Reihe niederer Routinen, die zuständig sind für Hardware-Steuerung, Bildschirm-Behandlung und Echtzeit-Ablauf.

Diese Firmware-Routinen dienen nicht nur dem Assembler-Programmierer – die meisten anspruchsvollen Programme sind in der Maschinensprache geschrieben –, sondern sind auch leicht in Programmen mit BASIC oder anderen benutzerfreundlichen Programmiersprachen anzuwenden. Somit konnte Schneider sein Ziel verwirklichen einen schnellen hochentwickelten Computer mit niedrigen Hardwarekosten herzustellen.

Dieses Buch gibt Ihnen einen sorgfältigen zuverlässigen Einblick in den Aufbau des Rechners. Es wird nicht nur erklärt, was die Firmware-Routinen bewirken, sondern auch wie sie arbeiten, warum sie so ablaufen und wie man diese Fähigkeiten ausnutzen kann. Schneider ist zu Recht stolz auf dieses Produkt, und wünscht Ihnen viel Erfolg mit diesen Manual.

Inhalt

5 Graphik-VDU

6 Bildschirm-Paket

7 Tongenerator-Verwaltung

8 Kassetten-Verwaltung

1 Die Firmware

Dieses Handbuch beschreibt die Firmware des Schneider CPC464 Mikrocomputer. Es beschreibt nicht die vollständige Programmiersprache BASIC dieses Systems, sondern nur diejenigen Bestandteile, die andere Programme betreffen. Bei der Beschreibung einiger Firmware-Merkmale wird BASIC als Beispielprogramm angeführt.

Die Firmware befindet sich im niederwertigen ROM (siehe Abschnitt 2). Ihre Aufgabe besteht in der Steuerung der Hardware des Computers, sowie in der Bereitstellung von nützlichen Hilfsprogrammen, die von anderen Programmen verwendet werden können. Damit entfällt die Notwendigkeit, in jedem Programm die Hilfsprogramme selbst erstellen zu müssen. Dieses Handbuch ist für denjenigen interessant, der die Arbeitsweise des Systems kennenlernen möchte. Unentbehrlich ist es für den Ersteller von Programmen in Maschinensprache, insbesondere für System- (z. B. andere Programmiersprachen) und Spiel-Programme.

Die in diesem Handbuch gegebene Information ist teilweise sehr detailliert. Sie umfaßt die Arbeitsweise der Firmware von der niedrigsten Ebene (z. B. die Ansteuerung des Tongenerator-Chip) bis zur höchsten Ebene (z. B. der Ablauf einer Ton-Sequenz). Für die Anwendung der Firmware ist es nicht erforderlich, die gesamte hier gegebene Information zu verstehen. Ein allgemeines Verständnis der Arbeitsweise des Systems ist für den Programmierer jedoch unerläßlich, um für eine bestimmte Aufgabe die optimale Methode auswählen zu können.

1.1 Die Hardware

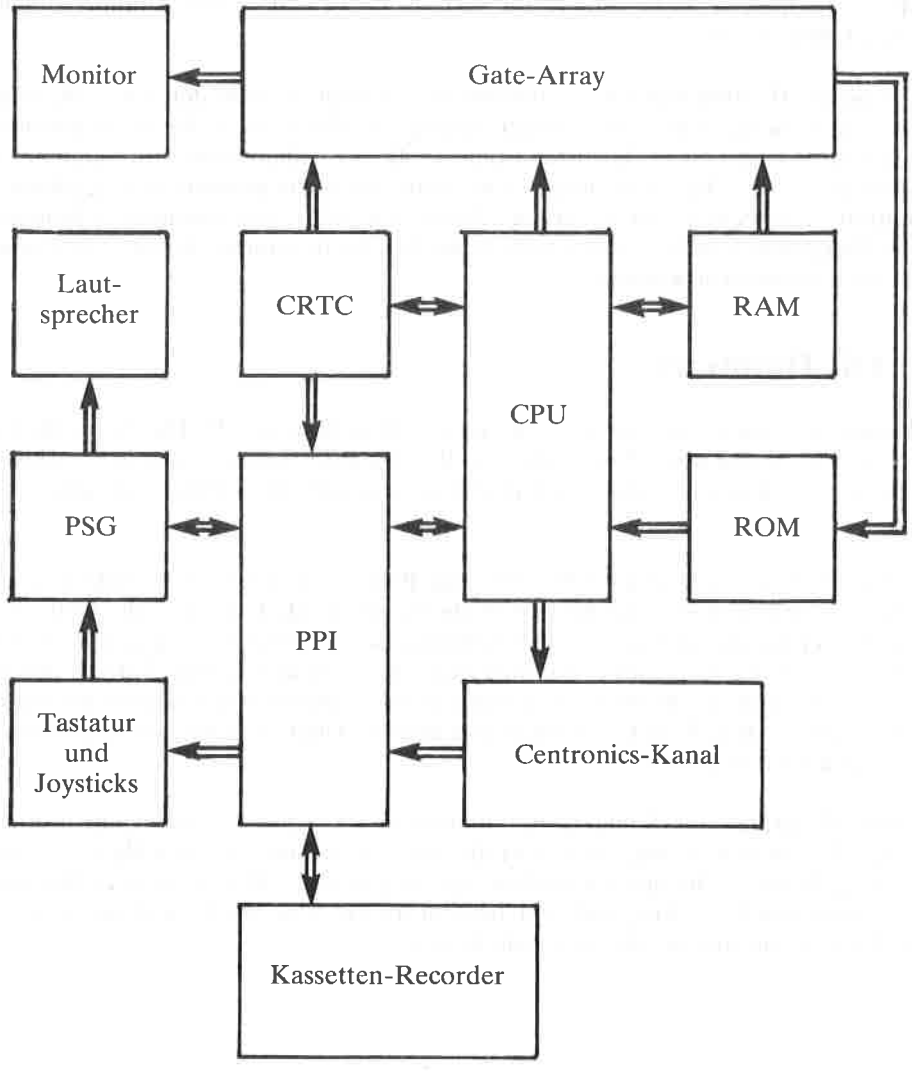
Das Diagramm der folgenden Seite gibt einen Überblick über die Hardware-Bestandteile des Systems und deren Verbindungen. Weitere Informationen zur Arbeitweise der Hardware finden sich in Anhang XII und den entsprechenden Datenblättern des Herstellers.

Das System basiert auf einer CPU (**C**entral **P**rocessing **U**nit), dem Mikroprozessor Z80A, mit einer Taktrate von 4 MHz. Von ebenso großer Bedeutung ist das Gate-Array. Es beinhaltet verschiedenartige Logik-Schaltkreise zur Steuerung einiger System-Bestandteile. Im Einzelnen sind es die Steuerung der Tintenfarbe (Ink-Farbe), des Bildschirm-Modus und der ROM-Freischaltung (siehe Abschnitt 7). Gemeinsam mit dem CRTC (**C**athode **R**ay **T**ube **C**ontroller), einem 6845-Chip, erzeugt das Gate-Array das Videosignal für den Monitor.

Der PSG (**P**rogrammable **S**ound **G**enerator) ist ein AY-3-8912. Dieser Chip besitzt drei Ton-Kanäle, einen Rauschgenerator, Hüllkurven-Steuerung für jeden Kanal und einen Ein-/Ausgabekanal. Die Inbetriebnahme der Tongenerator-Hardware ist in Abschnitt 7 beschrieben. Der Ein-/Ausgabekanal arbeitet im Eingabe-Modus und tastet den Zustand der Tastatur und des Joystick-Schalters ab.

Das PPI (**P**arallel **P**eripheral **I**nterface), ein 8255, steuert den Rest des Systems. Es besitzt drei Kanäle. Kanal C wird als Ausgabekanal zur Steuerung des Kassettenrecorder-Motors, zum Schreiben von Daten auf Kassette, zum Übernehmen von Daten aus oder Übergeben von Daten in den PSG und zur Auswahl der Tastatur-Reihen verwendet. Kanal B wird als Eingabe-Kanal geschaltet und überwacht das Bildrücklauf-Signal, das Busy-Signal des Centronics-Kanals und verschiedene optionelle Verbindungen, sowie das Lesen der Daten von der Kassette. Kanal A dient der Kommunikation mit dem PSG und ist wahlweise in den Eingabe- oder Ausgabe-Modus geschaltet.

Speicherzugriffe sind mit der Video-Logik im Mikrosekunden-Raster synchronisiert. Auf diese Weise wird jeder Z80-Maschinenzyklus auf ein Vielfaches von 4 Taktzyklen gestreckt. Praktisch ändert sich damit die Befehlsausführungszeit so, als wäre die effektive Taktrate annähernd 3,3 MHz.



1.2 Einteilung der Firmware

Die Firmware ist in „Pakete" unterteilt, wovon jedes mit einem bestimmten Systemabschnitt, gewöhnlich einem Hardware-Gerät, befaßt ist. Jedem Paket ist ein Abschnitt dieses Handbuches zugeordnet, in dem die Arbeitsweise im Detail geschildert wird. Die Systemabschnitte und ihre zugehörigen Pakete sind:

Tastatur:	Tastatur-Verwaltung
Bildschirm:	Text-VDU, Graphik-VDU, Bildschirm-Paket
Kassette:	Kassetten-Verwaltung
Tongenerator:	Tongenerator-Verwaltung
Betriebssystem:	Kern, Maschinen-Paket, Sprungtabelle

a) Tastatur-Verwaltung

Die Tastatur-Verwaltung ist in Abschnitt 3 beschrieben. Ihre Aufgabe ist die Abfrage der Tastatur, der Funktionstasten und des Joysticks, sowie das Testen auf Abbruchbedingungen und das Erzeugen von Zeichen.

b) Text-VDU

Der Text-VDU ist in Abschnitt 4 beschrieben. Seine Aufgabe ist das Senden von Zeichen an den Bildschirm einschließlich Cursor und die Umsetzung von Steuer-Codes.

c) Graphik-VDU

Der Graphik-VDU ist in Abschnitt 5 beschrieben. Seine Aufgaben sind das Darstellen von Bildpunkten, das Testen von Bildpunkten und das Zeichnen von Linien auf dem Bildschirm.

d) Bildschirm-Paket

Das Bildschirm-Paket ist in Abschnitt 6 beschrieben. Es bildet die Schnittstelle zwischen Text- und Graphik-VDU einerseits und der Bildschirm-Hardware andererseits. Das Bildschirm-Paket ist mit den Aspekten des Bildschirms befaßt, die mit beiden Paketen zusammenhängen, wie z. B. Bildschirm-Modus oder Ink-Farben.

e) Tongenerator-Verwaltung

Die Tongenerator-Verwaltung ist in Abschnitt 7 beschrieben. Sie ist mit Ton-Sequenzen, Hüllkurven, Synchronisation und Tonerzeugung befaßt.

f) Kassetten-Verwaltung

Die Kassetten-Verwaltung ist in Abschnitt 8 beschrieben. Sie bewältigt das Lesen vom Band, das Schreiben auf Band und das Steuern des Kassetten-Motors.

g) Betriebssystem-Kern

Der Betriebssystem-Kern ist in Abschnitt 2, 9, 10 und 11 beschrieben. Er ist das Zentrum des Betriebssystems und verwaltet Unterbrechungen, Ereignisse, die Auswahl der ROMs und den Ablauf der Programme.

h) Maschinen-Paket

Das Maschinen-Paket ist in Abschnitt 12 dokumentiert. Es verwaltet den Drucker und die Ansteuerung der Hardware auf unterster Ebene.

i) Sprungtabelle

Die Sprungtabelle ist in Abschnitt 13 aufgelistet. Die Einträge in die Sprungtabelle sind in Abschnitt 14 detailliert beschrieben. Die Sprungtabelle wird durch Jumper aufgebaut.

1.3 Steuerung der Firmware

Die Firmware wird bevorzugt durch Benutzer-Aufrufe von Routinen und nicht durch Benutzer-Einträge von System-Variablen-Werten gesteuert. Diese zu bevorzugende Möglichkeit erlaubt eine Änderung der Firmware-Variablen ohne Beeinflussung des Anwenders.

Die Adressen der vom Benutzer aufzurufenden Routinen sollten bei einer Firmware-Änderung unangetastet bleiben. Dies ist durch Sprungtabellen erreichbar.

Der Vorteil einer Routinen-Schnittstelle besteht in der Veränderbarkeit einer Reihe unterschiedlicher System-Variablen durch die Firmware in einer Operation und auf immer gleiche Art und Weise. Wenn jedoch System-Variablen durch Benutzer-Einträge gesetzt würden, könnte die Firmware in einem unbestimmbaren Zustand verbleiben. Dies ist dann der Fall, wenn einige Variablen gesetzt, andere aber nicht gesetzt sind. Die Routinen-Schnittstelle gewährleistet somit, daß alle erforderlichen Randbedingungen einer Änderung automatisch berücksichtigt werden und der Benutzer von Detail-Problemen entlastet wird. Ein Beispiel ist die Änderung des Bildschirm-Modus (siehe Abschnitt 6.1). Die Veränderung der Bildschirmgröße erfordert eine Reihe zusätzlicher Informationen an anderer Stelle, damit Fehler wie z. B. verbotene Bildschirmpositionen und Inks (Tinten) nicht auftreten können.

1.4 Sprungtabellen

Eine Sprungtabelle besteht aus einer Reihe von Sprungbefehlen, die im Speicher auf wohldefinierten Adressen liegen. Die Sprungbefehle führen in die verschiedenen Firmware-Routinen, die der Benutzer aufrufen möchte. Programme, die von den Routinen Gebrauch machen möchten, sollten den entsprechenden Sprungtabellen-Eintrag aufrufen.

Wird die Firmware modifiziert, so ist es sehr wahrscheinlich, daß sich die Adressen einiger Routinen ändern. Durch Beibehaltung der Sprungtabellen-Adressen und Änderung der Einträge in den Sprungtabellen (so, daß sie auf die neuen Routinen-Adressen zei-

gen), bleibt die Modifikation dem Benutzer verborgen; vorausgesetzt der Benutzer ruft die Routinen nur über die Sprungtabelle und nicht direkt über die Firmware auf.

Um die Firmware-Modifikationen aus der Sicht des Benutzers zu verbergen, ist es ebenso notwendig, die Einsprung- und Aussprung-Bedingungen der über die Sprungtabelle gerichteten Routinen-Zugriffe einzuhalten. Ein großer Anteil dieses Handbuches befaßt sich detailliert mit den Einsprung- und Aussprung-Anforderungen der Sprungtabellen-Einträge.

Die Sprungtabelle befindet sich im RAM, damit der Benutzer die Einträge ändern kann. Dies ermöglicht dem Anwender bestimmte Einträge aufzufinden und die Standard-Firmware-Routinen durch neue Routinen zu ersetzen.

Unter der Voraussetzung, daß die neue Routine den Einsprung- und Aussprung-Anforderungen genügt, kann diese Substitution nicht zu unbeabsichtigten Programmabstürzen führen.

Alle vier Sprungtabellen sind in Abschnitt 13 aufgelistet. Die erste und größte Sprungtabelle ist die Haupt-Firmware-Sprungtabelle (siehe Abschnitt 13.1 und 14). Sie ermöglicht den Aufruf der meisten Firmware-Routinen. Die zweite Sprungtabelle ist die Indirections-Sprungtabelle (siehe Abschnitt 13.2 und 15). Die Einträge in dieser Sprungtabelle werden durch die Firmware zu bestimmten Zeitpunkten verwendet, um dem Anwender die Änderung von Firmware-Funktionen zu ermöglichen. Die beiden letzten Sprungtabellen dienen sehr speziellen Zwecken. Sie befassen sich mit dem Betriebssystem-Kern, der Freischaltung von ROMs und dem Aufruf von ROM-Routinen (siehe Abschnitt 13.3, 13.4, 16 und 17).

Abschnitt 1.7 zeigt als Beispiel, wie ein Sprungtabellen-Eintrag geändert werden sollte, um die Firmware-Funktionen umzustellen.

1.5 Konventionen

a) Notation

Auf Prozessor-Befehle wird in der Regel durch normale Z80-Mnemonics Bezug genommen. Eine Ausnahme sind die Restart-Befehle. Die Mnemonics RST0 ... RST7 werden statt der gewöhnlichen Z80-Mnemonics RST#00 ... RST#38 verwendet.

Die Register werden ebenso durch die Standard-Z80-Namen bezeichnet. Das gesamte Flag-Register wird mit F bezeichnet, die einzelnen Flags jedoch mit ihrem vollen Namen, z. B. Carry. Die Flags befinden sich im Zustand wahr, wenn sie gesetzt und im Zustand falsch, wenn sie gelöscht sind. Damit ist die Sprungbedingung in einem JNC-Befehl erfüllt, wenn das Carry falsch ist und nicht erfüllt, wenn das Carry wahr ist.

Hexadezimale Zahlen werden durch ein vorgestelltes # gekennzeichnet, d. h. #7F entspricht der Dezimalzahl 127. Alle Zahlen ohne vorgestelltes # sind Dezimalzahlen.

Große Zahlen sind häufig durch die Darstellung in Vielfachen von 1024 abgekürzt. So bedeutet beispielsweise 32 KBytes, 32 x 1024 (entsprechend 32768) Bytes.

b) Anwendung

Routinen übergeben und übernehmen gewöhnlich Werte in Registern. Wo mehr Informationen an eine Routine übergeben werden müssen, als in einem Register möglich wäre, ist die Adresse eines Datenbereiches angegeben. Der Speicherplatz für diese Datenbereiche kann manchmal kritisch werden, siehe hierzu Abschnitt 2.4.

Ob eine Routine diese Bedingung erfüllt oder verletzt wird normalerweise durch den Zustand des Carry-Flags angezeigt. Ist das Carry-Flag wahr, bedeutet dies in der Regel Erfolg, während falsch normalerweise Fehler anzeigt.

Der Registersatz AF' BC' DE' HL' ist für das System reserviert. Der Anwender sollte weder eine EX AF, AF' noch einen EXX-Befehl verwenden, da dies recht unangenehme Konsequenzen haben könnte (siehe Anhang XI).

c) Allgemeines

Die logischen Werte wahr und falsch werden im allgemeinen durch #FF bzw. #00 dargestellt. Oftmals wird jedoch jeder Wert, der nicht Null ist, als wahr interpretiert.

Die Bits innerhalb eines Bytes sind von 0 bis 7 durchnumeriert, wobei Bit 0 das niederwertigste und Bit 7 das höchstwertigste Bit ist.

Dort, wo zwei Bytes (ein Wort) gespeichert werden (z. B. in Tabellen), sind sie immer mit dem niederwertigen Byte zuerst und dem höherwertigen Byte als zweites abgelegt, es sei denn, ein spezieller Hinweis widerspricht dieser Festlegung. Diese Anordnung entspricht der Standard-Z80-Speicherbelegung.

In Tabellen ist Byte 0 immer das erste Byte. Wenn nicht ausdrücklich anders vermerkt, ist eine gegebene Tabellen-Adresse immer als die Adresse von Byte 0 der Tabelle zu interpretieren.

Wenn der Computer eingeschaltet wird (oder bei Reset), initialisiert er sich selbst, bevor irgend ein Programm abgearbeitet werden kann. Diese Initialisierung wird hier als „Early Morning Startup" oder abgekürzt EMS bezeichnet.

1.6 Routinen-Dokumentation

Jede in diesem Handbuch beschriebene Routine besitzt ganz bestimmte Einsprung- und Aussprung-Bedingungen. Sind zu den Routinen noch weitere Anmerkungen erforderlich, so finden sie sich im Abschnitt nach den entsprechenden Einsprung- und Aussprung-Bedingungen. Solche Anmerkungen beinhalten z. B. die Interrupt-(Unterbrechungs-)Freigabe, sowie eine ausführliche Beschreibung der Parameter und Randbedingungen der Routine. Es bestehen zwei Gründe für die Mitteilung dieser Informationen. Erstens wird dem Anwender dadurch deutlich, was beim Aufruf dieser Routinen geschieht, zweitens erfährt er, was von einer Ersatz-Routine erwartet wird.

Die Einsprung-Bedingungen teilen dem Aufrufer der Routine mit, welche Informationsübergabe die Routine erwartet. Beim Aufruf einer Routine müssen alle spezifizierten Werte übergeben werden. Werte dürfen nur dann ausgelassen werden, wenn sie in der Routine als Option gekennzeichnet sind. Soll über diese

Schnittstelle eine Ersatz-Routine arbeiten, so dürfen nur die spezifizierten Informationen verwendet werden. Es ist jedoch nicht erforderlich, alle Informationen zu übergeben.

Die Aussprung-Bedingungen teilen dem Aufrufer der Routine mit, welche Werte die Routine übergibt und welche Prozessor-Register geschützt sind. Register, die als verfälscht bezeichnet werden, können u. U. von der Routine verändert werden. Der Anwender sollte den Inhalt dieser Register ignorieren. Beim Erstellen einer Routine für diese Schnittstelle ist es von großer Wichtigkeit, zu beachten, daß die als geschützt deklarierten Register tatsächlich geschützt werden. Weiterhin sollten die von der Routine übergebenen Werte kompatibel zu denen der Original-Routine sein. Das Verändern eines Registers oder das Fortlassen eines Resultats verursacht in der Regel einen Systemfehler, der häufig unerwartet eintritt und heimtückische Auswirkungen haben kann.

Oftmals besitzt eine Routine mehrere unterschiedliche Aussprung-Bedingungen in Abhängigkeit von bestimmten anderen Bedingungen. Unter diesen Umständen sind die speziellen Unterschiede in den Aussprung-Bedingungen für alle auftretenden Fälle und Bedingungen aufgeführt, unabhängig davon, ob diese Fälle bereits in einem gesonderten Abschnitt behandelt wurden (markiert mit „generell").

Eine Reihe von Beispielen für Routinen-Schnittstellen finden sich in Abschnitt 14 bis 17.

1.7 Sprungtabellen-Behandlung

Es folgt ein Beispiel über die Arbeitsweise einer Sprungtabelle. Auf dieser Ebene werden dem Leser einige eingeführte Begriffe unverständlich sein. Da jedoch die Änderung von Sprungtabellen eine sehr wichtige Technik für die Systemanpassung an bestimmte Anwendungsfälle ist, soll dieses Beispiel hier aufgeführt werden. Spätere Abschnitte erklären die hier verwendeten Begriffe genauer.

Man nehme an, ein Assembler-Programm soll geschrieben werden, welches einen Drucker verwendet. Während der Erstellung dieses Programmes ist es zeit- und papiersparender, wenn statt des Druckers der Bildschirm zur Darstellung verwendet wird. Bei entsprechender Änderung des Programmes jedoch, könnten Fehler bei der Rückstellung auf den Drucker auftreten. Was zur Vermeidung dieser Fehler nötig wäre, ist eine Möglichkeit zur Änderung der Firmware-Funktion, die den Drucker ansteuert. Genau diesem Zweck dient die Sprungtabelle im RAM.

Die Technik, die hierzu angewendet wird, ist das Hineinprojezieren des Druckers in ein bestimmtes Text-Fenster. Dies kann durch Erstellen einer kurzen Routine, die Zeichen zum Bildschirm sendet und die Einträge in die Sprungtabelle für die Funktion „Zeichen zum Drucker senden!" (MC PRINT CHAR) erstellt, erreicht werden. Auf diese Weise wird ein Sprung in diese Routine und nicht in die normale Routine erreicht.

Die Ersatz-Routine muß die Einsprung- und Ausssprung-Bedingungen für MC PRINT CHAR berücksichtigen (siehe Abschnitt 14). Zusammengefaßt lauten sie wie folgt:

MC PRINT CHAR:

Einsprung-Bedingungen:

A beinhaltet das zu druckende Zeichen.

Aussprung-Bedingungen:

Zeichen korrekt übertragen:
Carry wahr.

Drucker ausgeschaltet:
Carry falsch.

Generell:
A und andere Flags verfälscht.
Alle anderen Register geschützt.

Die Funktion der Ersatz-Routine besteht darin, den Bildschirm-Kanal auszuwählen, auf dem die Drucker-Ausgabe erscheinen soll, das Zeichen auf diesem Kanal auszugeben und anschließend den ursprünglich ausgewählten Kanal wiederherzustellen. Für diese Aktionen benötigt die Ersatz-Routine den Aufruf der Routinen TXT STR SELECT und TXT OUTPUT. Die vollständige Beschreibung der Sprungtabellen-Einträge befindet sich in Abschnitt 14. Die Einsprung- und Aussprung-Bedingungen lauten wie folgt:

TXT STR SELECT:

Einsprung-Bedingungen:

A beinhaltet die selektierte Kanal-Nummer.

Aussprung-Bedingungen:

A beinhaltet die vorher selektierte Kanal-Nummer.
HL und Flags verfälscht.
Alle anderen Register geschützt.

TXT OUTPUT:

Einsprung-Bedingungen:

A beinhaltet auszudruckendes Zeichen.

Aussprung-Bedingungen:

Alle Register und Flags geschützt.

Die Ersatz-Routine könnte folgendermaßen aussehen (Kanal 7 wurde als Kanal für die Ausgabe der Drucker-Daten gewählt):

```
PUSH HL
PUSH BC
;
  LD B, A                    ; Sichern des auszudruckenden Zeichens
;
  LD A, 7                    ; Drucker-Kanal-Nummer
  CALL TXT_STR_SELECT        ; Auswahl des Drucker-Kanals
  LD C, A                    ; Sichern der Original-Kanal-Nummer
;
  LD A, B                    ; Zeichen erneut holen
  CALL TXT _OUTPUT           ; Zeichen zum Bildschirm
;
  LD A, C                    ; Hole Original-Kanal-Nummer
  CALL TXT_STR_SELECT        ; Selektiere Original-Kanal
;
  POP BC
  POP HL
  SCF                        ; Zeichen korrekt übertragen
  RET
```

Hinweise:

1. MC PRINT CHAR schützt HL und BC. Die obige Routine verwendet B und C als temporären Speicher. HL ist durch TXT STR SELECT verfälscht. Daher werden HL und BC durch die Ersatz-Routine mittels Push und Pop gerettet.

2. MC PRINT CHAR übergibt eine Erfolg-/Fehler-Meldung im Carry-Flag. Da die obige Routine nicht fehlerhaft ist, wird das Carry Flag immer gesetzt und zeigt damit den Erfolg an.

3. Die obige Routine verändert den selektierten Text-Kanal nicht. Sie selektiert den Kanal, über den gedruckt werden soll und stellt, nachdem das Zeichen gedruckt wurde, den ursprünglich selektierten Kanal wieder ein. Die Firmware ermöglicht eine Wiederherstellung des Original-Zustands, nachdem die Routinen abgearbeitet sind.

Zur Anwendung der Ersatz-Routine ist es erforderlich, sie in den Speicher einzuschreiben und die Sprungtabellen-Einträge für MC PRINT CHAR zu ändern. Nehmen wir an, daß der Speicherplatz ab #AB00 für die Ersatz-Routine reserviert wurde und die Routine bereits im Speicher steht. Der Sprungtabellen-Eintrag für MC PRINT CHAR steht unter Adresse #BD2B (siehe auch Abschnitt 13.1.8). Die drei Bytes des Sprungtabellen-Eintrags sollten mit dem Befehl JP#AB00 wie folgt belegt werden:

```
#BD2B     #C3
#BD2C     #00
#BD2D     #AB
```

Nun erscheint jeder zum Drucker ausgegebene Text über Kanal 7 auf dem Bildschirm. Natürlich muß das Text-Fenster von Kanal 7 so eingestellt werden, daß es nicht andere Bildschirm-Kanäle beeinflußt.

Diese Umstellung bleibt in Kraft bis der ursprüngliche Sprungtabellen-Eintrag wiederhergestellt wird. Dies kann durch erneute Änderung des Sprungtabellen-Eintrags, durch Aufruf von JUMP RESTORE oder durch eine EMS-Initialisierung mittels System-Reset geschehen.

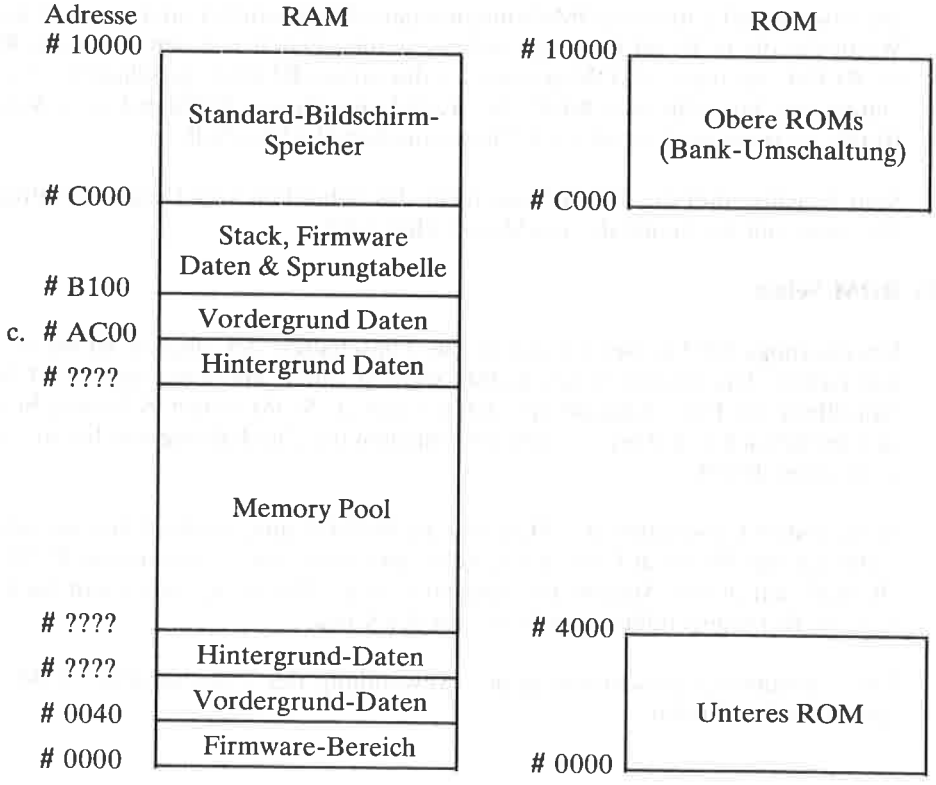
2 ROM, RAM und die Restart-Befehle

Das System ist mit 32 KB ROM und 64 KB RAM im 64 KB-Adressraum des Z80 belegt. Um dies zu ermöglichen, kann das ROM freigeschaltet oder gesperrt werden. Durch zusätzlich selektierbare Erweiterungs-ROMs können bis zu 4632 KB Programm-Bereich genutzt werden.

Mit einer Ausnahme sind alle Z80-Restart-Befehle für System-Anwendungen reserviert. RST1 bis RST5 dienen der Erweiterung des Befehlssatzes durch Implementierung spezieller Call- und Jump-Befehle für die Freischaltung und Sperrung der ROMs. RST6 steht dem Anwender zur Verfügung.

2.1 Speicherbelegung

Die Speicherbelegung ist durch die Tatsache, daß in den 64KB-Adressraum des Z80 64 KB RAM und 32 KB ROM unterzubringen sind, sowie Vorkehrungen für eine ROM-Erweiterung auf bis zu 252 x 16KB (ca. 4MB) getroffen wurden, kompliziert geworden. Der Adressraum ist folgendermaßen aufgeteilt:



Die Größe der beiden Hintergrund-Bereiche hängt von den Hintergrund-ROMs der Maschine ab (siehe Abschnitt 9).

Der obere Vordergrund-Datenbereich muß nicht zwangsläufig mit seiner Anfangsadresse auf #AC00 liegen, dies ist nur die Standardadresse, auf die sich BASIC bezieht. Der untere Vordergrund-Datenbereich muß nur dann reserviert werden, wenn er auch tatsächlich benötigt wird (dieser Bereich wird durch BASIC nicht belegt und ist auf die Länge 0 gesetzt). Der Speicherbereich zwischen den Hintergrund-Datenbereichen (Memory Pool) ist auch für das Vordergrund-Programm belegbar (siehe Abschnitt 9).

Der 32K-ROM-Speicher des Systems ist in zwei Abschnitte geteilt, die separat gehandhabt werden. Künftig werden diese beiden Abschnitte beschrieben, als wären es separate ROMs. Die Firmware befindet sich im unteren ROM. BASIC befindet sich im oberen ROM. Dieses obere ROM ist ‚bankweise' umschaltbar, so daß es durch bis zu 252 Erweiterungs-ROMs (siehe Abschnitt 9) in der Speicherbelegung ersetzt werden kann.

2.2 ROM-Auswahl

Es bestehen zwei Möglichkeiten, ROMs ein- oder auszublenden:

a) ROM-State (ROM-Zustand)

Die oberen und unteren ROMs können separat freigeschaltet oder gesperrt werden. Wenn das obere ROM freigeschaltet ist, werden Daten von den Adressen #C000 bis #FFFF aus diesem ROM gelesen. Ist das untere ROM freigeschaltet, so werden Daten von den Adressen #0000 bis #3FFF aus diesem ROM gelesen. Wenn die ROMs gesperrt sind, werden alle Daten aus dem RAM geholt.

Man beachte, daß der ROM-State nicht das Schreiben von Daten beeinflußt, da hierdurch nur der Inhalt des RAM verändert wird.

b) ROM-Select

Erweiterungs-ROMs werden durch das Umschalten des oberen ROM-Bereichs unterstützt. Die Erweiterungs-ROMs werden durch ein separates ROM-Select-Adreßbyte im Ein-/Ausgabe-Bereich adressiert. ROM-Select-Adressen befinden sich im Bereich von 0 bis 251 und ermöglichen die Zuschaltung von bis zu 252 Erweiterungs-ROMs.

Beim ersten Einschalten der Maschine ist ROM 0 ausgewählt. Dadurch wird gewöhnlich das ROM auf der Karte selektiert, aber ein Erweiterungs-ROM kann ebenfalls auf diesen Adreßbereich gelegt werden. Bevorzugt wird dann das Erweiterungs-ROM gegenüber dem ROM auf der Karte.

Eine vollständige Beschreibung der Anwendung des Erweiterungs-ROMs ist in Abschnitt 9 zu finden.

2.3 Restart-Befehle

Der Betriebssystem-Kern unterstützt die Speicher-Belegung auf vielfältige Weise. Im Einzelnen stehen eine Reihe von Möglichkeiten zur Handhabung von erweiterten Subroutinen-Adressen für den Einsatz von ROM-Select- und/oder ROM-State-Informationen zur Verfügung. Einige Restart-Befehle werden zur Erweiterung des bestehenden Z80-Befehlssatzes verwendet. Die anderen Restart-Befehle sind reserviert.

Der Firmware-Bereich zwischen #0000 und #003F ist aktiviert, so daß die Restart-Befehle unabhängig vom momentanen ROM-State funktionieren. Der Benutzer sollte den Inhalt dieses Bereichs, mit Ausnahme der in Abschnitt 17 aufgeführten Besonderheiten, nicht ändern.

Die Restart-Befehle sind wie folgt belegt (eine vollständige Beschreibung folgt in Abschnitt 17):

a) Der erweiterte Befehlssatz

LOW JUMP (RST1)

RST1 springt in eine Routine im niederwertigen 16KB-Speicherbereich. Die beiden dem Restart-Befehl folgenden Bytes werden als die „niederwertige Adresse" interpretiert. Der RST1 kann somit als 3-Byte-Befehl ähnlich einem JP-Befehl betrachtet werden.

Die oberen beiden Bits der „niederwertigen Adresse" definieren den erforderlichen ROM-Freischalt-/Sperr-Zustand; die unteren 14 Bits geben die aktuelle Sprung-Adresse (im Bereich von #0000 bis #3FFF) an, wenn der ROM-State gesetzt ist. Wenn die Routine verlassen wird, ist das ROM in seinen Original-Zustand zurückgesetzt.

Die Firmware-Sprungtabelle, über die Firmware-Routinen aufgerufen werden sollten, macht regen Gebrauch von den LOW JUMPs. Diese LOW JUMPs fordern die Freischaltung des unteren ROMs an. Damit kann das untere ROM abgeschaltet werden, sofern die Firmware nicht aktiv ist.

SIDE CALL (RST2)

RST2 ruft eine Routine in einem zugeschalteten ROM auf. Er dient sehr speziellen Anwendungsfällen. Ein Vordergrund-Programm (siehe Abschnitt 9) erfordert z. B. mehr als 16KB ROM. Die SIDE CALL-Funktion ermöglicht Aufrufe zwischen zwei, drei oder vier zugeschalteten ROMs ohne Bezugnahme auf ihre aktuellen ROM-Select-Adressen, unter der Voraussetzung, daß die ROMs nebeneinander und in der richtigen Reihenfolge installiert sind.

Die zwei nach dem Restart-Befehl folgenden Bytes geben die „Side-Address" der aufzurufenden Routine an. Somit kann der RST2 ähnlich einem CALL-Befehl als 3-Byte-Befehl aufgefaßt werden. Die oberen 2 Bits der „Side Address" spezifizieren das zu selektierende ROM; die unteren 14 Bits ergeben nach der Addition mit #C000 die aktuelle Routinen-Adresse. Das obere ROM ist damit freigeschaltet, das untere ROM gesperrt. Sowohl ROM-State als auch ROM-Select sind nach dem Verlassen der Routine in ihrem Original-Zustand.

FAR CALL (RST3)

RST3 kann überall im Speicher, sowohl im RAM als auch in jedem ROM, eine Routine aufrufen. Die beiden auf den Restart-Befehl folgenden Bytes werden als Adresse einer „Far Address" interpretiert. Die „Far Address" ist ein 3-Byte-Objekt der Form:

Bytes 0...1: Aktuelle Adresse der aufzurufenden Routine.
Byte 2: ROM-Select/-State

Das ROM-Select- bzw. -State-Byte kann folgende Werte annehmen:

0...251: Selektiere das obere ROM unter dieser ROM-Select-Adresse. Schalte das obere ROM frei und sperre das untere ROM.

252...255: Keine Änderung des ROM-Select; ROM-Freischaltung/-Sperrung wie folgt:

252: Freischaltung oberes ROM. Freischaltung unteres ROM
253: Freischaltung oberes ROM, Sperrung unteres ROM
254: Sperrung oberes ROM, Freischaltung unteres ROM
255: Sperrung oberes ROM, Sperrung unteres ROM

Man beachte, daß die „Far Address" nicht selbst im Befehl enthalten ist, sondern nur auf sie gezeigt wird. Der Grund hierfür ist, daß die ROM-Select-Adresse von der konkreten Anordnung der Erweiterungs-ROMs durch den Anwender abhängig ist und erst zur Laufzeit festgelegt wird.

Sowohl ROM-State als auch ROM-Select werden nach dem Rücksprung aus der Routine in ihren Original-Zustand zurückgesetzt.

RAM LAM (RST4)

RST4 liest das Byte aus dem RAM. dessen Adresse im HL-Register steht. Er sperrt beide ROMs vor dem Lesen und stellt den Ausgangszustand anschließend wieder her. Dieser Befehl verhindert, daß der Benutzer eine Lese-Routine innerhalb des zentralen 32KB-RAM-Bereichs unterbringen muß, um auf ein durch ROM überlagertes RAM zugreifen zu können.

Das Schreiben auf einen Speicherplatz ändert unabhängig vom ROM-Freischalt-Zustand immer den Inhalt des RAMs.

FIRM JUMP (RST5)

RST5 schaltet das untere ROM ein und springt in eine Routine. Die beiden auf den Restart-Befehl folgenden Bytes dienen als Sprungziel. Dadurch stellt sich der RST5 ähnlich einem JP-Befehl als 3-Byte-Befehl dar. Das untere ROM wird vor dem Einsprung in die Routine freigeschaltet und nach dem Rücksprung aus der Routine gesperrt. Der Zustand des oberen ROMs bleibt unverändert.

b) Weitere Restarts

RESET (RST0)

RST0 setzt das System in den Grundzustand zurück, als wäre es soeben eingeschaltet worden.

USER RESTART (RST6)

RST6 steht dem Anwender frei zur Verfügung. Er kann zur Erweiterung des Befehlssatzes ähnlich den anderen Restarts oder für andere Zwecke, z. B. für einen Unterbrechungs-Befehl in einem Debugger, verwendet werden.

Die Speicherplätze #0030 bis inklusive #0037 im RAM können zur Steuerung der Restart-Befehle belegt werden. Ist das untere ROM während der Restart-Ausführung freigeschaltet, so verursacht der unter dieser Adresse gespeicherte Code im ROM die Sicherung des momentanen ROM-Zustands unter Adresse #002B. Dadurch ist das untere ROM gesperrt und die Firmware verursacht einen Sprung auf Adresse #0030 im RAM. Wenn das untere ROM gesperrt ist, wird wie bei einem normalen Z80-Restart-Befehl die Adresse #0030 aufgerufen.

INTERRUPT (RST7)

RST7 ist für Unterbrechungen reserviert (siehe Abschnitt 10) und darf nicht durch ein Programm ausgeführt werden.

2.4 RAM und die Firmware

Der ROM-State sollte für den Anwender transparent sein. Wenn sich das momentan bearbeitete Vordergrund-Programm (siehe Abschnitt 9) im ROM befindet, ist der normale ROM-State so, daß das obere ROM freigeschaltet und das untere ROM gesperrt ist. Befindet sich das momentan bearbeitete Programm jedoch im RAM, so sind im Normalzustand beide ROMs gesperrt. Diese Zustände ermöglichen dem Vordergrund-Programm freien Zugriff auf den Memory-Pool. Beim Aufruf einer Firmware-Routine ist das untere ROM freigeschaltet und das obere ROM gewöhnlich gesperrt. Dies ermöglicht der Firmware freien Zugriff auf den Standard-Bildschirm-Speicherbereich (aber nicht auf den gesamten Memory Pool). Nach dem Rücksprung aus der Firmware-Routine ist der ROM-State automatisch wiederhergestellt.

Die Umstände, unter denen der ROM-State von Bedeutung ist, sind:

a) Stack

Der Hardware-Stack sollte niemals unterhalb #4000 liegen, da sonst bei Zugriffen auf den Stack ein gefährliches Durcheinander herrscht, wenn das untere ROM freigeschaltet ist (z. B. beim Auftreten von Unterbrechungen oder Firmware-Aufrufen).

Andererseits ist es auch nicht ratsam, den Stack oberhalb von #C000 zu legen, da dann bei Stack-Zugriffen niemals das obere ROM freigeschaltet werden darf.

Das System stellt einen über 256 Bytes großen Stack-Bereich unmittelbar unterhalb #C000 zur Verfügung. Dies ist in den meisten Fällen ausreichend.

b) Kommunikation mit der Firmware

Die meisten Firmware-Routinen übernehmen ihre Parameter in Registern. Einige verwenden jedoch auch Datenbereiche im Speicher zum Austausch von Informationen. Viele Firmware-Routinen, die Datenbereiche im Speicher verwenden, lesen diese direkt ohne RAM LAMs oder deren Äquivalent. Diese Routinen werden vom ROM-State und ROM-Select beeinflußt. Sie lesen Daten aus einem ROM, sofern das ROM freigeschaltet und eine passende Adresse angelegt wird (man beachte, daß die Sprungtabelle das obere ROM sperrt, wenn die Firmware aufgerufen wird). Wieder andere Firmware-Routinen, die Datenbereiche im Speicher verwenden, lesen immer aus dem RAM. Sie bleiben vom ROM-State und ROM-Select unbeeinflußt.

Bei Routinen, die immer auf RAM-Bereiche zugreifen, wird dies in der Routinen-Beschreibung erwähnt. Die anderen Routinen sollten als vom ROM-State beeinflußbar betrachtet werden. Insbesondere die vom Betriebssystem-Kern benötigten Datenblöcke müssen im zentralen 32KB RAM liegen.

c) Kommunikation zwischen den oberen ROMs

Programme in den oberen ROMs können durch Anwendung der verschiedenen Möglichkeiten des Betriebssystem-Kerns Routinen in anderen ROMs aufrufen. In der Firmware sind jedoch keine Vorkehrungen getroffen worden, um durch ein Programm in einem ROM auf Konstanten in einem anderen ROM zugreifen zu können.

Die Mehrzahl der Firmware-Routinen wird über die Firmware-Sprungtabelle aufgerufen, die bei Adresse #BB00 im Firmware-RAM-Bereich beginnt. Die Betriebssystem-Kern-Routinen für die Speicherbelegung werden über einen von zwei Sprungtabellen-Bereichen aufgerufen:

– dem LOW-Bereich zwischen #0000 und #003F
und
– dem HIGH-Bereich beginnend ab #B900.

Alle entsprechenden Bestandteile dieser Routinen und Sprungtabellen werden bei der Initialisierung des Betriebssystem-Kerns vom unteren ROM in den Firmware-RAM-Bereich kopiert. Dadurch arbeiten sie unabhängig vom ROM-State.

3 Tastatur

Die Tastatur-Verwaltung ist das Software-Paket zur Erzeugung und Steuerung der Tastatur-Attribute. Diese Attribute beinhalten Repeat-Geschwindigkeit, Shift- und Control-Tasten, Funktions-Tasten und Tasten-Übersetzung. Die Joysticks werden ebenfalls von der Tastatur-Verwaltung abgetastet.

Die Tastatur-Verwaltung besitzt drei Operations-Ebenen. Die niedrigste Ebene fragt die Tastatur ab, die mittlere Ebene übersetzt die betätigten Tasten in Tasten-Werte und die oberste Ebene übersetzt die Tasten-Werte in Zeichen. Der Anwender kann die Tastatur-Verwaltung auf der Ebene ansprechen, die für ein gegebenes Programm am zweckmäßigsten erscheint, Es ist jedoch für ein Programm gewöhnlich wenig vorteilhaft, Zugriffe auf unterschiedlichen Ebenen zu mischen.

3.1 Tastatur-Abfrage

Die Tastatur ist vollständig Software-gesteuert und wird automatisch alle 20 ms abgetastet. Die Tastatur-Hardware wird gelesen und eine Bit-Belegung aufgebaut, welche die betätigten Tasten erkennen läßt. Diese Bit-Belegung steht für Testzwecke zur Verfügung, wenn bestimmte Tasten gedrückt werden (siehe KM TEST KEY). Sobald die Bit-Belegung aufgebaut ist, werden erneut gedrückte Tasten erkannt und Marken in einem Puffer gespeichert. Wenn keine erneut betätigte Taste erkannt wird, kann die zuletzt gedrückte Taste, sofern sie noch niedergedrückt ist, wiederholt werden (siehe Abschnitt 3.5). Die Tastatur ist entprellt, indem eine gedrückte Taste erst dann in der Bit-Belegung als inaktiv gekennzeichnet wird, wenn sie für zwei aufeinanderfolgende Abtast-Zeitpunkte losgelassen ist. Dieses Entprellen unterdrückt mehrfache Operationen der Tastatur beim Öffnen oder Schließen der Tastaturschalter.

Vorerst sollen hier nur vier Tasten behandelt werden. Die beiden Shift-Tasten und die Control-Taste werden nicht selbst im Tastenpuffer abgespeichert. Statt dessen werden die Zustände der Shift- und Control-Tasten erkannt und in den Puffer übernommen, wenn irgend eine andere Marke gespeichert wird. Die Escape-Taste erzeugt üblicherweise eine Marke, kann aber auch in Abhängigkeit von der Ausgestaltung des Abbruch-Mechanismus (siehe Abschnitt 3.6) ganz anders geartete Effekte zeigen.

Es gibt ein Problem bei der Tastatur-Abfrage. Wenn drei Tasten in den Ecken eines Rechtecks in der Tasten-Matrix gleichzeitig gedrückt werden, erscheint die Taste in der vierten Ecke ebenfalls als gedrückt. Es besteht keine Möglichkeit, dieses Problem zu lösen, da es ein Merkmal der Tastatur-Hardware ist. Alle durch die Firmware (und BASIC) verwendeten Tastenkombinationen sind speziell zur Vermeidung dieses Effekts ausgelegt.

3.2 Tasten-Übersetzung

Wenn der Anwender eine Taste abfragt (KM WAIT KEY oder KM READ KEY), wird die nächste durch Tastendruck erzeugte Marke aus dem Tasten-Puffer gelesen. Die Marke wird in eine Tasten-Nummer übersetzt, die in einer der drei Übersetzungsebenen benötigt wird. Die dann verwendete Tabelle hängt davon ab, ob die Shift- und Control-

Tasten gleichzeitig betätigt wurden oder nicht. Eine Tabelle steht für gedrückte Control-Taste, eine andere für betätigte Shift-Taste und nicht betätigte Control-Taste, die dritte für nicht betätigte Shift-Taste und nicht betätigte Control-Taste. Der Inhalt dieser Tabellen kann – falls erforderlich – geändert werden (durch KM SET CONTROL, KM SET SHIFT und KM SET TRANSLATE).

Der aus der Tabelle entnommene Wert kann ein System-Wert, Erweiterungs-Wert oder ein Zeichen sein. Erweiterungs-Werte und Zeichen werden durch die oberste Ebene der Tastatur-Verwaltung bearbeitet (siehe Abschnitt 3.3) und von der mittleren Ebene, nach dem Auffinden in einer Tabelle, übergeben. Es gibt drei System-Werte, die unmittelbar nach dem Auffinden in einer Tabelle abgearbeitet werden.

Nach Bearbeitung dieser Werte wird die nächste Marke aus dem Puffer gelesen und übersetzt.

Die Standard-Übersetzungstabellen sind in Anhand II beschrieben.
Die unmittelbar bearbeiteten System-Werte sind:

a) Ingore (#FF)
Die betätigte Taste wird ignoriert.

b) Shift-Lock (#FD)
Der Shift-Lock wird „gekippt" (d. h. eingeschaltet, wenn er momentan ausgeschaltet ist und ausgeschaltet, wenn er momentan eingeschaltet ist).

c) Caps-Lock (#FD)
Der Caps-Lock wird „gekippt" (d. h. eingeschaltet, wenn er momentan ausgeschaltet ist und ausgeschaltet, wenn er momentan eingeschaltet ist).

3.3 Zeichen von der Tastatur

Wenn der Benutzer in der obersten Ebene der Tastatur-Verwaltung ein Zeichen abfragt (KM WAIT CHAR oder KM READ CHAR) wird ein Tasten-Wert von der mittleren Ebene abgeholt. Ist dieser ein Zeichen (#00...#7F oder #A0...#FC), wird er direkt zurückgegeben. Ist er ein Erweiterungs-Wert (#80...#9F), so wird der mit diesem Wert verbundene String (Zeichenkette) abgearbeitet. Die Zeichen in diesem String werden bei jeder Zeichen-Abfrage eines nach dem anderen ausgegeben, bis das String-Ende erreicht ist.

Es gibt nur ein Zeichen auf dieser Ebene mit einer speziellen Bedeutung. Dies ist das Zeichen mit dem Wert #EF. Es wird durch Drücken der Escape-Taste erzeugt und ist ein Abbruch-Ereignis (siehe Abschnitt 3.6). Es hat keinen großen Einfluß und ist lediglich eine Marke für den Platz im Puffer, an dem ein Abbruch-Ereignis stattfand. Die beabsichtigte Anwendung besteht in der gegebenen Möglichkeit, alle Zeichen vor einem Abbruch-Ereignis zu annulieren. Dieses Zeichen wird nicht durch die Übersetzungstabelle erzeugt und kann demzufolge durch eine Änderung derselben nicht beeinflußt werden.

Ein einzelnes „Rückgabe-Zeichen" wird unterstützt. Wenn der Benutzer ein Zeichen zurückgibt, wird dies beim nächsten Aufruf an die oberste Ebene der Tastatur-Ver-

waltung weitergegeben. Dieser Vorgang ist für Programme erforderlich, die das nächste von der Tastatur zu lesende Zeichen testen müssen, ohne es zu verlieren.

3.4 Shift- und Caps-Lock

a) Shift-Lock

Wenn Shift-Lock gesetzt ist, werden die gedrückten Tasten übersetzt, als wäre eine Shift-Taste betätigt. Shift-Lock wird durch einen System-Wert „gekippt", der normalerweise durch Drücken von CTRL und CAPS LOCK erzeugt wird.

b) Caps-Lock

Wenn Caps-Lock gesetzt ist, werden von der Tastatur gelesene Zeichen in ihre Großbuchstaben-Äquivalente umgesetzt. Diese Umsetzung findet statt, bevor Erweiterungs-Werte expandiert werden.

Caps-Lock wird durch einen System-Wert „gekippt", der normalerweise durch Drücken von CAPS LOCK (ohne CTRL) erzeugt wird.

3.5 Tasten-Wiederholung

Es gibt eine Tabelle, die diejenigen Tasten beinhaltet, die beim Betätigen wiederholt werden dürfen (siehe KM SET REPEAT). Diese Tabelle kann vom Anwender nach Belieben geändert werden. Die Standard-Belegung dieser Tabelle ist in Anhang III beschrieben. Die Standard-Belegung ermöglicht eine Wiederholung aller Tasten mit Ausnahme von ESC, TAB, CAPS LOCK, SHIFT, ENTER, CTRL und den 12 Tasten des numerischen Tastenfeldes (Funktionstasten).

Die Wiederholtaste der Tasten und die Verzögerung bis zur ersten Wiederholung kann durch den Anwender festgelegt werden (siehe KM SET DELAY).

Die Standard-Wiederholrate beträgt 25 Zeichen/sec. mit einer Anfangsverzögerung von 0,6 Sekunden.

Für die Tasten-Wiederholung müssen folgende Bedingungen erfüllt sein:

1. Die entsprechende Zeit seit dem ersten Tastendruck oder der letzten Tasten-Wiederholung muß verstrichen sein.
2. Die Taste muß noch niedergedrückt sein.
3. Keine andere Taste darf nach dem Betätigen der zu wiederholenden Taste gedrückt werden.
4. Die Taste muß in der Wiederholungs-Tabelle als wiederholbar markiert sein.
5. Es dürfen keine Tasten im Tasten-Puffer gespeichert sein.

Bedingung 5 bezieht sich darauf, daß die Wiederholrate und die Anfangsverzögerung auf ihre maximalen Werte gesetzt sind. Ist ein Programm in der Entnahme von Tasten aus dem Puffer zu langsam, so stellt sich die Tasten-Erzeugung selbständig darauf ein. Dadurch ist es möglich, daß eine große Anzahl von im Puffer gespeicherten Tasten entstehen, die allein durch Niederdrücken einer Taste erzeugt werden.

Beim Lesen oder Schreiben von der Kassette wird die ESC-Taste auf andere Weise gehandhabt (siehe Abschnitt 8.12).

3.6 Abbrüche

Abbrüche können auftreten, wenn die Tastatur-Abtastung feststellt, daß die ESC-Taste gedrückt ist. Ist dies der Fall, so wird KM TEST BREAK aufgerufen, um den Abbruch zu bearbeiten. Die Standard-Einstellung dieser Routine testet, ob die SHIFT-, CTRL- und ESC-Tasten und keine anderen gedrückt sind. Ist dies der Fall, so wird das System rückgesetzt (durch Ausführung von RST0), andernfalls wird der Abbruch-Mechanismus aufgerufen.

Ist der Abbruch-Mechanismus außer Funktion, so wird keine weitere Reaktion eingeleitet, als die Einfügung der Marke für die Escape-Taste in den Tasten-Puffer. Ist der Abbruch-Mechanismus in Funktion, so werden zwei zusätzliche Aktionen eingeleitet. Zuerst wird eine spezielle Marke in den Tasten-Puffer eingetragen, die nach dem Auffinden das Zeichen #EF erzeugt (unabhängig von der Übersetzungs-Tabelle). Dadurch kann erreicht werden, daß die vor dem Abbruch im Puffer befindlichen Zeichen annulliert werden. Zweitens wird das synchrone Abbruch-Ereignis angestoßen.

Der Abbruch-Mechanismus kann zu jeder Zeit aktiviert oder deaktiviert werden (durch Aufruf von KM ARM BREAK oder KM DISARM BREAK). Deaktiviert ist der Standard-Zustand. Um das mehrfache Auftreten von Abbrüchen zu verhindern, wird der Mechanismus nach der Erkennung eines Abbruchs automatisch deaktiviert. Die Methode, mit der BASIC Abbrüche handhabt, sollte als Modell für andere Programme dienen. Die Aktionen in BASIC sind folgende:

Der Abbruch-Mechanismus ist aktiviert. Nach jedem BASIC-Befehl wird die synchrone Ereignis-Warteschlange abgefragt. Liegt ein Abbruch-Ereignis vor (weil es, wie oben beschrieben, angestoßen wurde), so wird die Abbruch-Ereignis-Routine gestartet.

Die Abbruch-Ereignis-Routine stoppt den Tongenerator (SOUND HOLD) und legt dann alle vor dem Abbruch eingegebenen Zeichen ab, indem solange Zeichen von der Tastatur eingelesen werden (KM READ CHAR), bis entweder der Puffer leer oder die Abbruch-Ereignis-Marke (Zeichen #EF) gefunden ist. BASIC schaltet dann den Cursor ein (TXT CUR ON) und wartet auf das nächste eingegebene Zeichen (KM WAIT CHAR).

Ist das nächste Zeichen das durch Escape gegebene Zeichen #FC (den durch die ESC-Taste erzeugten Standard-Wert), so wird ein Flag gesetzt, um BASIC (oder der Benutzer-Subroutine ON BREAK GOSUB) die weitere Ausführung zu überlassen und aus der Abbruch-Ereignis-Routine auszusteigen.

Ist das nächste Zeichen ein anderes als das Escape-Zeichen, so wird der Abbruch ignoriert. Ist es irgendein anderes Zeichen als das Leerzeichen, so wird es „zurückgegeben" (KM CHR RETURN). Bevor der Rücksprung aus der Ereignis-Routine erfolgt, wird der Cursor ausgeschaltet (TXT CUR OFF), der Tongenerator erneut gestartet (SOUND CONTINUE) und der Abbruch-Mechanismus aktiviert. BASIC setzt dann die Bearbeitung fort, als wäre nichts geschehen.
Beim Lesen oder Schreiben der Kassette wird die ESC-Taste anders behandelt (siehe Abschnitt 8.12).

3.7 Funktions-Tasten und Erweiterungs-Werte

Die Tastatur-Verwaltung ermöglicht 32 Erweiterungs-Werte (Werte #80...#9F), die in der Tasten-Übersetzungs-Tabelle plaziert werden können. Jeder Wert bezieht sich auf einem im Erweiterungs-Puffer gespeicherten String.

Wenn der Anwender von der obersten Ebene ein Zeichen anfordert, wird von der mittleren Ebene eine Taste abgeholt. Handelt es sich bei dieser Taste um ein Zeichen, so wird es zurückgegeben. Ist es jedoch ein Erweiterungs-Wert, so wird der zugehörige String aufgesucht. Die Zeichen in diesem String werden bei jeder Zeichen-Anfrage eines nach dem anderen ausgegeben, bis das String-Ende erreicht ist. Die Werte #80 ...#9F, #EF und #FD...#FF im Erweiterungs-String werden wie Zeichen behandelt und nicht weiter bearbeitet.
Der Anwender kann den mit einem Erweiterungs-Wert verbundenen String zuordnen (siehe KM SET EXPAND) und die Tasten der Tastatur, die einen Erweiterungs-Wert erzeugen, bestimmen. Die Standard-Einstellung für die Erweiterungs-Werte und die mit ihnen verbundenen Tasten sind in Anhang IV aufgeführt. Der Anwender kann auch die Größe und den Speicherplatz des Erweiterungs-Puffers festlegen (siehe KM EXP BUFFER). Der Standard-Puffer ist mindestens 100 Bytes lang.

3.8 Joysticks

Es können zwei Joysticks mit dem System verbunden werden. Sie werden auf die gleiche Weise abgefragt, wie die Tasten der Tastatur. Der zweite Joystick nimmt den gleichen Speicherplatz in der Tasten-Matrix in Anspruch, wie bestimmte andere Tasten auch und ist von ihnen nicht zu unterscheiden. Der Zustand der Joysticks kann durch Aufruf der Routine KM GET JOYSTICK bestimmt werden.

Da die Joysticks wie andere Tasten abgefragt werden, kann auch das Betätigen der Joystick-Knöpfe, wie das der Tasten, erkannt werden. Erstens können unterschiedliche Richtungen oder Knöpfe in der Tasten-Bit-Belegung durch Aufruf von KM TEST KEY (siehe Abschnitt 3.1) getestet werden. Zweitens erzeugen die Joystick-Knöpfe beim Betätigen bestimmte Zeichen (vorausgesetzt, die Übersetzungs-Tabelle ist entsprechend vorbereitet), die dann erkannt werden können. Das Hauptproblem in der zweiten Methode besteht darin, daß die Rate, mit der Zeichen erzeugt werden, von der eingestellten Wiederholrate der Tastatur abhängig ist. Wenn die Wiederholrate gesteigert wird, um den Joystick reaktionsschneller zu machen, kann die Tastatur u. U. nicht mehr verwendet werden.

In Anhang I ist die Numerierung der Tasten und Joystick-Knöpfe aufgeführt, in Anhang II die Standard-Übersetzungs-Tabelle.

4 TEXT-VDU

Der Text-VDU ist ein zeichenorientierter Bildschirm-Treiber. Er steuert 8 unterschiedliche Kanäle, wobei jedem ein Bildschirm-Bereich (ein Fenster) zugeordnet sein kann. Der Text-VDU ermöglicht das Schreiben/Lesen von Zeichen auf den/vom Bildschirm. Er behandelt auch gewisse „Zeichen" als „Steuer-Codes", die verschiedene Auswirkungen, vom Bewegen des Cursors bis zum Einstellen einer Ink-Farbe, haben können.

4.1 Koordinaten-Systeme des Text-VDU

Der Text-VDU verwendet zwei Koordinaten-Systeme, ein logisches und ein physikalisches. Im allgemeinen gibt der Anwender dem Text-VDU Positionen in logischen Koordinaten an. Physikalische Koordinaten werden durch den Anwender gelegentlich und intern verwendet, um Positionen für den Text-VDU zu spezifizieren. Beide Systeme verwenden vorzeichenbehaftete 8-Bit-Zahlen und bestimmen Zeichenpositionen. Jede Zeichenposition ist 8 Pixel (Punkte) breit und 8 Pixel hoch. Dies bedeutet, daß die Koordinaten-Positionen auf dem Bildschirm vom Bildschirm-Modus abhängig sind.

Physikalische Koordinaten sind aufgeteilt in Spalten – von links nach rechts – und Reihen – von oben nach unten –. Die Zeichenposition in der obersten linken Ecke des Bildschirms besitzt die Koordinaten Reihe 0 und Spalte 0.

Logische Koordinaten gleichen den physikalischen Koordinaten mit der Ausnahme, daß sich die Zeichenposition der obersten linken Ecke des momentanen Text-Fensters in der Reihe 1 und Spalte 1 befindet.

4.2 Kanäle

Der Text-VDU kann bis zu 8 Kanäle gleichzeitig bedienen. Jeder Kanal hat einen von den anderen unabhängigen Zustand (obgleich einige Merkmale allen gemeinsam sind, die bei einer Änderung dann auf alle Kanäle wirken). Die vom Kanal abhängigen Merkmale sind:

- VDU-Freischaltung
- Cursor-Freischaltung (freischalten oder sperren, ein oder aus)
- Cursor-Position
- Fenstergröße
- PEN (Zeichenstift)- und PAPER (Papier)-Ink
- Zeichen-Schreibmodus (undurchsichtig oder transparent)
- Graphikzeichen-Schreibmodus

Merkmale, die alle Kanäle beeinflussen, sind:

- Zeichen-Matrix
- Steuer-Code-Puffer
- Text-VDU-Zustand
- Bildschirm-Modus

Alle diese Merkmale sind in den folgenden Abschnitten detailliert beschrieben.

Unter der Voraussetzung, daß der Steuer-Code-Puffer nicht verwendet wird (siehe Abschnitt 4.7), kann der momentan selektierte Kanal zu jeder Zeit und ohne nachteiligen Effekt verändert werden. Ein Kanal bleibt solange selektiert, bis ein anderer ausgewählt wird. Dies bedeutet, daß ein Programm nicht wissen muß, welchen Kanal es verwendet.

Der Standard-Kanal (selektiert beim EMS) ist Kanal 0. BASIC erweitert das Kanal-Konzept und bezieht die Drucker- und Kassetten-Dateien ein. Diese Erweiterung ist nicht Bestandteil der Firmware.

4.3 Text-Pen- und Paper-Inks

Jedem Kanal ist eine Pen (Zeichenstift)- und Paper (Papier)-Ink zugeordnet. Die Text-Pen-Ink dient zur Erzeugung der Vordergrund-Pixel von Zeichen (siehe Abschnitt 4.6). Das Text-Paper wird zur Einstellung der Hintergrund-Pixel von Zeichen und zum Löschen des Text-Fensters verwendet.

Die Pens und Papers können auf jede Ink, die während des momentanen Bildschirm-Modus gültig ist, eingestellt werden (siehe Abschnitt 6.1). Die Standard-Einstellung eines Kanals für Paper ist Ink 0 und für den Pen Ink 1. Die Änderung einer Pen- oder Paper-Ink verändert nicht den Bildschirm direkt, sondern lediglich die zukünftige Schreibweise der Zeichen.

4.4 Text-Fenster

Jeder Kanal hat ein ihm zugeordnetes Text-Fenster. Dieses Fenster spezifiziert den Bereich des Bildschirm, in den über diesen Kanal Zeichen geschrieben werden dürfen. Dies ermöglicht den verschiedenen Kanälen die Verwendung unterschiedlicher Bildschirm-Bereiche ohne gegenseitige Störung.

Die Fenster sind so ausgerichtet, daß sie in dem momentanen Bildschirm-Ausschnitt (dessen Größe mit dem Bildschirm-Modus variiert, siehe Abschnitt 6.1) hineinpassen. Die kleinste mögliche Fenstergröße ist 1 Zeichen breit und 1 Zeichen hoch.

Vor dem Beschreiben des Bildschirms wird die Schreibposition zwangsweise in den Fenster-Bereich gelegt (siehe Abschnitt 4.5). Dies kann ein Scrollen (Rollen) des Fensters zur Folge haben. Andere Operationen, wie z. B. die Ausführung von Steuer-Codes, verursachen ebenfalls ein Hineinzwingen der Schreibposition in den Fenster-Bereich.

Ein Text-Fenster, das nicht den ganzen Bildschirm überstreicht, wird von der Firmware gescrollt, indem Bereiche des Bildschirm-Speichers umkopiert werden. Es gibt hierfür keine andere Methode. Das Scrollen großer Fenster nimmt dadurch sehr viel Zeit in Anspruch.

Ein den gesamten Bildschirm überstreichendes Fenster wird durch die Hardware und nicht durch das Kopieren von Speicher-Bereichen gescrollt. Der Offset des Bildschirm-Anfangs kann im Bildschirm-Speicher festgelegt werden (siehe Abschnitt 6.4). Durch Änderung dieses Offsets um +80 oder −80 kann der gesamte Bildschirm um eine Zeichen-zeile auf- oder abgescrollt werden.

Es ist offensichtlich eine gute Idee, Fenster vor dem Überlappen zu schützen. Wäre eine Überlappung möglich, so würden die mehrfach verwendeten Bestandteile lediglich das zuletzt Geschriebene enthalten. Eine Bevorzugung gewisser Fenster existiert nicht. Ein den gesamten Bildschirm einnehmendes Fenster würde die anderen Fenster überlappen und beim Scrollen den Inhalt der anderen Fenster ebenfalls bewegen. Die beim EMS und nach einer Änderung des Bildschirm-Modus eingestellten Standard-Fenster überstreichen den gesamten Bildschirm.

4.5 Momentane Position und Cursor

Jeder Kanal besitzt eine ihm zugeordnete momentane Position. Diese befindet sich dort, wo das nächste auf dem Bildschirm darzustellende Zeichen erscheinen soll. Befindet sich die momentane Position zum Zeitpunkt einer anstehenden Zeichendarstellung außerhalb des Text-Fensters, so wird sie in das Text-Fenster hineingezwungen. Es werden die folgenden Aktionen eingeleitet, um die momentane Position in das Text-Fenster hineinzubringen:

1. Die momentane Position wird an die rechte Fensterbegrenzung und eine Zeile aufwärts bewegt, wenn sie sich links von der linken Fensterbegrenzung befindet.

2. Die momentane Position wird an die linke Fensterbegrenzung und eine Zeile abwärts bewegt, wenn sie sich rechts von der rechten Fensterbegrenzung befindet.

3. Die momentane Position wird auf die oberste Zeile des Fensters bewegt und der Inhalt des Fensters um eine Zeile abwärts gerollt, wenn sie sich oberhalb der obersten Zeile des Fensters befindet.

4. Die momentane Position wird auf die unterste Zeile des Fensters bewegt und der Inhalt des Fensters um eine Zeile aufwärts gerollt, wenn sie sich unterhalb der untersten Zeile des Fensters befindet.

Wenn der Cursor freigeschaltet ist, wird die momentane Position mit dem Cursor-Symbol gekennzeichnet. Vor der Plazierung des Cursor-Symbols auf dem Bildschirm wird jedoch die momentane Position in das aktuelle Fenster hineingezwungen. Dies kann ein Verschieben der momentanen Position zur Folge haben.

Ist der Cursor nicht freigeschaltet, so darf die momentane Position außerhalb des Fensters liegen. Sie wird solange nicht in das Fenster hineingebracht, bis beispielsweise ein Zeichen dargestellt wird.

Die momentane Position kann direkt (durch Aufruf von TXT SET CURSOR, TXT SET ROW oder TXT SET COLUMN) oder durch Übergabe von Steuer-Codes an den Text-VDU geändert werden. Der Platz, an den die momentane Position bewegt wird, wird nicht unmittelbar durch Hineinzwingen in das Fenster erreicht, sondern nur dann, wenn in das Fenster etwas hineingeschrieben wird (siehe oben). Hierdurch kann gegebenenfalls die Änderung der momentanen Position durch Bewegen über eine außerhalb des Fensters liegende Position erreicht werden.

Es bestehen zwei Möglichkeiten, den Cursor abzuschalten und das Erscheinen des Cursor-Symbols auf dem Bildschirm zu unterdrücken. Die Erste –Cursor Ein/Aus– ist für die Verwendung durch System-Programme vorgesehen. Sie wird zum Beispiel durch BASIC angewandt, um den Cursor auszuschalten, wenn eine Eingabe erwartet wird. Die Zweite –Cursor freischalten/sperren– ist für die Anwendung durch den Benutzer vorgesehen. Das Cursor-Symbol erscheint nur dann auf dem Bildschirm, wenn es freigeschaltet ist und sich im Zustand Ein befindet.

Das Cursor-Symbol ist normalerweise in Invers-Darstellung sichtbar. Das Zeichen auf der Cursor-Position ist in den entgegengesetzten Text-Pen- und Paper-Inks dargestellt. Dies erleichtert die Wiederherstellung der Originalform der Zeichen-Position nach Cursor-Verschiebungen. Der Benutzer hat die Möglichkeit, die Form des Cursor-Symbols durch TXT DRAW CURSOR und TXT UNDRAW CURSOR zu verändern.

4.6 Zeichen und Matrizen

Ein Zeichen wird in einem Bereich von 8 Pixel Breite und 8 Pixel Höhe auf dem Bildschirm dargestellt. Dadurch hängt die maximale Anzahl der Zeichen auf dem Bildschirm vom Bildschirm-Modus ab (siehe Abschnitt 6.1). Jedes Zeichen hat eine Matrix (ein 8-Byte-Vektor), die die Form des Zeichens bestimmt. Das erste Byte des Vektors bezieht sich auf die oberste Zeile des Zeichens, das letzte Byte des Vektors auf die unterste Zeile des Zeichens. Das höchstwertige Bit eines Vektor-Bytes bezieht sich auf das am weitesten links befindliche Pixel einer Zeile des Zeichens. Das Pixel befindet sich im Vordergrund, wenn ein Bit in der Matrix gesetzt, und im Hintergrund, wenn ein Bit in der Matrix gelöscht ist.

Ein Vordergrund-Pixel des Zeichens ist immer auf die Pen-Ink gesetzt. Die Behandlung eines Hintergrund-Pixels hängt von dem Zeichen-Schreibmodus des VDU ab. Im Standard-Modus, dem Überschreibe-Modus, sind die Hintergrund-Pixel auf die Paper-Ink gesetzt. Es gibt einen weiteren Modus, den Transparent-Modus, in dem die Hintergrund-Pixel nicht verändert werden. Auf diese Weise (im Transparent-Modus) werden die Zeichen über das obere Ende des momentanen Bildschirminhalts hinausgeschrieben. Dies ist hilfreich bei Kommentaren in Bildern oder zur Erzeugung von zusammengesetzten Zeichen.

Der Text-VDU kann 256 unterschiedliche Zeichen darstellen. Zur Darstellung der ersten 32 Zeichen, die gewöhnlich als Steuer-Codes interpretiert werden, sind besondere Anstrengungen vonnöten. Die Matrizen für die Zeichen befinden sich normalerweise im ROM. Der Anwender kann jedoch die Matrizen für eine beliebige Anzahl Zeichen im RAM speichern, wo sie dann geändert werden können. Die Standard-Einstellung beim EMS ist die Speicherung aller Matrizen im ROM (BASIC sieht während der eigenen Initialisierung die Erzeugung von 16 Benutzer-definierten Matrizen vor). Der Standard-Zeichensatz ist in Anhang VI beschreiben.

Wenn der Anwender durch Aufruf von TXT SET MTABLE eine Tabelle mit Benutzerdefinierten Matrizen erstellt, wird sie mit der momentanen Matrizen-Einstellung aus dem ROM oder RAM initialisiert. D. h. eine Erweiterung der Tabelle ändert die momentanen Matrizen nicht. Durch Verkürzen der Tabelle gehen die Zeichen verloren und es wird auf die Standard-Matrizen im ROM Rückgriff genommen.

Beim Lesen der Zeichen vom Bildschirm (durch Aufruf von TXT RD CHAR) werden die Pixel auf dem Bildschirm in die Form einer Matrix übersetzt. Diese wird mit den momentanen Zeichen-Matrizen verglichen, um herauszufinden, welches Zeichen es ist. Dies bedeutet, daß eine Änderung der Zeichen-Matrizen oder des Bildschirms ein Zeichen unkenntlich machen kann. Insbesondere kann die Änderung der Pen- oder Paper-Ink Verwirrung stiften. Gewöhnlich führen diese Probleme dazu, daß diese Zeichen als Leerzeichen (Zeichen #20) erscheinen. Damit sind nun besondere Vorsichtsmaßnahmen getroffen, um die Erzeugung von Leerzeichen zu verhindern.

Die Art und Weise, wie Zeichen auf den Bildschirm geschrieben bzw. vom Bildschirm gelesen werden, kann durch TXT WRITE CHAR und TXT UNWRITE geändert werden.

4.7 Zeichen-Ausgabe und Steuer-Codes

Die Haupt-Routine für Zeichenausgabe des Text-VDU ist TXT OUTPUT. Sie reagiert auf Steuer-Codes (Zeichen 0...31) und stellt alle anderen Zeichen dar. An TXT OUTPUT übergebene Zeichen durchlaufen verschiedene Bearbeitungsebenen und können durch verschiedene Ausgabe-Routinen behandelt werden.

TXT OUTPUT verwendet TXT OUT ACTION, um herauszufinden, ob das Zeichen ein darzustellendes Zeichen, ein Steuer-Code oder ein Parameter eines Steuer-Codes ist.

TEXT OUT ACTION wiederum ruft TXT WRITE CHAR auf, um Zeichen auf den Bildschirm auszugeben. Wenn jedoch der Graphikzeichen-Schreib-Modus selektiert ist, werden die Zeichen unter Verwendung der Zeichen-Schreibroutine des Graphik-VDU ausgegeben (siehe Abschnitt 5.6).

Dieser Modus kann auf einer Zeichen-für-Zeichen-Basis unter Verwendung eines Steuer-Codes oder als Block („Alle Zeichen übertragen") selektiert werden (siehe TXT SET GRAPHIC). Wenn der Graphikzeichen-Schreib-Modus ausgewählt ist, werden die Steuer-Codes nicht ausgeführt, sondern stattdessen durch die Graphik-Routine dargestellt.

TXT OUT ACTION bearbeitet Steuer-Codes folgendermaßen:

1. Der Code wird am Anfang des Steuer-Code-Puffers gespeichert.

2. Der Code wird in der Steuer-Code-Tabelle aufgesucht, um die Anzahl der erforderlichen Parameter zu bestimmen.

3. Wenn keine Parameter erforderlich sind, entfällt Aktion 4.

4. Wenn ein oder mehrere Parameter erforderlich sind, wird aus TXT OUT ACTION herausgesprungen. Das nächste übergebene Zeichen wird dann jedoch nicht ausgeführt oder dargestellt, sondern dem Steuer-Code-Puffer zugeführt. Dies wird fortgesetzt, bis ausreichend viele Parameter-Zeichen empfangen worden sind.

5. Der Code wird in der Steuer-Code-Tabelle aufgesucht, um die für die Bearbeitung des Steuer-Code relevante Adresse der aufzurufenden Routine herauszufinden. Diese Routine wird dann ausgeführt.

6. Der Steuer-Code-Puffer wird gelöscht und das nächste übergebene Zeichen kann dargestellt bzw. als Anlaß für den Start einer neuen Steuer-Code-Sequenz genommen werden.

Der Anwender kann die Funktionsweise des Steuer-Codes durch Änderung des Eintrags in der Steuer-Code-Tabelle beeinflussen (siehe TXT GET CONTROLS). Diese beinhaltet für jeden Code einen 3-Byte-Eintrag in aufsteigender Ordnung (d. h. den Eintrag für #00 zuerst, für #01 als nächsten usw.).

Das erste Byte eines Eintrags spezifiziert die Anzahl der erforderlichen Parameter. Diese müssen in dem Bereich von 0 bis 9 liegen, da der Steuer-Code-Puffer nur bis zu 9 Parameter speichern kann.

Das zweite und dritte Byte ist die für die Bearbeitung des Steuer-Code maßgebliche Adresse der aufzurufenden Routine. Diese Routine sollte in den zentralen 32KB RAM liegen. Sie sollte weiterhin die folgenden Einsprung-/Aussprung-Bedingungen erfüllen:

Einsprung:

A beinhaltet das letzte zum Puffer addierte Zeichen.

B beinhaltet die Anzahl der im Puffer befindlichen Zeichen (einschl. Steuer-Code).

C beinhaltet das gleiche wie A.

HL beinhaltet die Adresse des Steuer-Code-Puffers (zeigt auf den Steuer-Code).

Aussprung:
AF, BC, DE und HL zerstört.
Alle anderen Register unverändert.

Der Steuer-Code-Puffer ist zwischen den Kanälen aufgeteilt. Eine Steuer-Code-Sequenz sollte vor der Änderung der Kanäle abgeschlossen sein, da andernfalls unerwartete Reaktionen auftreten können.

Die Standard-Funktionen des Steuer-Code – nach dem EMS und nach Aufruf von TXT RESET – sind im Anhang VII beschrieben.

Durch den Aufruf von TXT VDU DISABLE kann ein Text-Kanal gesperrt werden, wodurch keine Zeichen mehr auf dem Bildschirm dargestellt werden können. Die normale Betriebsart kann durch Aufruf von TXT VDU ENABLE wiederhergestellt werden. Es sollte jedoch beachtet werden, daß der Aufruf dieser Routine den Steuer-Code-Puffer leert. Dieser Effekt schützt vor Problemen, wenn der Zustand des Steuer-Code-Puffers nicht bekannt ist (z. B. beim Ausgeben einer Fehlermeldung).

5 Graphik-VDU

Der Graphik-VDU ermöglicht das Setzen oder Testen bestimmter Pixel und das Zeichnen von Linien auf dem Bildschirm. Die Darstellungen erfolgen auf einem idealen Bildschirm, der immer 640 Bildpunkte breit und 400 Bildpunkte hoch ist. Dies bedeutet, daß mehr als ein Bildpunkt des idealen Bildschirms mit einem bestimmten Pixel des realen Bildschirms zusammenfallen können. Die Breite des idealen Bildschirms (640 Punkte) ist die Anzahl der horizontalen Pixel des Bildschirms in dem Modus mit der höchsten Auflösung (Modus 2). Die Höhe des idealen Bildschirms (400 Punkte) ist in allen Modi die doppelte Anzahl der vertikalen Pixel des Bildschirms. Dadurch ist sichergestellt, daß das Seitenverhältnis des Bildschirms annäherungsweise einheitlich ist, d. h. ein dargestellter Kreis ist rund und nicht elliptisch.

5.1 Koordinaten-Systeme des Graphik-VDU

Der Graphik-VDU verwendet 4 Koordinaten-Systeme. Der Anwender spezifiziert Positionen in Anwender-Koordinaten, relativen Koordinaten oder Standard-Koordinaten. Der Graphik-VDU verwendet intern Basis-Koordinaten (oder gelegentlich Standard-Koordinaten).

Anwender-Koordinaten, relative Koordinaten und Standard-Koordinaten ähneln sich sehr stark. Sie alle verwenden vorzeichenbehaftete 16-Bit-Zahlen und arbeiten mit X-Koordinaten von links nach rechts und Y-Koordinaten von unten nach oben. Der Bildschirm ist in jedem Bildschirm-Modus 400 Punkte hoch und 640 Punkte breit. Dies bedeutet, daß ein Pixel auf dem Bildschirm 8 Punkte in Modus 0, 4 Punkte in Modus 1 und 2 Punkte in Modus 2 belegt. Der Ursprung (Koordinaten (0,0)) dieses Systems variiert:

In Standard-Koordinaten befindet sich der Ursprung in der untersten linken Ecke des Bildschirms.

Der Ursprung der Anwender-Koordinaten kann durch den Benutzer festgelegt werden. Der Standard-Ursprung ist in der untersten linken Ecke des Bildschirms. Dadurch werden die Standard-Anwender-Koordinaten zu Standard-Koordinaten.

Der Ursprung der relativen Koordinaten ist die momentane Position (siehe Abschnitt 5.2). Dadurch können Darstellungen unabhängig von der Position auf dem Bildschirm ausgeführt werden. Nützlich ist dies auch, wenn bestimmte Figuren mehrfach auf dem Bildschirm wiederholt werden müssen oder wenn es schwierig ist, ständig den momentanen Standpunkt zu verfolgen.

Basis-Koordinaten sind Bestandteil eines physikalischen Koordinaten-Systems, das mit Bildpunkten arbeitet. X-Koordinaten verlaufen von links nach rechts, Y-Koordinaten von unten nach oben. Pixel (0,0) befindet sich in der untersten linken Ecke des Bildschirms. Da dieses Koordinaten-System mit Pixeln arbeitet, hängen die Positions-Koordinaten auf dem Bildschirm vom Bildschirm-Modus ab. Basis-Koordinaten sind vorzeichenlose 16-Bit-Zahlen, und nur Koordinaten, die sich auf einen Pixel im Bildschirmbereich beziehen, sind gültig.

Graphik-Routinen übersetzen wenn nötig relative Koordinaten in Anwender-Koordinaten, und anschließend – vor dem Zugriff auf den physikalischen Bildschirm – die Anwender-Koordinaten in Basis-Koordinaten. Diese Übersetzung führt zu einer Einbuße an Genauigkeit, da ein einzelner Pixel mit mehreren Bildpunkten belegt sein kann. Dadurch können auf dem Bildschirm dargestellte Figuren unsymmetrisch werden (insbesondere Kreise). Der Graphik-VDU vermeidet diesen Effekt, indem er die Koordinaten-Werte, bezogen auf den Ursprung, auf- oder abrundet. Um die Vorteile dieser Rundung auszunutzen, sollten symmetrische Figuren auch symmetrisch zum Anwender-Ursprung dargestellt werden. Ist die Figur nicht auf den Anwender-Ursprung zentriert, erscheint die Figur asymmetrisch.

5.2 Momentane Graphik-Position

Der Graphik-VDU speichert eine momentane Graphik-Position. Dies ist die Anwender-Koordinate des letzten durch den Graphik-VDU spezifizierten Punktes (oder der Ursprung nach dem Löschen des Graphik-Fensters). Der Ursprung der relativen Koordinaten ist in diesem Punkt spezifiziert, wodurch relative Koordinaten zu einem Offset, bezogen auf die momentane Position, werden.
Beim Zeichnen einer Linie befindet sich ein Endpunkt auf der spezifizierten Position, während sich der andere Endpunkt auf der momentanen Graphik-Position befindet. Beim Zeichnen eines Zeichens unter Verwendung der Schreib-Routine für Graphik-Zeichen wird das Zeichen mit der momentanen Graphik-Position in der obersten linken Ecke derselben dargestellt.

Nach dem Darstellen oder Testen eines Punktes bzw. dem Zeichnen einer Linie wird die momentane Graphik-Position zur spezifizierten Position bewegt. Nach dem Schreiben eines Zeichens wird die momentane Graphik-Position um eine Zeichenbreite nach rechts verschoben, damit das nächste zu schreibende Zeichen vorbereitet werden kann.

5.3 Inks für Graphik -Pen und -Paper

Der Graphik-VDU besitzt eine Pen (Vordergrund)- und eine Paper (Hintergrund)-Ink. Die Graphik-Pen-Ink dient dem Darstellen von Pixeln, Zeichnen von Linien und, beim Schreiben von Zeichen unter Verwendung der Graphik-Schreib-Routine (siehe Abschnitt 5.6), dem Setzen von Vordergrund-Pixeln. Die Graphik-Paper-Ink dient dem Löschen des Graphik-Fensters und, beim Schreiben von Zeichen unter Verwendung der Graphik-Schreib-Routine, dem Setzen von Hintergrund-Pixeln.

Pen und Paper können auf jede im momentanen Bildschirm-Modus gültige Ink eingestellt werden (siehe Abschnitt 6.2). Standardmäßig ist Paper auf Ink 0 und Pen auf Ink 1 eingestellt. Die Änderung einer Pen- oder Paper-Ink verändert nicht den Bildschirm direkt, sondern lediglich die zukünftige Darstellung der Pixel.

5.4 Graphik-Schreib-Modus

Durch den Graphik-VDU dargestellte Pixel werden im momentan gültigen Graphik-Schreib-Modus bearbeitet. Dieser spezifiziert, wie die darstellende Ink mit der eingestellten Ink eines Pixels zusammenwirken muß.

Es gibt vier Schreib-Modi:

0:	FORCE:	NEW = INK
1:	EXCLUSIVE-OR:	NEW = INK xor OLD
2:	AND:	NEW = INK and OLD
3:	OR:	NEW = INK or OLD

NEW ist die Ink, auf die die Pixel gesetzt werden.
OLD ist die Ink, auf die die Pixel momentan gesetzt sind.
INK ist die Ink, mit der dargestellt wird.

Der Standard-Modus ist der FORCE-Modus. Der Text-VDU setzt die Pixel immer, als würde er in diesem Modus arbeiten. Auch das Graphik-Fenster wird durch Schreiben im FORCE-Modus, unabhängig vom aktuellen Schreib-Modus, gelöscht.

Unter der Voraussetzung, daß die entsprechenden Ink-Einstellungen erfolgt sind, können durch den AND- und OR-Modus bestimmte Bits eines Pixels gelöscht oder gesetzt werden. Dies ermöglicht dem Graphik-VDU das Schreiben in „Bit-Ebenen". Durch die Auswahl entsprechender Ink-Farben können überlappende Figuren gezeichnet und automatisch miteinander verschachtelt werden.

Der EXCLUSIVE-OR-Modus kann durch Auswahl geeigneter Inks verwendet werden, um einen momentan gültigen Bildschirminhalt zu übermalen. Dies ist sehr hilfreich, da eine Figur durch erneutes Zeichnen vom Bildschirm entfernt werden kann. Eine zweimalige EXCLUSIVE-OR-Verknüpfung mit der gleichen Ink stellt den Original-Zustand eines Pixels wieder her.

Der Graphik-Schreib-Modus kann durch Aufruf von SCR ACCESS oder mittels Steuer-Code (siehe Anhang VII) eingestellt werden.

5.5 Graphik-Fenster

Mit dem Graphik-VDU kann ein einzelnes Fenster spezifiziert werden. Dadurch ist dem Anwender die Möglichkeit gegeben, Text und Graphik ohne gegenseitige Beeinflussung auf dem Bildschirm zu mischen. Wenn die Text-Fenster mit dem Graphik-Fenster überlappen dürfen, wird beim Scrollen der Text-Fenster auch der Inhalt des Graphik-Fensters bewegt. Das Graphik-Fenster selbst kann nicht gescrollt werden.

Beim Darstellen von Punkten, Zeichnen von Linien oder Schreiben von Zeichen kann zu keiner Zeit ein Pixel außerhalb des Graphik-Fensters geschrieben werden. Im Gegensatz zu Text-Fenstern wird nichts unternommen, um einen Punkt in das Fenster hineinzuzwingen. Alle außerhalb des Fensters vorgenommenen Aktionen bleiben wirkungslos. Umgekehrt gilt, daß beim Testen von Punkten alle Punkte außerhalb des Fensters so bewertet werden, als wären sie auf die momentan gültige Graphik-Paper-Ink eingestellt. Punkte innerhalb des Fensters können korrekt geschrieben und gelesen werden.

Das Graphik-Fenster nimmt Bezug auf einen spezifischen Bildschirm-Bereich und nicht auf das Anwender-Koordinaten-System. Dadurch hat eine Verschiebung des Ursprungs im Anwender-Koordinaten-System keinen Einfluß auf die Lage des Bildschirm-Fensters, obgleich die Anwender-Koordinaten der Punkte innerhalb des Fensters geändert werden.

Das beim EMS und nach Änderung des Bildschirm-Modus erzeugte Standard-Graphik-Fenster überstreicht den gesamten Bildschirm.

5.6 Schreiben von Zeichen

Die Zeichen-Schreib-Routine des Graphik-VDU schreibt ein Zeichen mit der momentanen Graphik-Position in der obersten linken Ecke des Zeichens. Die momentane Position wird um eine Zeichenbreite nach rechts bewegt. Der dadurch erzielte Abstand variiert; in Modus 0 beträgt er 32 Punkte; in Modus 1, 16 Punkte; und in Modus 2, 8 Punkte. Steuer-Codes (Zeichen 0...31) werden dargestellt und nicht ausgeführt.

Das Zeichen wird – unabhängig vom dem durch den Text-VDU zum Schreiben von Zeichen verwendeten Modus – immer undurchsichtig geschrieben. Der Zeichen-Hintergrund ist auf die Graphik-Paper-Ink, der Vordergrund auf die Graphik-Pen-Ink eingestellt.

Zum Darstellen der Pixel des Zeichens wird jedoch der momentane Graphik-Schreib-Modus herangezogen (siehe Abschnitt 5.4).

6 Bildschirm-Paket

Das Bildschirm-Paket wird durch den Text- und Graphik-VDU aufgerufen, um auf die Hardware des Bildschirms zuzugreifen. Es steuert ebenso die Eigenschaften des Bildschirms, die den Text- und Graphik-VDU betreffen, wie z. B. den Bildschirm-Modus.

6.1 Bildschirm-Modi

Der Bildschirm kennt drei Betriebsarten, Modus 0, 1 und 2. Diese Modi haben verschiedene Auflösungen und stellen eine unterschiedliche Anzahl Inks zur Verfügung.

Alle Betriebsarten haben eine vertikale Auflösung von 200 Bildpunkten (Bildelemente oder Punkte auf dem Bildschirm). Die horizontale Auflösung variiert zwischen 160 und 640 Bildpunkten. Da jedes Zeichen ein Format von 8 x 8 Pixel hat, variiert die über den Bildschirm darstellbare Anzahl Zeichen mit der Betriebsart (von 20 bis 80 Zeichen). Der Bildschirm ist jedoch immer 25 Zeichen hoch.

Die Anzahl der auf dem Bildschirm darstellbaren Inks hängt von der Bildschirm-Auflösung ab. Wenn der Bildschirm 640 Bildpunkte breit ist, können nur 2 Inks dargestellt werden. Bei 320 Bildpunkten sind es 4 Inks, bei 160 Bildpunkten bereits 16 Inks.

Zusammenfassend gilt:

Modus	Anzahl Pixel	Anzahl Zeichen	Inks
0	160 x 200	20 x 25	16
1	320 x 200	40 x 25	4
2	640 x 200	80 x 25	2

Der Standard-Bildschirm-Modus beim EMS ist Modus 1. Der Bildschirm-Modus kann durch Aufruf von SCR SET MODE eingestellt werden. Dies hat noch weitere Effekte.

Erstens wird dadurch der Bildschirm gelöscht und auf Ink 0 gesetzt. Sollten die Text- und Graphik-Paper-Inks nicht auf Ink 0 gesetzt sein, so wird dies beim Schreiben von Zeichen oder Löschen der Fenster sichtbar. Wenn der Anwender aus irgendwelchen Gründen diese Betriebsart verändern möchte, kann dies mit SCR MODE CLEAR erfolgen.

Zweitens wird der Text- und Graphik-VDU in den Grundzustand versetzt. Die Fenster sind alle so eingestellt, daß sie den gesamten Bildschirm einnehmen. Wenn die Pen- und Paper-Inks im neuen Modus nicht definiert sind, werden sie mit # 01 oder # 03 maskiert. Die momentanen Text-Positionen werden in die oberste linke Ecke des Bildschirms bewegt und der Text-Cursor ausgeschaltet (siehe TXT CUR OFF). Die momentane Graphik-Position und der Anwender-Ursprung werden in die unterste linke Ecke des Bildschirms bewegt.

6.2 Inks und Farben

Die verschiedenen Bildschirm-Modi ermöglichen die Einstellung der Pixel auf eine unterschiedliche Anzahl Inks wie folgt:

Modus 0:	16 Inks (0...15)
Modus 1:	4 Inks (0... 3)
Modus 2:	2 Inks (0... 1)

Wie die Inks für einen Pixel innerhalb eines Bytes codiert sind, ist in Abschnitt 6.4 beschrieben. Die Einstellung der Pixel-Inks bestimmt die Farbe, in der diese Pixel dargestellt werden. Die einer Ink zugeordnete Farbe ist jedoch nicht konstant, sondern kann geändert werden.

Es stehen 27 Farben zur Verfügung. Jede Ink kann auf eine dieser Farben eingestellt werden. Der Bildschirmrand besitzt Ink-Eigenschaften, deren Farbe einstellbar ist. Die Bildschirm-Hardware holt sich vor der Darstellung jedes Pixels den Ink-Wert aus dem Bildschirm-Speicher. Dieser Ink-Wert wird zum Zugriff auf einen kleinen RAM-Bereich innerhalb des Gate-Arrays – die sogenannte „Palette" – verwendet. Die Palette beinhaltet die aktuelle, für diese bestimmte Ink maßgebliche Farbe, in der auf dem Monitor dargestellt wird. Die Änderung der Einträge in dieser Palette bewirkt, daß alle auf diese Ink eingestellten Pixel bei der nächsten Darstellung ihre Farbe wechseln (ungefähr innerhalb 20 msec.).

Durch das Bildschirm-Paket können zwei Farben einer Ink (oder dem Bildschirmrand) zugeordnet werden. Diese werden nacheinander und Software gesteuert in die Palette geladen. Wenn die beiden der Ink zugeordneten Farben unterschiedlich sind, blinkt die Ink. Sind die Farben gleich, so bleibt die Ink konstant. Der Benutzer kann die standardmäßig eingestellte Frequenz von 5 Zyklen pro Sekunde gegebenenfalls durch SCR SET FLASHING verändern.

Bei der Farbenfestlegung verwendet das Bildschirm-Paket eine der Grauskala eines monochromatischen Monitors vergleichbare Anordnung. Diese erstreckt sich von der dunkelsten Farbe 0 (schwarz) bis zur hellsten Farbe 26 (helles weiß). Die Farben erscheinen in der Darstellung auf einem Farb-Monitor so, als unterlägen sie keiner bestimmten Ordnung.

Die Palette arbeitet mit einem andersgearteten (und nicht wahrnehmbaren) Numerierungs-Schema für die Farben. Das Bildschirm-Paket übersetzt die Nummern der Grau-Skala automatisch in die Hardware-Nummern und umgekehrt. Mit den Hardware-Nummern wird der Anwender nur dann konfrontiert, wenn er die Hardware direkt ansteuern will.

Die Standard-Einstellung für die Ink-Farben und eine Liste der 27 verfügbaren Farben befinden sich in Anhang V.

6.3 Bildschirm-Adressen

Das Bildschirm-Paket verwendet selbst kein Koordinaten-System sondern nur Bildschirm-Adressen. Es arbeitet jedoch mit dem physikalischen und dem Basis-Koordinaten-System des Text- und Graphik-VDU (siehe Abschnitt 4.1 bzw. 5.1). Zur Umwandlung von Positionen in physikalischen oder Basis-Koordinaten in die entsprechenden Bildschirm-Adressen werden Routinen bereitgestellt.

Eine Bildschirm-Adresse ist die Adresse eines Bytes innerhalb des Bildschirm-Speichers. Zur Bestimmung eines bestimmten Pixels wird häufig eine Bildschirm-Adresse gemeinsam mit einer Maske, die exakt den benötigten Pixel anzeigt, an eine Routine übergeben. Weitere Routinen werden zur Verfügung gestellt, um die Bildschirm-Adresse eines Bytes rechts, links, über oder unter der momentanen zu bestimmen (was bei dieser Bildschirm-Belegung nicht einfach ist).

6.4 Bildschirm-Speicher-Belegung

Der Bildschirm-Speicher ist pixel-orientiert und nimmt in allen Betriebsarten 16KB RAM in Anspruch. Standardmäßig liegt der 16 KB-Bildschirm-Speicher nach einem EMS ab Adresse # C000 aufwärts.

Dieser Bereich liegt unter dem oberen ROM, sofern freigeschaltet, wodurch der Bildschirm-Speicher separat vom übrigen System liegt. Dies bedeutet jedoch, daß das obere ROM beim Lesen des Bildschirm-Speichers gesperrt werden muß. Die Firmware-Sprungtabelle schaltet das obere ROM durch LOW JUMP-Restart ab, um auf den Bildschirm-Speicher zuzugreifen.

Der Bildschirm-Speicher kann auf jeden der vier 16KB-Speicherblöcke in 16KB-Schritten gelegt werden (siehe SCR SET BASE). Nur die Adressen #C000 und #4000 sind jedoch sinnvoll, da die Adressen #0000 und #8000 die Firmware-Sprungtabelle oder andere System-Bereiche überlappen. Die folgende Beschreibung geht von der Standard-Bildschirm-Belegung ab Adresse #C000 aus.

Die Bildschirm-Speicher-Belegung ist recht kompliziert. Glücklicherweise ist es nicht erforderlich, sie zu verstehen, da Text- und Graphik-VDU ein idealisiertes Modell des Bildschirms bereitstellen. Für bestimmte Anwendungen (z. B. lebhafte Spiele) ist es jedoch zur Erreichung der maximalen Abarbeitsgeschwindigkeit erforderlich, auf dem Bildschirm-Speicher direkt zuzugreifen.

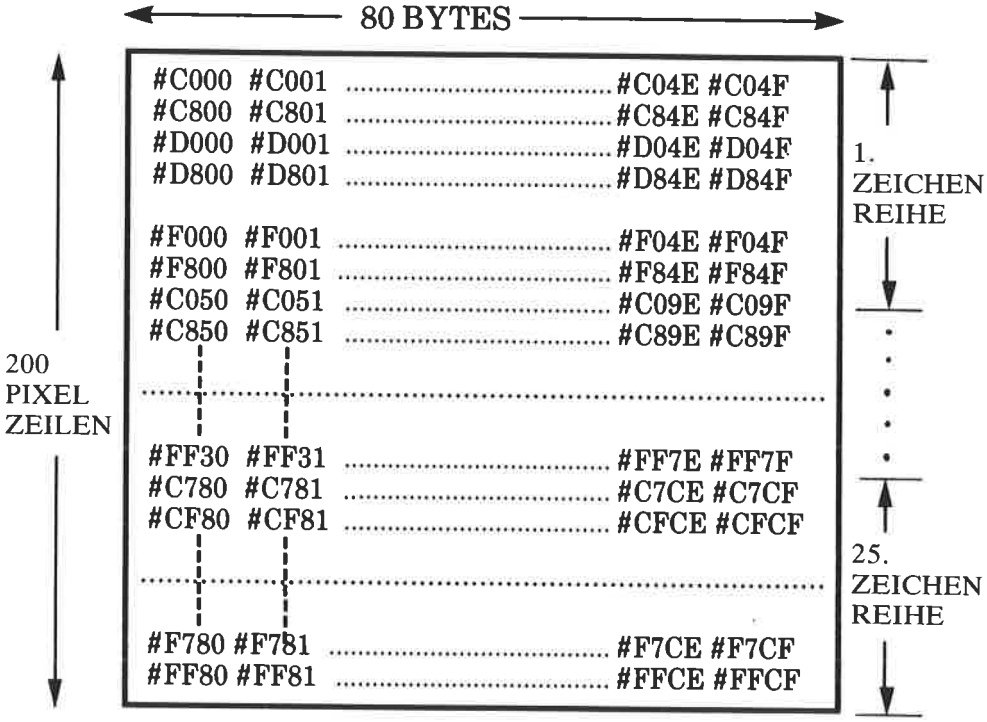
Der Bildschirm-Speicher ist in 8 Blöcke von jeweils 2KB unterteilt. Block 0 reicht von #C000 bis #C7FF, Block 1 von #C800 bis #CFFF usw. Jede Pixel-Zeile auf dem Bildschirm belegt 80 aufeinanderfolgende Bytes eines Blocks. Die oberste Bildschirmzeile kommt von Block 0, die zweite Zeile von Block 1 usw. bis zur achten Zeile, die von Block 7 kommt. Diese Sequenz beginnt erneut mit Block 0 auf der neunten Zeile und wiederholt sich auf diese Weise über den gesamten Bildschirm. Die aufeinanderfolgenden Zeilen in einem Block sind fortlaufend gespeichert, so daß am Ende jedes Blocks 48 unbenutzte Bytes verbleiben.

Es gibt eine weitere Komplikation in der Speicher-Belegung. Die obige Beschreibung geht davon aus, daß das erste aus diesem Block dargestellte Byte auch das erste Byte des Blocks ist. In der Praxis kann jedoch der Offset des ersten dargestellten Bytes in einem Block auf jeden geradzahligen Wert gesetzt werden (siehe SCR SET OFFSET). Dieser Offset ist dann für alle acht Blöcke gültig. Ein Block verläuft vom letzten Byte zu seinem ersten Byte, so daß #C7FE, #C7FF und #C000 aufeinanderfolgende Bytes in Block 0 sind und auf der gleichen Zeile des Bildschirms liegen können. Die Änderung des Offsets um +/− 80 MOD 2048 (die Länge einer Zeile) scrollt den Bildschirm um eine Zeichen-Zeile (9 Pixel-Zeilen) auf oder ab. Dieser Effekt wird durch den Text-VDU zum Scrollen des gesamten Bildschirms verwendet.

Die Bedeutung der oben beschriebenen Bytes verändert sich mit dem Bildschirm-Modus. Jedes Byte speichert die Inks für 2, 4 oder 8 Pixel. Die zur Codierung jedes Pixels verwendeten Bits sind in keiner bestimmten Art und Weise angeordnet. Die folgende Tabelle gibt an, welche Bits im Bildschirm-Speicher verwendet werden, um bestimmte Pixel in den verschiedenen Modi zu codieren. Die in der Tabelle angeführten Bit-Nummern bezeichnen die Bits eines Bildschirm-Bytes. Sie sind in der Reihenfolge der Bits in einem Pixel angegeben. Das erste Bit ist das höchstwertige Bit eines Pixels und das letzte Bit das niederwertigste.

	Modus 0	Modus 1	Modus 2
Linksbündiges Pixel	Bits 1,5,3,7	Bits 3,7	Bit 7
.			Bit 6
.		Bits 2,6	Bit 5
.			Bit 4
.	Bits 0,4,2,6	Bits 1,5	Bit 3
.			Bit 2
Rechtsbündiges		Bits 0,4	Bit 1
Pixel			Bit 0

Das folgende Diagramm zeigt die Belegung der Pixel auf dem Bildschirm mit den Adressen im Bildschirm-Speicher für den einfachsten Fall, d. h. mit der Basis-Adresse #C000 und einem Offset von 0.



#C7D0..#C7FF, #CFD0..#CFFF, ... , #FFD0..#FFFF nicht verwendet.

7 Tongenerator-Verwaltung

Die Tongenerator-Verwaltung bearbeitet den Tongenerator-Chip. Sie ermöglicht die Einstellung verschiedener Hüllkurven und Töne, sowie das Abspielen unter Benutzer-Steuerung. Den größten Steuerungsanteil hat die Software und nicht die Tongenerator-Hardware.

7.1 Tongenerator-Chip

Der Tongenerator-Chip ist ein AY-3-8912 von General Instruments. Er besitzt drei Kanäle und einen Pseudo-Zufalls-Rauschgenerator, der mit jedem der drei Kanäle verbunden werden kann. Der Chip stellt eine begrenzte Anzahl an Lautstärke-Hüllkurven (siehe Anhang IX) zur Verfügung. Die Software-Hüllkurven haben jedoch eine gleiche bis höhere Leistungsfähigkeit als die Hardware. Ton-Hüllkurven werden durch die Software erstellt; es gibt dafür keine Hardware-Unterstützung.

Die durch den Chip erzeugten Töne basieren auf Rechteck-Wellenformen. Es gibt keine Möglichkeit, andere Wellenformen zu erzeugen.

Wenn erforderlich, kann auf den Tongenerator-Chip direkt zugegriffen werden. Zum Beschreiben der Register des Tongenerator-Chips sollte jedoch die Routine MC SOUND REGISTER verwendet werden. Der Grund dafür ist, daß die Tastatur direkt mit dem Ein-/Ausgabe-Kanal des Tongenerator-Chips verbunden ist und die Abtast-Routine der Tastatur den Tongenerator-Chip in seinem Standard-Zustand (d. h. inaktiv) erwartet. Weiterhin bestehen Timing-Probleme beim Zugriff auf den Chip, deren Berücksichtigung durch Verwendung der Routine MC SOUND REGISTER vermieden werden kann.

Der Tongenerator-Chip besitzt drei unabhängige Ton-Kanäle. Ihre Ausgänge werden zur Bildung zweier Stereo-Kanäle gemischt. Die Ton-Kanäle A und B für einen Stereo-Kanal, B und C für den anderen. Der Stereo-Ton steht über eine Ausgangsbuchse auf der Rückseite des Geräts zur Verfügung. Da nur ein interner Lautsprecher vorhanden ist, werden die beiden Stereo-Kanäle nochmals gemischt, um ihn anzusteuern. Die Lautstärke des internen Lautsprechers kann durch den Laustärkeregler seitlich neben dem EIN/AUS-Schalter eingestellt werden. Der Lautstärkeregler hat gegenüber den weiter unten beschriebenen Lautstärke-Steuerungsmethoden Vorrang.

7.2 Ton-Perioden und Lautstärken

Der Tongenerator-Chip ermöglicht 16 unterschiedliche Lautstärke-Einstellungen im Bereich 0...15. Lautstärke 0 ist nicht hörbar und Lautstärke 15 ist das Maximum.

Die Tonhöhe einer zu erzeugenden Note wird besser durch die Periode, als durch die Frequenz spezifiziert. Diese Periode ist in 8 Mikrosekunden-Einheiten gegeben. Die Ton-Periode und -Frequenz wird dadurch folgendermaßen berechnet:

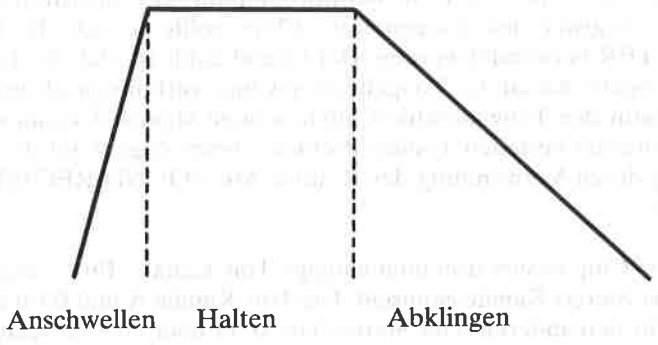
Ton-Periode = 125 000/Frequenz

Eine Auflistung der empfohlenen Perioden zur Erzeugung von Musiknoten befindet sich in Anhang VIII.

7.3 Hüllkurven

Echte Musik hat selten eine konstante Lautstärke. Die Hüllkurven ermöglichen eine Näherung an die Lautstärke-Variation natürlicher Töne. Die Hüllkurve ist in Abschnitte eingeteilt, wovon jeder mit ansteigender, abfallender oder konstanter Lautstärke versehen werden kann. Die Länge dieser Abschnitte ist variabel, ebenso die Steigerungs- oder Abschwächungs-Rate der Lautstärke. Eine durch ein Musikinstrument erzeugte Note ist – in 3 Abschnitte zerlegt – wie folgt vorstellbar:

Anschwellen:	Die Lautstärke einer Note steigt schnell auf ihren Spitzenwert.
Halten:	Die Lautstärke der Note bleibt während ihrer Spieldauer konstant.
Abklingen:	Die Lautstärke fällt nach dem Notenende langsam auf Null ab.



Die Tongenerator-Verwaltung kennt zwei Hüllkurven-Typen: Lautstärke-Hüllkurven zur Steuerung der Ton-Lautstärke und Ton-Hüllkurven zur Steuerung der Tonhöhe (die Tonhöhe wird ähnlich wie die Lautstärke variiert). Der Anwender kann bis zu 15 unterschiedliche Hüllkurven jedes Typs einstellen. Die genauen Formate der Datenblöcke zur Spezifikation von Hüllkurven sind durch SOUND AMPL ENVELOPE und SOUND TONE ENVELOPE gegeben.

a) Lautstärke-Hüllkurven

Die Lautstärke-Hüllkurve wird zur Steuerung der Lautstärke und Länge eines Tons verwendet. Sie kann bis zu 5 Abschnitte haben. Jeder Abschnitt kann ein Hardware- oder Software-Abschnitt sein. Software-Abschnitte sind entweder absolut oder relativ.

Hardware-Abschnitte schreiben Werte in die Tongenerator-Chip-Register 11, 12 und 13, um eine Hardware-Hüllkurve einzustellen (siehe Anhang IX für eine Beschreibung der Tongenerator-Chip-Register). Im allgemeinen folgt einem Hardware-Abschnitt ein Software-Abschnitt, der nichts anderes tut, als eine gewisse Zeit zu warten, bis die Hardware-Hüllkurve bearbeitet ist.

Ein absoluter Software-Abschnitt spezifiziert eine einzustellende Lautstärke und eine Wartezeit, bis zur Ausführung des nächsten Abschnitts.

Ein relativer Software-Abschnitt spezifiziert eine Schrittgröße, eine Schrittanzahl und eine Wartezeit. Für jeden angeforderten Schritt wird die momentane Lautstärke durch die angegebene Schrittgröße geändert. Nach jedem Schritt und vor Bearbeitung eines neuen Schrittes läßt die Tongenerator-Verwaltung die angegebene Wartezeit verstreichen.

Lautstärke-Hüllkurven werden durch Aufruf von SOUND AMPL ENVELOPE eingestellt.

b) Ton-Hüllkurven

Eine Ton-Hüllkurve steuert die Tonhöhe. Sie kann bis zu fünf Abschnitte haben. Jeder Abschnitt kann entweder absolut oder relativ sein. Die Abschnitte einer Ton-Hüllkurve sind nicht zwangsläufig mit denen der Lautstärke-Hüllkurve vergleichbar. Ein absoluter Abschnitt spezifiziert eine einzustellende Ton-Periode und eine Wartezeit bis zur Ausführung des nächsten Abschnitts.

Ein relativer Abschnitt spezifiziert eine Schrittgröße, eine Schrittanzahl und eine Wartezeit. Für jeden angeforderten Schritt wird die momentane Ton-Periode durch die angegebene Schrittgröße geändert. Nach jedem Schritt und vor Bearbeitung eines neuen Schrittes läßt die Tongenerator-Verwaltung die angegebene Wartezeit verstreichen.

Ist die Ton-Hüllkurve vor dem Ablauf der Tondauer abgeschlossen (siehe Abschnitt 7.4 f), so wird die letzte Tonhöhe konstant gehalten. Die Ton-Hüllkurven können alternativ auch so eingestellt werden, daß sie sich automatisch wiederholen. Hierdurch können Tremolo-Effekte erzeugt werden.

Ton-Hüllkurven werden durch Aufruf von SOUND TONE ENVELOPE eingestellt.

7.4 Ton-Befehle

Wenn durch Aufruf von SOUND QUEUE ein zu spielender Ton an die Tongenerator-Verwaltung übergeben wird, müssen eine Reihe zusätzlicher Informationen spezifiziert werden. Diese sind im folgenden beschrieben. Die genaue Auslegung eines Ton-Befehl-Datenblocks ist in SOUND QUEUE beschrieben.

a) Anfangs-Ton-Periode

Der Ton wird mit einer Anfangs-Ton-Periode ausgegeben. Die Tonhöhe kann ausgehend von diesem Anfangswert durch eine Ton-Hüllkurve variiert werden. Wenn keine Ton-Hüllkurve spezifiziert ist, bleibt die Tonhöhe konstant. Eine Anfangs-Ton-Periode von Null bedeutet, daß kein Ton erzeugt wird; vermutlich wird der Ton nur aus reinem Rauschen bestehen (siehe weiter unten).

b) Anfangs-Lautstärke

Der Ton wird mit einer Anfangs-Lautstärke ausgegeben. Die Lautstärke des Tons kann ausgehend von diesem Anfangswert durch eine Lautstärke-Hüllkurve variiert werden. Wenn keine Lautstärke-Hüllkurve spezifiziert ist, bleibt die Lautstärke konstant.

c) Ton-Hüllkurve

Sie spezifiziert die zu verwendende Ton-Hüllkurve. Wenn keine Hüllkurve spezifiziert ist, bleibt die Tonhöhe konstant.

d) Lautstärke-Hüllkurve

Sie spezifiziert die zu verwendende Lautstärke-Hüllkurve. Wenn keine Hüllkurve spezifiziert ist, wird die standardmäßige System-Hüllkurve verwendet. Sie hält die Ton-Lautstärke konstant und endet nach 2 Sekunden.

e) Rausch-Periode

Wenn die Rausch-Periode Null ist, wird dem Ton kein Rauschen beigemischt. Jeder andere Wert setzt die Periode für den Pseudo-Zufalls-Rauschgenerator und mischt dem erzeugten Ton Rauschen hinzu. Man beachte, daß es nur einen Rauschgenerator gibt. Wenn zwei Töne ihn gleichzeitig verwenden, müssen sie in der Periode übereinstimmen.

f) Ton-Dauer

Die Länge eines Tons kann auf zwei Arten spezifiziert werden, entweder als absolute Zeit (Ton-Dauer) oder in Vielfachen der Lautstärke-Hüllkurve. Im letzten Fall wird die Hüllkurve einmal oder mehrfach gestartet. Der Ton ist nach der Ausführung der angegebenen Anzahl Hüllkurven beendet. Endet die Ton-Dauer (absolute Zeit) vor der Hüllkurve, so wird der Ton zeitlich beschnitten. Wenn die Ton-Dauer länger als die Hüllkurve ist, wird die End-Lautstärke gehalten, bis die Ton-Dauer beendet ist.

g) Kanäle und Synchronisations-Bits

Der Ton kann auf einem oder mehreren Kanälen ausgegeben werden. Wenn der Ton auf mehr als einem Kanal ausgegeben wird, stellen sich diese Kanäle automatisch aufeinander ein. Die Bedingungen an diese gegenseitige Einstellung können auch gesondert gesetzt werden (Rendevous-Technik). Der Ton kann festgehalten oder die Ton-Warteschlange durchlaufen werden (siehe Abschnitt 7.6).

7.5 Ton-Warteschlangen

Jeder Kanal hat eine mit ihm verbundene Warteschlange (Queue). Jede Warteschlange besitzt Platz, um mindestens drei Töne zu speichern. Die Töne am Kopf jeder Warteschlange können zum Abspielen gebracht werden oder auf verschiedene Synchronisations-Bedingungen warten (siehe Abschnitt 7.6). Die Ausgabe eines Ton-Befehls führt dazu, daß der Ton in die durch den Befehl spezifizierte Warteschlange eingereiht wird. Wenn der Ton den Kopf der Warteschlange erreicht hat und die Synchronisations-Bedingungen erfüllt sind, wird er gespielt.

Wird ein Ton mit gesetztem Durchlauf-Bit in die Warteschlange eingereiht, so werden alle für diesen Kanal eingereihten Töne unterdrückt und die Ausführung dieser Töne unmittelbar gestoppt. Dadurch wird ein Ton mit gesetztem Durchlauf-Bit direkt zum Kopf der Warteschlange gebracht und mit der Ausführung begonnen.

Die Routine SOUND CHECK ist bereitgestellt, um den Status des Tons am Kopf der Warteschlange zu testen und den freien Platz in einer Warteschlange zu bestimmen. Durch Aufruf von SOUND ARM EVENT kann ein Ton-Ereignis für jede Warteschlange eingestellt werden. Dieses Ereignis wird angestoßen, wenn die Warteschlange Platz frei hat. Die Funktionsweise des Ton-Ereignisses ermöglicht die Bearbeitung der Ton-Erzeugung als Hintergrund-Aufgabe, während gleichzeitig andere Aufgaben bewältigt werden.

7.6 Synchronisation

Es gibt zwei Mechanismen, mit denen Töne auf unterschiedliche Kanälen synchronisiert werden können. Diese sind das Festhalten von Tönen und die gegenseitige Einstellung (Rendezvous). Der Zweck einer Synchronisation ist das gleichzeitige Starten von Tönen. Für die Simulation eines Instruments kann z. B. ein Kanal zur Erzeugung der fundamentalen Note und ein anderer Kanal zur Erzeugung der Harmonien dieser Note verwendet werden. Der Synchronisations-Mechanismus, insbesondere des Rendezvous sorgt dafür, daß die fundamentale Note und die Harmonien exakt gleichzeitig gestartet werden.

Ein Ton kann während seiner Ausgabe festgehalten werden, d. h. der Ton wird, wenn er den Kopf der Warteschlange erreicht hat, nicht unmittelbar ausgeführt. Stattdessen wartet er, bis er ausdrücklich (durch Aufruf von SOUND RELEASE) zum Start freigegeben wird.

Ein Ton kann bei seiner Ausgabe mit bestimmten Rendezvous-Bedingungen versehen werden. Bei der Ausgabe eines Tons auf mehrere Kanäle werden diese Kanäle automatisch auf bestimmte gegenseitige Einstellungen gesetzt. Ein mit einem Rendezvous versehener Ton wird beim Erreichen des Warteschlangen-Kopfes nicht unmittelbar ausgeführt. Stattdessen wartet er, bis Töne mit übereinstimmenden Rendezvous den Kopf ihrer Warteschlange erreicht haben. Nur wenn alle gegenseitigen Einstellungen der Töne anliegen und bereit zur Ausführung sind, wird gestartet.

Wenn beispielsweise ein Ton auf Kanal A mit einem Rendezvous für einen Ton von Kanal B versehen ist, startet er nicht eher, bis ein Ton auf Kanal B eintrifft, der mit einem Rendezvous für einen Ton von Kanal A versehen ist. Dies gilt auch umgekehrt. Wenn ein Ton auf Kanal B bereit ist, der nicht mit einem Rendezvous für Kanal A versehen ist, wird er ausgegeben. Der Ton auf Kanal A wartet jedoch weiterhin auf ein Rendezvous.

7.7 Festhalten von Tönen

Es besteht durch Aufruf von SOUND HOLD die Möglichkeit, einen Ton während seiner Ausführung anzuhalten. Dadurch wird ein Kanal gestoppt und der Zustand des Tons gesichert. Der Ton kann durch Aufruf von SOUND CONTINUE von seinem Stop-Zustand ausgehend erneut gestartet werden. Wenn beim Festhalten des Tons eine Hardware-Hüllkurve bearbeitet wurde, kann der auftretende Effekt beim erneuten Starten des Tons nicht vorhergesagt werden. Die Hardware-Hüllkurve kann oder kann nicht von ihrem Stop-Zustand ausgehend fortgesetzt werden.

Der Aufruf von SOUND HOLD unterscheidet sich vom Setzen des Halte-Bits während einer Ton-Ausgabe, wie er in Abschnitt 7.6 beschrieben ist. SOUND HOLD stoppt alle zu einer beliebigen Zeit gestarteten Töne, während die Halte-Bits eine Möglichkeit bieten, Töne miteinander zu synchronisieren und einen bestimmten Ton beim Erreichen des Warteschlangen-Kopfes vor dem Starten zu bewahren.

8 Kassetten-Verwaltung

Die Kassetten-Verwaltung liest Dateien vom und schreibt Dateien zum Band. Diese Funktion kann entweder Zeichen für Zeichen oder als gesamte Datei ausgeführt werden. Der eingebaute Kassetten-Mechanismus („Datacorder") ist komplett Software-gesteuert. Es gibt keine Hardware-Unterstützung für die Kassette, sogar das Timing für das Lesen oder Schreiben von Bits wird durch die Software bewerkstelligt.

Das Datenformat auf dem Band wird sehr detailliert beschrieben. Für die meisten Anwender ist diese Information sicher nur von theoretischem Interesse. Allgemeinere Informationen finden sich ab Abschnitt 8.4

8.1 Datei-Format

Eine auf dem Band gespeicherte Datei ist in Blöcke aufgeteilt, die jeweils mit einem Header-Record (Kopfsatz) und einem Daten-Record (Datensatz) mit bis zu 2KB Daten versehen sind. Der durch Software gesteuerte Kassetten-Motor wird zwischen den Datei-Blöcken abgeschaltet, um Zeit zur Verarbeitung der gelesenen Daten bzw. zur Erzeugung der Schreibdaten zu gewinnen. Die Motor-Startlücke dient zur Trennung der Blöcke.

Das allgemeine Format eines Blocks ist wie folgt:

Der erste und letzte Block einer Datei besitzt zur Trennung der Dateien auf dem Band jeweils vorher bzw. nachher eine Extra-Lücke. Ihr Format ist:

Erster Block:

Letzter Block:

Es besteht eine klare Trennung zwischen Kopf-Satz und Daten-Satz. Der Kopf-Satz ist unter Verwendung des Synchronisationszeichens #2C geschrieben, der Daten-Satz unter Verwendung von #16, d.h. die Kassetten-Verwaltung kann niemals fehlerhaft einen Daten-Satz als Kopf-Satz identifizieren und umgekehrt. Die Verwendung der Synchronisationszeichen ist in Abschnitt 8.2 beschrieben.

8.2 Satz-Format

Ein Satz kann jede beliebige Anzahl Datenbytes zwischen 1 und 65536 beinhalten. Die Daten sind in Segmente von jeweils 256 Bytes aufgeteilt. Das letzte Segment kann beim Beschreiben gegebenenfalls mit bis zu 256 Null-Bytes aufgefüllt werden. Beim Lesen eines Satzes wird jedes Extra-Byte ignoriert, obgleich es im CRC gesammelt wird.

Ein Satz ist folgendermaßen aufgebaut:

Satz-Vorspann	Segment 1	101100110101101	Segment N	Satz-Nachspann

Es gibt N Segmente, wobei 256 x N die einschließlich Auffüllung zu schreibende Datenlänge ist.

Ein Kopf-Satz beinhaltet immer ein Segment; ein Daten-Satz beinhaltet 1....8 Segmente (gewöhnlich 8 Segmente).

a. Satz-Vorspann

Zu Beginn aller Sätze wird ein Satz-Vorspann mit folgender Belegung geschrieben:

Anfangs-lücke	2048 1-Bits	0 Bit	Sync-Byte

Die Anfangslücke dient zur Absicherung gegen Synchronisationsfehler am Ende des vorhergehenden Satzes oder gegen bereits vorher auf dem Band befindliche Daten, die falsch aufgezeichnet wurden.

Die lange Kette von 1-bits wird zur Berechnung der Daten-Schreibgeschwindigkeit und des zur Unterscheidung von 1- und 0-Bits erforderlichen Schwellen-Wertes verwendet.

Das einzelne 0-Bit dient zur Markierung des bevorstehenden Endes des Satz-Vorspanns und zur Bestimmung einer invertierten Aufzeichnung (siehe Abschnitt 8.3).

Das Synchronisationsbyte (Sync-Byte) dient zum Schutz vor unechter Synchronisation aufgrund von Bit-Sequenzen, die im Satz gefunden werden. Wird ein unkorrekter Wert für das Sync-Byte erkannt, so läßt sich daraus schließen, daß ein Synchronisations-Versuch in der Mitte eines Satzes oder in einem falschen Satz-Typ unternommen wurde. Dieses Byte dient zur Unterscheidung zwischen Kopf-Satz (#2C) und Daten-Satz (#16) in einem Datei-Block.

b. Segmente

Jedes Segment beinhaltet 256 Datenbytes und hat das folgende Format:

Byte 1	Byte 2	.....	Byte 256	CRC 1	CRC 2

„CRC 1" ist das höherwertige und „CRC 2" das niederwertige Byte der durch den CRC für die 256 Segment-Bytes berechneten logischen NOT-Verknüpfung. (Das verwendete CRC-Polynom lautet '$X^{15} + X^{12} + X^5 + 1$' mit dem Anfgangswert #FFFF).

c. Satz-Nachspann

Der Satz-Nachspann besteht aus 32 am Ende des Satzes geschriebenen 1-Bits.

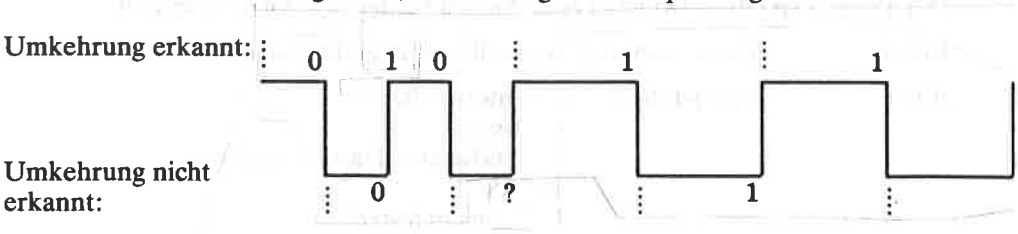
8.3 Bit-Format

Ein Bit wird auf das Band als eine Periode im Low-Zustand gefolgt von einer gleichen Periode im High-Zustand geschrieben. Eine logische Eins hat auf dem Band eine doppelt so lange Periode wie eine logische Null. Die Länge einer Periode für 0 kann durch den Anwender eingestellt werden (siehe CAS SET SPEED).

Die Band-Schaltkreise haben eine Tendenz zur Verschiebung der Flanken (Übergänge von High nach Low oder von Low nach High), so daß die Unterschiede zwischen den auf das Band geschriebenen Einsen und Nullen ausgeglichen werden. Es wird eine Vorkompensation verwendet, die auf die 1-Bit-Periode aufaddiert und von der 0-Bit-Periode subtrahiert wird. Dadurch wird die Wellenform beim Lesen idealisiert.

Beim Lesen wird die Aufzeichnungsgeschwindigkeit durch das Timing der 1-Bits im Satz-Vorspann bestimmt. Wenn es eine lange Sequenz gleicher Bits ist, werden die Flanken nicht verschoben und keine Vorkompensation durchgeführt. Da die Aufzeichnungsgeschwindigkeit unabhängig für jeden Satz bestimmt wird, sind die meisten Bandgeschwindigkeits-Schwankungen automatisch berücksichtigt.

Daten werden durch Low-High-Übergänge geschrieben, können beim Lesen jedoch invertiert (d.h. High-Low) werden. Es ist für die Firmware wichtig zu bestimmen, ob die gelesene Wellenform invertiert (umgekehrt) ist oder nicht. Ist dies nicht der Fall, so werden die Bits nicht korrekt gelesen, wie das folgende Beispiel zeigt:



Das 0-Bit im Satz-Vorspann wird verwendet, um zu bestimmen, ob die Aufzeichnung invertiert ist oder nicht.

Auf das Band gebrachte Bytes werden mit dem höchstwertigen Bit zuerst und dem niederwertigsten Bit zuletzt geschrieben.

8.4 Vorspann

Der Vorspann in einem Datei-Block beinhaltet Informationen über die Datei und die Daten in dem folgenden Daten-Satz. Einige Vorspann-Einträge werden zu unterschiedlichen Zwecken durch das System verwendet. Die verbleibenden Einträge stehen dem Anwender zur Verfügung. Diese Einträge sind der Datei-Typ (Byte 18) und alle Anwender-Felder (Bytes 24...63) einschließlich der logischen Länge (Bytes 24...25) und Einsprung-Adresse (Bytes 26...27). Das gesamte Anwender-Feld wird auf Null gesetzt, wenn es nicht benutzt wird.

Der Vorspann ist folgendermaßen aufgebaut:

System-Felder

Bytes 0...15	Dateiname	Auf 16 Bytes mit Nullen aufgefüllt.
Byte 16	Block-Nummer	Der erste Block ist normalerweise Block 1, und die weiteren Blöcke wachsen um 1.
Byte 17	Letzter Block	Ein Wert ungleich Null besagt, daß dies der letzte Block einer Datei ist.
Byte 18	Datei-Typ	Ein Wert, der den Datei-Typ aufzeichnet (siehe weiter unten).
Bytes 19...20	Daten-Länge	Die Anzahl Datenbytes im Daten-Satz.
Bytes 21...22	Daten-Speicherplatz	Woher die Daten ursprünglich geschrieben wurden.
Byte 23	Erster Block	Ein Wert ungleich Null besagt, daß dies der erste Block einer Datei ist.

Anwender-Felder

Bytes 24...25	Logische Länge	Totale Länge der Datei in Bytes.
Bytes 26...27	Einsprung-Adresse	Die Ausführungsadresse für Maschinencode-Programme.
Bytes 28...63	Nicht festgelegt	Können beliebig verwendet werden.

Der Datei-Typ (Byte 18) ist in eine Anzahl Felder, wie folgt, unterteilt:

Bit 0	Datei geschützt	Wenn dieses Bit gesetzt ist, ist die Datei geschützt.
Bits 1...3	Datei-Inhalt	0 = Internes BASIC 1 = Binär 2 = Bildschirm-Darstellung (Bild) 3 = ASCII 4...7 = nicht festgelegt
Bits 4..7	Version	ASCII-Dateien sollten Version 1, alle anderen Dateien Version 0 sein.

8.5 Lese- und Schreibgeschwindigkeit

Die Kassetten-Verwaltung kann auf Daten mit Lese- und Schreibgeschwindigkeiten zwischen 700 und 2500 Baud zugreifen. Gewöhnlich werden zwei Geschwindigkeiten in diesem Bereich verwendet, 1000 Baud (die durch einen EMS selektierte Standard-Geschwindigkeit) und 2000 Baud. Die Standard-Geschwindigkeit ist nahe der langsamsten Geschwindigkeit gewählt, um eine maximale Zuverlässigkeit zu erlangen. Die Zuverlässigkeit bei 2000 Baud ist zwar gut, jedoch nur, wenn auf der gleichen Maschine aufgenommen und eingelesen wird.

Bits werden als einzelne Tonzyklen auf das Band geschrieben. Der Ton für eine Eins hat immer die halbe Frequenz des Tons für eine Null. Dadurch sind Einsen auf dem Band doppelt so lang wie Nullen, d.h. die oben angegebenen Baud-Raten sind nur Durchschnittswerte und variieren in Abhängigkeit von den geschriebenen Daten.

Die Kassetten-Verwaltung muß mit dem eingebauten Kassetten-Mechanismus die auf das Band geschriebene Wellenform vorkompensieren, um die angegebenen Geschwindigkeiten einzuhalten. Dies bedeutet, daß die Länge der geschriebenen Bits geändert wird (Einsen verlängert, Nullen gekürzt), um die gelesenen Wellenformen, nachdem die Flanken durch die Kassetten-Schaltkreise verschoben wurden, wieder dem Ideal näherzubringen.

Hierzu ist es nur erforderlich, die Schreibgeschwindigkeit der Kassetten-Verwaltung einzustellen. Beim Lesen eines Satzes vom Band wird der Satz-Vorspann zur Berechnung der Aufzeichnungsgeschwindigkeit verwendet. Dadurch sind Bandgeschwindigkeits-Schwankungen zwischen unterschiedlichen Maschinen tolerierbar.

8.6 Auflistung

Zum Erstellen einer Auflistung vom Band liest die Kassetten-Verwaltung eine Sequenz Datei-Blöcke und druckt Informationen dazu aus. Die Datei-Blöcke können von einer beliebigen Datei stammen und in beliebiger Reihenfolge auftreten. Die Auflistung setzt sich solange fort, bis der Anwender die Escape-Taste drückt.

Die Information wird wie folgt dargestellt:

```
FILENAME block N L OK
```

FILENAME ist entweder der Name der Datei; oder „Unnamed File“, wenn der Dateiname mit einer Null beginnt.

Die Blocknummer N benennt den entsprechenden Block innerhalb der Datei. Normalerweise ist Block 1 der erste Block einer Datei.

L ist ein Zeichen, das den Typ und den Datenschutz-Status einer Datei angibt. Es wird erzeugt, indem #24 (Zeichen '$') zum Dateityp aus dem Vorspann addiert und mit #0F maskiert wird. Dadurch ergeben sich die folgenden Zeichen:

$	ein ungeschütztes BASIC-Programm
%	ein geschütztes BASIC-Programm
&	eine Binär-Datei
'	eine geschützte Binär-Datei
*	eine ASCII-Datei

Andere Zeichen sind möglich, aber die o.g. sind Standard-Datei-Typen, die durch das ROM auf der Karte geschrieben werden.

Die obige Information wird ausgegeben, wenn der Vorspann korrekt gelesen wurde.

OK wird ausgegeben, wenn der Daten-Satz korrekt gelesen wurde.

8.7 Lesen von Dateien

Bevor eine Datei gelesen werden kann, muß sie eröffnet werden (durch Aufruf von CAS IN OPEN). Es wird der Dateiname zugeordnet (siehe Abschnitt 8.10) und der erste Block der Datei gelesen, um den Vorspann anzusehen.

Die Datei kann entweder für eine Zeicheneingabe oder für direkte Eingabe, jedoch nicht beides, eröffnet werden. Der Eingabe-Modus wird durch den ersten Datei-Zugriff und nicht durch die Eröffnung bestimmt. Sobald ein Modus selektiert ist, kann der andere Modus zum Dateizugriff nicht mehr verwendet werden.

Zeicheneingabe (CAS IN CHAR) ermöglicht dem Anwender das Zeichenweise sequentielle Lesen der Datei. Blöcke der Datei werden dann vom Band gelesen und in den Puffer gebracht, wenn sie benötigt werden. Dies ist beim Lesen von Text-Dateien und ähnlichen Anwendungen wichtig.

Direkte Eingabe (CAS IN DIRECT) liest die gesamte Datei auf einmal in den Speicher. Dies ist beim Laden von Maschinencode-Programmen oder Bildschirm-Darstellungen und ähnlichen Anwendungen wichtig.

Unterbrechungen sind während des Lesens vom Band gesperrt, da andernfalls ernsthafte Timing-Probleme auftreten. Die Sperrung der Unterbrechungen schützt ebenfalls vor dem Auftreten der verschiedenen Zeitgeber-Unterbrechungen (siehe Abschnitt 10.1). Dadurch wird der Tongenerator-Chip deaktiviert, so daß ein lang anhaltendes Rauschen vermieden und die Tongenerator-Verwaltung stillgelegt (siehe SOUND RESET) werden kann.

8.8 Schreiben von Dateien

Bevor eine Datei geschrieben werden kann, muß sie eröffnet werden (durch Aufruf von CAS OUT OPEN). Es wird der Dateiname zugeordnet (siehe Abschnitt 8.10) und der in jedem Daten-Block enthaltene Vorspann belegt.

Die Datei kann entweder für eine Zeichenausgabe oder für direkte Ausgabe, jedoch nicht beides, eröffnet werden. Der Ausgabe-Modus wird durch das erste Datei-Schreiben und nicht durch die Eröffnung bestimmt. Sobald ein Modus selektiert ist, kann der andere Modus zum Datei-Schreiben nicht mehr verwendet werden.

Zeichenausgabe (CAS OUT CHAR) ermöglicht dem Anwender das zeichenweise Schreiben in die Datei. Die Zeichen werden zwischengespeichert, bis ein vollständiger Block (2048 Zeichen) zum Schreiben bereit ist; woraufhin dieser Daten-Block auf das Band geschrieben wird.

Direkte Ausgabe (CAS OUT DIRECT) schreibt die gesamte Datei auf einmal aus dem Speicher. Die geschriebenen Daten sind in Blöcke von jeweils 2048 Bytes aufgeteilt.

Die Ausgabe-Datei muß unabhängig vom verwendeten Ausgabe-Modus korrekt geschlossen werden (mittels CAS OUT CLOSE), da sonst der letzte Block der Datei nicht geschrieben werden kann.

Unterbrechungen sind während des Schreibens auf das Band gesperrt, da andernfalls ernsthafte Timing-Probleme auftreten. Die Sperrung der Unterbrechungen schützt ebenfalls vor dem Auftreten der verschiedenen Zeitgeber-Unterbrechungen (siehe Abschnitt 10.1). Dadurch wird der Tongenerator-Chip deaktiviert, so daß ein lang anhaltendes Rauschen vermieden und die Tongenerator-Verwaltung stillgelegt (siehe SOUND RESET) werden kann.

8.9 Gleichzeitiges Lesen und Schreiben

Die Kassetten-Verwaltung ermöglich die gleichzeitige Eröffnung von zwei Dateien. Die eine muß zum Lesen, die andere zum Schreiben eröffnet werden. Dadurch besteht die Möglichkeit, gleichzeitig von einer Datei zu lesen und in die andere hineinzuschreiben.

Wenn die Kassetten-Verwaltung einen Block lesen will, fordert sie den Anwender auf, die PLAY-Taste zu betätigen. Voraussetzung ist jedoch, daß das Band mit der zu lesenden Datei eingelegt ist. Ähnlich ist es mit einem zu schreibenden Block. Der Anwender wird aufgefordert, die REC- und PLAY-Taste zu drücken. Das Band, auf welches die Datei geschrieben werden soll, muß natürlich eingelegt sein. Die Kassetten-Verwaltung geht davon aus, daß das Band nicht gewechselt wurde und die entsprechenden Kassetten-Steuertasten gedrückt bleiben, bis eine Prompt-Meldung ausgegeben wird. Weiterhin nimmt sie an, daß die durch eine Prompt-Meldung angeforderte Taste gedrückt wird.

Es ist wenig sinnvoll, das gleichzeitige Lesen und Schreiben mit unterdrückten Meldungen der Kassetten-Verwaltung durchzuführen. Der einzige Hinweis auf das zu ladende Band wird durch Prompt-Meldungen gegeben.

8.10 Dateinamen

Wenn der Anwender eine Datei zum Lesen oder Schreiben eröffnet, wird der Dateiname spezifiziert. Der Dateiname ist ein String von jeweils 16 Zeichen (#00...#FF). Wenn

der spezifizierte Dateiname länger als 16 Zeichen ist, wird er abgeschnitten. Ist er kürzer als 16 Zeichen, so wird er auf 16 Zeichen mit Nullen (Zeichen #00) aufgefüllt.

Das Eröffnen einer zu lesenden Datei mit einem Dateinamen der Länge Null, oder mit einer Null am Anfang, hat eine spezielle Bedeutung. Es wird dann die nächste Datei vom Band gelesen. Die Kassetten-Verwaltung durchsucht das Band, bis sie den ersten Block der Datei gefunden hat und liest diese Datei. Wenn erst einmal der erste Block der Datei gefunden ist, wird ausschließlich von dieser Datei gelesen.

BASIC verwendet eine erweiterte Form der Dateinamen. Wenn das erste Zeichen eines BASIC-Dateinamens ein Ausrufungszeichen (Zeichen #21) ist, schaltet BASIC die Prompt-Meldungen ab (siehe Abschnitt 8.11) und entfernt das Ausrufungszeichen vor dem Namen. Diese Möglichkeit wird auf der Kassetten-Verwaltungsebene nicht geboten.

8.11 Meldungen der Kassetten-Verwaltung

Die Kassetten-Verwaltung gibt mehrere Meldungen aus, um den Anwender zu informieren und vor Fehlern zu warnen. Diese Meldungen können mit Ausnahme der Fehler- Meldungen durch CAS NOISY beliebig ein- oder ausgeschaltet werden.

a. Prompt-Meldungen

```
Press Play then any Key:
```

Diese Meldung wird ausgegeben, wenn die Kassetten-Verwaltung den ersten Block einer Datei vom Band lesen will, oder wenn sie einen Block lesen will, nachdem auf das Band geschrieben wurde (siehe Abschnitt 8.9). Sie weist darauf hin, daß das Band mit der zu lesenden Datei eingelegt und die PLAY-Taste des Bandgeräts gedrückt werden soll. Die Kassetten-Verwaltung gibt diese Meldung zu keinem anderen Zeitpunkt aus, da sie annimmt, daß das korrekte Band eingelegt und die PLAY-Taste gedrückt ist.

```
Press REC and PLAY then any Key:
```

Diese Meldung wird ausgegeben, wenn die Kassetten-Verwaltung den ersten Block einer Datei auf das Band schreiben will, oder wenn sie einen Block schreiben will, nachdem vom Band gelesen wurde. Sie weist darauf hin, daß das mit der Datei zu beschreibende Band eingelegt und die REC- und PLAY-Taste des Bandgeräts gedrückt werden soll. Die Kassetten-Verwaltung gibt diese Meldung zu keinem anderen Zeitpunkt aus, da sie annimmt, daß das korrekte Band eingelegt und die REC- und PLAY-Taste gedrückt ist.

b. Informations-Meldungen

```
Found FILENAME block N
```

Diese Meldung wird ausgegeben, wenn beim Lesen vom Band ein Vorspann gefunden wurde, der aus irgendwelchen Gründen nicht mit dem erwarteten Satz zusammenpaßt. Dies kann darauf hinweisen, daß das Band nicht korrekt positioniert (zu früh oder zu spät) oder das falsche Band verwendet wurde.

```
Loading FILENAME block N
```

Es wurde ein Block der Datei gefunden, der nun vom Band gelesen wird.

```
Saving FILENAME block N
```

Ein Block der Datei wird auf das Band geschrieben. FILENAME ist entweder der Name der Datei; oder „Unnamed File", wenn der Dateiname mit einer Null beginnt.

Die Blocknummer N benennt den entsprechenden Block, der gelesen oder geschrieben wird. Der erste Block einer Datei ist normalerweise Block 1, der zweite Block 2 usw.

c. Fehler-Meldungen

```
Rewind tape
```

Während der Suche nach einem Block der zu lesenden Datei wurde ein Block gefunden, der höher numeriert ist, als der gesuchte. Der gesuchte Block wird vermißt. Diese Meldung taucht häufig dann auf, wenn ein Lesefehler im gesuchten Block auftrat und der nächste Block bereits gefunden wurde.

```
Read error X
```

Beim Lesen vom Band trat ein Fehler auf. Das Band sollte zurückgespult und der Block erneut gelesen werden. Das X ist ein einzelner Buchstabe, der den aufgetretenen Lesefehler identifiziert:

'a'	Bit zu lang	Es wurde eine außergewöhnlich lange Eins oder Null gemessen. Dies weist häufig auf ein Lesen nach dem Ende eines Records hin.
'b'	CRC-Fehler	Daten wurden unkorrekt vom Band gelesen.
'd'	Block zu lang	Der Daten-Satz beinhaltet mehr als die erwarteten 2048 Datenbytes.

```
Write error a
```

Beim Schreiben auf das Band trat ein Fehler auf. Es gibt nur einen möglichen Schreibfehler. Dieser besagt, daß die Kassetten-Verwaltung nicht in der Lage war, ein Bit mit der geforderten Geschwindigkeit zu schreiben. Dieser Fehler kann nur auftreten, wenn der Anwender die Schreibgeschwindigkeit größer als die maximal mögliche eingestellt hat.

8.12 Escape-Taste

Die Escape-Taste der Tastatur kann zu bestimmten Zeitpunkten zum Abbrechen einer Kassetten-Operation verwendet werden.

Wenn die Kassetten-Verwaltung eine der Prompt-Meldungen ausgibt, ruft sie wiederholt KM READ CHAR auf, um den Tasten-Puffer zu leeren. Daraufhin ruft sie KM WAIT KEY auf und wartet, bis der Anwender eine Taste zur Quittierung der Prompt-Meldung drückt. Wenn der von der gedrückten Taste erzeugte Wert ein #FC ist (der normalerweise von der Escape-Taste erzeugt wird), bricht die Kassetten-Verwaltung das Lesen oder Schreiben ab und übergibt eine Fehler-Bedingung an die aufrufende Routine.

Beim Lesen oder Beschreiben der Kassette sind die Unterbrechungen gesperrt und die normale Abtastfunktion der Tastatur inaktiv. Die Kassetten-Verwaltung fragt beim Lesen oder Schreiben des Satz-Vorspanns die Tastatur selbst ab, um zu testen, ob die Escape-Taste (Taste 66) gedrückt ist. Die Kassetten-Verwaltung bricht die Schreib- oder Leseoperation ab und kehrt in die aufrufende Routine zurück (mit einer entsprechenden Fehler-Bedingung), wenn die Escape-Taste betätigt wurde. Beim Lesen oder Schreiben von Daten eines Satzes gibt es keine Möglichkeit, die Kassetten-Verwaltung zu unterbrechen, d.h. die Betätigung der Escape-Taste wird für einige Sekunden nicht erkannt.

8.13 Kassetten-Steuerung der unteren Ebene

Um dem Anwender die Möglichkeit zu geben, ein neues Schreib-/Lesesystem aufzubauen, befinden sich die Record-Routinen CAS READ und CAS WRITE in der Firmware-Sprungtabelle. Es gibt auf dieser Ebene eine dritte Routine, CAS CHECK, deren Eigenschaften durch die höheren Ebenen der Kassetten-Verwaltung nicht verwendet werden. Sie ermöglicht den Vergleich der auf das Band geschriebenen Daten mit den Daten im Speicher. Dadurch ist ein „Read after Write-Check" möglich, d.h. ein Testen der geschriebenen Daten durch erneutes Lesen.

In der Firmware-Sprungtabelle stehen auch noch Routinen zum Ein- und Ausschalten des Kassetten-Motors zur Verfügung (CAS START MOTOR und CAS STOP MOTOR). Es ist nicht notwendig, vor und nach dem Aufruf von CAS READ, CAS WRITE oder CAS CHECK den Motor ein- und auszuschalten, da diese Routinen automatisch dafür Sorge tragen.

9 Erweiterungs-ROMs, residente System-Erweiterungen und RAM-Programme

Das System kann bis zu 252 Erweiterungs-ROMs adressieren. Ihre Start-Adresse ist #C000 und sie überlagern damit die oberen 16KB des Speichers. Der Betriebssystem-Kern unterstützt zwei Varianten der Erweiterungs-ROMs, Vordergrund- und Hintergrund-ROMs. Eine residente System-Erweiterung (RSX) ist in der Anwendung mit einem Hintergrund-ROM vergleichbar, muß jedoch vor der Benutzung in das RAM geladen werden.

Ein Vordergrund-ROM beinhaltet ein oder mehrere Programme, wovon jedoch nur eines zu einem Zeitpunkt abgearbeitet werden kann. BASIC ist das standardmäßige Vordergrund-Programm. Andere mögliche Vordergrund-Programme sind:

- weitere Systeme, wie z.B. FORTH oder CP/M
- Anwendungen, wie z.B. Word Processor oder Spread Sheet
- Werkzeuge (Tools), wie z.B. ein Assembler oder Debugger

Ein geladenes RAM-Programm „übernimmt" die Maschine auf gleiche Art und Weise wie ein Vordergrund-Programm. Spiele z.B. sind generell RAM-Programme. Es sind bis zu 7 Hintergrund-ROMs installierbar, wovon jedes eine bestimmte Aufgabe unabhängig vom Vordergrund-Programm übernehmen kann. Es wird davon ausgegangen, daß jede Erweiterungs-Peripherie ein ihr zugeordnetes Hintergrund-ROM mit geeigneten Unterstützungs-Routinen besitzt. Andere Hintergrund-ROMs könnten die bestehende Maschinen-Software erweitern, z.B. durch die Bereitstellung weiterer Graphik-Funktionen.

Eine geladene residente System-Erweiterung (RSX) übernimmt bestimmte Aufgaben ähnlich den Hintergrund-ROMs. Eine RSX kann z.B. spezielle Unterstützung für einen bestehenden Drucker bereitstellen, sofern es günstiger ist, die Software auf Kassette und nicht im ROM (oder PROM) unterzubringen.

9.1 ROM-Adressierung

Erweiterungs-ROMs haben Adressen im Bereich von 0 bis 251. Um ein gegebenes ROM zu selektieren, bestimmt der Betriebssystem-Kern seine ROM-Adresse, indem er auf die Ein-/Ausgabe-Adresse #DF00 schreibt. Wenn sich ein ROM auf der selektierten Adresse befindet, werden bei allen Lesezugriffen auf die oberen 16KB des Speichers Daten aus dem Erweiterungs-ROM geholt. Befindet sich kein ROM auf der momentan selektierten Adresse, so wird der Inhalt des ROMs auf der Karte gelesen.

Beim ersten Einschalten der Maschine ist ROM 0 für das Vordergrund-Programm selektiert. Wenn sich kein Erweiterungs-ROM auf ROM-Adresse 0 befindet, wird das ROM auf der Karte verwendet und BASIC aufgerufen. Ein Erweiterungs-ROM erhält Priorität gegenüber dem ROM auf der Karte, wenn es sich auf ROM-Adresse 0 befindet.

Hintergrund-ROMs müssen auf ROM-Adressen im Bereich von 0 bis 7 festgelegt werden. Jedes außerhalb dieses Bereichs installierte ROM wird nicht als Hintergrund-ROM erkannt.

Vordergrund-ROMs müssen zusammenhängend von ROM-Adresse 1 (oder 0) aufwärts installiert sein. Der Betriebssystem-Kern beginnt bei der Suche nach den ROMs mit Adresse 0 aufwärts, bis die erste unbenutzte Adresse gefunden ist. Wenn ein Erweiterungs-ROM 0 installiert ist, kann auf das ROM der Karte durch Verwendung der ersten unbenutzten ROM-Adresse zugegriffen werden.

Der Betriebssystem-Kern unterstützt eine „Far-Address", über die Subroutinen in Erweiterungs-ROMs aufgerufen werden können. Die „Far-Address" ist ein 3-Byte-Objekt, deren letztes Byte eine ROM-Select-Nummer ist. Da die Anordnung der ROMs auf einer Erweiterungskarte beliebig ist, muß der ROM-Select-Teil einer „Far-Address" zur Laufzeit festgelegt sein. Die Möglichkeiten der „Sideway"-ROM-Adressierung erlauben einem Vordergrund-Programm die Inanspruchnahme von vier zusammenhängenden ROM-Select-Adressen. Subroutinen-Aufrufe zwischen den ROMs werden ebenfalls unterstützt, ohne daß das Programm die aktuelle ROM-Adresse kennen muß.

9.2 Format eines Erweiterungs-ROMs

Ein Erweiterungs-ROM kann maximal 16KB groß sein. Das erste Byte beginnt auf Adresse #C000. Die ersten Bytes des ROMs sind das „ROM Prefix" und müssen das folgende Format haben:

Byte 0: ROM-Typ
Byte 1: ROM-Markierungs-Nummer
Byte 2: ROM-Versions-Nummer
Byte 3: ROM-Modifikations-Ebene
Byte 4: Externe Befehlstabelle

Der ROM-Typ spezifiziert die Art des ROMs und muß einen der folgenden Werte annehmen:

0: Vordergrund-ROM
1: Hintergrund-ROM
2: Erweiterungs-ROM

Das ROM auf der Karte muß einheitlich Bit 7 des Typ-Bytes gesetzt haben (dadurch ist sein Typ-Byte #80). Diese Marke wird zum Erkennen des Vordergrund-ROM-Endes verwendet. Wenn ein Vordergrund-Programm nicht in ein einzelnes ROM hineinpaßt, sollten die erforderlichen Zusatz-ROMs als Erweiterungs-ROMs gekennzeichnet werden.

Die Markierungs-Nummer, Versions-Nummer und Modifikations-Ebene können auf jeden erforderlichen Wert gesetzt werden.

Die externe Befehlstabelle enthält eine Liste von Befehlsnamen und eine Sprungtabelle. Jeder Befehlsname ist in sich mit dem gleichen numerierten Eintrag versehen, wie die Sprungtabelle. Die Tabelle hat folgendes Format:

Bytes 0...1: Adresse der Befehlsnamen-Tabelle
Bytes 2...4: Sprungtabellen-Eintrag 0
Bytes 5...7: Sprungtabellen-Eintrag 1
... usw. ... usw.:

Die Befehlsnamen-Tabelle ist eine Liste von Namen, von denen jeder bis zu 16 Zeichen lang sein kann. Nur im letzten Zeichen jedes Namens muß das Bit 7 gesetzt sein. Die Tabelle wird durch eine Null (Zeichen 0) nach dem letzten Zeichen des letzten Namens abgeschlossen. Abgesehen von der Tatsache, daß alle Zeichen im Bereich 0...127 liegen müssen und das erste Zeichen keine Null sein darf, bestehen keine weiteren Einschränkungen für Zeichen in Befehlsnamen. Wenn jedoch ungeeignete Zeichen gewählt werden, kann es sich für Programme wie BASIC als unmöglich erweisen, auf die Befehle zuzugreifen.

BASIC erwartet alphabetische Zeichen (Großbuchstaben) und erlaubt keine Zeichen wie Leerzeichen oder Komma in den Befehlsnamen.

Das „ROM Prefix“ für das ROM auf der Karte ist:

```
                ORG   #C000           ; Start des ROM

                DB    #80+0           ; ROM auf der Karte (Vorder-
                                        grund)
                DB    1               ; Marke 1
                DB    0               ; Version 0
                DB    0               ; Modifikation 0

                DW    NAME_TABLE      ; Adresse der Namen

                JP    START_BASIC     ; Der einzige Einsprung in der
                                        Sprungtabelle
NAME_TABLE:     DB    'BASI','C'+#80  ; Der einzige Befehlsname
                DB    0               ; Ende der Namenstabelle
```

Das „ROM Prefix“ für eine serielle Ein-/Ausgabe-Karte kann sein:

```
                ORG   #C000                ; Start des ROM

                DB    1                    ; Hintergrund-ROM
                DB    0                    ; Marke 0
                DB    5                    ; Version 5
                DB    0                    ; Modifikation 0

                DW    NAME_TABLE

                JP    EMS_ENTRY            ; 0 ROM-Power-up-Eintrag
                JP    RESET                ; 1
                JP    SET_BAUD_RATE        ; 2
                JP    GET_CHARACTER        ; 3
                JP    PUT_CHARACTER        ; 4
                .... etc.

NAME_TABLE:     DB    'SIO DRIVE','R'+#80      ; 0
                DB    'SIO RESE','T'+#80       ; 1
                DB    'SIO SET.BAU','D'+#80    ; 2
                DB    'SIO GET.CHA','R'+#80    ; 3
                DB    'SIO PUT.CHA','R'+#80    ; 4
                .... etc.
                DB    0                        ; Ende der Namenstabelle
```

Man beachte, daß der Befehlsnamen-Tabelleneintrag für den Power-up-Eintrag ein Leerzeichen enthält. Dies ist trotzdem ein legaler Name, aber BASIC wird aufgrund seiner Behandlung von Leerzeichen niemals in der Lage sein, ihn zu erzeugen. Da BASIC diesen Namen nicht erzeugen kann, ist es einem BASIC-Anwender nicht möglich, den Power-up-Eintrag fälschlicherweise aufzurufen (siehe Abschnitt 9.4).

9.3 Vordergrund-ROMs und RAM-Programme

Jeder der Einträge eines Vordergrund-ROMs repräsentiert ein separates Programm, dessen Name durch den korrespondierenden Eintrag in der Namens-Tabelle gegeben ist. Der erste Eintrag von ROM 0 ist der standardmäßige Einsprung-Punkt eines Power-up am Ende des EMS. Nach dem Laden eines RAM-Programms wird es ähnlich wie ein Vordergrund-ROM gehandhabt. Sie unterscheiden sich nur dadurch, daß das RAM kein „ROM Prefix“ besitzt und der erforderliche Einsprung-Punkt separat bestimmt wird.

Bevor ein Vordergrund-Programm ausführbar ist, wird die Maschine in ihren EMS-Zustand versetzt, d.h. die gesamte Hardware und Firmware wird initialisiert. Die Umgebung und die Einsprung-Bedingungen sind wie folgt:

Speicher:

Abschnitt 2 beschreibt die Speicherbelegung des Systems. Drei Speicherbereiche stehen dem Programm zur Verfügung:

1. Der Bereich für statische Variablen

Der Bereich von #AC00 bis einschließlich #BOFF ist für die Benutzung durch das Vordergrund-Programm reserviert, obwohl es mehr oder weniger Speicherplatz verwenden kann, als es eigentlich benötigt. Gegebenenfalls kann auch ein Vordergrund-Datenbereich ab Adresse #0040 reserviert werden.

2. Der Stack

Der Hardware-Stack ist auf einen Bereich unmittelbar unterhalb #C000 gesetzt und belegt mindestens 256 Byte.

3. Der Memory Pool

Ein Großteil des verbleibenden Speicherplatzes ist für das Vordergrund-Programm verfügbar. Er ist davon abhängig, welchen Speicherplatz die durch das Vordergrund-Programm zur Initialisierung ausgewählten Hintergrund-ROMs in Anspruch nehmen.

Register:

Die Basis und die Grenze des freien Speicherbereichs werden dem Programm in Registern übergeben.

BC = Adresse des obersten verwendbaren Bytes im Speicher (#B0FF).
DE = Adresse des untersten Bytes im Memory Pool (#0040).
HL = Adresse des obersten Bytes im Memory Pool (#ABFF).

Man beachte, daß das Programm jeden Speicherplatz zwischen den in DE und BC befindlichen Adressen (d.h. von #0040 bis #B0FF) verwenden kann. Der Inhalt von HL gibt den Standard-Speicherplatz für statische Variablen an. Das Programm darf durchaus mehr oder weniger Speicherplatz dafür belegen. Auch der Vordergrund-Datenbereich kann am unteren Ende des Speichers reserviert werden. Das Programm sollte vor der Initialisierung eines Hintergrund-ROMs die Register HL und DE besetzen, um den für Variablen verwendeten Bereich zu kennzeichnen.

SP ist auf den von der Maschine festgelegten Bereich ab #C000 gesetzt. Das Programm kann bis zu 256 Byte Stack verwenden.

Der Inhalt der anderen Register ist unbestimmt. Zu beachten ist, daß der alternative Registersatz (AF' BC' DE' HL') für das Programm nicht zur Verfügung steht (siehe Anhang XI).

ROM-Select und -State:

Für ROM-Programme:	Das Vordergrund-ROM ist selektiert. Das obere ROM ist freigeschaltet. Das untere ROM ist gesperrt.
Für RAM-Programme:	Kein ROM ist selektiert. Das obere ROM ist gesperrt. Das untere ROM ist gesperrt.

Generell:

Unterbrechungen sind freigeschaltet.
Die gesamte Hardware und Firmware befindet sich im Anfangszustand.
Insbesondere sind die installierten Erweiterungs-Geräte rückgesetzt, aber noch nicht initialisiert.

Es liegt in der Verantwortung des Vordergrund-Programms, jedes Hintergrund-ROM zu initialisieren und die RSX zu laden und zu initialisieren. Der Einsprung KL ROM WALK des Betriebssystem-Kerns sucht die Hintergrund-ROMs und initialisiert die gefundenen. Der Einsprung KL INIT BACK des Betriebssystem-Kerns initialisiert ein bestimmtes Hintergrund-ROM.

Dem Betriebssystem-Kern müssen die Adressen des ersten und letzten Bytes des Memory Pools übergeben werden, da das Vordergrund-Programm seine festen Datenbereiche reservieren muß, ehe es nach den Hintergrund-ROMs sucht. Die Hintergrund-ROMs können sich Speicherplatz für ihren eigenen Gebrauch zuteilen, indem sie eine oder beide Speichergrenzen verschieben. Wenn daher das Vordergrund-Programm die Hintergrund-ROMs freigibt, muß es mit einem Memory Pool auskommen, dessen Grenzen solange nicht festgelegt sind, bis alle Hintergrund-ROMs initialisiert sind. Man beachte, daß die Datenbereiche des Vordergrund-Programms festgelegt sind, während ein Hintergrund-Programm mit variablen Datenbereichen arbeiten muß.

Wenn Hintergrund ROMs nicht initialisiert sind, ist die Speicherbelegung recht einfach. Da jedoch Disketten, Lichtgriffel etc. sehr wahrscheinlich Hintergrund-ROMs für ihre Unterstützungs-Software verwenden, ist es sehr einschränkend, Hintergrund-ROMs nicht freizuschalten. Das auf der Karte implementierte BASIC initialisiert alle Hintergrund-ROMs beim EMS. Der Benutzer kann selbst entscheiden, ob er eine RSX vom Band laden möchte.

9.4 Hintergrund-ROMs

Hintergrund-ROMs bleiben unbenutzt, bis sie durch das Vordergrund-Programm initialisiert werden.

Während der Initialisierung kann sich die Hintergrund-Software ihren Speicher selbst zuordnen, sowie die Hardware und alle Datenstrukturen initialisieren. Unmittelbar nach der Initialisierung übernimmt der Betriebssystem-Kern das ROM in die Liste der potentiellen Aufrufer externer Befehle.

Der erste Eintrag in einer Hintergrund-ROM-Sprungtabelle ist seine Initialisierungs-Routine. Diese Routine wird nur durch die Firmware zur ROM-Initialisierung aufgerufen und ist nicht für einen Benutzer-Aufruf vorgesehen. Tricks, wie das Einfügen eines Leerzeichens in den Namen, machen es BASIC unmöglich, einen korrekten Namen zu erzeugen. Dem Anwender bleibt die Möglichkeit verwehrt, den Eintrag aufzurufen. Die Einsprung- und Aussprung-Bedingungen für die Initialisierungs-Routine sind:

Einsprung:

DE beinhaltet die Adresse des untersten Bytes im Memory Pool.
HL beinhaltet die Adresse des obersten Bytes im Memory Pool.

Aussprung:

Das Carry ist wahr.
DE beinhaltet die neue Adresse des untersten Bytes im Memory Pool.
HL beinhaltet die neue Adresse des obersten Bytes im Memory Pool.

A, BC und andere Flags sind zerstört.
Alle anderen Register sind unverändert.

Hinweise:

Das obere ROM ist freigeschaltet und selektiert.
Das untere ROM ist gesperrt.

Die Routine kann den wechselseitigen Registersatz nicht verwenden.

Das ROM kann sich Speicherplatz am oberen oder unteren (oder beiden) Ende des Memory Pool zuweisen, indem die entsprechenden Register verändert und der neue Wert zurückgegeben wird. Um beispielsweise 256 Bytes mit der gegebenen Adresse #AB7F am oberen Ende des Pools zu reservieren, würde das Programm 256 vom HL-Register subtrahieren, wodurch die neue obere Pool-Adresse #AA7F wäre und der reservierte Bereich von #AA80 bis einschließlich #AB7F reichen würde.

Wenn aus der Initialisierungs-Routine zurückgekehrt wird, speichert der Betriebssystem-Kern die Basis-Adresse des oberen vom ROM selbst zugeteilten Bereiches (d.h. HL+1). Beim Aufruf eines ROM-Eintrags wird diese Adresse im IY-Index-Register übergeben. Dadurch können die ROM-Routinen leicht auf ihre oberen Variablen-Bereiche zugreifen, selbst wenn sie dynamisch zugeteilt wurden. Zugriffe auf einen unteren Variablen-Bereich sollten über Zeiger im oberen Bereich erfolgen. Da Hintergrund-ROMs keine absoluten Speicherbereiche verwenden, können auch keine Probleme mit dem gegenseitigen Zusammenstoßen von Hintergrund-ROMs oder von Hintergrund-ROMs mit Vordergrund-Programmen auftreten. Man beachte, daß der obere Datenbereich eines Hintergrund-ROMs sehr leicht oberhalb von #4000 liegen kann, so daß unabhängig vom Freischaltzustand des unteren ROMs auf ihn zugegriffen werden kann.

Der Betriebssystem-Kern plaziert auch das ROM in seine Liste der möglichen Aufrufer externer Befehle. Man beachte, daß beim Abtasten der Liste nach externen Befehlen die zuletzt durchgeführte Addition zuerst versucht wird. Der Eintrag KL ROM WALK bearbeitet die ROMs in umgekehrter Adressenordnung (7,6, ...1) und ignoriert jede Lücke oder jedes Vordergrund-ROM, wodurch die ROMs in der Reihenfolge 1,2, ...7 gesucht werden.

9.5 Residente System-Erweiterungen

Eine RSX ist ähnlich einem Hintergrund-ROM. Die Verantwortung für das Laden einer RSX und die Bereitstellung von Speicherplatz obliegt dem Vordergrund-Programm. Um mit der dynamischen Zuteilung von Speicherplatz für Vordergrund-Programme übereinzustimmen, ist es empfehlenswert, die RSXen unabhängig oder beim Laden zuzuteilen. Eine RSX kann durch Schreiben eines kurzen Ladeprogramms in BASIC zugeteilt werden. Dieses Programm sollte die RSX in einem Format lesen, das einfach zugeteilt und mittels POKE-Befehl in den Speicher gebracht werden kann.

Nach dem Laden einer RSX kann sie in die Liste der potentiellen Aufrufer externer Befehle aufgenommen werden, indem KL LOG EXT angestoßen, die externe Befehlstabellen-Adresse der RSXen und ein Speicherblock von vier Bytes (in den zentralen 32KB RAM) für den Betriebssystem-Kern übergeben wird. Das Format der Tabelle entspricht exakt der eines Hintergrund-ROMs (siehe Abschnitt 9.2). Der einzige Unterschied besteht in der Interpretation der Tabelle. Der erste Sprungtabellen-Eintrag wird nicht automatisch durch den Betriebssystem-Kern aufgerufen und muß nicht unbedingt die Initialisierungs-Routine der RSXen sein.

Die Externe Befehlstabelle für eine BASIC-Graphik-Erweiterung kann folgendermaßen aussehen:

```
                DW    NAME_TABLE        ; Adresse der Namenstabelle

                JP    DRAW_CIRCLE       ; 0
                JP    DRAW_TRIANGLE     ; 1
                JP    FILL_AREA         ; 2
                .... etc.
NAME_TABLE:
                DB    'CIRCL','E'+#80         ; 0
                DB    'TRIANGL','E'+#80       ; 1
                DB    'FIL','L'+#80           ; 2
                .... etc.
                DB    0                       ; Ende der Namenstabelle
```

Man beachte, daß beim Abtasten der Liste auf externe Befehle die zuletzt ausgeführte Addition zuerst versucht wird. Da RSXen im allgemeinen nach dem Initialisieren der Hintergrund-ROMs geladen werden, haben RSX-Befehle gegenüber den in Hintergrund-ROMs befindlichen Vorrang.

9.6 Externe Befehle

Wenn ein Vordergrund-Programm feststellt, daß ein externer Befehl ansteht, sollte es den Betriebssystem-Kern-Eintrag KL FIND COMMAND aufrufen und ihm einen String mit dem Befehlsnamen übergeben. Diese Routine geht zuerst einmal davon aus, eine RSX oder ein Hintergrund-ROM aufzufinden, dessen externe Befehlstabelle den Befehl enthält. Nur die RSXen und ROMs, die entsprechend initialisiert sind, werden berücksichtigt. Nach dem Auffinden des Befehls wird die „Far-Address" des korrespondierenden Sprungtabellen-Eintrages zurückgegeben (siehe Abschnitt 2.3). Wird der Befehl nicht gefunden, so startet die Routine bei ROM 0 und sucht nach einem Vordergrund-ROM, dessen externe Befehlstabelle den Befehl enthält. Ist dieses Vordergrund-ROM gefunden, wird das System zurückgesetzt und in das entsprechende Vordergrund-Programm gesprungen. Ein Fehlerhinweis wird zurückgegeben, falls keine Übereinstimmung mit diesem Befehl gefunden wird.

Zu beachten ist, daß der externe Befehlsmechanismus Hintergrund- und RSX-Routinen auffinden kann und die Umschaltung von Vordergrund-Programmen ermöglicht. Der erste Befehlsname in einem Hintergrund-ROM sollte nicht als Befehl verwendet werden, da er mit dem impliziten Initialisierungs-Eintrag korrespondiert.

Bei der ersten Anwendung einer Hintergrund- oder RSX-Routine sollte der externe Befehlsmechanismus zur Feststellung seiner Sprungtabellen-Adresse benutzt werden. Diese sollte gespeichert und direkt für aufeinanderfolgende Routinen-Aufrufe verwendet werden. Es ist unklug anzunehmen, daß ein bestimmtes Hintergrund-ROM immer im gleichen Sockel steckt oder eine verschiebbare RSX immer auf der gleichen Adresse steht.

Es ist die Aufgabe des Vordergrund-Programms, den externen Befehl, dessen Adresse gefunden wurde, aufzurufen und die Parameter in geeigneter Form zu übergeben. Die Funktionsweise von BASIC (auf der Karte) sollte als Modell für andere Vordergrund-Programme dienen, um die gemeinsame Verwendung von Befehlen durch andere Systeme zu ermöglichen.
Sie ist wie folgt:

Ein externer Befehl wird durch einen vertikalen Strich ('**|**'), gefolgt vom Befehlsnamen und – manchmal – einer Parameter-Liste gekennzeichnet. Der Strich ist nicht Bestandteil des Befehlsnahmens. Der Befehlsname muß aus alphabetischen Zeichen (die in Großbuchstaben übersetzt werden), numerischen Zeichen oder Punkten bestehen.

Parameter werden als Werte übergeben. Jeder Parameter kann ein numerischer Ausdruck, dessen berechneter Wert übergeben wird, oder eine Adresse sein. Anzahl und Typ des Parameters müssen zwischen dem BASIC-PROGRAMM und dem Befehl übereinstimmen, da BASIC dies nicht überwacht.

Jeder übergebende Parameter ist eine 2-Byte-Nummer, dessen Interpretation von seinem Typ abhängt:

Integer-Ausdruck:	2er – Komplement des Integer-Ergebnisses.
Real-Ausdruck:	Das Realzahl-Ergebnis als vorzeichenlose Integerzahl.
Variablen-Referenz:	Adresse eines Variablen-Wertes (für einen String ist dies die Adresse eines Deskriptors).

Ein String-Deskriptor ist drei Bytes lang. Byte 0 beinhaltet die Länge des Strings. Die Bytes 1 und 2 beinhalten die Adresse, unter welcher der String gespeichert ist. Die Adresse des Strings ist bedeutungslos, wenn die String-Länge 0 ist. String-Variablen dürfen gewechselt werden, wenn der String-Deskriptor unverändert bleibt.

Einsprung:

A beinhaltet die Anzahl der Parameter.
IX beinhaltet die Adresse der Parameter.
IY beinhaltet die Adresse des oberen ROM-Datenbereichs, wenn der Befehl in einem Hintergrund-ROM gefunden wurde. IY ist undefiniert, wenn der Befehl in einer externen RSX-Befehlstabelle gefunden wurde.

Aussprung:

AF, BC, DE, HL, IX und IY sind verfälscht.
Alternative Register bleiben unberührt.

Hinweis:

Index-Register IX beinhaltet die Adresse der Parameter. Bei n Parametern befindet sich der i-te Parameter auf dem Offset (n-i)x 2 der Index-Register-Adresse. Dadurch befindet sich der erste Parameter auf dem größten Offset; auf den letzten Parameter wird durch IX gezeigt.

Das IY-Register wird durch den Betriebssystem-Kern und nicht durch BASIC gesetzt. Die Register A und IX und der Parameter-Bereich werden durch BASIC gesetzt.

10 Unterbrechungen

Es gibt in einer nicht erweiterten Maschine nur eine Unterbrechungsquelle, nämlich eine normale Zeit-Unterbrechung. Erweiterungs-Karten können weitere Unterbrechungen erzeugen, müssen jedoch geeignete Software zur Bearbeitung der Zusatz-Unterbrechungen bereitstellen.

Das System arbeitet die meiste Zeit mit freigeschalteten Unterbrechungen. Es ist nicht ratsam, die Unterbrechungen für einen längeren Zeitraum zu sperren, da die Zeit-Unterbrechungen vermißt werden.

Eine große Anzahl der Firmware-Routinen geben Unterbrechungen frei, was in ihren Beschreibungen entsprechend vermerkt ist. Insbesondere die mit ROMs und den Restart-Befehlen (z.B. LOW JUMP) befaßten Betriebssystem-Kern-Routinen schalten Unterbrechungen frei.

10.1 Zeit-Unterbrechung

Die Zeit-Unterbrechung tritt ungefähr alle 1/300 Sekunden auf. Bei Maschinen mit PAL- (wie in Deutschland) oder SECAM-Monitor (wie in Frankreich) ist der Zeitgeber mit einem Bildrücklauf von 1/60, bei Maschinen mit NSTC-Monitor (wie in den USA) mit einem Bildrücklauf von 1/50 synchronisiert. Die Zeit-Unterbrechung wird durch den Betriebssystem-Kern bearbeitet und dem System folgendermaßen zur Verfügung gestellt:

a) Schnelle Unterbrechungen. Periode = 1/300 Sekunde

Für hohe Auflösung oder sehr kurze Perioden (nicht für allgemeine Verwendung beabsichtigt).

b) Ton-Erzeugungs-Unterbrechung. Periode = 1/100 Sekunde

Diese Unterbrechung treibt die Tongenerator-Firmware und ist dem System auf andere Weise nicht zugänglich.

c) Bildrücklauf-Unterbrechung. Periode = 1/50 oder 1/60 Sekunde

Für Aktionen, die während eines Bildrücklaufes getätigt werden müssen. Das Blinken der Ink wird z.B. während einer Bildrücklauf-Unterbrechung bewerkstelligt.

d) Normale Unterbrechung. Periode = 1/50 Sekunde

Dies ist eine Zeit-Unterbrechung für allgemeine Zwecke. Die Tastatur wird beim Start jeder normalen Unterbrechung abgefragt.

e) System-Takt

Es gibt einen Zeitgeber, der kurze Zeitintervalle von 1/300 Sekunde erzeugt. Dieser kann zur Zeitmessung verwendet werden, ohne ein relativ aufwendiges Ereignis einstellen zu müssen (siehe Abschnitt 10.5). Der Zeitgeber kann durch Aufruf von KL TIME PLEASE gelesen und durch KL, TIME SET eingestellt werden.

10.2 Externe Unterbrechungen

Der Z80 arbeitet im Unterbrechungs-Modus 1, d.h. alle Unterbrechungen verursachen einen auszuführenden RST 7. Der Bearbeitungscode für Unterbrechungen im Betriebssystem-Kern kann zwischen Zeit-Unterbrechungen und externen Unterbrechungen unterscheiden. Er erreicht dies durch erneutes Freischalten der Unterbrechungen mit Hilfe der Unterbrechungs-Routine. Wenn sich diese Unterbrechung wiederholt, wird sie als externe Unterbrechung erkannt. Ansonsten gilt sie als Zeit-Unterbrechung. Zu beachten ist, daß die Quelle der externen Unterbrechung die Unterbrechungs-Bedingung nicht löschen sollte bis die Software sie rücksetzt.

Bevor eine externe Unterbrechung freigeschaltet wird, muß der Unterbrechungs-Bearbeiter (Handler) installiert sein.

Dies wird durch Umkopieren der 5 Bytes von Adresse #003B auf einen neuen Speicherplatz und das Ersetzen durch geeigneten Code (vielleicht einschließlich Jump) erreicht. Wenn der Betriebssystem-Kern eine externe Unterbrechung erkennt, ruft er die Adresse #003B im RAM zur Bearbeitung der Unterbrechung auf:

Einsprung:

Keine Bedingungen.

Aussprung:

AF, BC, DE und HL sind verfälscht.
Alle anderen Register geschützt.

Hinweise:

Unterbrechungen sind gesperrt und müssen gesperrt bleiben.
Das untere ROM ist gesperrt.
Die oberen ROM-Select und -State sind unbestimmt.
Der alternative Register-Satz darf nicht verwendet werden.

Die Unterbrechungs-Routine muß feststellen, ob die Unterbrechung bearbeitet werden kann. Wenn ja, muß sie die Unterbrechung am Schluß löschen. Liegt die Unterbrechung nicht im Bearbeitungsbereich dieser Routine, so sollte sie auf die Kopie-Adresse der von Speicherplatz #003B geholten Bytes springen, die diese Unterbrechung bearbeiten kann. Dies macht erforderlich, daß der auf Speicherplatz #003B befindliche Code unabhängig positioniert wird, sofern ein zweiter Unterbrechungs-Handler installiert ist. Der beim EMS auf Adresse #003B gebrachte Code ist positions-unabhängig und führt lediglich zum Rücksprung.

Man achte darauf, daß der Unterbrechungs-Handler-Code im RAM irgendwo zwischen #0040 und #BFFF sein muß. Die Unterbrechungs-Handler sollten so kurz wie möglich sein. Wenn eine Unterbrechung viel Bearbeitungszeit bis zum Löschen der Unterbrechungs-Bedingung erfordert, sollte die Unterbrechung ein Ereignis anstoßen, um die Behandlung außerhalb des Unterbrechungs-Bereiches durchzuführen.

10.3 Nichtmaskierbare Unterbrechungen

Es sind in der Firmware keine Vorkehrungen zur Bearbeitung der nichtmaskierbaren Unterbrechungen (NMI) getroffen worden (ungeachtet der Tatsache, daß der NMI auf dem externen Bus-Anschluß verfügbar ist). Es bestehen für verschiedene aktive Firmware-Routinen Timing-Beschränkungen beim Auftreten eines NMI. Das gilt insbesondere für diejenigen, die mit dem Centronics-Kanal, dem PPI für Zugriffe auf den Tongenerator-Chip und die Tastatur, sowie der Kassette befaßt sind. Es wird empfohlen, den NMI nicht zu verwenden.

10.4 Unterbrechungen und Ereignisse

Als allgemeine Regel gilt, daß Hardware-Unterbrechungen so schnell wie möglich in ihre Software-Äquivalente übersetzt werden sollten. Die Bearbeitung von Ereignissen ist flexibler als die Bearbeitung von Hardware-Unterbrechungen. Es bestehen z.B. keine Einschränkungen hinsichtlich des Speicherplatzes von Ereignis-Routinen oder hinsichtlich der Unterbrechungs-Freischaltung.

Ereignisse werden durch einen Ereignis-Block beschrieben. Dieser Block beinhaltet die Ereignis-Klasse, die Ereignis-Zahl und eine Ereignis-Routineadresse. Beim Auftreten eines Ereignisses wird der Ereignis-Block angestoßen, und der Betriebssystem-Kern bearbeitet die für jeden Anstoß aufzurufende Ereignis-Routine (die Anzahl der offenstehenden Anstöße wird im Ereignis-Block festgehalten). Die Ereignis-Routine wird nicht notwendigerweise sofort aufgerufen. Wann die Ereignis-Routine tatsächlich abläuft, hängt von der Ereignis-Klasse wie folgt ab:

a) Spezielle Asynchron-Ereignisse

Dies ist eine ungewöhnliche Ereignis-Klasse. Die Ereignis-Routine wird unmittelbar während der Unterbrechungs-Bearbeitung aufgerufen. Die Routine muß über den Unterbrechungscode ansprechbar sein, darf die Unterbrechungen nicht freischalten, IX- oder IY-Register nicht verfälschen oder den alternativen Registersatz verwenden. Die Routine sollte so kurz wie möglich sein.

b) Normale Asynchron-Ereignisse

Dies ist die flexibelste Ereignis-Art. Beim Anstoßen des Ereignisses wird die Ereignis-Routine nicht aufgerufen, sondern der Ereignis-Block in die Warteschlange für anstehende Unterbrechungen und Ereignisse eingereiht.

Nachdem die momentane Unterbrechung bearbeitet ist und bevor der Betriebssystem-Kern aus dem Unterbrechungsbereich zurückkehrt, werden alle Ereignise der Warteschlange für anstehende Unterbrechungen und Ereignisse bearbeitet. Während der Ereignis-Bearbeitung läuft das System mit freigeschalteten Unterbrechungen und kann nicht mehr als im Unternehmungsbereich befindlich betrachtet werden. Sie verwendet ihren eigenen Stack, nicht den System-Stack. Dieser private Stack ist 128 Bytes groß.

Die asynchrone Ereignis-Routine wird kurz nach dem Anstoßen des Ereignisses aufgerufen und ist in der Bearbeitungsweise oder der Anordnung im Speicher nicht eingeschränkt. Die Ereignis-Routine darf so lang sein wie erforderlich. Jeder weitere während der Bearbeitung einer Ereignis-Routine empfangene Anstoß wird auf die Ereignis-Zahl aufaddiert und vor der Rückkehr in das unterbrochene Programm bearbeitet.

c) Synchrone Ereignisse

Synchrone Ereignisse werden in die Warteschlange für anstehende synchrone Ereignisse eingereiht. Sie werden solange nicht bearbeitet, bis das Vordergrund-Programm die Abarbeitung der Warteschlange ermöglicht. Dies kann zur Steuerung der Wechselwirkungen zwischen unterschiedlichen Programmteilen verwendet werden.

10.5 Unterbrechungs-Warteschlangen

Die verschiedenen Zeit-Unterbrechungen liefern drei Quellen für Ereignis-Anstöße. Die beim Auftreten einer der Unterbrechungen anzustoßenden Ereignisse werden in Warteschlangen gespeichert. Für jede Anstoß-Quelle gibt es eine Warteschlange. Der Anwender stellt einen Speicherbereich für die Verwendung durch den Betriebssystem-Kern zur Verfügung. Die Größe dieses Bereichs hängt von der ihn benutzenden Warteschlange ab. Die letzten 7 Bytes des Bereichs sind immer ein Ereignis-Block, den der Anwender entsprechend initialisieren sollte. Anhang X beschreibt die Belegung dieses Blocks detailliert.

a) Schnelle Ereignisse

Ereignisse der Warteschlange für schnelle Ereignisse werden mit jeder schnellen Unterbrechung angestoßen, d.h. jede 1/300 Sekunde. Ein Block für schnelle Ereignisse ist 9 Bytes groß.

b) Normale Ereignise

Jedes Ereignis der Warteschlange für normale Ereignisse ist auf einen Zeitgeber bezogen. Der Zeitgeber kann ein „One Shot" sein, der einmal ausschaltet, oder ein Wiederholer, der periodisch ausschaltet. Der Zeitgeber zählt normale Unterbrechungen alle 1/50 Sekunde und schaltet ab, wenn sie erfolgreich auftraten. Jedes Ausschalten des auf ein Ereignis bezogenen Zeitgebers stößt ein Ereignis an. Ein Block für normale Ereignisse ist 13 Bytes groß.

c) Bildrücklauf-Ereignisse

Ereignisse der Warteschlange für Bildrücklauf-Ereignisse werden bei jeder Bildrücklauf-Unterbrechung, d.h. alle 1/50 Sekunde für PAL- oder SECAM-Maschinen und alle 1/60 Sekunde für NSTC-Maschinen angestoßen. Ein Bildrücklauf-Block ist 9 Bytes groß.

11 Ereignisse

Der Ereignis-Mechanismus wird primär durch den Betriebssystem-Kern bereitgestellt, um die Bearbeitung von Unterbrechungen und anderen externen Ereignissen zu unterstützen. Der Mechanismus kann jedoch auch zur Bearbeitung interner Ereignisse in komplizierten Programmen (z.B. eine Simulation) verwendet werden. Ein Ereignis ist folgendermaßen charakterisiert:

a) **Ereignis-Klasse** (siehe Abschnitt 11.1)

Ereignisse können synchron oder asynchron, speziell oder normal sein.

b) **Ereignis-Priorität** (siehe Abschnitt 11.1)

Synchrone Ereignisse haben eine zugeordnete Priorität.

c) **Ereignis-Zahl** (siehe Abschnitt 11.2)

Bei jedem Auftreten eines Ereignisses wird die Zahl inkrementiert. Nach jeder Ereignis-Bearbeitung wird die Zahl dekrementiert. Das Ereignis kann durch Einstellung einer negativen Zahl deaktiviert werden.

d) **Ereignis-Routine** (siehe Abschnitt 11.3)

Die Adresse der zur Ereignis-Bearbeitung aufzurufenden Routine.

Ein Ereignis stellt sich dem Betriebssystem-Kern als ein Datenblock mit obigem Inhalt dar (siehe Anhang X für die exakte Belegung eines Ereignis-Blocks). Der Block muß in den zentralen 32KB des Speichers liegen, so daß der Betriebssystem-Kern unabhängig vom ROM-Freischaltzustand darauf zugreifen kann.

Tritt ein Ereignis auf, so wird der entsprechende Ereignis-Block durch Aufruf von KL EVENT angestoßen. Dieser Anstoß wird ignoriert, wenn die Ereignis-Zahl negativ ist. Ansonsten wird die Ereignis-Zahl inkrementiert (bis maximal 127) und die Ereignis-Routine zu einem Zeitpunkt, der von der Ereignis-Klasse abhängig ist, aufgerufen. Beim Rücksprung aus der Ereignis-Routine wird die Ereignis-Zahl dekrementiert, bis sie Null oder negativ geworden ist.

11.1 Ereignis-Klasse

Ereignisse sind entweder synchron oder asynchron. Asynchrone Ereignisse sind für die Bearbeitung externer Ereignisse vorgesehen, die meistens eine unmittelbare Bearbeitung erfordern. Die Bearbeitung asynchroner Ereignisse hat Vorrang gegenüber dem Haupt-Programm. Die Bearbeitung synchroner Ereignisse steht unter der Steuerung des Haupt-Programms, das sie im allgemeinen handhabt, sofern es günstig ist.

a) Asynchrone Ereignisse

Ein asynchrones Ereignis wird unmittelbar nach dem Ereignis-Anstoß bearbeitet, zumindest dann, wenn der Anstoß im Unterbrechungsbereich auftrat (siehe Abschnitt 10 bei Unterbrechungen). Der Betriebssystem-Kern ermöglicht keine Abstimmung zwischen asynchronen Ereignissen und dem Hauptprogramm oder anderen Ereignissen, so daß die Vermeidung von Wechselwirkungen beachtet werden sollte. Es ist nicht ratsam, Programme aufzurufen, die reentrant sind, wie z.B. die Bildschirm-Treiberroutinen der Firmware. Wenn die Ereignis-Zahl beim Rücksprung aus der Ereignis-Routine größer als Null ist, wird sie dekrementiert. Bleibt die Zahl größer als Null, so wird die Bearbeitung wiederholt (d.h. die Ereignis-Routine wird erneut aufgerufen und die Ereignis-Zahl dekrementiert), bis die Zahl Null oder negativ wird (siehe Abschnitt 11.2).

b) Synchrone Ereignisse

Synchrone Ereignisse werden nicht bearbeitet, wenn das Ereignis angestoßen ist, sondern in die Warteschlange für synchrone Ereignisse eingereiht, wo sie auf die Bearbeitung warten. Ereignisse werden in absteigender Prioritätsordnung eingereiht, Ereignisse gleicher Priorität nach denen, die sich bereits in der Warteschlange befinden.

Das Vordergrund-Programm sollte die Warteschlange für synchrone Ereignisse regelmäßig abfragen, um anstehende Ereignisse zu erkennen. Stehen Ereignisse an, dann sollte es sie bearbeiten. Der Unterschied zwischen synchronen und asynchronen Ereignissen ist, daß das Vordergrund-Programm den Zeitpunkt der Bearbeitung von synchronen Ereignissen bestimmt, während der Ereignis-Anstoß den Zeitpunkt der Bearbeitung von asynchronen Ereignissen festlegt. Unter der Voraussetzung, daß das Vordergrund-Programm korrekt arbeitet, dürften keine Schwierigkeiten in der Handhabung der Wechselwirkungen und Resourcen-Aufteilung zwischen synchronen Ereignissen und dem Vordergrund-Programm auftreten.

Wenn das Vordergrund-Programm feststellt, daß die Warteschlange für synchrone Ereignisse nicht leer ist, sollte es (ist aber nicht gezwungen) den Betriebssystem-Kern zur Bearbeitung des ersten Ereignisses in der Warteschlange veranlassen. Wenn eine synchrone Ereignis-Routine in Arbeit ist, erkennt der Betriebssystem-Kern die Priorität. In der Ereignis-Routine kann die Warteschlange für synchrone Ereignisse abgefragt werden; der Betriebssystem-Kern ignoriert jedoch alle Ereignisse, deren Priorität kleiner oder gleich der des momentan bearbeiteten Ereignisses ist. Beim Verlassen der Ereignis-Routine wird die vorhergehende Ereignis-Priorität wiederhergestellt, so daß die Verarbeitung von Ereignissen verschachtelt werden kann.

Die Prioritäten synchroner Ereignisse sind in zwei Gruppen eingeteilt, speziell und normal. Alle speziellen Ereignisse haben eine höhere Priorität als sämtliche normalen Ereignisse. Der Betriebssystem-Kern stellt einen Mechanismus zur Verfügung, um die Bearbeitung normaler Ereignisse zu sperren, ohne spezielle Ereignisse zu beeinflussen. Dies kann zur Implementierung „kritischer Regionen" dienen, über die normale Ereignisse sich gegenseitig beeinflußen können. Das durch den Abbruch-Mechanismus der Tastatur-Verwaltung angestoßene synchrone Ereignis ist ein Beispiel für ein spezielles synchrones Ereignis.

11.2 Ereignis-Zahl

Die Hauptaufgabe der Ereignis-Zahl ist die Überwachung der Differenz zwischen der Anzahl angestoßener Ereignise und der Anzahl bearbeiteter Ereignisse. Dies stellt sicher, daß ein auftretender Anstoß nicht verloren geht, bevor der vorhergende verarbeitet wurde. Die Ereignis-Zahl wird normalerweise inkrementiert, wenn das Ereignis angestoßen wurde und dekrementiert, wenn die Ereignis-Routine zurückkehrt. Die exakte Arbeitsweise hängt von der Ereignis-Zahl wie folgt ab:

Inkrementieren:

−128...−2:	Die Zahl wird nicht verändert, das Ereignis ignoriert.
−1:	Dieser Wert ist illegal.
0:	Die Zahl wird inkrementiert und die Ereignis-Bearbeitung entsprechend der Ereignis-Klasse eingeleitet.
1...126:	Die Zahl wird inkrementiert und keine weitere Aktion eingeleitet. Das Ereignis wartet auf die Bearbeitung eines vorhergehenden Anstoßes oder auf den Abschluß der Bearbeitung.
127:	Die Zahl wird nicht verändert, der Anstoß ignoriert.

Dekrementieren:

−128:	Dieser Wert ist illegal.
−127...0:	Die Zahl wird nicht verändert, das Ereignis wurde deaktiviert.
1:	Die Zahl wird dekrementiert und die Ereignis-Bearbeitung abgeschlossen.
2...127:	Die Zahl wird dekrementiert und die Ereignis-Bearbeitung fortgesetzt.

Man beachte, daß sich die Ereignis-Routine durch Setzen einer negativen Zahl selbst deaktivieren und unerwünschte Anstöße durch Setzen der Zahl auf 1 beseitigen kann.

11.3 Ereignis-Routine

Im allgemeinen wird die Adresse der Ereignis-Routine als 3-Byte-„Far-Address" angegeben (siehe Abschnitt 2 bei Speicherbelegung). Dadurch kann die Routine in jedem ROM oder irgendwo im RAM untergebracht werden.

Eine spezielle Form der Ereignis-Klasse kann die Routine so spezifizieren, als wäre sie auf einer „Near-Address" untergebracht. Dies verändert nicht den ROM-State. Somit muß die Routine entweder im unteren ROM oder in den zentralen 32KB RAM untergebracht werden. Das ROM-Select-Byte der „Far-Address" wird ignoriert und die anderen zwei Bytes als Adresse der Routine genommen. Der Aufruf einer „Near-Address"-Ereignis-Routine erfordert weniger Arbeit als der einer vollständigen „Far-Address". Ersterer wird durch die Firmware selbst verwendet.

11.4 Deaktivierende und reinitialisierende Ereignisse

Bevor ein Ereignis-Block reinitialisiert werden kann, muß das Ereignis deaktiviert sein. Dadurch wird sichergestellt, daß das Ereignis aus den verschiedenen Warteschlangen für anstehende Ereignisse entfernt und die Ereignis-Warteschlangen bei der Initialisierung des Ereignis-Blocks vor Zerstörung geschützt werden. Ein asynchrones Ereignis muß nicht durch seine Ereignis-Routine reinitialisiert werden, da in diesem Fall ein Deaktivieren des Ereignisses nicht zur Folge hat, daß es aus der Warteschlange für anstehende Unterbrechungen und Ereignisse entfernt wird.

Synchrone und asynchrone Ereignisse werden auf unterschiedliche Weise deaktiviert.

a) Asynchrone Ereignisse

Ein asynchrones Ereignis sollte durch Aufruf von KL DISARM EVENT deaktiviert werden. Sie setzt die Ereignis-Zahl auf einen negativen Wert (-64), wodurch Anstöße keine Wirkung haben. Befindet sich das Ereignis in der Warteschlange für anstehende Unterbrechungen und Ereignisse, so wird es nur dann beseitigt, wenn ein Versuch zur Bearbeitung des Ereignisses gemacht wurde und nicht unmittelbar, wenn das Ereignis deaktiviert wird.

b) Synchrone Ereignisse

Ein synchrones Ereignis sollte durch Aufruf von KL DEL SYNCHRONOUS deaktiviert werden. Sie setzt die Ereignis-Zahl auf einen negativen Wert (-64) und beseitigt den Ereignis-Block aus der Warteschlange für anstehende synchrone Ereignisse, sofern er sich in der Warteschlange befindet.

Die obigen Prozeduren schützen das Ereignis vor einem erfolgreichen Anstoß, jedoch nicht vor Anstoß-Versuchen. Ein schnelles, normales oder Bildrücklauf-Ereignis (siehe Abschnitt 4.5) befindet sich immer in seiner entsprechenden Warteschlange und empfängt durchaus reguläre Anstoß-Versuche. Um Zeitverschwendungen zu vermeiden (und dadurch das System zu verlangsamen), sollte das Ereignis durch Aufruf von KL DEL FAST TICKER, KL DEL FRAME FLY oder KL DEL TICKER aus der Warteschlange für Unterbrechungen entfernt werden.

12 Maschinen-Paket

Das Maschinen-Paket bearbeitet die Hardware auf unterster Ebene. Es übergibt Daten an den Centronics-Kanal (und von dort an den Drucker) und ist in der Lage, Lade- und Start-Programme zu bearbeiten.

12.1 Hardware-Schnittstellen

Die Routinen zum Treiben der Hardware sollten nur von denen, die ein Verständnis für die Hardware und die Arbeitsweise der Firmware besitzen, verwendet werden. Der Anwender sollte auf die Hardware nicht direkt zugreifen, wenn dafür eine Routine des Maschinen-Pakets bereitsteht.

Oftmals stehen Routinen höherer Ebene bereit, welche die gleichen Auswirkungen haben und zusätzlich die Firmware von der momentanen Einstellung informieren. Wenn möglich, sollten diese Routinen höherer Ebene statt der Maschinen-Paket-Routinen verwendet werden. Die Benutzung der Maschinen-Paket-Routinen kann die Firmware dazu verleiten, von fehlerhaften Annahmen über die momentane Einstellung auszugehen und entsprechend falsch zu reagieren.

Das Maschinen-Paket geht bei Zugriffen auf die Hardware von bestimmten Annahmen über den Zustand derselben aus. Es geht davon aus, daß sich der PPI-Kanal A im Ausgabe-Modus befindet und der Tongenerator-Chip, ULA, CRTC und Centronics-Kanal inaktiv sind. Es ist wichtig, beim direkten Zugriff auf die Hardware alle Unterbrechungen zu sperren.

Es gibt vier Hauptbereiche, welche das Maschinen-Paket bearbeitet:

a) Bildschirm

Drei Funktionen des Bildschirms können unter Verwendung der Maschinen-Paket-Routinen eingestellt werden. Diese sind Bildschirm-Modus (durch Aufruf von MC SET MODE), Bildschirm-Basis und -Offset (durch Aufruf von MC SET OFFSET).

Der Bildschirm-Modus bestimmt, wieviele Pixel auf dem Bildschirm dargestellt werden und wieviele Inks wie folgt verwendbar sind:

Modus	Auflösung	Inks
0	160 x 200	16
1	320 x 200	4
2	640 x 200	2

Die Bildschirm-Basis legt fest, welcher 16KB-Block des Speichers als Bildschirm-Speicher verwendet wird. Theoretisch kann #0000, #4000, #8000 oder #C000 verwendet werden, in der Praxis haben sich jedoch #4000 und #C000 als günstig erwiesen.

Der Bildschirm-Offset legt fest, welches Byte im Bildschirm-Speicher als erstes dargestellt wird. Die Änderung des Offsets bewegt den Inhalt des Bildschirms in einem Zug und wird zum Rollen des Bildschirms verwendet. Eine umfangreichere Beschreibung der Bildschirm-Belegung und sein Bezug auf diese Aspekte ist in Abschnitt 7 (Bildschirm-Paket) zu finden.

Wenn zum Beispiel bei der Benutzung eines Lichtgriffels Adressen vom CRT-Steuerbaustein gelesen werden sollen, ist zur Übersetzung der vom Chip gelesenen Bildschirmadressen in die aktuelle Bildschirmposition die Kenntnis der Speicheradressierung unerläßlich.

Das Maschinen-Paket stellt die Routine MC WAIT FLYBACK zur Verfügung, die solange wartet, bis der Bildrücklauf (der Beginn der vertikalen Strahlrückführungsperiode) erfolgt. Dadurch kann das Operieren auf dem Bildschirm mit geringstmöglicher Unterbrechung des Monitor-Bildes geschehen, da während dieser Periode kein Bild erzeugt wird. Alternativ zum Warten auf den Bildrücklauf, besteht die Möglichkeit, ein Bildrücklauf-Ereignis zu setzen (siehe Abschnitt 10.5).

Die vertikale Strahlrückführungsperiode ist nicht sehr lang. Ungefähr 100 Mikrosekunden nach ihrem Start wird eine Zeit-Unterbrechung erzeugt, welche die Bearbeitung des Bildrücklauf-Ereignisses einleitet (siehe Abschnitt 10). Dadurch wird eine gewisse Zeit der Rückführungsperiode in Anspruch genommen.

b) Inks

Das Maschinen-Paket bearbeitet die Einstellung der Ink-Farben. In Anhang 6.2. befindet sich eine etwas ausführliche Beschreibung der Beziehung zwischen Inks und Farben. Die Farbe für jede Ink und den Bildschirmrand kann unabhängig voneinander spezifiziert und geändert werden. Zu beachten ist jedoch, daß das Maschinen-Paket die Hardware-Repräsentation der Farben und nicht die Farben der Grauskala, die das Bildschirm-Paket verwendet, behandelt. Das Bildschirm-Paket spezifiziert ebenso die beiden wechselnden Farben beim Blinken der Inks.

Zwei Routinen sind zur Einstellung der Ink-Farben bereitgestellt. MC SET INKS ermöglicht die Einstellung der Farben aller 16 Inks und des Bildschirmrandes (obgleich nicht alle Inks im aktuellen Modus auf dem Bildschirm sichtbar sind). MC CLEAR INKS setzt den Bildschirmrand und alle 16 Inks auf die gleiche Farbe. Sie wird beim Löschen des Bildschirms verwendet, damit die Operation gleichzeitig und sofort sichtbar wird.

c) Tongenerator-Chip

Die Routine MC SOUND REGISTER wird bereitgestellt, um ein Register des Tongenerator-Chips zu beschreiben. Sie wird durch die Tongenerator-Verwaltung für Hardware-Zugriffe benutzt.

d) Centronics-Kanal

Es gibt zwei Routinen, um auf den Centronics-Kanal zuzugreifen. MC BUSY PRINTER prüft, ob er beschäftigt ist. MC SEND PRINTER sendet Daten über ihn. Daten sollten nicht ausgegeben werden, solange er beschäftigt ist.

Der Centronics-Kanal wird von den Drucker-Routinen verwendet, die Bestandteil des Maschinen-Pakets sind.

12.2 Drucker

Die Routine MC PRINT CHAR ruft eine Firmware-Indirection auf (MC WAIT PRINTER). Sie sendet Zeichen zum Drucker bzw. Centronics-Kanal.

MC WAIT PRINTER wartet, bis der Centronics-Kanal nicht mehr beschäftigt ist und sendet dann die übergebenen Zeichen. Wenn der Kanal für eine längere Zeit beschäftigt ist, wartet die Routine eine gewisse Zeit, kehrt zurück und zeigt an, daß die Übersendung des Zeichens nicht geklappt hat. Dieses Abwarten der Zeitperiode bewahrt Programme vor dem „Aufhängen", da sie auf eine Fertigmeldung (Ready) vom (möglicherweise nicht existierenden) Drucker warten.

MC WAIT PRINTER ermöglicht dem Anwender das Abfangen von zum Drucker zu sendenden Zeichen. Dadurch können gegebenenfalls spezielle Escape-Sequenzen eingefügt, der Drucker gesperrt oder die Zeitperiode geändert werden.

12.3 Laden und Ablauf der Programme

Das Maschinen-Paket stellt zwei Routinen – MC START PROGRAM und MC BOOT PROGRAM – zum Ablauf der Programme zur Verfügung.

MC START PROGRAM ist die einfachere der beiden Routinen. Sie reinitialisiert die gesamte Firmware und springt dann in das entsprechende Programm.

MC BOOT PROGRAM ist wesentlich komplexer. Sie lädt ein Programm in das RAM und läßt es ablaufen. Der Anwender übergibt eine Routine an MC BOOT PROGRAM, die das Program lädt und ihren Einsprung-Punkt übergibt. Vor dem Aufruf dieser Lade-Routine wird von der Firmware soviel wie möglich rückgesetzt, so daß der Speicherbereich zwischen #0040 und der Basis des Firmware-RAMs auf Adresse #B100 dem Anwender zur Verfügung steht. Wäre das System nicht rückgesetzt worden, könnte eine aktive Indirection, Ereignis- oder Unterbrechungs-Routine mit verheerenden Konsequenzen überschrieben werden.

Wird das Programm erfolgreich durch MC BOOT PROGRAM geladen, so ist die Firmware komplett initialisiert und in das Programm eingesprungen worden. Beim fehlerhaften Laden wird eine entsprechende Meldung ausgegeben und das vorherige Vordergrund-Programm erneut gestartet. War das vorherige Programm selbst eine RAM-Programm, so wird stattdessen in das Standard-ROM eingesprungen, da die Möglichkeit besteht, daß das vorherige Programm durch den Lade-Versuch des neuen Programms beeinflußt wurde.

13 Firmware-Sprungtabellen

Die Firmware stellt einige Sprungtabellen zur Verfügung, von denen die Haupt-Firmware-Sprungtabelle am größten ist. Sie dient den Programmen zum Zugriff auf die Firmware-Routinen im niederwertigen ROM. BASIC verwendet diese Sprungtabelle beispielsweise, die Firmware selbst benutzt jedoch diese Sprungtabelle nicht für ihre interne Kommunikation. Dies bedeutet, daß eine Änderung dieser Sprungtabelle keinen Einfluß auf das Verhalten der Firmware hat, sehr wohl aber auf BASIC.

Die zweitwichtigste Sprungtabelle ist die Indirections-Sprungtabelle. Indirections sind Sprünge, die durch die Firmware an Schlüsselpositionen verwendet werden. Sie ermöglichen dem Anwender die Änderung der Funktionsweise von Firmware-Routinen. Die Einträge in dieser Sprungtabelle dienen nicht Benutzer-Aufrufen, sondern sind nur durch die Firmware aufrufbar. Die Änderung einer Indirection ist eine Methode zur Beeinflussung des Firmware-Verhaltens.

Die verbleibenden zwei Sprungtabellen sind mit dem Betriebssystem-Kern verknüpft. Eine dient dem Anwender zum Aufruf verschiedener Betriebssystem-Kern-Routinen, die z.B. den ROM-State ändern u.ä. Die andere ist in diesem Sinne keine Sprungtabelle, sondern nur ein Bereich, in dem sich die Routinen auf Public-Adressen befinden (d.h. sie sind für andere zugänglich). Es sind im allgemeinen Hilfs-Routinen und Restarts. Diese Bereiche sollten durch den Anwender nicht geändert werden.

Die Routinen in dieser Sprungtabelle sind im folgenden aufgelistet. Umfangreichere Beschreibung zu diesen Routinen sind in den Abschnitten 14, 15 und 16 enthalten.

13.1 Haupt-Sprungtabelle

Die Haupt-Sprungtabelle der Firmware liegt im RAM zwischen den Adressen #BB00 und #BD39. Jeder Eintrag in der Sprungtabelle belegt drei Byte und ist für die Verwendung von LOW JUMP-Restarts (RST1) initialisiert. Sie schalten das niederwertige ROM frei, so daß die Firmware-Routinen ablaufen können, und sperren das höherwertige ROM, damit auf den Bildschirm-Speicher während der Firmware-Aktivitäten zugegriffen werden kann.

Nachdem die Sprungtabelle beim EMS aufgebaut wurde, wird sie von der Firmware nicht mehr verändert bis das System wieder initialisiert wird. Wenn irgendwelche Einträge geändert werden, liegt es in der Verantwortung des Anwenders, diese Änderungen rückgängig zu machen. Dies kann durch Aufruf von JUMP RESTORE erfolgen, wodurch die gesamte Sprungtabelle initialisiert wird.

13.1.1 Einträge für die Tastatur-Verwaltung

Die Tastatur-Verwaltung bearbeitet die Tastatur und die Joysticks.

INITIALISIERUNG			
0	#BB00	KM INITIALISE	Initialisiere die Tastatur-Verwaltung.
1	#BB03	KM RESET	Rücksetzen der Tastatur-Verwaltung, Löschen aller Puffer, Wiederherstellen der Standard-Tasten-Erweiterungen und Indirections.
ZEICHEN			
2	#BB06	KM WAIT CHAR	Warte auf nächstes Zeichen von der Tastatur.
3	#BB09	KM READ CHAR	Prüfe, ob ein Zeichen von der Tastatur da ist.
4	#BB0C	KM CHAR RETURN	Gib beim nächstenmal ein einzelnes Zeichen an die Tastatur.
5	#BB0F	KM SET EXPAND	Setze einen Erweiterungs-String.
6	#BB12	KM GET EXPAND	Hole ein Zeichen von einem Erweiterungs-String.
7	#BB15	KM EXP BUFFER	Richte einen Puffer für Erweiterungs-Strings ein.
TASTEN			
8	#BB18	KM WAIT KEY	Warte auf die nächste Taste von der Tastatur.
9	#BB1B	KM READ KEY	Prüfe, ob eine Taste von der Tastatur verfügbar ist.
10	#BB1E	KM TEST KEY	Prüfe, ob eine Taste gedrückt ist.
11	#BB21	KM GET STATE	Hole die Caps-Lock- und Shift-Lock-Zustände.
12	#BB24	KM GET JOYSTICK	Hole den momentanen Zustand der/des Joysticks.
ÜBERSETZUNGSTABELLEN			
13	#BB27	KM SET TRANSLATE	Setze den Eintrag in der Tasten-Übersetzungstabelle ohne Shift oder Control.

14	#BB2A	KM GET TRANSLATE	Hole den Eintrag aus der Tasten-Übersetzungstabelle ohne Shift oder Control.
15	#BB2D	KM SET SHIFT	Setze den Eintrag in der Tasten-Übersetzungstabelle, wenn die Shift-Taste gedrückt ist.
16	#BB30	KM GET SHIFT	Hole den Eintrag aus der Tasten-Übersetzungstabelle, wenn die Shift-Taste gedrückt ist.
17	#BB33	KM SET CONTROL	Setze den Eintrag in der Tasten-Übersetzungstabelle, wenn die Control-Taste gedrückt ist.
18	#BB36	KM GET CONTROL	Hole den Eintrag aus der Tasten-Übersetzungstabelle, wenn die Control-Taste gedrückt ist.
TASTENWIEDERHOLUNG			
19	#BB39	KM SET REPEAT	Setze, wenn eine Taste wiederholt werden kann.
20	#BB3C	KM GET REPEAT	Frage an, ob eine Taste wiederholt werden kann.
21	#BB3F	KM SET DELAY	Setze die Anfangsverzögerung und Repeat-Geschwindigkeit.
22	#BB42	KM GET DELAY	Hole die Anfangsverzögerung und die Repeat-Geschwindigkeit.
ABBRÜCHE			
23	#BB45	KM ARM BREAK	Erlaube, daß Abbruch-Ereignisse erzeugt werden.
24	#BB48	KM DISARM BREAK	Schütze vor der Erzeugung von Abbruch-Ereignissen.
25	#BB4B	KM BREAK EVENT	Erzeuge ein Abbruch-Ereignis, (sofern erlaubt).

13.1.2 Einträge für den Text-VDU

Der Text-VDU ist ein zeichenorientierter Bildschirm-Treiber.

INITIALISIERUNG			
26	#BB4E	TXT INITIALISE	Initialisiere den Text-VDU.

27	#BB51	TXT RESET	Setze den Text-VDU zurück – stelle die Standard-Indirections und Steuer-Code-Funktionen wieder her.
28	#BB54	TXT VDU ENABLE	Erlaube die Plazierung von Zeichen auf dem Bildschirm.
29	#BB57	TXT VDU DISABLE	Verbiete die Plazierung von Zeichen auf dem Bildschirm.
ZEICHEN			
30	#BB5A	TXT OUTPUT	Übergebe ein Zeichen oder einen Steuer-Code an den Text-VDU.
31	#BB5D	TXT WR CHAR	Schreibe ein Zeichen auf den Bildschirm.
32	#BB60	TXT RD CHAR	Lies ein Zeichen vom Bildschirm.
33	#BB63	TXT SET GRAPHIK	Schalte die Zeichen-Schreib-Option des Graphik-VDU ein oder aus.
FENSTER			
34	#BB66	TXT WIN ENABLE	Setze die Größe des aktuellen Bildschirm-Fensters.
35	#BB69	TXT GET WINDOW	Hole die Größe des aktuellen Bildschirm-Fensters.
36	#BB6C	TXT CLEAR WINDOW	Lösche das aktuelle Fenster.
CURSOR			
37	#BB6F	TXT SET COLUMN	Setze die horizontale Position des Cursors.
38	#BB72	TXT SET ROW	Setze die vertikale Position des Cursors.
39	#BB75	TXT SET CURSOR	Setze die Corsor Position.
40	#BB78	TXT GET CURSOR	Frage die aktuelle Cursor-Position ab.
41	#BB7B	TXT CUR ENABLE	Ermögliche die Cursor-Darstellung (Anwender).
42	#BB7E	TXT CUR DISABLE	Verbiete die Cursor-Darstellung (Anwender).
43	#BB81	TXT CUR ON	Ermögliche die Cursor-Darstellung (System).
44	#BB84	TXT CUR OFF	Verbiete die Cursor-Darstellung (System).

45	#BB87	TXT VALIDATE	Prüfe, ob eine Cursor-Position innerhalb des Fensters liegt.
46	#BB8A	TXT PLACE CURSOR	Gib ein Cursor-Symbol auf den Bildschirm.
47	#BB8D	TXT REMOVE CURSOR	Nimm ein Cursor-Symbol vom Bildschirm.
INKS			
48	#BB90	TXT SET PEN	Setze die Ink für das Schreiben von Zeichen.
49	#BB93	TXT GET PEN	Hole die Ink für das Schreiben von Zeichen.
50	#BB96	TXT SET PAPER	Setze die Ink für das Schreiben des Text-Hintergrunds.
51	#BB99	TXT GET PAPER	Hole die Ink für das Schreiben des Text-Hintergrunds.
52	#BB9C	TXT INVERSE	Vertausche die aktuellen Pen- und Paper-Inks.
53	#BB9F	TXT SET BACK	Erlaube oder verbiete, daß Hintergrund geschrieben wird.
54	#BBA2	TXT GET BACK	Frage, ob der Hintergrund beschrieben wird.
MATRIZEN			
55	#BBA5	TXT GET MATRIX	Hole die Adresse einer Zeichen-Matrix.
56	#BBA8	TXT SET MATRIX	Setze eine Zeichen-Matrix.
57	#BBAB	TXT SET M TABLE	Setze die Adresse einer Benutzer-definierten Matrix-Tabelle.
58	#BBAE	TXT GET M TABLE	Hole die Adresse einer Benutzer-definierten Matrix-Tabelle.
STEUER-CODES			
59	#BBB1	TXT GET CONTROLS	Hole die Adresse der Steuercode-Tabelle.
KANÄLE			
60	#BBB4	TXT STR SELECT	Selektiere einen Text-VDU-Kanal.
61	#BBB7	TXT SWAP STREAMS	Vertausche die Zustände zweier Kanäle.

13.1.3 Einträge für den Graphik-VDU

Der Graphik-VDU arbeitet mit einzelnen Pixeln.

INITIALISIERUNG			
62	#BBBA	GRA INITIALISE	Initialisiere den Graphik-VDU.
63	#BBBD	GRA RESET	Setze den Graphik-VDU zurück – stelle die Standard-Indirections wieder her.
AKTUELLE POSITION			
64	#BBC0	GRA MOVE ABSOLUTE	Bewege auf eine absolute Position.
65	#BBC3	GRA MOVE RELATIVE	Bewege relativ zur aktuellen Position.
66	#BBC6	GRA ASK CURSOR	Hole die aktuelle Position.
67	#BBC9	GRA SET ORIGIN	Setze den Ursprung der Anwender-Koordinaten.
68	#BBCC	GRA GET ORIGIN	Hole den Ursprung der Anwender-Koordinaten.
FENSTER			
69	#BBCF	GRA WIN WIDTH	Setze linke und rechte Ränder des Graphik-Fensters.
70	#BBD2	GRA WIN HEIGHT	Setze obere und untere Ränder des Graphik-Fensters.
71	#BBD5	GRA GET W WIDTH	Hole linke und rechte Ränder des Graphik-Fensters.
72	#BBD8	GRA GET W HEIGHT	Hole obere und untere Ränder des Graphik-Fensters.
73	#BBDB	GRA CLEAR WINDOW	Lösche das Graphik-Fenster.
INKS			
74	#BBDE	GRA SET PEN	Setze die darstellende Ink für Graphik.
75	#BBE1	GRA GET PEN	Hole die momentan darstellende Ink für Graphik.
76	#BBE4	GRA SET PAPER	Setze die Hintergrund-Ink für Graphik.
77	#BBE7	GRA GET PAPER	Hole die momentane Hintergrund-Ink für Graphik.

DARSTELLUNG

78	#BBEA	GRA PLOT ABSOLUTE	Stelle einen Punkt an einer absoluten Position dar.
79	#BBED	GRA PLOT RELATIVE	Stelle einen Punkt relativ zur momentanen Position dar.

TESTEN

80	#BBF0	GRA TEST ABSOLUTE	Teste einen Punkt an einer absoluten Position.
81	#BBF3	GRA TEST RELATIVE	Teste einen Punkt relativ zur momentanen Position.

ZEICHEN VON LINIEN

82	#BBF6	GRA LINE ABSOLUTE	Zeichne eine Linie zu einer absoluten Position.
83	#BBF9	GRA LINE RELATIVE	Zeichne eine Linie relativ zur momentanen Position.

ZEICHNEN VON ZEICHEN

84	#BBFC	GRA WR CHAR	Gib ein Zeichen an den Bildschirm an der momentanen Graphik-Position aus.

13.1.4 Einträge für das Bildschirm-Paket

Das Bildschirm-Paket ist die Schnittstelle zwischen Text- und Graphik-VDU einerseits und der Bildschirm-Hardware andererseits. Bildschirm-Funktionen, die sowohl Text als auch Graphik (d.h. Ink-Farben) betreffen, befinden sich im Bildschirm-Paket.

INITIALISIERUNG

85	#BBFF	SCR INITIALISE	Initialisiere das Bildschirm-Paket.
86	#BC02	SCR RESET	Setze das Bildschirm-Paket zurück – stelle die Standard-Indirections, Ink-Farben und Blinkraten wieder her.

BILDSCHIRM-HARDWARE

87	#BC05	SCR SET OFFSET	Setze den Offset für den Beginn des Bildschirms.
88	#BC08	SCR SET BASE	Setze den RAM-Bereich für den Bildschirm-Speicher.
89	#BC0B	SCR GET LOCATION	Hole die momentane Basis- und Offset-Einstellung.

MODUS

90	#BC0E	SCR SET MODE	Setze den Bildschirm in einen neuen Modus.
91	#BC11	SCR GET MODE	Hole den momentanen Bildschirm-Modus.
92	#BC14	SCR CLEAR	Lösche den Bildschirm (auf Ink 0).
93	#BC17	SCR CHAR LIMITS	Hole die Bildschirmgröße in Zeichen-Einheiten.

BILDSCHIRM-ADRESSEN

94	#BC1A	SCR CHAR POSITION	Übersetze physikalische Koordinaten in eine Bildschirm-Position.
95	#BC1D	SCR DOT POSITION	Übersetze Basis-Koordinaten in eine Bildschirm-Position.
96	#BC20	SCR NEXT BYTE	Gehe mit der Bildschirm-Adresse um ein Byte nach rechts.
97	#BC23	SCR PREV BYTE	Gehe mit der Bildschirm-Adresse um ein Byte nach links.
98	#BC26	SCR NEXT LINE	Gehe mit der Bildschirm-Adresse um ein Byte nach unten.
99	#BC29	SCR PREV LINE	Gehe mit der Bildschirm-Adresse um ein Byte nach oben.

INKS

100	#BC2C	SCR INK ENCODE	Codiere eine Ink zum Überstreichen aller Pixels in einem Byte.
101	#BC2F	SCR INK DECODE	Decodiere eine codierte Ink.
102	#BC32	SCR SET INK	Setze die zur Darstellung einer Ink erforderlichen Farben.
103	#BC35	SCR GET INK	Hole die Farben, in denen eine Ink momentan dargestellt wird.
104	#BC38	SCR SET BORDER	Setze die zur Darstellung des Bildschirmrands benötigten Farben.
105	#BC3B	SCR GET BORDER	Hole die Farben, in denen der Bildschirmrand momentan dargestellt wird.
106	#BC3E	SCR SET FLASHING	Setze die Blink-Perioden.
107	#BC41	SCR GET FLASHING	Hole die aktuellen Blink-Perioden.

VERSCHIEDENES

108	#BC44	SCR FILL BOX	Fülle einen Zeichen-Bereich des Bildschirms mit einer Ink.
109	#BC47	SCR FLOOD BOX	Fülle einen Byte-Bereich des Bildschirms mit einer Ink.
110	#BC4A	SCR CHAR INVERT	Invertiere eine Zeichen-Position.
111	#BC4D	SCR HW ROLL	Bewege den gesamten Bildschirm um acht Pixel-Zeilen (ein Zeichen) auf- oder abwärts.
112	#BC50	SCR SW ROLL	Bewege einen Bildschirm-Bereich um acht Pixel-Zeilen (ein Zeichen) auf- oder abwärts.
113	#BC53	SCR UNPACK	Expandiere eine Zeichen-Matrix für den momentanen Bildschirm-Modus.
114	#BC56	SCR REPACK	Komprimiere eine Zeichen Matrix auf das Standard-Format.
115	#BC59	SCR ACCESS	Setze den Bildschirm-Schreib-Modus für den Graphik-VDU.
116	#BC5C	SCR PIXELS	Schreibe ein Pixel auf den Bildschirm und ignoriere den Schreib-Modus des Graphik-VDU.

ZEICHNEN VON LINIEN

117	#BC5F	SCR HORIZONTAL	Zeichne eine genau horizontale Linie.
118	#BC62	SCR VERTICAL	Zeichne eine genau vertikale Linie.

13.1.5 Einträge für die Kassetten-Verwaltung

Die Kassetten-Verwaltung steuert das Lesen von Dateien vom Band und das Schreiben von Dateien auf das Band.

INITIALISIERUNG

119	#BC65	CAS INITIALISE	Initialisiere die Kassetten-Verwaltung, schließe alle Kanäle, setze die Standard-Geschwindigkeit und erlaube Meldungen.
120	#BC68	CAS SET SPEED	Setze die Aufzeichnungs-Geschwindigkeit.
121	#BC6B	CAS NOISY	Erlaube oder sperre Prompt-Meldungen.

MOTOR-STEUERUNG

122	#BC6E	CAS START MOTOR	Starte den Kassetten-Motor.
123	#BC71	CAS STOP MOTOR	Stoppe den Kassetten-Motor.
124	#BC74	CAS RESTORE MOTOR	Stelle den vorherigen Kassetten-Motor-Zustand wieder her.

LESEN VON DATEIEN

125	#BC77	CAS IN OPEN	Eröffne eine Datei für Eingabe.
126	#BC7A	CAS IN CLOSE	Schließe die Eingabe-Datei korrekt.
127	#BC7D	CAS IN ABANDON	Schließe die Eingabe-Datei unmittelbar.
128	#BC80	CAS IN CHAR	Lies ein Zeichen aus der Eingabe-Datei.
129	#BC83	CAS IN DIRECT	Lies die Eingabe-Datei in den Speicher.
130	#BC86	CAS RETURN	Gib das zuletzt gelesene Zeichen zurück.
131	#BC89	CAS TEST EOF	Wurde das Ende der Eingabe-Datei erreicht?

SCHREIBEN VON DATEIEN

132	#BC8C	CAS OUT OPEN	Eröffne eine Datei zur Ausgabe.
133	#BC8F	CAS OUT CLOSE	Schließe die Ausgabe-Datei korrekt.
134	#BC92	CAS OUT ABANDON	Schließe die Ausgabe-Datei unmittelbar.
135	#BC95	CAS OUT CHAR	Schreibe ein Zeichen in die Ausgabe-Datei.
136	#BC98	CAS OUT DIRECT	Schreibe die Ausgabe-Datei direkt vom Speicher.

KATALOG

137	#BC9B	CAS CATALOG	Erzeuge einen Katalog des Bandes.

RECORDS

138	#BC9E	CAS WRITE	Schreibe einen Satz auf Band.
139	#BCA1	CAS READ	Lies einen Satz vom Band.
140	#BCA4	CAS CHECK	Vergleiche einen Satz auf Band mit dem Speicherinhalt.

13.1.6 Einträge für die Tongenerator-Verwaltung

Die Tongenerator-Verwaltung steuert den Tongenerator-Chip.

INITIALISIERUNG			
141	#BCA7	SOUND RESET	Setze die Tongenerator-Verwaltung zurück, schalte den Tongenerator-Chip ab und lösche alle Ton-Warteschlangen.
TON-WARTESCHLANGEN			
142	#BCAA	SOUND QUEUE	Reihe einen Ton in die Ton-Warteschlange ein.
143	#BCAD	SOUND CHECK	Frage, ob in der Ton-Warteschlange Platz frei ist.
144	#BCB0	SOUND ARM EVENT	Setze ein Ereignis, welches abläuft, wenn eine Ton-Warteschlange nicht voll wird.
TÖNE			
145	#BCB3	SOUND RELEASE	Ermögliche die Ton-Erzeugung.
146	#BCB6	SOUND HOLD	Stoppe alle Töne mittendrin.
147	#BCB9	SOUND CONTINUE	Starte alle gestoppten Töne erneut.
HÜLLKURVEN			
148	#BCBC	SOUND AMPL ENVELOPE	Setze eine Amplituden-Hüllkurve.
149	#BCBF	SOUND TONE ENVELOPE	Setze eine Ton-Hüllkurve.
150	#BCC2	SOUND A ADDRESS	Hole die Adresse einer Amplituden-Hüllkurve.
151	#BCC5	SOUND T ADDRESS	Hole die Adresse einer Tonhüllkurve.

13.1.7 Einträge für den Betriebssystem-Kern

Der Betriebssystem-Kern behandelt synchrone und asynchrone Ereignisse. Er überwacht die Speicherbelegung und schaltet ROMs ein und aus. Neben den unten aufgelisteten Einträgen hat der Betriebssystem-Kern seine eigene Sprungtabelle und eine Reihe eigener Routinen, deren Adressen zugänglich sind. Diese gesonderten Einträge sind in Abschnitt 13.3 und 13.4 aufgelistet.

INITIALISIERUNG

152	#BCC8	KL CHOKE OFF	Setze den Betriebssystem-Kern zurück, lösche alle Ereignis-Warteschlangen etc.
153	#BCCB	KL ROM WALK	Finde und initialisiere alle Hintergrund-ROMs.
154	#BCCE	KL INIT BACK	Initialisiere ein bestimmtes Hintergrund-ROM.
155	#BCD1	KL LOG EXT	Führe eine RSX in die Firmware ein.
156	#BCD4	KL FIND COMMAND	Suche nach einer RSX, einem Hintergrund- oder Vordergrund-ROM zur Bearbeitung eines Befehls.

BILDRÜCKLAUF-LISTE

157	#BCD7	KL NEW FRAME FLY	Initialisiere und übergib einen Block an die Bildrücklauf-Liste.
158	#BCDA	KL ADD FRAME FLY	Übergib einen Block an die Bildrücklauf-Liste.
159	#BCDD	KL DEL FRAME FLY	Entferne einen Block aus der Bildrücklauf-Liste.

LISTE FÜR SCHNELLE TAKTE

160	#BCE0	KL NEW FAST TICKER	Initialisiere und übergib einen Block an die Liste für schnelle Takte.
161	#BCE3	KL ADD FAST TICKER	Übergib einen Block an die Liste für schnelle Takte.
162	#BCE6	KL DEL FAST TICKER	Entferne einen Block aus der Liste für schnelle Takte.

LISTE FÜR NORMALE TAKTE

163	#BCE9	KL ADD TICKER	Übergib einen Block an die Liste für normale Takte.
164	#BCEC	KL DEL TICKER	Entferne einen Block aus der Liste für normale Takte.

EREIGNISSE

165	#BCEF	KL INIT EVENT	Initialisiere einen Ereignis-Block.
166	#BCF2	KL EVENT	Stoße einen Ereignis-Block an.
167	#BCF5	KL SYNC RESET	Lösche die Warteschlange für synchrone Ereignisse.

168	#BCF8	KL DEL SYNCHRONOUS	Entferne ein synchrones Ereignis aus der Ereignis-Warteschlange.
169	#BCFB	KL NEXT SYNC	Hole das nächste Ereignis aus der Warteschlange.
170	#BCFE	KL DO SYNC	Bearbeite eine Ereignis-Routine.
171	#BD01	KL DONE SYNC	Beende die Bearbeitung eines Ereignisses.
172	#BD04	KL EVENT DISABLE	Sperre normale synchrone Ereignisse.
173	#BD07	KL EVENT ENABLE	Erlaube normale synchrone Ereignisse.
174	#BD0A	KL DISARM EVENT	Verhindere das Auftreten eines Ereignisses.
ABGELAUFENE ZEIT			
175	#BD0D	KL TIME PLEASE	Frage nach der abgelaufenen Zeit.
176	#BD10	KL TIME SET	Setze die abgelaufene Zeit.

13.1.8 Einträge für das Maschinen-Paket

Das Maschinen-Paket ist die Schnittstelle zu der Maschinen-Hardware. Die meisten Pakete verwenden das Maschinen-Paket zum Zugriff auf die Hardware. Die wichtigste Ausnahme ist die Kassetten-Verwaltung, die aus Geschwindigkeitsgründen ihre Hardware-Zugriffe selbst durchführt.

PROGRAMME			
177	#BD13	MC BOOT PROGRAMM	Lade und starte ein Vordergrund-Programm.
178	#BD16	MC START PROGRAMM	Starte ein Vordergrund-Programm.
BILDSCHIRM			
179	#BD19	MC WAIT FLYBACK	Warte auf Bildrücklauf.
180	#BD1C	MC SET MODE	Setze den Bildschirm-Modus.
181	#BD1F	MC SCREEN OFFSET	Setze den Bildschirm-Offset.
182	#BD22	MC CLEAR INKS	Setze alle Inks auf eine Farbe.
183	#BD25	MC SET INKS	Setze Farben für alle Inks.
DRUCKER			
184	#BD28	MC RESET PRINTER	Setze die Drucker-Indirection zurück.

185	#BD2B	MC PRINT CHAR	Versuche, ein Zeichen an den Centronics-Kanal zu senden.
186	#BD2E	MC BUSY PRINTER	Teste, ob der Centronics-Kanal beschäftigt ist.
187	#BD31	MC SEND PRINTER	Sende ein Zeichen an den Centronics-Kanal.

TONGENERATOR-CHIP

188	#BD34	MC SOUND REGISTER	Sende Daten an ein Register des Tongenerator-Chips.

13.1.9 Einträge für Jumper

Der Jumper baut die Haupt-Sprungtabelle auf.

INITIALISIERUNG

189	#BD37	JUMP RESTORE	Stelle die Standard-Sprungtabelle wieder her.

13.2 Firmware-Indirections

Die hier angeführten Firmware-Indirections werden an Schlüsselpositionen in der Firmware bereitgehalten. Der Anwender kann mit ihrer Hilfe Ersatz-Routinen für viele Firmware-Funktionen erstellen, ohne ein komplettes Firmware-Paket umplazieren zu müssen. Diese Indirections sollten nicht durch den Anwender aufgerufen werden, da normalerweise eine besser geeignete Routine höherer Ebene in der Haupt-Firmware-Sprungtabelle bereitsteht.

Die Indirections werden beim Aufruf der Reset-Routine (oder der Initialisierung) bzw. beim EMS durch die ihnen zugeordneten Pakete eingestellt. Sie werden ansonsten von der Firmware nicht geändert.

Die Indirections sind alle drei Bytes lang und verwenden Standard-Sprungbefehle (#C3). Wenn ein anderer ROM-Status (als obere ROMs gesperrt und untere ROMs freigeschaltet) erforderlich ist, kann der entsprechende Restart-Befehl ersetzt werden (siehe Abschnitt 2.3). Die Indirections befinden sich zwischen den Adressen #BDCD und #BDF3.

Auf dieser Betriebsebene werden sehr wenig Gültigkeitsprüfungen ausgeführt. Wenn unkorrekte Parameter übergeben werden oder eine Ersatz-Routine ein Register ungeachtet der dokumentierten Schnittstellen-Absprachen verfälscht, wird die Firmware nicht wie erwartet funktionieren.

Eine detaillierte Beschreibung dieser Routinen befindet sich in Abschnitt 15.

13.2.1 Text-VDU-Indirections

0	#BDCD	TXT DRAW CURSOR	Plaziere das Cursor-Symbol auf dem Bildschirm (wenn erlaubt).
1	#BDD0	TXT UNDRAW CURSOR	Entferne das Cursor-Symbol vom Bildschirm (wenn erlaubt).
2	#BDD3	TXT WRITE CHAR	Schreibe ein Zeichen auf den Bildschirm.
3	#BDD6	TXT UNWRITE	Lies ein Zeichen vom Bildschirm.
4	#BDD9	TXT OUT ACTION	Gib ein Zeichen oder Steuer-Code aus.

13.2.2 Graphik-VDU-Indirections

5	#BDDC	GRA PLOT	Stelle einen Punkt dar.
6	#BDDF	GRA TEST	Teste einen Punkt.
7	#BDE2	GRA LINE	Zeichne eine Linie.

13.2.3 Bildschirm-Pakekt-Indirections

8	#BDE5	SCR READ	Lies ein Pixel vom Bildschirm.
9	#BDE8	SCR WRITE	Schreibe (ein) Pixel auf den Bildschirm im momentanen Graphik-Schreib-Modus.
10	#BDEB	SCR MODE CLEAR	Lösche den Bildschirm auf Ink 0.

13.2.4 Tastatur-Verwaltung-Indirections

11	#BDEE	KM TEST BREAK	Teste auf Abbruch (oder Reset).

13.2.5 Maschinen-Paket-Indirections

12	#BDF1	MC WAIT PRINTER	Drucke ein Zeichen oder warte die Zeitperiode ab.

13.3 Die Sprungtabelle des oberen Betriebssystem-Kerns

Die Sprungtabelle des oberen Betriebssystem-Kerns ermöglicht es dem Anwender ROMs ein- und auszuschalten und auf Speicherbereiche zuzugreifen, die unter den ROMs liegen, während diese freigeschaltet sind. Die Einträge in dieser Sprungtabelle sind nicht nur Sprungbefehle; einige Einträge sind Startpunkte der Routinen. Der Anwender sollte keinen dieser Sprungtabellen-Einträge ändern. Die Sprungtabelle des oberen Betriebssystem-Kerns belegt den Speicher von Adresse #B900 aufwärts. Eine detaillierte Beschreibung dieser Routinen ist in Abschnitt 16 zu finden.

0	#B900	KL U ROM ENABLE	Schalte das aktuelle obere ROM ein.
1	#B903	KL U ROM DISABLE	Schalte das obere ROM aus.
2	#B906	KL L ROM ENABLE	Schalte das untere ROM ein.
3	#B909	KL L ROM DISABLE	Schalte das untere ROM aus.
4	#B90C	KL ROM RESTORE	Stelle den vorherigen ROM-Status wieder her.
5	#B90F	KL ROM SELECT	Selektiere ein bestimmtes oberes ROM.
6	#B912	KL CURR SELECTION	Frage, welches obere ROM momentan selektiert ist.
7	#B915	KL PROBE ROM	Frage nach Klasse und Version eines ROMs.
8	#B918	KL ROM DESELECT	Wiederherstellen des vorherigen oberen ROM-Select.
9	#B91B	KL LDIR	Verschiebe Speicherplatz (LDIR) bei gesperrten ROMs.
10	#B91E	KL LDDR	Verschiebe Speicherplatz (LDDR) bei gesperrten ROMs.
11	#B921	KL POLL SYNCHRONOUS	Prüfe, ob ein Ereignis mit höherer Priorität als das momentane ansteht.

13.4 Die Sprungtabelle des unteren Betriebssystem-Kerns

Der Kern stellt eine Reihe nützlicher Routinen im Speicherbereich zwischen #0000 und #003F zur Verfügung. Diese sind in einigen Fällen als Public-Routinen-Adresse und als Restart-Befehl verfügbar. Im allgemeinen stehen diese Routinen sowohl im ROM, als auch im RAM. Dadurch ist es unerheblich, ob das untere ROM freigeschaltet ist oder nicht. Weiterhin stehen einige Bereiche für den Anwender zur Verfügung, um die RST6-Befehle und Unterbrechungen der externen Hardware unterzubringen.

Der Anwender sollte die Sprungtabelle des unteren Betriebssystem-Kerns nicht verändern. Unter bestimmten Umständen kann es jedoch erforderlich sein, sie zu ändern. Es wäre denkbar, daß ein Programm den INTERRUPT ENTRY (durch Ändern des Jump auf Adresse #0038) oder den RESET ENTRY (durch Ändern der Bytes von Adresse #0000 bis #0007) abfangen muß. Wenn ein Programm irgendeinen Speicherplatz in dieser Sprungtabelle verändert (einen anderen, als die im USER RESTART- oder EXT INTERRUPT-Bereich befindlichen), liegt es im Verantwortungsbereich des Programms, dafür zu sorgen, daß das untere ROM freigeschaltet ist oder der Original-Inhalt wiederhergestellt wird, wenn irgendein anderes Programm abläuft. Insbesondere muß der Zustand beim Auftreten von Unterbrechungen durch das Programm bestimmt werden (daher die Notwendigkeit, den INTERRUPT ENTRY zu ändern).

Genauere Beschreibungen der Routinen in dieser Sprungtabelle befinden sich in Abschnitt 17.

#0000 RST 0	RESET ENTRY	Vollständiger Reset der Maschine, wie beim Einschalten.
#0008 RST 1	LOW JUMP	Spring in das untere ROM oder RAM, nimm eine implizite „LOW-Address" zum Einsprung.
#000B	KL LOW PCHL	Spring in das untere ROM oder RAM, HL beinhaltet die „Low-Address" zum Einsprung.
#000E	PCBC INSTRUCTION	Spring auf die Adresse in BC.
#0010 RST 2	SIDE CALL	Aufruf an ein „Sideways-ROM", nimmt eine implizite „Side-Address" zum Aufruf.
#0013	KL SIDE PCHL	Aufruf an ein „Sideways-ROM", HL beinhaltet die „Side-Address" zum Aufruf.
#0016	PCDE INSTRUCTION	Sprung auf die Adresse in DE.
#0018 RST 3	FAR CALL	Aufruf einer Routine im ROM oder RAM, nimmt eine implizite Adresse der „Far-Address" zum Aufruf.
#001B	KL FAR PCHL	Aufruf einer Routine im ROM oder RAM, C und HL beinhalten die „Far-Address" zum Aufruf.
#001E	PCHL INSTRUCTION	Sprung auf die Adresse in HL.
#0020 RST 4	RAM LAM	LD A, (HL) mit gesperrten ROMs.
#0023	KL FAR ICALL	Aufruf einer Routine im ROM oder RAM, HL zeigt auf die „Far-Address" zum Aufruf.

#0028 RST 5	FIRM JUMP	Sprung in das untere ROM, nimmt eine implizite Adresse als Sprungziel.
#0030 RST 6	USER RESTART	Die ROM-Version sichert den momentanen ROM-State in #002B, schaltet das untere ROM aus und springt in die RAM-Version. Die RAM-Version kann durch den Anwender zwischen #0030 und inkl. #0037 eingetragen werden.
#0038 RST 7	INTERRUPT ENTRY	Dieser Restart ist nicht verfügbar, da er für Unterbrechungen verwendet wird (Z80-Unterbrechungs-Modus 1).
#003B	EXT INTERRUPT	Wenn eine Unterbrechung auf dem Erweiterungskanal auftritt, ruft die Firmware Adresse #003B im RAM auf. Der Anwender sollte #003B bis inkl. #003F belegen, um dieses Auftreten abzufangen.

14 Die Firmware-Hauptsprungtabelle

Dieser Abschnitt beschreibt detailliert die Einsprung- und Aussprungbedingungen und die Auswirkungen aller Routinen der Firmware-Hauptsprungtabelle.
Die Firmware-Hauptsprungtabelle ist in Abschnitt 13.1 beschrieben.

Dem Anwender wird empfohlen, die entsprechenden Abschnitte für jedes Paket durchzulesen, bevor der Versuch unternommen wird, die Eingänge der Sprungtabelle zu verstehen.

Die jeweils zugehörigen Abschnitte:

Tastatur-Verwaltung	(KM)	Abschnitt 3
Text VDU	(TXT)	Abschnitt 4
Graphik VDU	(GRA)	Abschnitt 5
Bildschirmpaket	(SCR)	Abschnitt 6
Ton-Verwaltung	(SOUND)	Abschnitt 7
Kassetten-Verwaltung	(CAS)	Abschnitt 8
Betriebssystemkern	(KL)	Abschnitte 2, 9, 10 und 11
Maschinen Paket	(MC)	Abschnitt 12

Die oberste Zeile jeder Beschreibung hat folgendes Format:

«Einsprung-Nr.» «Einsprung-Name» «Einsprung-Adresse»

Die Einsprünge in der Tabelle sind beginnend bei 0 fortlaufend durchnumeriert. Die Einsprungadresse ist die Adresse, die aufgerufen werden muß, um die Firmware-Routine aufzurufen oder die Adresse der 3 Bytes, die überschrieben werden müssen, um die Routine abzufangen.

Die Einsprung-Adresse kann folgendermaßen berechnet werden:

Einsprung-Adrese = Beginn der Sprungtabelle + 3 ★ Einsprungnummer

Jeder Einsprung hat einen Namen, über den auf ihn in diesem Manual Bezug genommen wird.

Der letzte Abschnitt jeder Beschreibung ist eine Aufzählung verwandter Einsprünge. Dem Anwender wird empfohlen, diese Liste genau anzusehen, weil darin u.U. für das Problem besser geeignete Routinen enthalten sind.
Umgekehrt könnten diese Routinen mehr Klarheit darüber verschaffen, wie die Originalroutine beutzt werden sollte.

Die Beschreibung der Routinen beziehen sich auf die Standardroutine, auf die der Einsprung verzweigt.

Der Anwender kann den Einsprung verändern, was wiederum die Aktion der Routine verändert.

Dem Anwender wird dringend geraten, sich strikt an die beschriebenen Einsprung- und Aussprungbedingungen zu halten, da ansonsten andere Programme, die diese Routine aufrufen (z.B. BASIC) aufhören könnten, korrekt zu arbeiten.

0: KM INITIALISE #BB00

Initialisiere die Tastaturverwaltung (Key Manager).

Aktion:

Vollständige Initialisierung der Tastaturverwaltung, wie es während EMS geschieht. Alle variablen Puffer und Zuweisungsroutinen der Tastaturverwaltung werden initialisiert. Der vorausgegangene Zustand der Tastaturverwaltung ist verloren.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Die Indirection (Umweisung, mittelbarer Zugriff) der Tastaturverwaltung (KM TEST BREAK) wird auf die Standardroutine gesetzt.

Der Tastaturpuffer wird aufgebaut (und freigemacht).
Der Erweiterungspuffer wird aufgebaut und jeder Erweiterung (Funktion) werden die Standardwerte (siehe Anhang IV) zugewiesen.

Die Tastenübersetzungstabellen (siehe Anhang II) werden mit ihren Standardwerten initialisiert.

Das Verzeichnis der zulässigen Tasten mit Daueranschlag (siehe Anhang III) wird auf seinen Anfangsstatus gesetzt.

Die Wiederholungsgeschwindigkeiten werden auf ihren Standardwert gesetzt.

Die Umschalttaste (SHIFT) und die Dauerumschaltung (CAPS LOCK) werden auf „aus“ gestellt.

Die Routine KM DISARM BREAK (24) wird durchgeführt.

Diese Routine ist unterbrechbar.

Verwandte Einsprünge:

KM RESET

1: KM RESET #BB03

Rücksetzen der Tastaturverwaltung.

Aktion:

Neuinitialisierung der indirekten Sprungtabelle und der Puffer für die Tastaturverwaltung.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Die Sprungadresse in der indirekten Sprungtabelle für die Tastaturverwaltung (KM TEST BREAK, Kap. 15) wird auf die Standardroutine gesetzt.

Der Tastaturpuffer wird aufgebaut und ist leer.

Der Funktionszeichenpuffer wird aufgebaut und jedem Funktionszeichen wird sein Standardwert (siehe Anhang IV) zugewiesen.

Die Routine KM BREAK EVENT (25) wird gesperrt

Alle noch ausstehenden Tasten und Zeichen werden „vergessen"

Verwandte Einsprünge:

KM DISARM BREAK
KM EXP BUFFER
KM INITIALISE

2: KM WAIT CHAR #BB06

Warte auf das nächste Zeichen von der Tastatur.

Aktion:

Es wird versucht, ein Zeichen vom Tastaturpuffer oder aus der laufenden Funktionszeichenkette zu lesen. Diese Routine wartet solange, bis ein Zeichen verfügbar ist.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

Das CARRY-Flag ist eingeschaltet.
Alle anderen Flags sind zerstört.
A enthält das Zeichen.
Alle anderen Register bleiben erhalten.

Anmerkungen:

Die möglichen Quellen, aus denen das nächste Zeichen entnommen werden kann, sind, in der Reihenfolge, in der Sie getestet werden, folgende:

Das „Rückgabe"-Zeichen,
das nächste Zeichen in der Funktionszeichenkette,
das erste Zeichen in der Funktionszeichenkette,
ein Zeichen aus einer Tastenübersetzungstabelle (Anhang II).

Funktionszeichen in der Tastenübersetzungstabelle werden auf ihr(e) zugeordnete(s) Zeichen(kette) gesetzt (erweitert).
Funktionszeichen jedoch, die in solchen zugeordneten Zeichenketten enthalten sind, werden nicht mehr umgesetzt, sondern als eben dieses Zeichen behandelt.

Verwandte Einsprünge:

KM CHAR RETURN
KM READ CHAR
KM WAIT KEY

3: KM READ CHAR #BB09

Untersuche, ob von der Tastatur ein Zeichen eingegeben wurde.

Aktion:

Es wird versucht, ein Zeichen vom Tastaturpuffer oder der laufenden Funktionszeichenkette zu lesen. Diese Routine wartet nicht, wenn kein Zeichen zur Verfügung steht.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

Wenn ein Zeichen gefunden wurde:
CARRY-Flag ist eingeschaltet, alle anderen Flags sind zerstört.
A enthält das Zeichen.
Alle anderen Register bleiben erhalten.

Wenn kein Zeichen gefunden wurde:
CARRY-Flag ist ausgeschaltet,
alle anderen Flags sind zerstört.
A ist zerstört,
alle anderen Register bleiben erhalten.

Anmerkungen:

Die möglichen Quellen, aus denen das nächste Zeichen entnommen werden kann, sind in der Reihenfolge der Überprüfung folgende:
Das „Rückgabe"-Zeichen
das nächste Zeichen in der Funktionszeichenkette
das erste Zeichen in der Funktionszeichenkette
ein Zeichen aus einer Tastenübersetzungstabelle (Anhang II)

Funktionszeichen in der Tastenübersetzungstabelle werden auf ihr(e) zugeordnete(s) Zeichen(kette) gesetzt (erweitert).

Funktionszeichen jedoch, die in solchen zugeordneten Zeichenketten enthalten sind, werden nicht mehr umgesetzt, sondern als eben dieses Zeichen behandelt.

Diese Routine stellt, wenn möglich, immer ein Zeichen zur Verfügung. Durch wiederholten Aufruf von KM READ CHAR können somit die Puffer der Tastaturverwaltung solange entleert werden, bis kein Zeichen mehr vorhanden (CARRY-Flag ausgeschaltet) ist.

Verwandte Einsprünge:

KM CHAR RETURN
KM READ KEY
KM WAIT CHAR

4: KM CHAR RETURN #BB0C

Gib beim nächsten Mal ein einzelnes Zeichen an die Tastatur.

Aktion:

Sicherstellung eines Zeichens für den nächsten Aufruf vom KM READ CHAR oder KM WAIT CHAR.

Einsprung-Bedingungen:

A enthält das „Rückgabe"-Zeichen.

Aussprung-Bedingungen:

Alle Register und Flags bleiben erhalten.

Anmerkungen:

Das „Rückgabe"-Zeichen hat immer Vorzug vor jedem anderen Zeichen aus der Tastatureingabe. Es wird nicht untersucht, ob es sich um ein Funktionszeichen handelt und somit auch nicht umgesetzt, sondern übernommen wie es ist. Das „Rückgabe"-Zeichen muß nicht von der Tastatur eingelesen werden, sondern kann vom Anwender für jeden beliebigen Zweck eingefügt werden.

Es ist jedoch nur ein einziges „Rückgabe"-Zeichen möglich. Wenn die Routine ohne Lesen eines Zeichens zweimal aufgerufen wird, ist das erste „Rückgabe"-Zeichen verloren. Außerdem ist es nicht möglich, das Zeichen 255 als „Rückgabe"-Zeichen zu verwenden, da es als Kennzeichen für „kein Rückgabezeichen vorhanden" verwendet wird.

Verwandte Einsprünge:

KM READ CHAR
KM WAIT CHAR

5: KM SET EXPAND #BB0F

Setze einen Erweiterungsstring.

Aktion:

Angabe einer Funktionszeichenkette in Verbindung mit der zugeordneten Funktionszeichennummer.

Einsprung-Bedingungen:

B enthält die Funktionszeichennummer
C enthält die Länge der Zeichenkette
HL enthält die Adresse der Zeichenkette.

Aussprung-Bedingungen:

Wenn die Umsetzung richtig durchgeführt wurde:
CARRY-Flag eingeschaltet
alle anderen Flags zerstört.
A, BC, DE, HL zerstört
alle anderen Register bleiben erhalten.

Wenn das Funktionszeichen ungültig war oder die Zeichenkette zu lang war:
CARRY-Flag ausgeschaltet
ansonsten wie oben

Anmerkungen:

Die Zeichenkette kann überall im RAM liegen.

Funktionszeichenketten können nicht direkt vom ROM übernommen werden.

Die einzelnen Zeichen einer Zeichenkette werden nicht mehr daraufhin untersucht, ob es sich um ein Funktionszeichen handelt; deshalb kann jedes Zeichen in eine Zeichenkette aufgenommen werden.

Wenn im Funktionszeichenpuffer nicht genügend Platz für die neue Zeichenkette zur Verfügung steht, wird keine Umsetzung vorgenommen.

Wenn die zu setzende Zeichenkette gerade benutzt wird, Zeichen zu erzeugen (durch KM READ CHAR oder KM WAIT CHAR), dann wird der ungelesene Teil der Zeichenkette zerstört. Das nächste Zeichen wird dann vom Tastenpuffer geholt.

Verwandte Einsprünge:

KM GET EXPAND
KM READ CHAR
KM WAIT CHAR

6: KM GET EXPAND #BB12

Hole ein Zeichen von einem Erweiterungsstring.

Aktion:

Lesen eines Zeichens aus einer Funktionskette. Die Zeichen in der Zeichenkette sind, beginnend bei Null, fortlaufend durchnumeriert.

Einsprung-Bedingungen:

A enthält ein Funktionszeichen
L enthält die Zeichennummer

Aussprung-Bedingungen:

Wenn das Zeichen gefunden wurde:
CARRY-Flag eingeschaltet
alle anderen Flags sind zerstört
A enthält das Zeichen
DE ist zerstört
alle anderen Register bleiben erhalten.

Wenn das Zeichen in Register A ungültig oder die Zeichenkette nicht lang genug war:
CARRY-Flag ausgeschaltet
alle anderen Flags zerstört
Register A und DE zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die Zeichen in der Funktionszeichenkette werden nicht mehr umgesetzt. Es kann daher jedes beliebige Zeichen in die Funktionszeichenkette eingestellt werden.

Verwandte Einsprünge:

KM READ CHAR
KM SET EXPAND

7: KM EXP BUFFER #BB15

Richte einen Puffer für Erweiterungs-Strings ein.

Aktion:

Bestimmen der Adresse und der Länge des Funktionspuffers. Initialisierung des Puffers mit den Standardfunktionszeichenketten.

Einsprung-Bedingungen:

DE enthält die Pufferadresse
HL enthält die Pufferlänge

Aussprung-Bedingungen:

Wenn der Puffer angelegt wurde:
CARRY-Flag eingeschaltet

Wenn der Puffer zu klein ist:
CARRY-Flag ausgeschaltet

In jedem Fall:
A, BC, DE, HL und andere Flags zerstört
alle anderen Register unverändert.

Anmerkungen:

Der Puffer darf nicht unterhalb eines ROM angelegt werden und muß mindestens 44 Bytes lang sein (d.h. genügend Platz, um die Standardfunktionszeichen aufzunehmen). Wenn der neue Puffer zu klein ist, bleibt der alte Puffer unverändert.

Die standardmäßigen Funktionszeichen sind in Anhang IV aufgelistet.

Jede Funktionszeichenkette, die gerade gelesen wird, wird zerstört.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KM GET EXPAND
KM SET EXPAND

8: KM WAIT KEY #BB18

Warte auf die nächste Tastatureingabe.

Aktion:

Es wird versucht aus dem Tastenpuffer zu lesen. Diese Routine wartet, bis eine Taste eingegeben wird, wenn nicht sofort ein Zeichen verfügbar ist.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

CARRY-Flag eingeschaltet
alle anderen Flags zerstört
A enthält das Zeichen oder die Funktionszeichennummer
alle anderen Register bleiben erhalten

Anmerkungen:

Das nächste Tastenzeichen wird vom Tastenpuffer gelesen und mittels der zugeordneten Tastenübersetzungstabelle übersetzt. Funktionszeichen werden nicht umgesetzt, sondern wie normale Zeichen an den Anwender zur weiteren Verarbeitung durchgereicht. Andere Funktionszeichen der Tastaturverwaltung (SHIFT LOCK, CAPS LOCK und IGNORE) werden beachtet, aber nicht durchgereicht.

Verwandte Einsprünge:

KM READ KEY
KM WAIT CHAR

9: KM READ KEY #BB1B

Untersuche, ob ein Zeichen von der Tastatur zur Verfügung steht.

Aktion:

Versuch, aus dem Tastenpuffer ein Zeichen zu lesen. Diese Routine wartet nicht, wenn kein Zeichen sofort verfügbar ist.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

Wenn ein Zeichen verfügbar war:
CARRY-Flag eingeschaltet
A enthält das Zeichen oder das Funktionszeichen

Wenn kein Zeichen verfügbar war:
CARRY-Flag ausgeschaltet
A zerstört

In jedem Fall:
alle anderen Flags zerstört
alle anderen Register unverändert.

Anmerkungen:

Das nächste Zeichen wird aus dem Tastenpuffer gelesen und mittels der dazugeordneten Tastenübersetzungstabelle übersetzt. Funktionszeichen werden nicht umgesetzt, sondern wie normale Zeichen an den Anwender zur weiteren Verarbeitung durchgereicht. Andere Funktionszeichen der Tastaturverwaltung (SHIFT LOCK, CAPS LOCK und IGNORE) werden beachtet, aber nicht durchgereicht.

Diese Routine stellt, wenn möglich, immer ein Zeichen zur Verfügung. Durch wiederholten Aufruf von KM READ KEY kann somit der Tastenpuffer solange entleert werden, bis schließlich kein Zeichen mehr zur Verfügung steht. Es ist jedoch zu beachten, daß das „Rückgabe"-Zeichen oder eine teilweise gelesene Funktionszeichenkette ignoriert wird. Es wird daher empfohlen, KM READ CHAR aufzurufen, wenn die Puffer der Tastaturverwaltung geleert werden sollen.

Verwandte Einsprünge:

KM READ CHAR
KM WAIT KEY

10: KM TEST KEY #BB1E

Untersuche, ob eine Taste gedrückt ist.

Aktion:

Untersuche, ob eine bestimmte Taste oder ein Knopf des Joysticks gedrückt wurde. Dies geschieht über die Tastenzustandstabelle und nicht durch Untersuchung der Tastatur.

Einsprung-Bedingungen:

A enthält eine Tastennummer

Aussprung-Bedingungen:

Wenn die Taste gedrückt wurde:
Zero-Flag ist aus

Wenn die Taste nicht gedrückt wurde:
Zero-Flag ist an

In jedem Fall:
CARRY-Flag ist ausgeschaltet
C enthält den momentanen Stand der SHIFT- und Kontrolltaste
A, HL und alle anderen Flags sind zerstört
alle anderen Register unverändert.

Anmerkungen:

Wenn eine Taste erkannt wurde, wird der Stand der SHIFT- und Kontrolltaste automatisch gelesen.
Wenn Bit 7 gesetzt ist, wurde die Kontrolltaste gedrückt und wenn Bit 5 gesetzt ist, wurde eine der SHIFT-Tasten gedrückt.

Die Tastennummer wird nicht überprüft. Eine ungültige Tastennummer liefert den richtigen Stand der SHIFT- und Kontrolltaste, der Stand der untersuchten Taste ist jedoch bedeutungslos. Die Tastenzustandstabelle, die von dieser Routine untersucht wird, wird von der Tastaturuntersuchungsroutine immer wieder auf den neuesten Stand gebracht. Dies geschieht normalerweise alle 1/50 Sekunde und so kann der Zustand sich laufend verändern. Das Loslassen einer Taste wird vermerkt, wenn 2 solcher oben genannter Untersuchungszyklen abgelaufen sind, während das Drücken einer Taste sofort entdeckt wird.

Verwandte Einsprünge:

KM GET JOYSTICK
KM GET STATE
KM READ KEY

11: KM GET STATE #BB21

Hole die CAPS-LOCK und SHIFT-LOCK-Zustände.

Aktion:

Abfrage auf den momentanen Stand der Tastatur bezüglich der CAPS-LOCK- und SHIFT-LOCK-Taste.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

L enthält den Stand von SHIFT LOCK
H enthält den Stand von CAPS LOCK
AF zerstört
alle anderen Register unverändert.

Anmerkungen:

Die Zustände bedeuten:

#00	Umschaltung ist aus
#FF	Umschaltung ist an

Der Standardwert ist jeweils #00

Verwandte Einsprünge:

KM TEST KEY

12: KM GET JOYSTICK #BB24

Hole den momentanen Zustand der/des Joysticks.

Aktion:

Abfrage auf den momentanen Zustand der Joysticks.
Dies geschieht durch Lesen der Tastenzustandstabelle und nicht durch Zugriff auf die Tastaturhardware.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

H enthält den Stand von Joystick 0
L enthält den Stand von Joystick 1
A enthält den Stand von Joystick 0
alle Flags sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Im Normalfall wird die Tastenzustandstabelle alle 1/50-Sekunden durch eine „Tasten-untersuchungsroutine" auf den neuesten Stand gebracht, so daß der zurückgegebene Zustand nicht mehr ganz dem aktuellen Stand entspricht.

Die Zustände der Joysticks sind, wie folgt, den einzelnen Bits zu entnehmen:

Bit 0	Auf
Bit 1	Ab
Bit 2	Links
Bit 3	Rechts
Bit 4	Feuer 2
Bit 5	Feuer 1
Bit 6	Reserve (normalerweise nicht angeschlossen)
Bit 7	immer Null

Wenn ein Bit gesetzt ist, wurde der zugeordnete Knopf gedrückt.

Der Joystick 2 ist von bestimmten Tasten auf der Tastatur nicht zu unterscheiden (siehe Anhang I).

Verwandte Einsprünge:

KM TEST KEY

13: KM SET TRANSLATE #BB27

Setze den Eintrag in der normalen Tastenübersetzungstabelle.

Aktion:

Angabe, welches Zeichen oder welche Funktionstaste für eine Taste eingetragen werden soll, wenn weder eine Shift- noch die Kontrolltaste gedrückt sind.

Einsprung-Bedingungen:

A enthält eine Tastennummer
B enthält die neue Übersetzung

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Wenn eine Tastennummer ungültig ist (größer als 79), erfolgt keine Durchführung.

Die meisten Werte der Tabelle werden als Zeichen behandelt und an den Anwender zurückgegeben.

Es gibt jedoch auch bestimmte Spezialwerte:

#80...#9F	sind Funktionszeichen, die zu Zeichenketten erweitert werden, wenn sie über KM READ CHAR oder KM WAIT CHAR aufgerufen werden, während sie bei Aufruf über KM READ KEY oder KM WAIT KEY wie jedes andere Zeichen durchgereicht werden
#FD	ist die CAPS LOCK-Funktion (Dauerumschaltung auf Großbuchstaben) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FE	ist die SHIFT LOCK-Funktion (Dauerumschaltung auf obere Tastaturzeichen) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FF	ist die IGNORE-Funktion (Ignorieren einer Taste) und bedeutet, daß die Taste nicht vorhanden sein soll.

Die Zeichen #E0...#FC haben für BASIC eine spezielle Bedeutung in Verbindung mit Zeilenaufbereitung, Setzen des Cursors und Unterbrechungen.

Im Anhang II finden Sie eine vollständige Auflistung der Standardumsetzungstabellen.

Verwandte Einsprünge:

KM GET TRANSLATE
KM SET CONTROL
KM SET SHIFT

14: KM GET TRANSLATE #BB2A

Hole einen Eintrag aus der normalen Tastenübersetzungstabelle.

Aktion:

Abfrage, in welches Zeichen oder Funktionszeichen eine Taste umgesetzt werden soll, wenn weder die SHIFT- noch die Control-Taste gedrückt ist.

Einsprung-Bedingungen:

A enthält eine Tastennummer

Aussprung-Bedingungen:

A enthält die momentane Übersetzung
HL und alle Flags sind zerstört
Alle anderen Register bleiben erhalten.

Anmerkungen:

Die Tastennummer wird nicht geprüft. Wenn sie ungültig ist (größer als 79), ist die zurückgegebene Übersetzung bedeutungslos.

Die meisten Werte der Tabelle werden als Zeichen behandelt und an den Anwender zurückgegeben. Es gibt jedoch bestimmte Spezialwerte:

#80...#9F	sind Funktionszeichen, die zu Zeichenketten erweitert werden, wenn sie über KM READ CHAR oder KM WAIT CHAR aufgerufen werden, während sie bei Aufruf über KM READ KEY oder KM WAIT KEY wie jedes andere Zeichen durchgereicht werden.
#FD	ist die CAPS LOCK-Funktion (Dauerumschaltung auf Großbuchstaben) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FE	ist die SHIFT LOCK-Funktion (Dauerumschaltung auf obere Tastaturzeichen) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FF	ist die IGNORE-Funktion (Ignorieren einer Taste) und bedeutet, daß die Taste nicht vorhanden sein soll.

Die Zeichen #E0...#FC haben für BASIC eine spezielle Bedeutung in Verbindung mit Zeilenaufbereitung, Setzen des Cursors und Unterbrechungen.

Im Anhang II finden Sie eine vollständige Auflistung der Standardumsetzungstabellen.

Verwandte Einsprünge:

KM GET CONTROL
KM GET SHIFT
KM SET TRANSLATE

15: KM SET SHIFT #BB2D

Setze einen Eintrag in die Tastenübersetzungstabelle mit Umschaltung.

Aktion:

Angabe, in welches Zeichen oder Funktionszeichen eine Taste übersetzt werden soll, wenn die Control-Taste nicht gedrückt, jedoch die Umschalttaste (SHIFT) oder Dauerumschaltung (SHIFT LOCK) an ist.

Einsprung-Bedingungen:

A enthält eine Tastennummer
B enthält die neue zugeordnete Übersetzung.

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Wenn die Tastennummer ungültig ist (größer als 79), wird keine Aktion durchgeführt.

Die meisten Werte der Tabelle werden als Zeichen behandelt und an den Anwender zurückgegeben. Es gibt jedoch bestimmte Spezialwerte:

#80...#9F	sind Funktionszeichen, die zu Zeichenketten erweitert werden, wenn sie über KM READ CHAR oder KM WAIT CHAR aufgerufen werden, während sie bei Aufruf über KM READ KEY oder KM WAIT KEY wie jedes andere Zeichen durchgereicht werden.
#FD	ist die CAPS LOCK-Funktion (Dauerumschaltung auf Großbuchstaben) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FE	ist die SHIFT LOCK-Funktion (Dauerumschaltung auf obere Tasaturzeichen) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FF	ist die IGNORE-Funktion (Ignorieren einer Taste) und bedeutet, daß die Taste nicht vorhanden sein soll.

Die Zeichen #E0...#FC haben für BASIC eine spezielle Bedeutung in Verbindung mit Zeilenaufbereitung, Setzen des Cursors und Unterbrechungen.

Im Anhang II finden Sie eine vollständige Auflistung der Standardumsetzungstabellen.

Verwandte Einsprünge:

KM GET SHIFT
KM SET CONTROL
KM SET TRANSLATE

16: KM GET SHIFT #BB30

Hole einen Eintrag aus der Tastenübersetzungstabelle unter Berücksichtigung der Umschaltung.

Aktion:

Abfrage, in welches Zeichen oder Funktionszeichen eine Taste umgesetzt werden soll, wenn die Control-Taste nicht gedrückt ist, jedoch die Umschalttaste (SHIFT) gedrückt oder die Dauerumschaltung (SHIFT LOCK) an ist.

Einsprung-Bedingungen:

A enthält eine Tastennummer

Aussprung-Bedingungen:

A enthält die zugeordnete Übersetzung
HL und Flags sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Wenn die Tastennummer ungültig ist (größer als 79), wird keine Aktion durchgeführt.

Die meisten Werte der Tabelle werden als Zeichen behandelt und an den Anwender zurückgegeben. Es gibt jedoch bestimmte Spezialwerte:

#80...#9F	sind Funktionszeichen, die zu Zeichenketten erweitert werden, wenn sie über KM READ CHAR oder KM WAIT CHAR aufgerufen werden, während sie bei Aufruf über KM READ KEY oder KM WAIT KEY wie jedes andere Zeichen durchgereicht werden
#FD	ist die CAPS LOCK-Funktion (Dauerumschaltung auf Großbuchstaben) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FE	ist die SHIFT LOCK-Funktion (Dauerumschaltung auf obere Tastaturzeichen) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FF	ist die IGNORE-Funktion (Ignorieren einer Taste) und bedeutet, daß die Taste nicht vorhanden sein soll.

Die Zeichen #E0...#FC haben für BASIC eine spezielle Bedeutung in Verbindung mit Zeilenaufbereitung, Setzen des Cursors und Unterbrechungen.
Im Anhang II finden Sie eine vollständige Auflistung der Standardumsetzungstabellen.

Verwandte Einsprünge:

KM GET CONTROL
KM GET TRANSLATE
KM SET SHIFT

17: KM SET CONTROL #BB33

Setze einen Eintrag in der Tastenübersetzungstabelle mit Control-Taste.

Aktion:

Angabe, in welches Zeichen oder Funktionszeichen eine Taste übersetzt werden soll, wenn die Control-Taste gedrückt ist.

Einsprung-Bedingungen:

A enthält eine Tastennummer
B enthält die neue zugeordnete Übersetzung

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Wenn die Tastennummer ungültig ist (größer als 79), wird keine Aktion durchgeführt.

Die meisten Werte der Tabelle werden als Zeichen behandelt und an den Anwender zurückgegeben. Es gibt jedoch bestimmte Spezialwerte:

#80...#9F	sind Funktionszeichen, die zu Zeichenketten erweitert werden, wenn sie über KM READ CHAR oder KM WAIT CHAR aufgerufen werden, während sie bei Aufruf über KM READ KEY oder KM WAIT KEY wie jedes andere Zeichen durchgereicht werden
#FD	ist die CAPS LOCK-Funktion (Dauerumschaltung auf Großbuchstaben) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FE	ist die SHIFT LOCK-Funktion (Dauerumschaltung auf obere Tastaturzeichen) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FF	ist die IGNORE-Funktion (Ignorieren einer Taste) und bedeutet, daß die Taste nicht vorhanden sein soll.

Die Zeichen #E0...#FC haben für BASIC eine spezielle Bedeutung in Verbindung mit Zeilenaufbereitung, Setzen des Cursors und Unterbrechungen.

Im Anhang II finden Sie eine vollständige Auflistung der Standardumsetzungstabellen.

Verwandte Einsprünge:

KM GET CONTROL
KM SET SHIFT
KM SET TRANSLATE

18: KM GET CONTROL #BB36

Hole einen Eintrag aus der Tastenübersetzungstabelle unter Berücksichtigung der Control-Taste.

Aktion:

Angabe, in welches Zeichen oder Funktionszeichen eine Taste umgesetzt werden soll, wenn die Control-Taste gedrückt ist.

Einsprung-Bedingungen:

A enthält eine Tastennummer

Aussprung-Bedingungen:

A enthält die zugeordnete Übersetzung
HL und Flags sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Wenn die Tastennummer ungültig ist (größer als 79), wird keine Aktion durchgeführt.

Die meisten Werte der Tabelle werden als Zeichen behandelt und an den Anwender zurückgegeben. Es gibt jedoch bestimmte Spezialwerte:

#80...#9F	sind Funktionszeichen, die zu Zeichenketten erweitert werden, wenn sie über KM READ CHAR oder KM WAIT CHAR aufgerufen werden, während sie bei Aufruf über KM READ KEY oder KM WAIT KEY wie jedes andere Zeichen durchgereicht werden.
#FD	ist die CAPS LOCK-Funktion (Dauerumschaltung auf Großbuchstaben) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FE	ist die SHIFT LOCK-Funktion (Dauerumschaltung auf obere Tastaturzeichen) und bewirkt, daß diese umgeschaltet wird („an", wenn „aus" und umgekehrt)
#FF	ist die IGNORE-Funktion (Ignorieren einer Taste) und bedeutet, daß die Taste nicht vorhanden sein soll.

Die Zeichen #E0...#FC haben für BASIC eine spezielle Bedeutung in Verbindung mit Zeilenaufbereitung, Setzen des Cursors und Unterbrechungen.

Im Anhang II finden Sie eine vollständige Auflistung der Standardumsetzungstabellen.

Verwandte Einsprünge:

KM GET SHIFT
KM GET TRANSLATE
KM SET CONTROL

19: KM SET REPEAT #BB39

Setze die Wiederholungsfunktion für eine Taste.

Aktion:

Setzen eines Eintrags in der Wiederholungstastentabelle, der bestimmt, ob für diese Taste eine Wiederholung erlaubt ist oder nicht.

Einsprung-Bedingungen:

Wenn eine Wiederholung erlaubt ist:

B enthält #FF

Wenn keine Wiederholung erlaubt ist:

B enthält #00

Immer:

A enthält die Tastennummer

Aussprung-Bedingungen:

AF, BC und HL sind zerstört
alle anderen Register unverändert.

Anmerkungen:

Wenn eine Tastennummer ungültig ist (größer als 79), wird keine Aktion durchgeführt.

Die Standardwiederholungstasten sind in Anhang III aufgelistet.

Verwandte Einsprünge:

KM GET REPEAT
KM SET DELAY

20: KM GET REPEAT #BB3C

Abfrage, ob es sich um eine Wiederholungstaste handelt.

Aktion:

Untersuchung des Eintrags in der Wiederholungstastentabelle, der aussagt, ob eine Wiederholung erlaubt ist oder nicht.

Einsprung-Bedingungen:

A enthält eine Tastennummer

Aussprung-Bedingungen:

Wenn die Taste eine Wiederholung erlaubt:

Zero-Flag ist „aus".

Wenn die Taste keine Wiederholung erlaubt:

Zero-Flag ist „an"

Immer:

Carry-Flag ist „aus"
A, HL und die anderen Flags zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die Tastennummer wird nicht geprüft. Wenn sie ungültig ist (größer als 79), ist der zurückgegebene Wiederholungsstatus bedeutungslos.

Die Standardwiederholungstasten sind in Anhang III aufgelistet.

Verwandte Einsprünge:

KM SET REPEAT

21: KM SET DELAY #BB3F

Setze die Anfangsverzögerung und die Wiederholungsgeschwindigkeit.

Aktion:

Setze die Zeitspanne vor der ersten Tastenwiederholung (Anfangsverzögerung) und die Zeitspanne zwischen den Wiederholungen (Wiederholungsgeschwindigkeit).

Einsprung-Bedingungen:

H enthält die neue Anfangsverzögerung
L enthält die neue Wiederholungsgeschwindigkeit.

Aussprung-Bedingungen:

AF zerstört
alle anderen Register unverändert.

Anmerkungen:

Beide Werte werden in Einheiten von 1/50 Sek. angegeben, wobei der Wert 0 jeweils 256 bedeutet.

Der Standardwert für die Anfangsverzögerung ist 30 Einheiten (0,6 Sek.) und der Standardwert für die Wiederholungsgeschwindigkeit ist 2 Einheiten (0,04 Sek. oder 25 Zeichen pro Sek.)

Es ist zu beachten, daß eine Taste von der Wiederholung ausgeschlossen ist, solange der Tastenpuffer nicht leer ist. Daher ist die tatsächliche Wiederholungsgeschwindigkeit langsamer als die angegebene und als die Geschwindigkeit, mit der die Zeichen aus dem Puffer geleert werden.
Dies geschieht in der Absicht, um den Anwender abzusichern, daß er bei einem langsam ablaufenden Programm zu schnell versorgt wird.

Die Anfangsverzögerung und die Wiederholungsgeschwindigkeit beziehen sich auf alle Tasten der Tastatur, die als wiederholbar markiert wurden.

Verwandte Einsprünge:

KM GET DELAY
KM SET REPEAT

22: KM GET DELAY #BB42

Hole die Anfangsverzögerung und die Wiederholungsgeschwindigkeit.

Aktion:

Abfrage auf die Zeitspanne vor der ersten Tastenwiederholung (Anfangsverzögerung) und die Zeitspanne zwischen den Wiederholungen (Wiederholungsgeschwindigkeit).

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

H enthält die Anfangsverzögerung
L enthält die Wiederholungsgeschwindigkeit
AF ist zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Beide Werte werden in Einheiten von 1/50 Sek. angegeben, wobei der Wert 0 jeweils 256 bedeutet.

Verwandte Einsprünge:

KM SET DELAY

23: KM ARM BREAKS #BB45

Zulassen von Unterbrechungen.

Aktion:

Einschalten des Unterbrechungsmechanismus. Der nächste Aufruf von KM BREAK EVENT erzeugt einen Unterbrechungszustand.

Einsprung-Bedingungen:

DE enthält die Adresse der Unterbrechungsroutine
C enthält die ROM-Auswahladresse für diese Routine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Der Unterbrechungsmechanismus kann durch Aufruf von KM DISARM BREAK (oder KM RESET) gesperrt werden.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KM BREAK EVENT
KM DISARM BREAK

24: KM DISARM BREAKS #BB48

Sperren von Unterbrechungen.

Aktion:

Sperren des Unterbrechungsmechanismus. Ab diesem Zeitpunkt werden Unterbrechungen, die durch KM BREAK EVENT erzeugt werden, unterdrückt.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Durch Aufruf von KM ARM BREAK können Unterbrechungen wieder zugelassen werden.

Der Standardzustand des Unterbrechungsmechanismus ist die Sperrung, so daß der Aufruf KM RESET ebenfalls Unterbrechungen sperrt.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KM ARM BREAK
KM BREAK EVENT

25: KM BREAK EVENT #BB4B

Erzeuge ein Abbruch-Ereignis (wenn erlaubt).

Aktion:

Versuch, einen Unterbrechungszustand zu erzeugen.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Wenn der Unterbrechungsmechanismus gesperrt ist, erfolgt keine Aktion. Ansonsten wird eine Unterbrechung erzeugt und im Tastenpuffer erfolgt eine spezielle Markierung. Diese Markierung erzeugt ein Unterbrechungszeichen (#EF), wenn sie vom Puffer gelesen wird. Der Unterbrechungsmechanismus wird nach der Erzeugung eines Unterbrechungszustands automatisch gesperrt, so daß mehrfache Unterbrechungen vermieden werden.

Diese Routine kann während einer Unterbrechung ablaufen und kann und darf deshalb keine weiteren Unterbrechungen zulassen.
Es ist jedoch zu beachten, daß die Anwendung eines LOW JUMP, um diese Routine aufzurufen, Unterbrechungen zuläßt und daher der Verzweigblock von Unterbrechungsroutinen nicht direkt benutzt werden darf.

Verwandte Einsprünge:

KM ARM BREAK
KM DISARM BREAK

26: TXT INITIALISE #BB4E

Initialisiere den Text-VDU.

Aktion:

Vollständige Initialisierung des Text-VDU (wie es während EMS geschieht). Alle Variablen und indirekten Routinen (siehe Kap. 15) des Text-VDU werden initialisiert; der frühere Zustand des VDU ist verloren.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF, BC DE und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die indirekten Verzweigungspunkte des Text-VDU (TXT DRAW CURSOR, TXT UNDRAW CURSOR, TXT WRITE CHAR, TXT UNWRITE und TXT OUT ACTION) werden auf ihre Standardroutinen gesetzt.

Die Controlcodetabelle wird so gesetzt, daß die standardmäßigen Controlcodeaktionen durchgeführt werden.

Die vom Anwender definierte Zeichentabelle wird auf „leer" gesetzt.
Ein-/Ausgabegerät (stream) 0 wird ausgewählt.
Alle Ein-/Ausgabegeräte (streams) werden auf ihren Standardwert gesetzt:
Das Text-Papier (Hintergrund) wird auf Ink 0 gesetzt.
Der Text-Stift (Vordergrund) wird auf Ink 1 gesetzt.
Das Text-Fenster wird auf den gesamten Bildschirm gesetzt.
Der Text-Corsor ist zugelassen, aber abgeschaltet.
Der Modus zur Zeichendarstellung wird auf undurchsichtig gesetzt.
Der VDU wird zugelassen.
Der Modus zur Grapikzeichendarstellung wird abgeschaltet.
Der Cursor wird an die linke obere Ecke des Fensters gestellt.

Der Standardzeichensatz und die standardmäßige Besetzung der Controlcodetabelle sind in Anhang VI und VII beschrieben.

Verwandte Einsprünge:

SCR INITIALISE
TXT RESET

27: TXT RESET #BB51

Rücksetzen des Text-VDU.

Aktion:

Neue Initialisierung der Text-VDU-Indirections und der Controlcodetabelle. Berührt keine anderen Eigenschaften des Text-VDU.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF, DE, BC und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die indirekten Verzweigungspunkte des Text-VDU wie TXT DRAW CURSOR, TXT WRITE CHAR, TXT UNWRITE und TXT OUT ACTION werden auf ihre Standardroutinen gesetzt.

Die Controlcodetabelle wird so gesetzt, daß die standardmäßigen Controlcodeaktionen durchgeführt werden (siehe Anhang VII).

Verwandte Einsprünge:

TXT INITIALISE

28: TXT VDU ENABLE #BB54

Zulassen, daß Zeichen am Bildschirm dargestellt werden.

Aktion:

Es wird zugelassen, daß Zeichen auf Anforderung (TXT OUTPUT oder TXT WR CHAR) ausgegeben werden. Diese Zulassung betrifft das momentan ausgewählte Ein-/Ausgabegerät (auch als stream oder Kanal bezeichnet). Das Corsorzeichen wird ebenfalls zugelassen (durch den Aufruf von TXT CUR ENABLE).

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF zerstört
alle anderen Register unverändert.

Anmerkungen:

Der Controlcodepuffer, der von TXT OUTPUT verwendet wird, wird geleert und jede unvollständige Controlcodefolge ist verloren.

Verwandte Einsprünge:

TXT CUR ENABLE
TXT OUTPUT
TXT VDU DISABLE
TXT WR CHAR

29: TXT VDU DISABLE #BB57

Verbiete, daß Zeichen am Bildschirm dargestellt werden.

Aktion:

Die Ausgabe von Zeichen am Bildschirm (wenn TXT OUTPUT oder TXT WR CHAR aufgerufen werden) wird unterdrückt. Diese betrifft das augenblicklich ausgewählte Ein-/Ausgabegerät (stream). Das Cursorzeichen wird ebenfalls unterdrückt (durch den Aufruf von TXT CUR DISABLE).

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF zerstört
alle anderen Register unverändert.

Anmerkungen:

Der Controlcodepuffer, der von TXT OUTPUT verwendet wird, wird geleert und jede unvollständige Controlcodefolge ist verloren.

Controlcodes werden noch beachtet von TXT OUTPUT

Verwandte Einsprünge:

TXT CUR DISABLE
TXT OUTPUT
TXT VDU ENABLE
TXT WR CHAR

30: TXT OUTPUT #BB5A

Übergib ein Zeichen oder Steuercode an den Text-VDU.

Aktion:

Gibt Zeichen an den Bildschirm aus und beachtet Controlcodes (Zeichen #00...#1F). Arbeitet auf dem augenblicklich ausgewählten Ein-/Ausgabegerät (stream).

Einsprung-Bedingungen:

A enthält das zu sendende Zeichen

Aussprung-Bedingungen:

Alle Register und Flags bleiben erhalten

Anmerkungen:

Diese Routine ruft den indirekten Verzweigungspunkt TXT OUT ACTION, um die Ausgabe des Zeichens durchzuführen oder den Controlcode, wie nachfolgend beschrieben, zu beachten.

Controlcodes können bis zu 9 Parameter umfassen. Diese bestehen aus Zeichen, die einem Anfangscontrolcode folgend gesendet werden. Die abgesandten Zeichen werden in einen Controlcodepuffer abgespeichert, bis alle erforderlichen Parameter besetzt sind. Der Controlcodepuffer ist nur so groß, daß er 9 Parameter aufnehmen kann.

Für alle Ein-/Ausgabegeräte (streams) gibt es nur einen Controlcodepuffer. Es ist deshalb möglich, ein nicht vorhersehbares Ergebnis zu erhalten, wenn während des Sendens einer Controlcodefolge das Ausgabegerät (stream) gewechselt wird.

Wenn der VDU gesperrt ist, werden keine Zeichen am Bildschirm ausgegeben. Controlcodes werden dagegen beachtet; der Anwender sollte jedoch diese Möglichkeit vermeiden, wann immer es möglich ist.

Wenn der Graphikzeichen-Modus zugelassen ist, werden alle Zeichen und Controlcodes über die Graphik-VDU-Routine GRA WR CHAR ausgegeben; es erfolgt keine Controlcodebeachtung.

Zeichen werden in der gleichen Weise ausgegeben wie TXT WR CHAR sie ausgibt.

Verwandte Einsprünge:

GRA WR CHAR
TXT OUT ACTION
TXT SET GRAPHIC
TXT VDU DISABLE
TXT VDU ENABLE
TXT WR CHAR

31: TXT WR CHAR #BB5D

Schreibe ein Zeichen auf den Bildschirm.

Aktion:

Ausgabe eines Zeichens an der Cursorposition des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream).
Controlcodes (Zeichen #00...#1F) werden dargestellt und nicht beachtet.

Einsprung-Bedingungen:

A enthält das auszugebende Zeichen

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört
alle anderen Register bleiben erhalten

Anmerkungen:

Wenn der VDU gesperrt ist, wird kein Zeichen ausgegeben.

Vor der Ausgabe eines Zeichens wird überprüft, ob die Cursorposition innerhalb des Textfensters liegt (siehe TXT VALIDATE). Nach der Zeichenausgabe wird der Cursor um 1 Zeichen nach rechts verschoben. Um das Zeichen auf dem Bildschirm darzustellen, ruft diese Routine den indirekten Verzweigungspunkt TXT WRITE CHAR.

Verwandte Einsprünge:

GRA WR CHAR
TXT OUTPUT
TXT RD CHAR
TXT WRITE CHAR

32: TXT RD CHAR #BB60

Lies ein Zeichen vom Bildschirm.

Aktion:

Lies von der Cursorposition des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream) ein Zeichen.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

Wenn ein Zeichen erkannt wurde:
CARRY-Flag „an“
A enthält das gelesene Zeichen

Wenn kein Zeichen erkannt wurde:
CARRY-Flag „aus“
A ist Null

Immer:
alle anderen Flags zerstört
alle anderen Register unverändert

Anmerkungen:

Vor dem Lesen eines Zeichens wird die Cursorposition nicht auf Gültigkeit (innerhalb des Fensters) geprüft. Um das Lesen von Zeichen außerhalb des Fensters (oder sogar außerhalb des Bildschirms) zu vermeiden, müssen Maßnahmen vorgesehen werden.

Das Lesen geschieht durch Vergleich der Matrix, die auf dem Bildschirm gefunden wurde, mit den Matrizen, die zur Darstellung der Zeichen verwendet wurden. Durch Veränderung einer Zeichenmatrix, der Stift- oder Papierfarbe oder des Bildschirms (z.B. Ziehen einer Linie durch ein Zeichen) können Zeichen nicht mehr lesbar werden.

Um das tatsächliche Lesen eines Zeichens vom Bildschirm durchzuführen, wird der indirekte Verzweigpunkt TXT UNWRITE aufgerufen.
Spezielle Vorsichtsmaßnahmen müssen getroffen werden, damit kein Zwischenraum erzeugt wird. Zunächst wird das Zeichen gelesen in der Annahme, daß das Zeichen mit der aktuellen Stiftfarbe geschrieben wurde und jede andere Farbe wird als Hintergrund angenommen. Wenn so kein erkennbares Zeichen erzeugt werden kann oder ein Zwischenraum erzeugt wurde, wird ein erneuter Versuch durchgeführt unter der Annahme, daß das Zeichen mit der momentanen Papierfarbe geschrieben wurde und jede andere Farbe wird als Vordergrund angenommen.

Die Zeichen werden beginnend mit #00 bis #FF untersucht.

Verwandte Einsprünge:

TXT UNWRITE
TXT WR CHAR

33: TXT SET GRAPHIC #BB63

Schalte die Zeichenausgabemöglichkeit des Graphik-VDU an oder aus.

Aktion:

Die Ausgabe von Graphikzeichen auf dem momentan ausgewählten Ein-/Ausgabegerät (stream) zulassen oder sperren.

Einsprung-Bedingungen:

Wenn die Grahikausgabe angeschaltet werden soll:

A muß ungleich Null sein

Wenn die Graphikausgabe ausgeschaltet werden soll:

A muß Null sein

Aussprung-Bedingungen:

AF ist zerstört
alle anderen Register unverändert

Anmerkungen:

Wenn die Graphikzeichenausgabe angeschaltet ist, werden alle Zeichen, die an TXT OUTPUT gesendet wurden, über den Graphik-VDU (siehe GRA WR CHAR) ausgegeben, anstatt über den Text-VDU (siehe TXT WR CHAR). Ebenso werden sämtliche Controlcodes ausgegeben, anstatt sie auszuführen.

Zeichen, die an TXT WR CHAR gesendet werden, werden normal ausgegeben.

Die Zeichenausgabe wird nicht verhindert durch Sperren des Text-VDU (mit TXT VDU DISABLE), wenn Graphikzeichenausgabe zugelassen ist.

Verwandte Einsprünge:

GRA WR CHAR
TXT OUTPUT

34: TXT WIN ENABLE #BB66

Setze die Größe des momentanen Textfensters.

Aktion:

Angabe der Grenzen für das Fenster auf dem augenblicklich ausgewählten Ein-/Ausgabegerät (stream). Die Ecken bestehen aus der ersten und letzten Zeichenspalte innerhalb des Fensters und der ersten und letzten Zeichenreihe innerhalb des Fensters.

Einsprungs-Bedingungen:

H enthält die physikalische Spalte einer Ecke
D enthält die physikalische Spalte der anderen Ecke
L enthält die physikalische Reihe einer Ecke
E enthält die physikalische Reihe der anderen Ecke

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört
alle anderen Register unverändert

Anmerkungen:

Die Eckenpositionen werden in physikalischen Bildschirm-Koordinaten angegeben, d.h. Reihe 0, Spalte 0 ist die obere linke Ecke des Bildschirms und die Koordinaten sind Zahlen mit Vorzeichen.

Wenn notwendig, wird das Fenster verkleinert, damit es auf den Bildschirm paßt.

Die linke Spalte des Fensters wird aus dem kleineren Wert aus H und D, die oberste Reihe aus dem kleineren Wert aus L und E gebildet.

Der Cursor wird auf die linke obere Ecke des Fensters gesetzt.

Das Fenster wird nicht gelöscht.

Wenn das Fenster den ganzen Bildschirm einnimmt und das Fenster „gerollt" wird, wird dazu die Hardware-Roll-Routine (siehe SCR HW ROLL), ansonsten die Software-Roll-Routine (siehe SCR SW ROLL) verwendet.

Das Standard-Textfenster umfaßt den gesamten Bildschirm und wird gesetzt, wenn TXT INITIALISE oder SET MODE aufgerufen wird.

Verwandte Einsprünge:

TXT GET WINDOW
TXT VALIDATE

35: TXT GET WINDOW #BB69

Hole die Größe des augenblicklichen Fensters.

Aktion:

Erfragen der Grenzen des Fensters für das momentan ausgewählte Ein-/Ausgabegerät (stream) und ob es den gesamten Bildschirm erfaßt.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

Wenn das Fenster den gesamten Bildschirm umfaßt:

CARRY-Flag „aus“

Wenn das Fenster nicht den gesamten Bildschirm umfaßt:

CARRY-Flag „an“

Immer:

H enthält die linke Spalte des Fensters
D enthält die rechte Spalte des Fensters
L enthält die oberste Reihe des Fensters
E enthält die unterste Reihe des Fensters
A ist zerstört
alle anderen Register unverändert.

Anmerkungen:

Die Grenzen des Fensters werden in physikalischen Koordinaten zurückgegeben, d.h. Reihe 0, Spalte 0 ist die linke obere Ecke des Bildschirms.

Die zurückgegebenen Grenzen können sich von denen unterscheiden, die über TXT WIN ENABLE gesetzt werden, da das Fenster verkleinert wird, um auf den Bildschirm zu passen.

Verwandte Einsprünge:

TXT VALIDATE
TXT WIN ENABLE

36: TXT CLEAR WINDOW #BB6C

Lösche das aktuelle Fenster.

Aktion:

Lösche das Textfenster des momentan ausgewählten Ein-/Ausgabegeräts (stream) mit der zugehörigen Papier-Farbe.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört
alle anderen Register unverändert.

Anmerkungen:

Der Cursor wird auf die linke obere Ecke des Fensters gesetzt.

Verwandte Einsprünge:

GRA CLEAR WINDOW
SCR CLEAR
TXT SET PAPER
TXT WIN ENABLE

37: TXT SET COLUMN #BB6F

Setze die horizontale Position des Cursors.

Aktion:

Setze die momentane Position des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream) auf eine neue Spalte. Das Cursorzeichen verschwindet von der momentanen Position und wird auf der neuen Position wieder dargestellt (wenn der Cursor zugelassen und angeschaltet ist).

Einsprung-Bedingungen:

A enthält die geforderte logische Spalte des Cursors.

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die angeforderte Spalte wird in logischen Koordinaten angegeben, d.h. Spalte 1 ist die linke Seite des Fensters.

Der Cursor kann außerhalb des Fensters gesetzt werden. Bevor jedoch irgendein Zeichen über den Text-VDU (siehe TXT VALIDATE) geschrieben oder das Cursorzeichen dargestellt wird, wird die Spalte innerhalb des Fensters verlegt.

Verwandte Einsprünge:

TXT GET CURSOR
TXT SET CURSOR
TXT SET ROW

38: TXT SET ROW #BB72

Setze die vertikale Position des Cursors.

Aktion:

Setze die momentane Position des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream) auf eine neue Reihe. Das Cursorzeichen verschwindet von der momentanen Position und wird auf der neuen Position wieder dargestellt (wenn der Cursor zugelassen und angeschaltet ist).

Einsprung-Bedingungen:

A enthält die geforderte logische Spalte des Cursors.

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die angeforderte Spalte wird in logischen Koordinaten angegeben, d.h. Reihe 1 ist die linke Seite des Fensters.

Der Cursor kann außerhalb des Fensters gesetzt werden. Bevor jedoch irgendein Zeichen über den Text-VDU (siehe TXT VALIDATE) geschrieben oder das Cursorzeichen dargestellt wird, wird die Spalte innerhalb des Fensters verlegt.

Verwandte Einsprünge:

TXT GET CURSOR
TXT SET COLUMN
TXT SET CURSOR

39: TXT SET CURSOR #BB75

Positioniere den Cursor.

Aktion:

Setze die momentane Position des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream) auf eine neue Reihe und Spalte. Das Cursorzeichen verschwindet und wird auf der neuen Position wieder dargestellt (wenn der Cursor zugelassen und angeschaltet ist).

Einsprung-Bedingungen:

H enthält die geforderte logische Spalte
L enthält die geforderte logische Reihe

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die angeforderte Position wird in logischen Koordinaten angegeben, d.h. Reihe 1, Spalte 1 ist die linke obere Ecke des Fensters.

Der Cursor kann außerhalb des Fensters positioniert werden. Bevor jedoch irgendein Zeichen über den Text-VDU (siehe TXT VALIDATE) geschrieben oder das Cursorzeichen dargestellt wird, wird die Cursorposition innerhalb des Fensters verlegt.

Verwandte Einsprünge:

TXT GET CURSOR
TXT SET COLUMN
TXT SET ROW

40: TXT GET CURSOR #BB78

Frage die aktuelle Cursorposition ab.

Aktion:

Abfrage nach der momentanen Position des Cursors und wie oft das Fenster des augenblicklich ausgewählten Ein-/Ausgabegerätes (stream) „gerollt" wurde.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

H enthält die logische Spalte des Cursors
L enthält die logische Reihe des Cursors
A enthält die Anzahl, wie oft das Fenster „gerollt" wurde
Alle Flags sind zerstört
alle anderen Register bleiben erhalten.

Anmerkungen:

Die Cursorposition wird in logischen Koordinaten angegeben, d.h. Reihe 1, Spalte 1 ist die linke obere Ecke des Fensters.

Der ausgegebene Zähler, wie oft das Fenster „gerollt" wurde, hat keine absolute Bedeutung. Er wird vermindert, wenn das Fenster „hochgerollt" wird und erhöht, wenn das Fenster „heruntergerollt" wird. Es kann zu einem Vergleich mit einem früheren Wert herangezogen werden, um zu bestimmen, ob das Fenster „gerollt" wurde.

Die zurückgegebene Position kann außerhalb des Fensters liegen und stellt daher nicht notwendigerweise die Position dar, an die das nächste Zeichen geschrieben wird. Um dies zu prüfen muß TXT VALIDATE verwendet werden.

Verwandte Einsprünge:

TXT SET COLUMN
TXT SET CURSOR
TXT SET ROW
TXT VALIDATE

41: TXT CUR ENABLE #BB7B

Lasse die Cursordarstellung (Anwender) zu.

Aktion:

Zulassen der Darstellung des Cursors für das augenblicklich ausgewählte Ein-/Ausgabegerät (stream) auf dem Bildschirm. Das Cursorzeichen wird sofort dargestellt, außer der Cursor ist ausgeschaltet (siehe TXT CUR OFF)

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF ist zerstört
alle anderen Register sind unverändert.

Anmerkungen:

Das Zulassen und Sperren des Cursors ist für den Anwender gedacht. Es wird ebenso verwendet, wenn der VDU gesperrt wird (siehe TXT VDU ENABLE und TXT VDU DISABLE).

Verwandte Einsprünge:

TXT CUR DISABLE
TXT CUR ON
TXT DRAW CURSOR
TXT UNDRAW CURSOR

42: TXT CUR DISABLE #BB7E

Verhindere die Cursordarstellung (Anwender).

Aktion:

Verhindern der Darstellung des Cursorzeichens für das augenblicklich ausgewählte Ein-/Ausgabegerät (stream) auf dem Bildschirm. Das Cursorzeichen verschwindet sofort vom Bildschirm, falls es vorhanden war.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF ist zerstört
alle anderen Register sind unverändert.

Anmerkungen:

Das Zulassen und Sperren des Cursors ist für den Anwender gedacht. Es wird ebenso verwendet, wenn der VDU gesperrt wird (siehe TXT VDU ENABLE und TXT VDU DISABLE).

Verwandte Einsprünge:

TXT CUR ENABLE
TXT CUR OFF
TXT DRAW CURSOR
TXT UNDRAW CURSOR

43: TXT CUR ON #BB81

Lasse die Cursordarstellung (System) zu.

Aktion:

Zulassen der Darstellung des Cursors für das augenblicklich ausgewählte Ein-/Ausgabegerät (stream) auf dem Bildschirm. Das Cursorzeichen wird sofort dargestellt, außer der Cursor ist gesperrt (siehe TXT CUR DISABLE).

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

alle Register und Flags unverändert.

Anmerkungen:

Das An- und Ausschalten des Cursors ist für den Gebrauch durch die System-ROMS gedacht.

Verwandte Einsprünge:

TXT CUR ENABLE
TXT CUR OFF
TXT DRAW CURSOR
TXT UNDRAW CURSOR

44: TXT CUR OFF #BB84

Verhindere die Cursordarstellung (System).

Aktion:

Verhindern der Darstellung des Cursorzeichens für das augenblicklich ausgewählte Ein-/Ausgabegerät (stream) auf dem Bildschirm. Das Cursorzeichen verschwindet sofort vom Bildschirm, falls es vorhanden war.

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

alle Register und Flags unverändert.

Anmerkungen:

Das An- und Ausschalten des Cursors ist für den Gebrauch durch die System-ROMS gedacht.

Verwandte Einsprünge:

TXT CUR DISABLE
TXT CUR ON
TXT DRAW CURSOR
TXT UNDRAW CURSOR

45: TXT VALIDATE #BB87

Prüfe, ob Cursorposition innerhalb des Fensters.

Aktion:

Überprüfe eine Bildschirmstelle, ob sie innerhalb des augenblicklichen Fensters liegt. Wenn dies nicht der Fall ist, bestimme die Bildschirmstelle innerhalb des Fensters, an der ein Zeichen unter Beachtung der Regeln für die Verlegung ausgegeben würde.

Einsprung-Bedingungen:

H enthält die logische Spalte der zu prüfenden Stelle
L enthält die logische Reihe der zu prüfenden Stelle

Aussprung-Bedingungen:

Wenn die Ausgabe des Zeichens kein „Rollen" des Fensters verursachen würde:
CARRY-Flag „an"
B ist zerstört

Wenn die Ausgabe des Zeichens ein „Hochrollen" des Fensters verursachen würde:
CARRY-Flag „aus"
B enthält #FF

Wenn die Ausgabe des Zeichens ein „Herunterrollen" des Fensters verursachen würde:
CARRY-Flag „aus"
B enthält #00

Immer:
H enthält die logische Spalte, an der das Zeichen ausgegeben würde
L enthält die logische Reihe, an der das Zeichen ausgegeben würde
A und die anderen Flags zerstört
alle anderen Register unverändert.

Anmerkungen:

Die Positionen auf dem Bildschirm werden in logischen Koordinaten angegeben, d.h. Reihe 1, Spalte 1 ist die linke obere Ecke des Fensters.

Bevor ein Zeichen ausgegeben oder das Cursorzeichen auf dem Bildschirm dargestellt wird, überprüft der Text-VDU die momentane Position, führt gegebenenfalls ein erforderliches „Rollen“ durch und schreibt an die passende Stelle.

Das Verfahren, die Ausgabeposition anhand der Prüfposition zu bestimmen, ist folgendes:

1. Wenn die Position rechts von der rechten Fensterseite liegt, wird sie auf die linke Fensterseite der nächsten Zeile verlegt,
2. Wenn die Position links von der linken Fensterseite liegt, wird sie auf die rechte Fensterseite der vorausgehenden Zeile verlegt,
3. Wenn die Position nun über der Oberseite des Fensters liegt, wird sie in die oberste Zeile des Fensters verlegt und ein „Herunterrollen“ des Fensters ist erforderlich.
4. Wenn die Position nun unter der Unterseite des Fensters liegt, wird sie in die unterste Zeile des Fensters verlegt und ein „Hochrollen“ des Fensters ist erforderlich.

Verwandte Einsprünge:

SCR HW ROLL
SCR SW ROLL
TXT GET CURSOR

46: TXT PLACE CURSOR #BB8A

Stelle das Cursorzeichen auf dem Bildschirm dar.

Aktion:

Ausgabe des Cursorzeichens auf dem Bildschirm an der Cursorposition für das augenblicklich ausgewählte Ein-/Ausgabegerät (stream).

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF zerstört
alle anderen Register unverändert.

Anmerkungen:

TXT PLACE CURSOR ermöglicht es dem Anwender, innerhalb eines Fensters mit mehreren Cursorn zu arbeiten. Um lediglich das normale Cursorzeichen auf dem Bildschirm darzustellen, sollte der indirekte Verzweigpunkt TXT DRAW CURSOR aufgerufen werden. Höherstufige Routinen, wie TXT OUTPUT oder TXT SET CURSOR löschen und setzen den Cursor automatisch richtig, alle anderen Cursor muß der Anwender selbst verwalten.

Es ist nicht gut, TXT PLACE CURSOR für eine bestimmte Bildschirmstelle zweimal aufzurufen, ohne zwischenzeitlich TXT REMOVE CURSOR aufgerufen zu haben, da ein Cursor fälschlicherweise auf dem Bildschirm stehenbleiben könnte, wenn die Cursorposition verändert wird.

Die Cursorposition wird vor der Darstellung des Cursorzeichens innerhalb des Fensters verlegt.

Das Cursorzeichen ist die inverse (gegensätzliche) Darstellung eines Rechtecks, gebildet durch ein exclusives-ODER des Bildschirminhalts an der Cursorposition mit dem exclusiven-ODER der aktuellen Stift- und Papier-Ink.

Verwandte Einsprünge:

TXT DRAW CURSOR
TXT REMOVE CURSOR

47: TXT REMOVE CURSOR #BB8D

Lösche ein Cursorsymbol vom Bildschirm.

Aktion:

Entfernen eines Cursorzeichens vom Bildschirm an der Cursorposition des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream).

Einsprung-Bedingungen:

keine

Aussprung-Bedingungen:

AF zerstört
alle anderen Register unverändert.

Anmerkungen:

TXT REMOVE CURSOR ermöglicht es dem Anwender, innerhalb eines Fensters mit mehreren Cursorn zu arbeiten. Um lediglich das normale Cursorzeichen auf dem Bildschirm zu löschen, sollte der indirekte Verzweigpunkt TXT UNDRAW CURSOR aufgerufen werden. Höherstufige Routinen, wie TXT OUTPUT oder TXT SET CURSOR löschen und setzen den Cursor automatisch richtig, alle anderen Cursor muß der Anwender selbst verwalten.

TXT REMOVE CURSOR sollte nur benutzt werden, um einen Cursor zu entfernen, der über den Aufruf TXT PLACE CURSOR gesetzt wurde.

Der Cursor sollte gelöscht werden, wenn sich die Cursorposition verändert (das „Rollen" des Fensters verändert die Cursorposition implizit) oder der Bildschirm gelesen oder geschrieben wird. Die falsche Anwendung dieser Routine kann ein fälschlich gesetztes Cursorzeichen zur Folge haben.

Die Cursorposition wird vor dem Löschen des Cursorzeichens innerhalb des Fensters verlegt (dies sollte nicht vorkommen, da TXT PLACE CURSOR dies bereits vorgenommen hat).

Das Cursorzeichen ist die inverse (gegensätzliche) Darstellung eines Rechtecks, gebildet durch ein exclusives-ODER des Bildschirminhalts an der Cursoposition mit dem exclusiven-ODER der aktuellen Stift- und Papier-Ink.

Verwandte Einsprünge:

TXT PLACE CURSOR
TXT UNDRAW CURSOR

48: TXT SET PEN #BB90

Setze die Ink für das Schreiben von Zeichen.

Aktion:

Angabe der Ink für den Textstift des augenblicklich ausgewählten Ein-/Ausgabegerätes (stream). Diese Ink wird zur Darstellung der Zeichen verwendet (Vordergrund).

Einsprung-Bedingungen:

A enthält die zu benutzende Ink.

Aussprung-Bedingungen:

AF und HL sind zerstört
alle anderen Register unverändert.

Anmerkungen:

Die Ink wird mit einer Maske verknüpft, um, abhängig vom augenblicklichen Bildschirmmodus, eine gültige Ink zu erhalten. Die Maske ist #0F in Modus 0, #03 in Modus 1 und #01 in Modus 2.

Das Cursorzeichen wird, wenn zugelassen, unter Berücksichtigung der neuen Ink neu erstellt.

Verwandte Einsprünge:

GRA SET PEN
SCR SET INK
TXT GET PEN
TXT SET PAPER

49: TXT GET PEN #BB93

Hole die Ink für das Schreiben von Zeichen.

Aktion:

Abfrage nach der Ink für den Stift des augenblicklich ausgewählten Ein-/Ausgabegerätes (stream). Diese Ink wird zur Darstellung der Zeichen verwendet (Vordergrundink).

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A enthält die Ink
Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine hat keine weiteren Auswirkungen.

Verwandte Einsprünge:

GRA GET PEN
SCR GET INK
TXT GET PAPER
TXT SET PEN

50: TXT SET PAPER #BB96

Setze die Ink für das Schreiben des Texthintergrundes.

Aktion:

Angabe der Paper-Ink für den Text des augenblicklich ausgewählten Ein-/Ausgabegerätes (stream). Diese Ink wird zur Darstellung des Hintergrundes von Zeichen und zum Löschen des Textfensters verwendet.

Einsprung-Bedingungen:

A enthält die zu benutzende Ink.

Aussprung-Bedingungen:

AF und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Ink wird mit einer Maske verknüpft, um, abhängig vom augenblicklichen Bildschirmmodus, eine gültige Ink zu erhalten. Die Maske ist #0F in Modus 0, #03 in Modus 1 und #01 in Modus 2.

Das Cursorzeichen wird, wenn zugelassen, unter Berücksichtigung der neuen Ink neu erstellt.

Diese Ink wird zum Löschen von Bereichen des Textfensters (durch TXT CLEAR WINDOW und bestimmte Controlcodes) verwendet.

Diese Routine löscht nicht das Textfenster.

Verwandte Einsprünge:

GRA SET PAPER
SCR SET INK
TXT GET PAPER
TXT SET PEN

51: TXT GET PAPER #BB99

Hole die Ink für das Schreiben des Texthintergrundes.

Aktion:

Abfrage nach der Paper-Ink für das augenblicklich ausgewählte Ein-/Ausgabegerät (stream). Diese Ink wird als Zeichenhintergrund und zum Löschen des Textfensters verwendet.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A enthält die Ink
Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine hat keine weiteren Auswirkungen.

Verwandte Einsprünge:

GRA GET PAPER
SCR GET INK
TXT GET PEN
TXT SET PAPER

52: TXT INVERSE #BB9C

Tauschen der Inks für Stift und Papier.

Aktion:

Tausche die Inks für Text-Stift und -Papier (Vordergrund und Hintergrund) des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream).

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das Cursorzeichen wird nicht neu erstellt und sollte deshalb bei Aufruf der Routine nicht auf dem Bildschirm sein.

Verwandte Einsprünge:

TXT SET PAPER
TXT SET PEN

53: TXT SET BACK #BB9F

Zulassen oder Sperren der Hintergrunddarstellung.

Aktion:

Setzen der Zeichendarstellung für das augenblicklich ausgewählte Ein-/Ausgabegerät (stream) auf undurchsichtig oder transparent.
Im Modus „undurchsichtig" wird das Zeichen mit Hintergrund dargestellt. Im Modus „transparent" wird das Zeichen über den aktuellen Bildschirminhalt gelegt.

Einsprung-Bedingungen:

Wenn der Hintergrund mit dargestellt werden soll (undurchsichtig):

A muß Null sein.

Wenn der Hintergrund nicht dargestellt werden soll (transparent):

A muß ungleich Null sein.

Aussprung-Bedingungen:

AF und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die transparente Darstellungsart wird verwendet, um Diagramme mit Bezeichnungen zu versehen oder bei ähnlich gelagerten Anwendungen. Die generelle Verwendung kann unglückliche Effekte haben, da das Überschreiben das darunterliegende Zeichen nicht löscht, und so ein nicht mehr zu entwirrendes Durcheinander auf dem Bildschirm entsteht.
Das Setzen des Transparent-Modus berührt nicht den Graphik-VDU, da TXT WR CHAR grundsätzlich im Modus „undurchsichtig" darstellt.

Verwandte Einsprünge:

TXT GET BACK
TXT WR CHAR
TXT WRITE CHAR

54: TXT GET BACK #BBA2

Abfrage, ob der Hintergrund dargestellt wird.

Aktion:

Abfrage nach dem Zeichendarstellungsmodus für das augenblicklich ausgewählte Ein-/Ausgabegerät.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn der Hintergrund mit dargestellt wird (undurchsichtig):

A enthält Null

Wenn der Hintergrund nicht dargestellt wird (transparent):

A ungleich Null

Immer:
DE, HL und Flags sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Dies bezieht sich nur auf den Text-VDU, der Graphik-VDU stellt immer undurchsichtig dar.

Verwandte Einsprünge:

TXT SET BACK

55: TXT GET MATRIX #BBA5

Hole die Adresse einer Zeichenmatrix.

Aktion:

Berechnen einer Zeigeradresse auf die Matrix für ein Zeichen und bestimmen, ob es sich um eine vom Anwender definierte Matrix handelt.

Einsprung-Bedingungen:

A enthält das Zeichen, dessen Matrix gefunden werden soll.

Aussprung-Bedingungen:

Wenn die Matrix in der vom Anwender definierten Tabelle steht:

CARRY-Flag „an“

Wenn die Matrix im unteren ROM steht:

CARRY-Flag „aus“

Immer:

HL enthält die Adresse der Matrix
A und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Matrix kann im RAM oder im ROM liegen. Der Text-VDU nimmt an, daß die zugehörigen ROM's zugelassen oder gesperrt sind, wenn die Routine aufgerufen wird (das untere ROM ist „an“, das obere ROM normalerweise „aus“).

Die Matrix ist als ein 8 Byte langer, bit-signifikanter Vektor abgespeichert. Das erste Byte definiert die oberste Zeile des Zeichens, das letzte Byte die unterste Zeile. Bit 7 eines Bytes verweist auf das linke Pixel einer Zeile und Bit 0 auf das rechte. Wenn ein Bit in der Matrix gesetzt ist, soll das Pixel in der Stiftfarbe dargestellt werden. Wenn ein Bit nicht gesetzt ist, soll das Pixel entweder in der Papierfarbe dargestellt oder freigehalten werden (abhängig vom Darstellungsmodus „undurchsichtig/transparent“).

Verwandte Einsprünge:

TXT SET MATRIX

56: TXT SET MATRIX #BBA8

Setze eine Zeichenmatrix.

Aktion:

Angabe einer Matrix für ein vom Anwender definiertes Zeichen. Wenn es sich nicht um ein vom Anwender definiertes Zeichen handelt, wird keine Aktion durchgeführt.

Einsprung-Bedingungen:

A enthält das Zeichen, dessen Matrix angegeben wird
HL enthält die Adresse der Matrixdefinition.

Aussprung-Bedingungen:

Wenn das Zeichen vom Anwender definierbar ist:
CARRY-Flag „an"

Wenn das Zeichen nicht vom Anwender definierbar ist:
CARRY-Flag „aus"

Immer:
A, BC, DE, HL und andere Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Matrix ist als ein 8 Byte langer, bit-signifikanter Vektor abgespeichert. Das erste Byte definiert die oberste Zeile des Zeichens, das letzte Byte die unterste Zeile. Bit 7 eines Bytes verweist auf das linke Pixel einer Zeile und Bit 0 auf das rechte. Wenn ein Bit in der Matrix gesetzt ist, soll das Pixel in der Stiftfarbe dargestellt werden. Wenn ein Bit nicht gesetzt ist, soll das Pixel entweder in der Papierfarbe dargestellt oder freigehalten werden (abhängig vom Darstellungsmodus „undurchsichtig/transparent").

Die Matrix wird von dem angegebenen Bereich in die Zeichenmatrixtabelle kopiert ohne die RAM LAMS zu benutzen, so daß die Matrizen vom ROM gesetzt werden können, vorausgesetzt, daß es zugelassen ist. (Es ist zu beachten, daß der Verzweigblock das obere ROM sperrt).

Das Abändern einer Zeichenmatrix verändert die Matrix für alle Ein-/Ausgabegeräte (streams). Auf dem Bildschirm wird kein Zeichen verändert; es wird nur das Zeichen verändert, wenn es das nächste Mal auf dem Bildschirm dargestellt wird.

Verwandte Einsprünge:

TXT GET MATRIX
TXT SET M TABLE

57: TXT SET M TABLE #BBAB

Setze die Adresse für eine benutzerdefinierte Matrixtabelle.

Aktion:

Angabe der benutzerdefinierten Matrixtabelle und der Zahl der Zeichen in dieser Tabelle. Die Tabelle wird mit den augenblicklichen Matrixdefinitionen initialisiert.

Einsprung-Bedingungen:

DE enthält das erste Zeichen der Tabelle
HL enthält die Startadresse der neuen Tabelle.

Aussprung-Bedingungen:

Wenn vorher noch keine benutzerdefinierte Matrixtabelle vorhanden war:

CARRY-Flag „aus"
A und HL zerstört.

Wenn vorher eine benutzerdefinierte Matrixtabelle vorhanden war:

CARRY-Flag „an"
A enthält das erste Zeichen der alten Tabelle
HL enthält die Adresse der alten Tabelle.

Immer:

BC, DE und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Wenn das erste angegebene Zeichen im Bereich von 0 bis 255 liegt, dann werden die Matrizen für alle Zeichen zwischen dem angegebenen und dem Zeichen 255 in der benutzerdefinierten Tabelle abgespeichert.

Wenn das erste angegebene Zeichen nicht im Bereich von 0 bis 255 liegt, wird angenommen, daß die benutzerdefinierte Matrixtabelle keine Matrizen enthält (und die Tabellenadresse wird ignoriert).

Die Tabelle muß (256 – erstes Zeichen) ★ 8 Bytes lang sein. Die Matrizen werden in der Tabelle in aufsteigender Folge abgespeichert. Die Tabelle wird mit den augenblicklichen Matrixdefinitionen initialisiert, unabhängig davon, ob sie vorher im RAM oder im ROM waren.

Die Tabelle darf nicht im RAM unterhalb eines ROM angelegt werden.

Es ist zulässig, daß sich alte und neue Tabellen überlappen (mit der Möglichkeit der Erweiterung oder Verkürzung), um vorzusehen, daß die Matrizen in der neuen Tabelle eine frühere oder die gleiche Adresse belegen, die sie in der alten Tabelle hatten.

Alle Ein-/Ausgabegeräte (streams) teilen sich die Matrixtabelle, so daß sich Änderungen auf alle Ein-/Ausgabegeräte (streams) auswirken.

Verwandte Einsprünge:

TXT GET M TABLE
TXT SET MATRIX

58: TXT GET M TABLE #BBAE

Hole die Adresse einer benutzerdefinierten Matrixtabelle.

Aktion:

Hole die Adresse der augenblicklich gültigen benutzerdefinierten Matrixtabelle und das erste Zeichen dieser Tabelle.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn keine benutzerdefinierte Tabelle vorhanden ist:

CARRY-Flag „aus“
A und HL zerstört.

Wenn eine benutzerdefinierte Tabelle vorhanden ist:

CARRY-FLAG „an“
A enthält das erste Zeichen der Tabelle
HL enthält die Startadresse der Tabelle

Immer:

alle anderen Flags zerstört
alle anderen Register unverändert.

Anmerkungen:

Die Matrizen für die Zeichen zwischen dem ersten Zeichen und dem Zeichen 255 sind in der Tabelle in aufsteigender Reihenfolge abgespeichert. Jede Matrix ist 8 Bytes lang.

Verwandte Einsprünge:

TXT GET MATRIX
TXT SET M TABLE

59: TXT GET CONTROLS #BBB1

Hole die Adresse der Controlcodetabelle.

Aktion:

Holen der Adresse der Controlcodetabelle.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

HL enthält die Adresse der Controlcodetabelle
alle anderen Register und die Flags unverändert.

Anmerkungen:

Alle Ein-/Ausgabegeräte (streams) teilen sich eine Controlcodetabelle, so daß sich jede Änderung auf alle Ein-/Ausgabegeräte (streams) auswirkt.

Die Controlcodetabelle hat für jeden Controlcode einen Eintrag von 3 Bytes. Die Einträge sind in aufsteigender Folge abgespeichert, so daß der erste Eintrag für #00 und der letzte für #1F gilt.

Das erste Byte jedes Eintrags beinhaltet die Anzahl der Parameter, die der Controlcode benötigt, die beiden anderen Bytes beinhalten die Adresse der Routine, die aufgerufen werden muß, um den Controlcode zu verarbeiten, wenn alle Parameter bereitgestellt wurden. Die Routine muß in den zentralen 32 K des RAM liegen. Sie muß folgende Schnittstellen beachten:

Einsprung:
- A enthält das letzte Zeichen, das gepuffert wurde
- B enthält die Länge des Puffers (einschl. des Controlcodes)
- C hat denselben Inhalt wie A
- HL enthält die Adresse des Controlpuffers (zeigt auf den Controlcode)

Aussprung:
AF, BC, DE, HL sind zerstört,
alle anderen Register sind unverändert.

Da der Controlcodepuffer lediglich 9 Parameterzeichen aufnehmen kann, sollte die Zahl der erforderlichen Parameter auf 9 oder weniger begrenzt werden.
Wenn TXT RESET aufgerufen wird, wird die Controlcodetabelle mit ihren Standardroutinen neu initialisiert.

Verwandte Einsprünge:

TXT OUTPUT

60: TXT STR SELECT #BBB4

Selektiere ein Text-VDU-Ein-/Ausgabegerät (stream).

Aktion:

Mache ein angegebenes Ein-/Ausgabegerät (stream) zum aktuell ausgewählten Ein-/Ausgabegerät (stream). (Wenn es nicht schon ausgewählt ist).

Einsprung-Bedingungen:

A enthält das angeforderte Ein-/Ausgabegerät (stream).

Aussprung-Bedingungen:

A enthält das vorherige ausgewählte Ein-/Ausgabegerät (stream)
HL und alle Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die angeforderte Nummer des Ein-/Ausgabegerätes (stream) wird mit einer Maske (#07) verknüpft, um eine legale Gerätenummer (Stream-Nummer) zu erhalten.

Viele Attribute des Text-VDU können unterschiedlich von den verschiedenen Ein-/Ausgabegeräten (streams) gesetzt werden. Es ist wichtig sich zu vergewissern, daß das richtige Ein-/Ausgabegerät (stream) ausgewählt wurde, wenn eines dieser Attribute verändert werden soll.

Folgende Attribute sind möglich:

- Stiftink (pen ink)
- Papierink (paper ink)
- Cursorposition
- Fensterbegrenzungen
- Zulassen / Sperren des Cursors
- An-/Abschalten des Cursors
- Zulassen / Sperren des VDU
- Zeichendarstellungsmodus
- Graphischer Zeichendarstellungsmodus

Wenn ein Ein-/Ausgabegerät (stream) bereits ausgewählt ist, wird die Routine schnell beendet. Es ist nicht vernünftig, ein Ein-/Ausgabegerät (stream) wiederholt auszuwählen (z.B. bis alle Zeichen gesendet wurden).

Verwandte Einsprünge:

TXT OUTPUT

61: TXT SWAP STREAMS #BBB7

Vertausche die Zustände zweier Ein-/Ausgabegeräte (streams).

Aktion:

Die bestimmenden Eigenschaften zweier Ein-/Ausgabegeräte (streams) werden ausgetauscht. Die Nummer des augenblicklich ausgewählten Ein-/Ausgabegeräts (stream) bleibt dieselbe (obwohl sich die Beschreibung verändert haben könnte).

Einsprung-Bedingungen:

B enthält die Nummer eines Ein-/Ausgabegerätes (streams)
C enthält die Nummer eines anderen Ein-/Ausgabegerätes (streams).

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die angeforderte Nummer des Ein-/Ausgabegerätes (stream) wird mit einer Maske (#07) verknüpft, um eine gültige Gerätenummer (Stream-Nummer) zu erhalten.

Die Attribute, die ausgetauscht werden, sind folgende:

Stiftink (pen ink)
Papierink (paper ink)
Cursorposition
Fensterbegrenzungen
Zulassen / Sperren des Cursors
An-/Abschalten des Cursors
Zulassen / Sperren des VDU
Zeichendarstellungsmodus
Graphischer Zeichendarstellungsmodus

Verwandte Einsprünge:

TXT STR SELECT

62: GRA INITIALISE #BBBA

Initialisiere den Graphik-VDU.

Aktion:

Der Graphik-VDU wird vollständig initialisiert (wie während EMS). Alle variablen und indirekten Verzweigungspunkte werden auf ihre Standardwerte gesetzt.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Folgende Maßnahmen werden vollständig durchgeführt:

Setzen der indirekten Verzweigungspunkte des Graphik-VDU (GRA PLOT, GRA TEST und GRA LINE) auf ihre Standardroutinen

Setzen des Graphikpapiers auf Ink 0

Setzen des Graphikstiftes auf Ink 1

Setzen des Koordinatenursprungs (Anwender) auf die linke untere Ecke des Bildschirms

Die aktuelle Position wird auf den Koordinatenursprung gesetzt

Setzen des Graphikfensters auf den gesamten Bildschirm

Das Graphikfenster wird nicht gelöscht.

Verwandte Einsprünge:

GRA RESET
SCR INITIALISE

63: GRA RESET #BBBD

Setze den Graphik-VDU zurück.

Aktion:

Neues Initialisieren der indirekten Verzweigungspunkte des Graphik-VDU auf ihre Standardroutinen

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Es werden die indirekten Verzweigungspunkte (GRA PLOT, GRA TEST und GRA LINE) auf ihre Standardroutinen gesetzt; ansonsten wird keine andere Aktion durchgeführt.

Verwandte Einsprünge:

GRA INITIALISE

64: GRA MOVE ABSOLUTE #BBC0

Bewege zu einer absoluten Position.

Aktion:

Verlege die augenblickliche Position auf eine absolute Position.

Einsprung-Bedingungen:

DE enthält die gewünschte Anwender-X-Koordinate
HL enthält die gewünschte Anwender-Y-Koordinate.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register bleiben unverändert.

Anmerkungen:

Die neue Position wird in Anwender-Koordinaten angegeben, d.h. relativ zum Koordinatensprung des Anwenders.

Die Routinen des Graphik-VDU für das Plotten, Testen und Zeichnen setzen alle die augenblickliche Graphikposition automatisch auf den angeforderten Punkt (oder Endpunkt).

Verwandte Einsprünge:

GRA ASK CURSOR
GRA MOVE RELATIVE

65: GRA MOVE RELATIVE #BBC3

Bewege relativ zur aktuellen Position.

Aktion:

Verlege die neue Position relativ zu ihrer augenblicklichen Position.

Einsprung-Bedingungen:

DE enthält einen Abstand mit Vorzeichen auf der X-Achse
HL enthält einen Abstand mit Vorzeichen auf der Y-Achse.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die neue Position kann außerhalb des Graphikfensters liegen.

Die Routinen des Graphik-VDU für das Plotten, Testen und Zeichnen setzen alle die augenblickliche Graphikposition automatisch auf den angeforderten Punkt (oder Endpunkt).

Verwandte Einsprünge:

GRA ASK CURSOR
GRA MOVE ABSOLUTE

66: GRA ASK CURSOR #BBC6

Hole die aktuelle Position.

Aktion:

Abfrage nach der momentanen Graphik-Position.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

DE enthält die Anwender-X-Koordinate
HL enthält die Anwender-Y-Koordinate
AF zerstört
alle anderen Register sind unverändert.

Anmerkungen:

Die augenblickliche Position wird in Anwenderkoordinaten angegeben, d.h. relativ zum Anwender-Ursprung.

Die Routinen des Graphik-VDU für das Plotten, Testen und Zeichnen setzen alle die augenblickliche Graphikposition automatisch auf den angeforderten Punkt (oder Endpunkt).

Die zurückgegebene Position ist daher wahrscheinlich der letzte Punkt, der geplottet oder getestet wurde.

Verwandte Einsprünge:

GRA MOVE ABSOLUTE
GRA MOVE RELATIVE

67: GRA SET ORIGIN #BBC9

Setze den Ursprung der Anwenderkoordinaten.

Aktion:

Angabe der Lage des Anwender-Ursprungs und Verlegung der augenblicklichen Position dorthin.

Einsprung-Bedingungen:

DE enthält die Standard-X-Koordinate des Ursprungs
HL enthält die Standard-Y-Koordinate des Ursprungs.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Position des Ursprungs wird in Standardkoordinaten angegeben, bei denen 0,0 die linke untere Ecke des Bildschirms bedeutet.

Die Standardposition des Ursprungs ist 0,0.

Jedes Mal, wenn der Bildschirmmodus durch Aufruf von SCR SET MODE verändert wird, wird der Ursprung auf seine Standardposition gesetzt.

Verwandte Einsprünge:

GRA GET ORIGIN

68: GRA GET ORIGIN #BBCC

Hole den Ursprung der Anwender-Koordinaten.

Aktion:

Frage nach der Lage des Ursprungs der Anwender-Koordinaten.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

DE enthält die Standard-X-Koordinate des Ursprungs
HL enthält die Standard-Y-Koordinate des Ursprungs.
alle anderen Register sind unverändert.

Anmerkungen:

Die Position des Ursprungs wird in Standardkoordinaten angegeben, bei denen 0,0 die linke untere Ecke des Bildschirms bedeutet.

Verwandte Einsprünge:

GRA SET ORIGIN

69: GRA WIN WIDTH #BBCF

Setze rechten und linken Rand des Graphikfensters.

Aktion:

Angabe der horizontalen Position des Graphikfensters.
Der linke und rechte Rand sind gleichzeitig die Punkte, die horizontal innerhalb des Fensters liegen.

Einsprung-Bedingungen:

DE enthält die Standard-X-Koordinate des einen Randes
HL enthält die Standard-X-Koordinate des anderen Randes.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Fensterecken werden in Standard-Koordinaten angegeben, wobei 0,0 die linke untere Ecke des Bildschirms bedeutet und die Koordinaten als 16-Bit-Zahlen mit Vorzeichen geschrieben werden. Der kleinere Wert für die beiden Seiten wird für die linke Seite genommen. Das Fenster wird, wenn nötig, verkleinert, um auf den Bildschirm zu passen.

Die Ränder werden auf Bildschirm-Bytegrenzen ausgerichtet, so daß das Fenster immer ganze Bytes enthält (die linke Seite wird nach links, die rechte nach rechts verlegt). Die Verlegung der Koordinaten der Seiten geschieht in den verschiedenen Modi wie folgt:

Modus	linker Rand	rechter Rand
0	Vielfaches von 2	Vielfaches von 2 minus 1
1	Vielfaches von 4	Vielfaches von 4 minus 1
2	Vielfaches von 8	Vielfaches von 8 minus 1

Das Standardfenster belegt den gesamten Bildschirm. Jedesmal, wenn der Bildschirmmodus verändert wird, wird das Fenster auf die Standardgröße gesetzt.

Alle Graphik-VDU Routinen zum Plotten und Zeichnen prüfen, ob die auszugebenden Punkte innerhalb des Fensters liegen; wenn nicht, werden die Punkte nicht ausgegeben.

Verwandte Einsprünge:

GRA GET W WIDTH
GRA WIN HEIGHT

70: GRA WIN HEIGHT #BBD2

Setze oberen und unteren Rand des Graphikfensters.

Aktion:

Angabe der vertikalen Position des Graphikfensters.
Der obere und untere Rand sind gleichzeitig die Punkte, die vertikal innerhalb des Fensters liegen.

Einsprung-Bedingungen:

DE enthält die Standard-Y-Koordinate für einen Rand
HL enthält die Standard-Y-Koordinate für den anderen Rand.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Fensterecken werden in Standard-Koordinaten angegeben, wobei 0,0 die linke untere Ecke des Bildschirms bedeutet und die Koordinaten als 16-Bit-Zahlen mit Vorzeichen geschrieben werden. Der größere Wert der beiden Angaben wird für den oberen Rand angenommen. Das Fenster wird, wenn nötig, verkleinert, um auf den Bildschirm zu passen. Die Ränder werden auf Bildschirm-Bytegrenzen ausgerichtet, so daß das Fenster immer ganze Bytes enthält (die obere Seite wird nach oben, die untere Seite nach unten verlegt). Die Verlegung der Koordinaten der Ränder geschieht in den verschiedenen Modi wie folgt:

Modus	oberer Rand	unterer Rand
0	Vielfaches von 2	Vielfaches von 2 minus 1
1	Vielfaches von 4	Vielfaches von 4 minus 1
2	Vielfaches von 8	Vielfaches von 8 minus 1

Das Standardfenster belegt den gesamten Bildschirm. Jedesmal, wenn der Bildschirmmodus verändert wird, wird das Fenster auf die Standardgröße gesetzt.

Alle Graphik-VDU Routinen zum Plotten und Zeichnen prüfen, ob die auszugebenden Punkte innerhalb des Fensters liegen; wenn nicht, werden die Punkte nicht ausgegeben.

Verwandte Einsprünge:

GRA GET W HEIGHT
GRA WIN WIDTH

71: GRA GET W WIDTH #BBD5

Hole rechten und linken Rand des Graphikfensters.

Aktion:

Abfrage nach der horizontalen Position des Graphikfensters. Der linke und rechte Rand sind gleichzeitig die ersten und letzten Punkte, die horizontal innerhalb des Fensters liegen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

DE enthält die Standard-X-Koordinate des linken Randes des Fensters
HL enthält die Standard-X-Koordinate des rechten Randes des Fensters
AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Fensterecken werden in Standard-Koordinaten angegeben, wobei 0,0 die linke untere Ecke des Bildschirms bedeutet.

Die Werte müssen nicht genau denen entsprechen, die bei GRA WIN WIDTH angegeben waren, da das Fenster verkleinert wird, um auf den Bildschirm zu passen und die Ränder auf Bildschirm-Bytegrenzen ausgerichtet werden, so daß das Fenster ganze Bytes enthält.

Verwandte Einsprünge:

GRA GET W HEIGHT
GRA WIN WIDTH

72: GRA GET W HEIGHT #BBD8

Hole oberen und unteren Rand des Graphikfensters.

Aktion:

Abfrage nach der vertikalen Position des Graphikfensters. Oberer und unterer Rand sind gleichzeitig die ersten und letzten Punkte, die vertikal innerhalb des Fensters liegen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

DE enthält die Standard-Y-Koordinate des oberen Randes des Fensters
HL enthält die Standard-Y-Koordinate des unteren Randes des Fensters
AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Fensterecken werden in Standard-Koordinaten angegeben, wobei 0,0 die linke untere Ecke des Bildschirms bedeutet.

Die Werte müssen nicht genau denen entsprechen, die bei GRA WIN WIDTH angegeben waren, da das Fenster u.U. verkleinert wird, um auf den Bildschirm zu passen und die Ränder auf Bildschirm-Bytegrenzen ausgerichtet werden, so daß das Fenster ganze Bytes enthält.

Verwandte Einsprünge:

GRA GET W WIDTH
GRA WIN HEIGHT

73: GRA CLEAR WINDOW #BBDB

Lösche das Graphikfenster.

Aktion:

Lösche das Graphikfenster mit der Ink des Graphikpapiers.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert

Anmerkungen:

Die augenblickliche Graphikposition wird auf den Ursprung der Anwenderkoordinaten gestellt.

Verwandte Einsprünge:

GRA SET PAPER
GRA WIN HEIGHT
GRA WIN WIDTH
SCR CLEAR
TXT CLEAR WINDOW

74: GRA SET PEN #BBDE

Setze die Ink für graphisches Plotten.

Aktion:

Setze die Ink für den Graphik-Stift. Diese Ink wird vom Graphik-VDU zum Plotten von Punkten, Zeichnen von Linien und Schreiben von Zeichen verwendet.

Einsprung-Bedingungen:

A enthält die gewünschte Ink.

Aussprung-Bedingungen:

AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Ink wird mit einer Maske verknüpft, um sie in den Ink-Bereich für den aktuellen Bildschirm-Modus zu bringen. Im Modus 0 ist die Maske #OF, im Modus 1 ist sie #O3 und im Modus 2 ist sie #O1.

Verwandte Einsprünge:

GRA GET PEN
GRA SET PAPER
SCR SET INK
TXT SET PEN

75: GRA GET PEN #BBE1

Hole die Ink, in der momentan graphisch geplottet wird.

Aktion:

Abfrage, welche Ink dem Graphikstift gerade zugeordnet ist. Diese Ink wird vom Graphik-VDU zum Plotten von Punkten, Zeichnen von Linien und Darstellen von Zeichen verwendet.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A enthält die Ink
Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Routine hat keine weiteren Auswirkungen.

Verwandte Einsprünge:

GRA GET PAPER
GRA SET PEN
SCR GET INK
TXT GET PEN

76: GRA SET PAPER #BBE4

Setze die Ink für den graphischen Hintergrund.

Aktion:

Angabe der graphischen Papier-Ink.

Einsprung-Bedingungen:

A enthält die gewünschte Farbe.

Aussprung-Bedingungen:

AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Ink wird mit einer Maske verknüpft, um sie in den gültigen Ink-Bereich für den aktuellen Bildschirm-Modus zu bringen. Im Modus 0 ist die Maske #0F, im Modus 1 ist sie #O3 und im Modus 2 ist sie #O1.

Die Papier-Ink wird auch zum Löschen des Graphikfensters und als Hintergrund bei der Zeichendarstellung verwendet. Beim Testen von Punkten wird angenommen, daß sie alle Bereiche außerhalb des Graphikfensters abdeckt.

Verwandte Einsprünge:

GRA GET PAPER
GRA SET PEN
SCR GET INK
TXT SET PAPER

77: GRA GET PAPER #BBE7

Hole die Ink des aktuellen graphischen Hintergrunds.

Aktion:

Abfrage, welche Ink dem Graphikpapier zugeordnet ist.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A enthält die Ink
Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Papier-Ink wird auch zum Löschen des Graphikfensters und als Hintergrund bei der Zeichendarstellung verwendet. Beim Testen von Punkten wird angenommen, daß sie alle Bereiche außerhalb des Graphikfensters abdeckt.

Verwandte Einsprünge:

GRA GET PEN
GRA SET PAPER
SCR GET INK
TXT GET PAPER

78: GRA PLOT ABSOLUTE #BBEA

Plotte einen Punkt an einer absoluten Position.

Aktion:

Die augenblickliche Graphikposition wird an die gelieferte Position verlegt. Wenn diese innerhalb des Graphikfensters liegt, wird der Punkt in der momentanen Ink des Graphikstifts unter Beachtung des gerade gültigen Graphikdarstellungsmodus ausgegeben. Wenn der Punkt außerhalb des Graphikfensters liegt, wird keine Aktion durchgeführt.

Einsprung-Bedingungen:

DE enthält die Anwender-X-Koordinate, auf der ausgegeben werden soll
HL enthält die Anwender-Y-Koordinate, auf der ausgegeben werden soll.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Ausgabeposition wird in Anwenderkoordinaten angegeben, d.h. relativ zum Anwender-Ursprung. Diese Routine ruft zum Plotten des Punktes den indirekten Verzweigpunkt GRA PLOT auf, der seinerseits den indirekten Verzweigpunkt GRA WRITE aufruft, um ein Pixel zu setzen (wenn innerhalb des Fensters).

Verwandte Einsprünge:

GRA PLOT
GRA PLOT RELATIVE
GRA TEST ABSOLUTE

79: GRA PLOT RELATIVE #BBED

Plotte einen Punkt relativ zur augenblicklichen Position.

Aktion:

Die augenblickliche Graphikposition wird an die gelieferte Position verlegt. Wenn diese innerhalb des Graphikfensters liegt, wird der Punkt in der momentanen Ink des Graphikstifts unter Beachtung des gerade gültigen Graphikdarstellungsmodus ausgegeben. Wenn der Punkt außerhalb des Graphikfensters liegt, wird keine Aktion durchgeführt.

Einsprung-Bedingungen:

DE enthält einen Abstand mit Vorzeichen auf der X-Adresse
HL enthält einen Abstand mit Vorzeichen auf der Y-Adresse.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Ausgabeposition wird in relativen Koordinaten angegeben, d.h. relativ zur augenblicklichen Graphikposition.

Diese Routine ruft zum Plotten des Punktes den indirekten Verzweigpunkt GRA PLOT auf, der seinerseits den indirekten Verzweigpunkt GRA WRITE aufruft, um ein Pixel zu setzen (wenn innerhalb des Fensters).

Verwandte Einsprünge:

GRA PLOT
GRA PLOT ABSOLUTE
GRA TEST RELATIVE

80: GRA TEST ABSOLUTE #BBF0

Teste einen Punkt an einer absoluten Position.

Aktion:

Die augenblickliche Graphikposition wird an die gelieferte Position verlegt. Wenn diese innerhalb des Graphikfensters liegt, wird der Punkt in der momentanen Ink des Graphikstifts unter Beachtung des gerade gültigen Graphikdarstellungsmodus ausgegeben. Wenn der Punkt außerhalb des Graphikfensters liegt, wird keine Aktion durchgeführt.

Einsprung-Bedingungen:

DE enthält die Anwender-X-Koordinate, die zu testen ist
HL enthält die Anwender-Y-Koordinate, die zu testen ist.

Aussprung-Bedingungen:

A enthält die Ink des angegebenen Punktes (oder die Ink des Graphikpapiers)
BC, DE, HL und die Flasgs sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Ausgabeposition wird in Anwenderkoordinaten angegeben, d.h. relativ zum Anwender-Ursprung.
Diese Routine ruft zum Plotten des Punktes den indirekten Verzweigpunkt GRA TEST auf, der seinerseits den indirekten Verzweigpunkt SCR READ aufruft, um ein Pixel zu setzen (wenn innerhalb des Fensters).

Verwandte Einsprünge:

GRA PLOT ABSOLUTE
GRA TEST
GRA TEST RELATIVE

81: GRA TEST RELATIVE #BBF3

Teste einen Punkt relativ zur aktuellen Position.

Aktion:

Die augenblickliche Graphikposition wird an die gewünschte Position verlegt. Wenn diese innerhalb des Graphikfensters liegt, wird der Punkt in der momentanen Ink des Graphikstifts unter Beachtung des gerade gültigen Graphikdarstellungsmodus ausgegeben. Wenn der Punkt außerhalb des Graphikfensters liegt, wird keine Aktion durchgeführt.

Einsprung-Bedingungen:

DE enthält einen Abstand mit Vorzeichen auf der X-Adresse
HL enthält einen Abstand mit Vorzeichen auf der Y-Adresse

Aussprung-Bedingungen:

A enthält die Ink des angegebenen Punktes (oder die Ink des Graphikpapiers)
BC, DE, HL und die Flasgs sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Ausgabeposition wird in relativen Koordinaten angegeben, d.h. relativ zur augenblicklichen Graphikposition.

Diese Routine ruft zum Plotten des Punktes den indirekten Verzweigpunkt GRA TEST auf, der seinerseits den indirekten Verzweigpunkt SCR READ aufruft, um das Pixel zu testen (wenn innerhalb des Fensters).

Verwandte Einsprünge:

GRA PLOT RELATIVE
GRA TEST
GRA TEST ABSOLUTE

82: GRA LINE ABSOLUTE #BBF6

Zeichne eine Linie zu einer absoluten Position.

Aktion:

Verlegen der augenblicklichen Graphikpsition an den angegebenen Endpunkt. Alle Punkte zwischen dieser und der vorigen Position, die innerhalb des Graphikfensters liegen, werden unter Berücksichtigung des momentanen Darstellungsmodus, in der augenblicklichen Ink des Graphikstifts dargestellt. Punkte außerhalb des Graphikfensters werden ignoriert.

Einsprung-Bedingungen:

DE enthält die Anwender-X-Koordinate des Endpunkts
HL enthält die Anwender-Y-Koordinate des Endpunkts.

Aussprung-Bedingungen:

AF, BC, DE, und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Position am Ende der Linie wird in Anwenderkoordinaten angegeben, d.h. relativ zum Anwender-Ursprung.

Diese Routine ruft zum Plotten des Punktes den indirekten Verzweigpunkt GRA LINE auf, der seinerseits den indirekten Verzweigpunkt SCR WRITE aufruft, um die Pixels zu schreiben (wenn innerhalb des Fensters).

Verwandte Einsprünge:

GRA LINE
GRA LINE RELATIVE

83: GRA LINE RELATIVE #BBF9

Zeichne eine Linie relativ zur augenblicklichen Position.

Aktion:

Verlegen der augenblicklichen Graphikpsition an den angegebenen Endpunkt. Alle Punkte zwischen dieser und der vorigen Position, die innerhalb des Graphikfensters liegen, werden unter Berücksichtigung des momentanen Darstellungsmodus, in der augenblicklichen Graphikstift-Ink dargestellt. Punkte außerhalb des Graphikfensters werden ignoriert.

Einsprung-Bedingungen:

DE enthält den Abstand mit Vorzeichen auf der X-Adresse
HL enthält den Abstand mit Vorzeichen auf der Y-Adresse.

Aussprung-Bedingungen:

AF, BC, DE, und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Ausgabeposition wird in relativen Koordinaten angegeben, d.h. relativ zur augenblicklichen Graphikposition.

Diese Routine ruft zum Plotten des Punktes den indirekten Verzweigpunkt GRA LINE auf, der seinerseits den indirekten Verzweigpunkt SCR WRITE aufruft, um die Pixel zu schreiben (wenn innerhalb des Fensters).

Verwandte Einsprünge:

GRA LINE
GRA LINE ABSOLUTE

84: GRA WR CHAR #BBFC

Schreibe ein Zeichen auf den Bildschirm an der augenblicklichen Graphikposition.

Aktion:

Ausgabe eines Zeichens auf dem Bildschirm an der momentanen Graphikposition.

Einsprung-Bedingungen:

A enthält das auszugebende Zeichen.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Das Zeichen wird mit der linken oberen Ecke an der augenblicklichen Graphikposition ausgegeben.

Alle Zeichen, auch Control-Codes (Zeichen #00...#1F), werden dargestellt.

Die augenblickliche Position wird um die Breite des Zeichens nach rechts verlegt (um für eine neue Zeichendarstellung bereit zu sein). Im Modus 0 beträgt diese Verlegung 32 Punkte, im Modus 1 16 Punkte und im Modus 2 8 Punkte.

Das Zeichen wird in der momentanen Ink des Graphikstifts und der Hintergrund in der momentanen Ink des Graphikpapiers dargestellt. Der Hintergrund wird in jedem Fall mit ausgegeben, auch wenn der Modus des Graphik-VDU auf „transparent" gestellt ist. Pixels, die außerhalb des Graphikfensters liegen, werden nicht dargestellt. Die Pixels werden über den indirekten Verzweigpunkt SCR WRITE ausgegeben, so daß der augenblickliche Graphikdarstellungsmodus berücksichtigt wird.

Verwandte Einsprünge:

TXT SET GRAPHIC
TXT WR CHAR

85: SCR INITIALISE #BBFF

Initialisiere das Bildschirmpaket.

Aktion:

Vollständige Initialisierung des Bildschirmpakets (wie während EMS). Sämtliche variablen und indirekten Verzweigungspunkte des Bildschirmpaketes werden initialisiert. Ebenso werden der Bildschirmmodus und die Inks auf ihre Standardwerte gesetzt.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die indirekten Verzweigungspunkte für den Bildschirm (SCR READ, SCR WRITE und SCR MODE CLEAR) werden auf ihre Standardroutinen gesetzt.
Die Inks werden auf ihre Standardfarben (siehe Anhang V) gesetzt.
Die Farbwechselperioden werden auf ihre Standardwerte gesetzt.
Der Bildschirm wird auf Modus 1 gesetzt.
Die Bildschirm-Basis wird so gesetzt, daß der Bildschirmspeicher bei #C000...#FFFF liegt (unterhalb des oberen ROM).
Der Bildschirmabstand wird auf 0 gesetzt.
Der Bildschirm wird mit Ink 0 gelöscht.
Der Darstellungsmodus des Graphik-VDU wird auf Modus FORCE gesetzt.
Das Ereignis-Kennzeichen für den Farbwechsel-Bildaufbau wird gesetzt.

Die Initialisierung wird in einer Weise durchgeführt, daß der frühere Bildschirminhalt nicht mehr sichtbar wird (bei EMS ist der Inhalt zufällig).

Verwandte Einsprünge:

GRA INITIALISE
SCR RESET
TXT INITIALISE

86: SCR RESET #BC02

Setze das Bildschirmpaket zurück.

Aktion:

Neue Initialisierung der indirekten Verzweigungspunkte (Indirections) für das Bildschirmpaket und die Farben der Inks.
Auch die Farbwechselgeschwindigkeit und der Darstellungsmodus des Graphik-VDU werden neu initialisiert.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die indirekten Verzweigungspunkte für den Bildschirm (SCR READ, SCR WRITE und SCR MODE CLEAR) werden auf ihre Standardroutinen gesetzt.
Die Inks werden auf ihre Standardfarben gesetzt (siehe Anhang V),
Die Farbwechselperioden werden auf ihre Standardwerte gesetzt.

Der Darstellungsmodus des Graphik-VDU wird auf FORCE gesetzt.

Die Inks werden nicht an die Hardware weitergegeben. Dies geschieht erst beim nächsten Farbwechsel.

Verwandte Einsprünge:

SCR INITIALISE
SCR SET ACCESS
SCR SET FLASHING
SCR SET INK

87: SCR SET OFFSET #BC05

Setze den Offset für den Beginn des Bildschrims.

Aktion:

Angabe des Abstands des ersten Zeichens auf dem Bildschirm. Durch Änderung dieses Abstands kann der Bildschirm „gerollt“ werden.

Einsprung-Bedingungen:

HL enthält den geforderten Abstand.

Aussprung-Bedingungen:

AF und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Der angegebene Abstand wird mit der Maske #O7FE verknüpft, um sicherzugehen, daß er nicht zu groß ist und daß er geradzahlig ist. (Der Bildschirm kann nur in Schritten von 2 Bytes „gerollt“ werden).

Die Bildschirmbasis wird zur Errechnung der Bildschirmadressen von SCR CHAR POSITION und SCR DOT POSITION herangezogen. Wenn der Bildschirmabstand fälschlicherweise durch den Aufruf der Gerätebehandlungsroutine MC SCREEN OFFSET verändert wird, dann benutzen Text- und Graphik-VDU falsche Adressen.

Wenn der Bildschirmmodus gesetzt oder der Bildschirm durch SCR CLEAR gelöscht wird, wird der Abstand auf Null gesetzt.

Verwandte Einsprünge:

MC SCREEN OFFSET
SCR GET LOCATION
SCR HW ROLL
SCR SET BASE

88: SCR SET BASE #BC08

Angabe des RAM-Bereichs für den Bildschirmspeicher.

Aktion:

Angabe der Basisadresse für den Bildschirmspeicher. Dies kann benutzt werden, um den Bildschirm aus dem Bereich unterhalb des oberen ROM auszulagern oder beispielsweise auch einen vorbereiteten Bildschirm auszugeben.

Einsprung-Bedingungen:

A enthält das signifikante Byte der Basisadresse.

Aussprung-Bedingungen:

AF und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Da der Bildschirmspeicher nur an einer 16-K-Grenze liegen kann, wird der angegebene Wert mit der Maske #C0 verknüpft. Die Standardbasis für den Bildschirm, bei EMS gesetzt, ist #C0.

Der Bildschirmabstand wird mit der Bildschirmbasis zu einem einzelnen Wert kombiniert, der an die Hardware weitergegeben wird.
Die Bildschirmbasis wird zur Errechnung der Bildschirmadressen von SCR CLEAR POSITION und SCR DOT POSITION verwendet. Wenn die Basisadresse fälschlicherweise durch einen Aufruf der Gerätebehandlungsroutine MC SCREEN OFFSET verändert wird, dann arbeiten Text- und Graphik-VDU mit falschen Bildschirmadressen.

Der Bildschirmspeicher wird nicht gelöscht, wenn die Bildschirmbasis gesetzt wird. Dazu muß SCR CLEAR aufgerufen werden.

Verwandte Einsprünge:

MC SCREEN OFFSET
SCR GET LOCATION
SCR SET OFFSET

89: SCR GET LOCATION #BC0B

Hole die aktuellen Werte für Basis und Offset.

Aktion:

Abfrage, wo der Bildschirmspeicher liegt und wo der Anfang des Bildschirms ist.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A enthält das signifikante Byte der Basisadresse
HL enthält den augenblicklichen Abstand (Offset)!
Flags sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die von dieser Routine zurückgegebenen Werte für BASIS und Abstand müssen nicht mit denen übereinstimmen, die über SCR SET BASIS und SCR SET OFFSET gesetzt wurden, da dort die Werte mit einer Maske verknüpft wurden, um sie gültig zu machen; außerdem wird der Bildschirmabstand noch verändert, wenn die Hardwareroutine zum „Rollen" des Bildschirms SCR HW ROLL benutzt wird.

Verwandte Einsprünge:

SCR SET BASE
SCR SET OFFSET

90: SCR SET MODE #BC0E

Setze einen neuen Bildschirmmodus.

Aktion:

Setzen des Bildschirms in einen neuen Modus und sicherstellen, daß die Text- und Graphik-VDUs korrekt aufgebaut werden.

Einsprung-Bedingungen:

A enthält den gewünschten Modus

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Der angegebene Modus wird mit der Maske #03 verknüpft. Wenn das Ergebnis 3 ist, wird keine Aktion durchgeführt. Ansonsten wird der Bildschirm auf eine der folgenden Modi gesetzt:

Modus 0:	160 x 200 Pixels (Punkte)	20 x 25 Zeichen
Modus 1:	320 x 200 Pixels (Punkte)	40 x 25 Zeichen
Modus 2:	640 x 200 Pixels (Punkte)	80 x 25 Zeichen

Der Bildschirm wird vorher gelöscht, um zu vermeiden, daß der alte Bildschirminhalt im falschen Modus dargestellt wird; dies geschieht über den Aufruf des indirekten Verzweigungspunkts SCR MODE CLEAR.

Alle Text- und Graphikfenster werden so gesetzt, daß sie den ganzen Bildschirm belegen und der graphische Anwender-Ursprung wird auf die linke untere Ecke des Bildschirms gelegt. Die Cursordarstellungen für alle Text- Ein-/Ausgabegeräte (streams) werden abgeschaltet.

Die aktuellen Inks für Text- und Graphikstift und Papier werden mit einer Maske verknüpft, die, abhängig vom neuen Modus, die Gültigkeit gewährleistet (siehe TXT SET PEN usw.). Wenn der Modus auf einen Modus verändert wird, der weniger Inks auf dem Bildschirm zuläßt, kann dies einen Wechsel der Stift- oder Papierinks erfordern.

Verwandte Einsprünge:

MC SET MODE
SCR GET MODE

91: SCR GET MODE #BC11

Hole den aktuellen Bildschirmmodus.

Aktion:

Holen und Testen des momentanen Bildschirmmodus.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn der augenblickliche Modus 0 ist:
CARRY-Flag „an“
ZERO-Flag „aus“
A enthält 0

Wenn der augenblickliche Modus 1 ist:
CARRY-Flag „aus“
ZERO-Flag „an“
A enthält 1

Wenn der augenblickliche Modus 2 ist:
CARRY-Flag „aus“
ZERO-Flag „aus“
A enthält 2

Immer:
die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Modi sind:

Modus 0:	160 x 200 Pixels	20 x 25 Zeichen
Modus 1:	320 x 200 Pixels	40 x 25 Zeichen
Modus 2:	640 x 200 Pixels	80 x 25 Zeichen

Verwandte Einsprünge:

SCR SET MODE

92: SCR CLEAR #BC14

Lösche den Bildschirm (auf Ink 0).

Aktion:

Lösche den gesamten Bildschirmspeicher mit Null.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Zunächst wir der Inkwechsel abgeschaltet und alle Inks werden auf dieselbe Farbe gesetzt wie Ink 0. Dadurch wird ein sofortiges Löschen des Bildschirms erreicht. Wenn der gesamte Bildschirmspeicher auf Null gesetzt ist, wird der Inkwechsel wieder angeschaltet (ein Inkwechsel-„Ereignis“ wird in die Bildaufbau-Quene eingestellt) und alle Inks werden auf ihre richtigen Farben gesetzt.

Wenn die Papier-Ink für Text und Graphik nicht auf 0 gesetzt wird, wird dies beim Darstellen von Zeichen oder Löschen von Fenstern sichtbar.
Der Bildschirmabstand wird auf 0 gesetzt.

Verwandte Einsprünge:

GRA CLEAR WINDOW
SCR MODE CLEAR
TXT CLEAR WINDOW

93: SCR CHAR LIMITS #BC17

Abfrage nach der Bildschirmgröße in Zeichen.

Aktion:

Hole die letzte Zeichenreihe und -spalte des Bildschirms im akutellen Modus.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

B enthält die letzte physikalische Spalte des Bildschirms
C enthält die letzte physikalische Reihe des Bildschirms
AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Bildschirmecken werden in physikalischen Koordinaten angegeben, d.h. Reihe 0, Spalte 0 ist die obere linke Ecke des Bildschirms. Dies bedeutet, daß die letzte Spalte des Bildschirms im Modus 0 die Spalte 19, im Modus 1 die Spalte 39 und im Modus 2 die Spalte 79 ist. Die letzte Reihe ist in allen Modi immer die Reihe 24.

Verwandte Einsprünge:

SCR GET MODE

94: SCR CHAR POSITION #BC1A

Übersetze physikalische Koordinten in eine Bildschirmposition.

Aktion:

Errechnen der Bildschirmadresse der linken oberen Ecke einer Zeichenposition auf dem Bildschirm. Zusätzlich wird die Größe eines Zeichens im augenblicklichen Modus zurückgegeben.

Einsprung-Bedingungen:

H enthält die physikalische Zeichenspalte
L enthält die physikalische Zeichenreihe

Aussprung-Bedingungen:

HL enthält die Bildschirmadresse der linken oberen Ecke des Zeichens
B enthält die Größe eines Zeichens im Bildschirmspeicher in Bytes
AF ist zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Zeichenposition wird in physikalischen Koordinaten angegeben, d.h. Reihe 0, Spalte 0 ist die linke obere Ecke des Bildschirms.

Die angegebene Zeichenposition wird nicht auf Gültigkeit geprüft. Eine ungültige Position (außerhalb der Bildschirmgrenzen) erzeugt eine Bildschirmadresse, die nichts aussagt.
Die Umrechnung auf die Bildschirmadresse geschieht nach folgender Formel:

Bildschirmadresse = Bildschirmbasis + (Block-Abstand MOD #0800)

wobei:

Blockabstand = (Reihe ★ 80) + (Spalte ★ Größe) + Bildschirmabstand

und:

Bildschirmbasis ist die Anfangsadresse des Bildschirmspeichers.

Größe ist die Größe eines Zeichens in Bytes im augenblicklichen Modus (4 in Modus 0, 2 in Modus 1 und 1 in Modus 2).

Bildschirmabstand ist der Abstand des ersten Bytes, das auf dem Bildschirm ausgegeben werden soll.

Verwandte Einsprünge:

SCR DOT POSITION
SCR NEXT BYTE
SCR NEXT LINE
SCR PREV BYTE
SCR PREV LINE

95: SCR DOT POSITION #BC1D

Übersetze Basis-Koordinaten in eine Bildschirmposition.

Aktion:

Errechnen der Bildschirmadresse und Maske für einen Bildpunkt (Pixel). Zusätzlich wird ein Hinweis auf die Anzahl von Punkten in einem Bildschirmbyte im augenblicklichen Modus zurückgegeben.

Einsprung-Bedingungen:

DE enthält die Basis-X-Koordinate eines Bild-Punktes
HL enthält die Basis-Y-Koordinate eines Bild-Punktes.

Aussprung-Bedingungen:

HL enthält die Bildschirmadresse des Bild-Punktes
C enthält die Maske für den Bild-Punkt
B enthält die um 1 verminderte Anzahl von Bild-Punkten in einem Byte
AF und DE sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Position des Bild-Punktes wird in Basis-Koordinaten angegeben, d.h. 0,0 ist der Bild-Punkt in der linken unteren Ecke des Bildschirms und jede Koordinatenangabe bestimmt einen einzelnen Bild-Punkt.

Die Position des Bildpunktes wird nicht auf Gültigkeit überprüft (innerhalb des Bildschirms). Bei einer ungültigen Position hat die errechnete Bildschirmadresse keinen Aussagewert.

Die Umrechnung auf die Bildschirmadresse geschieht nach folgender Formel:

Bildschirmadresse = Bildschirmbasis + (Zeile in Reihe ★ #0800) + (Reihenabstand MOD #0800)

wobei:

Bildschirmbasis = ist die Startadresse des Bildschirmspeichers
Zeile in Reihe = (199 – Y-Koordinate) MOD 8
Reihenabstand = (Reihennummer ★ 80) + Byte in der Reihe + Bildschirmabstand

und:

Reihennummer = (199 – Y-Koordinate) /8
Byte in der Reihe = X-Koordinate / Bytegröße

Bildschirmabstand ist der Abstand des ersten Bytes, das auf dem Bildschirm ausgegeben werden soll.

Bytegröße ist die Anzahl von Bildpunkten in einem Byte im augenblicklichen Modus (2 bei 0, 4 bei 1, 8 bei 2).

X-Koordinate MOD Bytegröße wird benutzt, um die Maske für den zugehörigen Bildpunkt zu errechnen.

Verwandte Einsprünge:

SCR CHAR POSITION
SCR NEXT BYTE
SCR NEXT LINE
SCR PREV BYTE
SCR PREV LINE

96: SCR NEXT BYTE #BC20

Gehe mit der Bildschirmadresse um 1 Byte nach rechts.

Aktion:

Errechnen der Bildschirmadresse für das Byte, das rechts der zur Verfügung gestellten Bildschirmadresse folgt.

Einsprung-Bedingungen:

HL enthält eine Bildschirmadresse.

Aussprung-Bedingungen:

HL enthält die berichtigte Bildschirmadresse
AF ist zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das Erreichen einer Position nach dem Ende einer Bildschirmzeile wird nicht überwacht. Die Bildschirmadresse verweist lediglich auf das nächste Byte im Bildschirmblock. Dies ist normalerweise das erste Byte einer Bildschirmzeile 8 Bildschirmzeilen unterhalb der alten Zeile (d.h. eine Zeichenreihe unterhalb). Beim Erreichen einer Position nach dem Ende der letzten Bildschirmzeile eines Blockes jedoch zeigt die Bildschirmadresse auf den Beginn der 48 Bytes im Block, die nicht auf dem Bildschirm ausgegeben werden.

Diese Routine ist zur Veränderung der Bildschirmadresse bei der Ausgabe von Zeichen oder beim Zeichnen von Linien auf dem Bildschirm vorgesehen.

Verwandte Einsprünge:

SCR CHAR POSITION
SCR DOT POSITION
SCR NEXT LINE
SCR PREV BYTE
SCR PREV LINE

97: SCR PREV BYTE #BC23

Gehe mit der Bildschirmadresse um 1 Byte nach links.

Aktion:

Errechnen der Bildschirmadresse für das Byte, das links der zur Verfügung gestellten Bildschirmadresse vorausgeht.

Einsprung-Bedingungen:

HL enthält eine Bildschirmadresse.

Aussprung-Bedingungen:

HL enthält die berichtigte Bildschirmadresse
AF ist zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das Erreichen einer Position nach dem Ende einer Bildschirmzeile wird nicht überwacht. Die Bildschirmadresse verweist lediglich auf das vorausgehende Byte im Bildschirmblock. Dies ist normalerweise das letzte Byte einer Bildschirmzeile 8 Bildschirmzeilen oberhalb der alten Zeile (d.h. eine Zeichenreihe oberhalb).

Beim Erreichen einer Position nach dem Beginn der ersten Bildschirmzeile eines Blokkes jedoch zeigt die Bildschirmadresse auf den Beginn der 48 Bytes im Block, die nicht auf dem Bildschirm ausgegeben werden.

Diese Routine ist zur Veränderung der Bildschirmadresse bei der Ausgabe von Zeichen oder beim Zeichnen von Linien auf dem Bildschirm vorgesehen.

Verwandte Einsprünge:

SCR CHAR POSITION
SCR DOT POSITION
SCR NEXT BYTE
SCR NEXT LINE
SCR PREV LINE

98: SCR NEXT LINE #BC26

Erhöhe die Bildschirmadresse um 1 Zeile.

Aktion:

Errechnen der Bildschirmadresse für das Byte, das unterhalb der zur Verfügung gestellten Bildschirmadresse folgt.

Einsprung-Bedingungen:

HL enthält eine Bildschirmadresse.

Aussprung-Bedingungen:

HL enthält die berichtigte Bildschirmadresse
AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Das Erreichen einer Position unterhalb des Bildschirms wird nicht überwacht (und auch nicht empfohlen); die Bildschirmadresse kann dann nicht mehr verwendet werden.

Diese Routine ist zur Veränderung der Bildschirmadresse bei der Ausgabe von Zeichen oder beim Zeichnen von Linien auf dem Bildschirm vorgesehen.

Verwandte Einsprünge:

SCR CHAR POSITION
SCR DOT POSITION
SCR NEXT LINE
SCR PREV BYTE
SCR PREV LINE

99: SCR PREV LINE #BC29

Vermindere die Bildschirmadresse um 1 Zeile.

Aktion:

Errechnen der Bildschirmadresse für das Byte, das oberhalb der zur Verfügung gestellten Bildschirmadresse steht.

Einsprung-Bedingungen:

HL enthält eine Bildschirmadresse.

Aussprung-Bedingungen:

HL enthält die berichtigte Bildschirmadresse
AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Das Erreichen einer Position oberhalb des Bildschirms wird nicht überwacht (und auch nicht empfohlen); die Bildschirmadresse kann dann nicht mehr verwendet werden.

Diese Routine ist zur Veränderung der Bildschirmadresse bei der Ausgabe von Zeichen oder beim Zeichnen von Linien auf dem Bildschirm vorgesehen.

Verwandte Einsprünge:

SCR CHAR POSITION
SCR DOT POSITION
SCR NEXT BYTE
SCR NEXT LINE
SCR PREV BYTE

100: SCR INK ENCODE #BC2C

Verschlüsseln einer Ink, die alle Bildpunkte (Pixel) eines Bytes bedecken soll.

Aktion:

Umsetzen einer Ink in eine codierte Form, so daß alle Bildpunkte des Bytes auf diese Ink gesetzt werden. Diese verschlüsselte Ink kann dann mit einer Maske verknüpft werden, um den richtigen Wert zu erzeugen, der einem einzelnen Bildpunkt zugeordnet werden soll.

Einsprung-Bedingungen:

A enthält eine Inknummer.

Aussprung-Bedingungen:

A enthält die codierte Ink
Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das Verschlüsseln ist nicht einfach, da die Pixels in einem Byte verstreut und auch die Bits eines Pixels in keiner deutlichen Folge liegen.

Die Bits eines Pixels sind (vom am meisten signifikanten zum am wenigsten signifikanten):

	Modus 0	Modus 1	Modus 2
Pixel ganz links		Bit 3,7	Bit 7
	Bit 1,5,3,7		Bit 6
		Bit 2,6	Bit 5
			Bit 4
		Bit 1,5	Bit 3
	Bit 0,4,2,6		Bit 2
		Bit 0,4	Bit 1
Pixel ganz rechts			Bit 0

Die Text- und Graphik-VDU's speichern ihre Stift- und Papier-Inks für eine schnelle interne Benutzung in dieser verschlüsselten Form. Dies spart für jedes auszugebende Pixel Zeit.

Das Verschlüsseln ist in den verschiedenen Modi unterschiedlich und so müssen alle Inks neu verschlüsselt werden, wenn der Bildschirmmodus sich ändert. SCR SET MODE übernimmt dies automatisch für die Stift- und Papier-Inks des Text-VDU und des Graphik-VDU.

Verwandte Einsprünge:

SCR INK DECODE

101: SCR INK DECODE #BC2F

Entschlüsseln einer verschlüsselten Ink.

Aktion:

Setze eine verschlüsselte Ink in die entsprechende Inknummer um.

Einsprung-Bedingungen:

A enthält die verschlüsselte Ink.

Aussprung-Bedingungen:

A enthält die Inknummer
Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Entschlüsselung wird durchgeführt, indem die verschlüsselte Ink des Pixels ganz links entschlüsselt wird. Die Ink für dieses Pixel ist in den folgenden Bits verschlüsselt (vom am meisten zum wenigsten signifikanten Bit):

Modus 0:	Bit 1, 5, 3, 7
Modus 1:	Bit 3, 7
Modus 2:	Bit 7

Verwandte Einsprünge:

SCR INK ENCODE

102: SCR SET INK #BC32

Setze die zur Darstellung einer Ink erforderlichen Farben.

Aktion:

Angabe der beiden Farben, die zur Ausgabe einer Ink benutzt werden sollen. Wenn die beiden Farben übereinstimmen, bleibt die Ink in der gleichen Farbe, ansonsten wechselt die Ink zwischen diesen Farben.

Einsprung-Bedingungen:

A enthält eine Inknummer
B enthält die erste Farbe
C enthält die zweite Farbe.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Inknummer wird mit der Maske #0F verknüpft, um sicher zu gehen, daß sie gültig ist, und die Farben werden mit der Maske #1F verknüpft. Die Farben 27...31 sind nicht zur Benutzung vorgesehen, sie sind lediglich Duplikate anderer möglicher Farben.

Die neuen Farben einer Ink werden nicht unverzüglich an die Hardware weitergegeben. Sie werden gespeichert und erscheinen beim nächsten Bildaufbau auf dem Bildschirm.

Die Zeitspanne, für die jeweils eine Farbe auf dem Bildschirm dargestellt werden soll, kann beim Aufruf von SCR SET FLASHING angegeben werden.

Die Inks werden bei EMS und beim Aufruf von SCR RESET auf ihre Standardfarben gesetzt.

Die verschiedenen Farbmöglichkeiten und die Standardfarben für die Inks sind in Anhang V beschrieben.

Verwandte Einsprünge:

GRA SET PAPER
GRA SET PEN
SCR GET INK
SCR SET BORDER
SCR SET FLASHING
TXT SET PAPER
TXT SET PEN

103: SCR GET INK #BC35

Hole die aktuellen Ink-Farben.

Aktion:

Holen der beiden Farben, die verwendet werden, um eine Ink am Bildschirm darzustellen.

Einsprung-Bedingungen:

A enthält eine Inknummer.

Aussprung-Bedingungen:

B enthält die erste Farbe
C enthält die zweite Farbe
AF, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Inknummer wird mit der Maske #0F verknüpft, um sicherzugehen, daß sie gültig ist. Die zurückgegebenen Farben müssen mit den ursprünglich gesetzten Farben nicht übereinstimmen, da sie beim Setzen mit einer Maske verknüpft wurden.

Die neuen Farben für eine Ink werden nicht unverzüglich an die Hardware weitergegeben. Sie werden gespeichert und erscheinen beim nächsten Bildaufbau auf dem Bildschirm. Dies bedeutet, daß die zurückgegebenen Farben für den Anwender noch nicht unbedingt sichtbar waren.

Die verschiedenen Farbmöglichkeiten und die Standardfarben für die Inks sind in Anhang V beschrieben.

Verwandte Einsprünge:

GRA GET PAPER
GRA GET PEN
SCR GET BORDER
SCR SET INK
TXT GET PAPER
TXT GET PEN

104: SCR SET BORDER #BC38

Setze die Farben für den Bildschirmrand.

Aktion:

Angabe der beiden Farben, die zur Ausgabe eines Rahmens benutzt werden sollen. Wenn beide Farben übereinstimmen, bleibt der Rahmen in der gleichen Farbe, ansonsten wechselt der Rahmen zwischen diesen Farben.

Einsprung-Bedingungen:

B enthält die erste Farbe
C enthält die zweite Farbe.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Farben werden mit der Maske #1F verknüpft, um sicherzugehen, daß sie gültig sind. Die Farben 27...31 sind nicht für die Anwendung vorgesehen, sie stellen lediglich Duplikate anderer möglicher Farben dar.

Die neuen Farben eines Rahmens werden nicht unverzüglich an die Hardware weitergegeben. Sie werden gespeichert und erscheinen beim nächsten Bildaufbau auf dem Bildschirm.

Die Zeitspanne, für die jeweils eine Farbe auf dem Bildschirm dargestellt werden soll, kann beim Aufruf von SCR SET FLASHING angegeben werden.

Der Rahmen wird bei EMS und beim Aufruf von SCR RESET auf seine Standardfarben gesetzt.

Die verschiedenen Farbmöglichkeiten und die Standardfarben sind in Anhang V beschrieben.

Verwandte Einsprünge:

SCR GET BORDER
SCR SET FLASHING
SCR SET INK

105: SCR GET BORDER #BC3B

Hole die aktuellen Farben des Bildschirmrandes.

Aktion:

Hole die beiden Farben, die zur Darstellung des Rahmens auf dem Bildschirm benutzt werden.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

B enthält die erste Farbe
C enthält die zweite Farbe
AF, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die zurückgegebenen Farben müssen mit den ursprünglich gesetzten Farben nicht übereinstimmen, da sie beim Setzen mit einer Maske verknüpft wurden.

Die neuen Farben für einen Rahmen werden nicht unverzüglich an die Hardware weitergegeben. Sie werden gespeichert und erscheinen beim nächsten Bildaufbau auf dem Bildschirm.

Dies bedeutet, daß die zurückgegebenen Farben für den Anwender noch nicht unbedingt sichtbar waren.

Die verschiedenen Farbmöglichkeiten und die Standardfarben sind in Anhang V beschrieben.

Verwandte Einsprünge:

SCR GET INK
SCR SET BORDER

106: SCR SET FLASHING #BC3E

Setze die Blink-Perioden.

Aktion:

Angabe, wie lange jede der beiden Farben für die Inks und für den Rahmen auf dem Bildschirm dargestellt werden sollen. Diese Angabe gilt für alle Inks und den Rahmen.

Aussprung-Bedingungen:

H enthält die Zeitspanne für die erste Farbe
L enthält die Zeitspanne für die zweite Farbe

Aussprung-Bedingungen:

AF und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Blinkperioden werden in Bildaufbaueinheiten angegeben (1/50 oder 1/60 Sek.). Eine Periode von 0 bedeutet 256.

Die Standardangabe für die Blink-Periode ist 10 Bildschirmaufbaueinheiten (1/5 oder 1/6 Sekunden). Dies wird bei EMS gesetzt und wenn SCR RESET aufgerufen wird.

Die neuen Blink-Perioden werden nicht sofort wirksam, sondern erst beim nächsten Wechsel.

Verwandte Einsprünge:

SCR GET FLASHING
SCR SET BORDER
SCR SET INK

107: SCR GET FLASHING #BC41

Hole die aktuellen Blink-Perioden.

Aktion:

Hole die Dauer, für die jede der beiden Farben für Ink und Rahmen ausgegeben wird.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

H enthält die Zeitspanne für die erste Farbe
L enthält die Zeitspanne für die zweite Farbe
AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Blink-Perioden werden in Bildaufbaueinheiten (1/50 oder 1/60 Sek.) angegeben.
Eine Periode von 0 bedeutet 256.

Verwandte Einsprünge:

SCR SET FLASHING

108: SCR FILL BOX #BC44

Fülle einen Zeichenbereich des Bildschirms mit einer Ink.

Aktion:

Fülle einen rechteckigen Bereich auf dem Bildschirm mit einer Ink. Die Grenzen dieses Bereichs werden in Zeichenposition angegeben.

Einsprung-Bedingungen:

A enthält die codierte Ink, mit der der Bereich gefüllt werden soll
H enthält die physikalische linke Spalte des zu füllenden Bereichs
D enthält die physikalische rechte Spalte des zu füllenden Bereichs
L enthält die physikalische obere Reihe des zu füllenden Bereichs
E enthält die physikalische untere Reihe des zu füllenden Bereichs

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Bereichsgrenzen sind in physikalischen Koordinaten angegeben, d.h. Reihe 0, Spalte 0 ist die linke obere Ecke des Bildschirms. Sie werden nicht auf Gültigkeit geprüft. Ungültige Grenzen (außerhalb des Bildschirms) rufen nicht vorhersehbare Ergebnisse hervor.

Der Bildschirm wird ohne Benutzung einer anderen Routine direkt geschrieben. Der augenblickliche Darstellungsmodus des Graphik-VDU wird daher ignoriert.

Verwandte Einsprünge:

SCR CLEAR
SCR FLOOD BOX
TXT CLEAR WINDOW

109: SCR FLOOD BOX #BC47

Fülle einen Bytebereich des Bildschirms.

Aktion:

Füllen eines rechteckigen Bereich auf dem Bildschirm mit einer Ink. Die Grenzen des Bereichs müssen auf einer Bytegrenze liegen. Die Routine füllt einen willkürlichen Bildschirmbereich nicht auf eine Bildpunktgrenze (Pixel) auf.

Einsprung-Bedingungen:

C enthält die codierte Ink, mit der der Bereich aufgefüllt werden soll
HL enthält die Bildschirmadresse der linken oberen Ecke des zu füllenden Bereichs
D enthält die Breite (ohne Vorzeichen) des zu füllenden Bereichs in Bytes
E enthält die Höhe (ohne Vorzeichen) des zu füllenden Bereichs in Bildschirmzeichen

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das gesamte Rechteck, das gelöscht werden soll, muß auf dem Bildschirm liegen, ansonsten sind die Folgen nicht vorhersehbar.

Eine Höhe oder Breite von 0 bedeutet 256 (was zu groß ist, um auf den Bildschirm zu passen).

Der Bildschirm wird ohne eine andere Schreibroutine zu benutzen direkt geschrieben. Der augenblickliche Darstellungsmodus des Graphik-VDU wird deshalb ignoriert.

Verwandte Einsprünge:

GRA CLEAR WINDOW
SCR CLEAR
SCR FILL BOX

110: SCR CHAR INVERT #BC4A

Invertiere eine Zeichenposition.

Aktion:

Alle Bildpunkte einer Zeichenposition, die in einer Farbe dargestellt werden, werden in einer zweiten Farbe neu dargestellt und umgekehrt. Dies ergibt eine inverse Darstellung einer Zeichenposition. Eine zweite Invertierung des Zeichens ergibt wieder die ursprünglichen Inks. Dieser Effekt wird benutzt, um die Cursor des Text-VDU's darzustellen.

Einsprung-Bedingungen:

B enthält eine codierte Ink
C enthält eine andere codierte Ink
H enthält eine physikalische Zeichenspalte
L enthält eine physikalische Zeichenreihe

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Zeichenposition wird in physikalischen Koordinaten angegeben, d.h. Reihe 0, Spalte 0 ist die linke obere Ecke des Bildschirms.

Die angegebenen Zeichenpositionen werden nicht auf Gültigkeit geprüft. Eine ungültige Position (außerhalb des Bildschirms) kann unvorhersehbare Folgen haben.

Alle Bildpunkte auf der Zeichenposition werden mit einem EXCLUSIVEN-ODER des EXCLUSIVEN-ODER'S der beiden Inks verknüpft. Bildpunkte auf der Zeichenposition in einer der angegebenen Inks werden somit auf die andere Ink gesetzt. Bildpunkte, die auf die andere Ink gesetzt sind, werden ebenso geändert.

Verwandte Einsprünge:

TXT PLACE CURSOR
TXT REMOVE CURSOR

111: SCR HW ROLL #BC4D

Bewege den gesamten Bildschrim um 8 Bildpunktzeilen (1 Zeichen) nach oben oder unten.

Aktion:

„Rollen" des Bildschirms durch die Hardware. Die neue Zeile auf dem Bildschirm ist gelöscht.

Einsprung-Bedingungen:

Wenn der Bildschirm „heruntergerollt" werden soll:
B muß Null sein

Wenn der Bildschirm „hochgerollt" werden soll:
B muß ungleich Null sein

Immer:
A enthält die codierte Ink, mit der die neue Zeile gelöscht werden soll.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Bildschirm wird durch Veränderung des Bildschirmabstandes (siehe SCR SET OFFSET) „gerollt".
Ein „Hochrollen" des Bildschirms bewegt den Bildschirminhalt nach oben und die neue Fußzeile wird gelöscht. Der Bildschirmabstand wird deshalb um 80 (MOD #0800) erhöht.

Ein „Herunterrollen" des Bildschirms bewegt den Bildschirminhalt nach unten und die neue Kopfzeile wird gelöscht. Der Bildschirmabstand wird deshalb um 80 (MOD #0800) vermindert.
Die neue Zeile wird durch eine direkte Ausgabe gelöscht, so daß der Darstellungsmodus des Graphik-VDU ignoriert wird.
Der Text VDU-Zähler für das „Rollen" wird durch diese Routine nicht verändert (siehe TXT GET WINDOW).

Besondere Vorsicht wurde verwendet, um sicherzustellen, daß der Bildschirm während des „Rollens“ und speziell während des Löschens der neuen Zeile noch vorzeigbar aussieht.

Dies beruht prinzipiell in einem Löschen der neuen Zeile in zwei Teilen. Zunächst wird der Teil, der auf dem Bildschirm nicht sichtbar ist (zurückzuführen auf die Bildschirmadressierung), gelöscht. Dann, nach dem Warten auf einen neuen Bildaufbau und Veränderung des Bildschirmabstands, wird die zweite Hälfte der Zeile, die Bestandteil der Zeile war, die gerade vom Bildschirm „weggerollt“ wurde, gelöscht.

Verwandte Einsprünge:

SCR SET OFFSET
SCR SW ROLL

112: SCR SW ROLL #BC50

Bewege einen Bildschirmbereich um 8 Bildpunktzeilen (1 Zeichen) nach oben oder unten.

Aktion:

„Rollen" eines Bildschirmbereichs durch Kopieren. Der Bereich, der „gerollt" werden soll, wird in Zeichenpositionen angegeben.

Einsprung-Bedingungen:

Wenn der Bildschirm „heruntergerollt" werden soll:
B muß Null sein

Wenn der Bildschirm „hochgerollt" werden soll:
B muß ungleich Null sein

Immer:
A enthält die codierte Ink, mit der die neue Zeile gelöscht werden soll.

Aussprung-Bedingungen:

H enthält die physikalische linke Spalte des zu „rollenden" Bereichs
D enthält die physikalische rechte Spalte des zu „rollenden" Bereichs
L enthält die physiskalische Kopfzeile des zu „rollenden" Bereichs
E enthält die physiskalische Fußzeile des zu „rollenden" Bereichs.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Bereichsgrenzen werden in physikalischen Koordinaten angegeben, d.h. Reihe 0, Spalte 0, ist die obere linke Ecke des Bildschirms. Die Grenzen (außerhalb des Bildschirms) verursachten unvorhersehbare Ergebnisse.

Ein „Hochrollen“ des Bildschirms bewegt den Bildschirminhalt nach oben und die neue Fußzeile wird gelöscht. Ein „Herunterrollen“ des Bildschirms bewegt den Bildschirminhalt nach unten und die neue Kopfzeile wird gelöscht. Die neue Zeile wird durch eine direkte Ausgabe gelöscht, so daß der Darstellungsmodus des Graphik-VDU ignoriert wird.

Der Text VDU-Zähler für das „Rollen“ wird durch diese Routine nicht verändert (siehe TXT GET WINDOW).

Besondere Vorsicht wurde verwendet, um sicherzustellen, daß der Bildschirm während des „Rollens“ vorzeigbar aussieht. Dies beruht prinzipiell in einem Warten auf den neuen Bildschirmaufbau, bevor das Kopieren durchgeführt wird.

Verwandte Einsprünge:

SCR HW ROLL

113: SCR UNPACK #BC53

Erweitere eine Zeichenmatrix für den augenblicklichen Bildschirmmodus.

Aktion:

Umsetzen einer Matrix von ihrer Standardform in einen Satz von Bildpunktmasken, die dem momentanen Bildschirmmodus entsprechen.

Einsprung-Bedingungen:

HL enthält die Matrixadresse
DE enthält die Adresse eines Bereiches, in den entpackt werden soll.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Matrix wird in eine Reihe von Masken umgesetzt, welche alle Bildschirmbytes des Zeichens umfassen. dies bedeutet, daß jedes Byte der Matrix umgesetzt wird in 4 Bytes im Modus 0, 2 Bytes im Modus 1 und 1 Byte im Modus 2.

Somit muß der Bereich zum entpacken 32, 16 oder 8 Bytes lang sein.

Wenn ein Bit in der Matrix gesetzt ist, dann wird die entsprechende Bildpunktmaske in die entpackte Version aufgenommen (die Bits werden auf 1 gesetzt). Ansonsten wird die Bildpunktmaske nicht in die entpackte Version aufgenommen (die Bits werden auf Null gesetzt).

Verwandte Einsprünge:

SCR REPACK

114: SCR REPACK #BC56

Komprimiere eine Zeichenmatrix auf ihre Standardform.

Aktion:

Ein Zeichen auf dem Bildschirm wird in eine Matrix umgesetzt, indem jeder Bildpunkt mit einer Ink verglichen wird. Wenn der Bildpunkt auf diese Ink gesetzt ist, wird das zugehörige Bit in der Zeichenmatrix gesetzt, ansonsten wird das Bit gelöscht.

Einsprung-Bedingungen:

A enthält die codierte Ink, mit der verglichen werden soll
H enthält die physikalische Zeichenspalte, aus der gelesen werden soll
L enthält die physikalische Zeichenreihe, aus der gelesen werden soll
DE enthält die Adresse des Bereichs, in dem die Matrix aufgebaut werden soll.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Zeichenposition wird in physikalischen Koordinaten angegeben, wobei Reihe 0, Spalte 0 die linke obere Ecke des Bildschirms bedeutet.

Die angegebene Zeichenposition wird nicht auf Gültigkeit geprüft. Die Angabe einer ungültigen Position (außerhalb des Bildschirms) kann unvorhersehbare Folgen haben.

Die erzeugte Matrix hat normales Format. Sie ist 8 Bytes lang, die Kopfzeile ist zuerst abgespeichert, die Fußzeile zuletzt, die am meisten signifikanten Bits eines Bytes beziehen sich auf die Bildpunkte ganz links in einer Zeile und die am wenigsten signifikanten Bits beziehen sich auf die Bildpunkte ganz rechts in einer Zeile.

Da die Bildpunkte nur daraufhin untersucht werden, ob sie mit einer Ink übereinstimmen, ist die erstellte Matrix keine exakte Wiedergabe von dem, was auf dem Bildschirm dargestellt ist. Es kann notwendig sein, diese Routine mit verschiedenen unterschiedlichen Inks aufzurufen, wenn Sie versuchen, Zeichen vom Bildschirm zu lesen.

Verwandte Einsprünge:

SCR UNPACK
TXT RD CHAR

115: SCR ACCESS #BC59

Setze den Bildschirmdarstellungsmodus für den Graphik-VDU.

Aktion:

Setze den Darstellungsmodus des Graphik-VDU so, daß der Graphik-VDU Bildpunkte plottet durch Schreiben, UND-Verknüpfung, ODER-Verknüpfung oder EXCLUSIVE-ODER-Verknüpfung.

Einsprung-Bedingungen:

A enthält den gewünschten Darstellungsmodus

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Darstellungsmodus wird mit der Maske #03 verknüpft, um ihn gültig zu machen. Folgende Darstellungsmodi sind möglich:

0:	FORCE-Modus:	NEW = INK
1:	XOR-Modus:	NEW = INK exclusive-oder OLD
2:	AND-Modus:	NEW = INK und OLD
3:	OR-Modus:	NEW = INK oder OLD

NEW: so soll der Pixel gesetzt werden
OLD: so ist der Pixel aktuell gesetzt
INK: ist die Tinte, in der geplottet werden soll.

Der Standardmodus ist FORCE (Modus 0) und wird bei EMS gesetzt und wenn SCR RESET aufgerufen wird.

Das Setzen des Darstellungsmodus beeinflußt die Art, wie die Routine des indirekten Verzweigungspunkts SCR WRITE die Bildpunkte ausgibt. Die Ausgaberoutinen des Graphik-VDU rufen diesen indirekten Verzweigungspunkt auf, um Bildpunkte auszugeben und somit beeinflußt der Darstellungsmodus den Graphik-VDU. Dagegen ruft keine Routine des Text-VDU diesen indirekten Verzweigungspunkt auf (er setzt alle Bildpunkte direkt auf den Bildschirm) und somit beeinflußt der Darstellungsmodus den Text-VDU nicht. Routinen, die Bildschirmbereiche löschen (z.B. GRA CLEAR WINDOW) arbeiten ähnlich wie die Text-VDU und werden somit vom Darstellungsmodus auch nicht beeinflußt.

Verwandte Einsprünge:

SCR WRITE

116: SCR PIXELS #BC5C

Setze einen Bildpunkt ohne Berücksichtigung des Darstellungsmodus des Graphik-VDU.

Aktion:

Gib einen Bildpunkt oder – punkte auf dem Bildschirm aus. Die Ausgabeposition wird durch eine Bildschirmadresse und eine Maske des Bildpunktes bestimmt. Der Bildpunkt wird immer auf die angegebene Ink gesetzt, unabhängig davon, welchen Darstellungsmodus die Graphik-VDU benutzt.

Einsprung-Bedingungen:

B enthält die codierte Ink zur Ausgabe
C enthält die Maske des/der Bildpunkte(s)
HL enthält die Bildschirmadresse des/der Bildpunkte(s)

Aussprung-Bedingungen:

AF ist zerstört,
alle anderen Register sind unverändert

Anmerkungen:

Die Bildschirmadresse wird nicht geprüft und deshalb kann eine ungültige Bildschirmadresse zu nicht vorhersehbaren Ergebnissen führen.

Die Bildpunktmaske kann eine kombinierte Maske für mehr als einen Bildpunkt sein (dies erhöht in bestimmten Fällen die Ausgabegeschwindigkeit).

Um einen Bildpunkt unter Beachtung des Darstellungsmodus des Graphik-VDU auszugeben, muß SCR WRITE aufgerufen werden. SCR PIXELS entspricht SCR WRITE, wenn der Standardmodus (FORCE) ausgewählt ist. Die Text-VDU gibt die Bildpunkte in den Zeichen immer im FORCE-Modus aus.

Verwandte Einsprünge:

SCR WRITE

117: SCR HORIZONTAL #BC5F

Zeichne eine vollkommen horizontale Linie.

Aktion:

Zeichne eine Linie auf den Bildschirm, die horizontal verläuft. Die Bildpunkte auf der Linie werden über den indirekten Verzweigpunkt SCR WRITE ausgegeben und benutzen somit den momentanen Darstellungsmodus des Graphik-VDU.

Einsprung-Bedingungen:

A enthält die codierte Ink, in der zu zeichnen ist
DE enthält die Basis-X-Koordinate des Anfangspunkts der Linie
BC enthält die Basis-X-Koordinate des Endpunkts der Linie
HL enthält die Basis-Y-Koordinate der Linie

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Endpunkte der Linie werden in Basiskoordinaten angegeben, d.h. 0,0 ist der Bildpunkt in der linken unteren Ecke des Bildschirms und jede Koordinatenposition bezieht sich auf einen einzelnen Bildpunkt.

Die Endpunkte werden nicht auf ihre Gültigkeit geprüft (innerhalb des Bildschirms). Ungültige Bildpunkte können nicht vorhersehbare Folgen haben.

Die X-Koordinate des Beginns muß kleiner/gleich der X-Koordinate des Endes sein.

Diese Routine kann entsprechend der Methode, die der Graphik-VDU zur Ausgabe einer Linie benutzt, verwendet werden – sie teilt eine Linie, die mehr horizontal als vertikal verläuft in eine Anzahl von Segmenten auf, die ausschließlich horizontal verlaufen und gibt diese separat aus.

Verwandte Einsprünge:

GRA LINE ABSOLUTE
GRA LINE RELATIVE
SCR VERTICAL

118: SCR VERTICAL #BC62

Zeichne eine vollkommen vertikale Linie.

Aktion:

Zeichne eine Linie auf den Bildschirm, die vertikal verläuft. Die Bildpunkte auf der Linie werden über den indirekten Verzweigpunkt SCR WRITE ausgegeben und benutzen somit den momentanen Darstellungsmodus des Graphik-VDU.

Einsprung-Bedingungen:

A enthält die codierte Ink, in der zu zeichnen ist
DE enthält die Basis-X-Koordinate der Linie
HL enthält die Basis-Y-Koordinate des Anfangspunktes der Linie
BC enthält die Basis-X-Koordinate des Endpunktes der Linie

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die Endpunkte der Linie werden in Basiskoordinaten angegeben, d.h. 0,0 ist der Bildpunkt in der linken unteren Ecke des Bildschirms und jede Koordinatenposition bezieht sich auf einen einzelnen Bildpunkt.

Die Endpunkte werden nicht auf ihre Gültigkeit geprüft (innerhalb des Bildschirms). Ungültige Bildpunkte können nicht vorhersehbare Folgen haben.

Die Y-Koordinate des Beginns muß kleiner/gleich der Y-Koordinate des Endes sein.

Diese Routine kann entsprechend der Methode, die der Graphik-VDU zur Ausgabe einer Linie benutzt, verwendet werden – sie teilt eine Linie, die mehr vertikal als horizontal verläuft in eine Anzahl von Segmenten auf, die ausschließlich vertikal verlaufen und gibt diese separat aus.

Verwandte Einsprünge:

GRA LINE ABSOLUTE
GRA LINE RELATIVE
SCR HORIZONTAL

119: CAS INITIALIISE #BC65

Initialisiere die Kassettenverwaltung.

Aktion:

Vollständige Initialisierung der Kassettenverwaltung (wie während EMS).

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Folgende Operationen werden durchgeführt:

Alle Ein-/Ausgabegeräte werden als „geschlossen" markiert.
Die standardmäßige Schreibgeschwindigkeit wird eingestellt.
Die Bestätigungsmeldungen werden angeschaltet.

Verwandte Einsprünge:

CAS IN ABANDON
CAS NOISY
CAS OUT ABANDON
CAS SET SPEED

120: CAS SET SPEED #BC68

Setze die Schreibgeschwindigkeit.

Aktion:

Angabe der Länge der Bitausgabe und der zugeordneten Vorprüflänge.

Einsprung-Bedingungen:

HL enthält die Länge für ein halbes Null-Bit
A enthält die zugeordnete Vorprüflänge

Aussprung-Bedingungen:

AF und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Die angegebene Geschwindigkeit ist die Länge in Mikrosekunden für ein halbes Null-Bit. Ein Eins-Bit wird in der zweifachen Dauer eines Null-Bit geschrieben. Die angegebene Geschwindigkeit kann zur durchschnittlichen Baud-Rate (unter der Annahme, daß Null-Bits und Eins-Bits gleich verteilt sind) durch folgende Gleichung in Beziehung gesetzt werden:

Durchschnittliche Baud-Rate
= 1 000 000 / (3 ★ Halbe Null-Bit-Länge)
333 333 / halbe Null-Bit-Länge

Die halbe Null-Bit-Länge muß zwischen 130 und 480 Mikrosekunden liegen, ansonsten treten Lese- oder Schreibfehler auf.

Die angegebene Vorprüflänge ist eine Extralänge in Mikrosekunden, die unter bestimmten Umständen zu den halben Eins-Bits dazugezählt und von den halben Null-Bits abgezogen werden muß. Die erforderliche Vorprüflänge ändert sich mit der Geschwindigkeit (bei höherer Baud-Rate größer).

Die Vorlauflänge kann zwischen 0 und 255 Mikrosekunden liegen, obwohl die höheren Werte nicht sinnvoll sind, da sie Lese- und Schreibfrehler verursachen können.

Die Standardwerte für die Länge eines halben Null-Bits ist 333 Mikrosekunden (1000 Baud) und 25 Mikrosekunden Vorprüflänge. Die allgemein angewendete höhere Geschwindigkeit ist 167 Mikrosekunden (2000 Baud) mit 50 Mikrosekunden Vorprüflänge. Diese Werte wurden bei ausführlichen Tests ermittelt und dem Anwender wird empfohlen, an ihnen festzuhalten.

Verwandte Einsprünge:

CAS INITIALISE

121: CAS NOISY #BC6B

Zulassen oder Sperren von Bereitschaftsmeldungen.

Aktion:

Das Sperren der Meldungen verhindert die Ausgabe von Bereitschafts- und Informationsmeldungen. Es verhindert nicht die Ausgabe von Fehlermeldungen. Das Zulassen von Meldungen ermöglicht die Ausgabe aller Meldungen.

Einsprung-Bedingungen:

Wenn Meldungen zugelassen werden sollen:

A muß Null enthalten

Wenn Meldungen gesperrt werden sollen:

A muß ungleich Null sein.

Aussprung-Bedingungen:

AF ist zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Folgende Bereitschafts- und Informationsmeldungen werden abgeschaltet:

Press PLAY then any key:
Press REC and PLAY then any key:
Found „FILENAME“ block ‚n‘
Loading „FILENAME“ block ‚n‘
Saving „FILENAME“ block ‚n‘

Folgende Fehlermeldungen werden nicht abgeschaltet:

Read error ‚X‘
Write error a
Rewind tape

Verwandte Einsprünge:

CAS INITIALISE

122: CAS START MOTOR #BC6E

Starte den Kassettenmotor.

Aktion:

Anschalten des Kassettenmotors und warten auf die normale Drehzahl, wenn er vorher abgeschaltet war.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn das Anlassen des Motors fehlerfrei funktionierte

CARRY-Flag „an“

Wenn der Benutzer die ESCAPE-Taste drückte:

CARRY-Flag „aus“

Immer:

A enthält den vorigen Zustand des Motors
alle anderen Register unverändert,
alle anderen Flags zerstört.

Anmerkungen:

Wenn der Motor nicht schon läuft, wartet die Routine ungefähr 2 Sekunden, um volle Bandgeschwindigkeit zu erreichen.

Der Motor wird immer durch die Routine angeschaltet. Wenn der Benutzer die ESCAPE-Taste drückt, wird die Wartezeit für das Erreichen der Motordrehzahl verkürzt.

Der vorherige Zustand des Motors kann an die Routine CAS RESTORE MOTOR weitergereicht werden.

Verwandte Einsprünge:

CAS RESTORE MOTOR
CAS STOP MOTOR

123: CAS STOP MOTOR #BC71

Stoppe den Kassettenmotor.

Aktion:

Abschalten des Kassettenmotors und Rückgabe des vorigen Zustands.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn der Motor richtig abgeschaltet wurde:

CARRY-Flag „an“

Wenn der Benutzer die ESCAPE-Taste drückte:

CARRY-Flag „aus“

Immer:

A enthält den vorigen Zustand des Motors
alle anderen Register unverändert,
alle anderen Flags zerstört.

Anmerkungen:

Der Motor wird immer durch die Routine abgeschaltet. Es gibt keine Verzögerung, die erlaubt, den Motor zu verlangsamen. Der vorige Zustand des Motors kann an CAS RESTORE MOTOR weitergereicht werden.

Verwandte Einsprünge:

CAS RESTORE MOTOR
CAS START MOTOR

124: CAS RESTORE MOTOR #BC74

Setze den Kassettenmotor auf den vorherigen Zustand.

Aktion:

Schalte den Motor wieder ein oder aus. Warte auf die richtige Drehzahl, wenn der Motor eingeschaltet wurde.

Einsprung-Bedingungen:

A enthält den vorherigen Zustand des Motors.

Aussprung-Bedingungen:

Wenn der Motor ein- oder ausgeschaltet wurde:

CARRY-Flag „an“

Wenn der Benutzer die ESCAPE-Taste drückte:

CARRY-Flag „aus“

Immer:

A und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine verwendet den vorherigen Zustand des Motors, wie er von CAS START MOTOR oder CAS STOP MOTOR zurückgegeben wurde.

Wenn der Aufruf dieser Routine dazu führt, daß der Motor eingeschaltet wird, weil er vorher aus war, dann wartet die Routine ungefähr 2 Sekunden, damit das Band volle Geschwindigkeit erreichen kann.

Der Motor wird durch diese Routine immer an- oder abgeschaltet (wie es nötig ist). Wenn der Benutzer die ESCAPE-Taste drückt, dann wird lediglich die Wartezeit auf die Motordrehzahl verkürzt.

Verwandte Einsprünge:

CAS START MOTOR
CAS STOP MOTOR

125: CAS IN OPEN #BC77

Eröffne eine Datei zur Eingabe.

Aktion:

Bereitmachen des Ein-/Ausgabegeräts zum Lesen einer Datei und Lesen des ersten Blocks.

Einsprung-Bedingungen:

B enthält die Länge des Dateinamens
HL enthält die Adresse des Dateinamens
DE enthält die Adresse eines 2 K-Puffers, der verwendet werden soll.

Aussprung-Bedingungen:

Wenn die Datei richtig eröffnet wurde:

CARRY-Flag „an“
ZERO-Flag „aus“
HL enthält die Pufferadresse des Dateietiketts
DE enthält die Datenspeicherung (aus dem Etikett)
BC enthält die logische Dateilänge (aus dem Etikett)
A enthält die Dateiart (aus dem Etikett)

Wenn die Eingabe bereits in Gebrauch ist:

CARRY-Flag „aus“
ZERO-Flag „aus“
A, BC, DE und HL zerstört.

Wenn der Benutzer ESCAPE drückte:

CARRY-Flag „aus“
ZERO-Flag „an“
A, BC, DE und HL zerstört.

Immer:

IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der angegebene 2 K-Puffer (2048 Bytes) wird verwendet, um den Inhalt eines Blocks der Datei abzuspeichern, wenn er vom Band gelesen wird. Er bleibt solänge in Gebrauch, bis die Datei durch den Aufruf entweder von CAS IN CLOSE oder von CAS IN ABANDON geschlossen wird. Der Puffer kann überall im Speicher, auch unterhalb eines ROM, liegen.
Der durchgereichte Dateiname wird in die Dateibeschreibung kopiert. Wenn er länger als 16 Zeichen ist, wird er auf 16 Zeichen gekürzt. Wenn er kürzer als 16 Zeichen ist, wird er mit Nullen (#00) auf 16 Stellen erweitert.
Obwohl der Dateiname alle Zeichen enthalten kann, vermeidet man am besten Nullen. Kleinbuchstaben (ASCII-Zeichen #61...#7A) werden in die zugehörigen Großbuchstaben (#...#5A) umgesetzt. Der Dateiname kann überall im RAM liegen, auch unterhalb eines ROM.

Der Dateiname enthält normalerweise den Namen der zu lesenden Datei. Ein Dateiname mit der Länge 0 (oder der mit einer Null beginnt) jedoch wird speziell behandelt. Es wird in diesem Fall unterstellt, daß die nächste Datei auf dem Band gelesen werden soll.

Wenn die Datei zum Lesen eröffnet wird, wird sofort der erste Block der Datei gelesen. Die Adresse des Bereichs, in dem das Etikett dieses Blocks abgespeichert ist, wird an den Anwender zurückgegeben, so daß daraus Informationen entnommen werden können. Dieser Bereich muß in den zentralen 32 K des RAM liegen. Der Anwender darf nicht in das Etikett schreiben, sondern nur aus ihm lesen. Die Kassettenverwaltung benutzt einige Felder im Etikett für eigene Zwecke und so kann sich dieses davon unterscheiden, wie es vom Band gelesen wurde. Die Dateiart, logische Satzlänge, Einsprungpunkt und alle Anwenderfelder bleiben unverändert (siehe Kapitel 8 für die Beschreibung des Etiketts).

Verwandte Einsprünge:

CAS IN ABANDON
CAS IN CHAR
CAS IN CLOSE
CAS IN DIRECT
CAS OUT OPEN

126: CAS IN CLOSE #BC7A

Schließe die Eingabedatei korrekt.

Aktion:

Markieren des Eingabegerätes als geschlossen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn das Eingabegerät richtig geschlossen wurde:

CARRY-Flag „an“

Wenn das Eingabegerät nicht eröffnet war:

CARRY-Flag „aus“

Immer:

A, BC, DE, HL und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine sollte zum Schließen einer Datei verwendet werden, wenn entweder über CAS IN CHAR oder CAS IN DIRECT von ihr gelesen wurde.
Der Anwender kann nach dem Aufruf dieser Routine über den in CAS IN OPEN angegebenen Puffer wieder verfügen.

Verwandte Einsprünge:

CAS IN ABANDON
CAS IN OPEN
CAS OUT CLOSE

127: CAS IN ABANDON #BC7D

Schließe die Eingabedatei sofort.

Aktion:

Abbrechen des Lesens vom Eingabegerät und Schließen der Eingabe.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine ist für den Fehlerfall oder ähnliche Umstände vorgesehen.
Nach Aufruf dieser Routine kann der Anwender über den in CAS IN OPEN angegebenen Puffer wieder verfügen.

Verwandte Einsprünge:

CAS IN CLOSE
CAS IN OPEN
CAS OUT ABANDON

128: CAS IN CHAR #BC80

Lies ein Zeichen von der Eingabedatei.

Aktion:

Lesen eines Zeichens vom Eingabegerät. Bei Bedarf wird ein Block vom Band geholt.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn das Zeichen richtig gelesen wurde:
CARRY-Flag „an“
ZERO-Flag „aus“
A enthält das Zeichen, das von der Datei gelesen wurde

Wenn Dateiende erkannt wurde:
CARRY-Flag „aus“
ZERO-Flag „aus“
A zerstört

Wenn der Benutzer ESCAPE drückte:
CARRY-Flag „aus“
ZERO-Flag „an“
A zerstört

Immer:
IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Wenn der Benutzer vorher ESCAPE gedrückt hat oder das Eingabegerät nicht wie erwartet eröffnet ist, wird dies als Dateiende erkannt.

Wenn einmal das erste Zeichen von einer Datei gelesen wurde, kann diese nur noch zeichenweise abgearbeitet werden. Es ist nicht möglich, auf direktes Lesen (über CAS IN DIRECT) umzuschalten.

Verwandte Einsprünge:

CAS IN CLOSE
CAS IN DIRECT
CAS IN OPEN
CAS OUT CHAR
CAS RETURN
CAS TEST EOF

129: CAS IN DIRECT #BC83

Lies die Eingabedatei in den Speicher.

Aktion:

Lesen der Eingabedatei in einem Schritt direkt in den Speicher, im Gegensatz zum zeichenweisen Lesen.

Einsprung-Bedingungen:

HL enthält die Adresse (irgendwo im RAM), an der die Datei abgestellt werden soll.

Aussprung-Bedingungen:

Wenn die Datei richtig gelesen wurde:
CARRY-Flag „an“
ZERO-Flag „aus“
A enthält die Eingangsadresse (aus dem Etikett)

Wenn die Datei nicht wie erwartet eröffent war:
CARRY-Flag „aus“
ZERO-Flag „aus“
HL zerstört

Wenn der Benutzer ESCAPE gedrückt hat:
CARRY-Flag „aus“
ZERO-Flag „an“
HL zerstört

Immer:
A, BC, DE, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das Eingabegerät muß neu eröffnet werden (über CAS IN OPEN). Wenn die Eingabe bereits zeichenwese abgearbeitet wurde (durch den Aufruf von CAS IN CHAR) ist es nicht möglich, die Datei direkt zu lesen.
Es ist auch nicht möglich, mehr als einmal direkt von der Datei zu lesen. Dies würde dann die Kopie der gelesenen Datei zerstören. Der Datenpuffer wird bei Eröffnung der Eingabe an die richtige Stelle kopiert und der Rest der Datei (falls vorhanden) wird ebenfalls in den Sepeicher gelesen.

Verwandte Einsprünge:

CAS IN CHAR
CAS IN CLOSE
CAS IN OPEN
CAS OUT DIRECT

130: CAS RETURN #BC86

Gib das zuletzt gelesene Zeichen zurück.

Aktion:

Stelle das zuletzt über CAS IN CHAR gelesene Zeichen zurück in den Eingabepuffer. Dieses Zeichen wird beim nächsten Aufruf von CAS IN CHAR erneut gelesen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

alle Register und Flags unverändert.

Anmerkungen:

Diese Routine kann nur dazu verwendet werden, das zuletzt über CAS IN CHAR gelesene Zeichen zurückzustellen. Mindestens ein Zeichen muß gelesen worden sein seit:

die Eingabe eröffnet wurde
oder das letzte Zeichen zurückgestellt wurde
oder die letzte Untersuchung auf Dateiende durchgeführt wurde

Verwandte Einsprünge:

CAS IN CHAR

131: CAS TEST EOF #BC89

Abfrage auf Dateiende.

Aktion:

Untersuchung, ob das Ende der Eingabedatei erreicht wurde.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn Dateiende noch nicht erreicht wurde:

CARRY-Flag „an“
ZERO-Flag „aus“

Wenn Dateiende erreicht wurde:

CARRY-Flag „aus“
ZERO-Flag „aus“

Wenn der Benutzer ESCAPE drückte:

CARRY-Flag „aus“
ZERO-Flag „an“

Immer:

A, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Aufruf dieser Routine stellt die Eingabe auf zeichenweise Verarbeitung um. Es ist nach Aufruf dieser Routine nicht möglich, das direkte Lesen zu verwenden.

Nach Aufruf dieser Routine ist es nicht möglich, CAS RETURN aufzurufen. Es muß vorher ein Zeichen gelesen werden.

Verwandte Einsprünge:

CAS IN CHAR

132: CAS OUT OPEN #BC8C

Eröffne eine Datei zur Ausgabe.

Aktion:

Bereitmachen des Ein-/Ausgabegerätes zum Schreiben.

Einsprung-Bedingungen:

B enthält die Länge des Dateinamens
HL enthält die Adresse des Dateinamens
DE enthält die Adresse eines 2-K-Puffers, der verwendet werden soll.

Aussprung-Bedingungen:

Wenn das Ausgabegerät bereits in Gebrauch ist:

CARRY-Flag „aus"
HL zerstört

Wenn die Datei richtig eröffnet wurde:

CARRY-Flag „an"
HL enthält die Adresse eines Puffers, der das Etikett enthält, welches bei jedem Dateiblock geschrieben wird.

Immer:
ZERO-Flag „aus"
A, BC, DE, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Wenn eine Datei zeichenweise erstellt wird, wird der angegebene 2-K-Puffer (2048 Bytes) dazu benutzt, den Inhalt eines Blocks der Datei zu speichern, bevor er auf Band geschrieben wird. Er bleibt im Gebrauch, bis die Datei geschlossen wird, entweder über CAS OUT CLOSE oder CAS OUT ABANDON. Der Puffer kann überall im Speicher stehen, auch unterhalb eines ROM.

Wenn Dateien direkt geschrieben werden, wird der angegebene Puffer nicht benutzt, da beim Aufruf von CAS OUT DIRECT kein Puffer notwendig ist.

Der durchgereichte Dateiname wird in die Ausgabebeschreibung kopiert. Wenn er länger als 16 Zeichen ist, wird er auf 16 Stellen gekürzt. Wenn er kürzer als 16 Zeichen ist, wird er mit Nullen (#00) auf 16 Stellen ergänzt. Obwohl der Dateiname jedes Zeichen enthalten kann, sollten Nullen am besten vermieden werden. Kleinbuchstaben (ASCII-Zeichen #61...#7A) werden in die entsprechenden Großbuchstaben (#41...#5A) umgesetzt. Der Dateiname kann überall im RAM, auch unterhalb eines ROM liegen.

Wenn das Ein-/Ausgabegerät zum Schreiben eröffnet wird, wird ein Etikett (Kopfsatz) aufgebaut, das am Anfang jedes Blocks der Datei geschrieben wird. Viele Felder im Etikett werden von der Kassettenverwaltung besetzt, der Rest kann jedoch vom Anwender benutzt werden. Die Adresse dieses Etikettes wird auch an den Anwender durchgereicht, so daß dort Informationen abgespeichert werden können.

Der Anwender kann in die Felder Dateiart, logische Satzlänge, Einsprungpunkt und alle Anwenderfelder schreiben; in andere Felder des Etiketts darf er nicht schreiben. Alle vom Anwender benutzen Felder werden zunächst mit Nullen initialisiert, mit Ausnahme der Dateiart, die auf „ASCII-ungeschützt (Version 1)" gesetzt wird (siehe Abschnitt 8.4 für die Beschreibung des Etiketts).

Verwandte Einsprünge:

CAS IN OPEN
CAS OUT ABANDON
CAS OUT CHAR
CAS OUT CLOSE
CAS OUT DIRECT

133: CAS OUT CLOSE #BC8F

Schließe die Ausgabedatei korrekt.

Aktion:

Markieren des Ausgabegerätes als „geschlossen“ und Schreiben des letzten Datenpuffers auf Band.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn das Ausgabegerät richtig geschlossen wurde:
CARRY-Flag „an“
ZERO-Flag „aus“

Wenn das Ausgabegerät nicht geöffnet war:
CARRY-Flag „aus“
ZERO-Flag „aus“

Wenn der Benutzer ESCAPE drückte:
CARRY-Flag „aus“
ZERO-Flag „an“

Immer:
A, BC, DE, HL, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Nach der Benutzung von CAS OUT CHAR oder CAS OUT DIRECT ist es notwendig, diese Routine aufzurufen, um den letzten Datenblock auf das Band zu schreiben. Wenn der Block eine Länge von 0 Bytes hat (es wurde nichts in die Datei geschrieben) wird auch nichts auf Band ausgegeben.

Wenn das Schreiben abgebrochen werden soll, muß CAS OUT ABANDON aufgerufen werden, um den letzten Datenblock auf Band auszugeben.

Wenn der Benutzer während des Schreibens des letzten Blockes ESCAPE drückt, wird die Datei offen gelassen und nicht geschlossen.

Nach Aufruf dieser Routine kann der Anwender wieder über den Puffer, den er in CAS OUT OPEN angegeben hat, verfügen.

Verwandte Einsprünge:

CAS IN CLOSE
CAS OUT ABANDON
CAS OUT OPEN

134: CAS OUT ABANDON #BC92

Schließe die Ausgabedatei sofort.

Aktion:

Schließe die Ausgabedatei sofort ab und markiere das Ausgabegerät als „geschlossen". Alle nicht geschriebenen Daten werden zerstört und nicht auf Band ausgegeben.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE, und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine ist für die Benutzung nach einem Fehler oder ähnlichen Umstand vorgesehen.

Verwandte Einsprünge:

CAS IN ABANDON
CAS OUT CLOSE
CAS OUT OPEN

135: CAS OUT CHAR #BC95

Schreibe ein Zeichen in die Ausgabedatei.

Aktion:

Füge ein Zeichen in den Puffer des Ausgabegerätes hinzu. Wenn der Puffer bereits voll ist, wird er auf Band ausgegeben, bevor das neue Zeichen eingestellt wird.

Einsprung-Bedingungen:

A enthält das auszugebende Zeichen.

Aussprung-Bedingungen:

Wenn das Ausgabegerät richtig ausgegeben wurde:
CARRY-Flag „an“
ZERO-Flag „aus“

Wenn die Datei nicht wie erwartet eröffnet war:
CARRY-Flag „aus“
ZERO-Flag „aus“

Wenn der Benutzer ESCAPE drückte:
CARRY-Flag „aus“
ZERO-Flag „an“

Immer:
A, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Wenn diese Routine die Bedingung zurückmeldet, daß die Datei nicht wie erwartet eröffnet ist, dann hat entweder der Benutzer vorher ESCAPE gedrückt oder die Datei wurde über CAS OUT DIRECT ausgegeben. In jedem Fall wird das gesendete Zeichen zerstört.

Nach dem Einstellen aller Zeichen in die Datei ist es notwendig, CAS OUT CLOSE aufzurufen, um sicherzustellen, daß der letzte Block der Datei auf Band ausgegeben wird.

Wenn diese Routine einmal aufgerufen worden ist, ist es nicht mehr möglich, auf die direkte Dateiausgabe umzuschalten.

Verwandte Einsprünge:

CAS IN CHAR
CAS OUT CLOSE
CAS OUT DIRECT
CAS OUT OPEN

136: CAS OUT DIRECT #BC98

Schreibe die Ausgabedatei direkt aus dem Speicher.

Aktion:

Schreibe den Speicherinhalt direkt auf die Ausgabedatei.

Einsprung-Bedingungen:

HL enthält die Adresse der auszugebenden Daten
DE enthält die Länge der auszugebenden Daten
BC enthält die Einsprungadresse (für den Kopfsatz)
A enthält die Dateiart (für den Kopfsatz).

Aussprung-Bedingungen:

Wenn die Datei richtig geschrieben wurde:
CARRY-Flag „an"
ZERO-Flag „aus"

Wenn die Datei nicht wie erwartet eröffnet war:
CARRY-Flag „aus"
ZERO-Flag „aus"

Wenn der Benutzer ESCAPE drückte:
CARRY-Flag „aus"
ZERO-Flag „an"

Immer:
A, BC, DE, HL, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Nach der Ausgabe der Datei muß sie mit CAS OUT CLOSE geschlossen werden um sicherzugehen, daß der letzte Block der Datei auf Band geschrieben wird.

Es ist nicht möglich, die Ausgabemethode vom zeichenweisen Ausgeben (über CAS OUT CHAR) in direktes Ausgeben (über CAS OUT DIRECT) und umgekehrt zu ändern, wenn einmal eine Methode ausgewählt wurde. Ebenso ist es nicht möglich, eine Datei in zwei oder mehreren Teilen direkt auszugeben, indem CAS OUT DIRECT mehr als einmal aufgerufen wird – dies zerstört die Daten. Jeder Versuch, diese Regeln zu durchbrechen, ergibt einen Fehler „Datei nicht wie erwartet eröffnet".

Verwandte Einsprünge:

CAS IN DIRECT
CAS OUT OPEN
CAS OUT CLOSE

137: CAS CATALOG #BC9B

Erzeuge einen Katalog des Bandes.

Aktion:

Lies Dateiblöcke, um ihre Gültigkeit zu prüfen und gib Informationen darüber auf dem Bildschirm aus.

Einsprung-Bedingungen:

DE enthält die Adresse eines 2 K-Puffers, der benutzt werden soll.

Aussprung-Bedingungen:

Wenn der Katalog richtig erstellt wurde:

CARRY-Flag „an"

Wenn das Eingabegerät in Gebrauch war:

CARRY-Flag „aus"

Immer:

ZERO-Flag „aus"
A, BC, DE, HL, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine benutzt das Eingabegerät und deshalb muß das Eingabegerät geschlossen sein, wenn diese Routine aufgerufen wird. Das Eingabegerät bleibt beim Verlassen der Routine geschlossen. Das Ausgabegerät wird von dieser Routine nicht berührt.

Die Bereitschaftsmeldungen (siehe CAS NOISY) werden von dieser Routine angeschaltet.

Beim Erstellen des Katalogs liest die Kassettenverwaltung einen Kopfsatz, gibt die darin enthaltene Information aus und liest den Datensatz. Dieser Kreislauf wird wiederholt, bis der Benutzer die ESCAPE-Taste drückt. Die ausgegebene Information sieht folgendermaßen aus:

DATEINAME Block N T OK

Dateiname ist der Name der Datei auf dem Band oder **‚unnamed file'** (Datei ohne Namen), wenn der Dateiname mit Null (#00) beginnt.

N ist die Blocknummer. Block 1 ist normalerweise der erste Block einer Datei.

T ist ein Kennzeichen über die Dateiart. Es wird erzeugt, indem #24 (das Zeichen „$") addiert und mit der Maske #0F verknüpft wird (um die Versionsnummer zu entfernen).

Es gibt folgende Dateiarten:

$	Standard BASIC Programm
%	geschütztes BASIC Programm
★	ASCII-Text-Datei (standardmäßige Dateiart)
&	Datei in Binärform
,	geschützte Datei in Binärform

Andere Dateiarten sind möglich, werden aber durch das BASIC im ROM des Gerätes nicht erstellt. Siehe Abschnitt 8.4 für die Beschreibung des Dateiartenbytes.

Am Ende eines Datensatzes wird OK ausgegeben.
Dies zeigt an, daß die Daten ohne Fehler gelesen wurden und dient auch zur Bestimmung des Endes der Daten auf Band (als Hilfe, ein Überschreiben der Banddatei zu vermeiden).

Verwandte Einsprünge:

CAS NOISY

138: CAS WRITE #BC9E

Schreibe einen Satz auf das Band.

Aktion:

Schreiben eines Satzes auf die Kassette. Diese Routine wird von höherstufigen Routinen (CAS OUT CHAR, CAS OUT DIRECT und CAS OUT CLOSE) verwendet, um den Kopfsatz und die Datensätze auszugeben, die zusammen eine Banddatei darstellen.

Einsprung-Bedingungen:

HL enthält die Adresse der Ausgabedaten
DE enthält die Länge der Ausgabedaten
A enthält das Satzkennzeichen, das am Ende des Vorspanns ausgegeben werden soll.

Aussprung-Bedingungen:

Wenn der Satz richtig geschrieben wurde:

CARRY-Flag „an“
A zerstört

Wenn ein Fehler auftrat oder der Benutzer ESCAPE drückte:

CARRY-Flag „aus“
A enthält den Fehlerschlüssel

Immer:

BC, DE, HL, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Eine angegebene Datenlänge von 0 wird von dieser Routine als 65536 angenommen und der gesamte Speicher wird auf Band ausgegeben (wahrscheinlich nicht sehr sinnvoll).

Die Ausgabedaten können überall im RAM stehen, auch unterhalb eines ROM.

Das Satzkennzeichen wird zur Unterscheidung herangezogen, ob es sich um einen Kopfsatz (#2C) oder um einen Datensatz (#16) handelt. Andere Satzkennzeichen können verwendet werden, erfordern jedoch spezielle Anwendungen, um sie zu lesen.
Der Fehlerschlüssel, der von dieser Routine zurückgegeben wird, bedeutet:

0	Unterbrechung	der Benutzer hat ESCAPE gedrückt
1	Überholung	der Kassettenverwaltung war es nicht möglich, ein Bit schnell genug auszugeben

Da das Lesen und Schreiben eines Bandes strenge Zeitberücksichtigungen erfordert, werden Unterbrechungen während der Bandausgabe gesperrt (möglicherweise über 5 Minuten). Es wäre unschön, wenn der Chip für die Tonerzeugung die ganze Zeit über ein Geräusch hervorrufen würde und deshalb wird die Ton-Verwaltung abgeschlossen (SOUND RESET). Wenn die Bandausgabe beendet ist, werden die Unterbrechungen wieder zugelassen.

Der Kassettenmotor wird von dieser Routine gestartet (falls er nicht schon läuft) und in den vorherigen Zustand versetzt, wenn die Ausgabe beendet ist.

Verwandte Einsprünge:

CAS CHECK
CAS READ

139: CAS READ #BCA1

Lies einen Satz vom Band.

Aktion:

Lesen eines Satzes oder Satzteils von der Kassette. Die Routine wird von höherstufigen Routinen (CAS INUT CHAR, CAS IN DIRECT und CAS CATALOG) verwendet, den Kopfsatz und die Datensätze zu lesen, die zusammen eine Datei darstellen.

Einsprung-Bedingungen:

HL enthält die Adresse, an der die zu lesenden Daten abgestellt werden sollen
DE enthält die Länge der zu lesenden Daten
A enthält das Satzkennzeichen, das am Ende des Vorspanns erwartet wird.

Aussprung-Bedingungen:

Wenn der Satz richtig gelesen wurde:

CARRY-Flag „an"
A zerstört

Wenn ein Fehler auftrat oder ESCAPE gedrückt wurde:

CARRY-Flag „aus"
A enthält den Fehlerschlüssel

Immer:

BC, DE, HL, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Eine angegebene Datenlänge von 0 wird von dieser Routine als 65536 angenommen (dies ist nicht sinnvoll).

Es ist nicht notwendig, einen ganzen Satz vom Band zu lesen. Wenn die angegebene Länge kürzer ist als die aktuelle Satzlänge, wird nur die angegebene Anzahl Bytes gelesen. Der Versuch, mehr Bytes von einem Satz zu lesen als die aktuelle Satzlänge, erzeugt einen Fehler, normalerweise einen Überlauffehler (siehe weiter unten).

Das Satzkennzeichen wird benutzt, um Kennsätze (#2C) von Datensätzen (#16) zu unterscheiden. Andere Satzkennzeichen können auftreten, wenn der Satz so erstellt wurde.

Die Fehlerschlüssel, die von dieser Routine zurückgegeben werden, sind folgende:

0	Unterbrechung	der Benutzer drückte ESCAPE
1	Überlauf	die Kassettenverwaltung brauchte zu lange, um ein Bit zu lesen
2	CRC	ein Lesefehler (**C**assette **R**ead **C**check) trat auf.

Da das Lesen und Schreiben eines Bandes strenge Zeitberücksichtigungen erfordert, werden Unterbrechungen während der Bandausgabe gesperrt (möglicherweise über 5 Minuten). Es wäre unschön, wenn der Chip für die Tonerzeugung die ganze Zeit über ein Geräusch hervorrufen würde und deshalb wird die Ton-Verwaltung abgeschlossen (SOUND RESET). Wenn die Bandausgabe beendet ist, werden die Unterbrechungen wieder zugelassen.

Der Kassettenmotor wird von dieser Routine gestartet (falls er nicht schon läuft) und in den vorherigen Zustand versetzt, wenn die Ausgabe beendet ist.

Verwandte Einsprünge:

CAS CHECK
CAS WRITE

140: CAS CHECK #BCA4

Vergleiche einen Satz auf dem Band mit dem Speicherinhalt.

Aktion:

Überprüfen, ob ein Bandsatz die richtige Version von angegebenen Daten enthält. Diese Routine ist vorgesehen, um nach dem Schreiben von Sätzen zu überprüfen, ob sie auch richtig erstellt wurden.

Einsprung-Bedingungen:

HL enthält die Adresse der zu prüfenden Daten
DE enthält die Länge der zu prüfenden Daten
A enthält das Satzkennzeichen, das am Ende des Vorspanns erwartet wird.

Aussprung-Bedingungen:

Wenn der überprüfte Satz richtig ist:

CARRY-Flag „an"
A zerstört

Wenn ein Fehler auftrat oder ESCAPE gedrückt wurde:

CARRY-Flag „aus"
A enthält den Fehlerschlüssel

Immer:

BC, DE, HL, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Eine angegebene Datenlänge von 0 wird von dieser Routine als 65536 angenommen (was einen Prüffehler erzeugt).

Es ist nicht notwendig, den gesamten Satz auf dem Band zu prüfen. Wenn die angegebene Länge kleiner als die aktuelle Satzlänge ist, wird nur die angegebene Anzahl Bytes überprüft. Der Versuch, mehr Bytes als die aktuelle Satzlänge zu prüfen, ergibt ebenfalls einen Fehler (siehe weiter unten).

Die zu prüfenden Daten können überall im RAM liegen, auch unterhalb des ROM.

Das Satzkennzeichen wird benutzt, um Kennsätze (#2C) von Datensätzen (#16) zu unterscheiden. Andere Satzkennzeichen können auftreten, wenn der Satz so erstellt wurde.

Die Fehlerschlüssel, die von dieser Routine zurückgegeben werden, sind:

0	Unterbrechung	Der Benutzer drückte ESCAPE
1	Überholung	die Kassettenverwaltung brauchte zu lange, ein Bit zu lesen
2	CRC	Lesefehler (**C**assette **R**ead **C**heck)
3	Unterschied	Die vom Band gelesenen Daten stimmen nicht mit denen im Speicher überein

Da das Lesen und Schreiben eines Bandes strenge Zeitberücksichtigung erfordert, werden Unterbrechungen während der Bandausgabe gesperrt (möglicherweise über 5 Minuten).

Es wäre unschön, wenn der Chip für die Tonerzeugung die ganze Zeit über ein Geräusch hervorrufen würde und deshalb wird die Ton-Verwaltung abgeschlossen (SOUND RESET). Wenn die Bandeingabe beendet ist, werden die Unterbrechungen wieder zugelassen.

Der Kassettenmotor wird von dieser Routine gestartet (falls er nicht schon läuft) und in den vorherigen Zustand versetzt, wenn die Eingabe beendet ist.

Verwandte Einsprünge:

CAS READ
CAS WRITE

141: SOUND RESET #BCA7

Setze die Tonverwaltung zurück.

Aktion:

Neues Initialisieren der Tonverwaltung.
Setze den Ton-Chip neu auf und lösche alle Warte-Schlangen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Tonwarteschlangen (Sound queues) werden gelöscht,
jeglicher laufende Ton wird angehalten,
der Chip des Tongenerators schweigt.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

SOUND HOLD

142: SOUND QUEUE #BCAA

Stelle einen Ton in die Warteschlange ein.

Aktion:

Versuch, einen Ton in die Warteschlange eines oder mehrerer Kanäle zu stellen. Wenn die Tonwarteschlange eines Kanals voll ist, dann wird an keinen Kanal ein Ton ausgegeben.

Einsprung-Bedingungen:

HL enthält die Adresse eines Tonprogramms.

Aussprung-Bedingungen:

Wenn der Ton in die Warteschlange(n) eingestellt wurde:

CARRY-Flag „ein“
HL zerstört

Wenn mindestens eine Warteschlange voll war:

CARRY-Flag „aus“
HL unverändert

Immer:

A, BC, DE, IX und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das Tonprogramm ist wie folgt ausgelegt:

Byte 0	Kanalangabe und Rendezvouserfordernisse
Byte 1	Amplituden-Hüllkurve
Byte 2	Ton-Hüllkurve
Byte 3..4	Ton-Periode
Byte 5	Geräuschperiode
Byte 6	Anfangsamplitude
Byte 7..8	Dauer oder Hüllkurven-Wiederholungszähler

Alle Werte des Tonprogramms werden mit einer Maske verknüpft, um einen gültigen Bereich zu gewährleisten, bevor sie verwendet werden.

Die Kanäle, auf die der Ton ausgegeben werden kann, sind in Byte 0 folgendermaßen verschlüsselt:

Bit 0	gib an Kanal A aus
Bit 1	gib an Kanal B aus
Bit 2	gib an Kanal C aus
Bit 3	warte auf Kanal A (Rendezvous)
Bit 4	warte auf Kanal B (Rendezvous)
Bit 5	warte auf Kanal C (Rendezvous)
Bit 6	Halte bis zur Freigabe
Bit 7	Lösche die Warteschlange.

Ein Kanal ignoriert die Anweisung, auf sich selbst zu warten. Töne, die an mehrere Kanäle ausgegeben wurden, warten implizit gegenseitig aufeinander. Töne, die die Anweisung haben, zu warten, werden gleichzeitig an den Tongenerator übergeben.

Das Setzen des Haltebits läßt den Ton solange laufen, bis er durch den Aufruf von SOUND RELEASE (oder einer Routine mit ähnlicher Wirkung) freigegeben wird. Das Setzen des Löschbits bewirkt, daß die Warteschlange geleert und jeder laufende Ton abgebrochen wird, um sofort einen neuen Ton starten zu können.

Die Amplituden-Hüllkurve liegt im Bereich von 0 bis 15. Die Hüllkurven 1 bis 15 können über SOUND AMPL ENVELOPE gesetzt werden. Die Hüllkurve 0 bedeutet, daß keine Amplituden-Hüllkurve gewünscht ist, sondern die Anfangsamplitude für 2 Sekunden oder der angegebenen Dauer gehalten werden soll.

Die Ton-Hüllkurve liegt im Bereich von 0 bis 15. Die Hüllkurven 1 – 15 können über SOUND TONE ENVELOPE gesetzt werden. Hüllkurve 0 bedeutet, daß keine Ton-Hüllkurve gewünscht ist, sondern lediglich der Anfangston gehalten werden soll.

Eine Ton-Periode von 0 bedeutet, daß kein Ton erzeugt werden soll. Die Ton-Perioden liegen im Bereich von 1 bis 4095 und werden in Einheiten von 8 Mikrosekunden angegeben.

Die Geräusch-Periode liegt im Bereich von 0 bis 31. Die Geräusch-Perioden 1 – 31 bestimmen den Geräuschanteil eines Tones. Eine Geräusch-Periode von 0 bedeutet, daß kein Geräusch gewünscht ist.

Die Anfangsamplitude liegt im Bereich von 0 bis 15. Ampitude 0 bedeutet kein Anfangston, Amplitude 15 bedeutet größte Lautstärke.

In den Bytes 7 und 8 ist die Tondauer gespeichert. Wenn diese Null ist, wird die Amplituden-Hüllkurve einmal beachtet. Wenn die Tondauer negativ ist, wird die Amplituden-Hüllkurve wiederholt und zwar so oft, wie im negativen Wert der Tondauer angegeben (1 bis 32768 mal).
Wenn die Tondauer positiv aber ungleich Null ist, wird der Wert als Dauer in Einheiten von 1/100 Sekunden angenommen.

Wenn während der Benutzung einer Amplituden-Hüllkurve eine Dauer angegeben wird, wird die Tonlänge auf die angegebene Dauer gesetzt. Wenn die Dauer kürzer als die Hüllkurve ist, wird die Hüllkurve verkürzt; wenn die Dauer länger als die Hüllkurve

ist, dann wird die letzte Amplitude gehalten, bis die Dauer abgelaufen ist. Ton-Hüllkurven werden in derselben Weise behandelt, außer daß sie nie die Tonlänge bestimmen. Das „Ton-Ereignis", das gesetzt wird, wenn die Tonwarteschlange einen freien Eintrag hat, wird für die in diesem Kommando angegebenen Kanäle gesperrt.
Alle Töne, die momentan durch SOUND HOLD angehalten werden, werden automatisch freigegeben, wenn diese Routine aufgerufen wird. Ebenso wird das Warteschlangen-„Ereignis" gesperrt (siehe SOUND ARM EVENT)

SOUND QUEUE läßt Unterbrechungen zu.

Verwandte Einsprünge:

SOUND ARM EVENT
SOUND CHECK
SOUND RELEASE

143: SOUND CHECK #BCAD

Frage, ob in einer Tonwarteschlange Platz frei ist.

Aktion:

Abfrage auf den Status eines Tonkanals. Der Status beinhaltet die Anzahl freier Bereiche in der Tonwarteschlange und sagt, ob ein Kanal gehalten wird.

Einsprung-Bedingungen:

A enthält das Bit für den zu testenden Kanal

Aussprung-Bedingungen:

A enthält den Kanalstatus
BC, DE, HL und die Flags sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Kanal, nach dessen Status gefragt wird, wird wie folgt verschlüsselt:

Bit 0:	Abfrage bezüglich Kanal A
Bit 1:	Abfrage bezüglich Kanal B
Bit 2:	Abfrage bezüglich Kanal C

Wenn mehr als ein Bit gesetzt ist, wird nur der Status eines Kanals zurückgegeben, und zwar in der oben beschriebenen Reihenfolge.

Der zurückgegebene Status ist folgendermaßen verschlüsselt:

Bit 0..2:	enthalten die Anzahl freier Einträge in der Tonwarteschlange für diesen Kanal
Bit 3:	Der Kanal wartet auf ein Rendezvous mit Kanal A
Bit 4:	Der Kanal wartet auf ein Rendezvous mit Kanal B
Bit 5:	Der Kanal wartet auf ein Rendezvous mit Kanal C
Bit 6:	Der Kanal wird angehalten
Bit 7:	Der Kanal ist aktiv (gibt einen Ton aus)

Der Aufruf dieser Routine sperrt das Warteschlangen-„Ereignis“, das gesetzt wird, wenn die Warteschlange für den zurückgegebenen Kanal einen freien Eintrag hat (siehe SOUND ARM EVENT).

Diese Routine läßt Unterbrechungen zu.

Verwandte Einsprünge:

SOUND ARM EVENT
SOUND QUEUE

144: SOUND ARM EVENT #BCB0

Setze ein „Ereignis“, wenn die Tonwarteschlange leer wird.

Aktion:

Zulassen, daß ein TON-„ereignis“ durchgeführt wird, wenn in der Tonwarteschlange eines Kanals ein freier Eintrag entsteht.

Einsprung-Bedingungen:

A enthält das Bit für den freizugebenden Kanal
HL enthält die Adresse des „Ereignis“-Blockes.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der freizugebende Kanal wird folgendermaßen verschlüsselt:

Bit 0: Kanal A
Bit 1: Kanal B
Bit 2: Kanal C

Wenn mehr als ein Bit gesetzt ist, wird nur ein Kanal bearbeitet und zwar in der oben genannten Reihenfolge.

Der angegebene „Ereignis“-Block muß initialisiert sein (über KL INIT EVENT).

Das „Ereignis“ wird angestoßen, wenn in der Warteschlange ein freier Eintrag auftritt. Wenn sich bei Aufruf dieser Routine in der Warteschlange bereits ein freier Eintrag befindet, erfolgt der Anstoß unverzüglich.

Das Ton-„Ereignis“ wird automatisch gesperrt, wenn SOUND QUEUE oder SOUND CHECK aufgerufen werden.
Es wird auch gesperrt, wenn das „Ereignis“ abläuft. Deshalb muß die „Ereignis“-Routine das TON-„Ereignis“ wieder aufsetzen, um es durchgehend am Laufen zu halten.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL INIT EVENT
SOUND CHECK
SOUND QUEUE

145: SOUND RELEASE #BCB3

Lasse zu, daß angehaltene Töne gestartet werden.

Aktion:

Freigabe angehaltener Töne auf einer Anzahl von Kanälen. Dadurch können Töne, die beim Senden über SOUND QUEUE mit dem Anhaltebit markiert wurden, gestartet werden (unter Beachtung anderer Faktoren).

Einsprung-Bedingungen:

A enthält die Bits für die freizugebenden Kanäle.

Aussprung-Bedingungen:

AF, BC, DE, HL und IX sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die freizugebenden Kanäle werden folgendermaßen verschlüsselt:

Bit 0:	Freigabe Kanal A
Bit 1:	Freigabe Kanal B
Bit 2:	Freigabe Kanal C.

Alle angegebenen Kanäle werden freigegeben.

Alle Töne, die momentan durch SOUND HOLD angehalten werden, werden automatisch freigegeben.

Diese Routine kann unterbrochen werden

Verwandte Einsprünge:

SOUND QUEUE

146: SOUND HOLD #BCB6

Halte alle Töne sofort an.

Aktion:

Stoppe sofort alle Töne. Die Töne können durch den Aufruf von SOUND CONTINUE wieder angestoßen werden.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn ein Ton aktiv war:

CARRY-Flag „an“

Wenn kein Ton aktiv war:

CARRY-Flag „aus“

Immer:

A, BC, HL und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Töne, die durch diese Routine angehalten werden, werden automatisch wieder angestoßen, wenn SOUND QUEUE oder SOUND RELEASE aufgerufen werden, genauso wie wenn SOUND CONTINUE selbst aufgerufen wird.

Ein Ton wird durch das Anhalten der Ausführung der Hüllkurven und durch das Setzen der Lautstärke für alle Kanäle auf 0 im Ton-Chip gestoppt.

Wenn der Ton wieder angestoßen wird, wird er so nahe wie möglich bei seinem Haltepunkt wieder aufsetzen.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

SOUND CONTINUE
SOUND RESET

147: SOUND CONTINUE #BCB9

Starte alle gestoppten Töne erneut.

Aktion:

Töne, die durch den Aufruf von SOUND HOLD angehalten wurden, sollen fortgesetzt werden.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und IX sind zerstört,
alle anderen Register unverändert.

Anmerkungen:

Wenn keine Töne angehalten wurden, wird keine Aktion durchgeführt.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

SOUND HOLD
SOUND RELEASE

148: SOUND AMPL ENVELOPE #BCBC

Setze eine Amplituden-Hüllkurve.

Aktion:

Angabe einer von 15 anwenderprogrammierbaren Amplituden-(Lautstärke)-Hüllkurven.

Einsprung-Bedingungen:

A enthält eine Hüllkurvennummer
HL enthält die Adresse eines Amplituden-Datenblocks

Aussprung-Bedingungen:

Wenn die Hüllkurve richtig gesetzt wurde:

CARRY-Flag „an“
HL enthält die Adresse des Datenblocks + 16
A und BC zerstört

Wenn die Hüllkurvennummer ungültig ist:

CARRY-Flag „aus“
A, B und HL unverändert

Immer:

DE und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Hüllkurve wird durch eine Nummer im Bereich von 1 – 15 benannt. Es wird keine Hüllkurve aufgebaut, wenn die Nummer außerhalb dieses Bereiches liegt.

Der Amplituden-Datenblock wird in die Hüllkurve kopiert. Der Datenblock kann im ROM oder im RAM liegen. Er darf jedoch nicht im RAM unterhalb des ROM's liegen.

Der Amplituden-Datenblock hat folgendes Format:

Byte 0:	Anzahl der Hüllkurvenabschnitte
Byte 1 – 3:	1. Abschnitt der Hüllkurve
Byte 4 – 6:	2. Abschnitt der Hüllkurve
Byte 7 – 9:	3. Abschnitt der Hüllkurve
Byte 10 – 12:	4. Abschnitt der Hüllkurve
Byte 13 – 15:	5. Abschnitt der Hüllkurve

Das erste Byte des Amplituden-Datenblocks gibt die Anzahl der verschiedenen Abschnitte in der Hüllkurve an. Nicht benutzte Abschnitte brauchen nicht angegeben zu werden. Eine Hüllkurve ohne Abschnitte hat eine spezielle Bedeutung:

Halte eine gleichförmige Lautstärke für 2 Sekunden.

Die Anzahl der Abschnitte wird nicht geprüft. Wenn die Anzahl außerhalb des Bereiches 0 – 5 liegt, kann das unvorhersehbare Folgen haben und sollte deshalb vermieden werden.

Jeder Abschnitt des Amplituden-Datenblockes kann eine Hardware- oder Software-Hüllkurve angeben. dies bestimmt das erste Byte des Abschnitts.

Ein Abschnitt für eine Software-Hüllkurve sieht folgendermaßen aus:

Byte 0:	Schrittzahl
Byte 1:	Schrittgröße
Byte 2:	Pausenlänge

Der Umstand, daß es sich um einen Abschnitt einer Software-Hüllkurve und nicht einer Hardware-Hüllkurve handelt, ist dadurch gekennzeichnet, daß das Bit 7 im Byte 0 nicht gesetzt ist.

Wenn die Schrittzahl im Bereich von 1 bis 127 liegt, wird die Schrittgröße sooft zur Lautstärke addiert, mit einer Pause zwischen den einzelnen Additionsschritten, wie in Pausenlänge angegeben, in Einheiten von 1/100 Sekunden.

Wenn die Schrittzahl 0 ist, wird die Schrittgröße als absolute Lautstärkenangabe angenommen. Es wird eine einzige Pause von 1/100 Sekunden gemacht.

Nach der Errechnung der neuen Lautstärke wird diese mit der Maske 0F verknüpft, um sicherzugehen, daß sie gültig ist. Jedoch werden alle Berechnungen an der Lautstärke Modulo 16 durchgeführt.

Eine Pausenlänge von 0 bedeutet 256 mal 1/100 Sekunden.

Ein Abschnitt für eine Hardware-Hüllkurve sieht folgendermaßen aus:

Byte 0:	Hüllkurvenform
Byte 1:	Hüllkurven-Schwingungsdauer

Der Umstand, daß es sich um einen Abschnitt einer Hardware-Hüllkurve und nicht einer Software-Hüllkurve handelt, ist dadurch gekennzeichnet, daß Bit 7 im Byte 0 gesetzt ist.

Die Hüllkurvenform (verknüpft mit Maske #7F) wird an das Register 13 des Tongenerators gesendet. Dies gibt die Form der Hardware-Hüllkurve an und sagt, ob sie sich wiederholt (für Einzelheiten siehe Anhang IX).

Die Hüllkurven-Schwingungsdauer wird an die Register 11 und 12 des Tongenerators gesendet. Diese geben die Dauer der Hardware-Hüllkurve an (für Einzelheiten siehe Anhang IX).

Der Abschnitt nach einem Hardwareabschnitt sollte eine Pause sein, die lang genug ist, die Hardwarehüllkurve auszuführen. Eine Pause kann durch die Anwendung einer Software-Hüllkurve mit der Schrittgröße 0 und mit einer Wiederholungszahl und Pausenlänge entsprechend der richtigen Gesamtdauer erreicht werden.

Es gibt keine Schutz gegen das Verändern der Hüllkurve, während sie abläuft. Dies kann unvorhersehbare Folgen haben und sollte vermieden werden.

Die Tondauer kann bestimmt werden durch die angegebene Dauer, wenn der Ton in die Warteschlange gestellt wird oder durch das Beenden der Hüllkurve (siehe SOUND QUEUE). Wenn die angegebene Dauer kleiner ist als die Hüllkurve, wird die Hüllkurve gekürzt. Wenn die Dauer größer ist als die Hüllkurve, wird die letzte Lautstärke gehalten, bis die Dauer erreicht ist.

Verwandte Einsprünge:

SOUND A ADDRESS
SOUND TONE ENVELOPE

149: SOUND TONE ENVELOPE #BCBF

Setze eine Ton-Hüllkurve.

Aktion:

Angabe einer von 15 anwenderprogrammierbaren Ton-Hüllkurven.

Einsprung-Bedingungen:

A enthält eine Hüllkurvennummer
HL enthält die Adresse eines Ton-Datenblocks

Aussprung-Bedingungen:

Wenn die Hüllkurve richtig gesetzt wurde:

CARRY-Flag „an"
HL enthält die Adresse des Datenblocks + 16
A und BC zerstört

Wenn die Hüllkurvennummer ungültig ist:

CARRY-Flag „aus"
A, B und HL unverändert

Immer:

DE und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die zu setzende Hüllkurve wird durch eine Nummer benannt, die im Bereich von 1 – 15 liegen muß, ansonsten wird keine Hüllkurve aufgebaut.

Der Ton-Datenblock wird in die Ton-Hüllkurve kopiert. Der Datenblock kann im ROM oder im RAM liegen. Er darf jedoch nicht im RAM unterhalb des ROM's liegen.

Der Ton-Datenblock hat folgendes Format:

Byte 0:	Zahl der Abschnitte in der Hüllkurve
Byte 1 – 3:	erster Abschnitt der Hüllkurve
Byte 4 – 6:	zweiter Abschnitt der Hüllkurve
Byte 7 – 9:	dritter Abschnitt der Hüllkurve
Byte 10 – 12:	vierter Abschnitt der Hüllkurve
Byte 13 – 15:	fünfter Abschnitt der Hüllkurve

Das erste Byte des Ton-Datenblocks (verknüpft mit der Maske #7F) gibt die Anzahl der Abschnitte der Hüllkurve an. Nicht benutzte Abschnitte müssen nicht angegeben werden. Eine Hüllkurve ohne Abschnitte verändert nicht den Ton (d.h. keine Hüllkurve). Die Zahl der verwendeten Abschnitte wird nicht geprüft.

Wenn die angegebene Zahl außerhalb des Bereiches 0 bis 5 liegt, ist das Ergebnis unvorhersehbar und sollte deshalb vermieden werden.

Das höchste Bit, Bit 7, der Anzahl wird zur Bestimmung herangezogen, ob es sich um eine Wiederholungs-Hüllkurve handelt. Wenn dieses Bit gesetzt ist, dann wird nach Beendigung des letzten Abschnittes wieder der erste Abschnitt verwendet.

Jeder Abschnitt des Ton-Datenblocks hat folgendes Format:

Byte 0:	Schrittzahl
Byte 1:	Schrittgröße
Byte 2:	Pausenlänge

Wenn die Schrittzahl im Bereich #00 bis #EF liegt, handelt es sich bei diesem Abschnitt um einen relativen Abschnitt. Das Schrittgrößenzeichen wird erweitert (Bit 7 wird in die bits 8 bis 15 kopiert) und sooft zu der laufenden Tonhöhe (Periode) addiert, wie Schrittzahl angibt. Nach jeder Addition erfolgt eine Pause in der Länge, wie in Pausenlänge in der Einheit von 1/100 Sekunden angegeben wurde. Der Ton-Chip benutzt nur die 12 unteren Bits der Tonhöhe (Periode) und somit werden alle Berechnungen Modulo #1000 durchgeführt. Eine Schrittzahl von 0 wird als 1 Schritt interpretiert, während bei einer Pausenlänge von 0 256 mal 1/100 Sekunden angenommen wird.

Wenn die Schrittzahl im Bereich #F0 bis #FF liegt, handelt es sich um einen absoluten Abschnitt. Die vier am wenigsten signifikanten Bits der Schrittzahl werden als die am meisten signifikanten der Tonhöhe (Periode) herangezogen und die Schrittgröße wird als das am wenigsten signifikante Byte verwendet. Diese Tonhöhe (Periode) wird unverzüglich gesetzt und von einer Pause gefolgt, deren Länge in Pausenlänge in der Einheit von 1/100 Sekunden angegeben wurde.

Es gibt keinen Schutz gegen die Veränderung einer Hüllkurve während sie benutzt wird. Dies kann unvorhersehbare Folgen haben und sollte vermieden werden.

Wenn die Ton-Hüllkurve vor dem Ende des Tones (wie beim Einstellen in die Warteschlange angegeben) beendet wird, wird der letzte Ton gehalten, d.h. die Ton-Hüllkurve beeinflußt nicht die Tonlänge.

Verwandte Einsprünge:

SOUND AMPL ENVELOPE
SOUND T ADDRESS

150: SOUND A ADDRESS #BCC2

Hole die Adresse einer Amplituden-Hüllkurve.

Aktion:

Abfrage nach dem Datenbereich, in dem eine Amplituden-Hüllkurve abgespeichert ist.

Einsprung-Bedingungen:

A enthält eine Hüllkurvennummer

Aussprung-Bedingungen:

Wenn die Hüllkurve richtig gefunden wurde:

CARRY-Flag „an“
HL enthält die Adresse der Amplituden-Hüllkurve
BC enthält die Länge einer Hüllkurve (16 Bytes)

Wenn die Hüllkurvennummer ungültig war:

CARRY-Flag „aus“
HL zerstört
BC unverändert

Immer:

A und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Hüllkurvennummer muß im Bereich von 1 bis 15 liegen.

Das Format der Amplituden-Hüllkurve ist in SOUND AMPL ENVELOPE beschrieben.

Verwandte Einsprünge:

SOUND AMPL ENVELOPE
SOUND T ADDRESS

151: SOUND T ADDRESS #BCC5

Hole die Adresse einer Ton-Hüllkurve.

Aktion:

Abfrage nach dem Datenbereich, in dem eine Ton-Hüllkurve abgespeichert ist.

Einsprung-Bedingungen:

A enthält eine Hüllkurvennummer

Aussprung-Bedingungen:

Wenn die Hüllkurvennummer richtig gefunden wurde:

CARRY-Flag „an“
HL enthält die Adresse der Ton-Hüllkurve
BC enthält die Länge einer Hüllkurve (16 Bytes)

Wenn die Hüllkurvennummer ungültig war:

CARRY-Flag „aus“
HL zerstört
BC unverändert

Immer:

A und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Hüllkurvennummer muß im Bereich von 1 bis 15 liegen.

Das Format der Ton-Hüllkurve ist in SOUND AMPL ENVELOPE beschrieben.

Verwandte Einsprünge:

SOUND A ADDRESS
SOUND TONE ENVELOPE

152: KL CHOKE OFF #BCC8

Setze den Betriebssystemkern zurück, lösche alle „Ereignis"-Warteschlangen, usw.

Aktion:

Dieser Eingang löscht alle „Ereignis"-Warteschlangen, die verschiedenen Timer (Zeitspannen für bestimmte Ereignisse) und die Einträge für den Bildaufbau und vieles mehr. Der Sinn ist das Loslösen von allen noch anstehenden „Ereignissen" und das Stoppen aller Timer-abhängigen Funktionen, außer Tonerzeugung und Tastaturüberwachung.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

B enthält die ROM-Auswahladresse des aktuellen Vordergrund-ROM's (wenn vorhanden)
DE enthält die Adresse, an der das aktuelle Vordergrund-ROM eingesetzt wurde
C enthält die ROM-Auswahladresse für ein RAM-Vordergrundprogramm
AF und HL sind zerstört,
alle anderen Register sind unverändert.

Anmerkungen:

Wenn das laufende Vordergrundprogramm im RAM ist, enthalten sowohl die zurückgegebene ROM-Auswahladresse als auch der zurückgegebene Einsprungpunkt NULL, d.h. das Standard ROM (ROM 0) als seine Einsprungadresse.

KL CHOKE OFF führt Teile von Abschlußroutinen durch, die notwendig sind, bevor ein neues RAM-Vordergrundprogramm geladen wird, das für MC BOOT PROGRAM erforderlich ist.

Die Abschlußroutinen müssen sicherstellen, daß keine Unterbrechungen oder andere Ereignisse aktiv sind und Speicherplatz benutzen, der durch das Laden eines neuen Programms in den Speicher zerstört werden könnte. Für einen vollständigen Abschluß führt MC BOOT PROGRAM folgendes durch:

SOUND RESET	um die Tonerzeugung abzubrechen
ein OUT auf den I/O-Port #F8FF	um jede externe Unterbrechungsquelle zurückzusetzen
KL CHOKE OFF	um Ereignisse usw. zurückzusetzen
KM RESET	um alle indirekten Verzweigungspunkte bezüglich der Tastatur zurückzusetzen
TXT RESET	um alle indirekten Verzweigungspunkte bezüglich des Text-VDU zurückzusetzen
SCR RESET	um alle indirekten Verzweigungspunkte bezüglich des Bildschirms zurückzusetzen

Die von KL CHOKE OFF zurückgegebenen Werte werden von MC BOOT PROGRAM benutzt, wenn das Laden des Programms fehlerhaft abläuft.

Diese Information wurde im Interesse des Anwenders eingefügt. MC BOOT PROGRAM wird empfohlen, wenn ein RAM Vordergrundprogramm zu laden und zu starten ist. MC START PROGRAM wird empfohlen, um ein ROM Vordergrundprogramm oder ein bereits geladenes RAM-Vordergrundprogramm zu starten.

KL CHOKE OFF kann nicht unterbrochen werden.

Verwandte Einsprünge:

MC BOOT PROGRAM
MC START PROGRAM

153: KL ROM WALK #BCCB

Suche und initialisiere alle Hintergrund-ROM's.

Aktion:

Hintergrund ROM's dienen zur Unterstützung von Hardware-Erweiterungen oder erweitern die Software-Möglichkeiten für das Gerät. Wenn die von den Hintergrund-ROM's vorgesehenen Möglichkeiten verfügbar sein sollen, muß das Vordergrundprogramm sie initialisieren. Diese Routine findet und initialisiert alle Hintergrund ROM's.

Einsprung-Bedingungen:

DE enthält die Adresse des ersten benutzbaren Bytes des neuen Speichers (niedrigste Adresse)
HL enthält die Adresse des letzten benutzbaren Bytes des neuen Speichers (höchste Adresse)

Aussprung-Bedingungen:

AF und BC zerstört,
alle anderen Register unverändert.

Anmerkungen:

Wenn ein Vordergrundprogramm gestartet wird, werden die Adressen des ersten und letzten Bytes des Speichers, den es benutzen darf, durchgereicht. Der Speicherbereich außerhalb dieser Angaben wird benutzt, um Firmware-Variablen, den Stapel, die Verzweigblöcke und den Bildschirmspeicher anzulegen. Vom Speicherbereich für Vordergrundprogramme müssen Bereiche für Hintergrundprogramme angelegt werden.

Das Vordergrundprogramm muß frühzeitig Hintergrund ROM's initialisieren, bevor es den übergebenden Speicher benutzt. Es hat die Wahl, Hintergrund ROM's zuzulassen oder nicht. KL INIT BACK kann verwendet werden, um ein bestimmtes Hintergrund ROM zu initialisieren oder KL ROM WALK wird verwendet, um alle verfügbaren Hintergrund ROM's zu initialisieren.

KL ROM WALK untersucht die ROM's an den ROM-Auswahladressen im Bereich von 1 bis 7. Der Einsprungpunkt für die Einschaltinitialisierung für jedes gefundene ROM wird aufgerufen. Dieser Einsprung kann einigen Speicher für Hintergrund – ROM's anlegen, indem die Werte in DE und HL vor der Rückgabe verändert werden.

Wenn die ROM's einmal initialisiert wurden, fügt der Kern sie in die Liste zur externen Kommandounterstützung hinzu und vermerkt die Basis des Bereichs, den das ROM für sich selbst angelegt hat (wenn überhaupt), im obersten Teil des Speichers. Nachfolgende Aufrufe von FAR CALL auf Eingänge in das ROM setzen automatisch die IY-Indexregister so, daß sie auf den oberen Bereich des Speicherbereichs zeigen.

Siehe Abschnitt 9.4 für eine vollständige Beschreibung von Hintergrund-ROM's.

Verwandte Einsprünge:

KL FIND COMMAND
KL INIT BACK
KL LOG EXT

154: KL INIT BACK #BCCE

Initialisiere ein bestimmtes Hintergrund-ROM.

Aktion:

Hintergrund ROM's dienen zur Unterstützung von Hardware-Erweiterungen oder erweitern die Software-Möglichkeiten für das Gerät. Wenn die von den Hintergrund-ROM's vorgesehenen Möglichkeiten verfügbar sein sollen, muß das Vordergrundprogramm sie initialisieren. Diese Routine wählt ein bestimmtes Hintergrund ROM aus und initialisiert dieses.

Einsprung-Bedingungen:

C enthält die ROM-Auswahladresse des zu initialisierenden ROM's
DE enthält die Adresse des ersten benutzbaren Bytes des neuen Speichers (niedrigste Adresse)
HL enthält die Adresse des letzten benutzbaren Bytes des Speichers (höchste Adresse)

Aussprung-Bedingungen:

DE enthält die Adresse des ersten benutzbaren Bytes des neuen Speichers (niedrigste Adresse)
HL enthält die Adresse des letzten benutzbaren Bytes des neuen Speichers (höchste Adresse)
AF und BC zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die ROM-Auswahladresse muß im Bereich von 1 bis 7 liegen und das ROM an dieser Adresse muß ein Hintergrund ROM sein, ansonsten wird die Anforderung ignoriert.

Wenn ein Vordergrundprogramm gestartet wird, werden die Adressen des ersten und letzten Bytes des Speichers, den er benutzen darf, durchgereicht. Der Speicherbereich außerhalb dieser Angaben wird benutzt, um Firmware-Variablen, den Stapel, die Verzweigblöcke und den Bildschirmspeicher anzulegen. Vom Speicherbereich für Vordergrundprogramme müssen Bereiche für Hintergrundprogramme angelegt werden.

Das Vordergrundprogramm muß frühzeitig Hintergrund ROM's initialisieren, bevor es den übergebenden Speicher benutzt. Es hat die Wahl, Hintergrund ROM's zuzulassen oder nicht. KL INIT BACK kann verwendet werden, um ein bestimmtes Hintergrund ROM zu initialisieren oder KL ROM WALK wird verwendet, um alle verfügbaren Hintergrund ROM's zu initialisieren.

Diese Routine verursacht einen Aufruf des Einsprungpunktes in die Routine zur Einschalt-Initialisierung der Hintergrund ROM's.

Der Einsprungpunkt für die Einschaltinitialisierung für jedes gefundene ROM wird aufgerufen. Dieser Einsprung kann einigen Speicher für Hintergrund-ROM's anlegen, indem die Werte in DE und HL vor der Rückgabe verändert werden. Wenn die ROM's einmal initialisiert wurden, fügt der Kern sie in die Liste zur externen Kommandounterstützung hinzu und vermerkt die Basis des Bereichs, den das ROM für sich selbst angelegt hat (wenn überhaupt), im obersten Teil des Speichers. Nachfolgende Aufrufe von FAR CALL auf Eingänge in das ROM setzen automatisch die IY-Indexregister so, daß sie auf den oberen Bereich des Speicherbereichs zeigen.

Siehe Abschnitt 9.4 für eine vollständige Beschreibung von Hintergrund-ROM's.

Verwandte Einsprünge:

KL FIND COMMAND
KL LOG EXT
KL ROM WALK

155: KL LOG EXT #BCD1

Führe eine RSX in die Firmware ein.

Aktion:

Resident **S**ystem **E**xtensions sind ähnlich den Hintergrund-ROM's, werden jedoch in das RAM geladen. Diese Routine muß aufgerufen werden, um die RSX in die Liste zur externen Kommandounterstützung des Betriebssystemkerns einzutragen.

Einsprung-Bedingungen:

BC enthält die Adresse der RSX-Kommandotabelle
HL enthält die Adresse eines 4 Byte langen Bereichs im RAM, den der Kern benutzen kann.

Aussprung-Bedingungen:

DE zerstört,
alle anderen Register unverändert.

Anmerkungen:

Sowohl die RSX-Kommandotabelle als auch der Speicherbereich des Kernes müssen in den zentralen 32 K des Speichers liegen, d.h. nicht unterhalb eines ROM's.

Das Format einer Kommandotabelle ist in Abschnitt 9.2 beschrieben und die RSX's werden in Abschnitt 9.5 erläutert.

Verwandte Einsprünge:

KL FIND COMMAND
KL INIT BACK

156: KL FIND COMMAND #BCD4

Suche nach einer RSX, einem Hintergrund oder Vordergrund-ROM, um ein Kommando zu verarbeiten.

Aktion:

Alle Erweiterungs-ROM's und RSX's haben Kommandotabellen mit dem gleichen Format. diese Routine durchsucht alle RSX's und Hintergrund-ROM's in der Liste des Kernes zur externen Kommandounterstützung nach einer Übereinstimmung mit dem angegebenen Kommandonamen ab. Wenn der Name gefunden wurde, wird die Adresse der zugeordneten Routine zurückgegeben. Wenn es sich nicht um ein Hintergrund- oder RSX-Kommando handelt, werden alle Vordergrund-ROM's, die gefunden werden, nach einem Vordergrundprogramm mit dem angegebenen Namen durchsucht. Wenn ein Vordergrundprogramm gefunden wurde, erfolgt vom System sofort ein Einsprung.

Einsprung-Bedingungen:

HL enthält den Kommandonamen, nach dem gesucht wird.

Aussprung-Bedingungen:

Wenn ein RSX oder Hintergrund-ROM gefunden wurde:

CARRY-Flag „an"
C enthält die ROM-Auswahladresse
HL enthält die Adresse der Routine

Wenn das Kommando nicht gefunden wurde:

CARRY-Flag „aus"
C und HL zerstört

Immer:

A, B und DE zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der durchgereichte Kommandoname muß im RAM, kann dabei jedoch unterhalb des ROM's liegen. Der Name kann beliebig lang sein; es sind jedoch nur die ersten 16 Zeichen signifikant. Alle Buchstaben im Namen müssen Großbuchstaben sein und im letzten Buchstaben des Namens muß Bit 7 gesetzt sein.

Die ROM-Auswahladresse und die Adresse der Routine können für den Aufruf von KL FAR PCHL verwendet werden.

Die Liste der externen Kommandounterstützung wird erzeugt, wenn die Hintergrund-ROM's und RSX's initialisiert werden (siehe KL ROM WALK, KL INIT BACK und KL LOG EXT). Die Kommandotabellen werden in der umgekehrten Reihenfolge untersucht, in der die Kommandounterstützung sie einstellte. So erhalten die RSX's gegenüber den Hintergrund-ROM's den Vorzug, da die RSX's im allgemeinen nach den Hintergrund-ROM's initialisiert werden. Hintergrund-ROM's werden normalerweise in der umgekehrten Reihenfolge der ROM-Auswahladresse initialisiert, so erhalten ROM's mit niedriger Nummer den Vorzug vor ROM's mit höherer Nummer.

Siehe Abschnitt 9.2 für eine vollständige Beschreibung des Formats von Erweiterungs-ROM-Kommandotabellen.

Der erste Eintrag in einer Kommandotabelle in einem Hintergrund-ROM kann als ROM-Name benutzt werden. KL FIND COMMAND kann daher benutzt werden um herauszufinden, ob ein bestimmtes Hintergrund-ROM initialisiert wurde.

Bei der Suche nach einem Vordergrundprogramm werden die ROM's, beginnend bei ROM 0, durchsucht, bis eine nicht benutzte ROM-Adresse gefunden wird.

Das eingebaute BASIC kann über das Kommando BASIC aufgerufen werden.

Wenn ein Kommando für das Vordergrund ROM gefunden wird, wird bedingungslos in das ROM verzweigt und diese Routine kehrt nicht zurück.

Verwandte Einsprünge:

KL INIT BACK
KL LOG EXT
KL ROM WALK
MC START PROGRAM

157: KL NEW FRAME FLY #BCD7

Initialisiere und übergib einen Block an die Bildaufbauliste.

Aktion:

Der Betriebssystemkern pflegt eine Liste von Ereignissen die jedesmal, wenn ein Bild aufgebaut wird, angestoßen werden müssen. Diese Routine initialisiert einen Block und stellt ihn in diese Liste.

Einsprung-Bedingungen:

HL enthält die Adresse des Bildaufbaublocks
B enthält die Ereignis-Klasse
C enthält die ROM-Auswahladresse der Ereignisroutine
DE enthält die Adresse der Ereignisroutine

Aussprung-Bedingungen:

AF, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Bildaufbaublock ist 9 Bytes lang und muß in den zentralen 32 K des RAM liegen. Die letzten 7 Bytes des Bildaufbaublocks sind ein Ereignisblock, der mit den durchgereichten Parametern B, C und DE initialisiert wird (siehe KL INIT EVENT).
Das genaue Format eines Bildaufbaublocks ist in Anhang X beschrieben.

Der Bildaufbaublock wird, wenn nicht vorhanden, in die Bildaufbauliste aufgenommen.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL ADD FRAME FLY
KL DEL FRAME FLY
KL INIT EVENT

158: KL ADD FRAME FLY #BCDA

Übergib einen Block an die Bildaufbauliste.

Aktion:

Der Betriebssystemkern pflegt eine Liste von Ereignissen, die jedesmal, wenn ein Bild aufgebaut wird, angestoßen werden muß. Diese Routine stellt einen Block in diese Liste.

Einsprung-Bedingungen:

HL enthält die Adresse des Bildaufbaublocks

Aussprung-Bedingungen:

AF, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Bildaufbaublock ist 9 Bytes lang und muß in den zentralen 32 K des RAM liegen. Die letzten 7 Bytes des Bildaufbaublocks sind ein Ereignisblock, der mit den durchgereichten Parametern B, C und DE initialisiert wird (siehe KL INIT EVENT).

Das genaue Format eines Bildaufbaublocks ist in Anhang X beschrieben.

Der Bildaufbaublock wird, wenn nicht vorhanden, in die Bildaufbauliste aufgenommen.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL DEL FRAME FLY
KL INIT EVENT
KL NEW FRAME FLY

159: KL DEL FRAME FLY #BCDD

Lösche einen Block in der Bildaufbauliste.

Aktion:

Der Betriebssystemkern pflegt eine Liste von Ereignissen, die jedesmal, wenn ein Bild aufgebaut wird, angestoßen werden muß. Diese Routine löscht einen Block in der Bildaufbauliste.

Einsprung-Bedingungen:

HL enthält die Adresse des Bildaufbaublocks

Aussprung-Bedingungen:

AF, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine führt nichts durch, falls der Block nicht in der Liste ist.

Das Löschen eines Blocks in der Liste verhindert lediglich ein erneutes Anstoßen des Ereignisses. Es berührt keines der noch ausstehenden Bildaufbauereignisse.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL ADD FRAME FLY
KL NEW FRAME FLY

160: KL NEW FAST TICKER #BCE0

Initialisiere und übergib einen Block an die Schnell-Takt-Liste.

Aktion:

Der Betriebssystemkern pflegt eine Liste von Ereignissen, die jedesmal ausgegeben werden, wenn alle 1/300 Sekunden eine Unterbrechung durch den Zeitgeber erfolgt. Diese Liste wird als Schnell-Taktliste (Fast ticker list) bezeichnet. Diese Routine initialisiert einen Block und stellt ihn in diese Liste.

Einsprung-Bedingungen:

HL enthält die Adresse eines Schnell-Takt-Blockes
B enthält die Ereignisklasse
C enthält die ROM-Auswahladresse der Ereignisroutine
DE enthält die Adresse der Ereignisroutine.

Aussprung-Bedingungen:

AF, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Schnell-Takt-Block ist 9 Bytes lang und muß in den zentralen 32 K des RAM liegen. Die letzten 7 Bytes des Takt-Blockes sind ein Ereignisblock, der mit den durchgereichten Parametern B, C und DE initialisiert wird (siehe KL INIT EVENT). Das genaue Format eines Schnell-Takt-Blocks ist in Anhang X beschrieben.

Der Schnell-Takt-Block wird, wenn nicht vorhanden, in die Schnell-Takt-Liste aufgenommen.

Die Möglichkeit des Schnelltakts sind nicht für den allgemeinen Gebrauch vorgesehen. Sie ermöglichen jedoch, daß relativ kurze Zeitabstände gemessen werden können und ergeben eine genauere Auflösung als die Möglichkeiten des normalen Takts.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL ADD FAST TICKER
KL ADD TICKER
KL DEL FAST TICKER
KL INIT EVENT
KL TIME PLEASE

161: KL ADD FAST TICKER #BCE3

Übergib einen Block an die Schnell-Takt-Liste.

Aktion:

Der Betriebssystemkern pflegt eine Liste von Ereignissen, die jedesmal ausgegeben werden, wenn alle 1/300 Sekunden eine Unterbrechung durch den Zeitgeber erfolgt. Diese Liste wird als Schnell-Takt-Liste (Fast ticker list) bezeichnet. Diese Routine stellt einen Block in diese Liste.

Einsprung-Bedingungen:

HL enthält die Adresse des Schnell-Takt-Blockes

Aussprung-Bedingungen:

AF, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Schnell-Takt-Block ist 9 Bytes lang und muß in den zentralen 32 K des RAM stehen. Die letzten 7 Bytes des Takt-Blockes sind ein Ereignisblock, der initialisiert sein muß, bevor diese Routine aufgerufen wird.

Das genaue Format eines Schnell-Takt-Blockes ist in Anhang X beschrieben.

Der Schnell-Takt-Block wird, wenn nicht vorhanden, in die Schnell-Takt-Liste aufgenommen.

Die Möglichkeiten des Schnelltakts sind nicht für den allgemeinen Gebrauch vorgesehen. Sie ermöglichen jedoch, daß relativ kurze Zeitabstände gemessen werden können und ergeben eine genauere Auflösung als die Möglichkeiten des normalen Taktes.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL ADD TICKER
KL DEL FAST TICKER
KL INIT EVENT
KL NEW FAST TICKER
KL TIME PLEASE

162: KL DEL FAST TICKER #BCE6

Lösche einen Block in der Schnell-Takt-Liste.

Aktion:

Der Betriebssystemkern pflegt eine Liste von Ereignissen, die jedesmal ausgegeben werden, wenn alle 1/300 Sekunden eine Unterbrechung durch den Zeitgeber erfolgt. Diese Liste wird als Schnell-Takt-Liste (Fast ticker list) bezeichnet. Diese Routine löscht einen Block in dieser Liste.

Einsprung-Bedingungen:

HL enthält die Adresse des Schnell-Takt-Blockes

Aussprung-Bedingungen:

AF, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine führt nichts durch, falls der Block nicht in der Liste ist.

Das Löschen eines Blocks in der Liste verhindert lediglich ein erneutes Anstoßen des Ereignisses. Es berührt keines der noch ausstehenden Schnell-Takt-Ereignisse.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL ADD FAST TICKER
KL DEL TICKER
KL NEW FAST TICKER

163: KL ADD TICKER #BCE9

Übergib einen Block an die (normale) Takt-Liste.

Aktion:

Die Zeitgeberunterstützung für den allgemeinen Gebrauch mißt die Zeit in Einheiten von 1/50 Sekunden.
Der Betriebssystemkern pflegt eine Liste von Takt-Blöcken, von denen jeder einen Zähler und einen Wiederanlaufwert enthält. Alle 1/50 Sekunden verarbeitet der Kern alle Takt-Blöcke und verändert jeden Zähler um 1. Wenn der Zählereintrag eines Blocks auf 0 läuft, wird das im Block gespeicherte Ereignis angestoßen und der Zähler auf den Wiederanlaufwert gesetzt.

Einsprung-Bedingungen:

HL enthält die Adresse des Takt-Blocks
DE enthält den Ausgangswert des Zählers
BC enthält den Wiederanlaufwert

Aussprung-Bedingungen:

AF, BC und DE zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Takt-Block ist 13 Bytes lang und muß in den zentralen 32 K des Speichers liegen. Die letzten 7 Bytes des Takt-Blocks sind ein Ereignisblock, der initialisiert sein muß, bevor diese Routine aufgerufen wird. Das genaue Format eines Takt-Blockes ist in Anhang X beschrieben.

Es werden die Werte für den Zähler und den Wiederanlauf eingetragen; dann wird der Block in die Takt-Liste eingestellt, falls er nicht schon vorhanden ist. Diese Routine kann daher dazu verwendet werden, Zähler und Wiederanlaufwert eines bereits existierenden Blocks zu verändern.

Blöcke mit dem Eintrag 0 im Zähler werden bei der Abarbeitung der Liste ignoriert. Wenn daher ein Wiederanlaufwert von 0 angegeben wird, wird ein einmaliger Taktanstoß aufgesetzt. Da für das Ignorieren eines Takt-Blockes Zeit des Betriebssystemkerns beansprucht wird, sollten alle nicht benötigten Blöcke so bald wie möglich aus der Liste entfernt werden.

Es ist nicht möglich vorauszusagen, wie lange es nach dem Anstoß dauert, bis die Ereignisroutine tatsächlich aufgerufen wird, vor allem bei gleichzeitigen Ereignissen. Ungeachtet dieser Verzögerungen kann der Ticker dazu herangezogen werden, eine genaue Anzahl von Anstößen während einer angegebenen Zeitspanne zu erhalten, da der Wiederanlaufmechanismus den Zähler sofort wieder zurücksetzt. Das Ereignis-Zählverfahren stellt sicher, daß kein Anstoß verloren geht, unter der Berücksichtigung, daß nie mehr als 127 gleichzeitig anstehen.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL ADD FAST TICKER
KL DEL TICKER
KL INIT EVENT

164: KL DEL TICKER #BCEC

Lösche einen Block in der (normalen) Takt-Liste.

Aktion:

Wenn sich der angegebene Block in der Takt-Liste befindet, wird er gelöscht. Der Blockinhalt wird dabei nicht berührt.

Einsprung-Bedingungen:

HL enthält die Adresse des Takt-Blocks

Aussprung-Bedingungen:

Wenn der Block in der Takt-Liste gefunden wurde:

CARRY-Flag „an“
DE enthält den Zähler vor dem nächsten Ereignis

Wenn der Block nicht in der Takt-Liste gefunden wurde:

CARRY-Flag „aus“
DE zerstört

Immer:

A, HL und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Blockinhalt, insbesondere die Fortsetzung der Verarbeitung ausstehender Ereignisse, wird durch das Löschen aus der Liste nicht berührt. Der Block kann später wieder in die Liste gestellt werden und wird mit dem Zähler fortgesetzt, wie er beim Löschen war.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL ADD TICKER
KL DEL FAST TICKER

165: KL INIT EVENT #BCEF

Initialisiere einen Ereignisblock.

Aktion:

Initialisierung aller Einträge eines Ereignisblockes.

Einsprung-Bedingungen:

HL enthält die Adresse des Ereignisblockes
B enthält die Ereignisklasse
C enthält die ROM-Auswahladresse der Ereignisroutine
DE enthält die Adresse der Ereignisroutine

Aussprung-Bedingungen:

HL enthält die Adresse des Ereignisblocks + 7,
alle anderen Register unverändert.

Anmerkungen:

Der Ereignisblock ist 7 Bytes lang und muß in den zentralen 32 K des RAM liegen. Das Format eines Ereignisblocks ist in Anhang X beschrieben. Siehe zur allgemeinen Erläuterung von Ereignissen des Kapitel 11.

Die ROM-Auswahladresse und die Adresse der Routine sind die „ferne Adresse" (far address) der Ereignisroutine (siehe Kapitel 2).

Die Ereignisklasse ist bit-signifikant und zwar:

Bit 0:	„nahe Adresse"
Bit 1 – 4:	Priorität bei gleichzeitigen Ereignissen
Bit 5:	muß 0 sein
Bit 6:	Express-Ereignis
Bit 7:	nicht-gleichzeitiges Ereignis

Wenn Bit 7 nicht gesetzt ist, handelt es sich um ein gleichzeitiges Ereignis, ansonsten um ein nicht gleichzeitiges Ereignis. Nicht gleichzeitige Ereignisse haben keine Prioritäten und daher wird das Prioritätsfeld ignoriert. Wenn das Expressbit gesetzt ist, handelt es sich um ein eiliges Ereignis. Die Bedeutung hängt dabei davon ab, ob es sich um ein gleichzeitiges Ereignis handelt oder nicht.

Alle eiligen gleichzeitigen Ereignisse haben eine höhere Priorität als ein normales gleichzeitiges Ereignis. Die Priorität eines gleichzeitigen Ereignisses ist in den Bits 1 bis 4 der Klasse verschlüsselt. Je höher die Nummer, desto höher die Priorität. Ein Ereignis darf nicht die Priorität 0 haben. Die Verarbeitung normaler gleichzeitiger Ereignisse kann gesperrt werden (durch den Aufruf von KL EVENT DISABLE), während dies bei eiligen gleichzeitigen Ereignissen nicht möglich ist.

Ein eiliges nicht gleichzeitiges Ereignis ruft seine Ereignisroutine direkt aus dem Unterbrechungspfad. Ein normales nicht gleichzeitiges Ereignis wird direkt vor der Rückkehr aus der Unterbrechung verarbeitet (mit der Zulassung von Unterbrechungen).

Wenn Bit 0 gesetzt ist, liegt die Ereignisroutine entweder im unteren ROM oder in den zentralen 32 K des RAM. Die ROM-Auswahladresse wird ignoriert und die Routine wird, im Gegensatz zum FAR CALL-Mechanismus direkt aufgerufen, um den Verarbeitungsaufwand für das Ereignis zu reduzieren. Wenn möglich, sollten nicht gleichzeitige Ereignisse an „nahen Adressen" liegen. Eilige nicht gleichzeitige Ereignisse müssen immer an nahen Adressen liegen.

Ereignisblöcke sind Bestandteile vieler anderer Blöcke, die durch den Betriebssystemkern verwaltet werden, einschließlich Bildaufbau-, Schnelltakt- und Taktblöcke. Diese Routine dient zur Initialisierung des Ereignis-Teilblocks dieser Blöcke.

Die Bytes nach dem letzten Byte des Ereignisblocks werden vom Kern nicht benutzt, auch wenn der Block ein Teil eines anderen Blockes ist.

Wenn die Ereignisroutine aufgerufen wird, wird ihr die Adresse des Blockes durchgereicht, so daß der Anwender weitere Informationen über das Ereignis an den Block anhängen kann. Dies ermöglicht bei mehreren gleichartigen Ereignissen die Verwendung der gleichen Ereignisroutine, wenn bei jedem Ereignis die entsprechenden „eigenen" Variablen am Ereignisblock angefügt sind.

Die Ereignisroutine hat folgende Einsprung- und Aussprungbedingungen:

Einsprung:

Wenn die Ereignisroutine an einer „fernen Adresse" liegt:

HL enthält die Adresse von Byte 5 des Ereignisblocks (so daß irgendwelche angehängte Daten auf der Adresse HL + 2 beginnen können)

Wenn die Ereignisroutine an einer „nahen Adresse" liegt:

HL enthält die Adresse von Byte 6 des Ereignisblocks (so daß irgendwelche angehängte Daten auf der Adresse HL + 1 beginnen können)

Aussprung:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Die Ereignisroutine kann die IX- und IY-Register verwenden, muß sie aber erhalten. Sie darf nicht den zweiten Registersatz benutzen. Eilige nicht gleichzeitige Ereignisse dürfen keine Unterbrechungen zulassen.

KL INIT EVENT läßt Unterbrechungen zu.

Verwandte Einsprünge:

KL DEL SYNCHRONOUS
KL DISARM EVENT
KL EVENT
KL NEW FAST TICKER
KL NEW FRAME FLY
KL NEW TICKER
KL SYNC RESET

166: KL EVENT #BCF2

Stoße einen Ereignisblock an.

Aktion:

Der Ereignis-Mechanismus veranlaßt den Aufruf einer Ereignisroutine als Folge von jedem Anstoß eines Ereignisblocks. KL EVENT führt diesen Anstoß aus.

Einsprung-Bedingungen:

HL enthält die Adresse des Ereignisblocks.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Im Gegensatz zum überwiegenden Teil der Routinen des Betriebssystemkerns kann diese Routine vom Unterbrechungspfad aus gerufen werden. Da der Befehl LOW JUMP Im Firmware Hauptverzweigblock Unterbrechungen zuläßt, muß der Anwender den Adressteil der „unteren Adresse" aus dem Verzweigblock herausholen und die obersten beiden Bits mittels einer Maske rücksetzen, um die Adresse in das untere ROM von KL EVENT zu bringen. Die folgende Routine führt dies durch:

```
LD    DE,(#BCF2+1)          ; Auszug des Adressteils vom LOW JUMP
RES   7,D                   ; Rücksetzen Status oberes ROM
RES   6,D                   ; Rücksetzen Status unteres ROM
CALL PCDE _ INSTRUCTION     ; Aufruf KL EVENT
```

(Wenn der Anwender diese Operation wiederholt durchführen will, wird empfholen, daß die Adresse einmal herausgezogen und irgendwo zwischengespeichert wird).

Die Folge des Anstoßes hängt vom Ereigniszähler im Ereignisblock ab:

Zähler < 0: Das Ereignis ist gesperrt und der Anstoß hat keine Folgen.
Zähler > 0: Es stehen noch andere Anstöße aus und das Ereignis wird gerade verarbeitet. Dieser Anstoß erhöht lediglich den Zähler (wenn er nicht schon das Maximum von 127 erreicht hat).
Wenn die Verarbeitung eines Ereignisses einmal begonnen hat, wird sie fortgesetzt, bis der Zähler auf Null läuft oder das Ereignis gesperrt wird.

Zähler = 0: Das Ereignis ist zugelassen, aber die Verarbeitung des Ereignisses läuft nicht. Der Zähler wird erhöht und die Verarbeitung des Ereignisses angestoßen.

Wie die Verarbeitung des Ereignisses angestoßen wird, hängt von der Ereignisklasse ab.

Gleichzeitige Ereignisse:
Gleichzeitige Ereignisse werden in die Warteschlange für gleichzeitige Ereignisse, nach Prioritäten geordnet, eingestellt. Es liegt in der Verantwortung des Vordergrundprogramms, diese Warteschlange regelmäßig zu verarbeiten.

Ereignisroutinen für gleichzeitige Ereignisse werden aufgerufen, wenn das Vordergrundprogramm KL DO SYNC aufruft, der Ereigniszähler wird erst behandelt, wenn KL DONE SYNC aufgerufen wird.

Nicht-gleichzeitige Ereignisse:

a) nicht im Unterbrechungspfad

Die Ereignisroutine wird sofort aufgerufen. Nach der Rückkehr aus der Routine wird der Zähler, falls er größer als Null ist, vermindert. Wenn er immer noch größer als Null ist, wird diese Prozedur wiederholt.

b) Im Unterbrechungspfad – normales nicht-gleichzeitiges Ereignis

Das Ereignis wird in die Warteschlange für ausstehende Unterbrechungsereignisse eingereiht. Beim Verlassen des Unterbrechungspfads verarbeitet der Kern alle Ereignisse in dieser Warteschlange wie oben in a) beschrieben. Das heißt, daß Ereignisroutinen für normale nicht-gleichzeitige Ereignisse in Erweiterung der normalen Verarbeitung zwischen Rückkehr aus der Unterbrechung und dem Hauptprogramm aufgerufen werden. Die Routine ist daher nicht Gegenstand irgendwelcher Restriktionen die sich auf Routinen des Unterbrechungspfads beziehen.

c) Im Unterbrechungspfad – eiliges nicht-gleichzeitiges Ereignis

Die Unterbrechungsroutine wird sofort im Unterbrechungspfad aufgerufen. Die Routine muß auf einer „nahen" Adresse liegen (siehe KL INIT EVENT). Die Routine darf unter keinen Umständen Unterbrechungen zulassen.

KL EVENT kann unterbrochen werden, außer wenn es vom Unterbrechungspfad aufgerufen wird.

Verwandte Einsprünge:

KL INIT EVENT
KL NEXT SYNC
KL POLL SYNCHRONOUS
KL SYNC RESET

167: KL SYNC RESET #BCF5

Löschen der Warteschlange für gleichzeitige (synchrone) Ereignisse.

Aktion:

Die Warteschlange für gleichzeitige Ereignisse wird gelöscht – alle ausstehenden Ereignisse werden entfernt. Die augenblickliche Ereignispriorität, die von KL POLL SYNCHRONOUS und KL NEXT SYNC benutzt wird, um mittels einer Maske Ereignisse mit niedrigen Prioritäten auszuschalten, wird zurückgesetzt.

Aussprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Es liegt in der Verantwortung des Anwenders, sicherzustellen, daß die gelöschten Ereignisse und etwaige aktive Ereignisse zurückgesetzt werden. Der Ereigniszähler gelöschter Ereignisse könnte größer Null sein, so daß weitere Anstöße lediglich den Zähler erhöhen, ohne das Ereignis in die Warteschlange für gleichzeitige Ereignisse zu stellen; die Ereignisse sind dann wirkungslos.

Verwandte Einsprünge:

KL DEL SYNCHRONOUS
KL NEXT SYNC
KL POLL SYNCHRONOUS

168: KL DEL SYNCHRONOUS #BCF8

Löschen ein gleichzeitiges (synchrones) Ereignis aus der Ereigniswarteschlange.

Aktion:

Das Ereignis wird ignoriert. Wenn es sich in der Warteschlange für gleichzeitige Ereignisse befindet, wird es dort gelöscht.

Einsprung-Bedingungen:

HL enthält die Adresse des Ereignisblocks.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Das Löschen eines Ereignisses aus der Warteschlange verhindert die Verarbeitung ausstehender Anstöße.

Bevor ein Ereignisblock für gleichzeitige Ereignisse zurückgesetzt oder neu initialisiert wird, sollte diese Routine dazu benutzt werden, sicherzustellen, daß er im Augenblick nicht in Arbeit ist.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL DISARM EVENT
KL INIT EVENT
KL SYNC RESET

169: KL NEXT SYNC #BCFB

Hole das nächste Ereignis aus der Warteschlange.

Aktion:

Wenn sich ein Ereignis, dessen Priorität höher ist als die augenblickliche Ereignispriorität (falls vorhanden), in der Warteschlange für gleichzeitige Ereignisse befindet, dann wird das Ereignis aus der Schlange entfernt, die augenblickliche Priorität auf die des gelöschten Ereignisses gesetzt und die ursprüngliche Ereignispriorität zurückgegeben.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn ein Ereignis zur Verarbeitung ansteht:

CARRY-Flag „an“
HL enthält die Adresse des Ereignisblocks
A enthält die ursprüngliche Ereignispriorität (falls vorhanden)

Wenn kein Ereignis zu verarbeiten ist:

CARRY-Flag „aus“
A und HL zerstört

Immer:

DE zerstört,
alle anderen Register unverändert.

Anmerkungen:

KL NEXT SYNC gibt die Adresse des nächsten Ereignisses, das verarbeitet werden soll, zurück, welches, falls vorhanden, aus der Warteschlange für gleichzeitige Ereignisse genommen wurde und dessen Priorität nun als Maske für Ereignisprioritäten verwendet wird.

Das Verarbeitungsverfahren für gleichzeitige Ereignisse ist wie folgt:

```
TRY_AGAIN:
   CALL    KL_NEXT_SYNC     ; Rückgabe des nächsten Ereignisses (falls
                            ; vorhanden)
   JR      NC,??????        ; Sprung, wenn kein Ereignis vorhanden
;
   PUSH    HL               ; Sichern der Ereignisadresse
   PUSH    AF               ; Sichern der ursprünglichen Ereignispriorität
   CALL    KL_DO_SYNC       ; Rücksetzen der Maske für Ereignisprioritäten
   POP     AF
   POP     HL
;
   CALL    KL_DONE_SYNC     ; Rücksetzen der Prioritätsmerkmale
                            ; Behandeln des Ereigniszählers;
                            ; erneutes Einstellen des Ereignisses
                            ; in die Warteschlange, falls der
                            ; Zähler noch größer als Null
   JR      TRY_AGAIN        ; untersuchen, ob noch etwas zur
                            ; Verarbeitung ansteht
```

Das Vordergrundprogramm sollte zur Prüfung ausstehender Ereignisse regelmäßig KL POLL SYNCHRONOUS aufrufen.
KL POLL SYNCHRONOUS ist eine kurze Routine im RAM, so daß der Aufruf nur wenig Verarbeitungszeit beansprucht. Wenn ein Ereignis aussteht, dann sollte die obige Prozedur angestoßen und wiederholt werden, bis die Ereigniswarteschlange leer ist.

Das Verfahren zur Behandlung der laufenden Ereignisprioritäten erlaubt es den Ereignisroutinen, Ereignisse mit höherer Priorität auszuwählen und zu verarbeiten. Die von dieser Routine zurückgegebene Priorität muß sichergestellt werden, bis KL DONE SYNC durchgelaufen ist.

KL NEXT SYNC kann unterbrochen werden.

Verwandte Einsprünge:

KL DONE SYNC
KL DO SYNC
KL EVENT
KL INIT EVENT
KL POLL SYNCHRONOUS

170: KL DO SYNC #BCFE

Bearbeite eine Ereignisroutine.

Aktion:

Aufruf der Ereignisroutine für ein gegebenes Ereignis.

Aussprung-Bedingungen:

HL enthält die Adresse des Ereignisblocks.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine ist vorgesehen, ein Ereignis zu verarbeiten, nachdem KL NEXT SYNC festgestellt hat, daß es ansteht.
Es wird nicht empfohlen, diesen Einsprung zu irgendeiner anderen Zeit aufzurufen.

Siehe KL NEXT SYNC für ein allgemeines Schema zur Verarbeitung gleichzeitiger Ereignisse.

KL DO SYNC selbst berührt nicht den Ereigniszähler.

Verwandte Einsprünge:

KL DONE SYNC
KL NEXT SYNC

171: KL DONE SYNC #BD01

Beende die Verarbeitung eines Ereignisses.

Aktion:

Wenn durch den Aufruf der Ereignisroutine über KL DO SYNC ein Ereignis einmal verarbeitet wurde, muß dieser Einsprung aufgerufen werden, um die laufende Ereignispriorität wiederherzustellen und den Ereigniszähler zu behandeln. Wenn der Zähler größer als Null bleibt, wird der Ereignisblock in die Warteschlange für gleichzeitige Ereignisse zurückgestellt.

Einsprung-Bedingungen:

C enthält die ursprüngliche Ereignispriorität
HL enthält die Adresse des Ereignisblocks.

Aussprung-Bedingungen:

AF, BE, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine ist vorgesehen, nach dem Aufruf von KL NEXT SYNC (suchen eines anstehenden Ereignisses) und nach dem Aufruf von KL DO SYNC (verarbeiten der Ereignisroutine), aufgerufen zu werden.

Sie benutzt die vorhergehende Ereignispriorität und die von KL NEXT SYNC zurückgegebene Adresse des Ereignisblocks. Es wird nicht empfohlen, diesen Sprung in anderer Art anzuwenden.

Siehe unter KL NEXT SYNC für ein allgemeines Schema zur Verarbeitung gleichzeitiger Ereignisse. die Wiederherstellung der laufenden Ereignispriorität ist ein wichtiger Schritt in der Vorgehensweise zur Verwaltung der Priorität bei gleichzeitigen Ereignissen.

Wenn der Ereigniszähler größer als Null ist, wird er um 1 vermindert. Wenn der Zähler immer noch größer als Null bleibt, stehen noch weitere Ereignisse an und das Ereignis wird wieder in die Warteschlange gleichzeitiger Ereignisse zurückgestellt. Das Ereignis kann zwischen KL NEXT SYNC und KL DONE SYNC gesperrt werden. Das Setzen des Ereigniszählers auf 1, bevor KL DONE SYNC aufgerufen wird, ermöglicht, daß mehrere Ereignisse als ein einziges Ereignis behandelt werden.

KL DONE SYNC kann unterbrochen werden.

Verwandte Einsprünge:

KL DO SYNC
KL NEXT SYNC

172: KL EVENT DISABLE #BD04

Sperre normale gleichzeitige (synchrone) Ereignisse.

Aktion:

Sperren normaler gleichzeitiger Ereignisse, aber Zulassen eiliger gleichzeitiger Ereignisse. Dies wird erreicht, indem die laufende Ereignispriorität höher gesetzt wird, als jede mögliche Priorität normaler gleichzeitiger Ereignisse.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

KL EVENT DISABLE sieht keinen Schutz gegen das Anstoßen von Ereignissen vor. Der Effekt ist das Ausschalten aller anstehenden normalen Ereignisse durch das Setzen einer Maske, so daß sie vom Vordergrundprogramm verborgen bleiben (wenn KL POLL SYNCHRONOUS oder KL NEXT SYNC aufgerufen werden) und somit nicht verarbeitet werden.

KL EVENT ENABLE ist das Gegenstück zu KL EVENT DISABLE.

Es ist nicht möglich, gleichzeitige Ereignisse aus einer Ereignisroutine heraus auf Dauer zu sperren, da die vorausgegangene Ereignispriorität bei Rückkehr aus der Ereignisroutine wiederhergestellt wird.

Verwandte Einsprünge:

KL DISARM EVENT
KL EVENT ENABLE
KL NEXT SYNC
KL POLL SYNCHRONOUS

173: KL EVENT ENABLE #BD07

Lasse normale gleichzeitige Ereignisse zu.

Aktion:

Die Verarbeitung normaler und eiliger gleichzeitiger Ereignisse wird zugelassen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Ereignisse sind standardmäßig zugelassen. KL EVENT ENABLE ist das Gegenstück zu KL EVENT DISABLE.

Es ist nicht möglich, gleichzeitige Ereignisse aus einer Ereignisroutine heraus auf Dauer zu sperren, da die vorausgegangene Ereignispriorität bei Rückkehr aus der Ereignisroutine wiederhergestellt wird.

Verwandte Einsprünge:

KL EVENT DISABLE
KL NEXT SYNC
KL POLL SYNCHRONOUS

174: KL DISARM EVENT #BD0A

Verhindere das Auftreten eines Ereignisses.

Aktion:

Sperre ein Ereignis durch das Setzen des Ereigniszählers auf einen negativen Wert. Jeder weitere Anstoß (Aufrufe von KL EVENT) für dieses Ereignis wird ignoriert, alle ausstehenden Ereignisse werden gelöscht.

Einsprung-Bedingungen:

HL enthält die Adresse des Ereignisblocks.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

KL DISARM EVENT sollte nur in Verbindung mit nicht-gleichzeitigen Ereignissen angewendet werden. Gleichzeitige Ereignisse können über den Aufruf von KL DEL SYNCHRONOUS gesperrt werden, was auch sicherstellt, daß das Ereignis sich nicht in der Warteschlange für gleichzeitige Ereignisse befindet.

Das Ereignis kann durch Neuinitialisierung (KL INIT EVENT) oder durch Setzen des Ereigniszählers auf Null (Byte 2 des Ereignisblocks) wieder zugelassen werden.

Verwandte Einsprünge:

KL DEL SYNCHRONOUS
KL INIT EVENT

175: KL TIME PLEASE #BD0D

Frage nach der abgelaufenen Zeit.

Aktion:

Der Betriebssystemkern wartet einen Zähler, den er bei jeder Zeitunterbrechung hochsetzt. Der Zähler mißt die Zeit in Einheiten von 1/300 Sekunden. Die Routine gibt den augenblicklichen Stand des Zählers zurück.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

DEHL enthält den 4-Byte-Zähler
(D enthält das am meisten signifikante Byte und L das am wenigsten signifikante Byte)
alle anderen Register unverändert.

Anmerkungen:

Der Zähler wird auf Null gestellt, wenn der Rechner eingeschaltet oder zurückgesetzt wird. Der Zähler kann auf einen anderen Ausgangswert durch den Aufruf von KL SET TIME gesetzt werden.

Der Zähler wird nicht auf dem laufenden gehalten, wenn Unterbrechungen lange Zeit gesperrt werden, wie z.B. während des Lesens oder Schreibens von Kassetten.

Der 4-Byte-Zähler läuft nach ungefähr

```
   14.316.557  Sekunden
=     238.609  Minuten
=       3.977  Stunden
=         166  Tagen                    über.
```

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL TIME SET

176: KL TIME SET #BD10

Setze die abgelaufene Zeit.

Aktion:

Der Betriebssystemkern pflegt einen Zähler, den er bei jeder Zeitunterbrechung hochsetzt. Der Zähler mißt die Zeit in Einheiten von 1/300 Sekunden. Die Routine setzt den Zähler auf einen vorgegebenen Wert.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF zerstört
alle anderen Register unverändert.

Anmerkungen:

Der 4-Byte-Zähler läuft nach ungefähr

```
    14.316.557 Sekunden
=      238.609 Minuten
=        3.977 Stunden
=          166 Tagen
```

über.

KL TIME SET kann verwendet werden, um den Zähler auf die aktuelle Tageszeit zu setzen, so daß der Betriebssystemkern eine Echtzeituhr verwaltet, anstatt lediglich die abgelaufene Zeit seit dem letzten Rücksetzen zu messen.

Der Zähler wird nicht auf dem laufenden gehalten, wenn Unterbrechungen lange Zeit gesperrt werden, wie z.B. während des Lesens oder Schreibens von Kassetten.

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

KL TIME PLEASE

177: MC BOOT PROGRAM #BD13

Lade und starte ein Programm.

Aktion:

Rücksetzen von möglichst vielen Systemkomponenten. Anschließend Laden eines Programms und Ablauf desselben. Wenn das Laden fehlerhaft abläuft, wird das vorausgegangene Vordergrundprogramm erneut gestartet.

Einsprung-Bedingungen:

HL enthält die Adresse der Routine, die aufgerufen wird, um das Programm zu laden.

Aussprung-Bedingungen:

keine Rückkehr.

Anmerkungen:

Vor dem Versuch, das Programm zu laden, wird das System teilweise zurückgesetzt. Externe Unterbrechungen werden gesperrt, genauso die Zeitgeber-, Bildaufbau- und Tastatur-Unterbrechungsereignisse. Die Tonerzeugung wird abgeschaltet, die indirekten Verzweigungspunkte werden auf ihre Standardroutinen gesetzt und der Stapel (stack) wird auf den Standard-System-Stapel zurückgesetzt.

Diese Vorgehensweise stellt sicher, daß kein Speicher außerhalb des Variablenbereichs der Firmware benutzt wird, wenn das Programm geladen wird. Das Überschreiben eines aktiven Ereignisblocks oder indirekten Verzweigungspunktes könnte ansonsten unglückliche Konsequenzen haben.

Das teilweise Rücksetzen des Systems verändert nicht den Status des ROM oder die ROM-Auswahl. Die Routine, die zum Laden des Programms abläuft, muß im zugreifbaren RAM oder einem zugelassenen ROM sein. Es ist zu beachten, daß der Firmware-Verzweigblock normalerweise das untere ROM zuläßt und das obere ROM sperrt, und deshalb muß die Routine normalerweise im RAM über #4000 oder im unteren ROM liegen.

Die Routine, die zum Laden des Programms abläuft, kann jeden Speicher ab #0040 bis zur Basis des Variablenbereichs für die Firmware benutzen und kann indirekte Verzweigungspunkte verändern und Unterbrechungen von externen Geräten, soweit erforderlich, zulassen. Sie sollte folgende Ausgangsbedingungen beachten:

Wenn das Programm erfolgreich geladen wurde:

CARRY-Flag „an“
HL enthält den Einsprungpunkt des Programms

Wenn das Laden des Programms fehlerhaft war:

CARRY-Flag „aus“
HL zerstört

Immer:

A, BC, DE, IX, IY und andere Flags zerstört.

Nach dem erfolgreichen Laden wird die Firmware vollkommen neu initialisiert (wie während EMS) und das Programm wird an der Einsprungadresse, die von der Laderoutine zurückgegeben wurde, aufgerufen.

Eine Rückkehr aus dem Programm setzt das System zurück (RST0).

Nach einem fehlerhaften Laden wird eine entsprechende Fehlermeldung ausgegeben und das vorausgegangene Vordergrundprogramm wird erneut gestartet. Wenn der vorausgegangene Vordergrund selbst ein RAM-Programm war, wird statt dessen das Standard-ROM benutzt, da das Programm während des fehlerhaften Ladens zerstört worden sein könnte.

Verwandte Einsprünge:

CAS IN DIRECT
KL CHOKE OFF
MC START PROGRAM

178: MC START PROGRAM #BD16

Starte ein Vordergrundprogramm.

Aktion:

Vollständiges Initialisieren des Systems und Verzweigen in ein Programm.

Einsprung-Bedingungen:

HL enthält die Adresse des Einsprungpunktes
C enthält die erforderliche ROM-Auswahl

Aussprung-Bedingungen:

keine Rückkehr.

Anmerkungen:

HL und C stellen zusammen die „ferne Adresse" des Einsprungpunktes des Vordergrundprogramms dar (siehe Kapitel 2).

Wenn in ein im ROM stehendes Vordergrundprogramm verzweigt wird, muß die ROM-Auswahl so gesetzt sein, daß sie dem erforderlichen ROM entspricht. Wenn in ein Vordergrundprogramm verzweigt wird, das im RAM steht, muß die ROM-Auswahl so zum Sperren oder Zulassen von ROM's gesetzt werden, wie es das RAM-Programm benötigt (ROM-Auswahladressen #FC bis #FF).

Diese Routine führt eine volle EMS-Initialisierung der Firmware durch, bevor in das Programm verzweigt wird. Eine Rückkehr aus dem Programm setzt das System zurück (Ablauf von RST0).

MC START PROGRAM ist vorgesehen für Programme, die im ROM ablaufen sollen oder fürProgramme, die bereits im RAM geladen sind. Um ein RAM Programm zu laden und auszuführen, wird MC BOOT PROGRAM verwendet.

Verwandte Einsprünge:

MC BOOT PROGRAM
RESET ENTRY (RST0)

179: MC WAIT FLYBACK #BD19

Warte auf den Bildrücklauf.

Aktion:

Warte bis ein neuer Bildrücklauf erfolgt.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

alle Register und Flags unverändert.

Anmerkungen:

Der Bildrücklauf ist ein Signal, das vom Bildschirm-(Kathodenstrahl-) Controller erzeugt wird, um den Beginn der vertikalen Rücklaufperiode anzuzeigen. Während dieser Periode wird der Bildschirm nicht beschrieben und so können größere Operationen ohne störende Effekte auf dem Bildschirm durchgeführt werden.

Das beste Beispiel ist das Rollen des Bildschirms.

Das Bildrücklaufsignal dauert nur einige hundert Mikrosekunden, die vertikale Rücklaufperiode dauert um vieles länger. In der Mitte des Bildaufbaus gibt es jedoch eine Taktunterbrechung, die veranlassen kann, die Vordergrundverarbeitung für eine bestimmte Zeitdauer anzuhalten.

Es ist daher wichtig, alle kritischen Verarbeitungen so schnell wie möglich nach Entdekkung eines Bildrücklaufs durchzuführen.

Verwandte Einsprünge:

KL ADD FRAME FLY

180: MC SET MODE #BD1C

Setze den Bildschirmmodus.

Aktion:

Laden der Hardware mit dem geforderten Bildschirmmodus.

Einsprung-Bedingungen:

A enthält den geforderten Modus.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der geforderte Modus wird geprüft; falls er ungültig ist, wird keine Aktion durchgeführt. Wenn er gültig ist, wird der neue Wert an die Hardware weitergegeben.

Es gibt folgende Bildschirm-Modi:

0:	160 x 200 Bildpunkte (Pixels)	20 x 25 Zeichen
1:	320 x 200 Bildpunkte (Pixels)	40 x 25 Zeichen
2:	640 x 200 Bildpunkte (Pixels)	80 x 25 Zeichen.

Eine Änderung des Bildschirmmodus ohne Meldung an die Bildschirmverwaltung kann seltsame Effekte auf dem Bildschirm erzeugen. Um den Bildschirmmodus zu ändern, sollte allgemein immer SCR SET MODE aufgerufen werden. Diese Routine gibt im Anschluß dann den neuen Modus an die Hardware weiter.

Verwandte Einsprünge:

SCR SET MODE

181: MC SCREEN OFFSET #BD1F

Setze den Bildschirmoffset.

Aktion:

Lade die Hardware mit dem Abstand des ersten Bytes auf dem Bildschirm vom Beginn eines 2 K-Bildschirmblocks und dem 16 K (Basis-)Block, in dem der Bildschirmspeicher liegt.

Einsprung-Bedingungen:

A enthält die neue Bildschirmbasis
HL enthält den neuen Bildschirmabstand.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Bildschirmbasisadresse wird mit der Maske #C0 verknüpft um sicherzustellen, daß sie auf einen gültigen Speicherbereich verweist. Die standardmäßige Bildschirmbasis ist #C0 (der Bildschirm liegt unterhalb des oberen ROM).

Der Bildschirmabstand (Offset) wird mit der Maske #07FE verknüpft, um ihn gültig zu machen. Es ist zu beachten, daß Bit 0 ignoriert wird, da die Hardware nur geradzahlige Abstände verwendet.

Wenn die Bildschirmbasis oder der Bildschirmabstand ohne Meldung an die Bildschirmverwaltung verändert wird, können unerwartete Effekte auf dem Bildschirm auftreten. Es sollte deshalb üblicherweise SCR SET BASE oder SCR SET OFFSET aufgerufen werden. Diese geben dann die Werte an die Hardware weiter.

Verwandte Einsprünge:

SCR SET BASE
SCR SET OFFSET

182: MC CLEAR INKS #BD22

Setze alle Inks auf eine Farbe.

Aktion:

Angabe der Farbe des Bildrandes und der Inks. Alle Inks werden auf die gleiche Farbe gesetzt, so daß der Eindruck entsteht, daß der Bildschirm einheitlich gelöscht wurde.

Einsprung-Bedingungen:

DE enthält die Adresse eines Ink-Vektors.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Ink-Vektor hat folgendes Format:

Byte 0:	Farbe für den Bildrand
Byte 1:	Farbe für alle Inks.

die angegebenen Farben sind die Nummern, die von der Hardware benutzt werden und nicht die Graustufennummern, die bei SCR SET INK angegeben werden (siehe Anhang V).

Wenn der Bildschirm gelöscht wurde, können die richtigen Inkfarben durch den Aufruf von MC SET INKS angegeben werden.

Diese Routine setzt die Farben für alle 16 Inks, unabhängig davon, ob sie im augenblicklichen Modus auf dem Bildschirm dargestellt werden können oder nicht.

Die Löschtechnik für Inks wird von der Bildschirmverwaltung angewandt, wenn der Bildschirm gelöscht wird (über SCR CLEAR) oder wenn der Modus geändert wird (über SCR SET MODE).

Verwandte Einsprünge:

MC SET INKS

183: MC SET INKS #BD25

Setze Farben für alle Inks.

Aktion:

Angabe der Farben aller Inks und des Bildrahmens.

Einsprung-Bedingungen:

DE enthält die Adresse eines Ink-Vektors.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Der Ink-Vektor hat folgendes Format:

Byte 0: Farbe des Bildrandes
Byte 1: Farbe der Ink 0
Byte 2: Farbe der Ink 1
...
Byte 16: Farbe der Ink 15.

Die angegebenen Farben sind die Nummern, die von der Hardware benutzt werden und nicht die Graustufennummern, die bei SCR SET INK angegeben werden (siehe Anhang V).

Diese Routine setzt die Farben für alle Inks, einschließlich derer, die im augenblicklichen Bildschirmmodus nicht sichtbar sind. Es ist jedoch nur für sichtbare Inks notwendig, erkennbare Farben anzugeben.

Die Bildschirmverwaltung setzt die Farben für alle Inks bei jedem Blinken der Inks und wenn eine Inkfarbe verändert wurde (über den Aufruf von SCR SET INK oder SCR SET BORDER).

Verwandte Einsprünge:

MC CLEAR INKS
SCR SET BORDER
SCR SET INK

184: MC RESET PRINTER #BD28

Setze den indirekten Verzweigpunkt für den Drucker zurück.

Aktion:

Setze den indirekten Verzweigpunkt (Indirection) für den Drucker, MC WAIT PRINTER, auf seine Standardroutine.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine hat keine anderen Auswirkungen.

Verwandte Einsprünge:

MC WAIT PRINTER
MC PRINT CHAR

185: MC PRINT CHAR #BD2B

Versuche, ein Zeichen an den Centronics-Ausgang zu senden.

Aktion:

Sende ein Zeichen an den Drucker (Centronics-Ausgang) oder führe eine Zeitabschaltung durch, falls der Drucker zu lange beschäftigt ist.

Einsprung-Bedingungen:

A enthält das zu sendende Zeichen (Bit 7 wird ignoriert).

Aussprung-Bedingungen:

Wenn das Zeichen richtig gesendet wurde:

CARRY-Flag „an"

Wenn eine Zeitabschaltung erfolgte:

CARRY-Flag „aus"

Immer:

A und die anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine ruft den indirekten Verzweigpunkt für die Geräteverwaltung, MC WAIT PRINTER, um das Zeichen zu senden. Die standardmäßige indirekte Routine wartet, bis der Centronics-Ausgang „nicht-beschäftigt" meldet, um das Zeichen zu senden. Wenn der Ausgang zu lange beschäftigt ist (ungefähr 0,4 Sekunden), führt die Routine eine Zeilenabschaltung durch und das Zeichen wird nicht gesendet. Diese Zeitabschaltung ist vorgesehen, um auf Unterbrechungen abfragen zu können, während der Drukker betrieben wird.

Verwandte Einsprünge:

MC RESET PRINTER
MC WAIT PRINTER

186: MC BUSY PRINTER #BD2E

Untersuche, ob der Centronics-Ausgang beschäftigt ist.

Aktion:

Untersuche, ob der Drucker (Centronics-Ausgang) beschäftigt (busy) ist.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn der Centronics-Ausgang beschäftigt ist:

CARRY-Flag „an“

Wenn der Centronics-Ausgang bereit ist:

CARRY-Flag „aus“

Immer:
alle anderen Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine hat keine anderen Auswirkungen.

Verwandte Einsprünge:

MC SEND PRINTER

187: MC SEND PRINTER #BD31

Sende ein Zeichen an den Centronics-Ausgang.

Aktion:

Sende ein Zeichen an den Drucker (Centronics-Ausgang), der nicht beschäftigt (busy) sein darf.

Einsprung-Bedingungen:

A enthält das zu sendende Zeichen (Bit 7 wird ignoriert).

Aussprung-Bedingungen:

CARRY-Flag „an“
A und die anderen Flags zerstört
alle anderen Register unverändert.

Anmerkungen:

Der Drucker darf nicht beschäftigt sein, wenn das Zeichen gesendet wird. Die höherstufige Routine MC PRINT CHAR wartet automatisch auf den Drucker, bis er „nicht-beschäftigt“ meldet und sollte daher bevorzugt angewendet werden.

Verwandte Einsprünge:

MC BUSY PRINTER
MC PRINT CHAR

188: MC SOUND REGISTER #BD34

Sende Daten an ein Ton-Chip-Register.

Aktion:

Setzen eines Tones in einem Register eines Ton-Chips. Dies ist aufgrund der Hardwarekonstruktion eine ziemlich verwickelte Aufgabe.

Einsprung-Bedingungen:

A enthält die Nummer des Ton-Chip-Registers
C enthält die zu sendenden Daten

Aussprung-Bedingungen:

AF und BC zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine kann unterbrochen werden.

Verwandte Einsprünge:

keine

189: JUMP RESTORE #BD37

Setze den Standard-Verzweigblock zurück.

Aktion:

Setze den Firmware-Hauptverzweigblock (Sprungtabelle, Jumpblock) auf seinen standardmäßigen Zustand, wie er in den Kapiteln 13.1 und 14 beschrieben ist.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Routine kann dazu verwendet werden, den Verzweigblock auf seine Standardroutine zurückzusetzen, wenn der Anwender in ihm Einträge verändert hatte. Der gesamte Verzweigblock wird gesetzt, so daß beim Zusammenspiel mit anderen Programmen Vorsicht angebracht ist.

Der indirekte Verzweigblock wird durch die verschiedenen Initialisierungs- und Rücksetzroutinen der einzelnen Pakete stückweise zurückgesetzt.

JUMP RESTORE setzt die indirekten Verzweigpunkte nicht zurück.

Verwandte Einsprünge:

GRA RESET
KM RESET
MC RESET PRINTER
SCR RESET
TXT RESET

15 Firmware Indirections

Dieser Abschnitt beschreibt detailliert die Einsprung- und Aussprung-Bedingungen, sowie die Wirkungen der Routinen in der Indirections-Sprungtabelle (auch indirekte Verzweigungstabelle). In Abschnitt 13.2 befindet sich eine Liste dieser Routinen.

Die Firmware-Indirections werden durch die Firmware aus Schlüsselpositionen entnommen. Sie ermöglichen dem Anwender das Abfangen und Ändern der Firmware-Aktionen, ohne ein komplett neues Firmware-Paket erstellen zu müssen.

Die Beschreibung bezieht sich auf die Standard-Einstellung der Indirections. Ersatz-Routinen müssen nicht unbedingt alle Eigenschaften der Standard-Routinen besitzen, empfehlenswert ist es dennoch.

IND: TXT DRAW CURSOR #BDCD

Plaziere das Cursor-Symbol auf dem Bildschirm (sofern freigeschaltet).

Standard-Aktion:

Nur wenn das Cursor-Symbol frei- und eingeschaltet ist, wird es auf dem Bildschirm dargestellt. Ist dies nicht der Fall, so geschieht nichts. Die momentane Text-Position wird in das Fenster hineingezwungen (siehe TXT VALIDATE) und das Cursor-Symbol an der sich so ergebenden Position dargestellt. Das Cursor-Symbol ist ein inverses Symbol. Diese Routine ist nur dann zweimal nacheinander aufrufbar, wenn TXT UNDRAW CURSOR zwischendurch aufgerufen wurde.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Indirection ermöglicht dem Anwender die Veränderung der Form des Cursor-Symbols. Unter TXT PLACE CURSOR ist die normale Schreibweise des Cursor-Symbols beschrieben.

Alle Text-VDU-Routinen, die vom Bildschirm lesen, auf den Bildschirm schreiben oder die momentane Position verändern, entfernen den Cursor (unter Verwendung von TXT UNDRAW CURSOR), bevor sie ihre Aktionen ausführen und plazieren ihn anschließend zurück (unter Verwendung von TXT DRAW CURSOR). Ein Beispiel für eine derartige Routine ist TXT WR CHAR, die ein Zeichen auf den Bildschirm schreibt.

Diese Indirection wird beim Aufruf von TXT INITIALISE oder TXT RESET eingestellt.

Verwandte Einsprünge:

TXT PLACE CURSOR
TXT UNDRAW CURSOR

IND: TXT UNDRAW CURSOR #BDD0

Entferne das Cursor-Symbol vom Bildschirm (sofern freigeschaltet).

Standard-Aktion:

Nur wenn das Cursor-Symbol frei- und eingeschaltet ist, wird es auf dem Bildschirm entfernt. Ist es dies nicht, so geschieht nichts. Das Cursor-Symbol ist ein inverses Symbol. Diese Routine wird nur aufgerufen, nachdem TXT DRAW CURSOR zum Plazieren des Cursors auf dem Bildschirm verwendet wurde.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Indirection ermöglicht dem Anwender die Veränderung der Form des Cursor-Symbols. Unter TXT REMOVE CURSOR ist die normale Schreibweise des Cursor-Symbols beschrieben. Die Text-VDU-Routinen rufen diese Indirection zum Entfernen des Cursors vom Bildschirm auf. Alle Text-VDU-Routinen, die vom Bildschirm lesen, auf den Bildschirm schreiben oder die momentane Position verändern, entfernen den Cursor (unter Verwendung von TXT UNDRAW CURSOR), bevor sie ihre Aktionen ausführen und plazieren ihn anschließend zurück (unter Verwendung von TXT DRAW CURSOR). Ein Beispiel für eine derartige Routine ist TXT WR CHAR, die ein Zeichen auf den Bildschirm schreibt.

Diese Indirection wird beim Aufruf von TXT INITIALISE oder TXT RESET eingestellt.

Verwandte Einsprünge:

TXT DRAW CURSOR
TXT REMOVE CURSOR

IND: TXT WRITE CHAR #BDD3

Schreibe ein Zeichen auf den Bildschirm.

Standard-Aktion:

Gebe ein Zeichen auf den Bildschirm an einer Zeichen-Position aus.

Einsprung-Bedingungen:

A beinhaltet das zu schreibende Zeichen.
H beinhaltet die physikalische Spalte, in der zu schreiben ist.
L beinhaltet die physikalische Reihe, in der zu schreiben ist.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Zeichen-Position, auf der zu schreiben ist, wird in physikalischen Koordinaten angegeben. So ist z.B. Reihe 0 und Spalte 0 die obere linke Ecke des Bildschirms. Die Position wird nicht auf Zulässigkeit überprüft.

TXT WRITE CHAR wird durch TXT WR CHAR aufgerufen, um ein Zeichen auf dem Bildschirm darzustellen. Das Entfernen des Cursor-Symbols und die Berechnung der neuen momentanen Position wird durch TXT WR CHAR und nicht durch TXT WRITE CHAR bewirkt.

Diese Indirection wird beim Aufruf von TXT INITIALISE oder TXT RESET eingestellt.

Verwandte Einsprünge:

TXT OUTPUT
TXT UNWRITE
TXT WR CHAR

IND: TXT UNWRITE #BDD6

Lies ein Zeichen vom Bildschirm.

Standard-Aktion:

Versuche, ein Zeichen auf einer Zeichen-Position vom Bildschirm zu lesen.

Einsprung-Bedingungen:

H beinhaltet die physikalische Spalte, von der gelesen werden soll.
L beinhaltet die physikalische Reihe, von der gelesen werden soll.

Aussprung-Bedingungen:

Wenn ein lesbares Zeichen gefunden wurde:
Carry-Flag „an„.
A beinhaltet das gelesene Zeichen.

Wenn kein erkennbares Zeichen gefunden wurde:
CARRY-Flag „aus“.
A beinhaltet Null.

Immer:
BC, DE, HL und andere Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Zeichen-Position, von der gelesen werden soll, ist in physikalischen Koordinaten angegeben. Das bedeutet, Reihe 0 und Spalte 0 ist die obere linke Ecke des Bildschirms. Die Position wird nicht auf Zulässigkeit geprüft.

Diese Indirection wird durch TXT RD CHAR zum Lesen eines Zeichens vom Bildschirm aufgerufen. TXT RD CHAR entfernt vor dem Aufruf dieser Indirection den Cursor vom Bildschirm.

Das Lesen wird durch Vergleich der auf dem Bildschirm gefundenen Matrix mit der zum Erzeugen von Zeichen verwendeten Matrix bewerkstelligt. Folglich führt die Änderung einer Zeichen-Matrix, der Pen- oder Paper-Inks oder des Bildschirms (z.B. das Zeichnen einer Linie über ein Zeichen) dazu, daß das Zeichen nicht mehr lesbar ist. Insbesondere verursacht das Cursor-Symbol Verwirrung. Es sollte sich zum Lese-Zeitpunkt nicht auf dem Bildschirm befinden.

Spezielle Vorsichtsmaßnahmen sind gegen die Erzeugung von Leerzeichen getroffen worden. Anfangs wird das Zeichen unter der Annahme gelesen, daß es mit der momentanen Pen-Ink geschrieben ist. Führt diese Annahme nicht zur Erzeugung eines erkennbaren Zeichens oder eines Leerzeichens (Zeichen #20), wird ein erneuter Versuch unter der Annahme, daß der Hintergrund des Zeichens mit der momentanen Paper-Ink geschrieben wurde, gestartet.

Die Zeichen werden beginnend mit #00 und abschließend mit #FF abgetastet. Wenn zwei mögliche Zeichen-Matrizen mit dem Bildschirm übereinstimmen, wird das kleinere der beiden Zeichen zurückgegeben.

Diese Indirection wird beim Aufruf von TXT INITIALISE oder TXT RESET eingestellt.

Verwandte Einsprünge:

TXT RD CHAR
TXT WRITE CHAR

IND: TXT OUT ACTION #BDD9

Gieb ein Zeichen oder einen Steuer-Code (Control-Code) aus.

Standard-Aktion:

Stelle ein Zeichen auf dem Bildschirm dar oder führe einen Steuer-Code aus (Zeichen #00...#1F). Arbeitet über den momentan selektierten Kanal (mit unten angeführter Ausnahme).

Einsprung-Bedingungen:

A beinhaltet das Zeichen oder den Code.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Indirection wird durch TXT OUTPUT aufgerufen, um ein Zeichen darzustellen oder einen Steuer-Code auszuführen. Sie wird dem Anwender zur Verfügung gestellt, um die Methode der Bearbeitung von Zeichen und Steuer-Codes ändern zu können oder die Ausgabe umzustellen (z.B. auf den Drucker). TXT OUTPUT schützt die Register nur vor und nach dem Aufruf von TXT OUT ACTION.

Steuer-Codes können bis zu 9 Parameter in Anspruch nehmen. Sie sind Zeichen, die dem Steuer-Code folgen. Die übergebenen Zeichen werden in einem Puffer gespeichert, bis alle Parameter erfolgreich empfangen sind. Der Steuer-Code-Puffer besitzt nur die Größe zur Speicherung von 9 Parameter-Zeichen.

Es gibt nur einen Steuer-Code-Puffer, der zwischen allen Kanälen aufgeteilt ist. Das Übergeben einer Steuer-Code-Sequenz kann unvorhersehbare Folgen haben, wenn der Ausgabe-Kanal gleichzeitig geändert wird.

Bei gesperrtem VDU werden keine Zeichen auf dem Bildschirm dargestellt. Steuer-Codes werden jedoch ausgeführt.

Ist der Schreib-Modus für Graphik-Zeichen freigeschaltet, so werden alle Zeichen und Steuer-Codes unter Verwendung des Graphik-VDU dargestellt (siehe GRA WR CHAR) und nicht ausgeführt. Normalerweise werden Zeichen durch den Text-VDU geschrieben (siehe TXT WR CHAR).

Diese Indirection wird beim Aufruf von TXT INITIALISE oder TXT RESET eingestellt.

Verwandte Einsprünge:

TXT OUTPUT
TXT WR CHAR

IND: GRA PLOT #BDDC

Stelle einen Punkt dar (Plotten).

Standard-Aktion:

Prüfe, ob der Punkt innerhalb des momentanen Fensters liegt. Wenn dies der Fall ist, schreibe ihn in der momentanen Graphik-Pen-Ink und unter Verwendung des momentanen Graphik-Schreib-Modus. Die momentane Graphik-Position wird immer auf den spezifizierten Punkt bewegt.

Einsprung-Bedingungen:

DE beinhaltet die Anwender-Koordinate X des darzustellenden Punktes.
HL beinhaltet die Anwender-Koordinate Y des darzustellenden Punktes.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Position des darzustellenden Punktes ist in Anwender-Koordinaten angegeben, d.h. relativ zum Anwender-Ursprung.

Diese Indirection wird durch GRA PLOT RELATIVE und GRA PLOT ABSOLUTE aufgerufen, um den geforderten Punkt darzustellen. Der Anwender hat die Möglichkeit, die Verfahrensweise der Darstellung zu ändern (z.B. die Ausgabe an einen X-Y-Plotter). GRA PLOT RELATIVE übersetzt relative in Anwender-Koordinaten und ruft dann diese Indirection auf. GRA PLOT ABSOLUTE ruft diese Indirection unmittelbar auf.

Zum Schreiben eines Punktes auf dem Bildschirm wird die Indirection SCR WRITE benutzt. Dadurch wird der Punkt unter Verwendung des momentanen Graphik-Schreib-Modus dargestellt.

Diese Indirection wird beim Aufruf von GRA INITIALISE oder GRA RESET eingestellt.

Verwandte Einsprünge:

GRA PLOT ABSOLUTE
GRA PLOT RELATIVE
GRA TEST
SCR WRITE

IND: GRA TEST #BDDF

Teste einen Punkt.

Standard-Aktion:

Teste, ob der Punkt innerhalb des Graphik-Fensters liegt und gib die Ink, auf die der Punkt momentan eingestellt ist, zurück. Die momentane Graphik-Position wird auf den spezifizierten Punkt bewegt.

Einsprung-Bedingungen:

DE beinhaltet die Anwender-Koordinate X des zu testenden Punktes.
HL beinhaltet die Anwender-Koordinate Y des zu testenden Punktes.

Aussprung-Bedingungen:

A beinhaltet die decodierte Ink des spezifierten Punktes.
BC, DE, HL und die Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Position des zu testenden Punktes ist in Anwender-Koordinaten angegeben, d.h. relativ zum Anwender-Ursprung.

Diese Indirection wird durch GRA TEST RELATIVE und GRA TEST ABSOLUTE zum Testen des geforderten Punktes verwendet. Der Anwender kann die Test-Methode ändern (vergleichbar mit der momentanen Pen-Ink). GRA TEST RELATIVE übersetzt relative in Anwender-Koordinaten und ruft dann diese Indirection auf. GRA TEST ABSOLUTE ruft diese Indirection unmittelbar auf.

Die Indirection SCR READ wird zum Testen der Ink eines Punktes innerhalb des Fensters verwendet.

Diese Indirection wird beim Aufruf von GRA INITIALISE oder GRA RESET eingestellt.

Verwandte Einsprünge:

GRA PLOT
GRA TEST ABSOLUTE
GRA TEST RELATIVE
SCR READ

IND: GRA LINE #BDE2

Zeichne eine Linie.

Standard-Aktion:

Zeichne eine Linie mit der momentanen Graphik-Pen-Ink zwischen der momentanen Graphik-Position und dem gegebenen Endpunkt unter Verwendung des momentanen Graphik-Schreib-Modus. Punkte auf der Linie, die außerhalb des momentanen Graphik-Fensters liegen, werden nicht dargestellt. Die momentane Graphik-Position wird immer auf den spezifizierten Endpunkt bewegt.

Einsprung-Bedingungen:

DE beinhaltet die Anwender-Koordinate X des Endpunktes.
HL beinhaltet die Anwender-Koordinate Y des Endpunktes.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die Position des Endpunktes ist in Anwender-Koordinaten angegeben, d.h. relativ zum Anwender-Ursprung.

Diese Indirection wird durch GRA LINE RELATIVE und GRA LINE ABSOLUTE zum Zeichnen der geforderten Linie verwendet. Das zum Zeichnen von Linien verwendete Verfahren kann geändert werden (z.B. für die Ausgabe an einen X-Y-Plotter). GRA LINE RELATIVE übersetzt relative in Anwender-Koordinaten und ruft dann diese Indirection auf. GRA LINE ABSOLUTE ruft die Indirection unmittelbar auf.

Die Linie ist in horizontale oder vertikale Abschnitte unterteilt, die separat gezeichnet werden (siehe SCR HORIZONTAL und SCR VERTICAL). Die Indirection SCR WRITE wird aufgerufen, um die Punkte in diesen Abschnitten zu schreiben. Die Linie wird also unter Verwendung des momentanen Graphik-Schreib-Modus dargestellt.

Diese Indirection wird beim Aufruf von GRA INITIALISE oder GRA RESET eingestellt.

Verwandte Einsprünge:

GRA LINE ABSOLUTE
GRA LINE RELATIVE
SCR HORIZONTAL
SCR VERTICAL

IND: SCR READ #BDE5

Lies einen Pixel vom Bildschirm.

Standard-Aktion:

Lies einen Pixel vom Bildschirm und decodiere seine Ink.

Einsprung-Bedingungen:

HL beinhaltet die Bildschirm-Adresse des Pixels.
C beinhalted die Maske für den Pixel.

Einsprung-Bedingungen:

A beinhaltet die decodierte Ink, auf die der Pixel eingestellt war.
Flags zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die mitgelieferte Maske muß eine Maske für einen einzelnen Pixel sein, da andernfalls die Decodierung der vom Bildschirm gelesenen Ink nicht korrekt arbeitet.

Diese Indirection wird durch Aufruf von SCR INITIALISE oder SCR RESET eingestellt. Sie wird durch GRA TEST aufgerufen.

Verwandte Einsprünge:

GRA TEST
SCR WRITE

IND: SCR WRITE #BDE8

Schreibe einen oder mehrere Pixel unter Verwendung des momentanen Graphik-Schreib-Modus.

Standard-Aktion:

Stelle einen oder mehrere Pixel unter Verwendung des momentanen Graphik-Modus dar.

Einsprung-Bedingungen:

HL beinhaltet die Bildschirm-Adresse der/des Pixel(s).
C beinhaltet die Maske für den/die Pixel.
B beinhaltet die codierte Ink für die Darstellung.

Aussprung-Bedingungen:

AF zerstört,
alle anderen Register unverändert.

Anmerkungen:

Die mitgelieferte Maske kann für einen oder mehrere Pixel (oder keinen Pixel) gelten. Die mitgelieferte Ink sollte zur Inanspruchnahme des gesamten Bytes codiert sein (siehe SCR INK ENCODE).

Der Pixel wird unter Verwendung des momentanen Schreib-Modus des Graphik-VDU dargestellt.

Diese Modi sind:

FORCE:	Pixel werden unabhängig von der alten Ink auf die neue Ink gesetzt.
XOR:	Pixel werden auf die Ink gesetzt, welche durch eine Exklusiv-Oder-Verknüpfung der neuen Ink mit der momentanen Einstellung entsteht.
AND:	Pixel werden auf die Ink gesetzt, welche durch eine Und-Verknüpfung der neuen Ink mit der momentanen Einstellung entsteht.
OR:	Pixel werden auf die Ink gesetzt, welche durch eine Oder-Verknüpfung der neuen Ink mit der momentanen Einstellung entsteht.

Der Schreib-Modus kann durch Aufruf von SCR ACCESS entsprechend eingestellt werden.

Diese Indirection wird zum Darstellen von Pixel auf dem Bildschrim durch sämtliche Schreib-Routinen des Graphik-VDU aufgerufen. Insbesondere durch GRA PLOT RELATIVE, GRA PLOT ABSOLUTE, GRA LINE RELATIVE, GRA LINE ABSOLUTE und GRA WR CHAR.

Diese Indirection wird durch Aufruf von SCR INITIALISE oder SCR RESET eingestellt.

Verwandte Einsprünge:

GRA PLOT
SCR ACCESS
SCR PIXELS
SCR READ

IND: SCR MODE CLEAR #BDEB

Lösche den Bildschirm (auf Ink 0).

Standard-Aktion:

Lösche den Bildschirm-Speicher (auf Nullen). Diese Indirection wird dem Anwender zur Verfügung gestellt, um ein Löschen des Bildschirms nach einer Modus-Änderung zu verhindern.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Normalerweise bewirkt diese Indirection die gleichen Aktionen, wie in SCR CLEAR beschrieben.

Diese Indirection wird durch Aufruf von SCR INITIALISE oder SCR RESET eingestellt.

Beim Aufruf dieser Indirection befinden sich der Text- und Graphik-VDU nicht im Standard-Zustand.

Verwandte Einsprünge:

SCR CLEAR
SCR SET MODE

IND: KM TEST BREAK #BDEE

Test auf Abbruch (oder Reset).

Standard-Aktion:

Teste, ob die Escape-Taste gedrückt ist. Wenn nicht, wird keine Aktion eingeleitet. Wenn Escape, Shift und Control und keine anderen Tasten gedrückt sind, wird das System rückgesetzt. Andernfalls wird ein Abbruch-Ereignis übergeben (siehe KM BREAK EVENT).

Einsprung-Bedingungen:

Unterbrechungen gesperrt.
C beinhaltet die Shift- und Control-Tasten-Zustände.

Aussprung-Bedingungen:

AF und HL zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Indirection wird durch die Unterbrechungsebene der Firmware aufgerufen. Demzufolge müssen Unterbrechungen gesperrt sein und bleiben.

Wenn Bit 7 in C gesetzt ist, wurde eine Control-Taste gedrückt. Wenn Bit 5 in C gesetzt ist, wurde eine der Shift-Tasten gedrückt.

Diese Indirection wird nach der Tasten-Abfrage, und nachdem die Escape-Taste gedrückt wurde, aufgerufen. Sie ermöglicht dem Anwender die Änderung der Arbeitsweise eines Abbruchs (insbesondere zum Schutz des System-Reset, siehe RESET ENTRY).

Diese Indirection wird durch Aufruf von KM INITIALISE oder KM RESET eingestellt.

Verwandte Einsprünge:

KM BREAK EVENT

IND: MC WAIT PRINTER #BDF1

Drucke ein Zeichen oder warte auf den Ablauf der Zeitperiode.

Standard-Aktion:

Warte, bis der Centronics-Kanal nicht mehr beschäftigt ist, und übergib dann ein Zeichen an ihn. Wenn der Kanal lange Zeit im Busy-Zustand verbleibt, wartet die Routine auf den Ablauf der Zeitperiode und übersendet das Zeichen nicht.

Einsprung-Bedingungen:

A beinhaltet das zu sendende Zeichen.

Aussprung-Bedingungen:

Wenn das Zeichen korrekt gesendet wurde:
Carry-Flag „an".

Wenn die Zeitperiode verstrichen ist:
Carry-Flag „aus".

Immer:
A und B zerstört,
alle anderen Register unverändert.

Anmerkungen:

Diese Indirection ermöglicht dem Anwender das Ansteuern des Druckers auf verschiedene Art und Weise.

Es können z.B. Escape-Sequenzen bearbeitet oder die Zeitperiode geändert werden.

Diese Indirection wird durch die Routine MC PRINT CHAR aufgerufen. Sie testet in der gleichen Weise wie MC BUSY PRINTER, ob der Drucker beschäftig ist, und sendet das Zeichen in gleicher Weise wie MC SEND PRINTER.

Diese Indirection wird durch den Aufruf von MC RESET PRINTER eingestellt.

Verwandte Einsprünge:

MC BUSY PRINTER
MC PRINT CHAR
MC SEND PRINTER

16 Sprungtabelle des oberen Betriebssystem-Kerns (High Kernel)

Neben der Haupt-Firmware-Sprungtabelle gibt es eine kleine Sprungtabelle für Betriebssystem-Kern-Routinen, die mit dem ROM-State und -Select befaßt ist. Die über diese Sprungtabelle erreichbaren Routinen befinden sich im RAM, um Verwechslungen während der Änderung des ROM-State und -Select zu vermeiden. Der RAM-Bereich wird bei der Power-up-Initialisierung aus dem ROM herauskopiert. Die Sprungtabelle sollte durch den Anwender nicht geändert werden.

Der Eintrag KL POLL SYNCHRONOUS ist ein Sonderfall unter den Routinen in dieser Sprungtabelle. Im Unterschied zu den anderen Bearbeitungs-Routinen für synchrone Ereignisse, die sich im unteren ROM befinden, ist diese Routine RAM-resident.

Dadurch wird der Aufwand beim Abfragen auf synchrone Ereignisse minimiert.

Eine zusammenfassende Auflistung der Einträge in dieser Sprungtabelle ist in Abschnitt 13.3 zu finden. Eine Erörterung der ROMs und der Speicherbelegung befindet sich in Abschnitt 2 und 9. Eine Beschreibung der Ereignisse ist in Abschnitt 11 zu finden.

HI: KL U ROM ENABLE #B900

Schalte das obere ROM frei.

Aktion:

Schaltet das momentan selektierte obere ROM frei. Durch Lesen von Adresse #C000 und aufwärts kann der Inhalt des ROM's bestimmt werden.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A beinhaltet den vorherigen ROM-State.
Flags zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Der bereitgestellte Mechanismus für den Aufruf von Subroutinen im oberen ROM und für die Auswahl der oberen ROMs schaltet automatisch das erforderliche ROM frei. Diese Routine wird überwiegend durch die Firmware verwendet.

Der vorherige ROM-State kann an KL ROM RESTORE übergeben werden, um auf den Zustand vor Aufruf dieser Routine zurückzusetzen. Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL L ROM ENABLE
KL ROM RESTORE
KL ROM SELECT
KL U ROM DISABLE

HI: KL U ROM DISABLE #B903

Sperre das obere ROM.

Aktion:

Sperrt das obere ROM. Durch Lesen von Adresse #C000 und aufwärts kann der Inhalt des RAM?s bestimmt werden.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A beinhaltet den vorherigen ROM-State.
Flags zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Die Sperrung des oberen ROM's ermöglicht Lesezugriffe auf die oberen 16KB RAM, die gewöhnlich als Bildschirm-Speicher verwendet werden. Man beachte, daß die Belegung der Speicherplätze im Bildschirm-Speicher mit den Bildschirm-Pixeln vom Modus und dem Bildschirm-Offset abhängig ist. Es ist nicht ratsam, das obere ROM zu sperren, während Befehle in ihm ausgeführt werden.

Der vorherige ROM-State kann an KL ROM RESTORE übergeben werden, um auf den Zustand vor Aufruf dieser Routine zurückzusetzen.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL L ROM DISABLE
KL ROM RESTORE
KL U ROM ENABLE

HI: KL L ROM ENABLE #B906

Schalte das untere ROM frei.

Aktion:

Schaltet das untere ROM frei. Durch Lesen unterhalb von Adresse #4000 kann der Inhalt des ROM's bestimmt werden.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A beinhaltet den vorherigen ROM-State.
Flags zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Das untere ROM ist normalerweise gesperrt, wenn keine Firmware-Routine aufgerufen wird. Die Firmware-Sprungtabelle arrangiert automatisch die Freischaltung des unteren ROM's und die Sperrung desselben nach dem Routinen-Rücksprung. Diese Routine wird überwiegend durch die Firmware verwendet.

Der vorherige ROM-State kann an KL ROM RESTORE übergeben werden, um auf den Zustand vor Aufruf dieser Routine zurückzusetzen.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL L ROM DISABLE
KL ROM RESTORE
KL U ROM ENABLE

HI: KL L ROM DISABLE #B909

Sperre das untere ROM.

Aktion:

Sperrt das untere ROM. Durch Lesen unterhalb von Adresse #4000 kann der Inhalt des RAM's bestimmt werden.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A beinhaltet den vorherigen ROM-State.
Flags zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Das untere ROM ist normalerweise gesperrt, wenn keine Firmware-Routine aufgerufen wird. Die Firmware-Sprungtabelle arrangiert automatisch die Freischaltung des unteren ROM's und die Sperrung desselben nach dem Routinen-Rücksprung.

Der vorherige ROM-State kann an KL ROM RESTORE übergeben werden, um auf den Zustand vor Aufruf dieser Routine zurückzusetzen.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL L ROM ENABLE
KL ROM RESTORE
KL U ROM DISABLE

HI: KL L ROM RESTORE #B90C

Stelle den vorherigen ROM-State wieder her.

Aktion:

Alle Änderungs-Routinen für den ROM-State übergeben einen Wert, der den vorherigen ROM-State bezeichnet. Die Übergabe dieses Wertes an KL ROM RESTORE stellt den Zustand vor der Änderung wieder her.

Einsprung-Bedingungen:

A beinhaltet den vorherigen ROM-State.

Aussprung-Bedingungen:

AF zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Der vorherige ROM-State ist der Wert, der von

KL U ROM ENABLE,
KL U ROM DISABLE,
KL L ROM ENABLE,
KL L ROM DISABLE, oder
KL ROM SELECT

übergeben wird.

Es besteht die Möglichkeit, KL U ROM DISABLE zum Umkehren der Auswirkungen von KL U ROM ENABLE zu verwenden (dies ist nur eine von mehreren möglichen Kombinationen). Der Aufruf von KL ROM RESTORE ist jedoch die zu bevorzugende Methode, da sie den ursprünglichen Zustand, unabhängig von der Freischaltmethode, wiederherstellt.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL L ROM DISABLE
KL L ROM ENABLE
KL ROM SELECT
KL U ROM DISABLE
KL U ROM ENABLE

HI: KL L ROM SELECT #B90F

Selektiere ein bestimmtes oberes ROM.

Aktion:

Selektiere ein gegebenes oberes ROM und schalte es frei.

Einsprung-Bedingungen:

C beinhaltet die ROM-Select-Adresse des geforderten ROM's.

Aussprung-Bedingungen:

C beinhaltet die ROM-Select-Adresse des vorher selektierten ROM's.
B beinhaltet den vorherigen ROM-State.
AF zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Der vorherige Zustand (State) kann an KL ROM RESTORE übergeben werden, um die Freischaltung des ROM's auf den ursprünglichen Zustand rückzusetzen. Der vorherige State und Select können an KL ROM DESELECT übergeben werden, um den vorherigen State und Select des vormals selektierten ROM's wiederherzustellen.

Der für den Routinen-Aufruf in Erweiterung-ROMs bereitgestellte Mechanismus führt die ROM-Auswahl automatisch durch (siehe Abschnitt 2).

Es ist nicht ratsam, ein anderes oberes ROM auszuwählen, während Befehle im oberen ROM ausgeführt werden.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL CURR SELECTION
KL PROBE ROM
KL ROM DESELECT
KL ROM RESTORE

HI: KL CURR SELECTION #B912

Frage, welches obere ROM momentan selektiert ist.

Aktion:

Übergibt die ROM-Select-Adresse des momentan selektierten oberen ROM's.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

A beinhaltet die ROM-Select-Adresse des momentan selektierten ROM's.
Alle anderen Register und Flags unverändert.

Anmerkungen:

Es ist nicht möglich, die ROM-Select-Adresse eines bestimmten Erweiterung-ROM's vorherzusagen. Die zur Bezugnahme von Subroutinen auf Erweiterung-ROM's verwendete „Far-Address" beinhaltet ein ROM-Select-Byte, das zur Laufzeit eingestellt werden muß. Diese Routine übergibt die ROM-Select-Adresse des momentanen ROM's, so daß sie die geeignete „Far-Address" einstellen kann.

Verwandte Einsprünge:

KL PROBE ROM
KL ROM SELECT

HI: KL PROBE ROM #B915

Frage nach Klasse und Version eines ROM's.

Aktion:

Die ersten Bytes aller oberen ROM's beinhalten Informationen (in einem Standard-Format) über die ROM's. Diese Routine extrahiert die Klasse, Markierungs- und Versions-Nummer aus dem ROM mit der gegebenen ROM-Select-Adresse.

Einsprung-Bedingungen:

C beinhaltet die ROM-Select-Adresse des zu prüfenden ROM's.

Aussprung-Bedingungen:

A beinhaltet die ROM-Klasse.
L beinhaltet die ROM-Markierungs-Nummer.
H beinhaltet die ROM-Versions-Nummer.
B und Flags zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Die übergebene ROM-Klasse kann folgende Werte annehmen:

0:	Vordergrund-ROM.
1:	Hintergrund-ROM.
2:	Erweiterungs-Vordergrund-ROM.
#80:	ROM auf der Karte (mit dem BASIC-Vordergrund-Programm).

Die Auswahl einer ROM-Adresse, deren zugeordnetes ROM nicht bestückt ist, selektiert das ROM auf der Karte und übergibt die Klasse #80.

Die Bedeutung der Markierungs- und Versions-Nummern hängt vom jeweiligen ROM ab.

In Abschnitt 9 befindet sich eine Beschreibung der Erweiterungs-ROM's.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL ROM SELECT
KL CURR SELECTION

HI: KL ROM DESELECT #B918

Stelle die vorherige obere ROM-Auswahl wieder her.

Aktion:

Versetze den ROM-State und oberen ROM-Select in den Zustand, den sie vor dem Aufruf von KL ROM SELECT hatten.

Einsprung-Bedingungen:

C beinhaltet die ROM-Select-Adresse des vorher selektierten ROM's.
B beinhaltet den vorherigen ROM-State.

Aussprung-Bedingungen:

C beinhaltet die ROM-Select-Adresse des momentan selektierten ROM's.
B zerstört.
Alle anderen Register und Flags unverändert.

Anmerkungen:

Der vorherige ROM-Select und -State sind die durch KL ROM SELECT übergebenen Werte. Das durch diese Routine übergebene momentan selektierte ROM ist das ROM, welches durch Aufruf von KL ROM SELECT selektiert wurde (falls nicht weitere Selektionen vorgenommen werden).

Der Mechanismus für den Aufruf von Subroutinen in Erweiterungs-ROM's bewerkstelligt die ROM-Auswahl automatisch.

Es ist nicht ratsam, während der Ausführung von Befehlen aus dem oberen ROM ein anderes oberes ROM zu selektieren.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL CURR SELECTION
KL ROM RESTORE
KL ROM SELECT

HI: KL LDIR #B91B

Verschiebe Speicherplatz (LDIR) mit ausgeschalteten ROM's.

Aktion:

Führt einen LDIR-Befehl (LoaD Increment and Repeat) mit gesperrten oberen und unteren ROM's aus.

Einsprung-Bedingungen:

BC, DE, HL, wie durch den LDIR-Befehl gefordert.

Aussprung-Bedingungen:

F, BC, DE, HL, wie durch den LDIR-Befehl gesetzt.
Alle anderen Register unverändert.

Anmerkungen:

Diese Routine kann zum Verschieben von RAM-Bereichen, unabhängig vom ROM-State, verwendet werden.

Diese Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL LDIR
RAM LAM (RST 4)

HI: KL LDDR #B91E

Verschiebe Speicherplatz (LDDR) mit ausgeschalteten ROM's.

Aktion:

Führt einen LDDR-Befehl (LoaD Decrement and Repeat) mit gesperrten oberen und unteren ROM's aus.

Einsprung-Bedingungen:

BC, DE, HL, wie durch den LDDR-Befehl gefordert.

Aussprung-Bedingungen:

F, BC, DE, HL, wie durch den LDDR-Befehl gesetzt.
Alle anderen Register unverändert.

Anmerkungen:

Diese Routine kann zum Verschieben von RAM-Bereichen, unabhängig vom ROM-State, verwendet werden.

Die Routine schaltet Unterbrechungen frei.

Verwandte Einsprünge:

KL LDIR
RAM LAM (RST 4)

HI: KL POLL SYNCHRONOUS #B921

Prüfe, ob ein Ereignis mit höherer Priorität als das momentane ansteht.

Aktion:

Wenn die Warteschlange für synchrone Ereignisse nicht leer ist, wird die Priorität des anstehenden Ereignisses der höchsten Priorität mit der momentanen Ereignis-Priorität (sofern vorhanden) verglichen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Wenn ein Ereignis höherer Priorität ansteht:

Carry-Flag „an".

Wenn kein Ereignis höherer Priorität ansteht:

Carry-Falg „aus".

Immer:

A und andere Flags zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Diese Routine befindet sich in der oberen Sprungtabelle, um den Aufwand für die Abfrage auf synchrone Ereignisse zu minimieren. Wenn die Warteschlange für synchrone Ereignisse leer ist, nimmt die Routine nur wenige Befehle in Anspruch.

Während der Bearbeitung eines synchronen Ereignisses erkennt der Betriebssystem-Kern seine Priorität. Die Routine für synchrone Ereignisse kann die Warteschlange selbst abfragen, es werden jedoch nur Ereignisse höherer Priorität gemeldet.

Die Routine kann Unterbrechungen freischalten.

Verwandte Einsprünge:

KL EVENT
KL DONE SYNC
KL DO SYNC
KL NEXT SYNC

17 Sprungtabelle des unteren Betriebssystem-Kerns (Low Kernel)

Der untere Speicherbereich von #0000 bis inklusive #003F ist mit dem Code für die Restart-Befehle (RST) und einigen Betriebssystem-Kern-Einträgen belegt. Die meisten dieser Einträge befassen sich mit Zugriffen auf Subroutinen im ROM und RAM. Die Restarts sind:

RST 0 bewirkt einen System-Reset.

RST-Befehle 1 bis inklusive 5 werden zur Erweiterung des Z80-Befehlssatzes verwendet und ermöglichen spezielle CALL- und JUMP-Befehle, die erweiterte Adressen zur Einbeziehung von ROM-State - und -Select-Komponenten benutzen.

RST 6 steht dem Anwender zur Verfügung.

RST 7 wird für Unterbrechungen benötigt.

Da alle bereitgestellten Einträge unabhängig davon, ob das untere ROM freigeschaltet oder gesperrt ist, verfügbar sein müssen, wird der Speicherbereich während der Power-up-Initialisierung vom ROM in das RAM kopiert.

Der Anwender sollte diese Sprungtabelle nicht verändern (mit Ausnahme der USER RESTART- und EXT INTERRUPT-Bereiche). Ändert der Anwender dennoch diese Bereiche, so liegt es in seiner Verantwortung, sicherzustellen, daß diese Änderungen andere Programme nicht beeinflussen. Dies kann einigermaßen sichergestellt werden, wenn das untere ROM beim Ablauf der Programme immer freigeschaltet bleibt. Da die anderen Programme jedoch das untere ROM sperren können, ist diese Maßnahme in den meisten Fällen erfolglos. Wenn diesbezüglich irgendwelche Zweifel bestehen, sollte der Original-Inhalt der Sprungtabelle wiederhergestellt werden.

Abschnitt 2 beinhaltet eine Erörterung der ROM's und der Speicherbelegung, Abschnitt 9 eine allgemeine Beschreibung externer ROM's. Eine zusammenfassende Übersicht der in diesem Bereich angelegten Routinen befindet sich in Abschnitt 13.4.

LOW: RESET ENTRY RST0 #0000

Vollständiger Reset der Maschine wie beim Power-up.

Aktion:

Beim ersten Einschalten der Maschine beginnt die Ausführung an dieser Stelle. Der Aufruf von oder Sprung auf #0000 bzw. die Ausführung des RST 0 bewirkt einen Reset der Maschine, die einen Anfangszustand wie beim Power-up einnimmt.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

keine.

Anmerkungen:

Die gesamte Hardware wird zurückgesetzt und die Firmware komplett initialisiert. Nachdem alle Tabellen und Sprungtabellen aufgebaut sind, wird die Steuerung an den Standard-Eintrag in ROM 0 übergeben (siehe Abschnitt 9).

Verwandte Einsprünge:

MC START PROGRAM

LOW: LOW JUMP RST1 #0008

Sprung in das untere ROM oder RAM, nimmt eine Inline-„Low-Address“ zum Einsprung.

Aktion:

RST 1 wird zur Erweiterung des Befehlssatzes verwendet. Er ist eine erweiterte Form des Sprung-Befehls. Er sollte mit einer 2-Byte-„Low-Address“ versehen sein, die das Sprungziel und den erforderlichen ROM-State spezifiziert.

Einsprung-Bedingungen:

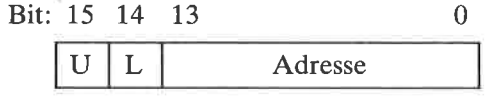
Alle Register und Flags werden unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Die dem Restart-Befehl folgende „Low-Address“ ist folgendermaßen belegt:



„U“-Bit gesetzt: oberes ROM gesperrt.
„L“-Bit gesetzt: unteres ROM gesperrt.
„Adresse“ ist die aktuelle Adresse der anzuspringenden Ziel-Routine, nachdem der ROM-State gesetzt wurde.

Bei der Rückkehr aus der Ziel-Routine wird der ROM-State auf den ursprünglichen Zustand (wie vor dem Jump) eingestellt. Um dies möglich zu machen, werden 4 Bytes auf den Stack „gepushed“. Die Überwachung des Stack sollte sorgsam geschehen (z.B. um die Adresse der Inline-Parameter aufzufinden).

Der LOW JUMP (RST 1) kann das erste Byte eines JP-Befehls (JUMP) ersetzen. Eine Anwendung ist für Sprungtabellen möglich. Die Haupt-Firmware-Sprungtabelle besteht überwiegend aus LOW JUMP-Befehlen.

Es wird vorausgesetzt, daß das Sprungziel eine Routine ist, die auf übliche Art und Weise zurückkehrt.

Der Restart-Befehl selbst kehrt nicht zurück. Der Wert des oberen Stack-Endes (Top of Stack) muß daher bei der Ausführung eines LOW JUMP eine Rückkehr-Adresse sein. Die Ausführung eines LOW JUMP schaltet Unterbrechung frei.

Verwandte Einsprünge:

FAR CALL (RST3)
FIRM JUMP (RST5)
KL FAR ICALL
KL FAR PCHL
KL LOW PCHL

LOW: KL LOW PCHL #000B

Sprung in das untere ROM oder RAM. Register HL beinhaltet die "Low Address" zum Einsprung.

Aktion:

Nimm eine „Low-Adress" als Parameter und zum Einsprung. Die „Low-Adress" spezifiziert die Adresse des Sprungziels und den erforderlichen ROM-State.

Einsprung-Bedingungen:

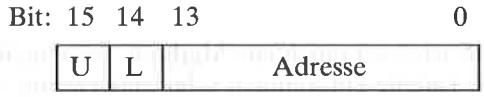
HL beinhaltet die „Low-Address" zum Einsprung.
Alle Register und Flags werden unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Die „Low-Address" ist folgendermaßen belegt:



„U"-Bit gesetzt: oberes ROM gesperrt.
„L"-Bit gesetzt: unteres ROM gesperrt.
„Adresse" ist die aktuelle Sprungadresse, wenn der ROM-State gesetzt wurde.

Bei der Rückkehr aus der Ziel-Routine wird der ROM-State auf den ursprünglichen Zustand (wie vor dem Jump) eingestellt. Um dies möglich zu machen, werden 4 Bytes auf den Stack „gepushed". Die Überwachung des Stack sollte sorgsam geschehen (z.B. um die Adresse der Inline-Parameter aufzufinden).

Es wird vorausgesetzt, daß das Sprungziel eine Routine ist, die auf übliche Art und Weise zurückkehrt. Der Wert des oberen Stack-Endes (Top fo Stack) muß daher bei der Ausführung eines LOW PCHL eine Rückkehr-Adresse sein.
Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

KL FAR ICALL
KL FAR PCHL
LOW JUMP (RST1)
PCHL INSTURCTION

LOW: PCBC INSTRUCTION #000E

Springe auf die Adresse in BC.

Aktion:

Äquivalent zum JP (HL)-Befehl (oder PCHL in einigen Assembler-Dialekten) mit dem Unterschied, daß das Ziel in BC und nicht in HL steht.

Einsprung-Bedingungen:

BC beinhaltet die Adresse zum Sprung.
Alle Register und Flags werden unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Der Aufruf PCBC INSTRUCTION ist eine brauchbare Methode, um eine Routine aufzurufen, deren Adresse aus einer Tabelle entnommen oder anderweitig zur Laufzeit festgelegt wurde.

Verwandte Einsprünge:

KL FAR PCHL
KL LOW PCHL
KL SIDE PCHL
PCDE INSTRUCTION
PCHL INSTRUCTION

LOW: SIDE CALL RST2 #0010

Aufruf in ein „Sideways-ROM“, nimmt eine Inline „Side-Address“ zum Aufruf.

Aktion:

RST 2 dient der Erweiterung des Befehlssatzes. Er ist eine erweiterte Form des CALL-Befehls. Er sollte mit einer 2 Byte langen „Side-Address“ versehen sein, die den aufzurufenden Speicherplatz und den erforderlichen ROM-Select spezifiziert.

Einsprung-Bedingungen:

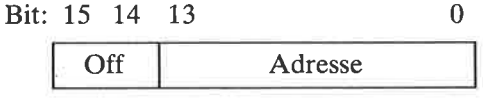
Alle Register und Flags werden mit Ausnahme von IY (IY zeigt auf den oberen Daten-Bereich des Hintergrund-ROM's) unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

IY ist geschützt.
Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Die dem Restart-Befehl folgende „Side-Address“ ist folgendermaßen belegt:



„Off“: Wert zwischen 0...3, der nach Addition mit der ROM-Select-Adresse des Haupt-Vordergrund-ROM's die ROM-Select-Adresse des erforderlichen ROM's angibt.

„Adresse“: Nach der Addition mit #C000 ist „Adresse“ die Adresse der aufzurufenden Routine.

Die Ziel-Routine springt auf den Befehl zurück, welcher der Inline „Side-Address“ unmittelbar folgt. Der ROM-Select und -State werden auf den Zustand vor dem Aufruf zurückgesetzt.

Um dies möglich zu machen, werden 6 Bytes auf den Stack „gepushed“. Die Überwachung des Stack sollte sorgsam geschehen (z.B. um die Adresse der Inline-Parameter aufzufinden).

Beim Einsprung in die Ziel-Routine ist das untere ROM gesperrt und das entsprechende obere ROM selektiert und freigeschaltet.

SIDE CALLs werden zur Unterstützung von über eine Anzahl ROMs (bis zu vier) verteilten Vordergrund-Programmen bereitgestellt (siehe Abschnitt 9 unter Erweiterungs-ROMs).

Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

FAR CALL (RST3)
KL SIDE PCHL

LOW: KL SIDE PCHL #0013

Aufruf in ein „Sideways-ROM“, HL beinhaltet die „Side-Address“ zum Aufruf.

Aktion:

Nimmt eine „Side-Address“ und ruft sie auf. Die „Side-Address“ spezifiziert die Adresse der aufzurufenden Routine und den ROM-Select des oberen ROM's.

Einsprung-Bedingungen:

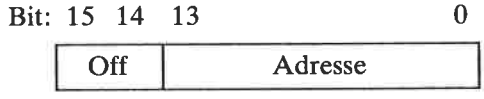
HL beinhaltet die „Side-Address“ zum Aufruf.
Alle Register und Flags werden mit Ausnahme von IY (IY zeigt auf den oberen Daten-Bereich des Hintergrund-ROM's) unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

IY ist unverändert.
Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Die „Side-Address“ ist folgendermaßen belegt:



„Off“: Wert zwischen 0...3, der nach Addition mit der ROM-Select-Adresse des Haupt-Vordergrund-ROM's die ROM-Select-Adresse des erforderlichen ROM's angibt.

„Adresse“: Nach der Addition mit #C000 ist „Adresse“ die Adresse der aufzurufenden Routine.

Beim Einsprung in die Ziel-Routine ist das untere ROM gesperrt und das entsprechende obere ROM selektiert und freigeschaltet.

Beim Rücksprung aus der Ziel-Routine werden der ROM-Select und -State auf den Zustand vor dem Aufruf zurückgesetzt. Um dies möglich zu machen, werden 6 Bytes auf den Stack „gepushed“. Die Überwachung des Stack sollte sorgsam geschehen (z.B. um die Adresse der Inline-Parameter aufzufinden).

SIDE CALLs werden zur Unterstützung von über eine Anzahl ROMs (bis zu vier) verteilten Vordergrund-Programmen bereitgestellt (siehe Abschnitt 9 unter Erweiterungs-ROMs).

Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

FAR CALL (RST3)
KL FAR ICALL
KL FAR PCHL

LOW: PCDE INSTRUCTION #0016

Sprung auf die Adresse in DE.

Aktion:

Äquivalent zum JP (HL)-Befehl (oder PCHL in einigen Assembler-Dialekten), mit dem Unterschied, daß das Ziel in DE und nicht in HL steht.

Einsprung-Bedingungen:

DE beinhaltet die Adresse zum Sprung.
Alle Register und Flags werden unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Der Aufruf von PCDE INSTRUCTION ist eine brauchbare Methode, um eine Routine aufzurufen, deren Adresse aus einer Tabelle entnommen oder anderweitig zur Laufzeit festgelegt wurde.

Verwandte Einsprünge:

KL FAR PCHL
KL LOW PCHL
KL SIDE PCHL
PCBC INSTRUCTION
PCHL INSTRUCTION

LOW: FAR CALL RST3 #0018

Aufruf einer Subroutine im RAM oder ROM, nimmt eine Inline-Adresse der „Far-Adress“

Aktion:

RST 3 dient der Erweiterung des Befehlssatzes. Er ist eine erweiterte Form des CALL-Befehls und ermöglicht den Aufruf von Routinen im RAM oder ROM. Der Restart ist mit der Adresse einer 3 Byte langen „Far-Address“ versehen, die den aufzurufenden Speicherplatz und den erforderlichen ROM-State und -Select spezifiziert.

Einsprung-Bedingungen:

Alle Register und Flags werden mit Ausnahme von IY (IY zeigt auf den oberen Datenbereich des Hintergrund-ROM's) unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

IY ist geschützt.
Alle anderen Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Der Restart-Befehl nimmt einen 2 Byte langen Inline-Parameter als die Adresse einer „Far-Address“. Die „Far-Address“ ist folgendermaßen belegt:

Byte: 0 1 2

Adresse	ROM

Bytes 0...1: Adresse der aufzurufenden Routine:
Byte 2 ROM-Select-Byte: übernimmt Werte wie folgt:

#00...#FB: Selektiere das gegebene ROM, schalte das obere frei und sperre das untere.
#FC: Keine Änderung des ROM-Select, schalte das obere frei und schalte das untere frei.
#FD: Keine Änderung des ROM-Select, schalte das obere frei und sperre das untere.
#FE: Keine Änderung des ROM-Select, sperre das obere und schalte das untere frei.
#FF: Keine Änderung des ROM-Select, sperre das obere und untere ROM.

Die „Far-Address“ ist nicht direkt im FAR CALL-Befehl enthalten, da das ROM-Select-Byte für ROM-Routinen von der spezifischen Konfiguration der Erweiterungs-ROM's abhängig ist und daher zur Laufzeit bestimmt und eingestellt werden muß.

Register werden mit Ausnahme von IY unberührt an die Ziel-Routine übergeben. Beim Einsprung in ein Hintergrund-ROM wird dieses so eingestellt, daß es auf die Basis des oberen ROM-Datenbereiches zeigt (siehe Abschnitt 9.4 und KL INIT BACK).

Die Ziel-Routine springt auf den Befehl zurück, welcher dem Inline Parameter unmittelbar folgt. Die ROM-Select und -State werden auf den Zustand vor dem Aufruf zurückgesetzt. Um dies möglich zu machen, werden Werte auf den Stack „gepushed“. Die Überwachung des Stack sollte nach einem FAR CALL-Befehl sorgsam geschehen. Der Stack verwendet 4 Bytes für ROM-Select-Bytes im Bereich von #FC... #FF und 6 Bytes für ROM-Select-Bytes im Bereich von #00... #FB.

Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

KL FAR ICALL
KL FAR PCHL
LOW JUMP (RST1)
SIDE CALL (RST2)

LOW: KL FAR PCHL #001B

Aufruf einer Subroutine im RAM oder irgendeinem ROM. C und HL beinhalten die „Far-Address“ zum Aufruf.

Aktion:

Der Far-Aufrufmechanismus ermöglicht den Aufruf von Subroutinen im RAM oder in jedem beliebigen ROM. Diese Routine übernimmt eine „Far-Address“, ruft die gegebene Routine auf und stellt den erforderlichen ROM-State und -Select ein.

Einsprung-Bedingungen:

HL beinhaltet die Adresse der aufzurufenden Routine.
C beinhaltet das ROM-Select-Byte.

Alle Register und Flags werden mit Ausnahme von IY (IY zeigt auf den oberen Datenbereich des Hintergrund-ROM's) unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

IY ist geschützt.
Alle anderen Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Das ROM-Select-Byte nimmt folgende Werte an:

#00...#FB:	Selektiere das gegebene ROM, schalte das obere frei und sperre das untere.
#FC:	Keine Änderung des ROM-Select, schalte das obere und das untere frei.
#FD:	Keine Änderung des ROM-Select, schalte das obere frei und sperre das untere.
#FE:	Keine Änderung des ROM-Select, sperre das obere und schalte das untere frei.
#FF:	Keine Änderung des ROM-Select, sperre das obere und untere ROM.

Register werden mit Ausnahme des IY-Index-Registers unberührt an die Ziel-Routine übergeben. Beim Einsprung in ein Hintergrund-ROM wird dieses so eingestellt, daß es auf die Basis des oberen ROM-Daten-Bereiches zeigt (siehe Abschnitt 9.4. und KL INIT BACK).

Wenn die Ziel-Routine zurückkehrt, werden der ROM-State und -Select auf ihren ursprünglichen Zustand vor dem Aufruf rückgesetzt. Um dies zu ermöglichen, werden Werte auf den Stack „gepushed". Die Überwachung des Stack sollte nach dem Aufruf dieser Routine sorgsam geschehen. Der Stack verwendet 4 Bytes für ROM-Select-Bytes im Bereich von #FC...#FF und 6 Bytes für ROM-Select-Bytes im Bereich von #00...#FB.

Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

FAR CALL (RST3)
KL FAR ICALL
KL LOW PCHL
KL SIDE PCHL

LOW: PCHL INSTRUCTION #001E

Sprung auf die Adresse in HL.

Aktion:

Der Eintrag enthält einen JP (HL)-Befehl (oder PCHL in einigen Assembler-Dialekten).

Einsprung-Bedingungen:

HL beinhaltet die Adresse zum Sprung.
Alle Register und Flags werden unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Der Aufruf von PCHL INSTRUCTION ist eine brauchbare Methode, um eine Routine aufzurufen, deren Adresse aus einer Tabelle entnommen oder anderweitig zur Laufzeit festgelegt wurde.

Verwandte Einsprünge:

KL FAR PCHL
KL LOW PCHL
KL SIDE PCHL
PCBC INSTRUCTION
PCDE INSTRUCTION

LOW: RAM LAM RST4 #0020

LD A, (HL) mit gesperrten ROM's.

Aktion:

RST 4 dient der Erweiterung des Befehlssatzes. Er ist ein Äquivalent zum LD A, (HL)-Befehl, mit der Ausnahme, daß er immer unabhängig von der Freischaltung oder Sperrung der ROM's aus dem RAM liest.

Einsprung-Bedingungen:

HL beinhaltet die Adresse des zu lesenden Speicherplatzes.

Aussprung-Bedingungen:

A beinhaltet den vom gegebenen Speicherplatz gelesenen Wert.
Alle anderen Register und Flags unverändert.

Anmerkungen:

Das Schreiben auf einen Speicherplatz beschreibt immer das RAM, auch wenn sich der Speicherplatz in einem der freigeschalteten ROM-Bereiche befindet. Der RAM LAM (RST4) ist ein Lese-Äquivalent.

Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

KL LDDR
KL LDIR

LOW: KL FAR ICALL #0023

Aufruf einer Subroutine im RAM oder irgendeinem ROM, HL zeigt auf die „Far-Address".

Aktion:

Der Far-Aufrufmechanismus ermöglicht den Aufruf von Subroutinen im RAM oder in jedem beliebigen ROM. Diese Routine übernimmt die Adresse einer „Far-Adress", ruft die gegebene Routine auf und stellt den erforderlichen ROM-State und -Select ein.

Einsprung-Bedingungen:

HL beinhaltet die Adresse der „Far-Address" zum Aufruf.
Alle Register und Flags werden mit Ausnahme von IY (IY zeigt auf den oberen Datenbereich des Hintergrund-ROM's) unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

IY ist unverändert.
Alle anderen Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Der übergebene Parameter ist die Adresse einer 3 Byte langen „Far-Address". Diese ist wie folgt belegt:

Bytes 0...1: Adresse der aufzurufenden Routine:
Byte 2: ROM-Select-Byte; kann folgende Werte annehmen:

- #00...#FB: Selektiere das gegebene ROM, schalte das obere frei und sperre das untere.
- #FC: Keine Änderung des ROM-Select, schalte das obere und untere frei.
- #FD: Keine Änderung des ROM-Select, schalte das obere frei und sperre das untere.
- #FE: Keine Änderung des ROM-Select, sperre das obere und schalte das untere frei.
- #FF: Keine Änderung des ROM-Select, sperre das obere und untere ROM.

Register werden mit Ausnahme des IY-Index-Registers unberührt an die Ziel-Routine übergeben. Beim Einsprung in ein Hintergrund-ROM wird dieses so eingestellt, daß es auf die Basis des oberen ROM-Daten-Bereiches zeigt (siehe Abschnitt 9.4. und KL INIT BACK).

Wenn die Ziel-Routine zurückkehrt, werden der ROM-State und -Select auf ihren ursprünglichen Zustand vor dem Aufruf rückgesetzt. Hierzu müssen Werte auf den Stack „gepushed" werden. Die Überwachung des Stack sollte nach dem Aufruf dieser Routine sorgsam geschehen. Der Stack verwendet 4 Bytes für ROM-Select-Bytes im Bereich von #FC...#FF und 6 Bytes für ROM-Select-Bytes im Bereich von #00...#FB.

Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

KL FAR CALL
KL FAR PCHL

LOW: FIRM JUMP RST5 #0028

Sprung in ein unteres ROM, nimmt eine Inline-Adresse als Sprungziel.

Aktion:

RST 5 dient der Erweiterung des Befehlssatzes. Er ist eine erweiterte Form des Sprung-Befehls und springt in Routinen im unteren ROM oder in den zentralen 32KB RAM. Der Restart ist mit der Adresse der anzuspringenden Routine versehen.

Einsprung-Bedingungen:

Alle Register und Flags werden unberührt an die Ziel-Routine übergeben.

Aussprung-Bedingungen:

Alle Register und Flags werden durch die Ziel-Routine gesetzt.

Anmerkungen:

Das untere ROM wird vor dem Einsprung freigeschaltet und beim Routinen-Rücksprung gesperrt (d.h. der Ursprungs-Zustand nicht wiederhergestellt). Weder der obere ROM-State, noch der ROM-Select werden geändert. Zwei Bytes werden auf den Stack „gepushed". Die Überwachung des Stack sollte sorgsam geschehen (z.B. beim Aufsuchen der Adresse der Inline-Parameter).

Es wird vorausgesetzt, daß das Sprungziel eine Routine ist, welche auf die gewohnte Weise zurückkehrt. Der Restart-Befehl selbst springt nicht zurück. Der Wert des oberen Stack-Endes (Top of Stack) muß daher bei der Ausführung eines FIRM JUMP eine Rückkehr-Adresse sein.

Der FIRM JUMP (RST5) kann das erste Byte eines JP-Befehls (JUMP), insbesondere in Sprungtabellen (ähnlich einem LOW JUMP), ersetzen. Ein FIRM JUMP ist schneller als ein LOW JUMP, während der LOW JUMP flexibler in der Behandlung von ROM-States ist.

Unterbrechungen sind freigeschaltet.

Verwandte Einsprünge:

LOW JUMP (RST1)

LOW: USER RESTART RST6 #0030

RST-Befehl ohne Zuordnung.

Aktion:

Die acht Bytes zwischen #0030 und inklusive #0037 können frei belegt werden.

Einsprung-Bedingungen:

Unbekannt.

Aussprung-Bedingungen:

Unbekannt.

Anmerkungen:

Wenn das untere ROM während der Ausführung eines RST 6-Befehls gesperrt ist, werden die auf den Speicherplätzen #0030 bis #0037 befindlichen Befehle normal ausgeführt.

Wenn das untere ROM während der Ausführung eines RST 6-Befehls freigeschaltet ist, sperrt die Firmware das untere ROM und springt auf Speicherplatz #0030, um die durch den Benutzer angelegten Befehle auszuführen.

Im allgemeinen ist das untere ROM gesperrt, es sei denn, die Firmware ist aktiv. Da es in der Firmware keine RST 6-Befehle gibt, treten bei der Ausführung eines RST 6-Befehles keine Probleme mit dem ROM-State auf. Um jedoch allen Eventualitäten vorzubeugen, wird der ROM-State vor dem Sperren des unteren ROM's in Speicherplatz #002B gesichert, sofern das untere ROM während der Ausführung des Restarts freigeschaltet war.

Speicherplatz #002B bleibt unberührt, wenn das untere ROM gesperrt vorgefunden wurde.

Der gespeicherte Wert kann an die Routine KL ROM RESTORE zum erneuten Freischalten des ROM's übergeben werden (obgleich KL L ROM ENABLE den gleichen Effekt hat).

Der Anwender kann nach der Ausführung des Restarts erkennen, ob das untere ROM freigeschaltet war. Dies ist beim Anlegen des RST 6-Bereiches (Speicherplatz #002B auf Null gesetzt) und nach Ausführung jedes Restarts möglich. Speicherplatz #002B ist Null beim Einsprung in den RST 6-Code, wenn das untere ROM gesperrt, und nicht Null, wenn das untere ROM freigeschaltet war.

Die beim Power-up eingestellte Standard-Aktion für den RST 6 ist die Ausführung eines RST 0, d.h. eines System-Resets.

Verwandte Einsprünge:

Keine.

LOW: INTERRUPT ENTRY RST7 #0038

Einsprung für Hardware-Unterbrechung.

Aktion:

Der Z80 arbeitet im Unterbrechungs-Modus 1, der normale Unterbrechungen wie RST 7-Befehle behandelt. Der Unterbrechungs-Handler der Firmware überwacht die eingebaute Zeit-Unterbrechung. Die durch Erweiterungs-Hardware erzeugten externen Unterbrechungen werden an die Anwender-Software übergeben.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

Alle Register und Flags sind unverändert.

Anmerkungen:

Der Anwender muß den RST 7 nicht verwenden, da er für die Bearbeitung von Unterbrechungen vorgesehen ist. Wenn die Unterbrechung von einer externen Quelle kommt, wird die durch den Anwender bereitgestellte Unterbrechungs-Routine EXT INTERRUPT aufgerufen.

In Abschnitt 10 ist eine ausführliche Beschreibung der Unterbrechungen zu finden. Wenn es unbedingt notwendig ist, kann der Anwender diesen Bereich (#0038 bis inklusive #003A) zum Abfangen von Unterbrechungen belegen (siehe Anhang XI).

Verwandte Einsprünge:

EXT INTERRUPT

LOW: EXT INTERRUPT #003B

Routine für externe Unterbrechungen.

Aktion:

Die fünf Bytes von #003B bis inklusive #003F müssen durch den Anwender belegt sein, wenn externe Unterbrechungen verarbeitet werden sollen. Beim Erkennen einer externen Unterbrechung durch den Unterbrechungs-Handler der Firmware wird das untere ROM gesperrt und der Code auf Adresse #003B aufgerufen.

Einsprung-Bedingungen:

keine.

Aussprung-Bedingungen:

AF, BC, DE, und HL zerstört.
Alle anderen Register unverändert.

Anmerkungen:

Beim Aufruf dieser Routine werden Unterbrechungen gesperrt. Der Anwender darf auf keinen Fall Unterbrechungen freischalten oder den zweiten Registersatz verwenden. Vor dem Routinen-Rücksprung muß die Unterbrechungs-Quelle gelöscht werden.

In Abschnitt 10.2 werden externe Unterbrechungen ausführlich beschrieben.

Bei der Einstellung einer Unterbrechungs-Routine sollte der momentane Inhalt von #003B bis #003F irgendwohin kopiert werden, bevor er ersetzt wird. Wenn eine aufgerufene Routine feststellt, daß ihre Hardware für die Unterbrechung nicht verantwortlich ist, sollte sie in die Kopie der vorhergehenden Routine für externe Unterbrechungen springen (dessen Hardware verantwortlich sein könnte).

Eine Unterbrechungs-Routine sollte die Unterbrechung so schnell wie möglich löschen und mit minimalem Aufwand bearbeiten. Während der Unterbrechungs-Bearbeitung werden keine weiteren Unterbrechungen erkannt. Erfordert eine Unterbrechung hohen Bearbeitungsaufwand, so sollte sie in ein Ereignis übersetzt werden. Dadurch wird das System im Unterbrechungsbereich nicht stärker verlangsamt als notwendig (siehe Abschnitt 10.3).

Die Unterbrechungs-Routine muß im RAM auf Adressen unterhalb #C000 liegen, da die ROM-Freischalt-/-Sperr-Routinen aus der Unterbrechungsebene nicht aufgerufen werden können.

Die Standard-Routine für externe Unterbrechungen springt lediglich zurück, d.h. die Unterbrechung wird nicht gelöscht und demzufolge solange wiederholt, wie Unterbrechungen erneut freigeschaltet werden. Dadurch blockiert sich die Maschine selbst.

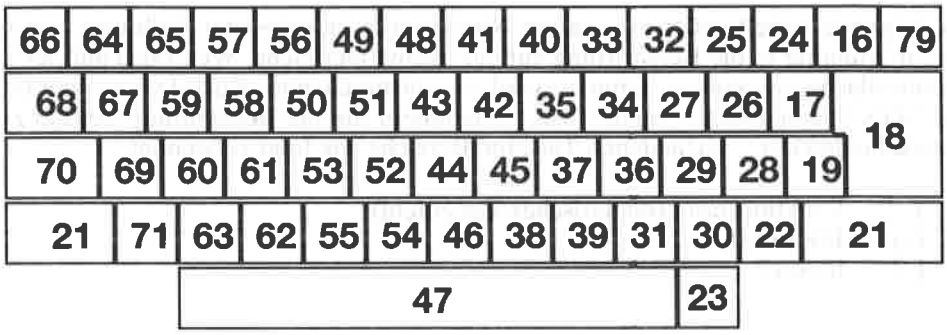
Verwandte Einsprünge:

INTERRUPT ENTRY
KL EVENT

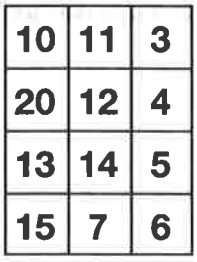
ANHANG I
Tasten-Numerierung

Auf die verschiedenen Tabellen in der Tastatur-Verwaltung, wie z.B. die Übersetzungs- oder Wiederholungs-Tasten-Tabelle, wird über die Tasten-Nummer zugegriffen. Die Numerierung der Tasten (und Joysticks) ist in den folgenden Diagrammen dargestellt:

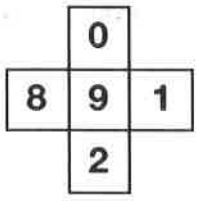
Haupt-Tastatur

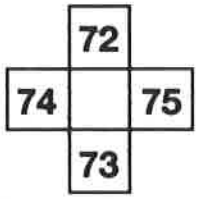


Funktions-/numerisches Tastenfeld

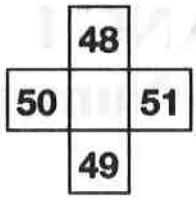


Cursor-Tasten



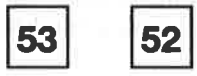


Joystick 0



Joystick 1

Feuer 1 **Feuer 2**



Feuer 1 **Feuer 2**

Man beachte, daß Joystick 1 die Tasten 48...53 überlagert und von ihnen nicht zu unterscheiden ist.

Die folgende Tabelle übersetzt Tasten-Nummern in umgekehrter Richtung; von der Tasten-Nummer in die Beschriftung auf der Tastenoberfläche. Wenn sich auf der Tastenoberfläche ein Symbol befindet, wird ein entsprechendes Kürzel verwendet (z.B. RECHTS für die rechte Cursor-Taste). Klammern um die Beschriftung werden zum Unterscheiden der verschiedenen Tastatur-Bereiche wie folgt verwendet:

{..} Funktionstaste (numerisches Tastenfeld)
(..) Joystick 0
[..] Joystick 1.

	0	1	2	3	4	5	6	7
0	AUF	RECHTS	AB	{9}	{6}	{3}	{ENTER}	{.}
8	LINKS	COPY	{7}	{8}	{5}	{1}	{2}	{0}
16	CLR	[	ENTER	]	{4}	SHIFT	\	CTRL
24	↑	–	@	P	;	:	/	.
32	0	9	O	I	L	K	M	,
40	8	7	U	Y	H	J	N	SPACE
48	6	5	R	T	G	F	B	V
	[AUF]	[AB]	[LINKS]	[RECHTS]	[FEUER 2]	[FEUER 1]	[SPARE]	
56	4	3	E	W	S	D	C	X
64	1	2	ESC	Q	TAB	A	CAPS	Z
72	(AUF)	(AB)	(LINKS)	(RECHTS)	(FEUER 2)	(FEUER 1)	(SPARE)	DEL

ANHANG II
Tastenübersetzungs-Tabellen

In Abschnitt 3 und insbesondere 3.2 befindet sich eine Beschreibung der Tasten-Übersetzung. Auch Anhang I ist in dieser Hinsicht von Interesse; dort ist das Tasten-Numerierungs-Schema zu finden.

Es werden drei Tastatur-Übersetzungs-Tabellen verwendet. Sie übersetzen Tasten in ihre zugehörigen Zeichen bzw. Werte. Eine Tabelle steht für die Übersetzung von Tasten bereit, wenn die Control-Taste gedrückt ist, eine andere für gedrückte Shift- oder Shift-Lock-Tasten und nicht betätigte Control-Taste und eine weitere für nicht gedrückte Shift- und Control-Tasten.

Das folgende Diagramm zeigt die Standard-Übersetzungs-Tabelle. Wo es möglich war, wurde das korrekte Zeichen auf der Taste angegeben. Der tatsächliche Wert für jedes dieser Zeichen ist in Anhang VI unter Zeichensatz zu finden. In den Fällen, in denen die Taste den Wert eines nicht darstellbaren ASCII-Zeichens erzeugt, werden die Abkürzungen der folgenden Tabelle verwendet. Die Standard-Einstellungen der Erweiterungs-Werte sind in Anhang IV ausgeführt.

Zeichen und Codes:

NUL	#00	ASCII Steuercode.
SOH	#01	ASCII Steuercode.
STX	#02	ASCII Steuercode.
ETX	#03	ASCII Steuercode.
EOT	#04	ASCII Steuercode.
ENQ	#05	ASCII Steuercode.
ACK	#06	ASCII Steuercode.
BEL	#07	ASCII Steuercode.
BS	#08	ASCII Steuercode.
HT	#09	ASCII Steuercode.
LF	#0A	ASCII Steuercode.
VT	#0B	ASCII Steuercode.
FF	#0C	ASCII Steuercode.
CR	#0D	ASCII Steuercode.
SO	#0E	ASCII Steuercode.
SI	#0F	ASCII Steuercode.

DLE	#10	ASCII Steuercode.
DC1	#11	ASCII Steuercode.
DC2	#12	ASCII Steuercode.
DC3	#13	ASCII Steuercode.
DC4	#14	ASCII Steuercode.
NAK	#15	ASCII Steuercode.
SYN	#16	ASCII Steuercode.
ETB	#17	ASCII Steuercode.
CAN	#18	ASCII Steuercode.
EM	#19	ASCII Steuercode.
SUB	#1A	ASCII Steuercode.
ESC	#1B	ASCII Steuercode.
FS	#1C	ASCII Steuercode.
GS	#1D	ASCII Steuercode.
RS	#1E	ASCII Steuercode.
US	#1F	ASCII Steuercode.
SPACE	#20	ASCII Leerzeichen.
UP	#5E	Aufwärts-Pfeil.
DEL	#7F	ASCII-Code.
LB	#A3	Pfund-Zeichen.

Erweiterungs-Werte:

F0	#80	Funktionstaste 0.
F1	#81	Funktionstaste 1.
F2	#82	Funktionstaste 2.
F3	#83	Funktionstaste 3.
F4	#84	Funktionstaste 4.
F5	#85	Funktionstaste 5.
F6	#86	Funktionstaste 6.
F7	#87	Funktionstaste 7.
F8	#88	Funktionstaste 8.
F9	#89	Funktionstaste 9.

F	#8A	Funktionstaste Stop (Full Stop).
FEN	#8B	Funktionstaste Enter ohne Control.
FRUN	#8C	Funktionstaste Enter mit Control.

Editier- und Cursor-Codes:

COPY	#E0	Copy-Taste.
INS	#E1	Einfügen-/Überschreiben-Taste.
WUP	#F0	Schreibe Cursor aufwärts.
WDN	#F1	Schreibe Cursor abwärts.
WLT	#F2	Schreibe Cursor links.
WRT	#F3	Schreibe Cursor rechts.
RUP	#F4	Lese Cursor aufwärts.
RDN	#F5	Lese Cursor abwärts.
RLT	#F6	Lese Cursor links.
RRT	#F7	Lese Cursor rechts.
BEG	#F8	Schreibe Cursor zum Textanfang.
END	#F9	Schreibe Cursor zum Textende.
STA	#FA	Schreibe Cursor zum Zeilenanfang.
FIN	#FB	Schreibe Cursor zum Zeilenende.

System-Werte:

BRK	#FC	Abbruch-Tasten-Wert.
CAPS	#FD	Caps-Lock-Wert.
SHIFT	#FE	Shift-Lock-Wert.
	#FF	Ignoriere.

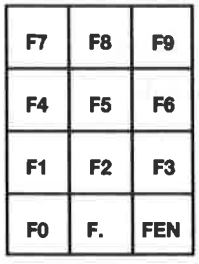
Tasten, die in den folgenden Diagrammen nicht aufgeführt sind, erzeugen den System-Wert #FF.

Normale Übersetzungs-Tabelle

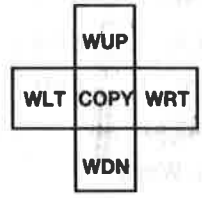
Das folgende Diagramm zeigt die Übersetzung, wenn weder Shift noch Control betätigt ist.

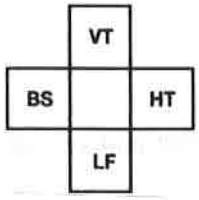
Haupt-Tastatur

Funktions-/numerisches Tastenfeld

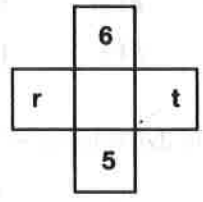


Cursor-Tasten

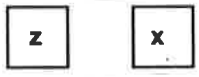




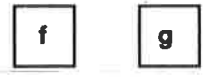
Joystick 0



Joystick 1



Feuer 1 **Feuer 2**

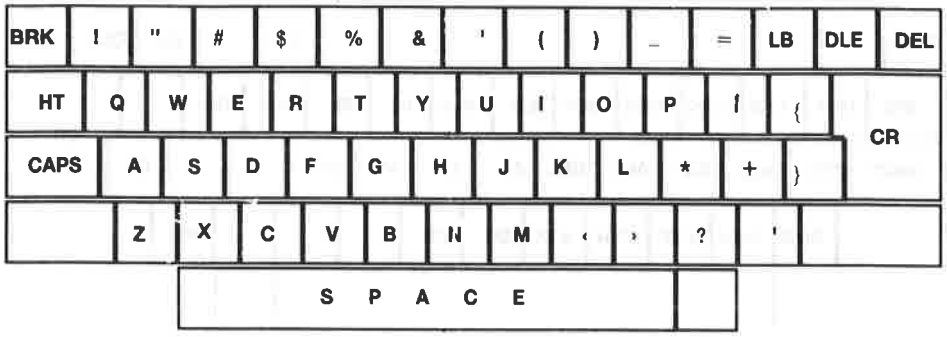


Feuer 1 **Feuer 2**

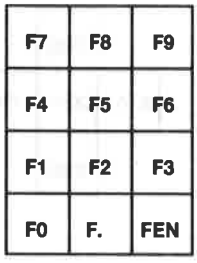
Shift-Übersetzungs-Tabelle

Das folgende Diagramm zeigt die Übersetzung, wenn entweder Shift oder Shift-Lock betätigt, und Control nicht betätigt ist.

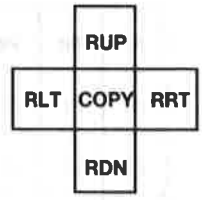
Haupt-Tastatur

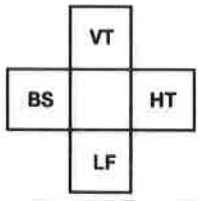


Funktions-/numerisches Tastenfeld



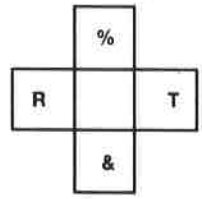
Cursor-Tasten



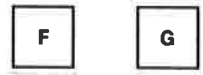


Joystick 0

Feuer 1 **Feuer 2**



Joystick 1

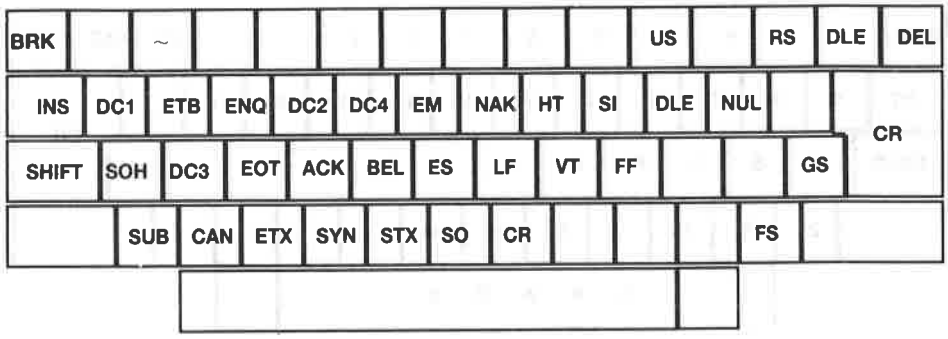


Feuer 1 **Feuer 2**

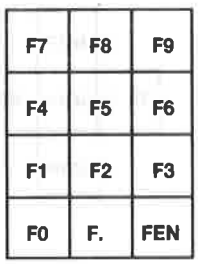
Control--Übersetzungs-Tabelle

Das folgende Diagramm zeigt die Übersetzung, wenn die Control-Taste betätigt ist.

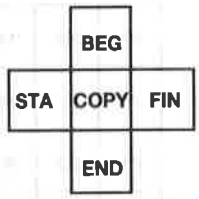
Haupt-Tastatur

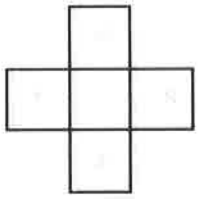


Funktions-/numerisches Tastenfeld

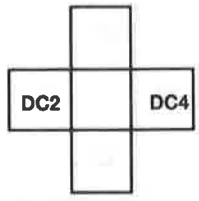


Cursor-Tasten





Joystick 0



Joystick 1

Feuer 1 **Feuer 2**

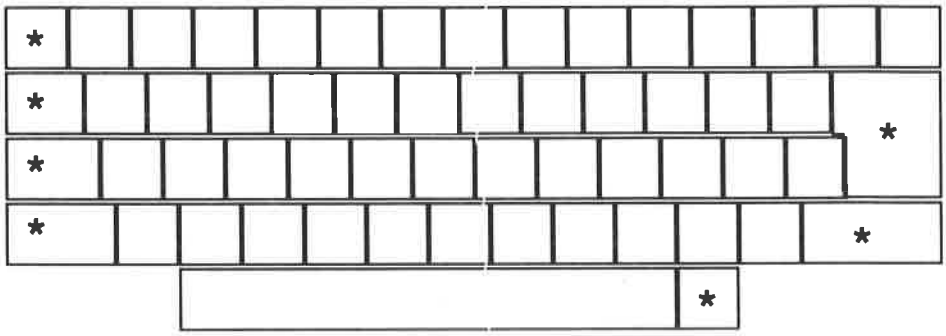
Feuer 1 **Feuer 2**

ANHANG III
Daueranschlag

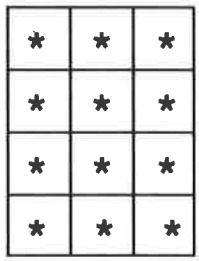
Welche Tasten wiederholbar sein dürfen, kann durch den Anwender bestimmt werden. Abschnitt 3 und insbesondere 3.5 geben eine vollständige Beschreibung der Tasten-Wiederholung. Anhang I zeigt das Tasten-Numerierungs-Schema.

Die Standard-Tabelle für Wiederholungs-Tasten (Daueranschlag) wird im folgenden Diagramm dargestellt. Nicht wiederholbare Tasten sind mit einem Sternchen gekennzeichnet.

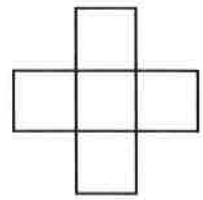
Haupt-Tastatur

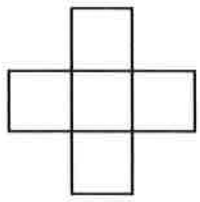


Funktions-/numerisches Tastenfeld

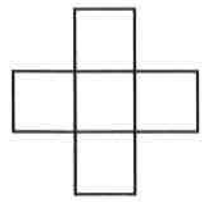


Cursor-Tasten





Joystick 0



Joystick 1

Feuer 1 * * **Feuer 2** **Feuer 1** **Feuer 2**

ANHANG IV
Funktions-Tasten und Erweiterungs-Strings

Funktions-Tasten sind in Abschnitt 3 und insbesondere 3.7 vollständig beschrieben. Die folgende Tabelle spezifiziert den Standard-String für jeden Erweiterungs-Wert und die Taste, welche dem Wert standardmäßig zugeordnet ist.

Wert	Tasten-Wert	Standard-String	Standard-Taste
0	#80	0	Funktionstaste 0.
1	#81	1	Funktionstaste 1.
2	#82	2	Funktionstaste 2.
3	#83	3	Funktionstaste 3.
4	#84	4	Funktionstaste 4.
5	#85	5	Funktionstaste 5.
6	#86	6	Funktionstaste 6.
7	#87	7	Funktionstaste 7.
8	#88	8	Funktionstaste 8.
9	#89	9	Funktionstaste 9.
10	#8A	.	Funktionstaste Stop (Full Stop).
11	#8B	CR	Funktionstaste Enter.
12	#8C	RUN" CR	Funktionstaste Enter mit Control.
13..31	#8D..#9F		Keine Funktion.

Die Werte 13..31 sind Leer-Strings und keiner Taste zugeordnet.

ANHANG V
Inks und Farben

Eine Beschreibung der Inks und Farben befindet sich in Abschnitt 6.2. Dieser Anhang zeigt die verfügbaren Farben und die Standard-Einstellungen der Inks.

Es gibt insgesamt 27 Farben. Das Bildschirm-Paket bezieht sich auf diese Farben über eine Nummer der Grauskala, deren Farbe 0 die dunkelste und Farbe 26 die hellste ist. Die Hardware benötigt diese Grauskala, um die Farben in den entsprechenden Hardware-Code zu übersetzen. Es ist sehr unwahrscheinlich, daß der Anwender jemals die Hardware-Nummern benötigt. Sie dienen daher nur zur Information.

Grauskala	Farbe	HW-Nummer
0	Schwarz	20
1	Blau	4
2	Hellblau	21
3	Rot	28
4	Magenta	24
5	Hellviolett	29
6	Hellrot	12
7	Purpur	5
8	Helles Magenta	13
9	Grün	22
10	Blaugrün	6
11	Himmelblau	23
12	Gelb	30
13	Weiß	0
14	Pastellblau	31
15	Orange	14
16	Rosa	7
17	Pastellmagenta	15
18	Hellgrün	18
19	Seegrün	2
20	Helles Blaugrün	19
21	Limonengrün	26
22	Pastellgrün	25
23	Pastellblaugrün	27
24	Hellgelb	10
25	Pastellgelb	3
26	Leuchtendweiß	11

Der Anwender kann die Farbe für die 16 Inks und den Bildschirmrand bestimmen. Die folgende Tabelle zeigt die Standard-Einstellung:

Ink	Farbe	Farbnummer
Bildschirmrand	Blau	(1/1)
0	Blau	(1/1)
1	Hellgelb	(24/24)
2	Helles Blaugrün	(20/20)
3	Hellrot	(6/6)
4	Leuchtendweiß	(26/26)
5	Schwarz	(0/0)
6	Hellblau	(2/2)
7	Helles Magenta	(8/8)
8	Blaugrün	(10/10)
9	Gelb	(12/12)
10	Pastellblau	(14/14)
11	Rosa	(16/16)
12	Hellgrün	(18/18)
13	Pastellgrün	(22/22)
14	Blinkendes Blau/Hellgelb	(1/24)
15	Blinkendes Himmelblau/Rosa	(11/16)

ANHANG VI
Darstellbarer Zeichensatz

Der darstellbare Zeichensatz besteht aus 256 Symbolen. Sie können alle ausgegeben werden, obwohl besondere Anstrengungen zur Darstellung der Zeichen 0...31, die meistens als Steuer-Code interpretiert werden, vonnöten sind. Der Anwender kann die Matrix für beliebig viele Zeichen selbst einstellen (siehe Abschnitt 4.6). Die folgende Liste stellt den Standard-Zeichensatz dar. Der Zeichensatz ist der Einfachheit halber in Bereiche eingeteilt:

0..31	(#00..#1F)	ASCII-Steuercode.
32..127	(#20..#7F)	ASCII-Zeichen.
128..143	(#80..#8F)	Block-Graphik.
144..159	(#90..#9F)	Linien-Graphik.
160..191	(#A0..#BF)	Weitere Zeichen.
192..255	(#C0..#FF)	Verschiedene Graphik-Symbole.

a. ASCII-Steuercode

0	#00	NUL	Quadrat.
1	#01	SOH	Liegendes L.
2	#02	STX	Liegendes T.
3	#03	ETX	Rückwärts-L.
4	#04	EOT	Blitz.
5	#05	ENQ	Quadrat mit diagonalem Kreuz.
6	#06	ACK	Vermerkzeichen.
7	#07	BEL	Wecker (Halbkreis mit Beinen).
8	#08	BS	Nach links weisender Pfeil.
9	#09	HT	Nach rechts weisender Pfeil.
10	#0A	LF	Abwärts weisender Pfeil.
11	#0B	VT	Aufwärts weisender Pfeil.
12	#0C	FF	Weihnachtsbaum (abwärts weisender Pfeil mit Schweif).
13	#0D	CR	Krummer, nach links weisender Pfeil.
14	#0E	SO	Kreis mit diagonalem Kreuz.
15	#0F	SI	Kreis mit zentralem Punkt.

16	#10	DLE	Quadrat mit horizontalem Balken.
17	#11	DC1	Kreis, drei Uhr.
18	#12	DC2	Kreis, halb vier Uhr.
19	#13	DC3	Kreis, halb zehn Uhr.
20	#14	DC4	Kreis, neun Uhr.
21	#15	NAK	Durchgestrichenes Vermerkzeichen.
22	#16	SYN	Rechteck-Wellenform.
23	#17	ETB	Seitliches T.
24	#18	CAN	Sanduhr.
25	#19	EM	Vertikaler Balken mit zentralem Punkt.
26	#1A	SUB	Umgekehrtes Fragezeichen.
27	#1B	ESC	Kreis mit horizontalem Balken.
28	#1C	FS	Quadrat, neun Uhr.
29	#1D	GS	Quadrat, halb zehn Uhr.
30	#1E	RS	Quadrat, halb vier Uhr.
31	#1F	US	Quadrat, drei Uhr.

b. ASCII-Zeichen

Die Zeichen 32...127 (#20...#7F) sind in der folgenden Tabelle aufgelistet. Sie sind der standardmäßige ASCII-Zeichensatz. Einige Zeichen sind schwierig darzustellen; sie sind in der Tabelle ausgelassen worden und finden sich am Schluß derselben.

	0 0	1 1	2 2	3 3	4 4	5 5	6 6	7 7	8 8	9 9	10 A	11 B	12 C	13 D	14 E	15 F
32 #20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48 #30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64 #40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80 #50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	-
96 #60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112 #70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

32	#20	Leerzeichen.
94	#5E	Aufwärts-Pfeil.
96	#60	Betonungszeichen.
127	#7F	Grobe Schattierung.

c. Block-Graphik

Die Zeichen 128...143 (#80...#8F) sind Bestandteil der Block-Graphik. Jedes Zeichen ist in vier Zellen unterteilt. Die Bits 0...3 der Zeichen-Nummer bestimmen die auszufüllenden Zellen. Wenn das entsprechende Bit gesetzt ist, wird die Zelle ausgefüllt, ansonsten bleibt sie leer. Die Zellen sind:

Bit 0	Bit 1
Bit 2	Bit 3

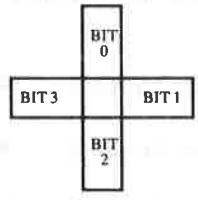
In der folgenden Tabelle sind die ausgefüllten Zellen mit einem ■ markiert und die leeren Zellen mit einem □.

128	#80	Block-Graphik	⊞
129	#81	Block-Graphik	▣
130	#82	Block-Graphik	▣
131	#83	Block-Graphik	▣
132	#84	Block-Graphik	▣
133	#85	Block-Graphik	▣
134	#86	Block-Graphik	▣
135	#87	Block-Graphik	▣
136	#88	Block-Graphik	▣
137	#89	Block-Graphik	▣
138	#8A	Block-Graphik	▣
139	#8B	Block-Graphik	▣

140	#8C	Block-Graphik	
141	#8D	Block-Graphik	
142	#8E	Block-Graphik	
143	#8D	Block-Graphik	

d. Linien-Graphik

Die Zeichen 144...159 (#90...#9F) sind Bestandteil der Linien-Graphik. Die Linien verbinden das Zentrum eines Zeichens mit dem Zentrum einer Kante. Jede Linie ist einem Bit der Zeichen-Nummer zugeordnet. Wenn das Bit gesetzt ist, wird die Linie dargestellt, andernfalls nicht. Der zentrale Block des Zeichens ist immer gesetzt. Die Linien sind den Bits folgendermaßen zugeordnet:



144	#90	Linien-Graphik	
145	#91	Linien-Graphik	
146	#92	Linien-Graphik	
147	#93	Linien-Graphik	
148	#94	Linien-Graphik	
149	#95	Linien-Graphik	
150	#96	Linien-Graphik	
151	#97	Linien-Graphik	
152	#98	Linien-Graphik	
153	#99	Linien-Graphik	
154	#9A	Linien-Graphik	
155	#9B	Linien-Graphik	
156	#9C	Linien-Graphik	
157	#9D	Linien-Graphik	
158	#9E	Linien-Graphik	
159	#9F	Linien-Graphik	

e. Weitere Zeichen

160	#A0	Circumflex
161	#A1	Accent
162	#A2	Umlaut
163	#A3	Pfund
164	#A4	Copyright
165	#A5	Pilcrow
166	#A6	Paragraph
167	#A7	Anführungszeichen
168	#A8	Ein Viertel
169	#A9	Ein Halb
170	#AA	Drei Viertel
171	#AB	Plus oder Minus
172	#AC	Division
173	#AD	Nicht
174	#AE	Umgekehrtes Frage-Zeichen
175	#AF	Umgekehrtes Ausrufe-Zeichen
176	#B0	Kleines Alpha
177	#B1	Kleines Beta
178	#B2	Kleines Gamma
179	#B3	Kleines Delta
180	#B4	Kleines Epsilon
181	#B5	Kleines Theta

182	#B6	Kleines Lambda
183	#B7	Kleines Mu
184	#B8	Kleines Pi
185	#B9	Kleines Sigma
186	#BA	Kleines Phi
187	#BB	Kleines Psi
188	#BC	Kleines Chi
189	#BD	Kleines Omega
190	#BE	Großes Sigma
191	#BF	Großes Omega

f. Verschiedene Graphik-Symbole

192	#C0	Diagonale Linie von oben nach links.
193	#C1	Diagonale Linie von oben nach rechts.
194	#C2	Diagonale Linie von unten nach rechts.
195	#C3	Diagonale Linie von unten nach links.
196	#C4	Diagonale Linie von oben nach links und rechts.
197	#C5	Diagonale Linie von rechts nach oben und unten.
198	#C6	Diagonale Linie von unten nach rechts und links.
199	#C7	Diagonale Linie von links nach oben und unten.
200	#C8	Diagonale Linie von oben nach links und unten nach rechts.
201	#C9	Diagonale Linie von oben nach rechts und unten nach links.

202	#CA	Viereck mit allen Kanten verbunden.
203	#CB	Beide Haupt-Diagonalen (großex X).
204	#CC	Vorwärts-Haupt-Diagonale (Schrägstrich, Slash).
205	#CD	Rückwärts-Haupt-Diagonale (Schrägstrich, Backslash).
206	#CE	Großkariertes Muster.
207	#CF	Kleinkariertes Muster.
208	#D0	Linie an der oberen Kante.
209	#D1	Linie an der rechten Kante.
210	#D2	Linie an der unteren Kante.
211	#D3	Linie an der linken Kante.
212	#D4	Dreieck in oberer linker Ecke.
213	#D5	Dreieck in oberer rechter Ecke.
214	#D6	Dreieck in unterer rechter Ecke.
215	#D7	Dreieck in unterer linker Ecke.
216	#D8	Kleinkariertes Muster in oberer Hälfte.
217	#D9	Kleinkariertes Muster in rechter Hälfte.
218	#DA	Kleinkariertes Muster in unterer Hälfte.
219	#DB	Kleinkariertes Muster in linker Hälfte.
220	#DC	Kleinkariertes Dreieck in oberer linker Ecke.
221	#DD	Kleinkariertes Dreieck in oberer rechter Ecke.
222	#DE	Kleinkariertes Dreieck in unterer rechter Ecke.
223	#DF	Kleinkariertes Dreieck in unterer linker Ecke.
224	#E0	Glückliches Gesicht.

225	#E1	Trauriges Gesicht.
226	#E2	Kreuz
227	#E3	Karo.
228	#E4	Herz.
229	#E5	Pik.
230	#E6	Leerer Kreis.
231	#E7	Voller Kreis.
232	#E8	Leeres Quadrat.
233	#E9	Volles Quadrat.
234	#EA	Symbol männlich (Mars).
235	#EB	Symbol weiblich (Venus).
236	#EC	Viertelnote.
237	#ED	Achtelnote.
238	#EE	Stern.
239	#EF	Rakete.
240	#F0	Nach oben weisender Pfeil-Kopf.
241	#F1	Nach unten weisender Pfeil-Kopf.
242	#F2	Nach links weisender Pfeil-Kopf.
243	#F3	Nach rechts weisender Pfeil-Kopf.
244	#F4	Nach oben weisendes Dreieck.
245	#F5	Nach unten weisendes Dreieck.
246	#F6	Nach links weisendes Dreieck.
247	#F7	Nach rechts weisendes Dreieck.

248	#F8	Tanzende Person, stehend.
249	#F9	Tanzende Person, Grätschstellung.
250	#FA	Tanzende Person, linkes Bein ausgestreckt.
251	#FB	Tanzende Person, rechtes Bein ausgestreckt.
252	#FC	Bombe.
253	#FD	Pilz (Wolke).
254	#FE	Pfeil auf und ab.
255	#FF	Pfeil rechts und links.

ANHANG VII
Steuer-Codes des Text-VDU

An die Haupt-Ausgabe-Routine (TXT OUTPUT) des Text-VDU übergebene Zeichen im Bereich von 0...31 stellen kein Zeichen auf dem Bildschirm dar, sondern werden als Steuer-Code interpretiert. Diese Codes können die Bedeutung der folgenden Zeichen verändern.

Alle Steuer-Codes wirken, wenn nicht anders vermerkt, auf den momentan selektierten Kanal. Die Einstellung des Pen (Code 15) setzt z.B. die Text-Pen-Ink für den momentan selektierten Kanal, während die Einstellung einer Ink-Farbe (Code 28) alle Kanäle (und den Graphik-VDU) beeinflußt.

Bestimmte Codes zwingen von ihrer Ausführung die momentane Position (die Cursor-Position) auf eine zulässige Position innerhalb des momentanen Fensters (nähere Erklärungen hierzu in Abschnitt 4.5). Der Cursor darf sich in einer unzulässigen Position befinden.

Die folgende Tabelle spezifiziert die Standard-Aktionen der Steuer-Codes. Durch die Änderung der Einträge in der Steuer-Code-Tabelle kann die Wirkungsweise dieser Codes gezielt verändert werden (siehe Abschnitt 4.7).

Code	Name	Parameter	Aktion
0	NUL	0	Keinen Einfluß.
1	SOH	1	Stelle das durch den Parameter gegebene Zeichen dar (siehe TXT WR CHAR). Dies ermöglicht den Ausdruck der Zeichen 0...31.
2	STX	0	Sperre das Cursor-Symbol (siehe TXT CUR DISABLE). Umkehrung des Effektes von ETX (Code 3).
3	ETX	0	Freischalten des Cursor-Symbols (siehe TXT CUR ENABLE). Umkehrung des Effektes von STX (Code 2).
4	EOT	1	Setze den durch den Parameter gegebenen Bildschirm-Modus (siehe SCR SET MODE). Der Parameter wird als MOD 4 übernommen und der Wert 3 ignoriert: 0 setzt Modus 0 (160 x 200). 1 setzt Modus 1 (320 x 200). 2 setzt Modus 2 (640 x 200).

Code	Name	Parameter	Aktion
5	ENQ	1	Stelle das durch den Parameter gegebene Zeichen unter Verwendung des Graphik-VDU dar, als wäre der Schreib-Modus für Graphik-Zeichen aktiv (siehe TXT SET GRAPHIC und GRA WR CHAR).
6	ACK	0	Freischalten des VDU (siehe TXT VDU ENABLE). Kehrt den Effekt von NAK (Code 21) um.
7	BEL	0	Kurzer Pieps-Ton. Die Ton-Warteschlangen werden geleert.
8	BS	0	Legalisiere die momentane Position und verschiebe ein Zeichen nach links.
9	TAB	0	Legalisiere die momentane Position und verschiebe ein Zeichen nach rechts.
10	LF	0	Legalisiere die momentane Position und verschiebe ein Zeichen nach unten.
11	VT	0	Legalisiere die momentane Position und verschiebe ein Zeichen nach oben.
12	FF	0	Lösche das momentane Fenster und bewege die momentane Position in die obere linke Ecke (siehe TXT CLEAR WINDOW).
13	CR	0	Legalisiere die momentane Position und bewege sie in der momentanen Zeile zur linken Kante des Fensters (siehe TXT SET COLUMN).
14	SO	1	Setze die Paper-Ink auf die durch den Parameter gegebene Ink (siehe TXT SET PAPER). Parameter wird als MOD16 übernommen.
15	SI	1	Setze die Pen-Ink auf die durch den Parameter gegeben Ink (siehe TXT SET PEN). Parameter wird als MOD16 übernommen.
16	DLE	0	Legalisiere die momentane Position und lösche sie auf die momentane Paper-Ink.
17	DC1	0	Legalisiere die momentane Position und lösche von der linken Fensterkante bis zur momentanen Position (inkl.) Die hiervon betroffenen Zellen werden auf die momentane Paper-Ink gesetzt.

Code	Name	Parameter	Aktion
18	DC2	0	Legalisiere die momentane Position und lösche von der rechten Fensterkante bis zur momentanen Position (inkl.). Die hiervon betroffenen Zellen werden auf die momentane Paper-Ink gesetzt.
19	DC3	0	Legalisiere die momentane Position und lösche vom Fensteranfang bis zur momentanen Position (inkl.). Die hiervon betroffenen Zellen werden auf die momentane Paper-Ink gesetzt.
20	DC4	0	Legalisiere die momentane Position und lösche von ihr bis zum Fensterende (inkl.). Die hiervon betroffenen Zellen werden auf die momentane Paper-Ink gesetzt.
21	NAK	0	Sperre den VDU (siehe TXT VDU DISABLE). Kehrt den Effekt von ACK (Code 6) um.
22	SYN	1	Setze den Zeichen-Schreib-Modus des Parameters (siehe TXT SET BACK). Der Parameter wird als MOD 2 übernommen: 0 setzt den Überschreib-Modus (der Standard-Modus). 1 setzt den Transparent-Modus.
23	ETB	1	Setze den Graphik-VDU-Schreib-Modus des Parameters (siehe SCR SET ACCESS). Der Parameter wird als MOD4 übernommen: 0 setzt den WRITE-Modus (der Standard-Modus). 1 setzt den XOR-Modus. 2 Setzt den AND-Modus. 3 Setzt den OR-Modus.
24	CAN	0	Tausche die momentanen Pen- und Paper-Inks (siehe TXT INVERSE).
25	EM	9	Setze die Matrix für ein Zeichen (siehe TXT SET MATRIX). Der erste Parameter spezifiziert das einzustellende Zeichen. Die nächsten Parameter sind die Matrix des Zeichens (von oben nach unten). Wenn das Zeichen nicht anwenderdefinierbar ist, wird nichts unternommen.

Code	Name	Parameter	Aktion
26	SUB	4	Setze die Grenzen des Text-Fensters (siehe TXT WIN ENABLE). Die ersten beiden Parameter spezifizieren die linke und rechte Spalte des Fensters (der kleinere ist die linke Spalte); die letzten beiden Parameter spezifizieren die obere und untere Reihe des Fensters (der kleinere ist die obere Reihe).
27	ESC	0	Keinen Einfluß – verfügbar für den Anwender.
28	FS	3	Setze die Farbe für die Darstellung einer Ink (siehe SCR SET INK). Der erste Parameter wird als MOD16 übernommen und spezifiziert die einzustellende Ink. Der zweite und dritte Parameter wird als MOD 32 übernommen und spezifiziert die beiden Farben der Ink.
29	GS	2	Setze die für den Bildschirmrand gültige Farbe (siehe SCR SET BORDER). Die beiden Parameter werden als MOD32 übernommen und spezifizieren die beiden Farben des Bildschirmrands.
30	RS	0	Bewege die momentane Position in die obere linke Ecke des Fensters (siehe TXT SET CURSOR).
31	US	2	Bewege die momentane Position auf die gegebene Position im momentanen Fenster (siehe TXT SET CURSOR). Der erste Parameter spezifiziert die Spalte, der zweite Parameter die Reihe, auf die bewegt wird (Reihe 1, Spalte 1 ist die oberste linke Ecke des Fensters).

ANHANG VIII
Noten und Ton-Perioden

Die folgende Tabelle gibt die empfohlenen Ton-Perioden für die wohltemperierte Stimmung des gesamten, 8 Oktaven umfassenden Bereiches an. Diese Periode wird folgendermaßen aus der Noten-Frequenz berechnet (die Periode ist in 8-Mikrosekunden-Einheiten gegeben):

Periode = 125000/Frequenz

Die Frequenz wird für jede Note aus dem internationalen Kammerton A wie folgt berechnet:

Frequenz = 440* (2↑(Oktave + (N-10)/12))

mit:

Oktave — ist die Oktaven-Nummer. 0 ist die Oktave des Kammertons A (und des mittleren C), −1 ist die Oktave unterhalb, +1 ist die Oktave oberhalb etc.

N — ist die Noten-Nummer. 1 ist C, 2 ist C#, 3 ist D etc.

Die Periode ist ein Integer-Wert, wodurch die Frequenz der gespielten Note nicht der exakten Frequenz des Tones entspricht. Der relative Fehler ist in der unteren Tabelle angegeben. Er wird folgendermaßen berechnet:

Fehler = (Noten-Frequenz – tatsächliche Freuquenz)/Noten-Frequenz

Note	Frequenz	Periode		Fehler	Oktave-3
C	32.703	3822	#0EEE	-0.007%	
C#	34.648	3608	#0E18	+0.007%	
D	36.708	3405	#0D4D	-0.007%	
D#	38.891	3214	#0C8E	-0.004%	
E	41.203	3034	#0BDA	+0.009%	
F	43.654	2863	#0B2F	-0.016%	
F#	46.249	2703	#0A8F	+0.009%	
G	48.999	2551	#09F7	-0.002%	
G#	51.913	2408	#0968	+0.005%	
A	55.000	2273	#08E1	+0.012%	
A#	58.270	2145	#0861	-0.008%	
B	61.735	2025	#07E9	+0.011%	

Note	Frequenz	Periode		Fehler	Oktave-2
C	65.406	1911	#0777	-0.007%	
C#	69.296	1804	#070C	+0.007%	
D	73.416	1703	#06A7	+0.022%	
D#	77.782	1607	#0647	-0.004%	
E	82.407	1517	#05ED	+0.009%	
F	87.307	1432	#0598	+0.019%	
F#	92.499	1351	#0547	-0.028%	
G	97.999	1276	#04FC	+0.037%	
G#	103.826	1204	#04D4	+0.005%	
A	110.000	1136	#0470	-0.032%	
A#	116.541	1073	#0431	+0.039%	
B	123.471	1012	#03F4	-0.038%	

Note	Frequenz	Periode		Fehler	Oktave-1
C	130.813	956	#03DC	+0.046%	
C#	138.591	902	#0386	+0.007%	
D	146.832	851	#0353	-0.037%	
D#	155.564	804	#0324	+0.058%	
E	164.814	758	#02F6	-0.057%	
F	174.614	716	#02CC	+0.019%	
F#	184.997	676	#02A4	+0.046%	
G	195.998	638	#027E	+0.037%	
G#	207.652	602	#025A	+0.005%	
A	220.000	568	#0238	-0.032%	
A#	233.082	536	#0218	-0.055%	
B	246.942	506	#01FA	-0.038%	

Note	Frequenz	Periode		Fehler	Oktave-0
C	261.626	478	#01DE	+0.046%	mittleres C
C#	277.183	451	#01C3	+0.007%	
D	293.665	426	#01AA	+0.081%	
D#	311.127	402	#0192	+0.058%	
E	329.628	379	#017B	-0.057%	
F	349.228	358	#0166	+0.019%	
F#	369.994	338	#0152	+0.046%	
G	391.995	319	#013F	+0.037%	Kammerton A
G#	415.305	301	#012D	+0.005%	
A	440.000	284	#011C	-0.032%	
A#	466.164	268	#010C	-0.055%	
B	493.883	253	#00FD	-0.038%	

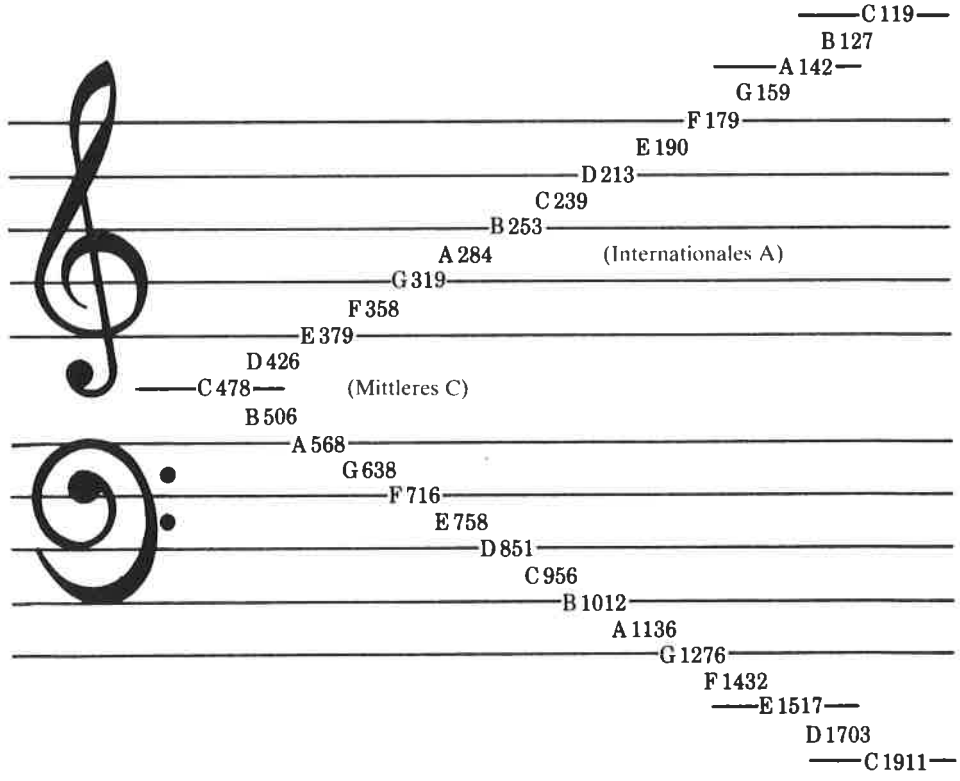
Note	Frequenz	Periode		Fehler	Oktave-1
C	523.251	239	#00EF	+0.046%	
C#	554.365	225	#00E1	-0.215%	
D	587.330	213	#00D5	+0.081%	
D#	622.254	201	#00C9	+0.058%	
E	659.255	190	#00BE	+0.206%	
F	698.457	179	#00B3	+0.019%	
F#	739.989	169	#00A9	+0.046%	
G	783.991	159	#009F	-0.277%	
G#	830.609	150	#0096	-0.328%	
A	880.000	142	#008E	-0.032%	
A#	932.328	134	#0086	-0.055%	
B	987.767	127	#007F	+0.356%	

Note	Frequenz	Periode		Fehler	Oktave 2
C	1046.502	119	#0077	-0.374%	
C#	1108.731	113	#0071	+0.229%	
D	1174.659	106	#006A	-0.390%	
D#	1244.508	100	#0064	-0.441%	
E	1318.510	95	#005F	+0.206%	
F	1396.913	89	#0059	-0.543%	
F#	1479.978	84	#0054	-0.548%	
G	1567.982	80	#0050	+0.350%	
G#	1661.219	75	#004B	-0.328%	
A	1760.000	71	#0047	-0.032%	
A#	1864.655	67	#0043	-0.055%	
B	1975.533	63	#003F	-0.435%	

Note	Frequenz	Periode		Fehler	Oktave 3
C	2093.004	60	#003C	+0.462%	
C#	2217.461	56	#0038	-0.662%	
D	2349.318	53	#0035	-0.390%	
D#	2489.016	50	#0032	-0.441%	
E	2637.021	47	#002F	-0.855%	
F	2793.826	45	#002D	+0.574%	
F#	2959.955	42	#002A	-0.548%	
G	3135.963	40	#0028	+0.350%	
G#	3322.438	38	#0026	+0.992%	
A	3520.000	36	#2924	+1.357%	
A#	3729.310	34	#0022	+1.417%	
B	3951.066	32	#0020	+1.134%	

Note	Frequenz	Periode		Fehler	Oktave 4
C	4186.009	30	#001E	+0.462%	
C#	4434.922	28	#001C	-0.662%	
D	4698.636	27	#001B	+1.469%	
D#	4978.032	25	#0019	+1.441%	
E	5274.041	24	#0018	+1.246%	
F	5587.652	22	#0016	-1.685%	
F#	5919.911	21	#0015	-0.548%	
G	6271.927	20	#0014	+0.350%	
G#	6644.875	19	#0013	+0.992%	
A	7040.000	18	#0012	+1.357%	
A#	7458.621	17	#0011	+1.417%	
B	7902.133	16	#0010	+1.134%	

Die Noten der C-Dur-Tonleiter sind unten in anschaulicher Form dargestellt.

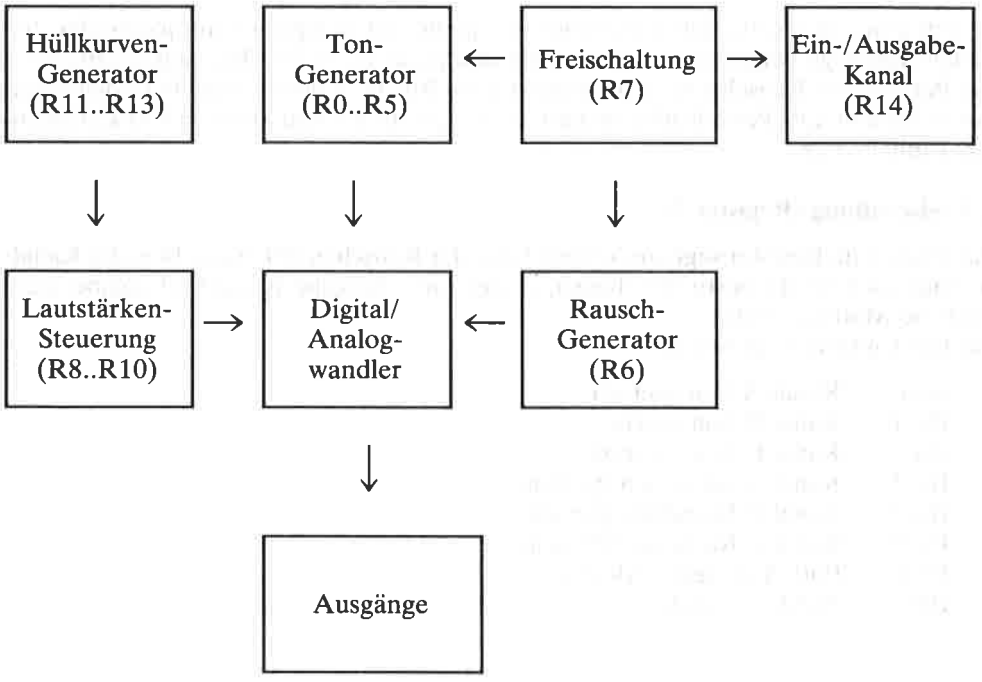


ANHANG IX
Programmierbarer Tongenerator

Der programmierbare Tongenerator (PSG) ist ein AY-3-8912-Chip. Er ist zusammenfassend in Abschnitt 7.1 beschrieben. Der PSG besitzt eine Anzahl Register, die unten beschrieben werden. Diese Information ist für den Anwender von Interesse, besonders wenn er Hardware-Hüllkurven verwendet (siehe hierzu Abschnitt e). Die durch die Tongenerator-Verwaltung ermöglichten Software-Hüllkurven haben mit Ausnahme sehr kurzer Anschwell- oder Abkling-Perioden den gleichen Leistungsumfang wie der Tongenerator-Chip.

Wenn der Anwender den Tongenerator-Chip direkt und ohne Einschaltung der Tongenerator-Verwaltung ansteuern möchte, reicht die hier gegebene Information nicht aus. Der Anwender sollte hierzu die Datenblätter des Herstellers einsehen. Ihm wird empfohlen, zum Beschreiben der Tongenerator-Chip-Register MC SOUND REGISTER aufzurufen, da sie die Timing-Bedingungen erfüllt.

Das folgende Diagramm verdeutlicht die Wechselwirkungen zwischen den verschiedenen Blöcken des Tongenerator-Chips:



Der Tongenerator-Chip hat die folgenden Datenregister:

Register 0:	Kanal A Ton-Periode, Feineinstellung.
Register 1:	Kanal A Ton-Periode, Grobeinstellung.
Register 2:	Kanal B Ton-Periode, Feineinstellung.
Register 3:	Kanal B Ton-Periode, Grobeinstellung.
Register 4:	Kanal C Ton-Periode, Feineinstellung.
Register 5:	Kanal C Ton-Periode, Grobeinstellung.
Register 6:	Rausch-Periode.
Register 7:	Freischaltung und Ein-/Ausgabe-Richtung.
Register 8:	Kanal A, Lautstärken- und Hüllkurven-Freischaltung.
Register 9:	Kanal B, Lautstärken- und Hüllkurven-Freischaltung.
Register 10:	Kanal C, Lautstärken- und Hüllkurven-Freischaltung.
Register 11:	Hüllkurven-Periode, Feineinstellung.
Register 12:	Hüllkurven-Periode, Grobeinstellung.
Register 13:	Hüllkurven-Form.
Register 14:	Eingabe von oder Ausgabe an Port A.
Register 15:	Nicht verwendet.

a. Tongeneratoren (Register 0...5)

Jeder Kanal hat zwei zugehörige Ton-Perioden-Register. Sie setzen die Periode des zu erzeugenden Tones (in 8 Mikrosekunden-Einheiten) für diesen Kanal. Das Register für die Feineinstellung speichert die niederwertigen 8 Bits des Tons. Das Register für die Grobeinstellung speichert die höchstwertigen 4 Bits der Periode. Um einen Ton über den Kanal auszugeben, muß das entsprechende Bit im Freischalt-Register gelöscht sein.

b. Rausch-Generator (Register 6)

Es gibt eine einzige Pseudo-Zufalls-Rauschquelle. Ihr Ausgang kann jedem der drei Kanal-Ausgänge beigemischt werden (wie durch das Freischalt-Register spezifiziert). Die Periode des Rauschgenerators ist durch die Bits 0...4 des Rausch-Perioden-Registers bestimmt. Die Periode gibt die mittlere Frequenz des Rauschens in 8 Mikrosekunden-Einheiten an.

c. Freischaltung (Register 7)

Das Freischalt-Register sagt aus, ob ein Ton oder Rauschen der Ausgabe jedes Kanals beigemischt wird. Es bestimmt ebenso, ob der Ein-/Ausgabe-Kanal im Eingabe- oder Ausgabe-Modus arbeitet.
Die Bits sind wie folgt belegt:

Bit 0:	Kanal A Ton sperren.
Bit 1:	Kanal B Ton sperren.
Bit 2:	Kanal C Ton sperren.
Bit 3:	Kanal A Rauschen sperren.
Bit 4:	Kanal B Rauschen sperren.
Bit 5:	Kanal C Rauschen sperren.
Bit 6:	Port A Ausgabe-Modus.
Bit 7:	Nicht verwendet.

Man beachte, daß Port A mit der Tastatur und dem Joystick verbunden ist und der Port demzufolge im Eingabe-Modus sein muß. Der Anwender muß sicherstellen, daß Bit 6 des Freischalt-Registers immer auf 0 gesetzt ist.

d. Lautstärken-Steuerung (Register 8...10)

Jeder Kanal hat ein ihm zugeordnetes Lautstärken-Steuerregister. Bit 4 dieses Registers bestimmt, ob eine Hardware-Hüllkurve für diesen Kanal verwendet wird. Wenn das Bit gesetzt ist, befindet sich die Kanal-Lautstärke unter der Steuerung des Hardware-Hüllkurven-Generators. Ist das Bit gelöscht, so wird die Lautstärke durch die Bits 0...3 des Registers gesetzt. Ein Wert 0 bedeutet, der Ton ist nicht hörbar, ein Wert 15 bedeutet maximale Lautstärke.

e. Hüllkurven-Generator (Register 11...13)

Der Tongenerator-Chip besitzt einen Hardware-Hüllkurven-Generator, der zur Steuerung jeder Kombination der drei Tongenerator-Kanäle, wie durch die Lautstärke-Register der Kanäle spezifiziert (siehe oben unter d), verwendet werden kann. Die Bits 0 bis 3 von Register 13 bestimmen die Form der Hüllkurve. Die folgende Tabelle zeigt die Werte für die Erzeugung der 8 möglichen Hardware-Hüllkurven. Andere Werte (0...7) kopieren die Hüllkurven 8 bis 15.

8:  Wiederholter Aufwärts-Sprung und Abwärts-Rampe.

9:  Aufwärts-Sprung und Abwärts-Rampe mit nachfolgendem Halten auf minimaler Lautstärke (Null).

10:  Aufwärts-Sprung und wiederholte Abwärts- und Aufwärts-Rampe.

11:  Aufwärts-Sprung und Abwärts-Rampe, dann Aufwärts-Sprung und Halten auf maximaler Lautstärke (15).

12:  Wiederholte Aufwärts-Rampe und Abwärts-Sprung.

13:  Aufwärts-Rampe mit nachfolgendem Halten auf maximaler Lautstärke (15).

14:  Wiederholte Aufwärts- und Abwärts-Rampe.

15:  Aufwärts-Rampe und Abwärts-Sprung mit nachfolgendem Halten auf minimaler Lautstärke (0).

Die Länge jeder Rampe (aufwärts oder abwärts) wird durch die Hüllkurven-Periode bestimmt. Die Hüllkurven-Periode ist ein 16-Bit-Wert, dessen niederwertiges Byte in Register 11 und dessen höherwertiges Byte in Register 12 gespeichert ist. Die Periode ist in 128-Mikrosekunden-Einheiten gegeben und stellt die Zeit zwischen den Abschnitten der Rampe dar. Da die Rampe 16 Abschnitte besitzt (entsprechend den 16 Lautstärke-Einstellungen), ist die für die gesamte Rampe erforderliche Zeit 1024 Mikrosekunden (Zeit der Hüllkurven-Periode) lang (d.h. die Hüllkurven-Periode setzt die gesamte Zeit für die Rampe in Millisekunden).

f. Ein-/Ausgabe-Kanal (Register 14)

Der Betriebs-Modus des PSG-Kanals wird durch ein Bit im Freischalt-Register (siehe Abschnitt c) bestimmt. Da Port A jedoch für das Lesen der Tastatur und Joysticks vorgesehen ist, sollte er immer im Eingabe-Modus betrieben werden. Der Port kann durch Lesen des Inhalts von Register 14 bestimmt werden. Das Abfragen der Tastatur ist ein komplexes Unterfangen und wird besser der Tastatur-Verwaltung überlassen, die entsprechende Vorkehrungen für den Zugriff auf Tasten getroffen hat.

Bezugnahmen auf Port B innerhalb der Datenblätter des Herstellers sollten ignoriert werden, da der AY-3-8912 eine Chip-Version ist, die keinen Port B besitzt.

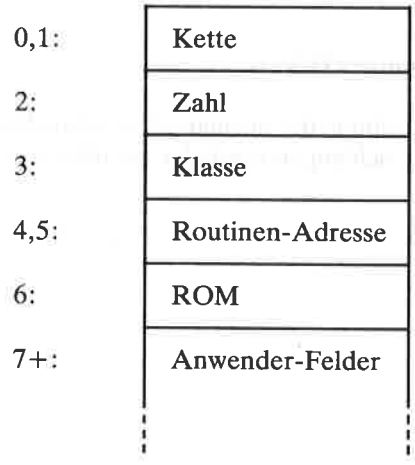
ANHANG X
Betriebssystem-Kern-Belegung

Der Anwender stellt dem Betriebssystem-Kern für verschiedene Zwecke Blöcke zur Verfügung. Die Belegung dieser Blöcke ist weiter unten beschrieben. Es gibt nur sehr wenige Möglichkeiten, die dem Anwender das Beschreiben eines dieser Blöcke zulassen. Die Routinen bewerkstelligen die meisten der vom Anwender benötigten Aktionen (siehe KL INIT EVENT, KL ADD TICKER, KL NEW FRAME FLY, KL NEW FAST TICKER und KL DISARM EVENT). Diese Routinen schreiben Werte aus Registern in den Block. Der Anwender sollte -mit den unten aufgeführten Ausnahmen- die Blökke nicht beschreiben.

Die folgenden Blöcke müssen in den zentralen 32KB RAM liegen (der Betriebssystem-Kern kann sonst nicht auf sie zugreifen).

a. Ereignis-Blöcke

Eine allgemeine Beschreibung der Ereignisse und Ereignis-Blöcke findet sich in Abschnitt 11. Ein Ereignis-Block ist folgendermaßen belegt:



Kette ist ein System-Pointer (Zeiger), der vom Anwender nie beschrieben werden darf. Er dient dem Speichern von Ereignissen in den verschiedenen Ereignis-Warteschlangen.

Klasse bezeichnet den Typ des Ereignisses und sollte vom Anwender nicht beschrieben werden.

Bit 0:	1 → „Near-Address", 0 → „Far-Address".
Bits 1...4:	Priorität des synchronen Ereignisses.
Bit 5:	Muß Null sein.
Bit 6:	1 → Spezielles Ereignis, 0 → Normales Ereignis.
Bit 7:	1 → Asynchrones Ereignis, 0 → Synchrones Ereignis.

Man beachte, daß viele System-Warteschlangen in der Prioritäts-Reihenfolge angeordnet sind. Dadurch muß der Block nach einer Änderung der Priorität erneut eingereiht werden. Es ist nicht ausreichend, lediglich die Priorität im Ereignis-Block zu ändern.

Zahl ist die Ereignis-Zahl. Sie registriert, wieviele Anstöße auf die Bearbeitung warten oder ob das Ereignis gesperrt ist. In Abschnitt 11.2 befindet sich eine vollständige Beschreibung zur Anwendung der Ereignis-Zahl.

Routinen-Adresse und ROM ersetzen die „Far-Address" der Ereignis-Routine. Ist das „Near-Address"-Bit der Ereignis-Klasse wahr, so befindet sich die Ereignis-Routine auf einer „Near-Address" und das ROM-Select-Byte (Byte 6) wird ignoriert, die Ereignis-Routine kann direkt aufgerufen werden. Ist das „Near-Address"-Bit falsch, so befindet sich die Ereignis-Routine auf einer „Far-Address" und die Bytes 4, 5 und 6 ersetzen die aufzurufende „Far-Address" zum Ablauf der Ereignis-Routine. Der Anwender kann die Routinen-Adresse und die ROM-Felder beschreiben (ebenso das Near-Address-Bit im Klasse-Byte). Diese Operation darf jedoch nicht unterbrochen werden (d.h. mit gesperrten Unterbrechungen).

Die Anwender-Felder sind optionell. Sie können verwendet werden, um einen für den Ereignis-Block spezifischen Daten-Bereich anzulegen, so daß eine Ereignis-Routine zwischen mehreren unterschiedlichen Ereignis-Blöcken aufgeteilt werden kann (die Adressen der Anwender-Felder werden an die Ereignis-Routine übergeben).

b. Blöcke für Warteschlangen normaler Ereignisse (Ticker)

Abschnitt 10 beinhaltet eine allgemeine Beschreibung der normalen Unterbrechungen und ihrer Warteschlangen. Ein Block für Warteschlangen normaler Ereignisse ist folgendermaßen belegt:

0,1:	Tick-Kette
2,3:	Tick-Zahl
4,5:	Auflade-Zahl
6+:	Ereignis-Block

Tick-Kette ist ein System-Pointer (Zeiger), der vom Anwender nie beschrieben werden darf. Er dient dem Speichern des Blocks in der Warteschlange für normale Ereignisse.

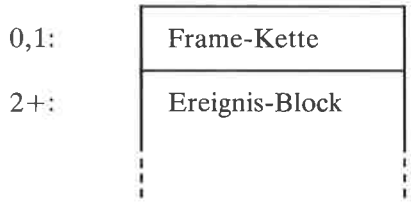
Tick-Zahl ist die Anzahl der Ereignis-Anstöße vor dem nächsten Auftreten eines Anstoßes. Eine Zahl 0 bedeutet, daß der Block ruht und keinen Anstoß erzeugt. Idealerweise sollte ein ruhender Block aus der Warteschlange entfernt werden, damit keine Zeit verschwendet wird. Der Anwender kann dieses Feld gegebenenfalls beschreiben; dies sollte jedoch ohne Unterbrechung geschehen.

Auflade-Zahl ist der Wert, auf den die Zahl nach jedem Anstoß gesetzt wird. Wenn die Auflade-Zahl Null ist, ruht der Block nach dem nächsten Anstoß. Der Anwender kann dieses Feld gegebenenfalls beschreiben; dies sollte jedoch ungeteilt geschehen.

Ereignis-Block ist ein Standard-Ereignis-Block, wie oben in Abschnitt a beschrieben.

c. Blöcke für Bildrücklauf-Ereignisse (Frame Flyback)

Abschnitt 10 beinhaltet eine allgemeine Beschreibung der Bildrücklauf-Ereignisse und ihrer Warteschlangen. Ein Block für Bildrücklauf-Ereignisse ist folgendermaßen belegt:

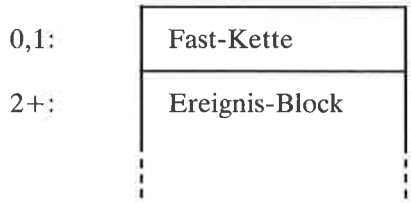


Frame-Kette ist ein System-Pointer (Zeiger), der vom Anwender nie beschrieben werden darf. Er dient dem Speichern des Blocks in der Warteschlange für Bildrücklauf-Ereignisse.

Ereignis-Block ist ein Standard-Ereignis-Block, wie oben in Abschnitt a beschrieben.

d. Blöcke für Warteschlangen schneller Ereignisse (Fast Ticker)

Abschnitt 10 beinhaltet eine allgemeine Beschreibung der schnellen Unterbrechungen und ihrer Warteschlangen. Ein Block für Warteschlangen schneller Ereignisse ist folgendermaßen belegt:



Fast-Kette ist ein System-Pointer (Zeiger), der vom Anwender nie beschrieben werden darf. Er dient dem Speichern des Blocks in der Warteschlange für schnelle Ereignisse.

Ereignis-Block ist ein Standard-Ereignis-Block, wie oben in Abschnitt a beschrieben.

ANHANG XI
Wechselseitiger Register-Satz

Der Z80-Mikroprozessor hat zwei Register-Sätze – den normalen Satz (AF, BC, DE und HL) und den wechselseitigen Satz (AF', BC', DE' und HL'). Wenn die in diesem Anhang empfohlenen Techniken befolgt werden, ist die Benutzung des wechselseitigen Register-Satzes durch den Anwender vermeidbar. Dies ist vorteilhaft, da der wechselseitige Register-Satz durch die Firmware (insbesondere dem Betriebssystem-Kern) zum Speichern bestimmter System-Werte und Flags verwendet wird. Wenn sichergestellt wird, daß der Anwender nicht in die Firmware einspringt, kann der wechselseitige Register-Satz ohne Einschränkungen benutzt werden. Natürlich kann der Anwender dann die Möglichkeiten der Firmware nicht mehr nutzen. Weiterhin muß er die Unterbrechungen sperren, da sie andernfalls die Ausführung von Firmware-Routinen zur Folge haben.

In den folgenden Abschnitten werden eine Reihe verschiedener Methoden beschrieben, die eine Überwindung dieser Einschränkungen ermöglichen. Die gewählte Methode hängt von der Verwendung des wechselseitigen Register-Satzes ab.

a. Benutzung des wechselseitigen Register-Satzes durch die Firmware

Der Betriebssystem-Kern speichert System-Variablen im wechselseitigen Register-Satz. Dies ermöglicht ihm den einfachen Zugriff auf diese Variablen, wodurch einige Operationen beschleunigt werden (insbesondere der Ein- und Aussprung in/aus Firmware-Routinen). Nur BC' und das wechselseitige Carry-Flag (Carry') speichern Werte. Routinen verwenden jedoch die anderen wechselseitigen Register ebenfalls, d.h. sie können durch Firmware-Routinen verfälscht werden.

B' dient zum Speichern der Ein-/Ausgabe-Adressen des Gate-Arrays (#7F). C' wird zum Speichern der für die Einstellung des momentanen ROM-State und Bildschirm-Modus erforderlichen Werte verwendet:

Bits 0...1:	Setze den Bildschirm-Modus.
Bit 2:	Sperrt das untere ROM.
Bit 3:	Sperrt das obere ROM.
Bits 4...7:	System-Wert zum Selektieren der Gate-Array-Funktion.

Durch Änderung der ROM-State-Bits und Ausführung eines OUT,C-Befehls können ROM's gesperrt oder freigeschaltet werden. Man beachte insbesondere, daß die Z80-Befehle OUT (C),r und IN (C),r das Register B für die oberen 8 Bits der Ein-/Ausgabe-Adresse verwenden. Die Hardware benötigt diese oberen Bits für die Dekodierung der Ein-/Ausgabe-Adresse und ignoriert die unteren 8 Bits! (OUT (C),C kann zur Änderung des ROM-State während der Unterbrechungs-Bearbeitung verwendet werden, wenn die normalen Betriebssystem-Kern-Einträge (z.B. KL U ROM ENABLE) nicht aufgerufen werden können, weil sie z.B. Unterbrechungen freischalten müssen.

Carry' befindet sich normalerweise im Zustand falsch. Ist Carry' wahr, so zeigt dies an, daß sich die Firmware im Unterbrechungs-Bereich befindet.

Die Firmware verwendet gelegentlich dieses Flag, um im Unterbrechungs-Bereich andere Aktionen zu tätigen, als im normalen Bearbeitungs-Bereich (gewöhnlich das Verhindern einer Unterbrechungs-Freischaltung).

b. Einfache Anwendungen des wechselseitigen Register-Satzes

Die in diesem Abschnitt beschriebene Technik ermöglicht die Benutzung des wechselseitigen Register-Satzes unter der Voraussetzung, daß keine Firmware-Routinen aufgerufen werden und Unterbrechungen gesperrt sind.

Nach Sperrung der Unterbrechungen können die Register A', DE' und HL' frei benutzt werden. Wenn die Register BC' oder F' (insbesondere Carry') verwendet wurden, muß ihr Original-Inhalt vor der erneuten Freischaltung der Unterbrechungen wiederhergestellt werden. Der Anwender kann Bits in C' ändern (wie unter a beschrieben) und muß den Original-Inhalt nur dann nicht wiederherstellen, wenn ein OUT (C),C-Befehl ausgeführt wird. Dieser Befehl sorgt dafür, daß die Hardware mit dem momentanen Zustand synchronisiert wird. Die Maschine arbeitet nicht korrekt, wenn bei freigeschalteten Unterbrechungen die Hardware und der Wert in C' nicht aufeinander abgestimmt sind.

Diese Technik setzt voraus, daß die Unterbrechungen für die Dauer der Bearbeitung gesperrt sind. Dies ist bei kurzen Bearbeitungszeiten akzeptabel, bei längeren jedoch nicht. Die Sperrung für eine längere Zeit stoppt einige Firmware-Funktionen, wie z.B. die Zeitgeber (und damit das Aufleuchten der Ink, die Ton-Erzeugung und die Tastatur-Abfrage). Dauert eine Bearbeitung länger, dann sollte stattdessen besser eine der Techniken, wie in Abschnitt c oder d beschrieben, verwendet werden.

Beispiel:

Der Anwender möchte eine Routine erstellen, die einen LD A,(BC) vom RAM ausführt (ähnlich einem RAM LAM-Pseudo-Befehl der Firmware).

Der Code für diese Routine könnte folgendermaßen aussehen:

```
A_FROM_BC:

        PUSH    BC
        DI                      ;** Dient zur Benutzung des wechselseitigen Registe
                                Satzes.
        EXX
        POP     HL              ; Transferiere BC nach HL'.
;
        LD      A,C
        SET     2,A             ; Setze das Bit für Sperrung unteres ROM.
        SET     3,A             ; Setze das Bit für Sperrung oberes ROM.
        OUT     (C),A           ; Übergib an die Hardware.
;
        LD      A,(HL)          ; Lies den Wert vom RAM.
        OUT     (C),C           ; Wiederherstellen des alten ROM-State.
```

```
;
        EXX
        EI                      ; ** Ende der Benutzung des wechselseitigen Registe
                                     Satzes.
        RET
```

Man beachte, daß diese Routine RAM-resident sein muß, da sonst die Sperrung der ROM's ungeahnte Auswirkungen haben kann.

c. Anwendung des wechselseitigen Register-Satzes mit freigeschalteten Unterbrechungen

Die in diesem Abschnitt beschriebene Technik ermöglicht die Verwendung des wechselseitigen Register-Satzes bei freigeschalteten Unterbrechungen. Sie verbietet jedoch den Aufruf von Firmware-Routinen.

Die einfachste Verwendung des wechselseitigen Register-Satzes durch Sperrung der Unterbrechungen (wie oben beschrieben) ist unbefriedigend, wenn die Unterbrechungen für eine längere Zeitperiode gesperrt bleiben. Durch Neubelegung des INTERRUPT ENTRY in der Sprungtabelle des unteren Betriebssystem-Kerns können Unterbrechungen abgefangen und entsprechende Aktionen zum Wiederherstellen der Firmware-Register eingeleitet werden. Die folgenden Aktionen sind erforderlich:

Vor Verwendung des wechselseitigen Register-Satzes wird das Firmware-Register BC' gesichert und INTERRUPT ENTRY neu belegt, so daß die Unterbrechungs-Routine des Anwenders benutzt werden kann.

Nach Abschluß der Bearbeitung mit dem wechselseitigen Register-Satz wird BC' und Carry' wiederhergestellt, sowie der INTERRUPT ENTRY für die Unterbrechungs-Routine der Firmware belegt.

Beim Auftreten einer Unterbrechung wird der wechselseitige Register-Satz gesichert, der Firmware-Register-Inhalt von BC' und Carry' wiederhergestellt und INTERRUPT ENTRY mit dem Eintrag der Firmware-Unterbrechungs-Routine belegt. Das letztere ist in den Fällen wichtig, in denen eine zweite Unterbrechung während der Bearbeitung von Ereignissen auftritt, welche durch die erste Unterbrechung der Unterbrechungs-Ebene angestoßen wurde (die Ereignis-Bearbeitung wird ja bekanntlich mit freigeschalteten Unterbrechungen durchgeführt).

Nach der Unterbrechungs-Bearbeitung wird der Firmware-Inhalt von BC' gesichert, der wechselseitige Register-Satz des Anwenders wiederhergestellt und INTERRUPT ENTRY mit dem Eintrag der Unterbrechungs-Routine des Anwenders belegt.

Man beachte, daß beim Belegen des INTERRUPT ENTRY unbedingt das untere ROM gesperrt sein und bleiben muß. Die ROM-Version des INTERRUPT ENTRY kann nicht neu belegt werden! Wenn eine Unterbrechung im freigeschalteten Zustand des unteren ROM's aufgetreten ist, springt die Firmware direkt und ohne vorheriges Wiederherstellen des wechselseitigen Register-Satzes in ihre Unterbrechungs-Routine.

Die folgenden Routinen sind Beispiele für die Realisierung der oben beschriebenen Technik.

```
;
; Die folgenden Speicherplätze werden belegt:
;
FIRM_BC:   DEFS 2    ; 2 Bytes zum Speichern des Firmware BC'.
FIRM_INT:  DEFS 2    ; 2 Bytes zum Speichern der Adresse der Firmware-
                       Unterbrechungs-Routine.
USER_AF:   DEFS 2    ; 2 Bytes zum Speichern des Anwender AF'.
USER_BC:   DEFS 2    ; 2 Bytes zum Speichern des Anwender BC'.
USER_DE:   DEFS 2    ; 2 Bytes zum Speichern des Anwender DE'.
USER_HL:   DEFS 2    ; 2 Bytes zum Speichern des Anwender HL'.
;
;    Diese Routine stellt die Umgebung für die Verwendung des
;    wechselseitigen Registersatzes her.
;    Man beachte, daß die Unterbrechungen freigeschaltet sind.
;
;
USER_ALTERNATE:
   DI                                ; Eine Unterbrechung wäre verheerend.
   EX       AF, AF'
   EXX                               ; Austausch des wechselseitigen
                                       Register-Satzes.
;
   LD       (FIRM_BC),BC             ; Sichern des Firmware BC'.
;
   LD       HL,(INTERRUPT_ENTRY+1)
   LD       (FIRM_INT),HL            ; Sichern der Firmware-Unter-
                                       brechungs-Routine.
;
   LD       HL,USER_INTERRUPT        ; Verwende die Ersatz-Unter-
                                       brechungs-Routine.
   LD       (INTERRUPT_ENTRY+1,HL
;
   LD       HL,(USER_AF)
   PUSH     HL
   POP      AF                       ; Wiederherstellen des Anwender AF'.
   LD       BC,(USER_BC)             ; Wiederherstellen des Anwender BC'.
   LD       DE,(USER_DE)             ; Wiederherstellen des Anwender DE'.
   LD       HL,(USER_HL)             ; Wiederherstellen des Anwender HL'.
;
   EXX                               ; Tausch auf Standard-Register-Satz.
   EX       AF, AF'
   EI                                ; Abschluß der Bearbeitung mit dem
                                       wechselseitigen Register-Satz.
   RET
;
```

```
; Diese Routine stellt die Firmware-Umgebung für die Verwendung
; des wechselseitigen Registersatzes wieder her.
; Man beachte, daß die Unterbrechungen gesperrt und nicht erneut
; freigeschaltet werden.
;
;
FIRM-ALTERNATE:
   DI                                  ; Eine Unterbrechung wäre verheerend.
   EX       AF, AF'
   EXX                                 ; Austausch des wechselseitigen
                                         Register-Satzes.
   LD       (USER_HL),HL               ; Sichern des Anwender HL'.
   LD       (USER_DE),DE               ; Sichern des Anwender DE'.
   LD       (USER_BC),BC               ; Sichern des Anwender BC'.
   PUSH     AF
   POP      HL
   LD       (USER_AF),HL               ; Sichern des Anwender AF'.
;
   LD       HL,(FIRM_INT)              ; Wiederherstellen der Firmware-
                                         Unterbrechungs-Routine.
;  LD       (INTERRUPT_ENTRY+1),HL
   LD       BC,(FIRM_BC)               ; Wiederherstellen des Firmware BC'.
   OR       A,A                        ; Setze das Firmware-Carry' auf falsch.
   EXX                                 ; Tausch auf Standard-Register-Satz.
   EX       AF, AF'
   RET                                 ; Kein EI beim Einsprung in die Unter-
                                         brechungs-Ebene.
;
;
; Diese Routine ersetzt die Unterbrechungs-Routine der Firmware,
; wenn der Anwender den wechselseitigen Register-Satz benutzt.
;
;
USER_INTERRUPT:
   CALL  FIRM_ALTERNATE                ; Schalte auf die Umgebung für die Firm-
                                         ware.
   CALL  INTERRUPT_ENTRY               ; Ablauf der normalen Unterbrechungs-
                                         Routine.
   JP    USER_ALTERNATE                ; Schalte auf die Umgebung für den
                                         Anwender zurück.
```

Zu Beginn der Benutzung des wechselseitigen Registersatzes führt der Anwender den Befehl

```
CALL USER_ALTERNATE
```

und zum Schluß die Befehle

```
CALL FIRM_ALTERNATE
EI
```

aus.

d. Aufruf von Firmware-Routinen während der Verwendung des wechselseitigen Register-Satzes

Die in diesem Abschnitt beschriebene Technik erweitert die in Abschnitt c diskutierte, damit der Anwender während der Benutzung des wechselseitigen Register-Satzes Firmware-Routinen aufrufen kann.

Der Aufruf einer Firmware-Routine erfordert genau die gleiche Vorgehensweise, wie der einer Unterbrechungs-Routine:

> Vor dem Aufruf einer Firmware-Routine wird der wechselseitige Register-Satz des Anwenders gesichert, das Firmware BC' und Carry' wiederhergestellt und INTERRUPT ENTRY mit dem Eintrag der Firmware-Unterbrechungs-Routine belegt. Letzteres ist für den Fall, daß eine Unterbrechung während der Ausführung der Firmware-Routine auftritt, von Wichtigkeit.
>
> Nach dem Ablauf der Firmware-Routine wird der Firmware BC' gesichert, der wechselseitige Register-Satz des Anwenders wiederhergestellt und INTERRUPT-ENTRY wieder mit dem Eintrag der Anwender-Unterbrechungs-Routine belegt.

Wie in Abschnitt c bemerkt, muß sichergestellt sein, daß das untere ROM während der Benutzung des wechselseitigen Register-Satzes gesperrt ist, da der INTERRUPT ENTRY im ROM nicht neu belegt werden kann und demzufolge direkt in die Firmware-Unterbrechungs-Routine gesprungen wird.

Unter Verwendung der in Abschnitt c definierten Routinen kann eine Firmware-Routine mit folgender Befehls-Sequenz aufgerufen werden:

```
        ..
CALL    FIRM_ALTERNATE          ; Schalte auf die Umgebung für die Firm-
                                  ware.
EI
CALL    firmware                ; Ablauf der Firmware-Routine.
CALL    USER_ALTERNATE          ; Schalte auf die Umgebung für den An-
                                  wender zurück.
        ..
```

Der obige Befehlscode ist relativ lang, da eine Menge Firmware-Aufrufe enthalten sind (10 Bytes pro Aufruf). Die folgende Routine nimmt eine Adresse der aufzurufenden Firmware-Routine als Reihen-Parameter (und benötigt nur 5 Bytes pro Aufruf).

```
;
; Diese Routine sichert den wechselseitigen Register-Satz des
; Anwenders, ruft eine Firmware-Routine mit Reihen-Adresse auf und
; stellt anschließend den wechselseitigen Register-Satz des
; Anwenders wieder her.
;
;
FIRM_ROUTINE:
   CALL     FIRM ALTERNATE      ; Schalte auf die Umgebung für die
                                  Firmware.
;
```

```
EXX                                ; Unterbrechungen sind gesperrt.
POP     HL                         ; Wiederherstellen der aufzurufenden
LD E,   (HL)                       ; Routinen-Adresse, verwendet Firm-
                                   ; ware DE'
                                   ; und HL', die verfälscht werden
INC     HL                         ; können.
LD D,   (HL)                       ; Hole die aufzurufende Routine in DE'.
INC     HL
PUSH    HL                         ; Gib die tatsächliche Rückkehr-Adresse
                                     zurück.
;
LD      HL,USER ALTERNATE          ; Wiederherstellen der Anwender-
                                   ; Umgebung beim Rücksprung der
                                   ; Firmware durch Ablegen einer
                                   ; gefälschten Rückkehr-Adresse
PUSH    HL                         ; auf dem Stack.
;
PUSH    DE                         ; Sichern der aufzurufenden Routine.
EXX
EI
RET                                ; Sprung in die aufzurufende Routine.
```

Um eine Firmware-Routine unter Verwendung der obigen Routine aufzurufen, sollte die folgende Befehls-Sequenz verwendet werden:

```
CALL    FIRM_ROUTINE
firmware                           ; Adresse der aufzurufenden Routine.
..                                 ; FIRM ROUTINE kehrt hierher zurück.
```

ANHANG XII
Hardware

A. Prozessor

Der Prozessor ist ein Z80A mit einer Takt-Frequenz von 4,00 MHz (±0,1%). Es gibt eine Logik, die unter Verwendung der CPU WAIT-Möglichkeit MREQ und IORQ verlängert, so daß der Prozessor nur einen Zugriff pro Mikrosekunde ausführen kann.

Der NMI-Pin des Prozessors ist mit einem Pull-up-Widerstand versehen und steht auf dem Erweiterungs-Bus zur Verfügung. Eine nicht-maskierbare Unterbrechung kann jedoch dazu führen, daß die Firmware einige Zeit-Bedingungen verletzt. Die Verwendung wird daher nicht empfohlen.

Der Unterbrechungs-Eingang des Prozessors wird durch ein Flip-Flop im Video-Gate-Array angesteuert. Dieses Flip-Flop wird durch jeden vertikalen Bildrücklauf und daraufhin alle 52 Abtast-Zeilen bis zum nächsten vertikalen Bildrücklauf gesetzt. Die Unterbrechung tritt ungefähr 2 Abtast-Perioden (125 Mikrosekunden) nach dem Beginn des 8 Abtast-Perioden (500 Mikrosekunden) langen, vertikalen Bildrücklauf-signals auf. Das Unterbrechungs-Latch wird durch die Unterbrechungs-Quittung des Prozessors oder durch einen besonderen Software-Befehl gelöscht. Das oberste Bit des „52-Abtast-Perioden-Zählers" wird ebenfalls gelöscht, wenn der Prozessor eine nach dem Überlauf dieses Zählers aufgetretene Unterbrechung quittiert. Hierdurch wird eine Erweiterung des Unterbrechungs-Systems ermöglicht.

B. Speicher

ROM

Auf der Prozessor-Karte befindet sich ein 32KB-ROM, das im Adressraum des Prozessors in zwei 16KB-Blöcke aufgeteilt ist. Die untere Hälfte des ROM's belegt die Adressen #0000 bis #3FFF, die obere Hälfte belegt #C000 bis #FFFF. Diese beiden ROM-Hälften können durch zwei Steuer-Latches im Video-Gate-Array separat freigeschaltet und gesperrt werden. Beim Power-up oder anderen System-Resets werden beide ROM-Hälften freigeschaltet. Ein Signal des Erweiterungs-Kanals (-Ports) kann zum Sperren dieses internen ROM's verwendet werden, damit stattdessen Zugriffe auf externe ROM's möglich sind. Diese werden durch Ausgabe-Befehle selektiert und ersetzen die obere Hälfte des 32KB-ROM's.

RAM

Die Prozessor-Karte ist mit 64KB dynamischem RAM auf den Adressen #0000 bis #FFFF bestückt. Die untersten und obersten 16KB sind überlagert, wenn ROM's freigeschaltet sind. Der Freischalt-Zustand der ROM's beeinflußt Lesezugriffe, hat jedoch keinen Einfluß auf Schreibzyklen, die korrekt durch das freigeschaltete ROM auf das überlagerte RAM stattfinden können.

VDU-Bildschirm-Speicher

Der Bildschirm verwendet 16KB des Prozessor-RAM's als Bildwiederhol-Speicher. Die verwendeten 16KB sind zwischen den Blöcken mit den Start-Adressen #0000, #4000, #8000 und #C000 umschaltbar, indem die oberen zwei Bits (Bit 12 und 13) des HD6845S-Start-Adressregisters programmiert werden (siehe Abschnitt 6.4).

Die Anordnung der Daten innerhalb des VDU-Bildschirm-Speichers ist vom momentan selektierten VDU-Modus abhängig. Man kann sich die Organisation des Speichers in allen Modi als 8K x 16 Bit-Worte vorstellen. Jedes Wort beinhaltet entweder 4, 8 oder 16 Pixel (P0...Pn) mit jeweils 1, 2 oder 4 Bits (B0...Bm). Diese Bits sind vom Modus wie folgt abhängig:

A0	Bit	Modus 0	Modus 1	Modus 2
0	D7	P0 B0	P0 B0	P0 B0
0	D6	P1 B0	P1 B0	P1 B0
0	D5	P0 B2	P2 B0	P2 B0
0	D4	P1 B2	P3 B0	P3 B0
0	D3	P0 B1	P0 B1	P4 B0
0	D2	P1 B1	P1 B1	P5 B0
0	D1	P0 B3	P2 B1	P6 B0
0	D0	P1 B3	P3 B1	P7 B0
1	D7	P2 B0	P4 B0	P8 B0
1	D6	P3 B0	P5 B0	P9 B0
1	D5	P2 B2	P6 B0	P10 B0
1	D4	P3 B2	P7 B0	P11 B0
1	D3	P2 B1	P4 B1	P12 B0
1	D2	P3 B1	P5 B1	P13 B0
1	D1	P2 B3	P6 B1	P14 B0
1	D0	P3 B3	P7 B1	P15 B0

Die Daten für die Bildschirm-Zeilen 0, 8, 16, 24 ... befinden sich im ersten 2KB-Block des Speichers, Daten für die Bildschirm-Zeilen 1, 9, 17, 25 ... in den entsprechenden Speicherplätzen des nächsten 2KB-Blockes des Speichers und die Zeichen 7, 15, 23, 31 ... im oberen 2KB-Block des 16KB-Speicherbereiches.

Die unteren 10 Bits des HD6845S-Start-Adressregisters definieren den Startpunkt des Bildschirms innerhalb dieser 2KB-Blöcke. Der Offset vom Beginn des 2KB-Blockes ist immer gerade und wird berechnet, indem der Register-Inhalt mit zwei multipliziert und als Modulo 2KB interpretiert wird. Wenn Daten oberhalb des 2KB-Blockes darzustellen sind, werden diese im nächsten 2KB-Block angeordnet (siehe Abschnitt 6.4).

C. Schnittstellen

Die Standard-Schnittstellen auf der Prozessorkarte belegen die Ein-/Ausgabekanäle des Z80 wie folgt:

Ein-/Ausgabe-Adresse	Ausgabe	Eingabe
#7Fxx	Video-Gate-Array	Nicht verwenden.
#BCxx	HD6845S CRTC-Adresse	Nicht verwenden.
#BDxx	HD6845S CRTC-Daten	Nicht verwenden.
#BExx	Nicht verwenden	Reserviert für CRTC-Status.
#BFxx	Nicht verwenden	HD6845S CRTC-Adresse.
#DFxx	Erweiterungs-Rom-Select	Nicht verwenden.
#EFxx	Centronics-Latch	Nicht verwenden.
#F4xx	µPD8255 Port A Daten	µPD8255 Port A Daten.
#F5xx	µPD8255 Port B Daten	µPD8255 Port B Daten.
#F6xx	µPD8255 Port C Daten	µPD8255 Port C Daten.
#F7xx	µPD8255 Steuerung	Undefiniert.
#F8xx	Erweiterungs-Bus	Erweiterungs-Bus.
#F9xx	Erweiterungs-Bus	Erweiterungs-Bus.
#FAxx	Erweiterungs-Bus	Erweiterungs-Bus.
#FBxx	Erweiterungs-Bus	Erweiterungs-Bus.
#FFxx	Nicht verwendet	Nicht verwendet.

Man beachte, daß die Z80-Befehle, die eine Adresse auf der oberen Hälfte des Adressbusses (A8...A15) ausgeben, verwendet werden müssen. Die Verwendung von Block-Ein-/Ausgabe-Befehlen, die während der Befehlsausführung das Register B verändern, wird nicht empfohlen. Ein-/Ausgabe-Adressen unterhalb von #7Fxx, die in der obigen Tabelle nicht aufgeführt sind, sollten nicht verwendet werden.

Die Peripherie des Erweiterungs-Busses muß die Adressen von A0...A7 mit Adress-Leitung A10 im LOW-Zustand decodieren. Zusätzlich können die Adress-Leitungen A8 und A9 zur Auswahl eines Geräts oder Registers der Peripherie decodiert werden. Ein-/Ausgabe-Kanäle des Erweiterungs-Busses im Adress-Bereich von #F800 bis #FBFF sind wie folgt reserviert:

Adresse (A0...A7)	Verwendung
#00... #7B	Nicht verwenden.
#7C... #7F	Reserviert für Disketten-Interface.
#80... #BB	Nicht verwenden.
#BC... #BF	Reserviert für zukünftige Anwendungen.
#C0... #DB	Nicht verwenden.
#DC... #DF	Reserviert für Kommunikations-Interfaces.
#E0... #FE	Verfügbar für Anwender-Peripherie.
#FF	Reset Peripherie.

Die gesamte Erweiterungs-Peripherie sollte bei einer Ausgabe an Ein-/Ausgabe-Kanal #F8FF zurückgesetzt werden. Insbesondere sollten die Peripherie-Einheiten, die Unterbrechungen erzeugen können, bis zur Reinitialisierung nach einer derartigen Ausgabe keine Unterbrechung erzeugen.

D. Programmierbarer Tongenerator AY-3-8912 (PSG)

Der PSG ist über die Kanäle A und C des µPD8255 ansprechbar. Man beachte, daß ein Schreiben oder Laden von AY-3-8912-Adressen eine maximale Dauer des Schreib- oder Lade-Befehls (mit BDIR im High-Zustand) von 10 Mikrosekunden haben kann. Der Takt-Eingang für den Tongenerator hat eine Frequenz von exakt 1,00 MHz. Das BC2-Signal befindet sich ständig im High-Zustand. Während des Power-up sollte der Ein-/Ausgabe-Kanal auf Eingabe programmiert werden.
Der Anwender sollte zum Beschreiben des PSG die Firmware-Routine MC SOUND REGISTER verwenden.

E. HD6845S CRT-Controller (HD6845S CRTC)

Der Zeichen-Takt des CRTC tritt jeweils nach zwei vom Speicher geholten Bytes, d.h. alle 1,0 Mikrosekunden, auf. Das erste Byte eines Paares hat eine gerade, das zweite eine ungerade Adresse. In der normalen Betriebsart sollte das interne Register wie folgt gesetzt werden:

Register	Funktion	PAL	SECAM	NTSC
0	Horizontal insgesamt	63	63	63
1	Horizontal dargestellt	40	40	40
2	Horizontale Sync.Posn.	46	46	46
3	Vsync., Hsync. Breite	#8E	#8E	#8E
4	Vertikal insgesamt	38	38	31
5	Vertikal-Einstellung insgesamt	0	0	6
6	Vertikal dargestellt	25	25	25
7	Vertikal Sync. Posn.	30	30	27
8	Interlace und Skew	0	0	0
9	Max. Raster-Adresse	7	7	7
10	Cursor-Start-Raster	X	X	X
11	Cursor-End-Raster	X	X	X
12	Anfangs-Adresse (H)	X	X	X
13	Anfangs-Adresse (L)	X	X	X
14	Cursor (H)	X	X	X
15	Cursor (L)	X	X	X

In der obigen Tabelle ist die Anzahl für PAL- und SECAM-Standards gleich. X in der Tabelle weist darauf hin, daß die Software während des Betriebes diese Werte verändern kann. Die Firmware verwendet lediglich das Start-Address-Register, welches zum Einstellen von Bildschirm-Basis und -Offset verwendet wird.

F. Video-Gate-Array

Die Software muß auf dieses Gate-Array zugreifen, um die Freischaltung und Sperrung der ROM's, die Betriebsart des VDU und das Laden der Farb-Information für die Inks im Paletten-Speicher zu steuern.

Ein Ein-/Ausgabe-Kanal wird für alle Befehle verwendet, deren obere zwei Daten-Bits den Befehl wie folgt spezifizieren:

Bit 7	Bit 6	Verwendung
0	0	Lade Paletten-Pointer-Register.
0	1	Lade Paletten-Speicher.
1	0	Lade Modus- und ROM-Freischalt-Register.
1	1	Reserviert.

MODUS- UND ROM-FREISCHALT-REGISTER

Dieses lediglich beschreibbare Register steuert den VDU-Modus und die ROM-Freischaltung wie folgt:

Bit 7: 1
Bit 6: 0
Bit 5: Reserviert (sende 0).
Bit 4: Lösche „52-Abtast-Perioden-Zähler".
Bit 3: Sperre die obere Hälfte des ROM's.
Bit 2: Sperre die untere Hälfte des ROM's.
Bit 1: VDU-Modus-Steuerung MC1.
Bit 0: VDU-Modus-Steuerung MC0.

Das Schreiben einer 1 auf das Bit 4 löscht das oberste Bit des „52-Abtast-Perioden-Zählers", der für die Erzeugung periodischer Unterbrechungen zuständig ist.

Der Modus wird durch den Modus-Steuer-Pin wie folgt gesteuert:

MC1	MC0	Modus
0	0	Modus 0, 160 x 200 Pixel in 16 Farben.
0	1	Modus 1, 320 x 200 Pixel in 4 Farben.
1	0	Modus 2, 640 x 200 Pixel in 2 Farben.
1	1	Nicht verwenden.

Die Gate-Array-Hardware synchronisiert Modus-Änderungen mit der nächsten horizontalen Rücklauf-Periode, um die Software, die unterschiedliche Bereiche des Bildschirms in verschiedenen Modi handhabt, zu unterstützen.

Beim Power-up und anderen System-Resets wird das Modus- und ROM-Freischalt-Register auf Null gesetzt, wodurch die beiden Hälften des ROM's freigeschaltet sind.

Paletten-Pointer-Register

Dieses lediglich beschreibbare Register steuert das Laden der VDU-Farb-Palette wie folgt:

Bit 7:	0
Bit 6:	0
Bit 5:	Reserviert (sende 0).
Bit 4:	Paletten-Pointer-Bit PR4.
Bit 3:	Paletten-Pointer-Bit PR3.
Bit 2:	Paletten-Pointer-Bit PR2.
Bit 1:	Paletten-Pointer-Bit PR1.
Bit 0:	Paletten-Pointer-Bit PR0.

Die Bits PR0 bis PR3 selektieren die zu ladende Ink-Farbe, vorausgesetzt Bit PR4 ist im Low-Zustand. Wenn Bit PR4 im High-Zustand ist, werden die Bits PR0 bis PR3 ignoriert und die Bildschirmrand-Ink-Farbe geladen.

Paletten-Speicher

Dieses lediglich beschreibbare Register steuert die VDU-Farb-Palette wie folgt:

Bit 7:	0
Bit 6:	1
Bit 5:	Reserviert (sende 0).
Bit 4:	Farb-Datenbit CD4.
Bit 3:	Farb-Datenbit CD3.
Bit 2:	Farb-Datenbit CD2.
Bit 1:	Farb-Datenbit CD1.
Bit 0:	Farb-Datenbit CD0.

Der Ink-Eintrag, auf den durch das Paletten-Pointer-Register gezeigt wird, ist mit der auf diesem Kanal zu sendenden Farbe geladen. Die Anzahl der zu ladenden Farben reicht von 2 Farben in Modus 2 bis 16 Farben im Modus 0. Zusätzlich zum Laden der Farben muß ein Extra-Datenbyte auf diesem Kanal gesendet werden, welches die Farbe des Bildschirmrandes definiert. Beim Power-up und anderen System-Resets ist der Inhalt der Palette undefiniert. Nur die Farbe des Bildschirmrands ist auf schwarz gesetzt, um beim Power-up unansehnliche Effekte zu vermeiden.

Die 32 Farb-Codes werden dekodiert, um die RGB-Signale zu dekodieren, welche 27 unterschiedliche Farben erzeugen können. Die Hardware-Farben sind in Anhang V aufgelistet.

G. Parallel Peripheral Interface µPD8255

Der PPI wird ebenso wie die 8 Port-Pins des PSG als Schnittstelle zur Tastatur, sowie zur Steuerung und Abfrage verschiedener Signale auf der Prozessor-Karte verwendet. Port A muß im Modus 0 entweder auf Eingabe oder Ausgabe programmiert sein, da dieses Port für das Beschreiben oder Lesen des PSG verwendet wird.

Verschiedene Schaltkreise sorgen für ein Rücksetzen des PPI während des System-Reset. Weitere Details zur Arbeitsweise des µPD8255 befinden sich in den NEC-Produkt-Spezifikationen.

Kanal A (Ein- oder Ausgabe)

Bit 7:	Daten/Adresse DA7 mit dem AY-3-8912 verbunden.
Bit 6:	Daten/Adresse DA6 mit dem AY-3-8912 verbunden.
Bit 5:	Daten/Adresse DA5 mit dem AY-3-8912 verbunden.
Bit 4:	Daten/Adresse DA4 mit dem AY-3-8912 verbunden.
Bit 3:	Daten/Adresse DA3 mit dem AY-3-8912 verbunden.
Bit 2:	Daten/Adresse DA2 mit dem AY-3-8912 verbunden.
Bit 1:	Daten/Adresse DA1 mit dem AY-3-8912 verbunden.
Bit 0:	Daten/Adresse DA0 mit dem AY-3-8912 verbunden.

Kanal B (Nur Eingabe)

Bit 7:	Lese Daten von Datacorder-Kassette.
Bit 6:	Centronics Busy-Signal.
Bit 5:	Aktiv-Signal des Erweiterungs-Ports.
Bit 4:	Option-Link LK4.
Bit 3:	Option-Link LK3.
Bit 2:	Option-Link LK2.
Bit 1:	Option-Link LK1.
Bit 0:	Bildrücklauf-Impuls.

Die Option-Links LK1...LK4 werden vom Hersteller gesetzt. LK4 ist auf 60 Hz-TV-Standards eingestellt, der 50 HZ-Standard ist offen.

Kanal C (Nur Ausgabe)

Bit 7:	AY-3-8912 BDIR-Signal.
Bit 6:	AY-3-8912 BC1-Signal.
Bit 5:	Schreibe auf Datacorder-Kassette.
Bit 4:	Kassetten-Motor des Datacorders einschalten.
Bit 3:	Tastatur-Reihen-Auswahl KR3.
Bit 2:	Tastatur-Reihen-Auswahl KR2.
Bit 1:	Tastatur-Reihen-Auswahl KR1.
Bit 0:	Tastatur-Reihen-Auswahl KR0.

H. Centronics-Port-Latch

Dieses Latch wird durch Ausgabe-Befehle an den entsprechenden Ein-/Ausgabe-Kanal mit Daten geladen. Es ist nicht lesbar. Die Timing-Anforderungen der Centronics-Schnittstelle spezifizieren allgemein, daß die Daten auf den sieben Daten-Leitungen mindestens 1 Mikrosekunde vor dem Strobe (Übergabe-Impuls) aktiv sind und mindestens 1 Mikrosekunde nach dem Strobe aktiv bleiben. Die Länge des Strobe-Impulses muß zwischen 1 und 500 Mikrosekunden liegen. Das Busy-Signal kann, um den Zeitpunkt für eine weitere Daten-Ausgabe zu bestimmen, abgefragt werden, sobald der Strobe-Impuls inaktiv ist.

Bit 7:	Centronics-Strobe-Signal (1 = aktiv).
Bit 6:	Datenbit 7 zum Centronics-Port.
Bit 5:	Datenbit 6 zum Centronics-Port.
Bit 4:	Datenbit 5 zum Centronics-Port.
Bit 3:	Datenbit 4 zum Centronics-Port.
Bit 2:	Datenbit 3 zum Centronics-Port.
Bit 1:	Datenbit 2 zum Centronics-Port.
Bit 0:	Datenbit 1 zum Centronics-Port.

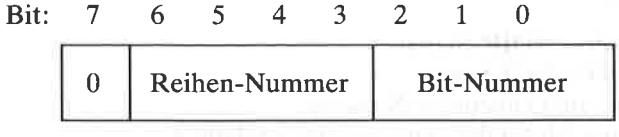
Beim Power-up und anderen System-Resets werden die Latch-Ausgänge gelöscht.

I. Tastatur und Joysticks

Die Tastatur und Joystick-Schalter werden durch die Auswahl einer von zehn Reihen unter Verwendung der vier Steuerbits von Kanal C des PPI, sowie Lesen der Daten vom PSG-Parallel-Port unter Verwendung von Port A des PPI abgetastet.

Die Tastatur und Joystick-Schalter sind in einer 10 x 8-Matrix angeordnet. Eine der zehn Reihen wird durch den Code auf KR0...KR3 selektiert. Die acht Datenbits werden daraufhin, wie oben beschrieben, vom Parallel-Port gelesen. Ein Schalter ist aktiv (geschlossen), wenn das zugehörige Datenbit logisch 0 ist.

Die jeder Taste zugeordnete Tasten-Nummer (siehe Anhang I) ist wie folgt aufgebaut:



Die dem Bit 5 in Reihe 4 zugeordnete Taste hat beispielsweise die Tasten-Nummer 37 (4*8+5).

Stichwortverzeichnis