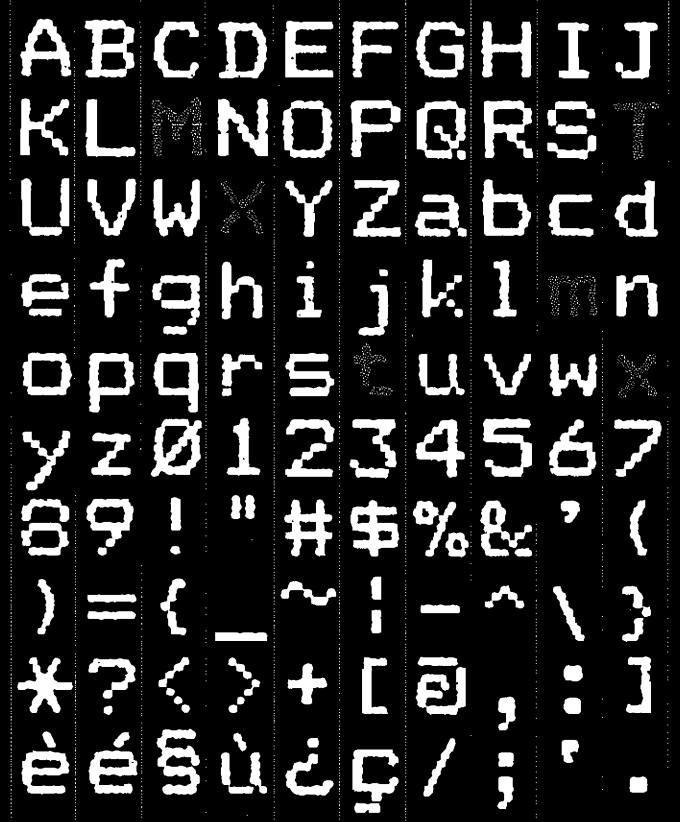
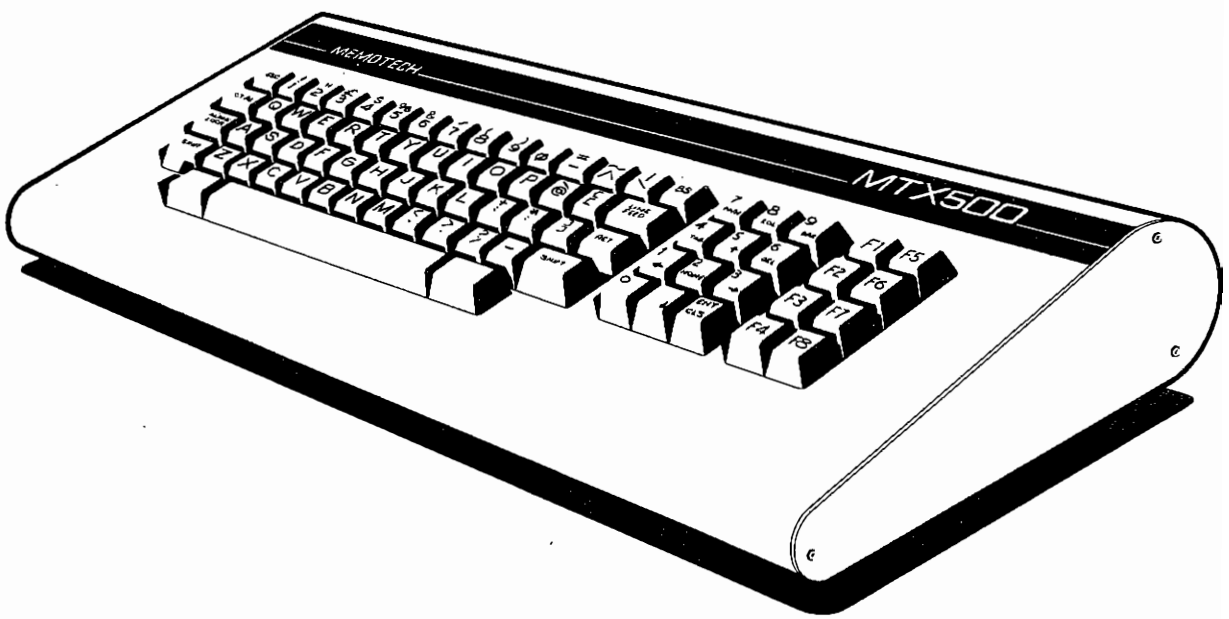


MEMOTECH

BASIC-LEHRGANG
TECHNISCHES HANDBUCH

MTX
SERIES

MTX-500/512
Handbuch



ISBN-Nr. 3-923985-03-7

Brian Pritchard (Principal Psychologist, TRC, OXFORD).

INHALTSVERZEICHNIS

TEIL 0

ES WIRD AUSGEPACKT

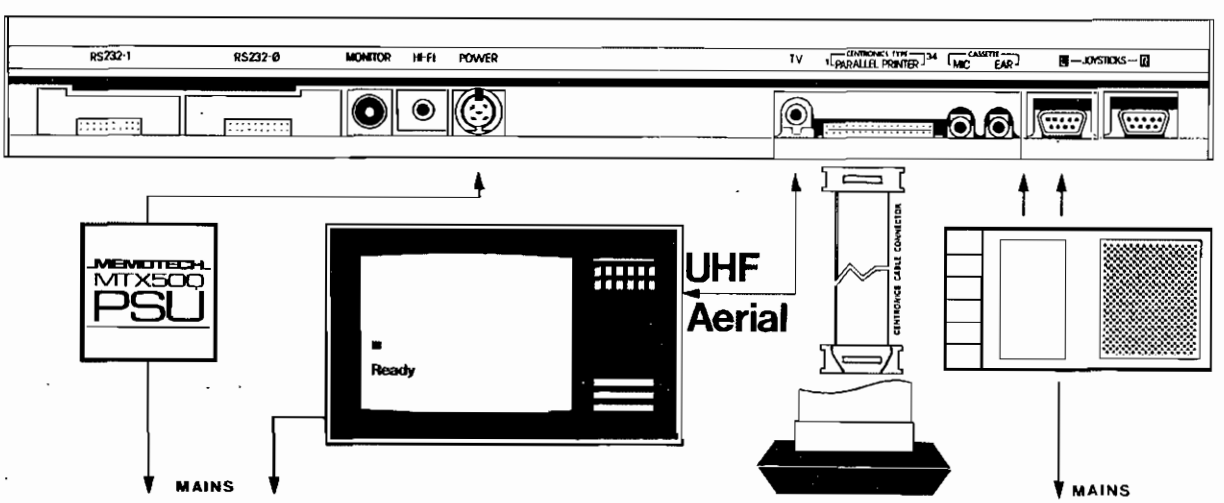
Wie Sie bereits gesehen haben werden, enthält der MTX-Karton mehr als nur einen Computer.

Zum gesamten Lieferprogramm gehören:

1. Der MTX-Computer.
2. Das MTX-Netzteil (PSU).
3. Zwei Verbindungskabel zum Cassettenrecorder.
4. Einen UHF-TV-Anschlußkabel.
5. Je eine Demonstrations-, Kopfreinigungs- und Leerkassette.
6. Eine aufsteckbare Schutzkappe für den Bus-Port.
7. Zwei Spiele von Continental Software.
8. Eine Garantiekarte

Sie brauchen lediglich ein Fernsehgerät und einen handelsüblichen Cassettenrecorder mit 3,5 mm Klinkenbuchsen.

Die Anschlußbelegung des Computers mit Netzteil, Cassettenrecorder und Fersehgerät verdeutlicht nachfolgende Abbildung.



Der erste Schritt besteht in der Überprüfung der korrekten Anschlüsse der gesamten Peripherie. Sobald der Computer mit dem TV verbunden ist, muß die Empfangsabstimmung so lange justiert werden, bis die Meldung "Ready" links unten vor einem blauen Hintergrund deutlich erscheint. Die meisten Fernsehgeräte haben leicht zugängliche Stationstasten, von denen einige noch nicht belegt sind.

Ihr MTX-Computer wurde als Vielzweck-Arbeitsmaschine entwickelt; sie können daher unter normalen Umständen eine einwandfreie Leistung des Gerätes erwarten. Zugleich ist es aber auch ein hochkompliziertes Gerät der Elektronik und bedarf gewisser Pflege. Vermeiden Sie bitte Arbeitsbedingungen, bei denen Flüssigkeiten oder extreme Hitze auf das Gerät einwirken können.

Zur Reinigung des MTX empfehlen wir einen sauberen Lederlappen.

Das MTX-Handbuch wurde für den völligen Laien geschrieben. Zugleich bieten wir dem geübten Programmierer datailliertere technische Informationen zum Gebrauch des MTX-BASIC, der Grafik, NODDY und dem Assembler. Sollten Sie noch Anfänger sein, empfiehlt sich das gewissenhafte Abarbeiten jedes einzelnen Kapitels. Ansonsten sollten die Anhänge zur Systembeschreibung für den geübten Programmierer ausreichen.

EINLEITUNG

Der nun folgende Programmierkurs wurde so geschrieben, daß auch der völlig kenntnislose Computerlaie folgen kann. Selbst wenn Sie bereits über BASIC-Kenntnisse verfügen, ist es sinnvoll, die Regeln des MTX-BASICs zu studieren. Im gesamten Verlauf dieses Kurses werden Übungsabschnitte mit stetig zunehmendem Schwierigkeitsgrad angeboten. Am Ende des Kurses angelangt, werden Sie feststellen, daß Sie über detaillierte Programmierkenntnisse verfügen.

Digitale Computer sind in der Lage, große Datenmengen (kurz DATA genannt) zu speichern und diese Daten mittels einfachster Rechenschritte mit sehr hoher Geschwindigkeit zu verarbeiten. Um zu einem Ergebnis zu gelangen, muß der Anwender dem Computer eine Folge von Anweisungen geben, die man in ihrer Gesamtheit als PROGRAMM bezeichnet.

Jede dieser Anweisungen muß genau formuliert werden, um von dem Computer "verstanden" zu werden. Diese Präzisierung drückt sich in der sogenannte PROGRAMMIERSPRACHE aus.

In der Betriebsstruktur Ihres MTX sind bereits vier PROGRAMMIERSPRACHEN integriert: MTX-BASIC, Z80 Assembler, NODDY und MTX-GRAFIK. NODDY ist eine völlig neue Sprache, von der Sie sicherlich noch nicht gehört haben.

Am ehesten läßt sich NODDY mit LOGO vergleichen. So, wie diese Sprache die Grafikdarstellung erleichtert, vereinfacht NODDY den Umgang mit Texten. MTX-GRAFIK besteht aus einem umfassenden Grafikbefehlssatz zur Implementierung von LOGO und für den Entwurf von anspruchsvoller Grafik. Aber davon später mehr. Sie werden die Anleitung zu NODDY hinter MTX-GRAFIK in den Kapiteln 2, 3 und 4 vorfinden. NODDY erfordert keinerlei Vorkenntnisse aus den nachfolgenden Kapiteln, so daß Sie, falls erwünscht, diese auslassen können.

Die von Ihnen als erstes zu erlernende Programmiersprache dürfte dennoch BASIC sein. Der Begriff stammt aus der Abkürzung Beginner All-Purpose Symbol Instruction Code. Es ist ratsam, zunächst jedes Kapitel vollständig - einschließlich der Übungsabschnitte - zu erarbeiten, ehe das nächste angefangen wird. Auf diese Weise wird vermieden, daß Sie sich hoffnungslos verzetteln und mutlos werden.

Alle Abschnitte des Kurses sind ähnlich gegliedert. Das LERNZIEL beschreibt das zu lösende Problem und erklärt den Nutzen des gelernten Sachverhaltes.

In den Abschnitten mit PROGRAMMEN finden Sie z.T. erstellte Flußdiagramme, die von Ihnen zu vervollständigen sind. Sie werden auch Testprogramme zum Üben vorfinden, sowie eine Übung, die der Erfolgskontrolle dient.

Das MTX-Tastenfeld ist in drei Bereiche unterteilt. Der größte, linke Block ist die eigentliche Schreibmaschinentastatur. Sollten Sie mit einer solchen Tastatur nicht vertraut sein, empfiehlt sich die Eingabe einiger Sätze, um die Arbeitsweise der Tastatur kennenzulernen. Sollten Sie zufällig einen für den Computer verständlichen BEFEHL eintippen, kann eine Fehlermeldung im linken unteren Bildabschnitt erscheinen . . . aber keine Angst. Der MTX kann auf diese Weise nicht beschädigt werden. Entdecken Sie die Funktionen der SHIFT und ALPHA LOCK Taste und entwickeln Sie ein Gespür für das Tastenfeld. Jede Taste wird, wenn sie länger gedrückt bleibt, das entsprechende Zeichen wiederholen. Dies nennt man AUTO-REPEAT.

Wenn Sie im BASIC-Modus arbeiten, können bis zu vier Zeilen im sogenannten Editierbereich am unteren Bildschirmrand eingegeben werden.

Es folgt nun ein Beispiel mit NODDY, mit das Sie am einfachsten den ganzen Bildschirm nutzen können.

Eingabe: NODDY und die (RET)-Taste drücken.

NODDY> erscheint links unten und der Computer wartet nun auf eine Seitenbenennung. In diesem Fall wollen wir die Seite AN nennen.

Eingabe: AN und die (RET)-Taste drücken.

Nun drücken Sie bitte vor Fertigstellung der Bildschirmseite nicht mehr die (RET)-Taste. Tippen Sie nun einfach beliebige Zeilen ein, bis der Bildschirm voll ist. Für die Positionierung des Kursors können Sie die Kursortasten auf dem mittleren Tastenfeld verwenden.

Nun bewirkt das Drücken der (RET)-Taste die erneute Anzeige von NODDY> links und die Eingabemöglichkeit für eine neue Seite, die BB genannt werden soll. Sollten Sie Tastaturfunktionen auslösen, die nicht verständlich sind, kann mittels RESET (Drücken der beiden nicht beschrifteten Tasten rechts und links der Leertaste) der Computer wieder zur Ausgangslage zurückgestellt werden.

Das alphanumerische Tastenfeld enthält einige Tasten, die nicht auf einer Schreibmaschine zu finden sind. Z.B. finden Sie zwei nicht bezeichnete Tasten rechts und links der Leertaste. Dies sind die RESET-Tasten, die - gleichzeitig gedrückt - den Computer in den Zustand versetzen, als wäre er gerade erst eingeschaltet worden. Dies kann dann von Nutzen sein, wenn man gravierende Fehler gemacht hat und neu beginnen möchte. Weniger hilfreich wäre es jedoch, versehentlich den Unterarm auf dem Tastenfeld abzustützen, um ein aufwendiges Programm zu löschen.

* (RET)-Taste: Diese Taste muß immer zum Abschluß einer Eingabe erfolgen; sie bewirkt erst die Freigabe des Befehls an den Rechner.

Die beiden anderen für Sie vielleicht neuen Tasten sind ESC (ESCAPE) und CTRL (CONTROL). Diese werden gelegentlich beim Programmablauf benutzt und erst später näher erklärt.

Das zweite Tastenfeld, bestehend aus zwölf Tasten, enthält die Editier- und numerischen Tasten für den EDITOR und für die bequeme Zifferneingabe. Der EDITOR wird beim MTX zur Kursorsteuerung, Fehlerkorrektur und als Joysticktastenfeld für Spiele eingesetzt.

Tippen Sie noch einige Zeichen in eine NODDY-Seite - wie zu Beginn dieses Abschnitts angegeben - und testen Sie dann in NODDY oder im BASIC die Funktionen der EDITOR-Tasten wie nachfolgend beschrieben:

Steuertasten des Kursors

← steuert den Cursor einen Schritt nach links.

↑ steuert den Cursor eine Zeile höher. Diese Taste funktioniert nicht im Editierbereich, da der Computer wie bei EOL diesen Bereich nur als eine Zeile ansieht, d.h., daß Sie zwar nach rechts und links, aber nicht nach oben und unten steuern können.

↓ steuert den Cursor nach unten (s. die Anmerkung zur Steuerung nach oben).

→ steuert den Cursor einen Schritt nach rechts.

TAB steuert den CURSOR in 8er-Schritten über den Bildschirm. Eine nützliche Taste, um z.B. Tabellen anzulegen.

HOME bringt den Cursor zurück zum ersten Zeichen des jeweils angesprochenen Bildschirms.

Löschtasten

DEL (DELETE) löscht das Zeichen, auf dem der Kursor gerade steht.

CLS (CLEAR SCREEN) löscht den z.Z. angesprochenen Bildschirm. Eine schnelle Funktion, wenn eine neue Seite begonnen werden soll. Allerdings sollte vorsichtig mit dieser Taste gearbeitet werden, damit nicht versehentlich zuviel gelöscht wird.

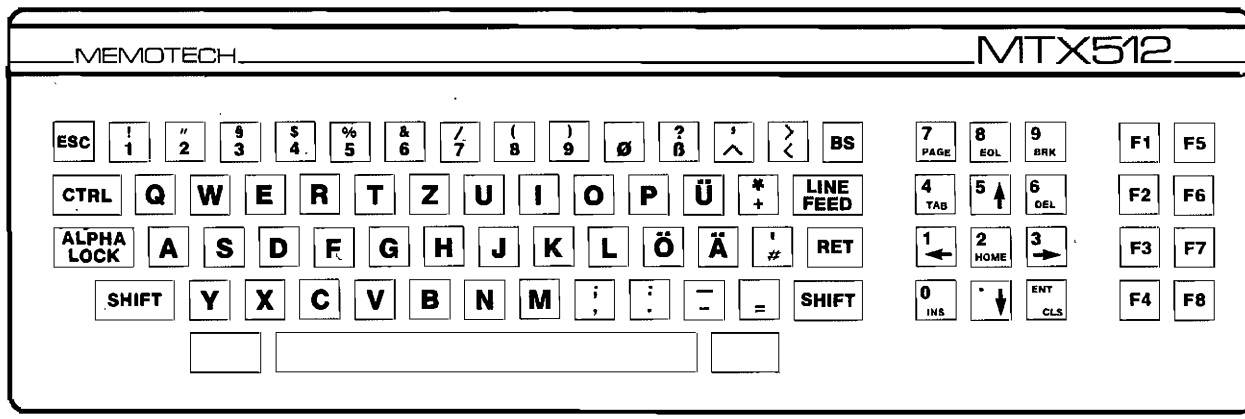
INS (INSERT) erlaubt das nachträgliche Einfügen von Text, ohne den bereits geschriebenen zu löschen. Zum Einfügen einfach INS drücken und den gewünschten Text eintippen. Der bereits vorhandene Text wird dabei nach rechts weitergeschoben. Der INSERT-Modus bleibt erhalten, bis die INS-Taste nochmals gedrückt wird.

EOL (End Of Line) löscht alle Zeichen rechts vom Kursor bis zum Zeilenende. Da, wie schon erwähnt, im BASIC-Modus die vier 'Zeilen' im Editierbereich oder auch EDIT-Feld eigentlich aus einer Zeile bestehen, werden hier alle Zeichen rechts und unterhalb des Kursors mit EOL gelöscht.

Programmkontrolltasten

BRK wird dazu eingesetzt, ein ablaufendes BASIC-Programm zu unterbrechen. Der Kursor erscheint daraufhin normal über der Ready-Meldung.

PAGE wird genutzt, um Programmlistings zu unterbrechen und um zwischen dem PAGE- und SCROLL-Modus zu wählen. Diese Begriffe werden später noch erklärt.



Das dritte, rechte Tastenfeld wird Funktionstastenfeld genannt. Die vorhandenen Tasten sind dazu da, bestimmte, vom Anwender definierte Funktionen mit einem Knopfdruck auszuführen. So ließe sich z.B. ein von Ihnen entworfenes Spiel damit steuern. Obwohl nur 8 Tasten vorhanden sind, lassen sich dennoch mit Hilfe der SHIFT-Taste bis zu 16 Funktionen speichern. Die ALPHALOCK-Taste wirkt nicht auf dieses Tastenfeld. Wie eine Funktion den jeweiligen Tasten zugeordnet wird, entnehmen Sie bitte dem Software-Anhang 6.

Einige Ausdrücke im folgenden Kursus werden neu sein, aber die meisten sind im Verlauf des Handbuchs erklärt. Sollten Sie dennoch den einen oder anderen Begriff nicht verstehen, empfehlen wir den Erwerb entsprechender Handwörterbücher.

TEIL 1

DER BASIC-LEHRGANG

KAPITEL 1

DER EINSATZ VORHANDENER PROGRAMME

LERNZIEL: In diesem Kapitel soll die Bedeutung und der Gebrauch der Befehle LOAD, RUN, SAVE und VERIFY erlernt werden.

DAS EINLADEN EINES PROGRAMMS

Der erste Schritt hier ist, sicherstellen, ob der Computer korrekt angeschlossen ist (siehe Abb. in Teil 0). Sobald Sie nun einschalten, erscheint der Kursor in der HOME-Position links unten auf dem Bildschirm; weiterhin erscheint das Wort "READY" (bereit) auf der untersten Zeile.

Um ein Programm von einer Cassette in den Speicher des Computers zu laden, benutzen wir den Befehl LOAD. Schließen Sie den Cassettenrecorder an Ihren MTX an und geben nun den Befehl LOAD über die Tastatur ein. Nun muß ein Leerzeichen folgen (Leertaste drücken) und der Name des gewünschten Programms in Anführungszeichen. So würde z.B. die vollständige Ladeanweisung für die Programmkassette CHESS wie folgt auf dem Bildschirm erscheinen:

```
LOAD "CHESS"
```

Sollte der Name des Programms unbekannt sein, reicht auch die Eingabe: LOAD "". Damit würde das erste Programm auf der Cassette vom Computer geladen. Obwohl diese Methode funktioniert, ist es immer besser, den vollen Programmnamen einzugeben.

Nun können Sie die RET-Taste des Computers und die PLAY-Taste des Cassettenrecorders drücken. Der Computer wird nun die Anzahl der zu übernehmenden Zeichen ausrechnen und sie beim Ladevorgang auszählen. Sobald dieser erfolgreich abgeschlossen ist, wird folgende Anzeige auf dem Bildschirm erscheinen:

```
LOAD "CHESS"
FOUND CHESS
LOADING
■

Ready
```

Um das nun übertragene Programm auch zu aktivieren, muß dem Computer ein entsprechender Befehl gegeben werden. Der hierzu erforderliche Befehl heißt RUN.

Geben Sie nun RUN gefolgt von RET ein.

Manche Programe starten automatisch nach dem Laden. In diesen Fällen ist RUN natürlich überflüssig.

Möglicherweise möchten Sie ein Programm nochmals ablaufen lassen; in diesem Fall muß nicht mehr neu geladen werden - das Programm befindet sich ja noch im Arbeitsspeicher des Computers - sondern lediglich das Programm mittels nochmaliger Eingabe von RUN (RET) gestartet werden. Es ist jedoch sinnvoll, vorher den Bildschirm mit dem Befehl CLS und (RET) zu löschen.

Die CLS-Funktionstaste unterscheidet sich vom Befehl CLS insofern, als sie sich nur auf den Bildschirmbereich, in dem der Kursor z.Z. aktiv ist (meistens der Editierbereich), bezieht, während der Befehl CLS den ganzen Bildschirm löscht. Sollten Sie sich beim Editieren einmal hoffnungslos "verrannt" haben, kann die CLS-Taste als Nothelfer schnell Abhilfe verschaffen.

PROGRAMMLISTINGS

Das nachfolgend aufgeführte Listing eines kleinen Programms soll als Beispiel dazu dienen, wie ein Programm in der Form eines Listings auf Papier (z.B. aus Fachzeitschriften, Büchern o.ä.) in den Arbeitsspeicher (RAM) des Computers übertragen wird. Es kommt hierbei nicht darauf an, daß Ihnen alle Befehle, Zeichen etc. bekannt sind, sondern darauf, daß Sie sich mit der Maschine vertraut machen und lernen, ein Listing exakt zu übertragen.

Falls Sie den MTX-Arbeitsspeicher bereits mit anderen Programmen oder Übungsbeispielen belegt haben, ist es ratsam, diese zuvor mit dem Befehl NEW zu löschen. Der Befehl NEW dient dazu, sämtliche Speicherinhalte zu löschen, so daß nach der Eingabe des Listings nur dieses Programm im Speicher präsent ist. Dieser Befehl ist immer dann angebracht, wenn ein Programm abgeschlossen ist und nicht mehr gebraucht wird.

```
10    REM PROGRAMMEINGABE
20    PRINT "Wie heißen Sie?"
30    INPUT N$
35    PRINT : PRINT
40    PRINT "Wie alt sind Sie?"
50    INPUT A
60    CLS
70    PRINT N$;" ist";A;" Jahre alt"
80    PRINT : PRINT
90    PRINT "NOCHMALS? (J/N)"
100   INPUT M$
110   IF  M$="J" OR M$="j" THEN GOTO 10 ELSE STOP
```

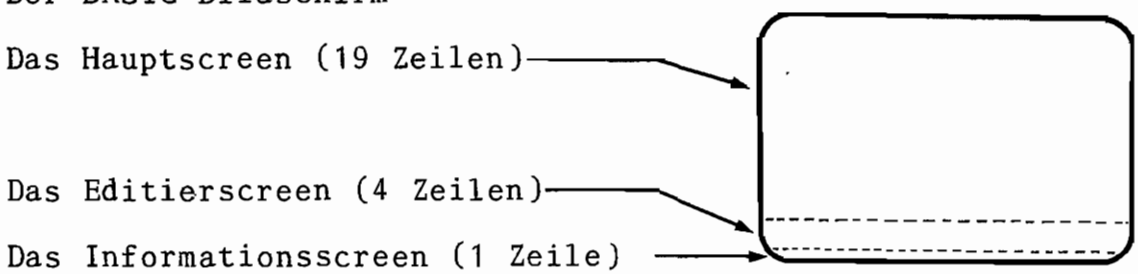
Geben Sie nun das obige Listing ein und vergessen Sie dabei bitte nicht, daß die Vorlage genau übertragen und daß jede Zeile mittels der RET-Taste quittiert werden muß. Sollte ein Programm nach genauer Überprüfung dennoch nicht laufen, ist der Befehl NEW mit einer nochmaligen Eingabe angebracht.

DIE BASIC-SCREENS

Bei der Eingabe des obigen Listings werden Sie bemerkt haben, daß der Bildschirm in verschiedene Bildschirmbereiche, die von nun an SCREENs genannt werden, eingeteilt ist.

Der gesamte Bildschirm, bestehend aus 24 Zeilen, besteht aus drei verschiedenen Screens:

Der BASIC-Bildschirm



Eingegebene Daten werden zunächst im Editierscreen abgebildet. Erst, wenn Sie mit der Eingabe zufrieden sind und mittels der RETurn-Taste dem Computer mitteilen, daß die Daten übernommen werden sollen, wird der Inhalt des Editierscreens zum Hauptscreen übertragen. Aber die Übernahme der Daten hängt nicht nur von Ihrer Einschätzung der Richtigkeit ab, sondern auch davon, ob der Rechner die Daten interpretieren kann.

Sollten Sie Daten in einer für den Computer fehlerhaften Form eingegeben haben, wird eine Fehlermeldung in der Informationszeile erscheinen. Ferner wird der Cursor an die Stelle der Zeile gesetzt, die die Fehlermeldung verursacht hat.

Sollten Sie mit der Korrektur der fehlerhaften Eingabe Schwierigkeiten haben, können Sie auch einfach die CLS-Taste bedienen und mit der Eingabe fortfahren.

Ein Beispiel für die Fehlermeldung ist die Anzeige:

MISMATCH

Dies bedeutet, daß Sie einen Fehler gemacht haben und ferner, daß dies ein Syntax-Fehler ist. Syntaxfehler sind in der Computersprache solche Fehler, die aufgrund einer falschen Wortwahl entstehen, wenn das Wort an dieser beanstandeten Stelle nicht stehen darf oder wenn irgendein Wort falsch geschrieben wurde.

Eine Fehlermeldung dieser Art würde z.B. vorkommen, wenn Sie folgendes Programm eingeben:

```
10 LET X$="Wie alt sind Sie?"
20 LET Y="21"
30 PRINT X$,Y
40 STOP
```

Probieren Sie es aus. Die Fehlermeldung erscheint dann, wenn Sie versuchen, die Zeile 20 in das Hauptscreen und damit in den Programmspeicher zu übertragen (mit RET). Der Grund liegt einfach darin, daß die "21" nicht als Zahl erkannt wird. Richtig muß es heißen:

```
20 LET Y=21
```

Es gibt verschiedene Möglichkeiten, diesen Fehler zu korrigieren. Die einfachste besteht darin, einfach die fehlerhafte Stelle zu überschreiben. Beim obigen Beispiel könnte man den Kursor über die Anführungszeichen setzen, um sie dann mit der DEL-Taste zu löschen. Den Kursor steuern Sie dabei mit den Steuertasten (Pfeil rechts, links). Wenn Sie nun meinen, den Fehler beseitigt zu haben, kann abermals mit der RET-Taste versucht werden, die Zeile zu übertragen. Wie Sie sehen, sind die Editiertasten des numerischen Tastenfeldes ideal für solche Korrekturarbeiten.

Um das Geschriebene zu überprüfen, können Sie mit dem Befehl LIST (RET) das Programm in der richtigen Reihenfolge auf dem Bildschirm abbilden. Es ist auch möglich mittels einer Erweiterung des LIST-Befehls nur bestimmte Abschnitte des Programms anzuzeigen. Obwohl unser Beispielprogramm so klein ist, daß hier kein großer Nutzen erkenntlich wird, wollen wir diese Variante kurz demonstrieren:

Eingabe: LIST 20,40

Sie sehen die Zeilen 20,30 und 40 auf dem Bildschirm.

Ähnlich bewirkt

```
LIST 30
```

daß das Programm ab der Zeile 30 abgebildet wird.

Da unser Beispiel nur sehr kurz ist, wird das gesamte Listing abgebildet. Da jedoch die Mehrzahl der Programme meist wesentlich länger sind und daher nur ausschnittsweise angezeigt werden, nimmt man die PAGE-Taste zur Hilfe.

Beginnt ein längeres Programm nach dem LIST-Befehl über den Bildschirm zu fahren, kann es mit der PAGE-Taste angehalten und mit abermaliger Betätigung wieder fortgeführt werden. Wenn die Anzeige des Listings mit der PAGE-Taste angehalten worden ist, wird mit Betätigung irgendeiner anderen Taste außer PAGE und BRK nur die nächste Seite dargestellt, so daß man damit ein Listing seitenweise 'durchblättern' kann. Diese Verfahrensweisen sind recht nützlich bei der Fehlersuche.

Nachdem Sie nun das Programm fehlerfrei eingegeben haben, kann es auch ausprobiert werden. (Da das Programm bereits im Arbeitsspeicher vorhanden ist, braucht es natürlich nicht mehr mit LOAD eingeladen werden.)

Eingabe: RUN (RET)

Sollten keine versteckten Fehler vorhanden sein, wird das Programm nun laufen. Es gibt jedoch auch Fehler, die nicht als Syntaxfehler erkannt werden, sondern nur als Laufzeitfehler auftauchen. Sie kommen dann vor, wenn zwar die richtigen Befehle eingegeben wurden, aber Daten, auf die sich die Befehle beziehen, nicht vorhanden oder falsch sind.

Wenn z.B. im Programm "NAME/ALTER" die Zeile 110 so abgeändert würde:

```
110 IF M$="J" OR M$="j" THEN GOTO 120 ELSE STOP
```

würde der Versuch, mit dieser Zeile zu arbeiten, vom Computer mit der Fehlermeldung

No Line

also 'keine Zeile' quittiert werden.

Da wir ja eine Zeile 110 haben, muß sich diese Meldung auf die Zeile 120, die mit dem GOTO-Befehl angesprungen werden soll, beziehen. Laufzeitfehler werden in der gleichen Art und Weise korrigiert wie schon die Syntaxfehler zuvor.

Aber auch die alternativen Editiermöglichkeiten sind es wert, angewandt zu werden.

So ist es z.B. möglich, ganze Programmzeilen neu zu schreiben und einzufügen. Ganz gleich, ob Sie bereits bei einer höheren Programmzeile angekommen sind, ist es möglich mit der Eingabe

```
50 INPUT A    (RET)
```

diese Zeile nachträglich an der richtigen Stelle einzufügen.

War bereits eine andere Eingabe in der Zeile 50, wird sie hiermit gelöscht. Auf diese Weise lassen sich auch ganze Zeilen löschen. So bewirkt die Eingabe

```
110      (RET)
```

daß eine eventuell vorhandene Zeile 110 gelöscht wird.

(Eine vollständige Liste der im MTX vorhandenen Fehlermeldungen finden Sie im Ergänzungsteil dieses Handbuchs.)

Wenn das Programm nun fehlerfrei übernommen wurde, auch zufriedenstellend läuft, werden Sie den Wunsch haben, es auf Cassette zu speichern.

DAS ABSPEICHERN EINES PROGRAMMS

Der englische Ausdruck für Abspeichern/Sichern ist SAVE, so daß wir wegen des gleichnamigen BASIC-Befehls SAVE oft als Synonym für Abspeichern benutzen werden.

Um ein Programm auf Cassette zu sichern, ist es erforderlich, den Computer wie beim LOAD-Vorgang mit einem Cassettenrecorder zu verbinden.

Zunächst muß eine Leercassette in den Recorder eingelegt werden. Wir empfehlen den Gebrauch von C10 oder C15 Cassetten mit je einem Programm pro Seite, so daß Ihre Programme immer leicht auffindbar bleiben. Sollten Sie jedoch längere Bänder benutzen wollen, ist es sinnvoll, eine genaue Datei mit dem Zählerstand jedes einzelnen Programms zu führen. Stellen Sie den Recorder auf RECORD und drücken Sie die PAUSE-Taste, bis der Computer bereit ist.

Das im Arbeitsspeicher vorhandene Programm wird mit dem Befehl SAVE auf das Band übertragen.

Eingabe: SAVE "PROG 1"

Nun muß der Recorder gestartet werden (Wenn Sie bereits RECORD und PAUSE gedrückt haben, muß nur noch die Pausentaste entsichert werden, ansonsten reicht RECORD). Lassen Sie das Band ein wenig vorlaufen, da das Magnetband bei Spulvorgängen am Anfang und Ende immer ein wenig gedehnt wird und dies die Aufnahmequalität beeinträchtigen kann. Mit der Eingabe

RET

beginnt jetzt die Übertragung des Programms auf die Cassette. Wie Sie eben gesehen haben, wurde zusätzlich zum Befehl SAVE auch ein Programmname in Anführungszeichen hinzugefügt. Damit kann der Rechner beim neuerlichen Einlesen (LOAD) erkennen, ob dies auch das richtige Programm ist. Es empfiehlt sich daher, sinnvolle Namen zu benutzen und diese auch auf der Cassette zu dokumentieren.

Sobald der SAVE-Vorgang beendet ist, ist folgende Meldung auf dem Bildschirm zu sehen:

```
SAVE "PROG 1"


Ready
```

Es ist nun angebracht, die auf Band abgespeicherte Version zu überprüfen. Dies geschieht mit Hilfe des Befehls VERIFY. Spulen Sie die Cassette zum Anfang zurück.

Eingabe: VERIFY "PROG 1" (RET)

Wenn Sie nun den Recorder mit PLAY starten, wird das aufgezeichnete Programm Zeichen für Zeichen mit dem Original im Arbeitsspeicher verglichen. Sobald der Vergleich beendet ist, muß folgende Meldung auf dem Bildschirm stehen:

```
VERIFY "PROG 1"
FOUND PROG 1
VERIFYING


Ready
```

Damit haben Sie Ihr erstes Programm erfolgreich auf Band gespeichert und überprüft.

FARBE

Ehe wir nun etwas schwierigere Materie angehen, wollen wir zunächst zeigen, wie die Bildschirmfarben gesteuert werden können. Der Bildschirmhintergrund wird PAPER (Papier) genannt und mit der gleichen Logik folgt, daß der Vordergrund INK (Tinte) heißt.

Denken Sie sich den Bildschirm einfach als Schreibblock und schon werden diese Definitionen verständlich. Während die Anzeige im EDIT-und COMMAND-Modus immer blau-weiß ist, können die Farben des resultierenden Programms beliebig bestimmt werden.

```
Eingabe: 10 PRINT : PRINT
         20 PRINT : PRINT
         30 PRINT " A A A A A A A A A A A A A A A A A A A A A A"
         40 PAPER 7
         50 INK   1
         60 PAUSE 8000
         RUN
```

Die Farben des Programms werden sich für ca. 8 Sekunden ändern. Jede der 16 vom Computer generierbaren Farben hat eine Ziffer (siehe Farbtafel im Anhang). Weitere Farben lassen sich wie folgt darstellen:

Eingabe: EDIT 40 (RET)

Sie haben nun die Zeile 40 im Editierfeld. Mit der Kursorsteuerung und DELETE lassen sich nun sämtliche Farben im obigen Programm ausprobieren. Ersetzen Sie die '7' durch '6', drücken RET und anschließend RUN (RET). Der Hintergrund wird nun dunkelrot.

Mit diesem Test können Sie zugleich feststellen, ob Ihr TV oder Monitor richtig abgestimmt ist.

FARBTAFEL

0	Transparent
1	Schwarz
2	Grün
3	Hellgrün
4	Dunkelblau
5	Hellblau
6	Dunkelrot
7	Cyan
8	Rot
9	Hellrot
10	Dunkelgelb
11	Hellgelb
12	Dunkelgrün
13	Magenta
14	Grau
15	Weiß

KAPITEL 2

ARITHMETISCHE AUSDRÜCKE

LERNZIEL: Den Computer mit Hilfe der Operationen Addition, Subtraktion, Multiplikation und Division als einfachen Rechner zu nutzen.

Zunächst untersuchen wir den PRINT-Befehl und seinen Einsatz bei der Darstellung einfacher Rechenoperationen. Der PRINT-Befehl bringt beliebige Werte auf den Bildschirm zur Anzeige, die dieser unmittelbar folgen. So bewirkt der Befehl 'PRINT 21' einfach die Darstellung der Zahl 21 auf dem Bildschirm. Versuchen Sie es einmal.

Eingabe: PRINT 21 (RET)

Wollen Sie Zahlenkolonnen anzeigen, müssen die Werte mit einem Komma getrennt werden.

Eingabe: PRINT 3,4,5

Jede Zahl wird an der nächstverfügbaren Tabulatorstelle dargestellt. Die Tab.-Positionen haben einen Abstand von 8 Zeichen.

Eingabe: PRINT 2+2 (RET)

Das Ergebnis 4 erscheint auf dem Bildschirm. Ähnlich funktioniert der Rechenschritt mit dem Minuszeichen.

Eingabe: PRINT 7-4 (RET)

Das Minuszeichen befindet sich auf der gleichen Taste wie das "="-Zeichen in der unteren Tastaturreihe.

Die Zeichen für Multiplikation und Division weichen jedoch von den in der Schule sonst gebräuchlichen ab.

Innerhalb kürzester Zeit werden Sie sich jedoch auch hieran gewöhnt haben.

Eingabe: PRINT 8*3

bewirkt eine Multiplikation von 8 und 3 und das Ergebnis 24 wird auf dem Bildschirm sichtbar.

PRINT 6/3 bewirkt die Division von 6 durch 3 und das Ergebnis 2 wird angezeigt.

2+2 ist ein Beispiel für einen arithmetischen Ausdruck. Bei jedem der obigen Beispiele wird der PRINT-Befehl dazu eingesetzt, das Ergebnis eines solchen Ausdrucks anzuzeigen.

****

ÜBUNG 1 ARITHMETISCHE AUSDRÜCKE

Setzen Sie den PRINT-Befehl ein, um die Werte folgender Ausdrücke anzuzeigen:

1) 2 + 2	2) 15 + 35	3) 288 + 397	4) 7945 + 3538
5) 7 - 4	6) 28 - 14	7) 654 - 289	8) 7986 - 3572
9) 8 + 3	10) 16 * 4	11) 244 * 6	12) 387 * 28
13) 6 / 2	14) 60 / 5	15) 288 / .06	16) 1080 / 72

****

Versuchen Sie sich an einigen Aufgaben. Beachten Sie dabei, daß das Ergebnis, sofern nicht ganzzahlig, automatisch als Dezimalzahl dargestellt wird.

KAPITEL 3

RECHENHIERARCHIE

LERNZIEL: Kenntnis über Reihenfolge der arithmetischen Operationen.

Im zweiten Kapitel setzten Sie den Computer als einfachen Taschenrechner ein. Jetzt werden Sie sich wahrscheinlich fragen, was der MTX sonst noch kann. (Es gibt inzwischen Taschenrechner für über DM 1000,-, obwohl natürlich die Tastatur nicht so gut ist). In diesem Abschnitt erfahren Sie mehr über mathematische Operationen; ferner werden wir den PRINT-Befehl einsetzen, um STRINGs (der Begriff wird später erläutert) zu behandeln. Nachdem Sie gelernt haben, den Bildschirm als Notizblock zu gebrauchen, können Sie nun kurze, einfache Programme schreiben.

Unser erstes Problem besteht darin, wie die Potenz einer Zahl so geschrieben werden kann, daß der Computer sie versteht. Nun, der Ausdruck:

3^2 wird mittels

PRINT 3^2 im MTX-BASIC dargestellt.

Die zweite Potenz wird als 2 geschrieben. Die Lösung des obigen Beispiels ist natürlich 9. Versuchen Sie folgende Beispiele:

****

ÜBUNG 2 POTENZEN

$9 = ?^2$ $20 = ?^2$ $10 = ?^3$ $2 = ?^{12}$

****

Die Berechnung in Kapitel 2 und die obigen Potenzen erfordern nur einen Rechenschritt. Versuchen Sie nun folgende Aufgabe:

2/3*6

Die Lösung lautet 4, da der Computer von links nach rechts rechnet und der folgenden Rechenhierarchie nachkommt.

Symbol	Operation	Beispiel
^	Potenzierung	4^2=16
* und /	Multiplikation und Division	3*2=6 6/3=2
+ und -	Addition und Subtraktion	3+2=5 3-2=1

Jedesmal, wenn der Computer eine Berechnung durchführen soll, arbeitet er von links nach rechts unter Berücksichtigung der genauen Hierarchie. Beachten Sie folgendes Beispiel:

```
          2
3 + 7 * 5 + 4 / 2 - 6 * 2
```

Um das Ergebnis anzuzeigen, geben Sie folgendes ein:

```
PRINT 3+7*5^2+4/2-6*2
```

Als Ergebnis bekommen wir 168. Der Computer kommt wie folgt zu dieser Zahl:

```
3+7*5^2+4/2-6*2    1. Schritt: Potenzierung

     25

3+7*25+4/2-6*2     2. Schritt: Multiplikation und Division

  175   2  12

3+175+2-12         3. Schritt: Addition und Subtraktion

   =168
```

Versuchen Sie nun folgende Übungen:

****

ÜBUNG 3 RECHENHIERARCHIE

Unterteilen Sie folgende Beispiele in einzelne Rechenschritte wie im vorangegangenen Beispiel. PRINTen Sie dann die Ergebnisse. Das erste Beispiel ist bereits z.T. gelöst.

1)

```
8 * 2^2 + 8 / 2^2 - 9^2 / 3

     ?         ?     ?

8 * ? + 8 / ? - ? / 3

32 + ? - 27

=7
```

2)

```
 38 + 64 + 6 / 3 - 6 ^ 2 / 3
```

3)

```
 16 + 18 / 3 ^ 2 - 2 * 3 ^ 2
```

4)

```
 64 - 7 ^ 2 + 6 ^ 2 * 2 - 2 ^ 6
```

****

Wird die Reihenfolge der Berechnung geändert, ergibt sich eine völlig andere Lösung. Betrachten wir das frühere Beispiel:

```
2 / 3 * 6 ergibt 4
2 / (3 * 6) ergibt den Wert 0.111111111
```

Dies ergibt sich, weil wir die Rechenreihenfolge durch die Klammern geändert haben. Der Computer berechnet zuerst den Inhalt der Klammer. Betrachten wir zunächst sehr simple Aufgaben. Wenn Markus 5 Äpfel hat und Rolf 3, wie können sie gerecht teilen? Testen Sie folgende Lösungsvorschläge:

```
    5 + 3 / 2
und
    (5 + 3) / 2
```

Die richtige Lösung ist natürlich die zweite, da ja zunächst die Äpfel zusammengezählt und erst dann zwischen beiden geteilt werden müssen.

Markus hat DM 2,60 und möchte dieses Geld mit Rolf teilen. Er schuldet Ute 30 Pfennig und muß sie zuerst auszahlen. Die Berechnung dieses Vorgangs wird wie folgt aussehen:

```
(2.60 - 0.30) / 2 = ?
```

Markus' Vater möchte ihm die sechsfache Summe dessen geben, was er besitzt. Die Lösung könnte wie folgt aussehen:

```
6 * ((2.60 - 0.30) / 2) = ?
```

Letztlich bietet ihm seine Mutter an, diesen Betrag zu quadrieren:

```
(6 * ((2.60 - 0.30) / 2))^2 = ?
```

Klammern, die so eingesetzt werden, nennt man geschachtelte Klammern. Wenn Klammern auf diese Weise eingesetzt werden, arbeitet der Computer von der innersten zur äußersten Klammer, und dann von links nach rechts unter Beibehaltung der festgelegten Hierarchie.

Lösen Sie die Aufgaben der Übung 4 mit Hilfe der Klammerrechnung:

****

ÜBUNG 4 KLAMMERRECHNUNG

Lösen Sie folgende Aufgaben mit PRINT:

1. Beim Pferderennen startet Michael mit DM 10. Er setzt DM 2.00 aufs erste Rennen und verdoppelt damit seinen Einsatz. Im zweiten Rennen verliert er DM 4.00 und setzt anschließend sein ganzes Geld auf zwei Pferde, von denen das erste den Einsatz verdreifacht und das zweite dieses nochmals verdoppelt. Wieviel verbleibt ihm am Ende des Tages?

```
((((10-2)+2*2)-4)*3)*2 = ?
```

2. Ein Bauer hat zwei identische kreisrunde Felder (Radius 200 m) und zwei gleich große quadratische Felder (Seitenlänge 75 m). Er kauft anschließend einen zweiten Hof mit der gleichen Fläche. Wie groß ist die Gesamtfläche seines Besitzes? Die Kreisfläche wird mit der Formel PI*R^2 errechnet, wobei PI schon als Variable bzw. als Funktion im MTX definiert ist.

```
(((PI * 200^2) *2) + ((75^2) *2)) * 2 = ?
```

****

Die Wurzelberechnung bereitet dem Computer eine weitere Schwierigkeit. Eine Möglichkeit, das Problem zu lösen, bietet die altbekannte mathematische Weisheit, daß Wurzeln nichts anderes sind als Potenzen mit einem Bruch als Exponenten. So ist die Wurzel aus 4 nichts anderes als 4 ^ (1/2). Beachten Sie bitte die Klammer, die sicherstellt, daß die Berechnung in der richtigen Reihenfolge geschieht.

Somit ist die 3. Wurzel von 16 auch in der Schreibweise 16^ (1/3) korrekt.

Möglicherweise denken Sie sich nun, daß dies ja alles recht umständlich ist, und in der Tat, es gibt noch andere, elegantere Wege. Davon später mehr.

****

ÜBUNG 5

Berechnen Sie mit PRINT folgende Wurzeln:

2. Wurzel aus 9	2. Wurzel aus 25	2. Wurzel aus 81
2. " " 144	3. " " 27	3. " " 216
4. " "50625	9. " " 512	

****

KAPITEL 4

STRINGs

Lernziel: Mit Hilfe des Befehls PRINT sollen Sie lernen, Texte in der einfachen Form eines STRINGs auf dem Bildschirm darzustellen.

Der PRINT-Befehl wird nicht zur Darstellung von Zahlen gebraucht, sondern für jegliche Abbildung von Daten auf dem Bildschirm. Text beliebiger Art besteht aus Buchstaben, Zahlen, aber auch Leerzeichen, die alle erst dann ihren Sinn ergeben, wenn die vorbestimmte Reihenfolge eingehalten wird. Wie Sie dem letzten Kapitel entnommen haben, berechnet der Computer vorgegebenes Zahlenmaterial in einer für ihn günstigen Reihenfolge. Würde diese Arbeitsweise auf Texte übertragen, käme als Ergebnis ein sinnloses Wirrwarr heraus. Text wird daher in einer solchen Weise in den Computer eingegeben, daß dieser nicht die Reihenfolge der Daten ändern kann.

Es gibt verschiedene Wege, dies zu bewirken, aber in diesem Kapitel wollen wir uns nur auf den Weg der STRING-Verarbeitung konzentrieren. STRINGs werden aus Buchstaben, Zahlen und Leerzeichen gebildet und in Anführungszeichen eingegliedert.

STRINGs werden genau so vom Computer eingespeichert wie sie eingegeben wurden. Setzen Sie den PRINT-Befehl ein, um folgendes auf den Bildschirm zu bringen:

```
PRINT "JOHN LENNON"
```

Jetzt können Sie auch Ihren Namen anzeigen.

STRINGs haben keine bestimmte Länge und können alle Zeichen des BASIC-Zeichenvorrats mit Ausnahme des Anführungszeichens enthalten. - So arbeitet z.B. auch

```
PRINT "ABCDEFGHIJHKLMNOPQRSTUVWXYZ1234567890!#$%&'()_=-"
```

****

ÜBUNG 6 EINFACHE STRINGS

Schreiben Sie folgendes auf den Bildschirm:

Ihren Namen
Ihre Anschrift
Ihr Geburtsdatum
Ihren Beruf

****

Genau so, wie es Operationen gibt, mit denen Zahlen verknüpft werden können, ist es auch möglich, STRINGs zu manipulieren. Die einfachste Operation besteht darin, STRINGs zu verknüpfen, um einen längeren STRING zu erhalten. Dies geschieht wie bei der Addition mit dem Zeichen +, darf aber nicht mit der Addition verwechselt werden.
Versuchen Sie z.B.

```
PRINT "FERN" + "SEHEN"
```

Daß STRINGs nicht miteinander dividiert, multipliziert usw. werden können, ist sicherlich verständlich. Wie wir später sehen werden, gibt es jedoch noch andere Wege, verschiedene Verknüpfungen herzustellen.

KAPITEL 5

DER DRUCKER

Lernziel: Sie lernen, Daten über einen Drucker auszugeben.

BASIC bietet eine Reihe von Möglichkeiten, einen Drucker zu steuern. Den mit Abstand besten Weg bietet der Befehl LPRINT. Dieser Befehl funktioniert in etwa wie PRINT, nur wird die Information nicht auf dem Bildschirm dargestellt, sondern auf dem Drucker ausgedruckt. Der Begriff LPRINT stammt übrigens aus der Groß-EDV, wo Drucker meist als Zeilendrucker zur Verfügung stehen (Line Printer).

Eingabe: LPRINT "MICK JAGGER"

Sofern der Drucker korrekt angeschlossen wurde, erfolgt nun der Ausdruck von: MICK JAGGER.

Sollten Sie etwas falsch gemacht haben oder den Drucker nicht eingeschaltet haben, macht das nichts, da der MTX wartet, bis Sie bereit sind. Mit Hilfe der BRK-Taste können Sie jederzeit den Druckvorgang abbrechen und die 'Ready'-Meldung erhalten.

Wie bei PRINT kann auch das Ergebnis einer Berechnung mit LPRINT dokumentiert werden.

```
LPRINT "Die Lösung von 2+2 lautet?"
LPRINT 2+2
```

ergibt den Ausdruck von

```
Die Lösung von 2+2 lautet?
4
```

Weitere Angaben zu LPRINT finden Sie im Nachschlageteil dieses Handbuchs unter den Stichworten LPRINT und PRINT.

Um ein eingetipptes Programm auszudrucken, müssen Sie den Befehl LLIST eingeben. Damit wird das z.Zt. im Arbeitsspeicher vorhandene Programm ausgedruckt.

Es ist auch möglich, nur einen Teil des Programmlistings auszudrucken. Dies geschieht in der gleichen Weise wie mit LIST:

```
LLIST 100,200
```

druckt alle Programmzeilen zwischen 100 und 200.

KAPITEL 6

DATENSPEICHERUNG: VARIABLEN

Lernziel: Sie sollen den Begriff 'Variable' verstehen und die Handhabung der Variablen mit dem Befehl LET beherrschen lernen.

Im 2. Kapitel wurde der PRINT-Befehl benutzt, um einfache Berechnungen durchzuführen.

Wenn Sie sich nachfolgende Befehlssätze ansehen, werden Sie feststellen, daß es auch Alternativen zur Darstellung von Operationen gibt.

```
LET A = 6
LET B = 2
LET C = A + B
PRINT C
```

A, B und C werden Variable genannt. Wenn eine Variable ins Programm eingesetzt wird, reserviert der Computer automatisch Speicherplatz für die von dieser Variablen besetzten Werte. Der reservierte Platz wird unter dem Variablennamen aufgerufen. Bei den obigen Ausdrücken werden die Werte 6 und 2 unter den Namen A und B abgespeichert. Die resultierende Summe ist nun im Speicherplatz C zu finden. Mittels PRINT wird der Inhalt des Speicherplatzes C dann ausgedruckt.

****

ÜBUNG 7 LET-ANWEISUNGEN

Zeigen Sie den Wert der jeweiligen LET-Anweisung mit PRINT an:

1)
```
LET X=6*4^2
```

2)
```
LET X=(4-2)^4*(6-3)^2
```

3)
```
LET A=6*8
LET B=16/2
LET C=B*6
LET X=A/C
```

4)
```
LET A=64^0.5
LET B=(A-4)^2
LET C=B-A
LET X=(A*C)/B
```

5)
```
LET X=4
LET Y=X^2
LET X=(Y-X)/6
```

****

Wie beim PRINT-Befehl kann auch der LET-Befehl sowohl auf Ziffern als auch auf Buchstaben bezogen werden. Wird ein Text einem bestimmten Namen zugeordnet, folgt dem Namen unmittelbar ein $-Zeichen, damit der Computer weiß, daß ein STRING folgt.

```
LET A$ = "ANNE"
```

ANNE wird in diesem Fall als STRING behandelt und muß somit in Anführungszeichen gesetzt werden. Wenn Sie dies eingeben und dann die Anweisung PRINT A$, wird der Ausdruck ANNE auf dem Bildschirm erscheinen.

```
Eingabe: LET B$ = " "
         LET C$ = "SCHNEIDER"
         PRINT A$ + B$ + C$
```

Erinnern Sie sich daran, daß dies keine Addition bedeutet. Das '+' Zeichen weist den Computer bloß an, die jeweiligen STRINGs zu verbinden. B$ setzt in diesem Fall lediglich ein Leerzeichen zwischen die Namen ANNE und SCHNEIDER. Verwenden Sie den LET-Befehl in Übung 8, um Ihre persönlichen Daten in STRING-Namen unterzubringen, um sie dann mit PRINT-Befehlen auf dem Bildschirm anzuzeigen.

****

ÜBUNG 8

Verwenden Sie folgende Strings, um die nachfolgenden Daten abzuspeichern: N$, A$, G$, B$.

Ihren Namen
Ihre Anschrift
Ihr Geburtsdatum
Ihren Beruf

****

Diese Methode der Datenspeicherung ermöglicht das Speichern von 26 verschiedene Daten, da es 26 Buchstaben im lateinischen Alphabet gibt: A$, B$, ... Z$. Dies reicht für die Mehrzahl aller Fälle. Wenn Sie den gleichen Buchstaben (z.B. A) einem STRING und einer Zahl zuordnen, erhält der STRING A$ eine gänzlich andere Speicheradresse zu der der Zahl A! Man kann die Anzahl der Speicheradressen dadurch erweitern, daß mehr als eine Zahl oder ein Buchstabe verwendet wird. Bei den folgenden Beispielen beziehen sich alle Variablenbezeichnungen auf verschiedene Speicheradressen.

z.B.: A$, AA$, A1$, A, AA, A1, NAME$, ALTER usw.

Es ist oft ratsam, den Speicher mit allen Variablen zu löschen, um Neuberechnungen vorzunehmen.

Vielleicht möchten auch Sie jetzt die Variablen der Übung 8 löschen. Dies geschieht mit dem Befehl CLEAR.

Alle Variablen werden ebenfalls in dem Augenblick gelöscht, wenn ein Programm mit RUN gestartet wird.

****

ÜBUNG 9 EINSTELLEN DER UHR

Der MTX-Computer hat eine eingebaute Uhr, die auf Normalzeit gestellt werden kann. Sie kann auch als Stoppuhr genutzt werden, indem man sie auf Null setzt. Um die Uhr z.B. auf 12.30 Uhr zu stellen, ist folgender Befehl notwendig:

```
CLOCK "123000"
```

Damit sind die Stunden auf 12, die Minuten auf 30 und die Sekunden auf 00 gesetzt.
Der STRING besteht aus sechs Stellen, von denen die ersten zwei die Stunden, die nächsten die Minuten und die verbleibenden zwei die Sekunden darstellen.

Um während des Arbeitens die Zeit anzuzeigen, reicht die Eingabe:

```
PRINT TIME$
```

Damit wird die Uhrzeit angezeigt.

Versuchen Sie folgendes Uhrenprogramm. Beachten Sie bitte die sechsstellige Zeitangabe.

```
10 INPUT "Wie spät ist es?"; A$
20 CLOCK A$
30 CLS
40 CSR 0,0
50 PRINT TIME$
60 GOTO 40
```

****

KAPITEL 7

PROGRAMMERSTELLUNG

Lernziel: Sie sollen die Methodik des Programmentwurfs mittels Flußdiagramm und Numerierung kennenlernen, sowie AUTO und REM einsetzen können.

Sie sind nun soweit, Programme in MTX-Basic zu entwickeln. Wie Sie bereits beim Übertragen des Programms in Kapitel 1 bemerkt haben werden, sind strenge Regeln zu beachten, wenn das Programm auch tatsächlich laufen soll.

Ferner wird Ihnen nicht entgangen sein, daß Programme komplex ausfallen können, weshalb es unbedingt notwendig wird, vorher genau zu planen, was geschehen soll, und sich auch genau an diesen Plan zu halten. Ordnung wird hier zur sichtlichen Notwendigkeit.

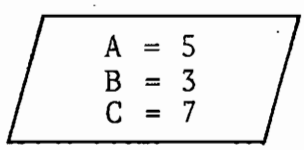
In diesem Kapitel werden wir zeigen, wie dies auch erzielt werden kann, indem Flußdiagramme und REM-Anweisungen benutzt werden.

FLUSSDIAGRAMME

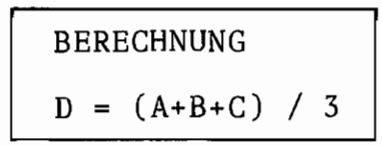
Ein Flußdiagramm ist nichts anderes als die minutiöse Beschreibung dessen, wie ein bestimmtes Problem gelöst werden kann. Betrachten wir z.B. ein Problem, bei dem Markus 5, Rolf 3 und Ute 7 Äpfel haben. Wie müßte ein Programmentwurf aussehen, der diese Äpfel gleichmäßig unter allen aufteilt? Die erforderliche Information wird in eine Tabelle eingetragen, um die Variablen zu bestimmen.

	VARIABLE	
A	MARKUS' ÄPFEL	(3)
B	ROLFS ÄPFEL	(5)
C	UTES ÄPFEL	(7)
D	DIE ANZAHL DER VERTEILTEN ÄPFEL	

Der erste Schritt wäre demnach, dem Computer mitzuteilen, wie viele Äpfel jede Person hat. Beim Erstellen eines Flußdiagramms wird die Ein- oder Ausgabe von Information in ein Parallelogramm eingetragen. Daher sähe unsere erste Arbeit so aus:

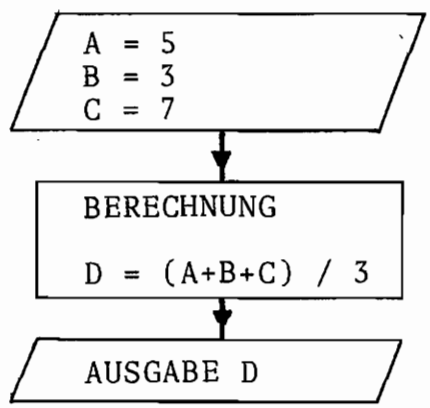


Der zweite Schritt beinhaltet die eigentliche Rechenarbeit. Wir wollen die Gesamtzahl der Äpfel feststellen und das Ergebnis durch die Anzahl der Empfänger teilen (wodurch jeder die gleiche Menge erhielte). Rechenschritte dieser Art werden in rechteckige Kästen eingetragen:



Da wir bereits wissen, wie der Computer Berechnungen durchführt, setzen wir die Summanden natürlich in Klammern, damit die resultierende Summe durch 3 geteilt wird.

Den Computer muß man als letztes noch anweisen, was mit dem Ergebnis zu geschehen hat. Da die Ergebnisanzeige eine Informationsausgabe ist, wird dieser Vorgang wieder durch ein Parallelogramm gekennzeichnet. Das vollständige Programm sieht dann wie folgt aus:



****

ÜBUNG 10 FLUSSDIAGRAMME

1. Vervollständigen Sie das Flußdiagramm. Das Programm wandelt Testergebnisse zu Prozentangaben um. Der Test besteht aus 25 Fragen, und die jeweils richtigen Antworten der Kandidaten lauten:

a) 24 b) 19 c) 20 d) 15 e) 7 f) 12

```
X = Ergebnis

BERECHNUNG

D = X * 100 / 25

AUSGABE P
```

Entwerfen Sie Flußdiagramme zur Lösung folgender Probleme:

2. Benzin kostet 1.25 pro Liter. Wieviel müßten Sie zahlen beim Kauf von:

a) 5 l b) 7.25 l c) 36.27 l d) 55 l.

3. Eine Gallone (engl. Hohlmaß) kostet 4.54mal soviel wie ein Liter. Bei einem Preis von 7.25/Gallone würden kosten:

a) 24 l b) 36 l c) 42 l.

****

PROGRAMMERSTELLUNG

Ein BASIC-Programm besteht aus einer Anzahl von Anweisungen in einer Sprache, die der Computer versteht (MTX-BASIC). Jede Anweisung wird im allgemeinen in einer separaten Programmzeile untergebracht und muß der vorgegebenen Sprache und ihren Regeln exakt entsprechen. Die Anweisungen werden dann in der Reihenfolge durchgeführt, die der jeweiligen Zeilennummer entspricht. Zeilennummern können von 1 bis 65536 reichen. Jede BASIC-Zeile muß mit einer ganzzahligen Nummer aus diesem Bereich beginnen. Man kann Abstände in der Numerierung lassen, um später Einfügungen vornehmen zu können. Es ist üblich, in 10er-Schritten zu arbeiten, so daß unser "Äpfelprogramm" so aussieht:

```
10 REM ÄPFELPROGRAMM
20 LET A = 5
30 LET B = 3
40 LET D = (A + B + C) / 3
50 PRINT D
35 LET C = 7
```

Die Zeile 35 wurde absichtlich ausgelassen, um zu demonstrieren, daß sie auch nachträglich eingefügt werden kann. Dies stört nicht die logische Abarbeitung des Programms, da die Programmzeilen automatisch in aufsteigender Rangfolge sortiert und abgearbeitet werden. Mit dem Befehl LIST wird dies auch optisch auf dem Bildschirm sichtbar.

Die REM-Anweisung in Zeile 10 enthält lediglich die Programmbenennung. REM ist die Abkürzung des englischen REMARK oder REMIND, was BEMERKUNG oder MERKEN bedeutet. Angaben hinter einer REM-Anweisung werden vom Computer nicht beachtet und sind nur dazu da, dem Programmierer die Möglichkeit zu geben, ERLÄUTERUNGEN oder HINWEISE zur späteren Orientierung zu geben. Dies ist bei kleinen Programmen vielleicht nicht so wichtig, macht sich aber später bei Programmen mit 500 oder 1000 Zeilen positiv bemerkbar.
So könnte obiges Programm für jeden Schritt im Flußdiagramm seine entsprechende REM-Anmerkung aufweisen:

```
10 REM ÄPFELPROGRAMM
20 REM ZUORDNUNG DER MENGE DER ÜPFEL
30 LET A = 5
40 LET B = 3
50 LET C = 7
60 REM BERECHNUNG DER TEILUNGSMENGE
70 LET D = (A + B + C) / 3
80 REM AUSDRUCK DER TEILUNGSMENGE
90 PRINT D
```

Versuchen Sie nun, Programme für die Flußdiagramme aus der Übung 10 zu schreiben. Verwenden Sie dabei REM-Anmerkungen und alle BASIC-Befehle, die bis jetzt erarbeitet wurden. Vergessen Sie dabei bitte nicht, daß ein fertiggstelltes Programm erst mit dem Befehl RUN tatsächlich abläuft. Es kann nur jeweils ein Programm zur gleichen Zeit ablaufen, so daß der NEW-Befehl vor der Eingabe eines neuen Programms unbedingt erforderlich ist. Experimentieren Sie mit CLEAR, LOAD, LIST und VERIFY!

Sie werden feststellen, daß zunächst etliche Fehler gemacht werden. Falls die Lage völlig hoffnungslos erscheint, hilft immer die RESET-Taste.

```
10 REM KEINE PANIK!
```

Sobald Sie mit BASIC etwas vertrauter sind, werden Sie den AUTO-Befehl als hilfreich emfinden. Dieser Befehl erzeugt nach jeder Zeileneingabe mit RET eine neue Zeilennummer. - Beispiel:

Eingabe: AUTO 100,25

Damit beginnt die Zeilennumerierung bei 100 und wird in 25er Schritten fortgeführt. Nach der Eingabe der letzten Zeilen eines Programms, oder wenn Sie eine Zeile einfügen möchten, muß die CLS-Taste gedrückt werden, gefolgt von RET. Damit ist der AUTO-Befehl aufgehoben.

Mit der CLS-Taste wird die gerade in Arbeit befindliche Zeile gelöscht - unabhängig davon, ob AUTO wirksam ist oder nicht.

KAPITEL 8

DATA

Lernziel: Sie sollen Programme entwickeln können, die größere Datenmengen mit Hilfe der Befehle READ und DATA verarbeiten.

Bei den bisherigen Befehlen haben wir Daten an eine bestimmte Stelle gesetzt und diese dann mittels der nachfolgenden Operationen verarbeitet. Sollten wir eine ähnliche Berechnung durchführen wollen, nur mit anderen Daten, müßten diese mit einem nicht unerheblichen Programmieraufwand ausgetauscht werden. Im "Äpfelprogramm" wäre es z.B. erforderlich, alle LET-Befehle zu ändern, wenn Markus 4, Rolf 5 und Ute 6 Äpfel besitzen sollten.

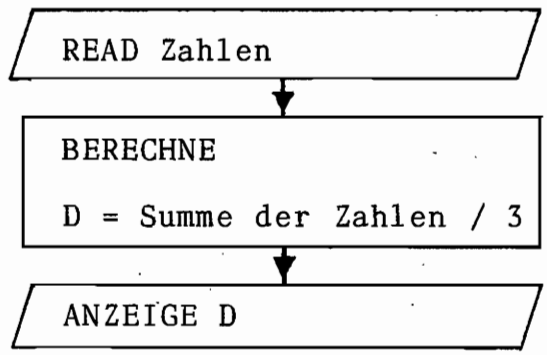
Eine Möglichkeit, dieses Problem zu umgehen, bietet der DATA-Befehl. So würde für das gesamte Programm eine Zeile am Schluß z.B. so lauten:

```
60 DATA 4,5,6
```

Beachten Sie bitte, daß jede Zahl durch ein Komma getrennt ist, daß dieses aber nicht hinter DATA oder am Schluß gesetzt wird.

Der Befehl, der diese Daten in den Arbeitsspeicher liest, heißt READ. Betrachten Sie nachfolgendes Flußdiagramm und das Programm mit der neuen Teilungslösung:

VARIABLEN	
A	MARKUS' ÄPFEL (4)
B	ROLFS ÄPFEL (5)
C	UTES ÄPFEL (6)
D	DIE ANZAHL DER GETEILTEN ÄPFEL




```
10 REM NEUE VERTEILUNG
20 READ A,B,C
30 LET D = (A + B + C) / 3
40 PRINT D
50 STOP
60 DATA 4,5,6
```

Im Listing finden Sie einen neuen Befehl, STOP. Die DATA-Anweisung in Zeile 60 ist kein Befehl, sondern hält lediglich Daten für den READ-Befehl in Zeile 20 bereit. STOP in Zeile 50 ist die Anweisung an den Computer, alles nachfolgende nicht zu verarbeiten. Das so geschriebene Programm kann nun für eine beliebige Teilungsaufgabe mit dem Quotienten 3 eingesetzt werden, solange die Zeile 60 (mit DATA) entsprechend angepaßt wird. Versuchen Sie folgende DATA-Variable für das obige Programm, ehe Sie sich an die Übung 11 heranmachen.

	VARIABLE
A	MARKUS' ÄPFEL (8) (12) (4)
B	ROLFS ÄPFEL (8) (10) (11)
C	UTES ÄPFEL (8) (11) (3)
D	DIE ANZAHL DER GEMEINSAM GETEILTEN ÄPFEL

Schreiben Sie das Programm neu, nun aber für fünf Personen, und setzen Sie verschiedene Zahlen in die DATA-Zeile.

Obwohl DATA-Anweisungen an beliebige Stellen eines Programms gesetzt werden können, ist es anzuraten, sie wegen der einfacheren Ergänzbarkeit ans Programmende zu setzen.

****

ÜBUNG 11 TESTERGEBNISSE

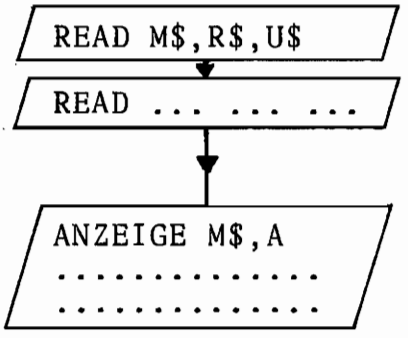
In einem Englischtest haben Markus, Rolf und Ute folgende Ergebnisse erzielt:

ENGLISCH TEST
MARKUS 64
ROLF 68
UTE 45

Vervollständigen Sie das Programm unter Zuhilfenahme des Flußdiagramms und der READ-, PRINT- und DATA-Anweisung, um eine Ergebnistabelle auszudrucken.

VARIABLENTABELLE	
MARKUS = M$	A = 64
ROLF = R$	B = 68
UTE = U$	C = 45

FLUSSDIAGRAMM



PROGRAMM

```
100 REM TESTERGEBNISSE
110 READ M$,R$,U$
120 READ A,B,C
130 PRINT M$,A
140 PRINT R$,B
150 PRINT U$,C
160 STOP
170 DATA MARKUS,ROLF,UTE
180 DATA 64,68,45
```

Erweitern Sie das Programm um die Schüler Christopher (72), Monika (48), Eva (56).

****

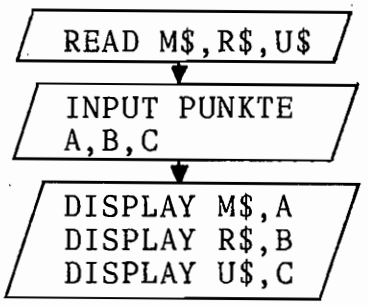
KAPITEL 9

DATA - EINGABE

Lernziel: Sie sollen Programme entwerfen können, die dem Anwender die DATA-Eingabe während des Programmablaufs gestattet.

Die DATA-Anweisung wird verwandt, um DATA in einem Programm zu speichern, ehe dies mit RUN abläuft. Es ist jedoch möglich, noch während des Programmablaufs Daten einzugeben. Diese Eingabe erfolgt mit dem Befehl INPUT (Weitere Hinweise zum INPUT-Befehl finden Sie im Anhang).

FLUSSDIAGRAMM



PROGRAMM

```
100 REM TESTERGEBNISSE 2
110 READ M$,R$,U$
120 INPUT "A= ? ";A
130 INPUT "B= ? ";B
140 INPUT "C= ? ";C
150 PRINT M$,A
160 PRINT R$,B
170 PRINT U$,C
180 STOP
190 DATA MARKUS,ROLF,UTE
```

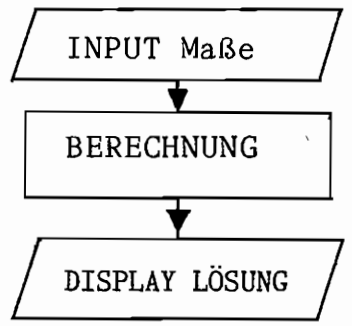
****

ÜBUNG 12 INPUT

Schreiben Sie das "Viereckspogramm" um für:

a) die Fläche eines Dreiecks
b) die Fläche eines Kreises
c) den Umfang eines Kreises

FLUSSDIAGRAMM



PROGRAMM

```
100 REM Fläche eines Vierecks
110 INPUT "Länge= ";L
120 INPUT "Breite= ";B
130 LET F = L * B
140 PRINT "Fläche= ";F
```

```
100 REM Fläche eines Dreiecks
110 INPUT "Hypotenuse ";HY
120 INPUT "Höhe ";H
130 LET F = H * HY / 2
140 PRINT "Fläche= ";F
```

```
100 REM Fläche eines Kreises
110 INPUT "Radius= ";R
120 LET F = R^2 * PI
130 PRINT "Fläche= ";F
100 REM Umfang eines Kreises
110 INPUT "Durchmesser ";D
120 LET U = D * PI
130 PRINT "Umfang ";U
```

****

KAPITEL 10

VERZWEIGTE PROGRAMME (ENTSCHEIDUNGEN UND BEDINGTE BEFEHLE)

Lernziel: Sie sollen ein Programm entwickeln können, mit dem der Computer Entscheidungen treffen kann.

Die bislang erprobten Programme beinhalteten eindeutige Befehle und Programmabläufe. Ein Ergebnis wurde erzielt, indem eine Variable (z.B. A=5) mit einer anderen verknüpft und anschließend angezeigt wurde (LET A+B=C, PRINT C).

In diesem Kapitel werden wir untersuchen, wie Entscheidungen in einem Programm getroffen werden können. Um eine Entscheidung treffen zu können, muß der Computer eine Aussage als wahr oder falsch einstufen, wobei verschiedene Lösungswege eingeschlagen werden können, die von zuvor fälligen LOGISCHEN Entscheidungen abhängen. BEDINGTE Befehle wie IF und THEN finden hier ihre Verwendung: IF (übersetzt: WENN) eine Lösung richtig ist, folgt eine Anweisung THEN (DANN), die angibt, was der Computer zu tun hat. Falls die gewünschte Lösung nicht erreicht wurde, kann mit einer dritten Anweisung ELSE (SONST) ein alternativer Befehl folgen. Überlegen Sie sich mal folgendes sprachliche Beispiel:

```
WENN (IF) die Milch frisch ist,
DANN (THEN) werde ich Kaffee mit Milch trinken,
SONST (ELSE) werde ich ihn schwarz trinken.
```

Die Verwendung von IF, THEN und ELSE in MTX-BASIC kann genauso betrachtet werden. Die Entscheidung hängt ab vom Gehalt der IF-Aussage.

```
10 INPUT "Ihr Alter? ";A
20 IF A=0 THEN GOTO 10 ELSE GOTO 30
30 PRINT "Ihr Alter ist ";A
```

Das Verhältnis zwischen A und 0 ist ein Verhältnis zwischen zwei Werten. Das obige Beispiel ist eine Operation, um das Verhältnis zweier Werte zu untersuchen und wird in der Fachsprache als BOOLEsche Operation bezeichnet. Es gibt weitere BOOLEsche Operationen, die unten aufgeführt sind:

OPERATION	RELATION	AUSDRUCK
=	ist gleich zu	A = B
<>	ist ungleich zu	A <> B
<	ist kleiner als	A < B
>	ist größer als	A > B
<=	ist kleiner oder gleich	A <= B
>=	ist größer oder gleich	A >= B

Wenn ein Ausdruck sowohl arithmetische als auch BOOLEsche Operationen enthält, werden zunächst die arithmetischen Aufgaben ausgeführt.

- Beispiel: Wenn eine deutsche Schule acht Schulen in England bitten würde, die Temperatur um 14.00 Uhr am Tage X mitzuteilen, wäre es gut möglich, daß, wegen der augenblicklichen Umstellung auf das metrische System, Angaben in Fahrenheit und Celsius einträfen. Für die Auswertung der Zahlen müßte also zunächst eine einheitliche Skala zugrunde liegen. Das folgende Programm ist ein Versuch, Fahrenheit in Celsius umzurechnen.

Celsius	Fahrenheit
Manchester 10	Sheffield 56
Liverpool 11	Oxford 62
Brighton 13	Glasgow 52
Cardiff 10	Newcastle 54

Sehen Sie sich bitte nachfolgendes Programm an. Zeile 110 speichert einen Temperaturwert in der Variablen T, und Zeile 120 speichert F oder C in B$, um den Wert als Fahrenheit oder Celsius zu kennzeichnen. In Zeile 130 muß der Computer durch die Abfrage von B$ entscheiden, ob der weitere Programmablauf mit F oder C zu erfolgen hat. Sollte die Variable B$ F angeben, muß die Verzweigung mit der Umrechnung in Celsius gewählt werden, so daß hier die Anweisung THEN GOTO Anwendung findet, um die Operation der Zeile 135 auszuführen. Wenn aber B$ C enthält, weist ELSE GOTO den Computer an, die Zeile 135 auszulassen und das Programm mit der PRINT-Anweisung in Zeile 140 fortzufahren.

INPUT TEMPERATUR

```
100 REM Temperaturumrechnung
110 INPUT "Temperatur= ";T
120 INPUT "C oder F ";B$
130 IF B$="F" THEN GOTO 135 ELSE GOTO 140
135 LET T = (T-32) * 5/9
140 PRINT "Temperatur in Celsius: ";T
```

ZÄHLER

Es gibt eine Reihe von Möglichkeiten, dieses Programm "anwenderfreundlicher" zu machen. Z.B. könnten wir einen einfachen Zähler hinzufügen, so daß der Computer weiß, wie viele Temperaturwerte berechnet werden sollen. Zähler (engl.: counter) werden dann eingesetzt, wenn eine Berechnung mit vielen Werten durchgeführt wird. Der erste Schritt beim Einsatz eines Zählers besteht darin, daß dieser initialisiert wird (INITIALISE). Dabei bestimmen Sie die Zahl, ab der gezählt werden soll (in der Regel 0, aber im Prinzip beliebig). Bei der eigentlichen Berechnung wird dann immer eine 1 hinzuaddiert, bis die gewünschte Anzahl von Berechnungen erreicht worden ist.

Am Beispiel einer Temperaturumrechnung könnte das Programm dann wie folgt umgeschrieben werden:

```
10 INPUT "Anzahl von Temperaturwerten ";N
20 LET C = 0
30 INPUT "Eingabe Temperatur ";T
40 INPUT "Fahrenheit (J/N)? ";B$
50 IF B$="N" THEN GOTO 70
60 LET T = (T-32) * 5/9
70 PRINT T
80 LET C = C + 1
90 IF C=N THEN GOTO 100 ELSE GOTO 30
100 PRINT "ENDE"
```

Eine andere Möglichkeit, das Programm effizienter zu gestalten, wäre, die Eingaberoutine so zu schreiben, daß alle Eingaben direkt erfolgen könnten. Da wir ja eine Temperaturübersicht des ganzen Landes wünschen, erfordert dies eine zusammenhängende Eingabe (INPUT) und eine übersichtliche Ausgabe (in Tabellenform).

Dies kann u.a. mit den READ- und DATA-Anweisungen realisiert werden.

****

ÜBUNG 13

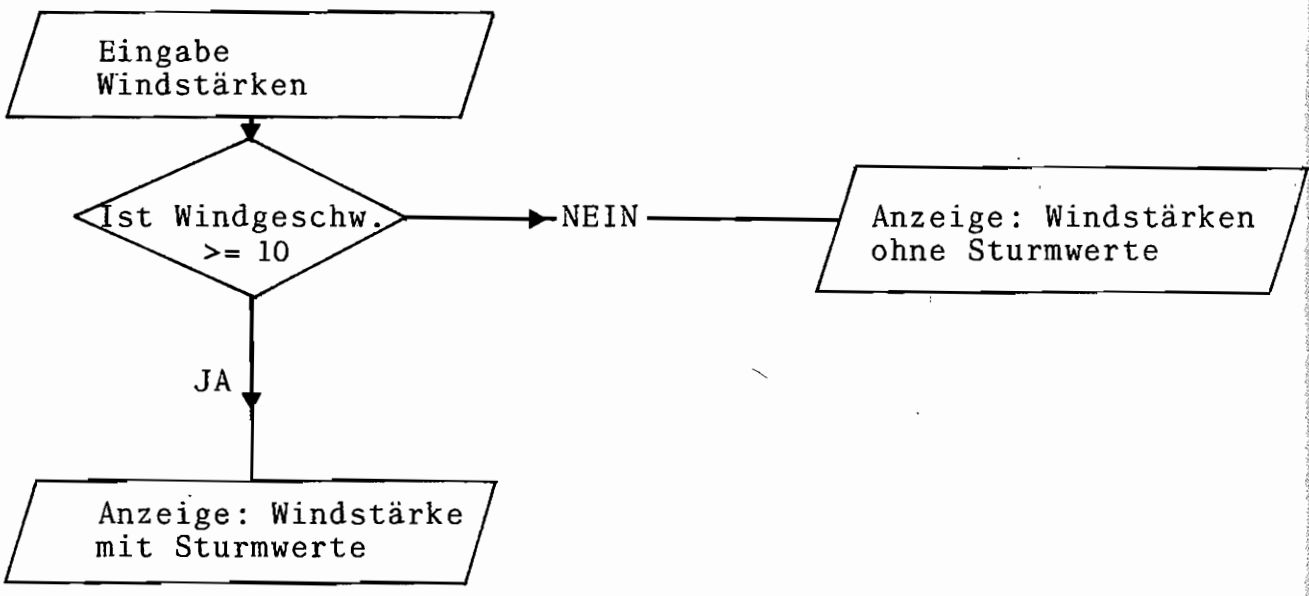
1. Ändern Sie das Temperaturumrechnungsprogramm mit Hilfe der READ- und DATA-Anweisungen, so daß eine geschlossene Tabelle resultiert.

2. Entwerfen Sie ein Programm, das Windstärken auswertet und getrennte Tabellen für normale und stürmische Windverhältnisse anzeigt.

Variablentabelle

Windgeschwindigkeit	
Windstärke 1	4 (A)
Windstärke 2	12 (B)
Windstärke 3	8 (C)
Windstärke 4	11 (D)
Windstärke 5	10 (E)
Windstärke 6	9 (F)

Ein Sturm ist dann angezeigt, wenn die Windstärke gleich oder größer 10 ist.



****

Hinweise: Anhang IF, THEN, ELSE, BOOLEsche Ausdrücke.

KAPITEL 11

UNTERPROGRAMME

Lernziel: Sie sollen den Gebrauch von Unterprogrammen kennenlernen und einfache Anwendungen beherrschen können.

Die GOTO-Anweisung ist dann hilfreich, wenn ein Programm einige wenige Verzweigungen aufweist und die Gesamtübersicht gegeben ist. Werden jedoch viele "Nebenberechnungen" durchgeführt, die sich gleichen, besteht leicht die Gefahr, daß der Programmierer die Übersicht verliert und zum Schluß "Überraschungen" unerwünschter Art resultieren. In solchen Fällen ist der Gebrauch von Unterprogrammen anzuraten.

Sobald der Computer auf eine Anweisung für ein Unterprogramm stößt, wird das Hauptprogramm verlassen, - wobei er sich die letzte abgearbeitete Programmzeile merkt -, um die Anweisungen des Unterprogramms auszuführen. Sobald diese erledigt sind, wird das Hauptprogramm ab der "Ausstiegsstelle" fortgeführt.

Die hierfür verwendeten Befehle heißen GOSUB (GO zur SUBroutine) und RETURN. Ein simples Beispiel sehen Sie weiter unten.

Bei diesem Beispiel ist der Wert für die Masse und das Volumen von einer Anzahl von Flüssigkeiten gegeben, woraus die Dichte zu berechnen und anschließend in Tabellenform auszudrucken ist.

```
100 REM Dichteprogramm
110 PRINT "Masse","Volumen","Dichte"
120 READ M                              READ DATA M
130 IF M=0 THEN STOP                    Ist M=0? Wenn ja, STOP
140 READ V                              Nein; READ DATA V
150 GOSUB 500                           GOSUB zu D=M/V
160 PRINT M,V,D                         ANZEIGE M,V,D
170 GOTO 120
180 DATA 8,4,6,3,12,6,0
500 REM Unterprogramm
510 LET D=M/V
520 RETURN
```

Beachten Sie, daß Zeile 130 dieses Programms dazu dient, dem Computer mitzuteilen, daß beim Erreichen von M = 0 angehalten werden muß. Am Ende der DATA-Zeile (180) finden Sie diesen Wert. Dies ist eine übliche Methode, um ein Programm zu stoppen.

Versuchen Sie weitere Umwandlungsprogramme.

****

ÜBUNG 14

Entwerfen Sie ein Programm mit Unterprogrammen, um folgendes zu lösen:

1. DM-Beträge sollen bei einem Wechselkurs von 3.9:1 in Pfund Sterling getauscht werden.
2. Zentimeter sollen in Zoll umgewandelt werden.
3. Suchen Sie sich eigene praktische Wandlungsaufgaben, die nach dem vorgegebenen Schema gelöst werden können.

Überprüfen Sie Ihre Ergebnisse anhand bekannter Werte.

KAPITEL 12

PROGRAMMSTRUKTURIERUNG

Lernziel: Sie sollen lernen, mit Hilfe von BASIC-Anweisungen Programme mit Programmschleifen zu strukturieren.

Bislang wurden GOTO-Anweisungen benutzt, um wiederkehrende Berechnungen auszuführen. Sehen Sie sich folgende Programme an, die beide eine Liste mit den Zahlen 1 bis 10 ausdrucken:

```
10 LET I=1                      10 FOR I=1 TO 10 STEP 1
20 PRINT I                      20 PRINT I
30 LET I = I + 1                30 NEXT I
40 IF I > 10 THEN STOP          40 STOP
50 GOTO 20
```

Schleifen werden so oft benötigt, daß es einen besonderen Befehl gibt, um sie schneller, flexibler und leichter verständlich zu machen.

Die FOR-Anweisung sagt dem Computer, daß er sich am Anfang einer Schleife befindet. Diese Schleife wird auch als FOR LOOP bezeichnet. Jede FOR LOOP besitzt eine zugeordnete Variable, die man Kontrollvariable nennt. Im obigen Beispiel ist die Kontrollvariable I, aber sie könnte auch eine beliebige andere numerische Variable sein.

```
10 FOR I ...
```

Wenn das Programm die erste FOR LOOP antrifft, erhält die Kontrollvariable einen Wert, in diesem Beispiel 1. Es könnte aber auch eine beliebige Zahl oder ein mathematischer Ausdruck sein.

```
10 FOR I = 1 ...
```

Die FOR-Schleife wird nun ausgeführt, bis eine NEXT-Anweisung mit der gleichen Kontrollvariablen angetroffen wird (Zeile 30).

Das Programm kehrt nun zurück zum Anfang der Schleife und überprüft, ob alle Schritte ausgeführt wurden. Dazu gibt es zwei weitere Werte in der FOR-Anweisung. Der zweite kennzeichnet das Limit (Begrenzung), und der dritte das Increment. Das Increment definiert die Schrittgröße, mit der die Kontrollvariable zunimmt, bis das Limit erreicnt wird. Am Beispiel ist das Limit 10 und das Increment 1.

```
10 FOR I = 1 TO 10 STEP 1
```

Wird das Limit erreicht, springt das Programm zur nächsten Zeile hinter der NEXT-Anweisung. Ist das Limit noch nicht erreicht, wird die Schleife nochmals wiederholt mit einer um den Wert des Increments erhöhten Kontrollvariablen. Nachfolgend ein praktisches Beispiel:

```
100 PRINT "G-Marke","Fahrenheit","Celsius"
110 FOR I=1 TO 8 STEP 1
120 LET F=250+I*25
130 LET C=(F-32)/9*5
140 PRINT I,F,,C
150 NEXT I
160 STOP
```

Dieses Programm wandelt die Temperaturstufe G-Marke eines Gasherds sowohl in Celsius als auch Fahrenheit um und druckt die resultierende Tabelle aus. I repräsentiert die Temperaturskala. Zeile 110 initialisiert die Programmschleife. Der Computer "sieht" also, daß die Stufen 1 bis 8 mittels einer Schleife berechnet werden sollen, und zwar mit einer Schrittweite von 1. Würde der Herd mit der Stufe 0.5 beginnen, könnte man auch die Schrittweite 0.5 einsetzen. Die Zeile müßte dann heißen:

```
110 FOR I = 0.5 TO 8 STEP 0.5
```

Zeile 150 ist das Ende der Schleife. Das Programm geht zurück zur FOR-Anweisung und muß nun die Berechnung für den nächsten Wert von I durchführen. I wird um den Wert von STEP erhöht. Erreicht I den Wert des Limits, ist die FOR LOOP abgearbeitet, und die nächste Zeile des Programms wird ausgeführt. Die Zeilen zwischen FOR und NEXT (120-140) beinhalten die Anweisungen für jeden Schritt innerhalb der Schleife.

****

ÜBUNG 15

Überarbeiten Sie Ihre Programme aus der Übung 14 mit Hilfe von FOR...NEXT und drucken Sie die Werte tabellenmäßig aus.

****

Bei den bisherigen Beispielen war es recht einfach, alle Daten für die FOR...NEXT-Anweisung zur Verfügung zu stellen. Vielleicht möchten Sie aber ein Programm schreiben, wo die Anzahl der Berechnungen ständig variiert. Ein solches Programm könnte z.B. die Ergebnisse der Bundesliga berechnen.

Ein Programm-Beispiel für diese Art von Anwendung:

```
100 REM Durchschnittsberechnungen
105 PRINT "Durchschnitt: ";
110 REM Summe auf Null setzen
120 LET S=0
130 REM Lesen der Gesamtanzahl
140 READ N
150 REM Schleifeninitialisierung
160 FOR I=1 TO N
170 REM Lesen der Werte
180 READ X
190 LET S=S+X
200 NEXT I
210 REM Durchschnittsberechnung
220 LET A=S/N
230 PRINT A
240 DATA 7
250 DATA 125,0,45,67,83,90,68
```

Im vorhergehenden Beispiel wurde die FOR-Anweisung an einem Problem angewandt, wo die Anzahl der auszuführenden Schleifendurchgänge variieren kann. Betrachten Sie nachfolgendes Programm, das mehr als eine FOR-Schleife einsetzt, um automatisch Reifendrücke auf 32 einzustellen (Simuliert durch die PRINT-Anweisung).

```
10 INPUT "Reifendruck: ";P
20 IF P<>32 THEN GOTO 50
30 PRINT "Reifendruck korrekt"
40 GOTO 10
50 IF P>32 THEN GOTO 100
60 FOR I=P TO 32 STEP 1
70 PRINT I
80 NEXT I
90 GOTO 30
100 FOR I=P TO 32 STEP -1
110 PRINT I
120 NEXT I
130 GOTO 30
```

KAPITEL 13

NOCH MEHR VERZWEIGTE PROGRAMME (BEDINGTE SPRÜNGE)

Lernziel: Sie sollen lernen, Programme mit komplexen Entscheidungsstrukturen zu erstellen.

Die vorangegangenen Kapitel haben gezeigt, wie Programme mit Verzweigungen und Schleifen erstellt werden. Diese Methoden sind aber kaum geeignet für die Eingabe von ständig variierenden Daten. Bei einem Billardspiel mit 4 Spielern wird z.B. der neueste Wert eines Spielers zu seinem bisherigen Gesamtwert hinzuaddiert, während die Werte seiner Mitspieler nicht betroffen sind.

Das nachfolgende Programm bietet einen Lösungsansatz:

```
10 REM Eingabeformat: Spielernr.,Punkte
20 LET S1=0:LET S2=0:LET S3=0:LET S4=0
30 INPUT "Spielernr. und Punkte: ";P,S
40 IF S = 5000 THEN GOTO 140
50 ON P GOTO 30,60,80,100,120
60 LET S1 = S1 + S
70 GOTO 30
80 LET S2 = S2 + S
90 GOTO 30
100 LET S3 = S3 + S
110 GOTO 30
120 LET S4 = S4 + S
130 GOTO 30
140 PRINT:PRINT
150 PRINT "SPIELER",,"PUNKTE"
160 PRINT
170 PRINT "   1",,S1
180 PRINT "   2",,S2
190 PRINT "   3",,S3
200 PRINT "   4",,S4
210 PRINT:PRINT
220 PRINT "SPIELENDE"
```

Beachten Sie bitte Zeile 40, wo die IF-Bedingung das Spiel bei 5000 Punkten abbrechen würde, also praktisch nie.

****

ÜBUNG 16 ON ... GOTO ON ... GOSUB

1. Entwickeln Sie ein Spieleprogramm für 4 Teilnehmer, wo jeder Teilnehmer eine Zahl zwischen 1 und 100 raten soll, die von einer fünften Person eingegeben wurde. Die Struktur des obigen Programms bietet dabei einen günstigen Ausgangspunkt.

KAPITEL 14

WEITERES ÜBER VARIABLE

Lernziel: Die Verwaltung und Manipulation von Variablen soll in ihren Grundzügen bekannt werden.

Bislang wurden Daten zur Berechnung von Lösungswerten eingesetzt, die der Computer ausschließlich auf dem Bildschirm ausdruckte. Es gibt aber auch Fälle, wo auch die Grunddaten sichtbar sein sollen. So braucht man schon bei den einfachen Berechnungsarbeiten nicht nur die Ergebnisse, sondern auch die Zwischenergebnisse und die Grunddaten für einen richtigen Überblick.

FELDER und DIM

Sie können sich vielleicht aus Kapitel 2 erinnern, daß Variable Platzhalter darstellen, die Daten enthalten und durch Buchstaben gekennzeichnet sind. Ferner ist es möglich, den Datenumfang durch Indizierung zu erweitern. So sieht zum Beispiel die Erweiterung der Variablen A wie folgt aus: A1, A2, A3,... Setzen wir die Zahl in eine Klammer (A(1), A(2) usw.), weiß der Computer, daß diese Variablen zusammengehören. In solchen Fällen haben wir nur eine Reihe von Zahlen (A), die durch die Indizierung (1,2,....9) genauer bezeichnet werden. Eine solche Zahlenreihe nennt man Feld (engl. ARRAY).

Um die Speicherplatzbelegung eines solchen FELDES zu organisieren, muß dem Computer mitgeteilt werden, wie viele Stellen benötigt werden. Dies geschieht mit der DIMensionierungsanweisung.

```
10 DIM S(40)
```

Dieses Beispiel besagt, daß wir ein Feld mit insgesamt 40 Zellen benötigen. Mit der Angabe (40) wird die Obergrenze für die Anzahl der möglichen Zellen dieses Feldes festgelegt. Praktisch sähe das Feld wie folgt aus:

```
S(1), S(2), S(3), S(4),... S(40).
```

DIM-Anweisungen können auch benutzt werden, um STRINGS zu dimensionieren. Ähnlich wie bei Zahlenfeldern wird für den jeweiligen String Platz reserviert, so daß das Programm nicht möglicherweise hinterher mangels Speicherplatz durcheinandergerät. Die DIM-Anweisung für einen STRING von 14 Zeichen sähe dann wie folgt aus:

```
10 DIM R$(14)
```

Dies würde die Eingabe

```
20 LET R$ = "JOSEF MEIERHOF"
```

aber nicht

```
20 LET R$ = "JOSEPH MEIERHOF"
```

gestatten.

Wird eine DIM-Anweisung nicht vor dem ersten Gebrauch des STRINGs ausgeführt, geht das Programm von einem eindimensionalen STRING aus. D.h., es wird nur Platz für eine einzige Zeichenkette reserviert (siehe auch die Erläuterung im Anhang). Damit ein FELD eine Anzahl von STRINGs enthält, muß eine DIM-Anweisung angeben, wie viele STRINGs möglich sein sollen und wie groß der größte STRING sein darf. Z.B.:

```
10 DIM R$(20,10)
```

schafft Platz für 20 STRINGs mit jeweils 10 Zeilen.

```
10 DIM R$(4,5)
20 DIM S(4)
30 FOR I=1 TO 4 STEP 1
40 READ R$(I),S(I)
50 PRINT R$(I),S(I)
60 NEXT I
70 STOP
80 DATA Brian,47,Joe,62,Ute,26,Chris,54
```

****

ÜBUNG 17 DIM-ANWEISUNGEN

Verändern Sie obiges Programm, um alle Vereine der Bundesliga aufzuführen; zugleich soll die durchschnittliche Torzahl ermittelt werden.

****

Versuchen Sie herauszufinden, wozu folgendes Programm geschrieben wurde:

```
10 LET A$ = "123456789"
20 FOR I = 1 TO 7
30 PRINT A$ (I,3)
40 NEXT I
```

Siehe Anhang zu DIM, STRING-Handhabung.

KAPITEL 15

SORTIEREN

Lernziel: Sie sollen lernen, unter Anwendung verknüpfter Schleifen und Felder Zahlen zu sortieren.

Einer der Vorzüge des Computers liegt in seiner Fähigkeit, große Datenmengen nach einem vorgegebenem Kriterium schnell und sicher zu ordnen. Ob diese Ordnung sich auf alphabetische oder numerische oder gar gemischte Daten bezieht, spielt kaum eine Rolle. Die Anzahl der Möglichkeiten ist so groß, daß hier nur exemplarisch einige wenige Methoden gezeigt werden können. Das Hauptproblem bei allen Sortierverfahren liegt in der großen Zahl der Tests, die durchgeführt werden müssen, d.h. der Computer muß die Grunddaten etliche Male überprüfen, bis eine gewünschte neue Ordnung erreicht ist. Nehmen wir z.B. drei Zahlen mit der Kennzeichnung A, B und C. Der erste Test bestünde darin zu fragen, ob A kleiner ist als B. Sollte dies der Fall sein, wäre eine erste Reihenfolge A, B festgelegt. Wenn dies nicht der Fall wäre, müßte die Reihenfolge geändert werden. Dies geschieht, indem man die Zahl von A gegen die Zahl von B austauscht. Dazu wird eine dritte Speicherstelle T benutzt:

```
10 IF A<B THEN STOP
20 LET T=B
30 LET B=A
40 LET T=B
```

Nachdem nun A und B in eine Reihe gesetzt wurden, muß nun C ebenfalls eingeordnet werden. Hier werden nun 2 Fragen nötig, obwohl wir vielleicht auch mit einer auskommen. Die Fragen lauten:

1. ist C kleiner als B?
2. ist C kleiner als A?

Diese Methode führt zu einer immer größeren Fragekette, je mehr Zellen insgesamt geordnet werden müssen. Überlegen Sie sich, wie viele Fragen erforderlich wären, um 50 Zellen zu ordnen!

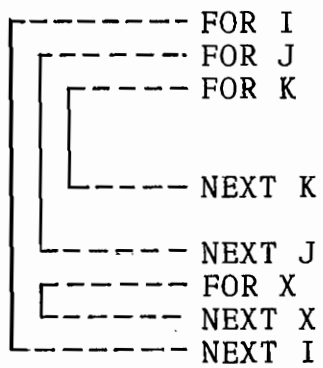
Eine Möglichkeit, hier effizient zu arbeiten, bietet der RIPPLE SORT. Dabei werden alle Zahlen in einer Reihe aufgestellt, und der Computer fragt nur, ob die jeweilige Zahl größer ist als die, die er bereits als größte registriert hat. Falls eine größere Zahl gefunden wird, werden beide Zahlen ausgetauscht. Dies hat zur Folge, daß die größte Zahl ganz zum Ende gelangt, die zweitgrößte zur vorletzten Stelle usw. Dieser Vorgang läuft so lange ab, bis alle Zahlen sortiert sind. Hierzu gibt es das sog. FLAG. Ein FLAG ist nichts weiter als eine einfache Markierung, die Auskunft gibt, ob der Sortiervorgang angehalten werden kann. Beim Zustand 0 wird das Programm gestoppt. Folgendes Programm verdeutlicht die Funktionsweise. Nach jedem Zahlenaustausch wird FLAG in Zeile 120 auf 1 gesetzt. Die Hauptschleife enthält die Zeile 120,

so daß, wenn am Ende eines Programmdurchlaufs noch immer 0 angezeigt wird, alle Zellen geordnet sind und das Programm am Ende ist.

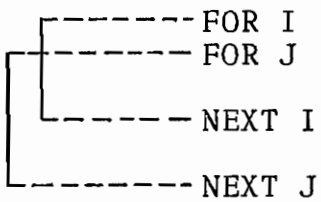
```
10 DIM A(20),B(20)
20 PRINT:PRINT
30 PRINT "ZAHLEN","SORTIERTE LISTE"
40 READ N
50 FOR I=1 TO N STEP 1
60 READ A(I)
70 LET B(I)=A(I)
80 NEXT I
90 LET F=0
100 FOR I=1 TO N-1
110 IF B(I)<=B(I+1) THEN GOTO 160
120 LET F=1
130 LET T=B(I)
140 LET B(I)=B(I+1)
150 LET B(I+1)=T
160 NEXT I
170 IF F=1 THEN GOTO 90
180 FOR I=1 TO N STEP 1
190 PRINT A(I),B(I)
200 NEXT I
210 DATA 10
220 DATA 6,4,8,1,3,2,5,7,9,10
```

Um die Sortiergeschwindigkeit weiter zu steigern, gibt es eine ganze Reihe von Methoden. Eine besteht darin, FOR-Schleifen ineinander zu verschachteln. Dies nennt man auch NESTING. Nachfolgend finden Sie einige Beispiele, für mögliche und verbotene FOR... NEXT-Schleifen in Kombination.

Erlaubte Schleifen



Verbotene Schleifen



Überlegen Sie sich, weshalb die verbotenen Schleifen nicht zugelassen sind und in der Tat garnicht funktionieren können.

Folgendes Programmbeispiel sortiert den Ausdruck der TOP TEN der Schlagerparade nach Verkaufsziffern.

```
10 DIM R$(40,40)
20 DIM S(40)
30 LET I=1
40 PRINT "Plattenname: ";
50 INPUT N$
60 IF N$="" THEN GOTO 150
70 LET R$(I)=N$
80 PRINT "Verkäufe: ";
90 INPUT S(I)
100 LET I=I+1
105 IF I>40 THEN GOTO 150
110 GOTO 40
150 CLS
200 LET N=I-1
210 FOR I=1 TO N STEP 1
220 LET K=1
230 LET W$=R$(1)
240 LET MAX=S(1)
250 FOR J=1 TO N STEP 1
260 IF S(J) MAX>THEN LET MAX=S(J):LET K=J:LET W$=R$(J)
270 NEXT J
280 LET S(K)=0
290 PRINT W$,MAX
300 NEXT I
```

****

ÜBUNG 18

1. Entwerfen Sie ein Programm zur Aufstellung der Bundesligamannschaften nach der Anzahl der erzielten Tore.
2. Entwerfen Sie ein Programm für die Bundesliga mit der Punktzahl und der Tordifferenz.

****

Sortieraufgaben innerhalb eines größeren Programms sind besser in einem Unterprogramm aufgehoben. Nachstehend finden Sie ein nützliches Unterprogramm mit einer solchen Sortieraufgabe. Die Anweisung für den Unterprogrammaufruf lautet natürlich GOSUB 1000.

```
1000 REM UNTERPROGRAMM SORT
1010 LET F=0
1020 FOR I=1 TO N-1
1030 IF A(I)<=A(I+1) THEN GOTO 1080
1040 LET F=1:LET T=A(I):LET A(I)=A(I+1)
1070 LET A(I+1)=T
1080 NEXT I
1090 IF F=1 THEN GOTO 1010
1100 RETURN
```

Denken Sie daran, daß dies nur ein Unterprogramm ist.

KAPITEL 16

MEHRDIMENSIONALE FELDER

Lernziel: Sie sollen Daten in Tabellenform auswerten können.

In Kapitel 14 wurde die DIM-Anweisung gebraucht, um Felder zu bestimmen. Wir werden nun untersuchen, wie man Felder einsetzen kann, um zweidimensionale Datensätze zu speichern. Wie zuvor nehmen wir die DIM-Anweisung. Betrachten Sie folgenden Programmabschnitt:

ZWEIDIMENSIONALE FELDER

```
10 DIM X(5,4)
```

Diese Anweisung generiert ein Feld, bestehend aus fünf Zeilen und vier Spalten. Das Feld sieht so aus:

X(1,1)	X(1,2)	X(1,3)	X(1,4)
X(2,1)	X(2,2)	X(2,3)	X(2,4)
X(3,1)	X(3,2)	X(3,3)	X(3,4)
X(4,1)	X(4,2)	X(4,3)	X(4,4)
X(5,1)	X(5,2)	X(5,3)	X(5,4)

und könnte folgende Daten speichern:

	X	2*X	X^2	X^3
X=1	1	2	1	1
X=2	2	4	4	8
X=3	3	6	9	27
X=4	...	... etc	...	...
X=5	...	...	...	...

****

ÜBUNG 19 ZWEIDIMENSIONALE FELDER

1. Entwickeln Sie ein Programm, um obige Tabelle zu vervollständigen und auszudrucken.

2. Im Beispiel unten werden Teile der Prüfungspunkte dreier Klassen gegeben. Jede Angabe stellt eine Prozentangabe dar. Um die Fortschritte der Schüler zu beobachten, muß ein Programm folgende Aussagen zur Verfügung stellen:

a) den Durchschnittswert jeder Klasse in jedem Fach.
b) den Jahresdurchschnitt aller je Fach.
c) den tabellarischen Ausdruck der Ergebnisse wie unten.
d) den tabellarischen Ausdruck geordnet nach Pkt. für jedes Fach.

Schuljahr 13

Klasse	Schülernr.	Geographie	Geschichte	Mathe
BC2	1	58	62	23
	2	45	58	29
	3	67	76	53
	4	53	45	12
	5	68	72	43
BC3	25	46	43	51
	26	48	41	45
	27	53	41	55
	28	49	36	62
	29	51	68	65
BC4	55	43	38	42
	56	54	37	51
	57	47	56	49
	58	43	43	28
	59	54	45	43

Den Eingabeabschnitt des Programms sowie einige Wegweiser, wie das Gesamtprogramm aussehen könnte, wurde bereits fertiggestellt. Beachten Sie, wie Felder durch die Inputangaben generiert werden, so daß das Programm auch für viele andere Zwecke gebraucht werden kann.

```
10 REM Eingabeprogramm
20 INPUT "Maximale Schüleranzahl pro Klasse: ";SCHUELER
30 INPUT "Anzahl der Klassen: ";KLASSE
40 INPUT "Anzahl der Fächer: ";FACH
50 DIM R(SCHUELER,KLASSE,FACH)
60 REM Start der Eingabe
70 INPUT "Schüler,Klasse,Fach,Punkte ";P,C,S,M
80 IF P=0 THEN GOTO 110
90 LET R(P,C,S)=M
100 GOTO 70
110 REM Bitte weiterführen
```

****

```
1000 REM DURCHSCHNITTSNOTE
1010 REM Ergeb. werden in R angegeben
1020 FOR K=1 TO KLASSE STEP 1
1030 LET T=0
1040 LET N=0
1050 FOR P=1 TO SCHUELER STEP 1
1060 FOR S=1 TO FACH STEP 1
1070 LET M=R(P,C,S)
1080 IF R(P,C,S)<>0 THEN LET T=T+M:LET N=N+1
1090 NEXT S
1100 NEXT P
1110 LET A=T/N
1120 PRINT "Klassenschnitt ";C;" ist ";A
1130 NEXT C
```

In diesem Beispiel werden die Grunddaten für eine Reihe von Berechnungen benutzt. MTX-Basic enthält einen Befehl, um die benutzte Datei zurück in den Arbeitsspeicher zu setzen. Der Befehl lautet RESTORE und wird zusammen mit READ und DATA-Anweisungen wie folgt angewandt.

```
10  READ N
20  LET T=0
30  FOR S=1 TO N STEP 1
40  READ  X
60  LET T=T+X
70  NEXT S
80  LET A=T/N
100 PRINT : PRINT
110 PRINT "Durchschnittsnote ist: ";A
120 RESTORE 0
125 PRINT
126 PRINT "Schüler","Punkte","Abweichung"
127 PRINT
130 READ N
140 FOR  I=1 TO N STEP 1
150 READ Y
160 PRINT I,Y,Y-A
170 NEXT I
180 STOP
190 DATA 5
200 DATA 45,67,89,34,51
```

Siehe auch RESTORE im Anhang

KAPITEL 17

FORMATIEREN MIT PRINT

Lernziel: Sie sollen mit dem PRINT-Befehl Format-Anweisungen beherrschen lernen.

Für den Ausdruck von Tabellen haben wir bislang PRINT und Kommas eingesetzt, so daß die jeweils nächste TAB-Position benutzt wurde. Es mag aber sein, daß Ihre erste Spalte nicht mit der fixierten TAB-Position identisch, sondern beliebig sein soll. Mit Hilfe des CSR (CURSOR) Befehls in Verbindung mit PRINT läßt sich diese Festlegung umgehen.

Der erste Schritt besteht darin, daß die Cursor-Stellung mittels zweier Koordinaten bestimmt wird. Die erste bestimmt die Spalte, die zweite die Zeile auf dem Bildschirm.

```
CSR 3,4
```

z.B. plaziert den Cursor drei Zeilen rechts und vier Zeichen unter der linken oberen Ecke des Bildschirms. (Eine weitergehende Erläuterung finden Sie im Anhang zur GRAPHIK.)

Versuchen Sie folgendes simples Beispiel um Ihren Namen in der Mitte des Bildschirms abzubilden:

```
10        REM NAMENSAUSDRUCK
20        CSR 12,15:PRINT "NAME"
```

Sie werden bemerkt haben, daß wir zwei Anweisungen in einer Zeile untergebracht haben. Dies ist durchaus zulässig, wenn zwischen den Anweisungen ein Doppelpunkt gesetzt wird. Sie können auf diese Weise so viele Befehle unter einer Zeilennummer führen wie der EDIT-Bildbereich Platz hat.

KAPITEL 18

MATHEMATISCHE AUSDRÜCKE

Lernziel: Sie sollen mathematische Ausdrücke und ihre Verwendung im Programm kennenlernen.

In Kap. 3 wurde eine Quadratwurzel mittels eines "Bruchexponenten" berechnet. Falls Sie nicht schon selbst darauf gestoßen sind, ist es höchste Zeit zu zeigen, daß dies auch eleganter geht. Vergleichen Sie folgende Programme:

```
10        LET X=16                10        LET X=16
20        LET Y=X^0.5             20        LET Y=SQR(X)
30        PRINT Y                 30        PRINT Y
```

Wie Sie sehen, gibt es einen besonderen Ausdruck für die Berechnung eines Wurzelwertes. Eine vollständige Auflistung aller mathematischer Sonderausdrücke finden Sie im Anhang dieses Handbuches. Es lohnt sich, diese einmal genau zu studieren und bei Bedarf einzusetzen.

****

ÜBUNG 21 WINKELFUNKTIONEN

1) Das folgende Programm druckt die Werte von SIN und COS für den Bereich 0.1 und 0.9 aus. Verändern Sie diesen Bereich, um auch andere Werte berechnen und ausdrucken zu können.

```
10 PRINT "X",,"SIN(X)",,"COS(X)"
20 FOR X=.1 TO .9 STEP .1
30 PRINT
40 PRINT X,SIN(X),COS(X)
50 NEXT X
```

****

KAPITEL 19

STRING-FUNKTIONEN

Lernziel: Sie sollen STRING-Funktionen als Steuerelement einsetzen lernen.

Es gibt 2 Arten von STRING-Funktionen. Die erste Art wird eingesetzt, um den Computer Operationen ausführen zu lassen, etwa im Gegensatz zu Programmangaben. Die zweite Art wird eingesetzt, um STRINGs zu manipulieren.

Wir wollen zunächst Stringfunktionen als Operation behandeln. Das Keyboard kann mittels der INKEY$-Funktion "abgelesen" werden. Dies ist eine in der Praxis recht nützliche Eigenschaft, da sie erlaubt, Programme zu schreiben, die eine aktive Kommunikation mit dem Benutzer bedingen. Das kurze Beispielprogramm verlangt z.B. vom Anwender die Eingabe von "J" ehe das Programm fortfährt.

```
10 PRINT "Drücke J zur Fortführung"
20 LET A$=INKEY$
30 IF A$<>"J" THEN GOTO 20
40 PRINT "Sie haben J gedrückt"
```

Falls Sie den Programmablauf nicht ganz verstehen, können Sie noch die Zeile

```
25 PRINT A$;
```

einfügen.

Eine weitere nützliche Funktion ist CHR$. Diese Funktion wandelt Zeichencodes in die entsprechenden Zeichen um, die diesem Code zugeordnet sind. Beispiel:

```
PRINT CHR$(65)
```

bildet ein großes A auf der nächsten Druckposition ab, da 65 den ASCII-Code von 'A' darstellt. Im Anhang finden Sie eine vollständige Tabelle des ASCII-Codes.

Beispiel:

```
PRINT CHR$(12)    Löscht den Bildschirminhalt
PRINT CHR$(26)    Bringt den Cursor zur Ausgangsstelle
PRINT CHR$(10)    Bewegt den Cursor nach unten.
```

Die zweite Art von Stringfunktionen betrifft die Stringmanipulation.

Wollen Sie z.B. nur einen Teil eines Strings anzeigen, könnte man MID$ anwenden. Betrachten Sie folgendes Programm:

```
10 LET A$="ABCDEFG"
20 PRINT MID$(A$,3,2)
```

Nach RUN werden die Buchstaben CD auf dem Bildschirm erscheinen. Mit der Anweisung MID$(A$,3,2) haben Sie den Computer angewiesen, zur dritten Stelle des Strings A$ zu gehen und zwei Zeichen auszudrucken.

LEFT$ und RIGHT$ werden benutzt, um den Computer anzuweisen, von aus links (LEFT) oder von rechts aus (RIGHT) zu zählen und dann die entsprechende Anzahl von Zeichen auszudrucken. Ersetzen Sie in obigem Programm Zeile 20 durch:

```
20 PRINT LEFT$(A$,3)      ABC erscheint
20 PRINT RIGTHT$(A$,3)    EFG erscheint
```

KAPITEL 20

EINFACHE SPIELE UND ZUFALLSZAHLEN

Lernziel: Sie sollen Methoden kennenlernen, mit denen einfache Spiele entworfen werden.

Die neuesten Computerspiele verwenden komplexe Graphik, Programmschleifen, bedingte Sprünge und Unterprogramme. In diesem Kapitel wollen wir weniger die Graphik behandeln und unser Augenmerk auf die Programmiertechnik werfen. Wenn Sie sich später sicherer fühlen, können Sie noch die Graphikeigenschaften des MTX voll ausnutzen.

Ein wichtiger Aspekt bei jedem Spiel ist die Unberechenbarkeit der nächsten Spielzüge. Um diese "Eigenschaft" zu integrieren, bedienen Sie sich der BASIC-Funktion RND und des Befehls RAND. Zufallszahlen (engl.: random numbers) werden von einem Zufallsgenerator erzeugt. Um diesen zu initialisieren, kann mit der RAND-Anweisung ein Startpunkt gesetzt und mit der RND-Funktion ein Zahlenlimit gegeben werden. Wir haben ein Programm entworfen, das Zahlen für eine Lotterie ausdruckt. In diesem Fall dürfen zwanzig Lose gezogen werden, so daß auch zwanzig Zahlen generiert werden müssen.

```
10 RAND 5
20 FOR I=1 TO 20 STEP 1
30 PRINT INT(RND*64+1),
40 NEXT I
50 PAUSE 5000
```

Falls Sie das Programm ausgiebig getestet haben, werden Sie feststellen, daß es einen "Schönheitsfehler" hat. Es geht immer von gleichen Ausgangspunkten aus und generiert demnach auch immer die gleichen Zahlen. RAND 6 würde zwar andere Zahlen produzieren, aber auch immer wiederkehrend. Dies gilt für alle möglichen ganzen Zahlen.

Um "echte" Zufallszahlen zu erhalten, verwenden wir daher einen negativen Wert. Versuchen Sie RAND -5 in Zeile 10. Wie bei positiven Werten kann jede beliebige ganze Zahl verwendet werden.

RND produziert nur Zufallszahlen zwischen 0 und 0.9999999. Um eine andere Skala zu erhalten, kann das Ergebnis mit einem Faktor multipliziert werden.

Zeile 30 besagt, daß der Computer eine ganzzahlige Zahl zwischen 1 und 64 ausdrucken soll. Der Computer würde nun bei 0 beginnen und bis 63 fortfahren, wenn keine weitere Anweisung erfolgt. Die einfachste Lösung besteht darin, zu jeder generierten Zahl eine eins hinzuzuzählen:

```
INT(RND*64+1)
```

****

ÜBUNG 22 ZUFALLSZAHLEN

1) Entwerfen Sie ein Programm, das Zufallszahlen zwischen 1 und 99 in zehn Spalten ausdruckt.

2) Vervollständigen Sie nachfolgendes Würfelprogramm.

```
10  RAND 5000
20  LET D1=INT(RND*6+1)
30  LET D2=INT(RND*6+1)
40  PRINT:PRINT
50  PRINT "Würfel 1= ";D1
60  PRINT "Würfel 2= ";D2
70  IF D1=D2 THEN GOTO 80 ELSE GOTO 100
80  PRINT "Leertaste für neuen Wurf"
85  LET A$=INKEY$
86  IF A$<>" " THEN GOTO 85
90  GOTO 20
100 INPUT "Neuer Versuch (J/N)? ";A$
110 IF A$="J" OR a$="j" THEN GOTO 20 ELSE GOTO 120
120 PRINT "Spielende"
```

3) Schreiben Sie das Würfelprogramm um, so daß 4 Würfel von 3 Spielern benutzt werden können. Fügen Sie eine Zähltafel für das jeweils aktuelle Würfelergebnis ein.

****

Folgende zwei Programme simulieren das Spiel Bingo.

Fügen Sie die Programme als Unterprogramme in ein Hauptprogramm ein, das die Karten in einem Feld formatiert, so daß es auch auf dem Bildschirm verfolgt werden kann. Versuchen Sie das Spiel zunächst mit 2 Spielern.

BINGO

```
10  REM BINGO ZAHLEN
20  DIM A(99)
30  CLS
40  PRINT "Drücke RETURN für die nächste Zahl"
50  INPUT A$
60  LET X=INT(99*RND+1)
70  IF A(X)=1 THEN GOTO 60
80  LET A(X)=1
90  PRINT X
100 GOTO 40
```

INGO KARTEN

```
10  REM BINGO KARTEN
20  DIM B(30)
30  DIM R(100)
40  CLS
50  FOR I=1 TO 15
60  LET X=INT(RND*99+1)
70  IF R(X)=1 THEN GOTO 60
80  LET R(X)=1
90  LET N=INT(RND*30+1)
100 IF B(N)<>0 THEN GOTO 80
110 LET B(N)=X
120 NEXT I
130 FOR I=1 TO 10
140 FOR J=1 TO 3
150 PRINT B((J-1)*10+I),
160 NEXT J
170 PRINT
180 NEXT I
```

KAPITEL 21

MATRIXRECHNUNG

Lernziel: Sie sollen Matrixoperationen in Basic kennenlernen.

Die zweidimensionalen Felder in Kapitel 14 waren eine Art von Matrix. Zwischen einem Feld und einer Matrix besteht nicht unbedingt ein grundlegender Unterschied in der Weise, wie ein Feld behandelt wird. Bei zweidimensionalen Feldern haben wir es z.B. mit zwei verschiedenen Parametern eines Feldes zu tun, wie z.B. die Geographieergebnisse der Schulklasse. Bei Matrixoperationen innerhalb eines Programms oder Unterprogramms können wir die Matrix als Einheit ansehen.

Es gibt hierfür eine ganze Reihe von nützlichen Anwendungen, die sich auf tabellarische Informationen beziehen. Matrizen können addiert, multipliziert, kopiert oder mit einer Konstante in Beziehung gesetzt werden. So lassen sich z.B. Verkaufszahlen eines Monats zu Quartals- und Jahresergebnissen addieren.

Untenstehende Unterprogramme können für das Aufstellen, Eingeben und Ausgeben von Matrixdaten benutzt werden. Wir haben hier ein 3x3 Matrix gewählt, aber es ist offenkundig, daß mit einer Änderung der Werte I,J eine Matrix beliebiger Groeße definiert werden kann.

UNTERPROGRAMM ZUM AUSDRUCK EINER MATRIX

```
1000 FOR I=1 TO 3
1010 FOR J=1 TO 3
1020 PRINT A(I,J)
1030 NEXT J
1040 NEXT I
1050 RETURN
```

UNTERPROGRAMM, UM DIE MATRIX A AUF NULL ZU SETZEN

```
2000 FOR I=1 TO 3
2010 FOR J=1 TO 3
2020 LET A(I,J)=0
2030 NEXT J
2040 NEXT I
2050 RETURN
```

UNTERPROGRAMM ZUR DATENEINGABE

```
3000 INPUT "I,J,DATA ";I,J,D
3010 IF D=99999 THEN RETURN
3020 LET A(I,J)=D
3030 GOTO 3000
```

Diese Methode ist geeignet für die Addition, Übertragungen und Multiplikationen von Konstanten zur Matrix. So würde der Ausdruck von Zeile 1020 in dem ersten Unterprogramm mit

```
1020      LET A(I,J)= B(I,J)
```

die Übertragung der Werte der Matrix B auf die Matrix A bewirken.

```
1020      LET C(I,J)= A(I,J)+B(I,J)
```

Hier wird die Matrix A der Matrix B hinzuaddiert und die Summe in die Matrix C kopiert.

```
1020      LET A(I,J)= 5*A(I,J)
```

Alle Elemente der Matrix A werden hier mit der Konstanten 5 multipliziert.

Die Multiplikation zweier Matrizen miteinander ist schon etwas komplizierter und muß den Regeln der Matrixalgebra entsprechen. Betrachten Sie folgendes Beispiel, wo Matrix A R Zeilen und C Spalten, Matrix B C Zeilen und S Spalten besitzt.

```
4000 REM Unterprogramm zur Multiplikation von Matrizen
4010 FOR I=1 TO R STEP 1
4020 FOR J=1 TO C STEP 1
4030 LET D(I,J)=0
4040 FOR K=1 TO C STEP 1
4050 LET D(I,J)=D(I,J)+A(I,K)*B(K,J)
4060 NEXT K
4070 NEXT J
4080 NEXT I
4090 RETURN
```

****

ÜBUNG 23

Herr Angelhaken besitzt 3 Fischläden, deren jeweiligen Umsätze während der letzten vier Quartale in der nachfolgenden Tabelle ausgedruckt sind:

		Barsch	Makrele	Hering
Geschäft 1	JAN - MAR	3,456	460	212
	APR - JUN	2,458	238	146
	JUL - SEP	1,845	67	35
	OKT - DEZ	4,153	354	286
Geschäft 2	JAN - MAE	2,998	342	189
	APR - JUN	2,135	154	89
	JUL - SEP	1,225	52	38
	OKT - DEZ	2,509	189	139
Geschäft 3	JAN - MAE	4,806	589	354
	APR - JUN	2,678	453	302
	JUL - SEP	2,688	220	148
	OKT - DEZ	4,766	554	386

Drücken Sie obige tabellarische Daten in der abgebildeten Form mit Hilfe der Unterprogramme aus. Da Matrizen addiert werden können, ist es möglich, folgende Daten zu ermitteln:

1) Die Halbjahresumsätze der Geschaäfte
2) Den Jahresumsatz der Geschäfte
3) Den Gesamtumsatz aller Geschäfte
4) Der prozentuale Anteil eines jeden Geschäftes am Gesamtmonatsumsatz

TEIL 2

NODDY

In dem vorangegangenen Kapiteln erlebten Sie, wie schwierig es oft ist, Bildschirmtexte in BASIC an der richtigen Stelle zu plazieren. Die neue Sprache NODDY wurde vor allem deshalb eingeführt, um diesen Umstand zu verbessern. Der zweite Vorteil von NODDY liegt darin, daß Sie - mit nur geringen Programmierkenntnissen - Ihre eigenen interaktiven Programme schreiben können. Wie wohl zu erwarten, bedarf die Programmerstellung eine vorausschauende Planung und das Verständnis um die zur Verfügung stehenden Mittel. Auch die Einführsprache NODDY bildet hier keine Ausnahme. Da NODDY jedoch nur mit insgesamt elf Befehlen auskommt, wird schon hier die Schwierigkeitsstufe niedrig angesetzt.

DER NODDY-BEFEHLSSATZ

B	BRANCH	E	ENTER	P	PAUSE
I	IF	A	ADVANCE	L	LIST
G	GOTO	R	RETURN	O	OFF
		S	STACK	D	DISPLAY

Bedeutung und Gebrauch dieser Befehle werden am Beispiel einiger Programme offensichtlich.

NODDY wird von BASIC aus mit der Eingabe "NODDY" aktiviert. Der Zugang zu NODDY über BASIC wurde bewußt gewählt, um Programmen zu ermöglichen, beide Sprachen nutzen können.

Eingabe: NODDY (RET)

Noddy>

erscheint am unteren Bildrand.

Eingabe: NAME

Noddy>NAME (Beachten Sie dabei für einen späteren Aufruf die Großschreibung)

Nach RETURN erscheint NAME am oberen Bildrand.

Dies ist nun der Titel dieser Bildschirmseite.

Steuern Sie nun den Cursor mit den Edit-Tasten über den Bildschirm, um an beliebigen Stellen einige persönliche Daten zu schreiben. Vergessen Sie dabei bitte nicht, daß jede NODDY-Seite als vollständige Eingabe behandelt wird. Sie dürfen also die RETURN-Taste erst dann drücken, wenn diese Seite abgeschlossen werden soll. Sollte demnach versehentlich (RET) vorzeitig ausgelöst werden, kann die betreffende Seite durch Eingabe des Seitennamens NAME und (RET) sofort zurückgeholt werden.

DRUCKEN SIE AUF KEINEN FALL DIE CLS-TASTE, DA HIERMIT DIE GESAMTE SEITE GELÖSCHT WIRD.

Sobald nun alle gewünschten Eingaben gemacht wurden, wird die Seite mit (RET) abgeschlossen und automatisch gespeichert.

Sie haben nun Ihre erste NODDY-Seite mit dem Namen NAME angelegt.

Wenn Sie nun DIR (RET) eingeben (NODDY-Befehle müssen immer in Großbuchstaben eingegeben werden), wird der Bildschirm gelöscht und NAME erscheint am linken oberen Bildschirmrand. DIR ist also nichts weiter als die Anweisung, das Inhaltsverzeichnis aller NODDY-Seiten anzuzeigen. Beachten Sie bitte die Eingabe mit DIR, da 'dir' eine neue Seite mit der Bezeichnung 'dir' anlegen würde.

Geben Sie nun nochmals NAME ein, erscheint das zuvor geschriebene auf dem Bildschirm. Eine Ergänzung oder Änderung über die Cursortasten ist jederzeit möglich und muß mit (RET) abgeschlossen werden.

Denken Sie daran, daß die CLS-Taste dazu dient, die abgebildete Seite völlig zu löschen. Obwohl diese Funktion hilfreich und schnell ist, um unerwünschte Seiten zu löschen, kann das versehentliche Auslöschen doch frustrierend sein.

NODDY ermöglicht das Anlegen und Speichern von textlichen Informationen.

Mit dem Eintippen von NAME (RET) wurde der Computer angewiesen, eine Bildschirmseite mit der Bezeichnung NAME anzulegen.

Erzeugen Sie weitere Bildseiten mit anderen Namen. Nach jeder neuen Namensgebung sollte nach (RET) Noddy> links unten erscheinen.

Sobald Noddy> erscheint, können Sie:

1) eine neue Seite durch Eingabe eines neuen Namens anlegen
2) DIR eintippen, um eine Auflistung aller Seiten zu sehen
3) eine bestimmte Seite durch Eintippen des bereits vorhandenen Namens ansehen
4) mittels CLS (RET) zurück ins BASIC gehen. Dies ist die einzige Gelegenheit, um CLS ohne Probleme anzuwenden.

Um sicherzugehen, daß der richtige Modus für CLS ansteht, ist es ratsam, zuerst immer DIR anzuwählen, ehe Sie zurück ins BASIC gehen. Dies bietet gleichzeitig die Gelegenheit zur Überprüfung, ob alle Bildseiten vorhanden sind und verhindert einen ungewollten Datenverlust.

Also: Rückkehr zum BASIC von NODDY aus:

Eingabe: DIR (RET)
CLS-Taste und (RET)
Ready erscheint unten am Bildrand.

Die Rückkehr ins BASIC bedeutet nicht den Verlust der NODDY-Seiten, und zwar so lange, wie der Computer eingeschaltet bleibt. Ein erneuter Aufruf von NODDY wird dies bestätigen.

NODDY-Daten können wie BASIC-Programme gemäß der SAVE-Anleitung in Kapitel 1 auf Kassette gespeichert werden. Dementsprechend bewirkt NEW auch das Löschen der NODDY-Daten. Sind BASIC- und NODDY-Daten in einem Programm untergebracht, gilt entsprechendes. Bei dem MTX sind NODDY und BASIC als Sprachen zu verstehen, die in sehr enger Beziehung zueinander stehen.

NODDY-Programme, die mit dem NODDY-Befehlssatz geschrieben werden, sind besondere Seiten, die gemäß obigem Verfahren angelegt werden. Um dies zu demonstrieren, haben wir vier Programme zur Speicherung von Telefonnummern geschrieben. In aufsteigender Folge zeigen wir, wie NODDY effizient benutzt werden kann.

Das erste Programm besteht aus einer Seite mit Telefonnummern und einer Programmseite. Die Telefonseite werden wir UDO, die Programmseite PROG1 nennen.

Eingabe: NODDY (RET)

Eingabe: UDO (RET)
Udo's Telefonnummer lautet: 0541/53905

Diese Seite wird nun mit (RET) abgespeichert.

Die Programmseite PROG1 nutzt die drei Befehle DISPLAY, PAUSE und RETURN. Jedem Befehl ist ein * vorausgestellt, damit der Computer das Nachfolgende als einen Befehl und nicht als Text versteht. Geben Sie nun folgendes ein:

```
PROG1
     *DISPLAY UDO.
     *PAUSE    *PAUSE
     *RETURN
     (RET)
```

Die erste Zeile '*DISPLAY UDO.' weist den Computer an, die Seite mit dem Namen UDO anzuzeigen. Der Bezug auf eine Bildseite in diesem Zusammenhang wird immer mit einem Punkt abgeschlossen. Falls dies vergessen wird, kann der Rechner die Anweisung nicht ausführen.

Die zweite Zeile bewirkt die Darstellung des Textes der Seite UDO für die Dauer zweier PAUSEn (PAUSE hat die Länge von ca. 1 sec.).

Die dritte Zeile mit dem Befehl RETURN bewirkt die Rückkehr zum BASIC, nachdem die Pause beendet ist.

Alle Befehle lassen sich auch in einer Kurzform eingeben, da für den NODDY-Interpreter immer nur der erste Buchstabe des einzelnen Befehls signifikant ist. Derselbe Effekt des obigen Programms PROG1 würde auch durch folgendes Programm erzielt:

```
PROG1
     *D UDO.
     *P *P
     *R
```

Eingabe: DIR zur Anzeige der Seitennummern

Um das Programm ablaufen zu lassen, müssen wir zurück ins BASIC, wofür nur (CLS) und (RET) erforderlich sind. Die Anzeige Noddy> wird durch Ready ersetzt. Die Anweisung für die Ausführung eines NODDY-Programms lautet PLOD. Hiernach muß immer die betreffende Programmseite in Anführungszeichen folgen.

Eingabe: PLOD "PROG1"

Soll ein NODDY-Programm mehrmals ablaufen, ist es sinnvoll, die PLOD-Anweisung mit einer Zeilennummer zu versehen.

```
10        PLOD "PROG1"
```

Hier kann man nun einfach RUN eingeben, wenn das NODDY-Programm aktiviert werden soll.

Das zweite NODDY-Programm (PROG2) erlaubt die Rückkehr zum BASIC durch Tastendruck (RET). Die Taste (RET) wird auch oft als (ENTER)-Taste bezeichnet. Der Befehl für diesen Ablauf lautet *ENTER und ähnelt dem bekannten INPUT in BASIC. Sobald *ENTER im Programm ansteht, wartet der Computer, bis eine oder mehrere Tasten und die (RET)-Taste gedrückt werden, ehe das Programm fortgesetzt wird.

Geben Sie das Programm ein und nennen Sie es PROG2. Sobald die Seite UDO erscheint, reicht eine beliebige Taste und (RET) oder nur (RET) und das Programm fährt fort, bis Ready die Rückkehr ins BASIC anzeigt.

```
PROG2
     *DISPLAY UDO.
     *ENTER
     *RETURN
```

Programm 3 nutzt die Befehle *IF, *GOTO und Labels, um Sie mit mehr Kontrollmöglichkeiten auszustatten. *IF wird dazu gebraucht, um Entscheidungen bezüglich der *ENTER-Eingabe zu treffen. Sollte die Entscheidung positiv bewertet werden, wird der Computer programmgemäß zu einem näher bezeichneten Programmschritt weitergehen. Dies geschieht mittels eines Labels. So bedingt die Anweisung der Zeile 3 (*IF R,r), daß die *ENTER-Eingabe mit R verglichen werden soll und, sollte die Taste 'R' gedrückt worden sein, die Programmausführung ab der Zeile, der ein 'r' voransteht, weitergeführt werden soll. Damit der Buchstabe 'r' am Anfang einer neuen Programmzeile nicht mit anderen verwechselt wird, ist diesem ein '^' vorangestellt.

(Label = engl. Marke) können beliebige Zeichen des Zeichenvorrats der Tastatur sein und Sie können sich Ihre Systematik hierfür aussuchen. Wesentlich ist, daß das voranzustellende Zeichen ^ angewandt wird.)

Wird eine andere Taste außer 'R' gedrückt, fährt das Programm mit *GOTO PROG3. fort. Der GOTO-Befehl wird in diesem Fall dazu benutzt, die Programmsteuerung zurück zum Anfang zu bringen. Sonst wird *GOTO zur Adressierung anderer Programmseiten eingesetzt. Beachten Sie den Punkt hinter PROG3 und die Stellung von ^r.

```
PROG3
     *DISPLAY UDO.
     *ENTER
     *IF R,r
     *GOTO PROG3.
  ^r *RETURN
```

BRANCH

Eine bessere Möglichkeit, Programmverzweigungen zu steuern, bietet sich mit dem Befehl *BRANCH. Programm 4 verdeutlicht die Anwendung von *BRANCH und den Gebrauch von Labels für den Einsatz von *ENTER, um UDO auf dem Bildschirm auszudrucken.

```
PROG4
     ^t   *ENTER
          *IF F,a
          *IF R,r
          *BRANCH t

     ^a   *DISPLAY UDO. *BRANCH t
     ^r   *RETURN
```

Das letzte Programm dieser Beispielreihe ermöglicht es uns, NODDY-Seiten als ein reguläres Telefonverzeichnis zu nutzen. Erste Voraussetzungen hierfür sind mehrere Seiten.

```
Eingabe: JOE (RET)
         Joe's Telefonnummer ist 0541/55488

         MONIKA (RET)
         Monika's Telefonnummer ist 0579/4873
```

Geben Sie nun die Programmseite ein:

```
PROG5
  ^t      *ENTER
          *IF U,a
          *IF J,b
          *IF M,c
          *IF RET,r
          *BRANCH t

  ^a      *DISPLAY UDO              *BRANCH t
  ^b      *DISPLAY JOE              *BRANCH t
  ^c      *DISPLAY MONIKA           *BRANCH t
  ^r      *RETURN
```

Die ersten sechs Zeilen des Programms bestehen aus einer Schleife, wo der Computer darauf wartet, eine Eingabe für U, J, M oder RET zu empfangen. Wird etwas anderes eingegeben, erfolgt bei *BRANCH eine Verzweigung nach t. Wird RET eingegeben, erfolgt die Verzweigung nach r und *RETURN mit der Rückkehr ins BASIC.

Wird U, J oder M gefolgt von (RET) gedrückt, geht die Kontrolle auf die Labels a,b oder c über.

Sobald die entsprechenden Seiten angezeigt wurden, verzweigt das Programm wieder nach t zum Anfang des Programms und Sie können das Ganze wiederholen.

****

ÜBUNG 24 NODDY

Verbessern Sie das letzte Programm, indem Sie am Programmbeginn ein Inhaltsverzeichnis anführen und das Programm nach jedem Suchlauf zu diesem Inhaltsverzeichnis zurückkehrt. Ein Inhaltsverzeichnis (oder auch Menu genannt), ist bereits fertiggestellt. Ihre Aufgabe ist die Korrektur bzw. Erweiterung der Programmseite PROG5.

MENU

Dieses Telefonverzeichnis enthält 3 Telefonnummern:

UDO, JOE, MONIKA

Die jeweilige Nummer wird durch Eingabe des ersten Buchstaben gefolgt von (RET) angezeigt.
Für eine Rückkehr ins BASIC geben Sie RET ein.

Eine weitere Übung besteht darin, ein eigenes Adressverzeichnisprogramm zu schreiben.

****

Als nächstes Beispiel für eine NODDY-Anwendung sehen wir nun, wie eine Buchsimulation erfolgen könnte.

Das Lesen eines Buches erfordert verschiedene physikalische Anstrengungen. Seiten müssen umgeblättert werden, Inhaltsverzeichnisse durchgelesen werden, um bestimmte Details zu finden. Es kann auch nur sein, daß der Leser bereits Gelesenes wiederfinden oder den Ausgang des Buches vorab lesen möchte.

Die Simulierung eines solchen Programms in BASIC würde einen beträchtlichen Aufwand bedeuten. NODDY benötigt lediglich eine Programmseite, eine Seite für das Inhaltsverzeichnis und eine Seite für jedes Kapitel. Der Plan für das Programm sieht wie folgt aus:

Stufe 1

WELCHES KAPITEL

KAP 1	KAP 2	KAP 3
HAUSTIERE	HOFTIERE	ZOO

Stufe 2

WELCHE SEITE

S1 HUNDE	S1 SCHAFE	S1 LOEWEN
S2 KATZEN	S2 SCHWEINE	S2 ZEBRAS
S3 MAEUSE	S3 RINDER	S3 SCHLANGEN

Es gibt keine zwingende Vorschrift darüber, wie das Buch angelegt wird. Es ist allerdings oft einfacher, mit dem Inhalt zu beginnen, da dieser eher geändert wird.

Der Titel des Buches lautet SAEUGETIERE und besteht aus vier Kapiteln (gemäß Stufenplan) mit der Titulierung HUNDE für Seite eins.

Eingabe: HUNDE (RET)

Der verbleibende Rest der Seite kann nun mit Informationen über Hunde aufgefüllt werden, bis mit (RET) alles abgespeichert wird. Diese Routine gilt nun für alle Tiere des Stufenplans.

Als nächstes muß ein Verzeichnis für jedes Kapitel angelegt werden.

Wie zuvor geben wir zuerst eine Seitenberechnung ein, gefolgt von den eigentlichen Daten:

```
KAP1    H  HAUSTIERE
        S1 HUNDE
        S2 KATZEN
        S3 MAEUSE
```

Vervollständigen Sie die Seiteninhalte für jedes Kapitel. Eine Kontrolle ermöglicht die Eingaben DIR, womit alle angelegten Seiten wie folgt angezeigt werden sollten:

KAP1	KAP2	KAP3
HUNDE	SCHAFE	LOEWEN
KATZEN	SCHWEINE	ZEBRAS
MAEUSE	RINDER	SCHLANGEN

Um das Buch an einem bestimmten Kapitel aufzuschlagen, benötigen wir noch ein Seitenverzeichnis mit den Kapitelangaben. Damit diese Seite nicht mit dem bereits bestehenden Inhaltsverzeichnis verwechselt wird, nennen wir diese Seite TITEL.

Eingabe: TITEL (RET)

Wählen Sie mit Eingabe 1,2 oder 3 ein Kapitel oder RET für die Rückkehr ins BASIC.

```
1 KAPITEL  HAUSTIERE
2 KAPITEL  HOFTIERE
3 KAPITEL  ZOO
R RETURN
```

Damit wird die Inhaltsbestimmung des Buches abgeschlossen, so daß nun nur noch die Programmseite geschrieben werden muß. Wie zuvor geben wir dieser Seite eine Bezeichnung (SAEUGETIERE) gefolgt von den Programmanweisungen. Wie Sie unten sehen, haben wir den vollständigen Befehlswortlaut, wie schon vorher erwähnt, durch einen einzigen Buchstaben ersetzt. Ferner ist zu beachten, daß mehrere Befehle auf einer Zeile Platz finden und daß alle *IFs zusammengefaßt sind. Die Label (gekennzeichnet durch ^) vor einem Buchstaben sind so plaziert, daß deutlich wird, ob ein Buchstabe sich auf eine Titelseite oder Textseite bezieht. Ihr NODDY-Programm läuft auch ohne diese vorsichtige Strukturierung. Eine Programmgliederung in der gezeigten Art vermindert jedoch erheblich die Anzahl der möglichen Fehler.

Prüfen Sie bei jeder Eingabe die Funktion der Zeile und wie die Befehle ausgeführt werden.

Zeile 1 zeigt das Hauptinhaltsverzeichnis mit dem Namen TITEL und befiehlt dem Computer, auf eine *ENTER-Eingabe über die Tastatur zu warten.

Zeile 2 vergleicht Ihre Eingabe mit den möglichen Eingaben 1,2,3 oder R für RETURN und befiehlt dem Computer, die entsprechenden Label a, b, c oder r zu finden und von da fortzufahren. Wird ein anderer Buchstabe oder eine andere Zahl eingegeben, bringen die Steuerbefehle das Programm mittels *BRANCH t zurück zum Anfang.

Untersuchen Sie auch die restlichen Programmschritte auf diese Weise.

```
SAEUGETIERE
 ^t              *D TITEL. *E
                 *I 1,a *I 2,b *I 3,c *I R,r *B t

 ^a              *D KAP1. *E
                 *I S1,g *I S2,h *I S3,i *B t
        ^g       *D HUNDE.      *B d
        ^h       *D KATZEN.     *B d
        ^i       *D MAEUSE.     *B d
 ^b              *D KAP2. *E
                 *I S1,j *I S2,k *I S3,l *B t
        ^j       *D SCHAFE.     *B d
        ^k       *D SCHWEINE.   *B d
        ^l       *D RINDER.     *B d
 ^c              *D KAP3. *E
                 *I S1,m *I S2,n *I S3,o *B t
        ^m       *D LOEWEN.     *B d
        ^n       *D ZEBRAS.     *B d
        ^o       *D SCHLANGEN. *B d
 ^d              *E *B t
 ^r              *R
```

Die Eingabe (RET) speichert die Programmseite ab und, falls keine Eingabefehler vorliegen, kann das Programm nun ablaufen. Ein Fehler wird durch den Rücksprung ins BASIC und eine dieser drei Fehlermeldungen angezeigt:

"MISSING SYMBOL" bedeutet ein fehlendes * oder . bei der Seitenbenennung.

"NO DATA ERROR" bedeutet, daß der Computer eine Seitenbezeichnung sucht, die nicht vorhanden ist oder falsch geschrieben wurde.

"OVERFLOW" kommt dann vor, wenn der Computer auf der Suche nach einem Label oder Befehl das Ende einer Programmseite ohne Ergebnis erreicht hat.

Diese Fehlermeldungen sind nur grobe Hinweise auf die Fehlerquelle, so daß die gesamte Seite auf mögliche Fehler hin untersucht werden sollte.

Die Seite kann mittels (CLS) und der Eingabe NODDY korrigiert werden. Erscheint Noddy> auf dem Bildschirm, ist die betreffende Seite nach der Eingabe der Titelbezeichnung SAEUGETIERE für eine Korrektur bereit. Sollten alle Zeichen und Befehle zur Zufriedenheit vorhanden sein, bringt (RET), (CLS) und (RET) Sie zurück ins BASIC.

Lassen Sie das Programm mit dem PLOD-Befehl ablaufen. Es ist ganz offensichtlich, daß dieses Programm nur bei einem Buch mit drei Kapiteln á drei Seiten funktionieren kann. Als Buch wäre es ohne eine radikale Änderung im Programmablauf sehr beschränkt brauchbar. Ferner würde so ein Programm kaum auf eine Seite passen. Dieses Problem wird mit dem Befehl *GOTO, der das Umschalten auf eine andere Programmseite ermöglicht, gelöst. Eine besondere Lösung bestände darin, jede neue Kapitelseite als neue Programmseite zu definieren, so daß anstelle von *B t die Anweisung *GOTO SAEUGETIERE. treten würde.

****

ÜBUNG 25 NODDY-BUCHPROGRAMM

Schreiben Sie das Programm so, daß 4 Kapitel mit je 4 Seiten zur Verfügung stehen.

****

Der *GOTO-Befehl ist einer von insgesamt vier Programmsteuerbefehlen.

Programmseiten werden auf einem Stapel, den man wie ein Stapel Teller behandeln kann, gespeichert. Werden Sie gebraucht, kann man sie, wie bei einem Stapel Teller, von oben abnehmen. Stellen Sie sich vor, es gäbe drei Programme, die nacheinander ablaufen sollen. Mit dem STACK-Befehl sähe die Vorrausetzung dazu so aus:

*STACK PROG3,PROG2,PROG1.

Die Programme könnten nun in der Reihenfolge PROG1,PROG2 und PROG3 vom Stapel genommen werden. Der Befehl *STACK teilt dem Computer mit, in welcher Reihenfolge die bezeichneten Programme auf den Stapel gebracht werden sollen. Die Programmnamen könnten also einer nach dem anderen von diesem Stapel genommen, die entsprechenden Programme aufgerufen werden und damit läuft PROG1 als erstes, PROG2 als zweites und PROG3 als drittes Programm ab, weil *STACK sie, wie oben aus dem Beispielbefehl zu ersehen, in der Reihenfolge PROG3, PROG2 und PROG1 auf den Stapel ablegt und man natürlich auch nicht den untersten Teller zuerst benutzen würde.

Der Befehl *ADVANCE befiehlt dem Computer, das oberste Programm vom Stapel zu nehmen und es auszuführen

Der *OFFSTACK-Befehl befiehlt dem Computer, das nächste Programm vom Stapel zu nehmen, ohne es jedoch auszuführen.

Wir haben eine Reihe von Programmen geschrieben, die die Funktionsweisen dieser Befehle verdeutlichen.

Legen Sie folgende drei Seiten mit den Namen AA, BB und CC an.

AA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

BB

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

CC

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

Die zentrale Programmseite heißt PROG und sieht so aus:

PROG

```
        *S PROG,PA,PB,PC.
        *A
```

Weiterhin gibt es noch die untergeordneten Programmseiten PA, PB und PC.

PA

```
        *D AA. *P *P *A
```

PB

```
        *D BB. *P *P *A
```

PC

```
        *D CC. *P *P *A
```

Sobald Sie mit PLOD "PROG" das Programm aufrufen, speichert der Computer vier Programmseiten auf dem Stapel ab (Hier bildlich wie ein Tellerstapel dargestellt):

```
        PC
        PB
        PA
        PROG
```

*A am Ende von PROG nimmt das erste Programm vom Stapel (PC) und führt es aus. CC wird für die Dauer der Pause *P angezeigt. Währenddessen wurde PC entfernt, so daß der Stapel nun wie folgt aussieht:

```
PB
PA
PROG
```

*A am Ende von PC instruiert der Computer PB vom Stapel zu nehmen, zum Arbeitsspeicher zu übertragen, die Routine auszuführen und dann zu entfernen. Dieser Vorgang wiederholt sich, bis PROG erreicht wird und der Stapel erneut angelegt wird. Auf diese Weise haben wir eine Programmschleife geschaffen.

Das Programm kann mittels der BRK-Taste unterbrochen werden. Fügen Sie einen *OFFSTACK-Befehl ins Programm PC ein (wie unten demonstriert), so daß PB jedesmal vom Stapel genommen, aber nicht ausgeführt wird.

PC

```
*D CC. *P *P *O *A
```

****

ÜBUNG 26 NODDY-PROGRAMMBEFEHLE

Entwerfen Sie anhand der hier gezeigten Programmstrukturen ein Programm zur effizienteren Durchsicht der vier Kapitelseiten aus Übung 25.

Ein weiterer Vorteil ergäbe sich aus der Einfügung von *OFFSTACK, um eine Option zu bilden, nur solche Seiten des Buches "durchzublättern", die noch nicht betrachtet wurden.

****

Einen Befehl haben wir noch nicht behandelt: *LIST. LLIST oder LPRINT können bei NODDY nicht genutzt werden, da wir es ja immer mit ganzen Seiten zu tun haben. Weiterhin muß der Drucker so eingestellt werden, daß er nur eine Zeilenlänge von 39 Zeichen verarbeiten kann. Hierzu dient das technische Handbuch Ihres Drukkers. Bei einem EPSON-Drucker z.B. wäre

```
LPRINT CHR$(27);"Q";CHR$(39)
```

erforderlich. Eine NODDY-Seite wird dann einfach mit dem NODDY-Befehl *LIST, gefolgt vom Seitentitel und einem Punkt, gedruckt.

```
*LIST TITEL.
```

Der Drucker druckt die Seite mit der Bezeichnung TITEL aus.

NODDY ist eine neue, noch entwicklungsfähige Sprache mit nur wenigen Regeln für die Programmstruktur. Dies ist nur ein bescheidener Versuch, Ihnen einige Hinweise zu geben, wie ein NODDY-Programm geschrieben werden könnte. Es liegt an Ihnen, eigene Programmiertechniken auszugestalten. (Die NODDY-Befehle im MTX-ROM sind nur ein Teil der vollständigen NODDY-Sprache, wie sie im NODDY-Bericht (1982) beschrieben wurden).

TEIL 3

GRAFIK

Der MTX-Computer kann komplexe grafische Effekte realisieren. Sie werden die Grafik-Modi auf verschiedene Art und Weise steuern, wozu z.B. die Parameter Bildformat, Format und Farbe gehören, aber auch anspruchsvolle bewegte Grafikprogramme erstellen können.

Bislang haben Sie den Bildschirm nur für Textseiten genutzt und dabei ein Bildformat von 39 Spalten und 24 Zeilen vorgefunden. Im Grafikmodus für hochauflösende Grafik verfügen Sie über ein Bildschirmformat von 32 Spalten und 24 Zeilen. Dieses Bild kann auch Text aufnehmen, was umgekehrt für die Anzeige von Grafiken in einer Textseite nicht möglich ist.

Es ist wichtig zu wissen, daß, selbst wenn Sie bereits Erfahrungen mit Bildschirmgrafik auf anderen Computern gemacht haben, dieser Abschnitt äußerst sorgfältig studiert wird. MTX-Grafik wird mit nur wenigen interaktiven Befehlen generiert. Obwohl Grafikprogramme somit einfacher zu realisieren sind, diese Grafikbehandlung auch eine umfassende Kenntnis über die Wirkungsweise dieser Befehle voraussetzt.

Der Abschnitt über die MTX-Grafik besteht aus fünf Teilen: Textsteuerung, Grafiksteuerung, BASIC-Grafik-Befehle, erweiterte Grafik und Animation. Der nachfolgende Teil behandelt die Textkontrolle.

DIE STEUERUNG DES TEXTBILDSCHIRMS

Obwohl nun die Steuerbefehle für die Handhabung von Textseiten eingeführt werden, werden Sie bald feststellen, daß sie auch im Zusammenhang mit anderen Grafikbefehlen benutzt werden. Zunächst wird die unmittelbare Verwendung der Befehle erläutert, um dann im letzten Teil ihren Gebrauch zusammen mit anderen Befehlen an Beispielprogrammen zu zeigen.

CLS — Die CLS-Taste wird in vielen MTX-BASIC Anwendungen dazu genutzt, den Bildschirm für eine neue Arbeit zu löschen. CLS kann aber auch als Befehl innerhalb eines Programms für ähnliche Zwecke eingesetzt werden.

CSR x,y — Der Befehl CSR (Cursor) plaziert den Cursor auf die Bildschirmkoordinaten x,y.

Zur Verdeutlichung:

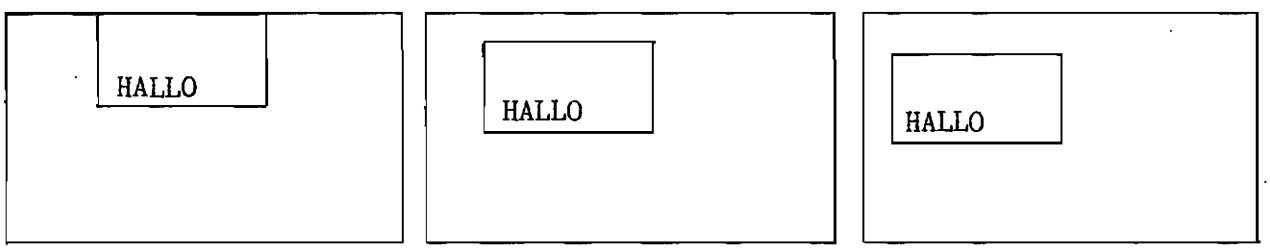
```
10 CLS:CSR 10,10:PRINT "HELLO"
```

Sobald Sie RUN eingeben, wird der Bildschirm gelöscht und HELLO erscheint ab der Koordinate 10,10 auf dem Schirm.

Setzen Sie die CLS- und CSR-Befehle ein, um Test an beliebigen Stellen des Schirms auszudrucken. Sie werden bald ein "Gefühl" dafür entwickeln, welche Koordinaten in etwa wo liegen, so daß bald komplexere Bildschirmsteuerungen programmiert werden können.

VS (Virtuelle Screens)

Alle Printbefehle erfolgen relativ zum benutzten Grafikmodus. Ganz gleich, ob Sie einen kleinen tragbaren Fernsehapparat oder einen großen Farbfernseher einsetzen - die Koordinaten 10,10 werden immer in etwa die Mitte des Bildschirms bezeichnen. Der MTX verfügt aber auch über eine Methode, um kleinere "Fenster" innerhalb des Gesamtbildschirms zu definieren. Diese nennen wir virtuelle Screens. Die bereits strapazierten Koordinaten 10,10 wären auch, relativ gesehen, in der Mitte eines solchen Screens, nur mit dem Unterschied, daß das Screen an beliebiger Stelle des Gesamtbildes plaziert sein kann. Hierzu folgende Beispiele:



MTX-BASIC kann 4 virtuelle Fenster definieren. Die schon vordefinierten Screens wären der Editierbereich, der den Namen VS 0 (Virtual Screen 0) benutzt und aus 4 Zeilen besteht, die nach außen wie eine Zeile wirken, das List-Screen (VS 1), bestehend aus 19 Zeilen, während das Fehlermeldungsfenster (VS 7) nur eine Zeile am unteren Bildrand beansprucht. Der Gesamtbildschirm heißt VS 5.

```
        VS 1  LIST-SCREEN    19 Zeilen
                                          VS 5
                                          (Gesamtbildschirm)

        VS 0 EDITOR           4 Zeilen
        VS 7 MELDE-SCREEN     1 Zeile
```

CRVS n,t,x,y,w,h,s

Dieser Befehl erzeugt ein Fenster nach Ihren Festlegungen. Die Angaben erfolgen als Parameter, die hinter den eigentlichen Befehl gesetzt werden.

n kennzeichnet die VS-Nummer (0-7) des Fensters, das benutzt werden soll*
t ist der Bildschirm-Modus (0 für Text, 1 für Grafik)
x ist eine Koordinate für die linke obere Ecke des Fensters
y ist die zugehörige senkrechte Koordinate für die linke obere Ecke des Fensters
w benennt die Fensterbreite in Zeichen
h benennt die Fensterhöhe in Zeilen
s kennzeichnet die Anzahl der Zeichen in einer Zeile entsprechend dem Parameter 't' (40 für Text, 32 für Grafik)

* Anmerkung: Verwenden Sie nicht die Fenster VS 0, 1, 5 oder 7, da diese wie oben angegeben von BASIC bereits belegt sind. VS 4 wird von BASIC als Grafikbildschirm benutzt. Sollten Sie dennoch eines dieser Fenster bestimmen, wird dessen Definition bei der Rückkehr zum BASIC-Ready umbenannt.

Ein einfaches Beispiel eines VS bestünde in der Definierung eines Quadrates in der Mitte des Bildschirms.

```
10 CRVS 2,0,20,10,10,10,40
20 VS 2
30 DSI
```

Zeile 10 definiert das Fenster, Zeile 20 wählt das neue Fenster an und Zeile 30 konfrontiert uns mit einem neuen Befehl DSI (Direct Screen Input). DSI ermöglicht die direkte Eingabe von Text über die Tastatur in das ausgewählte Fenster. Wenn Sie also nun einen Text eingeben, werden Sie sehen, welche Lage das Fenster einnimmt.

Durch gleichzeitiges Drücken der Tasten (CTRL) und (^) erscheint der Kursor auch im Fenster und kann nun wie sonst auch gesteuert werden. Sollten Sie von einem Fenster zum nächsten wechseln, bleibt der Cursor jeweils an seiner letzten Position.

Nutzen Sie diese Übung auch um andere Keyboardfunktionen zu testen.

Die (PAGE)-Taste bestimmt, ob wir uns im PAGE- oder im SCROLL-Modus befinden. Im PAGE-Modus springt der Cursor wieder zur HOME-Position der Seite, sobald eine Seite bis zur letzten Zeile voll ist. Beim SCROLL-Modus "rollt" der Text nach oben weg, sobald das Ende der letzten Zeile erreicht worden ist. Wechseln Sie zwischen den beiden Modi, um sich mit den verschiedenen Wirkungsweisen vertraut zu machen.

Experimentieren Sie mit (ESC)I und (ESC)J. Hier müssen Sie im Gegensatz zu den CTRL-Funktionen (ESC) und I oder J nacheinander eingegeben. Im Anhang finden Sie unter dem Befehl DSI eine Liste von Escape (ESC) und Kontroll (CTRL) Zeichen, die nützliche Dienste leisten können.

****

ÜBUNG 27 VIRTUELLE SCREENS

Bilden Sie 3 Fenster, um Ihren Namen, Anschrift und Geburtstag aufzunehmen. Stellen Sie sicher, daß alle Eintragungen Platz finden und keine Überlappung vorkommt.

****

Die beiden verbleibenden Befehle zur Textsteuerung (PAPER, INK) haben wir bereits in früheren Kapiteln behandelt.

Ehe wir nun zur Grafik übergehen, sollte nochmals darauf hingewiesen werden, daß der Modus nach Einschalten des Computers immer auf BASIC-TEXT geschaltet ist. Zur Grafikarbeit müssen wir demnach erst auf den Grafikmodus umschalten. Dazu zeigen wir Ihnen ein einfaches Programmbeispiel am Ende des Kapitels, in dem Sie sehen werden, daß schon ein vordefiniertes virtuelles Screen VS 4 existiert.

BASIC-GRAFIK

Die hier verwendeten Befehle finden sich in der Mehrzahl aller BASIC-Computer. Man benötigt sie, um Punkte, Linien, Bögen, Kreise etc. zu definieren.

PLOT x,y erzeugt einen Pixel (Bildpunkt) an der Stelle x,y.

LINE x,y,p,q erzeugt eine Gerade zwischen den beiden Koordinaten x,y, und p,q.

CIRCLE x,y,r erzeugt einen Kreis mit einem Radius r und dem Mittelpunkt x,y.

Hier ein kleines Programm zur Verdeutlichung:

```
10      VS 4
20      CLS
30      FOR I = 1 TO 191
40      PLOT I,I
50      NEXT I
60      CIRCLE 100,100,50
70      INPUT A$
80      IF A$="S" THEN STOP ELSE GOTO 10
```

Die Zeilen 70 und 80 sind insofern wichtig, als der Computer die Aufgabe sehr schnell ausführt. Anstelle dieser beiden Zeilen kann man auch schreiben

```
70        PAUSE   10000
```

Sie sehen den Effekt in einer 10 sec. Pause, ehe das Programm zurück ins BASIC geht.

Experimentieren Sie mit diesem Programm. Sie könnten z.B. auch folgendes einfügen:

```
65   LINE 10,20,150,170
68   LINE 35,150,170,55
```

Schreiben Sie sich Ihre eigenen Programme mit extensiver Grafik. Mit der Zeit werden die Befehle und die Bildaufführung so vertraut sein, daß Experimente nicht mehr erforderlich sein werden.

Sie werden sicherlich feststellen, daß der Kreis auf dem Bildschirm wie eine Ellipse aussieht. Dieser Effekt wird durch eine Stauchung in der Bildschirmgenerierung hervorgerufen, durch die die Pixel in einer Vertikalen enger zusammenstehen als die Pixel in einer Horizontalen. Die CIRCLE-Routine im ROM wurde dieser Stauchung deswegen nicht angepaßt, damit z.B. ein programmierter Bildschirmausdruck auf einem Drucker oder Plotter auch einem Kreis entspricht.

Nachfolgend ein Beispiel, wie ein Kreisprogramm aussehen könnte:

```
10 VS 4
20 CLS
30 LET RADIUS=60
40 LET STRFKR=1.4
50 LET X=100:LET Y=100
60 FOR I=0 TO 2*PI STEP .6/RADIUS
70 PLOT X+RADIUS*COS(I),Y+STRFKR*RADIUS*SIN(I)
80 NEXT I
90 GOTO 90
```

X und Y stellen die Koordinate dar, die Variable RADIUS spricht für sich und der Streckungsfaktor STRFKR ist für elliptische Effekte natürlich auch frei wählbar. Mit (BRK) kehren Sie ins BASIC zurück.

ERWEITERTE GRAFIK

BASIC-Grafik ermöglicht die Darstellung von z.B. Linien auf dem Bildschirm. In jüngster Zeit hat die Entwicklung von TURTLE-Grafik zu komplexeren Grafikformen (z.B. LOGO) geführt. Der MTX ist hiermit ausgestattet. Bevor wir jedoch zu Programmabläufen kommen, wollen wir die hierfür erforderlichen Befehle untersuchen.

Es kommen vier neue Befehle zur Anwendung: ANGLE, PHI, DRAW and ARC. Die ersten drei werden gemeinsam beschrieben. Sie werden eingesetzt, um die Richtung von Linien oder von Mustern zu bestimmen. Der Computer merkt sich jeweils die letzten Richtungswerte, die durch ANGLE und PHI gesetzt werden.

ANGLE (rad)

Der ANGLE-Befehl bestimmt die Richtung eines DRAW-Befehls von 0 Grad, was einer horizontalen Ebene entspricht, bis 360 Grad. Der Winkelwert wird allerdings nicht in Grad, sondern in Radiant angegeben. Folgende Umrechnungsformeln helfen Ihnen weiter:

Um den Radiant zu erhalten	Um Grad zu erhalten
R=2*PI*G/360	G=360*R/(2*PI)

wobei G einem Wert in Grad und R einem Wert in Radiant entspricht.

Ein gesamtes Programm verlangt daher folgende Daten:

1. Der Winkelwert von ANGLE in rad. Sie wissen, daß der Ausgangswert 0 die DRAW-Richtung nach rechts in einer Horizontalen bedeutet. Wenn rad-Werte hinzuaddiert werden, vergrößert sich der Winkel zwischen der Horizontalen und unserer DRAW-Richtung und zwar im entgegengesetzten Uhrzeigersinn. ANGLE kann daher auch genutzt werden, um eine Abbildung auf einer Achse zu drehen. Da Sie es wahrscheinlich nicht gewohnt sind, mit rad zu arbeiten, haben wir ein kleines Umrechnungsprogramm geschrieben:

```
10 INPUT "Bitte Winkel eingeben ";A
20 LET A=A*2*PI/360
30 PRINT "Bogenmaß= ";A
40 PRINT:PRINT:PRINT:PRINT
50 PRINT "Möchten Sie einen anderen Winkel?"
60 INPUT "J für Ja; N für Nein ";B$
70 IF B$="J" THEN GOTO 10 ELSE GOTO 80
80 STOP
```

Überarbeiten Sie das Programm so, daß auch die umgekehrte Berechnung möglich ist.

PHI (rad)

Der zweite Schritt ist der Gebrauch des PHI-Befehls. Sobald PHI im Programm angesprochen wird, erfolgt eine Aufaddierung des ANGLE-Wertes um den Wert PHI. Beispiel:

```
10        VS 4:CLS
20        ANGLE 0
30        FOR I=1 TO 10
40        PHI .1
50        PLOT 120,100
60        DRAW 50
70        PRINT ,,,I
80        NEXT I
90        GOTO 90
```

Sie sehen an diesem Programmablauf, daß nach jeder FOR-Schleife ein neuer PHI-Wert hinzuaddiert wird, so daß eine laufende Richtungsänderung der gezeichneten Linien von dem PLOT-Punkt aus erfolgt. Dies zeigt die einfachste Beziehung zwischen ANGLE und PHI. Weiter unten finden Sie ein anderes Beispiel, wo die Kombination ANGLE/PHI dynamischer wirken und Bögen und Spiralen erzeugen.

DRAW x

Dieser Befehl zeichnet eine Gerade der Länge x ab der aktuellen Plotposition in die Richtung, die durch ANGLE und PHI bestimmt sind.

Wir haben nachfolgend drei Programmbeispiele, um das Zusammenwirken der drei Befehle zu demonstrieren.

```
10 VS 4:CLS
20 ANGLE 0
30 PLOT 100,20
40 FOR I=1 TO 8
50 DRAW 70
60 PHI PI/4
70 PAUSE 10000
80 NEXT I
```

Durch Ändern der Zeilen 40 und 60 können Sie beliebige symmetrische geometrische Formen zeichnen. Die Anzahl von Seiten wird durch die FOR-Anweisung und die Größe des Winkels festgelegt. PI/4 z.B. ist ein Winkel von 45 Grad. Nach dieser Methode würde ein Quadrat durch eine Änderung in Zeile 40 (FOR I=1 TO 4) 4 Seiten erlangen und durch Zeile 60 PHI PI/2 richtig gezeichnet.

```
10 VS 4:CLS              Löschte vorherige Grafik
20 PLOT 100,100          Bestimmt die Startposition
30 ANGLE 0               Bestimmt die Ausgangsrichtung
40 FOR I = 0 TO 1 STEP .01
50 DRAW 7                Zeichnet eine Gerade der Länge 7
60 PHI I                 Addiert den Winkel I zur Richtung
70 NEXT I
```

Mit der Änderung des Wertes von I wird auch PHI in Zeile 60 verändert, so daß eine Spiralwirkung erzielt wird. Versuchen Sie das Programm durch Ändern der Werte von ANGLE und DRAW zu variieren. Tauschen wir den Wert von STEP in Zeile 40 gegen .001 aus, wird die Spirale entsprechend größer. Damit diese noch auf den Bildschirm paßt, müssen wir die Länge der Gerade in Zeile 50 auf weniger als 2.2 festlegen. Alternativ ließe sich ein kleinerer Wert in Zeile 20 mit PLOT bestimmen.

****

ÜBUNG 28 ANGLE, PHI UND DRAW

Ändern Sie Zeile 40 des obigen Programm so, daß die Spirale in der Mitte beginnend langsam nach außen wächst.

****

Nehmen Sie schrittweise folgende Modifizierungen vor und untersuchen Sie deren Auswirkungen:

Fügen Sie den PLOT-Befehl in Zeile 20 innerhalb der FOR-Schleife an Zeile 45 ein. Fügen Sie eine Testroutine in Zeile 65 ein, um das Anhalten des Programms zu verhindern.

```
65 IF I=1 THEN GOTO 45
```

Das zweite Programm arbeitet nach demselben Prinzip, um ein Muster zu erzeugen. Auch hier sollten Sie durch Ändern der Parameter deren Wirkungsweise untersuchen.

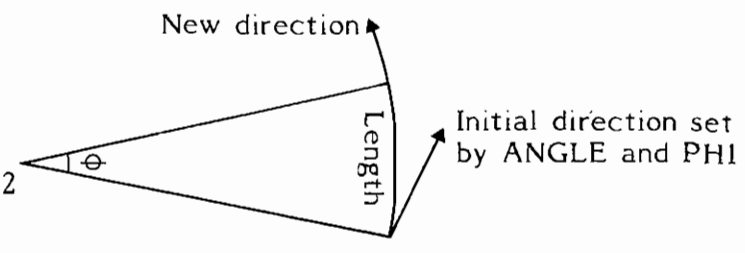
```
10 VS 4:CLS
20 PLOT 200,55
30 LET I=0
40 ANGLE 0
50 DRAW 1
60 LET I=I-.1
70 PHI I
80 GOTO 50
```

Die bislang erlernten Befehle ermöglichen bereits eine wirkungsvolle Grafiksteuerung. Die Kurven in den beiden obigen Programmen sind hilfreich zur Darstellung von Spiralen.

ARC x,theta

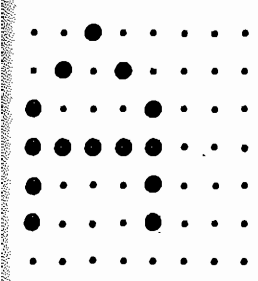
Dieser Befehl zeichnet einen Bogen mit der Länge x und dem Winkelwert theta. Das folgende Programm nutzt ARC um eine Reihe von Linien spiralartig von der PLOT-Position aus zu zeichnen. Durch Hinzufügen einer weiteren Schleife und durch Umkehrung des Effekts kann eine Blume mit 11 Blütenblättern gezeichnet werden.

```
10 VS 4:CLS
20 ANGLE 0
30 FOR I=1 TO 11
40 PLOT 120,100
50 ARC 100,2: PHI 2
60 PAUSE 1000
70 NEXT I
```



Die Steuerung der Grafik-Screens

Ehe wir nun mit weiteren Grafikarten fortfahren, ist es angezeigt, sich daran zu erinnern, wie die Grafik-Screens arbeiten. In allen Grafik-Modi bestehen sie aus kleinsten Bildpunkten (Pixel), die vom Computer an- oder ausgeschaltet werden können. Im normalen Textmodus ist der Hintergrund (paper) blau und der Text (ink) weiß. Drückt man nun eine Alphataste (z.B. A), werden die PIXEL, aus denen A zusammengesetzt wird, von blau auf weiß umgeschaltet, so daß 'A' auf dem Bildschirm erscheint. Der Buchstabe A besteht aus einem Pixelmuster in einer 8 x 8 Matrix.



Beachten Sie die Leerfelder unten und rechts vom Buchstaben, die dafür da sind, um dieses Zeichen vom nächsten deutlich zu trennen.

Ist Ihnen vielleicht aufgefallen, daß die Zeilen im Grafikmodus größer als die im Textmodus erscheinen? Dies rührt daher, daß die Zeichen im Textmodus ohne die beiden rechten Pixel-Spalten auskommen müssen, damit 40 Zeichen in eine Zeile passen. Wenn man den Bildschirm genau betrachtet, lassen sich oft die einzelnen Pixel erkennen, aus denen ein Buchstabe zusammengesetzt ist. Die Anzahl von Bildpunkten in einer Textseite (24 Zeilen x 40 Zeichen) kann wie folgt berechnet werden:

```
40 X 24 Zeichen (alphanumerisch)
          =   (40x6) X (24x6) Punkte
          =   240 X 192       Punkte
```

ASCII-Zeichen sind nichts anderes als eine Anzahl von international anerkannten Zeichenmustern, denen eindeutig eine Codenummer (aus dem ASCII-Code) zugeordnet sind.

Geben Sie folgendes Programm ein:

```
10      VS 4
20      CLS
30      PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
40      DSI
```

Dies ermöglicht die Eingabe von ASCII-Zeichen im Grafikmodus und Sie sollten den oben besprochenen Aufbau eines Zeichens mittels der Pixel erkennen können; in diesem Fall auch das komplette 8x8 Zeilenmuster. Die Berechnung der Gesamtbildpunkte sieht nun wie folgt aus:

```
32 x 24 Zeichen
          =   (32 x 8) x (24x8) Punkte
          =   256 x 192         Punkte
```

Im Grafikmodus haben wir daher einen Bildschirm, bestehend aus einer 256 x 192 Matrix, deren einzelne Punkte von einem Koordinatensystem mit dem Nullpunkt links unten erfasst werden können.

Im Textmodus bestimmten wir Vorder- und Hintergrundfarben mit den Befehlen INK and PAPER. Die einzigen Zeichen, die dargestellt wurden, waren ASCII-Zeichen. Der Grafikmodus ist in dieser Hinsicht wesentlich flexibler, so daß sowohl die Farbdefinition als auch die Zeichengebung weiterer Befehle bedürfen.

Der Gebrauch von PLOT, DRAW und ARC bewirkt nichts weiter als ein Einschalten bestimmter Pixel. Mit CLS werden alle Pixel wieder ausgeschaltet und damit mit der Farbe versehen, die durch PAPER bestimmt ist. Die Eingabe von Text oder der Gebrauch von PLOT schaltet die betreffenden Pixel in der gewählten INK-Farbe an. Das Wechseln von Farbe auf diese Weise erzeugt die Illusion von zu- und abschaltenden Bildpunkten.

Screensteuerungen sind im wesentlichen für Text und Grafik identisch. Anders sieht es aus mit der Farbgestaltung im Grafikmodus. Die Befehle COLOUR und ATTR werden zur Definition von Parametern eingesetzt. Sie ermöglichen eine bequemere Steuerung komplexer Farbgrafik.

COLOUR p,n (nur für Grafikscreens)

COLOUR ist der Befehl zur Bestimmung der einzusetzenden Farbe.

p ist ein Parameter
n ist die Farbe

Der Parameter bestimmt die Stellen des Bildschirms, für die die Farbe 'n' gelten soll. Die Werte für n sind wie die Parameter in den Befehlen PAPER und INK. Die Werte für p werden nun näher erklärt. Um den Gebrauch von Farbe im Grafikmodus zu verstehen, muß Ihnen immer bewußt sein, wie der Grafikbildschirm arbeitet, was ja oben erst behandelt wurde. Haben Sie z.B. gerade Text einzugeben, sind die folgenden PAPER- und INK-Parameter angebracht.

p = 0 = Für PAPER bei Text
p = 1 = Für INK bei Text

Die Pixel, die ihren Farbwert von PAPER zu INK beim Ausdruck eines Zeichens wechseln müssen, werden durch den ASCII-Code bestimmt.

Wenn jedoch Grafikbefehle zur Grafikdarstellung eingesetzt werden, wird - theoretisch - jedes Pixel individuell angesteuert. Im Grafikmodus besteht, wie schon bekannt, der Bildschirm aus 256 x 192 einzelnen Bildpunkten. Daher kann jeder dieser Punkte die gleiche Farbeigenschaft besitzen, wie die in einer Textseite. Um diese anzusprechen, nehmen wir 'non-print paper' und 'non-print ink' Befehle, also Anweisungen, die für die Einzelansteuerung der Pixel gelten. In diesem Fall würden also PLOT-Bildpunkte die Farbe annehmen, die durch die Parameter 2 und 3 gesetzt werden.

p = 2 = non-print (plot) PAPER (Für PAPER bei Grafikbefehlen)
p = 3 = non-print(plot) INK (Für INK bei Grafikbefehlen)

Der verbleibende Parameter (p=4) betrifft den restlichen Bildschirm. Die Definition dieses Wertes bestimmt die Farbe des Bildschirmrands.

p = 4 = Bildrandfarbe

Testen Sie folgendes Programm und variieren Sie die Farbparameter in den Zeilen 20,30 und 40 mit Zuhilfenahme der Farbtafel aus Kapitel 1.

```
10        VS 4:CLS
20        COLOUR 2,5
30        COLOUR 3,3
40        COLOUR 4,6
50        ANGLE 0
70        PLOT 120,100
80        DRAW 50
90        PHI .2
100       GOTO 70
```

ATTR p,Zustand (nur Grafikscreen)

Dieser zweite Grafikbefehl ermöglicht eine weitere Manipulation der Bildpunkte, die vom COLOUR-Befehl bestimmt wurden. ATTR ermöglicht die Zuordnung neuer Eigenschaften zu den durch PRINT oder PLOT angesteuerten Bildpunkten. Hier ein kleines Demoprogramm, das Ihnen zeigt, wie dieser Befehl bei Texten arbeitet.

```
10 VS 4:CLS
20 INPUT "ATTR P N ?    ";P,N
30 ATTR P,N
40 DSI
50 GOTO 20
```

Beim Ablaufen dieses Programms müssen Sie Werte für p und n eingeben. 'n' schaltet einfach die Funktion des Parameters 'p' bei dem Befehl ATTR an oder aus, wobei 1 an und 0 aus bedeutet.

Tippen Sie einige Zeichen ein und testen Sie die Auswirkung für verschiedene Attribute. Unterbrechen Sie die Programmschleife mittels (RET), um ein neues ATTRibut zu wählen.

p =0 ; inverse print ATTR (Nur für Texte)

Setzen Sie den Wert 'n' auf 1 (also Eingabe von 0,1), so werden Sie feststellen, daß die Zeichen auf dem Bildschirm invertiert dargestellt werden, d.h. daü die Farben für PAPER und INK ausgetauscht sind. Schreiben Sie vom Ausgangspunkt (Taste (HOME)) erneut Zeichen auf den Schirm, ergeben die Buchstabenüberlappungen an den Stellen des Textes interessante Bildmuster.

Ist das ATTRibut zugeschaltet, kann es mit der Eingabe p,0 abgeschaltet werden (p=1,2,3 oder 4). Die ATTR-"Schalter-Stellung" wird also ähnlich wie die PAGE-Taste gehandhabt, um von einem Modus auf den anderen umzuschalten.

Schalten Sie das Attribut 0 aus. Geben Sie dazu 0,0 als p,n Werte ein.

p = 1 ; overprint ATTR (Nur für Texte)

Wenn Sie dieses Attribut nun durch die Eingabe 1,1 einschalten und zehnmal 'D', zehnmal die Leertaste usw. betätigen, den Kursor zurück zur HOME-Stellung bringen und die D-Taste gedrückt halten, werden die D-Zeichen durch Leerfelder und umgekehrt ersetzt. Das Zuschalten dieses Attributs hat den Effekt, daß Punkte, die über andere Punkte gesetzt werden, immer die zuvor gezeichneten auslöschen. Daher kann ein D, daß über ein anderes D gedruckt wird, dieses löschen.

Die ATTR-Befehle können auch in Kombination eingesetzt werden, um besondere Effekte zu erzeugen. Wenn Sie im Demoprogramm 0,1 eingeben, wird dies zur Folge haben, daß 1,1 eingeschaltet bleibt, nun aber ein Eingeben von Zeichen mit dem zusätzlichen Effekt der ausgetauschten PAPER-/ INK-Farben aus der 0,1 Kombination geschieht.

Versuchen Sie nun folgende ATTR-Effekte ein- und auszuschalten und verschiedene Befehle zu kombinieren.

p = 2 ; unplot ATTR (Nur für Grafik)

Die Aktivierung dieses Befehls bewirkt, daß anstelle eines Setzens (Plottens) das Löschen eines bereits gesetzten Bildpunkts erfolgt (Falls kein Punkt gesetzt war, geschieht natürlich nichts). Mit anderen Worten: Wir haben einen PLOT-Befehl in der Farbe des Bildschirmhintergrunds (PAPER).

p = 3 ; over plot ATTR (Nur für Grafik)

Wird p = 3 gesetzt, erfolgt bei

a) einem PLOT-Befehl das Setzen eines Pixels bei einer noch unbeschriebenen Bildschirmstelle. Da, wo bereits Bildpunkte vorhanden sind, werden diese gelöscht.

b) CLS und anderen vergleichbaren Funktionen lediglich ein eventuell programmiertes Wechseln der Farben, während der Text oder die Grafik unverändert bleibt.

Werden unplot und over plot gleichzeitig aktiviert, passiert während des Plottens garnichts. Dies kann nützlich sein, um die PLOT-Position für z.B. DRAW zu wechseln, ohne den Bildschirm zu ändern. Sie können hiermit auch PLOT SPRITE (siehe unten) über den Bildschirm steuern.

ANIMATION

Bewegte Grafik-Animation läßt sich mit dem MTX mit Hilfe von SPRITES realisieren.

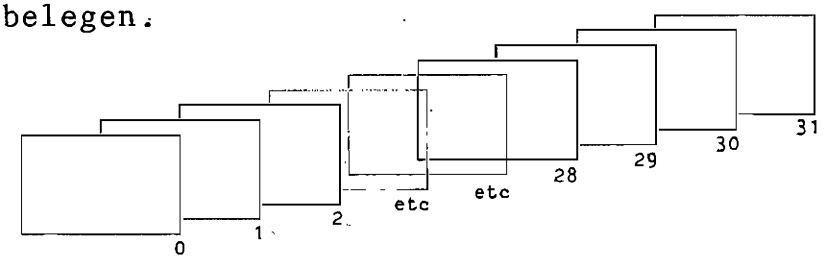
Sie können sich einen Sprite als eine kleine Maltafel vorstellen, die mittels näher zu erläuternder Befehle über ein schon erstelltes Bildschirmscreen bewegt werden kann, ohne das Screen zu zerstören.

Testen Sie folgendes Programm, das einen Pfeil von links nach rechts über den Schirm 'fliegen' läßt:

```
10 VS 4:CLS
20 CTLSPR 0,1
30 CTLSPR 2,1
40 CTLSPR 3,1
50 CTLSPR 5,1
60 CTLSPR 6,1
70 GENPAT 3,1,24,4,2,255,255,2,4,24
80 SPRITE 1,1,1,100,10,0,1
90 GOTO 90
```

Sprites sind ähnlich wie Zeichen aufgebaut und bestehen entweder aus einer 8 x 8 oder 16 x 16 Pixelmatrix, können aber im Gegensatz zu Zeichen nur eine Farbe haben.

Es gibt 32 Sprites mit der Benennung 0 bis 31, die wie folgt den Bildschirm belegen.



Jede dieser Bildebenen bietet ein zweidimensionales Bewegungsfeld für einen Sprite. Dies bietet den Vorteil, daß sowohl ein räumlicher Effekt durch 'Übereinanderlegen' dieser Ebenen erzielt werden kann, als auch die Möglichkeit, Richtung und Geschwindigkeit zu bestimmen.

Da auch die Farbgebung der Sprites unabhängig ist, können mehrere Sprites zusammengefaßt mehrfarbige Objekte bilden. Beachten Sie jedoch, daß mehr als 4 Sprites in einer horizontalen Linie zu unberechenbarer Wirkung führen kann.

Hinter den Spriteebenen befindet sich noch die PAPER- und INK-Ebene. Wenn sich die Sprites bewegen, bleibt der Hintergrund statisch. Damit Sprites, wenn sie auf dem Bildschirm erscheinen oder diesen verlassen, realistisch aussehen, wurde die Spriteebene größer gewählt als die eigentliche Bildfläche. Sprites können somit - fast wie Schauspieler im Theater - rechts und links hinter dem 'Vorhang' warten, ehe sie erscheinen. Ebenso ist es möglich, Sprites endlos kreisen zu lassen, so daß ein Sprite beim Verlassen des rechten Bildschirms plötzlich auf gleicher Höhe wieder links auftaucht.

Ändern Sie Zeile 80 des letzten Programms wie folgt:

```
80 SPRITE 1,1,1,100,120,0,1
```

Sie werden sehen, daß der Pfeil nun eine Kreisbahn der beschriebenen Art verfolgt, d.h., daß nach einiger Zeit der Pfeil wieder von links nach rechts fährt.

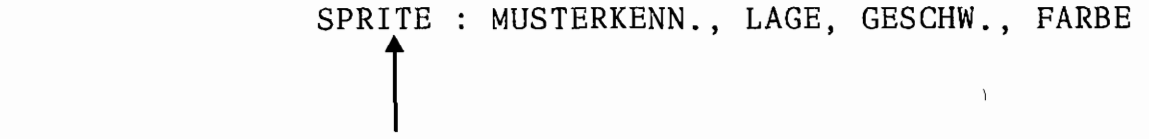
Die für diese komplexe Grafik erforderlichen Befehle beeinflussen sich gegenseitig. Das heißt, jeder Steuerbefehl wirkt sich auf andere Steuerbefehle aus. Wir haben bereits festgestellt, wie die Pixel im Text- und Grafikmodus manipuliert werden können, um interessante Muster zu erzeugen. In ähnlicher Weise ergibt die Vielzahl der Sprites zusammengenommen ein komplexes Gesamtbild.

Bevor nun die Spritebefehle einzeln behandelt werden, geben wir Ihnen eine Übersicht aller Befehle, deren Hierarchie und Wechselwirkung.

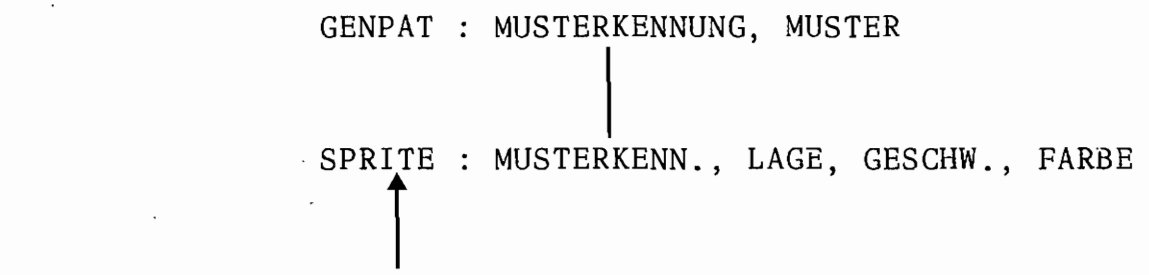
Grundlage der Animation ist die Bewegung. Daher lautet der erste Befehl, den wir untersuchen wollen, MVSPR (movesprite). Bestandteil aller Spritebefehle ist ein Bezug auf die Spritenummer (0-31). MVSPR bestimmt außerdem, wie ein Sprite bewegt werden soll und in welche Richtung.

```
MVSPR : BEWEGUNG, SPRITENR., RICHTUNG
```

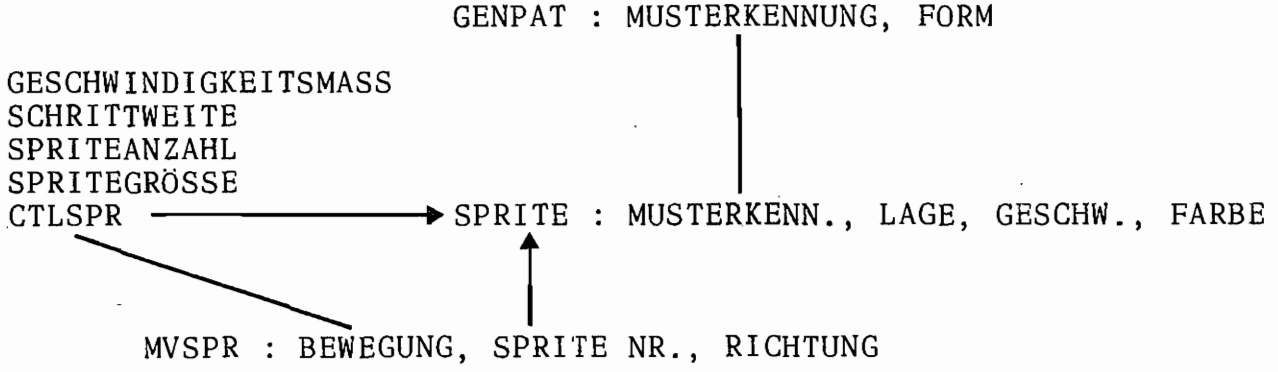
Obwohl obiger Befehl bereits die Spritenummer definiert und die Bewegungsrichtung bestimmt hat, würden wir nach diesem Befehl nichts sehen, da das Sprite noch ohne Form und Farbe ist. Wir verwenden daher den SPRITE-Befehl, um ein Sprite zu definieren.



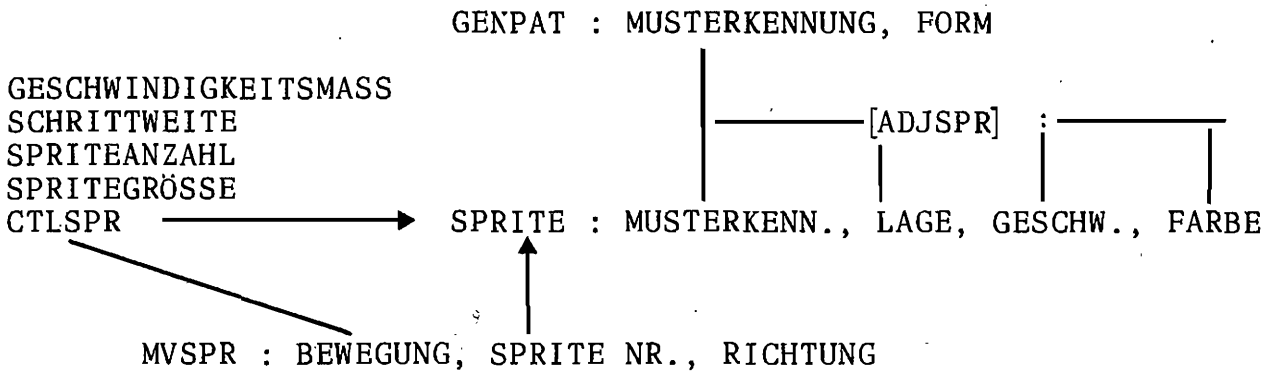
Der SPRITE-Befehl bestimmt die Lage, Geschwindigkeit, Farbe und die Muster- (Sprite-) Nummer eines Sprites. Die Musterkennung wählt die Form des Sprites an, die bereits vom GENPAT-Befehl bestimmt wurde.



Nachdem das Sprite nun auf diese Weise definiert wurde, muß es auch gesteuert werden können. Jedesmal, wenn der MVSPR-Befehl aktiviert wird, erhält das Sprite den Befehl, einen Schritt in angegebener Richtung zu tun. Die Schrittweite wird vom CTLSPR-Befehl definiert, der ferner auch andere Parameter wie Spritegröße, Geschwindigkeitsmaß und die Anzahl der anzusprechenden Sprites bestimmt.



Beachten Sie, wie der CTLSPR-Befehl alle Sprites anspricht, während sich der SPRITE-Befehl nur auf einen Sprite bezieht. Wollen wir die Farbe, Geschwindigkeit oder Lage eines Sprite verändern, muß dies nicht mit dem SPRITE-Befehl geschehen, sondern kann mit dem Befehl ADJSPR getätigt werden. Dies hat den Vorteil, daß nur ein einzelner Parameter durch Angabe der Spritenummer, des Parameters und des neuen Wertes schnell und unproblematisch verändert werden braucht.



Wie bereits erwähnt, gibt es zwei Spritegrößen: Eine mit 8x8 und eine zweite mit 16x16 Pixel. Das kleinere Sprite benötigt nur eine GENPAT-Anweisung, während das größere deren 4 erfordert. Alle z.Z. verwendeten bzw. anzusprechende Sprites müssen gleich groß sein, da die Größe vom CTLSPR-Befehl bestimmt sein muß. Nachdem die Größe bestimmt worden ist, kann das Sprite z.B. auch mittels des MAGNIFY-Parameter im CTLSPR-Befehl in seinen Ausmaßen verdoppelt werden.

CTLSPR definiert auch die Maßeinheit für Entfernungen. Diese Einheit bestimmt die Anzahl der Pixel, die während eines MVSPR-Befehls in der gegebenen Richtung übersprungen werden. CTLSPR kann aber auch einzelne Sprites bewegen. Diese Sprites werden von GENPAT und SPRITE wie bisher definiert, müssen nun aber eine Geschwindigkeit erhalten. Die Geschwindigkeit eines auf diese Weise gesteuerten Sprites wird vom Geschwindigkeitsparameter des CTLSPR-Befehls bestimmt. Dieser CTLSPR-Parameter bestimmt die Schrittgeschwindigkeit, d.h. die Anzahl der übersprungenen Pixel pro Sekunde. Die tatsächliche Geschwindigkeit bestimmt dann der entsprechende Parameter des SPRITE-Befehls (=Schritte/Sek.). Hätten wir z.B. eine Schrittweite von 20 Pixel/Sek. im CTLSPR-Befehl und eine Schrittgeschwindigkeit von 5 Schritten/Sek. im SPRITE-Befehl, ergibt dies eine tatsächliche Geschwindigkeit von

20 Pixel/Sek. x 5 Schritte/Sek. = 100 Pixel/Sek.

Als nächstes wollen wir ein neues Spriteprogramm entwickeln. Sie werden anfangs die obigen Diagramme hilfreich finden, um eine Orientierung zwischen all den neuen Befehlen zu haben. Mit jedem Befehl werden Sie auch alle hierzu möglichen Parameter erklärt bekommen.

MUSTERPROGRAMM ANIMATION

Denken Sie daran, zunächst immer einen Grafikmodus (VS) anzuwählen.

```
10 VS 4:CLS
20 CTLSPR 2,1
30 GENPAT 3,0,255,129,129,129,129,129,129,255
40 SPRITE 1,0,128,96,0,0,1
50 CTLSPR 1,1
60 LET Y=ASC(INKEY$)-48
70 IF Y>8 OR Y<1 THEN GOTO 60
80 MVSPR 9,1,Y
90 GOTO 60
```

Zeile 20 (CTLSPR) ist dazu da, dem Computer die Anzahl der vorkommenden Sprites mitzuteilen. Wenn Sie sich einige Seiten weiter hinten die Befehlserklärung zu Befehl 1 ansehen, werden Sie feststellen, daß der CTLSPR-Befehl wie folgt arbeitet:

Die Funktion von x variiert für jeden Parameter p wie in der Tabelle angegeben. Am Beispiel von Zeile 20 besagt Parameter 2, daß der folgende Platzhalter x die Anzahl der Sprites angibt und daß 1 als Wert für x für diesen Fall bestimmt, daß nur 1 Sprite vorkommt. Dieser Parameter wird also ähnlich wie eine DIM-Anweisung dazu benutzt, den Speicherbedarf festzulegen.

Zeile 30 bestimmt das Spritemuster. Studieren Sie die Einzelheiten zum Befehl 2. Sie werden erkennen, daß GENPAT 3,... (Parameter P=3) festlegt, daß die Matrix eines 8 x 8 Pixelfeldes mit den nachfolgenden Angaben definiert werden soll.

```
30 GENPAT 3,0,...
```

Die 0 kennzeichnet die Musterkennung. Jedes Sprite, dem die Musterkennung 0 zugewiesen wird, wird in dieser Musterebene plaziert.

```
30 GENPAT 3,0,255,129,129,129,129,129,129,255
              r1  r2  r3  r4  r5  r6  r7  r8
```

Die verbleibenden Zahlen der Zeile 30 (r1-r8) bestimmen das Muster der 8x8 Matrix. Jede Pixel-Zeile des Sprites wird durch eine einzige Zahl eindeutig bestimmt.

Um ein Spritemuster zu entwerfen, ist es angebracht, ein 8x8 Matrixfeld zu zeichnen (Karopapier o.ä.) und wie folgt vorzugehen:

```
128 64 32 16   8  4  2  1

  0  0  0  1   1  0  0  0        r1=24
  0  0  0  0   0  1  0  0        r2=4
  0  0  0  0   0  0  1  0        r3=2
  1  1  1  1   1  1  1  1        r4=255
  1  1  1  1   1  1  1  1        r5=255
  0  0  0  0   0  0  1  0        r6=2
  0  0  0  0   0  1  0  0        r7=4
  0  0  0  1   1  0  0  0        r8=24
```

Wie unschwer zu erkennen ist, findet sich hier unser Pfeil des ersten Beispiels wieder ein. Die Zahl eines GENPAT-Befehls ergeben sich, wenn die Kopfzahlen jeder Spalte, die eine 1 enthalten, Zeile für Zeile (r1 bis r8) addiert werden.

Damit Sie noch mehr Erfahrung im Umgang mit GENPAT-Mustern gewinnen, haben wir folgendes Programm parat. Da die erste Programmzeile 100 ist, kann das zuvor eingegebene Programm im Speicher bleiben. Nach RUN wird der Kursor ans obere Ende eines Fensters plaziert werden und eine 4 mit einem Fragezeichen erscheint. Die 4 weißt darauf hin, daß ein GENPAT 4 Befehl eingegeben wird und das Fragezeichen verlangt nach der Eingabe von acht Zahlen zwischen 0 und 255. Jede Eingabe muß durch ein Komma getrennt werden.

Versuchen Sie als erstes diese Zahlenfolge:

```
255,129,129,129,129,129,129,255
```

Nach (RET) erscheint das Sprite am unteren Bildrand, wobei in der linken oberen Ecke ein Quadrat auftaucht. Im Fenster kommt dann die Frage "Zufrieden?". Sie haben somit die Möglichkeit, wenn Sie nicht "j" drücken, das Muster noch einmal einzugeben. Sollten Sie zufrieden sein, müssen Sie nun "j" drücken und mit GENPAT 5 weitermachen. "n" löscht das Muster und GENPAT 4 muß neu eingegeben werden. Experimentieren Sie mit verschiedenen Zahlen für die anderen Zeilen. Sobald vier Viertel 4, 5, 6 und 7 eingegeben sind, ist das komplette Sprite fertiggestellt.

```
100 CRVS 4,1,1,3,30,10,0
110 VS 4
120 CLS
130 CTLSPR 2,32
140 CTLSPR 5,0
150 CTLSPR 6,3
160 FOR I=4 TO 7 STEP 1
170 PRINT I
180 INPUT A,B,C,D,E,F,G,H
190 GENPAT I,O,A,B,C,D,E,F,G,H
200 SPRITE 1,0,100,30,0,0,1
210 INPUT "Zufrieden? ";A$
220 IF A$="j" THEN GOTO 230 ELSE GOTO 170
230 NEXT I
240 GOTO 160
```

Zurück zur Programmerklärung. Zeile 40 bestimmt die Parameter zur Spritekontrolle.

```
40 SPRITE 1,0,128,96,0,0,1
```

Die erste Ziffer kennzeichnet die Spritenummer, so daß der Computer weiß, welche Bildebene angesteuert werden muß. Die zweite Ziffer bezieht sich auf die GENPAT-Kennung und bestimmt damit das abzubildende Muster. Die Zahl 128 bestimmt den Mittelpunkt der Matrix auf der x-Achse, während 96 die Koordinate der y-Achse darstellt, so daß hiermit eine eindeutige Lagebestimmung erfolgt. Die Koordinaten werden wie bei PLOT mit 0,0 links unten am Bildschirm beginnend definiert.

Die fünften und sechsten Ziffern bestimmen die Geschwindigkeit des Sprites, wobei die fünfte die Geschwindigkeit entlang der x-Achse, die sechste die entsprechende Geschwindigkeit entlang der y-Achse festlegt. Da das Sprite in diesem Beispiel keine unabhängige Geschwindigkeit haben soll, sind beide Werte auf Null gesetzt. Die letzte Ziffer der Parameterkette bestimmt den Farbwert; in diesem Fall 1 (Schwarz).

Zeile 50 des Programms beinhaltet den Befehl über die Bewegung des Sprites. Die Parameter 1,1 sagen aus, daß ein Sprite bei 'Aufforderung' durch z.B. MVSPR eine Pixelweite bewegt werden soll.

```
50 CTLSPR 1,1
```

Zeilen 60 und 70 ermöglichen eine gezielte Steuerung des Sprites mittels Tastendruck. Jeder Richtung ist eine gesonderte Taste zugeordnet (der Subtrahend 48 soll den ASCII-Zeichenvorrat auf 1-8 einschränken). Wird eine andere Taste als die erlaubten gedrückt, sorgt eine Abfrage in Zeile 70 dafür, daß das Programm zurück zur Zeile 60 geführt wird.

```
60 LET Y=ASC(INKEY$)-48
70 IF Y>8 OR Y<1 THEN GOTO 60
```

Zeile 80 verwendet den Befehl MVSPR (Befehl 4), damit der Computer entsprechend den Steuerbefehlen per Tasteneingabe reagiert. Der CTLSPR-Befehl definierte die Bewegung. Der MVSPR-Befehl sorgt nun für die Ausführung dieser Bewegung und setzt die Spritebewegung auf eine der 8 Richtungen fest . MVSPR nutzt dazu ein Bitmuster, dem GENPAT nicht unähnlich, um die spezifischen Bewegungsparameter zu definieren. Indem verschiedene Parameter addiert werden, kann eine Anzahl von Anweisungen mit einer einzigen Zahl bestimmt sein.

```
80 MVSPR 9,1,Y
```

Die Schlußzeile 90 gibt die Steuerung zurück an die Eingabezeile 60.

```
90 GOTO 60
```

Es gibt aber zwei Befehle, die hier nicht angewandt wurden: ADJSPR und VIEW.

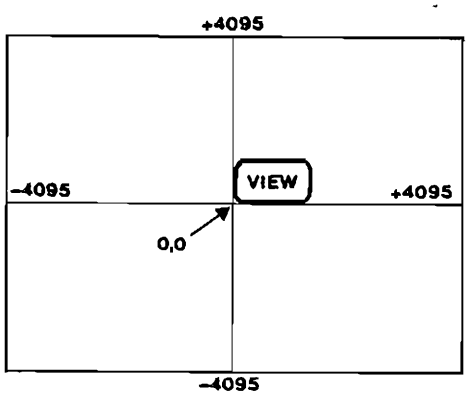
ADJSPR dient dazu, einen beliebigen Parameter der zuvor festgelegten SPRITE-Parameter zu ändern. Wollten wir z.B. die Farbe des Sprites 3 nachträglich von 1 auf 5 ändern, würde es ausreichen

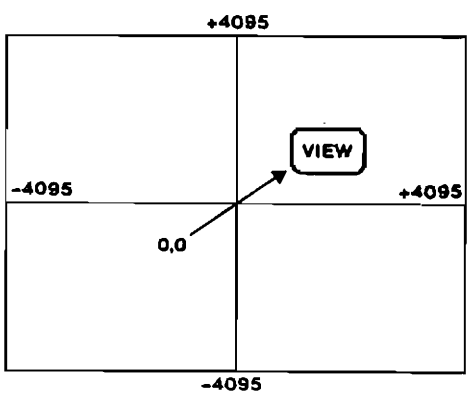
```
ADJSPR 1,3,5
```

einzugeben (Siehe Befehl 5). Dies hat den Vorteil der schnelleren Eingabe, weil nur ein Parameter geändert werden muß.

Der VIEW-Befehl hat den Effekt eines Fensterblicks auf die Spriteebenen (8192 x 8192 Pixel), wobei der Fensterrahmen vom Bildschirm bestimmt ist (256 x 192 Pixel).

In seiner Ausgangslage liegt das Fenster in etwa über der Mitte der Spriteebene, wobei die Koordinaten 0,0 der Grafikebene mit den gleichen Koordinaten der Spriteebene übereinstimmen.





Während der MVSPR-Befehl Sprites relativ zur Grafikebene bewegt, kann der VIEW-Befehl die Grafikebene relativ zu allen Sprites bewegen. Dies bedeutet, daß komplizierte Spritemuster bestehend aus vielen verschiedenen Sprites mit geringem Auswand bewegt werden können. Ferner können Sprites so lange im nicht sichtbaren Bereich der Spriteebene versteckt werden, bis das Fenster über diesen Bereich gelegt wird.

Wir haben versucht, Ihnen eine grobe Einsicht in die Arbeitsweise der MTX-Grafikbefehle zu geben. Experte werden Sie aber erst nach längerem eigenem experimentieren. Unten stehende Befehlszusammenstellung enthält alle Befehle und ihre Parameter. Was daraus nicht erkennbar wird, ist die gegenseitige Beeinflussung zweier Befehle. Dies müssen Sie selbst mit Hilfe der vorangegangenen Relationsdiagramme herausfinden. Zum Üben sind noch zwei Programme aufgeführt.

BEISPIELPROGRAMM

```
10 CTLSPR 0,6
20 CTLSPR 2,10
30 CTLSPR 6,2
40 GENPAT 4,0,1,0,1,2,3,15,1,3:GENPAT 5,0,2,2,2,6,6,0,0,0:
GENPAT 6,0,64,128,192,160,224,248,192,224:GENPAT 7,0,32,32
,32,48,48,0,0,0
50 SPRITE 1,0,0,0,0,0,6
60 CTLSPR 4,1
70 CRVS 6,1,0,0,32,24,0:PAPER 15:COLOUR 4,6:INK 1:CLS
80 ATTR 3,0:ATTR 2,0
90 PLOT 80,80:ANGLE 0
100 FOR I=1 TO 11
110 ARC 100,2:PHI 2
120 NEXT I
```

Zeile 10 regelt die Geschwindigkeit des Sprites. Wenn Sie diese gemäß BEFEHL 1 ändern, läßt sich die Geschwindigkeit des Sprites beliebig erhöhen oder verlangsamen.

Durch ändern von Zeile 30 kann auch eine Vergrößerung des Sprites bewirkt werden:

```
30 CTLSPR 6,3
```

Sie können das Programm auch durch Einfügen der Zeile 200 anstelle von Zeile 80 verlängern:

```
200 ATTR 3,1:ATTR 2,1
```

Und gefolgt von:

```
210 CTLSPR 5,3
220 CTLSPR 0,1
230 LET S=25
240 SPRITE 3,0,100,130,S,0,2
250 FOR W=1 TO 20
260 LET Y=0
270 FOR Z=1 TO 8
280 LET Y=Y+1
290 FOR X=1 TO 25
300 NEXT
310 LET D=Y-8
320 MVSPR 12,3,-D
330 ADJSPR 1,3,-D+2
340 NEXT
350 LET S=S+5
360 ADJSPR 4,3,S
370 NEXT
```

BEISPIEL

```
10 VS 4:CLS
20 PAPER 1:INK 7:CLS:ATTR 3,1
30 FOR X=0 TO 255
40 LINE X,191,255-X,0
50 NEXT
60 FOR Y=1 TO 190
70 LINE 0,Y,255,191-Y
80 NEXT
90 FOR K=2 TO 94 STEP 4
100 CIRCLE 128,96,K
110 NEXT
120 GOTO 30
```

(Das Programm läuft, bis die BRK-Taste gedrückt wird.)

BEFEHL 1: CTLSPR p,x

p = Parameter und kann eine der folgenden sechs Funktionen bestimmen, wonach sich der Wert von x richtet:

0 Geschwindigkeit
x: 1 bis 255 und 0 (1=Höchstgeschwindigkeit)

1 Entfernung
Bewirkt die Bewegung des Sprite um x-Pixel, wenn angesprochen.

2 Anzahl der Sprites
x = 1 bis 31 (Die Anzahl muß mindestens 1 sein)

3 Anzahl der kreisenden Sprites, d.h., die Sprites, die nach einiger Zeit wieder am entgegengesetzten Bildschirmrand auftauchen. (Diese Anzahl darf die Gesamtzahl der Sprites nicht überschreiten).

4 PLOT SPRITE
Hiermit kann ein PLOT SPRITE bestimmt werden, das immer dann auftaucht, wenn ein Punkt geplottet wird. Dieses Sprite wird jedem Punkt oder jeder Linie, die über den Bildschirm mit gezogen wird, folgen. Das PLOT SPRITE kann ein beliebiges der 32 in der üblichen Art definierten Sprites sein (x=Spritenummer).

5 Anzahl der beweglichen Sprites
Bestimmt die Anzahl jener Sprites, die ihre eigene Bewegung entsprechend den Parametern von SPRITE und ADJSPR durchführen können (x=0 bis 32).

6 Größe und Vergrößerung
x=0 Größe 8X8 Vergr. 1
x=1 Größe 8X8 Vergr. 2
x=2 Größe 16X16 Vergr. 1
x=3 Größe 16X16 Vergr. 2

BEFEHL 2: GENPAT p,n,d1,d2,d3,d4,d5,d6,d7,d8

Der GENPAT-Befehl dient dazu, alle Muster, die von BASIC für die Generierung von Zeichen und SPRITES gebraucht werden, zur Verfügung zu stellen. Es gibt 5 Modi.

1. Um ein ASCII-Zeichen neu zu definieren (CODE 32 - 127)
2. Um ein nicht-ASCII-Zeichen zu definieren (CODE 129 - 154)
3. Um eine Farbe für jede Zeile eines Zeichens zu bestimmen. Dies bezieht sich nur auf anwenderbestimmte Zeichen mit dem CODE 147 - 154.
4. Um ein 8 x 8 Pixel Spritemuster zu bestimmen.
5. Um die vier Quadrate eines 16 x 16 Pixel Spritemuster zu bestimmen.

Anwenderbestimmbare Zeichen haben die Codes 129 - 154.

Modus 1 erlaubt die Neudefinierung eines Standard-ASCII-Zeichens. Beachten Sie bitte, daß ASCII-Zeichen am häufigsten von dem Computer benutzt werden.

Modus 2 gestattet die Aufstellung eines eigenen Zeichensatzes, ohne die Standard ASCII-Zeichen zu zerstören.

Modus 3 gestattet eine weitere Bestimmung einiger der Zeichen aus Modus 2 insofern, als INK- und PAPER-Farben für jede der acht Pixel-Zeilen eines Zeichens definiert werden können.

Die Werte für INK- und PAPER-Farben werden bereits in Kap. 1 angegeben. In diesem Fall bestimmen wir aber beide Farben mit einem Parameter. d1 - d8 bestimmen allein die Farbkombination:

```
bit    0  1  2  3:  4  5  6  7
Wert   1  2  4  8: 16 32 64 128
         PAPER   :     INK
```

Parameter = 16 * INK + PAPER

z.B. rot (INK) auf blau (PAPER)

= ROT : BLAU

= ROT * 16 + PAPER

= 9 * 16 + 4

= 148

Es folgt nun eine Tabelle, die anzeigt, welche Werte für p und n bei welchem Modus benutzt werden müssen.

MODUS	P	N
1	0	ASCII-Code (32 - 127)
2	1	Anwenderdefiniert (Code 129 - 154)
3	2	Anwenderdefiniert für Farbe (Code 147 - 154)
4	3	Matrixnummer 8 x 8 Sprite
5	4	Matrixnummer 16 x 16 NW-Viertel
	5	Matrixnummer 16 x 16 SW-Viertel
	6	Matrixnummer 16 x 16 NO-Viertel
	7	Matrixnummer 16 x 16 SO-Viertel

Anmerkung: Wenn Sie bei einem Zeichen das Muster und die Farbe selbstdefinieren wollen (Code 147 - 154), müssen Sie zwei GENPAT-Befehle mit p = 1 und 2 und demselben Code programmieren. Die Farben werden allerdings erst in einem Grafikscreen wirksam.

BEFEHL 3: SPRITE

SPRITE (= definiere Sprite)

SPRITE n,pat,xp,yp,xs,ys,col

n bezeichnet die Spritenummern 0 - 31
pat bezieht sich auf eine Musterkennung 0 - 127 (Größe 0)
0 - 31 (Größe 1)

xp kennzeichnet die Mittelpunktslage x des Sprites
yp kennzeichnet die Mittelpunktslage y des Sprites
(Beide Angaben bewegen sich innerhalb der Werte -4095 bis 4095)
0,0 kennzeichnet eine Lage am linken unteren Bildrand, was der gleichen Koordinierung wie bei PLOT entspricht.

Bem.: SPRITE-Koordinaten sind absolut und orientieren sich nicht an der Fenster-Dimensionierung (es wird immer ein Bildbereich von 32 x 24 Zeichen angenommen).

xs kennzeichnet die Geschwindigkeit auf der x-Achse (-128 bis 127), wobei eine Geschwindigkeitseinheit des Sprite 1/8 Pixel je Zyklus gemäß CTCSPR 0 entspricht.

ys kennzeichnet die Geschwindigkeit auf der y-Achse im Bereich -128 bis 127.

col kennzeichnet die Farbe (0 - 15)

BEFEHL 4 MVSPR p,n,d

MVSPR ist ein Vielzweckbefehl mit 4 deutlich bestimmbaren Funktionen:

p bedeutet:

1 Bewegung
2 Musterkennung
4 Richtungsänderung
8 PLOT im Mittelpunkt

Diese Funktionen sind kombinierbar, um komplexere Bewegungsabläufe mit einem einzigen Befehl steuern zu können. Die Art der Aktion wird nach obiger Tabelle angewählt: Kombinationen verlangen einfach die Addition der jeweiligen Kennziffer.

BEISPIELE:

		Beisp. 1	Beisp. 2	Beisp. 3
BEWEGUNG	1	ja	ja	ja
MUSTERK.	2	nein	ja	nein
RICHTUNGSÄ.	4	nein	ja	ja
PLOT	8	ja	nein	ja
GESAMTWERT FÜR p		9	7	13

n gibt die Spritenummer an, auf die sich dieser Befehl bezieht.

d ist etwas komplizierter bei kombinierten Funktionen anzuwenden, da der Bereich von d sich teilweise unterscheidet. Sollte der Wert von d nicht innerhalb des angewählten Bereichs liegen, können Fehler vorkommen.

BEWEGUNG (p=1) bringt das SPRITE einen Schritt in die von d angegebene Richtung. Die Schrittweite wird von CTLSPR 1 festgelegt und die Richtung muß im Bereich 0 - 8 liegen. 0 und 8 sind dabei identisch. Welcher Wert welche Richtung bedeutet, ist aus dem Diagramm unter Befehl 6 zu ersehen.

MUSTERKENNUNG (p=2) wechselt vom vorhandenen Spritemuster zum Muster d. Dieses Muster muß vorher von GENPAT generiert worden sein.

RICHTUNGSÄNDERUNG (p=4) übernimmt den anstehenden Richtungsvektor und überträgt ihn auf die neue Richtung d.

PLOT im Mittelpunkt bewirkt die Abbildung eines Punktes in der Mitte eines durch n gekennzeichneten Sprites. d hat hier keine Auswirkung.

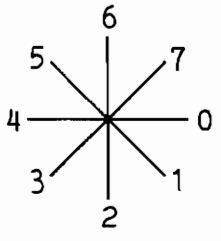
BEFEHL 5 ADJSPR p,n,v

n gibt hier immer die Spritenummer an.

p	Bedeutung	Dimension von v
0	Muster	0 - 31 (Größe 1), 0 - 127 (Größe 0)
1	Farbe	0 - 15
2	x Pos.	0 - 255
3	y Pos.	0 - 255
4	x Geschw.	0 - 255 (128 - 255 = neg.)
5	y Geschw.	0 - 255 (128 - 255 = neg.)

BEFEHL 6 VIEW Richtung,Entfernung

Richtung = 0 - 7
Entfernung = 1 - 255 - 0



SPK$ (screen peek)

Ergibt das Zeichen, auf dem der Kursor im Textmodus gerade steht. Ein Beispielprogramm, daß einen zufällig erzeugten Textbildschirm teilweise in ein Grafikscreen kopiert.

```
1 CLS
2 FOR X=1 TO 24*39
3 PRINT CHR$(RND*26+65);
4 NEXT X
10 DIM A$(24*32)
20 FOR Y=0 TO 23
30 FOR X=0 TO 31
40 CSR X,Y: LET A$=A$+SPK$
50 NEXT X
60 NEXT Y
70 VS 4: CLS
80 PRINT A$
90 GOTO 90
```

Anwendung von SPK$: Das Speichern von Bildschirmen, Erfassen von DSI-Eingaben usw.

GR$ (x,y,b)

x und y sind Koordinaten eines Bildfensters. b entspricht der Anzahl der zu lesenden Bits ab x,y. (Wenn b = 1 entspricht dies der POINT-Funktion). GR$ liest b Pixel unterhalb der Position x,y einschließlich und ergibt ein Character, dessen Code sich aus den untersten b bits, die vorher gesetzt und somit gelesen worden sind, errechnet. Zum besseren Verständnis ein Beispiel:

Bits werden wie gesagt vertikal abgelesen. Z.B. ergibt GR$ (20,190,4) ergibt folgendes Zeichen:

```
bit 7          0
bit 6          0
bit 5          0
bit 4          0
bit 3          Pixel bei 20,190
bit 2          Pixel bei 20,189
bit 1          Pixel bei 20,188
bit 0          Pixel bei 20,187
```

Wenn vorher die Funktion LINE 20,187,20,190 ausgeführt worden ist, läuft der Befehl GR$ folgendermaßen ab: Es werden die vier bits bei (20,190) (Position), (20,189), (20,188) und (20,187) gelesen. Diese vier bits werden nun auf die untersten vier bits des Wertes von GR$ übertragen, so daß, da alle bits gesetzt sind, ASC(GR$(20,190,4)) den Wert 15 ergibt. ASC(GR$(20,191,4)) ergäbe 7 und ASC(GR$(20,190,5)) ergäbe 30; experimentieren Sie weiter!

Mit dem Befehl GR$ lassen sich sehr einfach Hardcopies von Grafikscreens ausdrucken. Nachfolgend ein kleines Programm für den Panasonic Drucker KX-P1090:

```
1000 REM Grafikscreen-Copy (UNTERPROGRAMM)
1001 VS 4
1005 LPRINT CHR$(27);"A";CHR$(8)
1006 LPRINT CHR$(27);"P";CHR$(0)
1010 FOR Y=24 TO 1 STEP -1
1020 LPRINT CHR$(27);"K";CHR$(0);CHR$(1);
1030 FOR X=0 TO 255
1040 LET A$=GR$(X,8*Y-1,8)
1045 LPRINT A$;
1050 NEXT X
1055 LPRINT
1060 NEXT Y
1070 RETURN
```

DSI (direct screen input)

DSI ermöglicht eine ungehindertes direkte Eingabe von Text in den Bildschirm, bis (RET) gedrückt wird.

Folgende Funktionen sind während eines DSI wirksam (Noch einmal zur Erinnerung: CTL-Funktionen erreicht man durch gleichzeitiges Drücken der entsprechenden Tasten, ESC-Funktionen durch aufeinanderfolgendes Drücken der Tasten.):

```
CTL W        = Tabulator zurück
CTL (Eckige Klammer zu)                              = PMODUS *
CTL (Schrägstrich von links oben n. rechts unten) = SMODUS *
CTL ^        = Kursor an
CTL          = Kursor aus *
CTL D, Buchstabe A - O = PAPER A - O (1 - 15)
CTL F, Buchstabe A - O = INK   A - O (1 - 15)
ESC I        = Einfügen einer Leerzeile
ESC J        = Lüschen einer Zeile
ESC K        = Duplizieren einer Zeile
```

* Diese Funktionen sind nur unter vorheriger Eingabe von POKE 64325,0 und dann auf anderen Tasten zu erreichen (Englisches Tastaturscanning).

TEIL 4

TON

Der MTX kann ein breites Spectrum an Tönen erzeugen, so daß Programme noch interessanter gestaltet werden können. Zugleich ist die integrierte Technik komplex genug, um den Computer als Synthesizer einzusetzen.

Ton kann auf zweierlei Art gewonnen werden:

1. Direkt. So angesteuert, wird eine einzelne Note gespielt, bis ein Stopbefehl kommt (oder der Stecker herausgezogen wird).

2. Fortdauernd. Tonsequenzen können in diesem Modus gespielt werden, indem sie in einem Tonspeicher zwischengespeichert werden.

In beiden Fällen wird eine Tonanweisung benutzt, um dem Computer mitzuteilen, was verlangt wird.

Direkter Ton erreichen Sie durch die Anweisung

SOUND TONKANAL,FREQUENZ,LAUTSTÄRKE

KANAL - Es stehen insgesamt 4 Tonkanäle zur Verfügung. 0,1 und 2 sind voll verwendbar, während Kanal 3 als Rauschkanal eingesetzt wird. Der Einsatz von von Rauschen wird später behandelt.

FREQUENZ - Die Frequenz wird durch Werte zwischen 0 - 1023 bestimmt. Tontabellen im Anhang erklären den Zusammenhang zwischen diesen Werten und der erzeugten Frequenz. Die erzeugten Noten werden ebenfalls angegeben.

LAUTSTÄRKE - Die Lautstärke wird durch die Angabe eines Wertes zwischen 0 - 15 bestimmt, wobei 15 die höchste Lautstärke darstellt.

Versuchen Sie folgende Eingaben:

```
SOUND 0,200,10   (RET)
```

Drücken Sie beide RESET-Tasten und versuchen folgendes:

```
SOUND 1,600,10
```

und

```
SOUND 2,900,10
```

Testen Sie beide Töne gleichzeitig, indem Sie nicht RESET drücken. Auf diese Weise erzeugen Sie einen einfachen Akkord.

Experimentieren Sie mit eigenen Tönen, indem Sie verschiedene Kanäle, Frequenzen und Lautstärken eingeben. Es ist sinnvoll, die Tontabelle zu Hilfe zu nehmen, um den Effekt zu bestimmen, so daß nach und nach ein Verständnis über die Arbeitsweise des Tonprozessors gewonnen wird.

CONTINUES SOUND (Kontinuierlicher Ton) wird durch eine etwas aufwendigere Anweisung mit sieben Parametern erzeugt. Auf diese Weise kann die Änderung des Tons in seiner Höhe, Lautstärke und Dauer bestimmt werden. Um einen solchen, sich ändernden Ton zu erzielen, speichert der Computer die Anweisung in einem Tonpuffer ab.

Der Tonpuffer besteht aus einem Speicherblock, der nur für diese Aufgabe reserviert bleibt. Die Größe des Blocks wird durch den SBUF-Befehl bestimmt. SBUF 3 z.B. reserviert 3 Blöcke je Kanal. Wird SBUF nicht benutzt, sind immer 2 Blöcke je Kanal reserviert. Werden mehr Anweisungen benutzt, als dafür Speicherblöcke frei sind, kann es sein, daß eine oder jede Anweisung nicht ausgeführt wird. Auf diese Weise ähnelt der SBUF-Befehl der DIM-Anweisung. Jeder Block erfordert 12 Bytes/Kanal, so daß mehr Blöcke entsprechend weniger Programmspeicherplatz übrig lassen.

Jedesmal, wenn ein Continuous Sound Befehl eingegeben wird, erfolgt ein Eintrag im Tonpuffer. Jede Eingabe wird erst vollständig eingetragen, ehe der nächste Befehl wirken kann. D.h., daß lange Töne (bis zu mehreren Minuten) sich abspielen lassen, während das Hauptprogramm weiterarbeitet.

Im folgenden Beispiel werden wir einen einfachen Grundbefehl für die Ansteuerung von zwei Pufferblöcken zur Demonstration anwenden. Um Sie jedoch an den SBUF-Befehl zu erinnern, werden wir den Puffer auf den Wert 10 einstellen. Sie können einen beliebigen Wert bis 255 einsetzen.

Eingabe: SBUF 10 (RET)

Die komplette Parametereingabe sieht wie folgt aus:

SOUND KAN.,FREQ,LAUTST.,FREQ ÄNDERUNG,LAUTST.ÄNDERUNG,ZEIT,AKTION

Wir wollen nun alle Parameter einzeln besprechen und an einem Beispiel erläutern. Sie sollten dann soweit sein, auch eigene Programme zu testen.

CONTINOUS SOUND erhalten wir durch eine Ergänzung des Grundbefehls. Zwischen den Werten für die verschiedenen Eingabemodi besteht ein erheblicher Unterschied. Wenn Sie sich die Tontabelle ansehen, werden Sie feststellen, daß es zwei Tabellen gibt, eine für DIRECT SOUND und eine für den Modus SOUND BUFFER. Die Werte für den BUFFER-Modus haben ein breiteres Spektrum, um Ihnen eine differenziertere Arbeit zu erlauben.

Der erste Parameter bezieht sich auf den Kanal, der angesprochen werden soll. Der Wert für diesen Parameter liegt zwischen 0 und 2. Rauschen wird in Kanal 3 generiert, so daß unser Befehl für einen reinen Tonkanal wie folgt beginnt:

```
SOUND (0-2), . . .                z.B. SOUND 1
```

Als nächstes wird die FREQenz aus einem Bereich zwischen 0 - 1023 (DIRECT SOUND) für den Tonkanal gewählt. Die Frequenz bestimmt die Tonhöhe, wobei ein Blick auf die Tontabelle verrät, daß der Frequenzparameter sich reziprok zur Tonhöhe verhält. Da wir mit einer sehr hohen Note beginnen möchten, fangen wir also mit 0 an.

```
SOUND 1,0, . . .
```

Die Lautstärke eines Tones (bei Verwendung des Sound Buffers) kann zwischen den Werten 0 und 240 schwanken. Als Beispiel nehmen wir 100.

```
SOUND 1,0,100  (Wenn wir die Parametereingabe hier durch (RET)
               abbrechen, wird bereits ein Ton hörbar.)
```

Soll aber die Tonlage wechseln, können wir dies mit dem FREQ ÄNDERUNG Parameter bestimmen. Dieser Parameter kann einen Wert zwischen -32767 und +32767 einnehmen. Das negative Vorzeichen bewirkt einen Anstieg der Frequenz um diesen Wert, während ein Plus eine Frequenzabnahme verursacht. Eine Null an dieser Stelle hält die Frequenz konstant. Da in diesem Beispiel der hohe Ton abfallen soll, werden wir hier einen Wert von 10 eingeben.

```
SOUND 1,0,100,10, . . .
```

Auf ähnliche Weise können wir mit Hilfe des LAUTST. ÄNDERUNG Parameter die Lautstärke variieren. Wie zuvor liegt der Wertebereich dieses Parameters zwischen -32767 und +32767 mit 0 als Konstante. Eine Zurücknahme der Lautstärke erfolgt durch negative Werte. Unser Beispiel soll eine konstante Lautstärke haben, so daß hier 0 gesetzt wird.

```
SOUND 1,0,100,10,0, . . .
```

Die Dauer des Tons oder auch die Anzahl der auszuführenden FREQ und LAUTSST. Änderungen wird mit dem ZEIT-Parameter bestimmt. Hier liegt eine Skala von 0 bis 65535 vor. Eine Einheit repräsentiert 1/64 einer Sekunde. Wenn wir unser Beispiel mit 160 belegen, erhalten wir eine Tondauer von ca. 2,5 sek.

```
SOUND 1,0,100,10,0,160, . . .
```

Bislang wurden nur Parameter zur Tonbestimmung angegeben. Sollen aber mehrere Töne verknüpft werden, muß die Art dieser Verknüpfung ebenfalls angegeben werden. Dies geschieht mit Hilfe des AKTION-Parameters.

Soll der Ton unabhängig generiert werden, brauchen wir natürlich überhaupt keine Verknüpfung, was an dieser Stelle mit einer 1 bestimmt wird. Soll aber ein zweiter Ton hieran angebunden werden, so daß wir einen Dauerton haben, wird eine 0 angegeben. Damit erhält der Computer die Anweisung, die Endwerte des einen Tons als Anfangswerte des zweiten Tons zu übernehmen. Unser Beispiel soll eigenständig sein, also geben wir eine 1 ein.

```
SOUND 1,0,100,10,0,160,1
```

Damit Tonbefehle leichter korrigiert und geändert werden können, ist es angeraten, sie in BASIC-Zeilen unterzubringen.

```
10        SBUF 10
20        SOUND 1,0,100,10,0,160,1
30        EDIT 20
```

Geben Sie obiges Programm gefolgt von RUN ein, sollte der Ton hörbar sein. Haben Sie auch den Tonausgang angeschlossen bzw. die Lautstärke des TV's aufgedreht?

Die Auswirkung des AKTION-Befehls läßt sich nun durch den Umtausch der 1 mit 0 testen:

```
20        SOUND 1,0,100,10,0,160,0
```

Sie werden feststellen, daß der Ton nach jedem neuen Programmdurchlauf weiter an Höhe verliert. Setzen Sie wieder die 1 ein, um den Unterschied festzustellen.

Testen Sie folgende Sounds:

```
1 SOUND 1,5,15,-6,-1,1000,1
2 SOUND 1,5,15,6,-1,1000,1
3 SOUND 1,5,15,0,-12,4000,1
```

TONFOLGEN

Töne können in einem BASIC-Programm dergestalt verknüpft werden, daß Tonkombinationen oder Tonfolgen entstehen. Testen Sie folgendes Programm und versuchen Sie sich dann an eigenen Verknüpfungen:

```
10 SBUF 10
20 SOUND 1,5,15,1,1,750,1
30 SOUND 2,1,0,10,0,750,1
40 SOUND 3,7,15
```

Die Zeilen 30 und 40 stellen in diesem Fall eine Besonderheit dar, als Zeile 30 unbedingt der Ergänzung durch Zeile 40 bedarf. Der Lautstärke-Parameter in Zeile 30 ist 0, was gleichbedeutend als 'Ton aus' verstanden werden kann. Durch die Lautstärke-Stellung 15 in Zeile 40 haben wir den Effekt eines plötzlichen Tonschwalls, sobald diese die Zeile 30 aktiviert. Die Tonerzeugung verlangt Fingerspitzengefühl, weshalb wir empfehlen, möglichst oft neue Kombinationen durchzuspielen und auf diese Weise ein "Gefühl" für die Möglichkeiten des Computers zu entwickeln. Wie obiges Beispiel andeutet, kann auch der Einbezug des Geräusch-Kanals Effekte erzeugen, die den Gesamteindruck wesentlich verbessern.

Ton kann auf vielerlei Weise zur Unterstützung des Programms einbezogen werden, insbesondere, um grafische Effekte zu verstärken. Am günstigsten ist es, das Programm so zu strukturieren, daß Unterprogramme immer dann aufgerufen werden, wenn dies angebracht erscheint. Wir haben ein nützliches Unterprogramm aufgelistet, daß Sie auch für Ihre Programme nutzen können:

```
4000 REM SOUND SUBROUTINE
4010 SBUF 2
4020 SOUND 0,100*8,15*64,1,-1,8*64,1
4030 SOUND 1,101*8,15*64,1,-1,8*64,1
4040 LET CHAN=1
4050 GOSUB 5000
4060 SOUND 0,0,0
4070 SOUND 1,0,0
4080 LET DELAY = 400
4090 FOR N=15 TO 0 STEP -1
4100 SOUND 3,4,N
4110 FOR J=0 TO DELAY
4120 NEXT J
4130 LET DELAY =DELAY -30
4140 NEXT N
4150 STOP

5000 REM TEST SUBROUTINE
5010 IF PEEK(CHAN*10+64082)<>PEEK(CHAN*10+64082+4) THEN GOTO
5010
5020 RETURN
```

Der letzte Teil des Unterprogramms regelt die Tongenerierung. Der zweite Teil ab Zeile 5000 ist für die Überwachung der so erzeugten Töne gedacht. Zeile 5010 z.B. überprüft die Vollständigkeit der Tonparameter der Zeilen 4010, 4020 und 4030. Nach erfolgtem Test kehrt das Programm zur Zeile 4060 zurück. In den Zeilen 4060 und 4070 werden die ersten Töne ausgeschaltet und in der Zeile 4100 wird der Rauschkanal zugeschaltet.

Wie Sie sehen, ist die Soundgenerierung sehr komplex und auch untereinander verzweigt. Sie werden sicherlich noch viele Sounds entsprechend den Angaben in diesem Teil und im Anhang dieses Handbuchs durchprobieren müssen, ehe Sie sofort wissen, welche Parameter für welche Sounds gelten.

TEIL 5

ASSEMBLER

Dieses Kapitel soll keine Lektion im Programmieren mit der Maschinensprache sein, sondern nur die Übertragung von mit dem MTX-Assembler geschriebener Assembler-Sprache in MTX-BASIC-Programme erklären.

Der Assembler wird dadurch aufgerufen, daß dem Computer mitgeteilt wird, daß man in Maschinensprache programmieren möchte und wohin im BASIC der Maschinencode plaziert werden soll.

Schauen Sie sich einmal das untenstehende Programm an. Die Zeilen 10, 20, 40 und 50 sind normale BASIC-Zeilen. Zeile 30 aber ist mit dem Assembler erstellt worden. Wenn Sie die Zeilen 10, 20, 40 und 50 eingeben und dann die nachfolgenden Instruktionen befolgen, werden Sie sehen, wie dies gemacht wird.

```
10        PRINT "Anfang des Programms"
20 POKE 40000,5

30 CODE

8030      LD A,(40000)
8033      INC A
8034      LD (40000),A
8037      RET

Symbols:

40 PRINT PEEK(40000)
50 PRINT "ENDE"
```

Geben Sie nun folgendes ein: ASSEM 30 (RET)

Am unteren Bildschirmrand erscheint das Wort Assemble>

Drücken Sie noch einmal (RET).

Die Anzeige sollte nun so aussehen:

```
8030      RET
```

Insert

(Anmerkung: Die Hex-Adressen gelten in diesem Fall für den MTX 500; wenn Sie von diesen Adressen 4000 Hex abziehen, erhalten Sie die Werte für den MTX 512).

Insert teilt Ihnen mit, daß der Computer sich jetzt im Insert-Modus befindet. Das bedeutet, daß, wenn Sie nun irgendwelche Maschinencodebefehle eingeben, nichts überschrieben oder gelöscht wird, was hier steht.

8030 ist die Adresse, ab der Eingaben eingefügt werden. Diese Zahl ist eine hexadezimale Zahl.

RET ist der Befehl, der im Augenblick in dieser Adresse steht.

Geben Sie nun LD A,(40000) (RET) ein.

```
     8033      RET
```

Insert

wird erscheinen. D.h., daß der nächste Maschinencodebefehl ab Adresse 8033 eingefügt wird und diese Adresse z.Z. durch einen RET-Befehl belegt ist.

Geben Sie ein: INC A

Auf dem Bildschirm steht nun:

```
8033           INC A   RET
```

Wenn Sie jetzt (RET) drücken, wird die Fehlermeldung 'Bad Code' erscheinen. Offensichtlich müssen Sie das RET am Ende der Zeile löschen. Verwenden Sie dazu die EOL-Taste auf dem abgesetzten Zehnerblock.

Drücken Sie jetzt noch einmal (RET) und geben Sie noch

LD(40000),A (RET)

ein. Drücken Sie (CLS) gefolgt von (RET) und Assembler> wird jetzt wieder auf dem Bildschirm erscheinen, womit Sie zum Assembler zurückgekehrt sind. Es ist möglich, sowohl vom Insert- als auch vom Edit-Modus zum Assembler zurückzukehren.

Um dieses Maschinencodeprogramm aufzulisten, muß erst der Programmzeiger zum Anfang des Programms gebracht werden. Dies erfolgt durch Eingabe von T (RET).

Drücken Sie nun L (RET).

Der Programmzeiger merkt sich die letzte Position und listet das Programm nur bis zu diesem Punkt auf:

```
8030      LD A,(40000)
8033      INC A
8034      LD(40000),A
8037      RET
```

Symbols:

Um zum BASIC zurückzukommen, löschen Sie den Bildschirm mit (CLS) und drücken die RETURN-Taste.

Ready wird wieder erscheinen.

Sie können sich nun das gesamte Programm anzeigen lassen (LIST) und es sollte wie auf den vorherigen Seiten nun mit der zusätzlichen Zeile 30 erscheinen. Wenn Sie das Programm starten, zeigt der Bildschirm folgendes:

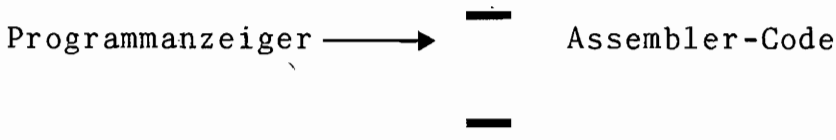
```
Anfang des Programms
 6
ENDE
```

Zusammenfassung

Der Assembler wird durch ASSEM (Zeilennummer) aufgerufen. Ins BASIC springen Sie durch (CLS) und (RET) zurück.

Um Maschinencodebefehle vom Assembler aus einzugeben, drücken Sie (RET), und (CLS) gefolgt von (RET), um zum Assembler zurückzukommen.

Das MC-Programm listen Sie im Assembler mit T (RET) gefolgt von L (RET) auf.



Der Programmanzeiger merkt sich die Zeile, die editiert wird oder ab der neue Befehle eingefügt werden.

T setzt den Zeiger zum Anfang des Programms. Dadurch müssen Sie sich vor dem Listen des Programms nicht merken, ab welcher Adresse das Programm beginnt.

INSERT MODUS

Im Insert-Modus werden eingegebene MC-Programmzeilen ab der angezeigten Adresse eingefügt. Der korrekte Speicherplatz wird für jede Zeile, wie sie eingeben wird, im RAM freigemacht.

Sie können auf vier Wegen in den Insert Modus gelangen:

1. (RET) beginnt bei der Programmzähler-Position.
2. #n beginnt bei der Hexadezimal-Adresse n.
3. n beginnt bei der Dezimal-Adresse n.
4. Label beginnt bei dem angegebenen Label, falls er existiert.

Aus dem Insert-Modus steigen Sie durch (CLS) (RET) aus.

EDIT MODUS

Im Edit Modus wird jede angezeigte Zeile durch die einzugebene ersetzt. Dadurch unterscheidet sich dieser Modus vom Insert-Modus, in dem Zeilen eingefügt werden, ohne etwas zu verändern. Wie beim Insert gelangt man auf vier Wegen in den Edit Modus:

1.	E (RET)	beginnt bei der Zeile, auf die der Programmzeiger zeigt.
2.	E #n	beginnt bei der Hex-Adresse n.
3.	E n	beginnt bei der Dezimal-Adresse n.
4.	E Label	beginnt bei dem bezeichneten Label, falls er existiert.

Falls ein Label mit Namen "E" in dem Maschinencodeprogramm existiert, steigt man mit dem Befehl E (RET) in den Insert Modus ab Label E ein und nicht ab der durch den Programmzeiger markierten Zeile in den Edit Modus.

LIST

1.	L (RET)	listet das MC-Programm ab der durch den Programmzeiger markierten Adresse auf
2.	L #n	listet das Programm ab der Hex-Adresse n auf.
3.	L n	listet das Programm ab der Dezimal-Adresse n auf.
4.	L Label	listet das Programm ab dem bezeichneten Label auf.

Wie beim Edit Modus führt bei einem vorhandenen Label L der Befehl L (RET) zu einem Aufruf des Insert Modus ab diesem Label und nicht zu einem Listing des Programms ab dem Programmzeiger.

LÖSCHEN

MC-Programmzeilen können im Insert und im Edit Modus gelöscht werden. Wenn eine Zeile angezeigt wird, erscheint der Kursor zwischen der Adresse und dem Mnemonikkürzel bzw. dem Code.

Nun können Sie (EOL) oder mehrere Leerzeichen, die den Code überschreiben, gefolgt von (RET) eingeben, worauf die Zeile gelöscht wird.

Falls Sie die Adresse ändern, wird sich auch der Programmzeiger entsprechend ändern, vorausgesetzt, daß sich die neue Adresse im Bereich des bereits vorhandenen Programms befindet.

LABELS

Adresslabels, denen ein Doppelpunkt folgen muß, können überall eingesetzt werden:

```
LABEL:NOP
      JP LABEL
```

Kommentare können hinter jedem Mnemonik-Kürzel eingefügt werden, vorausgesetzt, daß zwischen Mnemonik und Kommentar ein Semikolon eingegeben wird:

```
RET;Ende des Programms
```

DS: Der Befehl DS wird benutzt, um einen Speicherleerraum bis zu 254 bytes ab der dem DS-Befehl zugehörigen Adressen zu schaffen, d.h., daß ab dieser Adresse die folgenden auf Null gesetzt werden:

```
DS 200
```

DB: Hiermit können die folgenden Adressen mit bestimmten Werten oder mit dem ASCII-Code von in Anführungszeichen eingeschlossener Buchstaben definiert werden:

```
DB 1,2,"ABC"
```

Am günstigsten ist es, wenn Sie Ihre MC-Programme in die ersten Zeilen eines BASIC-Programms setzen, da sie nicht verändert werden, wenn BASIC-Zeilen mit höheren Zeilennummern editiert werden.

Nachdem Sie Ihr MC-Programm geschrieben haben, können Sie es wie in BASIC mit RUN (RET) ausführen lassen.

FRONT PANEL

Das FRONT PANEL Betriebssystem können Sie verwenden, um Ihre MC-Programme zu testen und Programmfehler herauszufinden.

Geben Sie ein: PANEL (RET)

Die Z80-Registerinhalte werden rechts oben und ein Speicherblock unten am Bildschirm angezeigt.

Drücken Sie: L2000

Und ein Teil eines MC-Programms wird aufgelistet. Das PANEL listet Programme auf, zeigt Speicher- und Registerinhalte an und bietet Ihnen die Möglichkeit, Ihr MC-Programm Schritt für Schritt ablaufen zu lassen und dabei die Ausführung jedes einzelnen Befehls zu überwachen.

Wenn unter Verwendung des MTX-Assemblers MC-Programme geschrieben worden sind und man es ablaufen läßt, hält die Break-Taste das Programm an und das Panel zeigt den augenblicklichen Status des Computers.

Es folgt nun noch eine Hilfsroutine, die Sie mit Hilfe des Assemblers leicht eingeben können und die Ihnen beim Programmieren in BASIC sehr wahrscheinlich nützlich sein wird; es ist eine RENUMBER-Routine zur Neunumerierung Ihrer BASIC-Programmzeilen. Die Parameter der Sprungbefehle wie GOTO, GOSUB oder ON x GOTO sowie LIST werden allerdings nicht neunumeriert. In HL steht die neue Anfangszeile des BASIC-Programms und in DE die Schrittweite.

```
10 REM ** RENUMBER **
20 CODE

401C          LD HL,10
401F          LD DE,10
4022          LD IX,16384; (MTX 500: 32768)
4026 LOOP:    LD C,(IX+0)
4029          LD B,(IX+1)
402C          XOR A
402D          CP C
402E          RET Z
402F          LD (IX+2),L
4032          LD (IX+3),H
4035          ADD IX,BC
4037          ADD HL,DE
4038          JR LOOP
403A          RET

Symbols:
LOOP      4026
```

Es folgt eine Gesamtauflistung sämtlicher MTX-BASIC-Befehle mit kurzen Erklärungen und ihren Kurzformen. Die Kurzformen haben den Vorteil, daß sie meist schneller einzugeben sind und daß dahinter keine Leerstelle folgen muß. Wie gewohnt werden die Abkürzungen durch einen Punkt abgeschlossen.

ABS(Zahl)

Ergibt den absoluten Wert der angegebenen Zahl, d.h., daß die Zahl immer positiv ist.

Beispiel:

```
ABS (59) = 59
ABS (-59)= 59
```

Kurzform: AB.

ADJSPR p,n,v

Dieser Grafikbefehl ändert den Wert eines bereits definierten Sprites (durch SPRITE oder Folgebefehle). Der Vorteil von ADJSPR liegt darin, daß nur ein Parameter geändert werden muß und als Folge die Befehlsausführung wesentlich beschleunigt wird.

n ist immer die Spritenummer.

p	Bedeutung	Skala von v
0	Muster	0- 31 (Größe 1), 0-127 (Größe 0)
1	Farbe	0- 15
2	x-Koordinate	0-255
3	y-Koordinate	0-255
4	x-Geschw.	0-255 (128-255=negativ)
5	y-Geschw.	0-255 (128-255=negativ)

Kurzform: AD.

AND

Siehe BOOLSCHE AUSDRÜCKE

ANGLE(Winkel)

Der Computer speichert eine Richtung, die für Anweisungen wie ARC, DRAW und MVSPR genutzt wird.

Die Richtung durch zwei Befehle geändert werden: PHI und ANGLE ANGLE initialisiert die Richtung des spezifizierten Winkels. Der Winkelwert wird in rad gemessen und nimmt aus der Horizontalen nach rechts weisend im entgegengesetzten Uhrzeigersinn zu.

Kurzform: AN.

ARC(Länge),(Winkel)

Dieser Befehl zieht einen Kreisbogen. Der Ausgangspunkt ist die aktuelle Plot-Position und die Richtung der aktuelle ANGLE-Wert. Sowohl Plot-Position als auch ANGLE-Wert werden durch diesen Befehl verändert. Die Länge gibt die Bogenlänge an. Der Winkelwert bestimmt die Krümmung des Bogens; einen Winkel von mehr als 360 Grad oder 2xPI rad ergibt einen rekursiven Bogenwert.

Kurzform: AR.

ASC(STRING)

Benennt den Code des ersten Zeichens eines Strings.

Beispiel:

```
10 LET A=ASC("B")
20 PRINT A
```

Ergibt den Ausdruck des ASCII-Codes von "B" auf dem Bildschirm (66).

ASSEM (Zeilen Nr.)

Aktiviert die Assemblerfunktion, um die Maschinencode-Programm-Eingabe zu ermöglichen und danach in die spezifizierte BASIC-Zeile zu assemblieren (siehe CODE). Sollte die Zeile bereits belegt sein, erfolgt nur dann ein Einstieg in den Assembler, wenn diese durch CODE indiziert ist. Sehen Sie hierzu auch den Abschnitt Assembler des Handbuchs.

Kurzform: A.

ATN(Zahl)

Ergibt den Arcustangens von Zahl.

ATTR p,Status

ATTR bestimmt die Wirkung eines Grafikbefehls (z.B. PLOT, DRAW oder ARC) im Grafikmodus.

Die Attribute sind nicht exclusiv und können in beliebiger Kombination eingesetzt werden.

Der Status bezieht sich auf EIN oder AUS, wobei

1 = EIN
0 = AUS bedeutet.

p bezieht sich auf einen Wert zwischen 0 - 3.

p = 0 Invertierte Printdarstellung. Zeichen werden in der PAPER-Farbe auf einen INK-Hintergrund abgebildet.

p = 1 Overprint. Zeichen werden über bereits vorhandene Zeichen geschrieben, wobei scnon gesetzte Pixel gelöscht werden.

p = 2 Unplot. Plot erfolgt in der PAPER-Farbe.

p = 3 Overplot. Plot erfolgt normal mit der Ausnahme, daß bereits gesetzte Pixel gelöscht werden. Bei CLS und anderen Funktionen wird der vorhandene Text nicht überschrieben, aber ein Farbwechsel kann vorkommen.

Kurzform: AT.

AUTO Zeilennr.,Increment

Dieser Befehl schaltet die automatische Zeilennumerierung ein. Die Numerierung erfolgt ab der genannten Zeile in Schritten, die durch das Inkrement bestimmt werden.

Beispiel:

AUTO 10,10

bewirkt die Zeilenzuteilung 10,20,30,40.....

AUTO wird durch CLS und (RET) ausgeschaltet.

Kurzform: AU.

AUTO SCROLL

Die AUTO SCROLL Einrichtung dient dazu, den Ausdruck weiterer Texte zum Bildschirm zu unterbrechen, sobald dieser voll ist.

AUTO SCROLL kann entweder vom Programm oder vom Anwender zu- oder abgeschaltet werden. Die Zu- und Abschaltung erfolgt über die PAGE-Taste.

Ist AUTO SCROLL eingeschaltet, reicht das Drücken einer beliebigen Taste, um ein SCROLL über eine ganze Seite auszulösen.

Beispiel:

```
10 FOR I=1 TO 1000
20 PRINT I
30 NEXT
RUN
```

Drücken Sie nun (PAGE), wird die Druckausgabe angehalten, um durch das Drücken einer beliebigen Taste beliebig oft fortzufahren. Abermaliges Drücken der PAGE-Taste schaltet diese Automatik wieder aus. Für ein Programm gilt entsprechendes über die EXC P Sequenz.

Beispiel:

```
10 PRINT CHR$(27);"P";
20 FOR I=1 TO 1000
20 PRINT I
30 NEXT
```

Sie werden bei RUN feststellen, daß AUTO SCROLL aktiviert ist und in der üblichen Weise gesteuert werden kann.

Siehe auch LIST,ASSEM,PRINT

BAUD Kanal,Baudrate

Bestimmt die Baudrate des angewählten RS232-Kanals. Folgende Baudraten sind zugelassen:

75	1200
110	2400
150	4800
300	9600
600	19200

Beispiel:

BAUD 0,1200 ergibt eine Ausgabegeschwindigkeit von 1200 Baud auf dem RS232-Port 0.

Kurzform: B.

BOOLSCHE AUSDRÜCKE

Für eine computergerechte Verarbeitung von Problemen mit WAHR und FALSCH Bedingungen bedarf es einer besonderen algebraischen Logik. Hierzu dienen die sog. Boolschen Ausdrücke, deren Regeln vom Computer verstanden werden und entsprechend verarbeitet werden können.

```
10        INPUT "ENTER X,Y ";X,Y
20        PRINT "X=1 ",X,(X=1)
30        PRINT "Y=2 ",Y,(Y=2)
40        PRINT "X=1 und Y=2",(X=1 AND Y=2)
50        GOTO 10
```

Beachten Sie bitte, daß Wahrheitsaussagen wie Y=1 im Zusammenhang mit Boolschen Ausdrücken immer in Klammern gesetzt werden.

Wenn Sie obiges Programm laufen lassen und Werte für X und Y eingeben, werden Sie feststellen, daß wahre Aussagen durch eine -1 und falsche Aussagen mit 0 quittiert werden. Es gibt keine anderen Wahrheitsangaben.

Ausdrücke, die einen Wahrheitsgehalt beinhalten, sind definitionsgemäß BOOLSCHE AUSDRÜCKE. Wird ein solcher Ausdruck in eine IF-Anweisung eingesetzt, entfällt die erforderliche Klammer.

Beispiel:

```
40 IF X=1 AND Y=2 THEN STOP
```

BOOLSCHE REGELN

Es gibt drei Boolsche Operationen: AND, OR und NOT.
Ferner sechs relative Operationen: <, >, =, <>, <= und >=.

Ein relativer Ausdruck besteht dann, wenn eine Relation zwischen zwei Werten gleicher Art hergestellt wird.

Beispiel:

```
X<>2
A$="AAA"+"BBB"
(X=2)=(Y=3)
```

Relative Ausdrücke enthalten Wahrheitswerte.

Da ein Boolscher Ausdruck Wahrheitswerte enthält, sind relative Ausdrücke per Definition ebenfalls Boolsche Ausdrücke. Indem man relative Ausdrücke mit Boolschen Operationen verknüpft, lassen sich komplexere Zusammenhänge zwischen einzelnen Werten herstellen und auswerten.

Beispiel:

```
10 PRINT (NOT 2=2)
20 PRINT (2=2 OR 3=2)
```

Beispiel für einen Boolschen Ausdruck:

```
10 INPUT X,Y
20 IF X=2 AND Y=2 OR Y=7 THEN STOP
30 GOTO 10
```

Dieses Programm endet dann, wenn X=2 und Y=2 oder X=2 und Y=7 sind.

Bei einem Beispiel wie

```
10 PRINT (2*2=5 OR 3+3=4 OR 2=2 AND 1=2)
```

müssen wir zunächst wissen, in welcher Reihenfolge der Ausdruck ausgewertet wird.

Genauso wie es in der Arithmetik eine Rechenhierarchie gibt, gelten auch bestimmte Regeln für die Abarbeitung Boolscher Ausdrükke. Es ist bereits bekannt, daß ein * Vorrang vor einem + hat; also:

3*4+5 = (3*4)+17 und nicht
3*(4+5) = 27

Um die Abarbeitung eines komplexeren Ausdrucks zu übersehen, müssen einige einfache Regeln beachtet werden.

Arithmetische Operationen haben die höchste Priorität.

Relative Operationen sind untereinander gleichrangig und den arithmetischen Operationen unter- und den Boolschen Operationen übergeordnet.

AND, OR und NOT haben somit die niedrigste Priorität.

Untereinander gilt die Rangfolge AND, OR, NOT, so daß von allen Operationen NOT zuletzt berücksichtigt wird.

PRIORITÄTENFOLGE

```
^
*       /
+       -
=       <>    <     >     <=     >=
AND
OR
NOT
```

Die Priorität bestimmt die Reihenfolge der Abarbeitung, wobei wie bei rein arithmetischen Ausdrücken Klammern diese Folge ändern können.

Beispiel:

```
10 PRINT (2*2=5 OR 3+3=4 OR 2=2 AND 1=2)
```

ist gleichbedeutend mit

```
10 PRINT ((2*2)=5) OR ((3+3)=4) OR ((2=2) AND (1=2))
```

CHR$(Zahl)

Ergibt das Zeichen, dessen Code durch Zahl bezeichnet wird.

Beispiel:

```
PRINT CHR$(65)
```

Ergebnis: "A"

Kurzform: CH.

CIRCLE X,Y,r

Zieht einen Kreis mit dem Radius r mit dem Mittelpunkt auf den Koordinaten X,Y.

Beispiel:

```
10 VS 4
20 CLS
30 CIRCLE 100,100,50
40 PAUSE 1000
```

Kurzform: CI.

CLEAR

CLEAR löscht alle Variablen.

Kurzform: CLE.

CLOCK String (CLOCK=Uhr)

Z.B. CLOCK "120000"

Die Uhr wird mit dem Wert des bezeichneten Strings initialisiert. Die im MTX eingebaute Echtzeituhr kann bis 100 Stunden hochzählen und dabei Stunden, Minuten und Sekunden angeben. Danach beginnt ein neuer Durchlauf ab 0.

Sehen Sie hierzu auch 'TIME$'.
Beispiel:

```
10 CLOCK "120000"
20 CLS
30 PRINT TIME$;CHR$(26);
40 GOTO 30
50 REM CHR$(26)=HOME
```

Kurzform: CLO.

CLS

Clear screen (Löschen des Bildschirms)

Im TEXT-Modus bewirkt CLS das Löschen des Bildschirms oder des Fensters.

Im GRAFIK-Modus löscht CLS ebenfalls den Bildschirm, es sei denn, einer der beiden Bildschirmattribute wurde mittels der ATTR-Funktion aktiviert. In diesem Fall ist es notwendig, ATTR auf AUS zu setzen, ehe CLS wirken kann.

Kurzform: C.

CODE

CODE bezeichnet weniger eine Anweisung als einen Hinweis dafür, daß die nachfolgende Zeilen als ASSEMBLER-CODE verstanden werden soll. Das Wort wird nicht eingetippt, sondern durch den ASSEMBLER in die BASIC-Zeile eingefügt.

Beispiel:

```
10          REM PROGRAMMBEGINN
20          CODE

801C        LABEL: LD A, (HL)
801D               RET
Symbols:
801C    LABEL

30          REM PROGRAMMENDE
```

Alle Labels werden zum Schluß eines Codeblocks aufgelistet. Label sind auf den jeweiligen Block bezogen.

COLOUR p,n (COLOUR=Farbe)

Die COLOUR-Anweisungen bestimmen die Farben des Bildschirms im GRAFIK-Modus.

n bestimmt den Farbwert (siehe Farbtafel).

Der Wert von p bestimmt, welche Gebiete des Bildschirms von der bezeichneten Farbe betroffen ist.

Es werden zwei Farbtafeln bestimmt. Die Printfarben beziehen sich auf jene Farben, die von PRINT angesprochen werden. Die non-print oder PLOT-Farben beziehen sich auf jene Farben, die bei PLOT oder andere Grafikfunktionen Anwendung finden.

p = 0 Print paper
p = 1 Print ink
p = 2 Non-print (plot) paper
p = 3 Non-print ink
p = 4 Border colour (Randfarbe)

Kurzform: COL.

CONT

CONT läßt sich unmittelbar nach einer STOP-Anweisung oder nach Drücken der BREAK-Taste anwenden, um den Programmablauf fortzuführen. Das Ändern eines Programms (z.B. durch EDIT) setzt die CONT-Funktion außer Kraft.

Kurzform: CO.

COS(Winkel)

Ergibt den Cosinus des in rad bezeichneten Winkels.

CRVS n,t,x,y,w,h,s

Um ein eigenes Fenster zu definieren, müssen die CRVS-Parameter Größe, Ort und Fensterindex enthalten. Das so definierte Fenster wird durch den VS-Befehl aufgerufen.

n Fensterindex (0-7)
t Bildmodus (0=Text, 1=Grafik)
x Horizontale Koordinate linker Bezugspunkt
y Vertikale Koordinate oberer Bezugspunkt
w Breite des Fensters in Zeichen
h Höhe des Fensters in Zeilen
s Breite des Gesamtbildschirms (40=Text, 32=Grafik)

Sollten Sie versehentlich den falschen Wert für s angegeben haben, wird das Fenster verzerrt abgebildet. Dies kann allerdings auch vorteilhaft ausgenutzt werden.

Kurzform: CR.

CSR X,Y

Setzt den Kursor auf die Position der X,Y Kooerdinaten.

Beispiel:

```
10 CSR 12,10
```

Der Kursor befindet sich nach Ausführung des Programms auf der Position 10,12. Jede nachfolgende PRINT-Anweisung wird eine Ausgabe an dieser Stelle erwirken.

Kurzform: CS.

CTLSPR p,x

p = Parameter für eine der folgenden Angaben. Die Werte für x werden in den jeweils nachfolgenden Erklärungen angegeben.

0 = Geschwindigkeit
1 - 255 und 0 (1=Höchstgeschwindigkeit)

1 = Entfernung
Enthält die Strecke 'x' Pixel für Bewegungsabläufe

2 = Spritezahl
0 - 32 (Es muß mindestens 1 Sprite angegeben werden)

3 = Spritezahl im Umlauf
Enthält die Anzahl der Sprites, die den Bildschirm "umkreisen" sollen. Die angegebene Zahl darf die Anzahl der Sprites aus Index 2 nicht überschreiten.

4 = PLOT SPRITE
Ein PLOT SPRITE kann definiert werden, so daß dieses bei jeder Plot-Anweisung an dieser Stelle auftaucht. Ein solches Sprite folgt jedem Punkt und jeder Linie, die durch GRAFIK-Befehle erzeugt werden. Die Plot-Eigenschaft kann jeder der auf übliche Weise erzeugten 32 Sprites zugeordnet werden. x definiert die Spritenummer.

5 = Anzahl der beweglichen Sprites 0 - 32
Bestimmt die Anzahl der Sprites, die sich selbsttätig mit durch entsprechende SPRITE und ADJSPR-Befehle x,y Geschwindigkeitsfaktoren bewegen.

6 = Größe und Größenfaktor
x=0 Größe 8x8 Faktor 1
x=1 Größe 8x8 Faktor 2
x=2 Größe 16x16 Faktor 1
x=3 Größe 16x16 Faktor 2

Kurzform: CT.

DATA w1,w2,w3, . . .

Sobald der Computer auf eine READ-Anweisung stößt, sucht er nach der nächstfolgenden DATA-Angabe. Sobald diese Angabe gefunden wurde, wird ein Zeiger für die darin enthaltenen Werte aktiviert. Nach jeder READ-Anweisung für einen neuen Wert wird der DATA-ZEIGER auf den nächsten Wert weitergesetzt.

Beispiel:

```
10        REM Ausdruck von Werten aus DATA-Befehlen
20        FOR I=1 TO 10
30        READ X
40        PRINT X
50        NEXT I
60        DATA 1,1,2,2,3,3
70        DATA 4,4,5,5
```

DATA-Angaben werden nicht selbst "aktiv", sondern stellen nur die Daten für eine READ-Anweisung zur Verfügung.

Sobald alle Daten einer Zeile von der READ-Anweisung gelesen wurden, sucht der Computer nach der nächsten DATA-Angabe und positioniert den DATA-ZEIGER entsprechend.

Sind keine neuen Daten vorhanden, erfolgt ein NO DATA Fehler. Sie werden feststellen, daß das Leerzeichen hinter DATA als Teil des DATA-Strings angenommen wird. Um dies mit Sicherheit auszuschließen, ist es empfehlenswert, die Abkürzung DA. zu benutzen.

Beispiel:

Eingabe: 10 DA.AAA,BBB

anstelle von

```
10 DATA AAA,BBB
```

Siehe auch READ und RESTORE

Kurzform: DA.

DIM (Feldbezeichnung)

Ehe ein Feld genutzt werden kann, muß der erforderliche Speicherplatz reserviert werden.

Beispiel:

```
10 DIM A(10,10),A$(1000)
```

Jedes Element eines numerischen Feldes beansprucht 5 Bytes und jedes Element eines alphanumerischen Feldes 1 Byte. Ein Feld kann nicht ohne vorheriges Löschen aller Variablen neu dimensioniert werden.

Beispiel:

```
10 DIM A(10)
20 DIM A(20)
```

Hieraus würde ein Fehler entstehen, da das Feld bereits vorhanden ist, sobald Zeile 20 abgearbeitet wird.

Beispiel:

```
10 LET A$="ABC"
20 DIM A$(100)
```

Auch hieraus ergibt sich ein Fehler, da die Zeile 10 automatisch ein Alphafeld definiert, um die Daten "ABC" unterzubringen. Tatsächlich würde das Feld oder eher dieser String 64 Zeichen unterbringen können.

Alle durch Variablen definierten Strings werden intern als Felder behandelt. Der Speicherplatz eines einfachen Strings oder Feldes richtet sich aber danach, ob es mit einem DIM-Befehl definiert worden ist oder direkt durch eine Variable.

Wenn ein DIM-Befehl benutzt wurde, richtet sich die Speicherkapazität nach der definierten Anzahl. Ein Beispiel:

```
10 DIM A$(1)
```

wird nur ein Zeichen speichern können.

Wenn ein String durch die direkte Zuweisung einer Zeichenkette zu einer Variablen definiert wurde, richtet sich die Speicherkapazität dieses Strings nach der Länge des ersten zugewiesenen Strings, was bedeutet, daß die Gesamtkapazität nicht mehr als das nächste Vielfache von 64 ausgehend von der Länge der ersten Stringdefinition darstellt. Dazu ein Beispiel:

```
10 LET A$="ABC"          Möglicher Speicherplatz:  64 bytes
20 LET B$(100)="X"             "          "      : 128 bytes
```

Kurzform: DI.

DRAW X

Zieht eine Linie der Länge x ab der augenblicklichen PLOT-Position in die aktuelle Richtung (Gesetzt durch ANGLE oder PHI). Die neue augenblickliche PLOT-Position wird dem Ende der Linie zugewiesen.

Kurzform: DR.

DSI (Direct Screen Input)

Diese Anweisung gestattet eine direkte Texteingabe in ein Screen mit freier Kursorsteuerung und Bildschirmeditierung. DSI wird erst durch Drücken der Return-Taste aufgehoben. Im DSI-Modus bewirkt die BREAK-Taste keinen Abbruch, sondern die Auslösung von CTL-C.

CTL W = Tab Rückstellung
CTL (Eckige Klammer zu) = PMODUS
CTL (Schrägstrich von links oben nach rechts unten) = SMODUS
CTL ^ = KURSOR EIN
CTL _ = KURSOR AUS
CTL D ,Buchstabe A-O = PAPER A-O (1-15)
CTL F ,Buchstabe A-O = INK A-O (1-15)
ESC I = Leerzeileneinschub
ESC J = Zeilenlöschung
ESC K = Zeilenduplizierung

Der PMODUS (Pagemodus), SMODUS (Scrollmodus) und KURSOR AUS sind beim MTX-Computer mit deutschem Zeichensatz nur mit vorheriger Eingabe von POKE 64325,1 (Englische Tastaturabfrage) zu erreichen, wobei dann die Tasten #, 'kleiner als' und = gelten.

Kurzform: DS.

EDIT (Zeilennummer)

Diese Anweisung bewirkt den Transfer der bezeichneten Zeile ins EDIT-Feld, wo diese Programmzeile editiert (korrigiert) werden kann.

Beispiel:

```
10 REM ABCDEFG

EDIT 10
```

Wird eine Zeilennummer geändert, erscheint auch der Inhalt der Zeile an der neuen Zeilennummer.

Eine CODE-Zeile läßt sich nur dann editieren, wenn der Assembler aktiviert und der Assembler-Editor angesprochen wird (siehe auch den Abschnitt über den Assembler).

Kurzform: E.

EDITOR (Variablenliste)

Der EDITOR gestattet dem Programm die freie Weiterverarbeitung von Daten, die direkt in einen vorbestimmten Sektor des Bildschirms eingegeben werden. Dieses Screen wird vom Fenster 0 bestimmt, dessen Parameter durch den CRVS-Befehl vorgegeben werden.

Beispiel:

```
10 CRVS 0,0,20,10,10,1,40
20 EDITOR A$
30 VS 5
40 PRINT A$
50 GOTO 20
```

Der EDITOR beläßt den aktuellen Bildschirm im VS-Modus (=0), so daß eine Anzeige über den gesamten Bereich für nachfolgende BASIC- oder andere Befehle nicht gegeben ist. Daher ist Zeile 30 eingefügt, die den Bildschirm wieder zurück in den Textmodus setzt.

Kurzform: EDITO.

ELSE

Siehe IF

Kurzform: EL.

EXP (Zahl)

EXP ist die Exponentialfunktion, dessen Wert sich aus der mathematisch definierten Zahl e mit dem Exponenten der angegebenen Zahl ergibt.

Beispiel:

```
EXP(1)=2.71828183
```

Kurzform: EX.

FOR (Kontrollvariable)=(Start) TO (Limit) STEP (Increment)

NEXT (Kontrollvariable)

Kontrollvariable ist in diesem Fall eine einfache numerische Variable.

Start, Limit und Increment sind numerische Ausdrücke. Wird STEP ausgelassen, geht der Computer von STEP 1 aus.
FOR und NEXT begrenzen die Programmschleife.

Beispiel:

```
10 FOR I=0 TO 10 STEP 1
20 PRINT I,I*I
30 NEXT I
```

Wird eine FOR-Anweisung im Programm angetroffen, wird (Start) der (Variablen) zugeordnet, ähnlich wie bei einer LET-Anweisung.

Das Programm wird nun fortgeführt, bis eine NEXT-Anweisung mit der gleichen Kontrollvariablen auftritt. An dieser Stelle wird das Increment hinzuaddiert und der nun geänderte Wert der Kontrollvariablen mit dem Limit (Grenze) verglichen. Ist der Wert der Kontrollvariablen nun höher als der des Limits, führt das Programm in der nächsten Zeile fort. Hat der Wert der Kontrollvariablen noch nicht den Wert des Limits überschritten, wird der weitere Programmablauf an die Zeile hinter der letzten FOR-Anweisung zurückgesetzt.

Das Increment kann auch negative Werte enthalten. Wird keine Kontrollvariable in der NEXT-Anweisung vorgefunden, nimmt der Computer eine geeignete Variable an.

Siehe auch NEXT.

Kurzformen: F., STE., N.

GENPAT p,n,d1,d2,d3,d4,d5,d6,d7,d8

Der GENPAT-Befehl wird dazu genutzt, sämtliche von BASIC gesteuerte Grafikzeichen und Zeichensätze zu generieren. Es gibt 5 Modi:

1. Um den ASCII-Zeichensatz umzustellen (CODES 32-127)
2. Um ein nicht-ASCII-Zeichen zu definieren (CODES 129-154)
3. Um Farben für jede Zeile eines Zeichens zu bestimmen.
   Dies bezieht sich nur auf die Zeichen mit den Codes 147-154.
4. Um ein 8*8 Pixel Spritemuster zu erzeugen.
5. Um jeden Quadranten eines 16*16 Pixelfeldes zu bestimmen.

Anwenderbestimmbare Zeichen haben den Codebereich 129-154.

Modus 1 erlaubt die Umbestimmung eines Standard-ASCII-Zeichens. Beachten Sie bitte, daß diese Zeichen am häufigsten vom Computer benutzt werden.

Modus 2 gestattet ebenfalls das Einsetzen eines eigenen Zeichensatzes, jedoch ohne die Standard-ACSII-Zeichen zu zerstören.

Modus 3 erlaubt eine weitergehende Gestaltung einiger dieser selbstdefinierbaren Zeichen, indem die Farben jeder Pixelreihe festlegbar sind.

Die Werte für INK und PAPER sind in der Farbtafel im Anhang aufgeführt, nur daß in diesem Fall sowohl die PAPER- als auch die INK-Farben in einem Parameter bestimmt werden. d1 bis d8 bestimmen beide Farben mit einer Zahl.

```
bit    0  1  2  3:  4  5  6  7
Wert   1  2  4  8: 16 32 64 128
         PAPER   :     INK
```

Parameter = 16 * INK + PAPER

z.B. rot (INK) auf blau (PAPER)

```
     = ROT : BLAU
     = ROT * 16 + PAPER
     =   9 * 16 + 4
     =   148
```

MODUS	P	N
1	0	ASCII-Code (32 - 127)
2	1	Anwenderdefiniert (Code 129 - 154)
3	2	Anwenderdefiniert für Farbe (Code 147 - 154)
4	3	Matrixnummer 8 x 8 Sprite
5	4	Matrixnummer 16 x 16 NW-Viertel
	5	Matrixnummer 16 x 16 SW-Viertel
	6	Matrixnummer 16 x 16 NO-Viertel
	7	Matrixnummer 16 x 16 SO-Viertel

Kurzform: GE.

GOSUB (Zeilennummer)

Führt einen Programmablauf ab der spezifizierten Zeilennummer durch, bis eine RETURN-Anweisung zur Rückkehr der regulären Programmfolge zwingt. Das Programm wird dann in der der GOSUB-Anweisung folgenden Zeile fortgesetzt.

Beispiel:

```
10 FOR I = 1 TO 10
20 GOSUB 100
30 NEXT
40 STOP
100 PRINT I,SIN(I),COS(I)
110 RETURN
```

Es können bis zu 34 GOSUB-Anweisungen ineinander verschachtelt werden. Bei Überschreitung dieser Anzahl erfolgt eine Fehlermeldung. Siehe auch RETURN.

Kurzform: GOS.

GOTO (Zeilennummer)

Der Programmablauf wird auf die bezeichnete Zeile übertragen.

Beispiel:

```
10 GOTO 40
20 PRINT "ZEILE 20"
30 STOP
40 PRINT "ZEILE 40"
```

Ist die angegebene GOTO-Zeile nicht vorhanden, erfolgt eine Fehlermeldung.

Kurzform: G.

GR$ x,y,b

GR$ interpretiert ein Bit-Muster im Grafik-Modus. Diese Funktion sollte dann eingesetzt werden, wenn eine grafische Abbildung als Grafik auf einen Matrixdrucker übertragen werden soll.

x und y sind Koordinaten des Bildschirms.

b gibt die Anzahl der vertikal zu übertragenden Pixel an. (Bei b=1 haben wir die POINT-Funktion).

Kurzform: GR.

IF (Boolscher Ausdruck) THEN (Anweisung) ELSE (Anweisung)

Die IF-Anweisung erlaubt die Verzweigung eines Programms abhängig vom Wahrheitsgehalt einer Aussage. Boolsche Ausdrücke sind solche Ausdrücke, die einen Wahrheitsgehalt beinhalten.

Beispiel: Y=2 AND X=3

(Anweisung) kann eine beliebige zulässige BASIC-Anweisung sein (einschließlich IF).

Ist der Boolsche Ausdruck tatsächlich als wahr erkannt, wird die Anweisung hinter THEN ausgeführt. Ist der Boolsche Ausdruck falsch, wird die Anweisung hinter ELSE ausgeführt. Ist der Boolsche Ausdruck falsch und ELSE nicht vorhanden, wird das Programm in der nächsten Zeile fortgeführt.

Beispiel:

```
10 INPUT "EINGABE J OR N";A$
20 IF A$="J" THEN GOTO 40 ELSE GOTO 100
30 GOTO 10
40 PRINT "JA"
50 STOP
100 PRINT "NEIN"
```

Siehe auch Boolsche Ausdrücke.

Kurzformen: T., EL.

INK (Farbe)

Wählt die INK-Farbe. INK ist jene Farbe, mit der Zeichen oder Grafik vor dem Hintergrund (PAPER) versehen sind, (Farbe) ist eine der 16 Zahlen der Farbskala (0-15).

Kurzform: I.

INKEY$

Diese Funktion liest während eines Programmablaufs die Tastatur und gibt eine Ziffer entsprechend der gedrückten Taste aus.

Beispiel:

```
10 IF INKEY$<>"" THEN GOTO 10
20 IF INKEY$="" THEN GOTO 20
30 PRINT INKEY$;:GOTO 10
```

Kurzform: INKE.

INPUT "STRING";(Variablenliste)

Die INPUT-Eingabe dient zur Dateneingabe an den Computer. Die Variablenliste ist eine Liste von Feld- oder numerischen Variablen, die durch Kommata getrennt werden.

Beispiel:

```
INPUT A,B,ABC$
```

Ist kein "STRING" vorhanden, erscheint ein Fragezeichen als Hinweis für eine INPUT-Aktivierung. Wird eine nicht numerische Eingabe für eine numerische Variable, oder nicht alle Werte eingegeben, erscheint ebenfalls ein Fragezeichen. Diese Daten müssen nochmals eingegeben werden. Ist ein "STRING" vorhanden, ersetzt dieser das Fragezeichen als Eingabehinweis. Dem "STRING" muß immer ein Semikolon folgen.

Beispiel:

```
INPUT "GEBEN SIE IHREN NAMEN EIN";N$
```

Kurzform: INP.

INT(Zahl)

Ergibt den ganzahligen Anteil einer Zahl.

Beispiel:

```
INT(2.2)=2
INT(-2.2)=-2
```

LEFT$(STRING,Zahl)

Der String auf die spezifizierte Anzahl von Stellen von links ab gekürzt.

Beispiel:

```
LEFT$("abcdef",3) ergibt "abc"
```

Kurzform: LEF.

LEN (STRING)

Ermittelt die Länge eines Strings.

Beachten Sie bitte, daß zwischen LEN und dem String ein Leerzeichen stehen muß.

Beispiel:

```
PRINT LEN ("ABC"+"DEF")              ;ergibt 6
```

LET Variable=Wert

Die LET-Anweisung ordnet der Variablen einen Wert zu.

Beispiel:

```
10 LET X=2
20 LET A$="abc"
```

Im ersten Beispiel erhält die Variable X den Wert 2 und im zweiten die Variable A$ den Wert "abc". Wert und Variable müssen der gleichen Gruppe angehören, d.h. Zahlen können nicht einer String-Variablen und umgekehrt zugeordnet werden.

Kurzform: LE.

LINE p,q,x,y

Zieht in einem Grafikscreen eine Linie von den Koordinaten p,q zu den Koordinaten x,y.

Beispiel:

```
10 VS 4: CLS
20 LINE 0,0,255,191
30 GOTO 30
```

Kurzform: LIN.

LIST Startzeile,Endzeile

LIST überträgt das Programm-Listing auf den Bilschirm. Es gibt, abhängig von der Anzahl der angegebenen Parameter, drei verschiedene Formate.

LIST listet das Gesamtprogramm.
LIST 100 listet das Programm ab Zeile 100.
LIST 100,200 listet das Programm zwischen den Zeilen 100 und 200.
(LIST 100,100 listet nur die Zeile 100.)

Siehe auch LLIST, AUTOSCROLL.

Kurzform: L.

LLIST

LLIST überträgt das Programmlisting auf den Drucker. Die verschiedenen Formate werden wie bei LIST gehandhabt.

Kurzform: LL.

LN(Zahl)

Ergibt den natürlichen Logarithmus des angegebenen Wertes.

LOAD "NAME"

Lädt das BASIC-Programm, das unter "NAME" abgespeichert worden ist, vom Kassettenrekorder in den Computer. Wenn kein Name angegeben wird (""), wird das nächste auf der Kassette befindliche Programm geladen.

Kurzform: LO.

LPRINT Ausdruck

LPRINT erfüllt genau jene Funktion wie PRINT, mit dem Unterschied, daß nicht der Bildschirm, sondern der Drucker angesteuert wird.

Siehe PRINT.

Kurzform: LP.

MANIPULATION von STRINGS

Beim MTX wird ein String als Zeichenfeld behandelt und zwar so, als ob bereits durch DIM Platz reserviert worden wäre.

Beispiel:

```
10 LET A$="AAA"

entspricht

10 DIM A$(64):LET A$="AAA"
```

Der MTX gestattet die Teilstringspezifizierung mit entweder einem Parameter mehr oder weniger als sonst erforderlich. Da ein String als eindimensionales Zeichenfeld betrachtet werden kann, würde die Angabe eines einzelnen Parameters ein bestimmtes Zeichen an dieser Position bestimmen.

Beispiel:

```
10 LET A$="ABCDEFG"
20 PRINT A$(3)
30 PRINT A$
40 PRINT A$(3,3)
```

Der Ausdruck von Zeile 20 ergibt den Buchstaben 'C'. Wird wie in Zeile 30 ein Parameter ausgelassen, ergibt dies den Ausdruck des gesamten Strings. Ist, wie in Zeile 40, ein weiterer Parameter vorhanden, wird ein Bereich des Strings ausgedruckt, der, beginnend mit der Position des ersten Parameters, so viele Stellen beinhaltet, wie der zweite Parameter angibt. Der Ausdruck von Zeile 40 ergibt daher 'CDE'. Diese Regeln können auf Zeichenfelder mit beliebiger Dimension übertragen werden.

MID$(String,X,Y)

Ergibt Y Zeichen beginnend bei der Stelle X des Strings.

Beispiel:

```
MID$("ABCD"+"EFGH",3,4)="CDEF"
```

Kurzform: MI.

MOD(X,Y)

Ergibt den Rest des Quotienten X/Y.

Beispiel:

MOD(10,7) = 10/7 = 1 Rest 3 = 3

MOD(X,Y) entspricht: X-INT(X/Y)*Y

Kurzform: MO.

MVSPR p,n,d

MVSPR ist ein Universalbefehl, der folgende 4 Funktionen enthält:

n	Bedeutung
1	Bewegung
2	Musterauswahl
4	Richtungsänderung
8	PLOT AT CENTRE

n ist dabei immer die Spritenummer und d richtet sich nach der/den spezifizierten Funktion/en.

Diese Funktionen können auch kombiniert werden, um komplexe Bewegungen mit einem Befehl auszuführen. Die jeweilige Tätigkeit wird von p bestimmt. Werden mehrere Tätigkeiten zeitgleich benötigt, müssen obige Werte nur addiert werden.

		Beisp. 1	Beisp. 2	Beisp. 3
BEWEGUNG	1	ja	ja	ja
MUSTERK.	2	nein	ja	nein
RICHTUNGSÄ.	4	nein	ja	ja
PLOT	8	ja	nein	ja
GESAMTWERT FÜR p		9	7	13

d ist etwas komplizierter bei kombinierten Funktionen anzuwenden, da der Bereich von d sich teilweise unterscheidet. Sollte der Wert von d nicht innerhalb des angewählten Bereichs liegen, können Fehler vorkommen.

BEWEGUNG (p=1) bringt das SPRITE einen Schritt in die von d angegebene Richtung. Die Schrittweite wird von CTLSPR 1 festgelegt und die Richtung muß im Bereich 0 - 8 liegen. 0 und 8 sind dabei identisch. Welcher Wert welche Richtung bedeutet, ist aus dem Diagramm unter Befehl 6 zu ersehen.

MUSTERKENNUNG (p=2) wechselt vom vorhandenen Spritemuster zum Muster d. Dieses Muster muß vorher von GENPAT generiert worden sein.

RICHTUNGSÄNDERUNG (p=4) übernimmt den anstehenden Richtungsvektor und überträgt ihn auf die neue Richtung d.

PLOT im Mittelpunkt bewirkt die Abbildung eines Punktes in der Mitte eines durch n gekennzeichneten Sprites. d hat hier keine Auswirkung.

Kurzform: M.

NEW

Dieser Befehl bringt alle Systemvariablen in den Zustand, den sie direkt nach dem Einschalten bekommen, damit ein neues Programm eingegeben werden kann.

NEXT Kontrollvariable

NEXT bezeichnet das Ende einer mit FOR begonnenen Programmschleife.

Wenn die Kontrollvariable angegeben wurde, wird das NEXT auf die entsprechende FOR-Anweisung bezogen und somit werden alle anderen vorher eventuell vorhandenen FOR-Befehle ausgelassen. Wenn die Kontrollvariable nicht angegeben wurde, nimmt der Computer an, daß das NEXT sich auf die letzte FOR-Anweisung bezieht.

Siehe auch unter FOR.

Kurzform: N.

NODDY

Programmsteuerung wird auf den NODDY-Editor übertragen. Auf der untersten Zeile sollte nun 'Noddy>' erscheinen.

NODDY-Befehle:

A	Führt das nächste Programm des Programmstapels aus.
B label	Springt in der Programmausführung zu label.
D page.	Display. Zeigt eine Noddy-Seite (auf VS 5) an.
E	Enter. Eingabe vom Bediener (auf VS 7).
G page,(label)	Goto. Geht zur angegebenen Programmseite und dort zu label, falls der Label spezifiziert ist.
I Vergl.,label	If Input. Falls Eingabe=Vergl., erfolgt ein Sprung zu label.
L page	LList. Druckt eine Noddy-Seite auf dem Drucker aus.
O	Omit. Entfernt eine Programmseite vom Stapel, ohne sie auszuführen.
P	Pause. Bewirkt eine Pause im Programmablauf.
R	Return. Rückkehr ins BASIC.
S s1,s2,...	Stack. Setzt Programmseiten auf den Stapel.

Der NODDY-Interpreter erhält Eingaben über Screen (VS) 7 und zeigt Ergebnisse über Screen (VS) 5 an.

Diese Fenster werden in der Regel den Gesamtbildschirm (für die Anzeige) und die unterste Zeile (für die Eingabe) ausmachen, können aber mit Hilfe des CRVS-Befehls umdefiniert werden.

Beispiel:

```
10        CRVS 5,0,10,10,20,5,40
20        CRVS 7,0,10,16,20,1,40
30        NODDY
RUN
```

Es werden zwei kleinere Fenster für NODDY eingerichtet, ansonsten aber nichts verändert.

Siehe auch den Abschnitt NODDY vom Handbuch.

Kurzform: NODD.

NOT

Siehe Boolsche Ausdrücke

ON GOTO und ON GOSUB

Der ON-Befehl wird benutzt, wenn GOTO oder GOSUB eine Programmverzweigung oder eine Unterprogrammausführung variablenbedingt durchführen sollen.

Beispiel:

```
10 LET X=2
20 ON X GOTO 100,200,300,400,500
```

An dieser Stelle würde das Programm, je nach tatsächlichem Wert von X, zur Zeile 100, 200 oder 300 etc. verzweigen.

Bei X=0 erfolgt die Verzweigung nach 100, bei X=1 nach 200 etc.

Ähnliches gilt für Unterprogramme wie im nachfolgenden Beispiel:

```
10 INPUT X
20 ON X GOSUB 100,200,300,400,500
30 REM Programmablauf fährt hier weiter
```

Kurzform: O.

OR

Siehe Boolsche Ausdrücke

OUT Port,Wert

Gibt den angegebenen Wert auf den angegebenen Port aus.

Siehe auch den technischen Anhang (System Block Diagram)

PANEL

Schaltet zur PANEL-Anzeige.

PANEL-Hilfsbefehle:

B(asic)	Exit?	Eingabe Y für eine Rückkehr in BASIC.
C(lear)		Löscht die List-Anzeige.
Display	Hex	Zeigt einen Speicherbereich um den Hexwert herum an.
G(o)	Hex1 to Hex2	Programmablauf von der Adresse Hex1 bis Adresse Hex2.
I		Wechsel der Anzeige von ASCII/HEX und umgekehrt.
L(ist)	Hex	Listing ab Hexadresse.
M(ove)	Hex1 Ende-Hex2 zu Hex3	Überträgt einen Programmabschnitt (Hex1 bis Hex2) zu Hex3
R(egister)	Hex	Eintrag des Wertes Hex in das durch den Register-Kursor bezeichnete Register.
S(ingle Step)		Einzelschrittabarbeitung; führt den durch den Programmzähler indizierten Befehl aus
T(race)		Wie S, aber Call-Befehle werden als Gesamtbefehl interpretiert.
X		Zeigt komplementären Registersatz an.
-		Setzt den Display-Kursor zurück.
(RET)		Setzt den Display-Kursor vor.
^ (Kursor hoch)		Setzt den Display-Kursor höher.
v (Kursor runter)		Setzt den Display-Kursor tiefer.

Beispiel einer PANEL-Anzeige:

```
0100      JP Z,#E8            AF >0000 C3 (Direkt über dem Regi-
0103      LD A,#0C3           BC  0000 C3 sterpaar AF werden die
0105      LD (#0050),A        DE  0000 C3 eventuell gesetzten
0108      LD (#0053),A        HL  0000 C3 Flags angezeigt. Die
010B      LD HL,#2B09         IX  0000 C3 Ein-Byte-Hexzahlen
010E      LD (#0051),HL       IY  0000 C3 hinter den Registerin-
0111      LD HL,#3B04         SP  0100 C3 halten geben den In-
0114      LD (#0054),HL       PC  0100 C3 halt der Adresse an,
0117      LD A,#0E1                       die durch den Regi-
0119      LD (#0028),A                    sterwert spezifiziert
011C      LD A,#7E                        wird. Dadurch läßt
011E      LD (#0029),A                    sich beim Single-Step-
0121      LD A,#FE                        ping teilweise schon
0123      LD (#0020),A                    vorher erkennen, was
                                          geschehen wird, z.B.
JP Z,#E8                                  bei dem Befehl LD
                                          A,(HL).)
     00F0: 21 2A 87 28 A9 A9 A7 2F
     00F8: D0 02 5C 4D A7 2F 9D 12
     0100: C3>E8 02 3E C3 32 50 00
     0108: 32 53 00 21 09 2B 22 51
     0110: 21 2A 87 28 A9 A9 A7 2F
     0118: D0 02 5C 4D A7 2F 9D 12
```

PAPER Farbe

Wählt die PAPER-Farbe. Farbe ist eine Zahl von 0 bis 15 aus der Farbtafel im Anhang.

Kurzform: PA.

PAUSE Zahl

Das Programm 'pausiert' für eine von der angegebenen Zahl abhängige Zeit. Diese Zeit kann nicht genau spezifiziert werden, da diese von den sonstigen Arbeiten des Rechners abhängt.

Kurzform: PAU.

PEEK (Adresse)

Gibt den Inhalt der angegebenen Adresse wieder. Beachten Sie bitte, daß der MTX-Speicher in Blöcken a 32k aufgeteilt ist. Sie müssen also sicherstellen, daß der richtige Block angesprochen wird. Die obersten 16k des Speichers sind für alle Blöcke zugänglich.

Kurzform: PE.

PHI(Winkel)

Jede PHI-Anweisung bewirkt eine Richtungsänderung der durch ANGLE spezifizierten Richtung. Beispiel:

```
10 VS 4
20 CLS
30 ANGLE 0
40 FOR I=1 TO 20
50 PLOT 100,100
60 PHI .1
70 DRAW 50
80 NEXT
90 GOTO 90
```

Kurzform: PH.

PI

Der MTX hat einen recht genauen PI-Wert gespeichert, der als numerische Variable betrachtet und dementsprechend eingesetzt werden kann.

Beispiel:

```
PRINT PI
PRINT COS(PI/2)
```

PLOD "String"

PLOD ist die BASIC-Anweisung für die Ausführung eines NODDY-Programms. "String" kennzeichnet die Noddy-Programmseite, die aufgerufen werden soll.

Beispiel:

```
10 PLOD "PROG1"
```

Damit aktivieren wir ein NODDY-Programm namens PROG1.

PLOD kann innerhalb eines BASIC-Programms wirken und die Programmsteuerung kann beliebig zwischen BASIC und NODDY wechseln. Gibt ein NODDY-Programm die Steuerung an BASIC zurück, wird das BASIC-Programm mit der der PLOD-Anweisung folgenden Zeile fortgesetzt.

Kurzform: PL.

PLOT x,y

Ein Punkt wird auf die Koordinaten x,y im Grafik-Modus gesetzt.

Siehe COLOUR und ATTR.

POKE Adresse,Wert

Die POKE-Anweisung überträgt den angegebenen Wert in die angegebene Speicheradresse.

Sind mehr als 32k Speicherplatz vorhanden, kann das BASIC-Programm über mehrere Speicherblöcke verteilt sein (siehe SPEICHERPLATZORGANISATION-RAM). Die obersten 16k werden immer zugänglich sein. Das POKEn in andere Bereiche erfordert zusätzliche Maßnahmen.

Beispiel:

```
POKE 50000,100
```

Der Wert 100 wird in der Speicheradresse 50000 abgelegt.

Kurzform: PO.

PRINT Alphanumerischer Ausdruck

Mit der PRINT-Anweisung werden Daten und Texte auf den Bildschirm übertragen.

Beispiel:

```
PRINT "SIN von 8 = ";SIN(8)
```

Der PRINT-Anweisung folgt eine Reihe von Ausdrücken, die durch Komma oder Semikolon getrennt werden. Die Ausdrücke können mathematischer Art, Strings oder Zahlen sein. Eine beliebige Anzahl von Kommas können zur Tabulation benutzt werden.

Ein Semikolon setzt den Kursor unmittelbar hinter die letzte ausgedruckte Information.

Kurzform: P.

RAND Zahl

Diese Anweisung initialisiert den Zufallsgenerator. Wird keine Ausgangszahl spezifiziert, greift der Computer auf eine Zufallszahl zurück.

Beispiel:

```
10 RAND 1000
20 FOR I=1 TO 100
30 PRINT INT(RND*50),
40 NEXT I
```

Siehe auch RND

Kurzform: RA.

READ Variablenliste

Beispiel:

```
10 READ A,B,A$,B$(8)
20 PRINT A,B,A$,B$
30 DATA 1,2,AA,BB
```

Werte der DATA-Anweisung werden sequentiell den READ-Variablen zugeordnet und weiterverarbeitet. Wird einer numerischen Variablen eine falsche Zuordnung gegeben, erfolgt eine Fehlermeldung.

DATA-Anweisungen können an beliebigen Stellen des Programms vorkommen. Sie werden in der angegebenen Reihenfolge entweder ab Programmbeginn oder entsprechend einer RESTORE-Definition verarbeitet.
Siehe auch DATA und RESTORE

Kurzform: REA.

REM Beliebig

Die REM-Anweisung erlaubt das Einfügen beliebiger Informationen in einem Programm. Sie dient zur Orientierung für den Programmierer (Hinweise, Warnungen etc.). REM-Zeilen haben keine Auswirkung auf das Programm.

Kurzform: R.

RESET

Ein RESET wird durch gleichzeitiges Drücken der beiden unbeschrifteten Tasten rechts und links neben der Leertaste erzeugt. Dadurch werden sämtliche Systemvariablen auf ihren Grundzustand gesetzt und der Computer ist für eine neue Eingabe bereit. Ein eventuell vorhandenes BASIC-Programm kann durch die Eingabe von POKE 64167,1 teilweise wieder angezeigt werden.

RESTORE Zeilennr.

RESTORE weist den Computer an, welche DATA-Anweisung nach welcher Zeilennummer als nächste von READ gelesen werden soll. Dies ist besonders hilfreich, wenn DATA-Informationen mehrfach, von verschiedenen Programmabschnitten gelesen werden sollen.

Beispiel:

```
10 DIM A(20)
20 FOR I= 1 TO 10
30 READ A
40 PRINT A
50 NEXT I
60 RESTORE 100
70 FOR I= 1 TO 10
80 READ A(I)
90 NEXT I
100 DATA 1,2,3,4,5,6,7,8,9,0
```

Siehe auch DATA,READ

Kurzform: RES.

RETURN

Überträgt den Programmablauf auf die der letzten GOSUB-Anweisung folgenden Zeile. Wurde zuvor keine GOSUB-Anweisung benutzt, erfolgt ein Fehler.

Siehe GOSUB

Kurzform: RET.

RIGHT$("String",Zahl)

Der benannte String wird gekürzt, und zwar hinter der durch die Zahl spezifizierte Stelle, wobei von rechts nach links gezählt wird.

Beispiel:

```
RIGHT$("ABCDEFG",3)="EFG"
```

Kurzform: RI.

RND

RND generiert eine Pseudo-Zufallszahl.

Siehe RAND

Kurzform: RN.

ROM ROM-Nummer

Überträgt die Programmsteuerung auf eine andere ROM-Einheit, z.B. auf PASCAL oder FORTH als ROM-Module.

Eine ROM-Anweisung sollte nur dann erfolgen, wenn das entsprechende Modul tatsächlich angeschlossen ist. Die jeweils zutreffenden ROM-Adressen sind den Modulen beigefügt.

Kurzform: RO.

RUN

RUN ist die Anweisung, mit der ein Programmablauf von der ersten Programmzeile an abläuft. Alle zuvor benutzten Variablen werden gelöscht.

Ein Programm kann auch mit einer GOTO-Anweisung gestartet werden. Hierbei bleiben Variablenwerte unverändert.

Kurzform: RU.

SAVE "Name"

Das im Speicher befindliche Programm wird unter der Bezeichnung "Name" auf Kassette gespeichert.

Kurzform: SA.

SBUF Zahl

Dieser Befehl reserviert den entsprechenden Speicherplatz (Zahl) in einem Sound Buffer für den SOUND-Befehl.

Beispiel:

```
10 SBUF 8
```

Hiermit wird Speicherplatz für 8 Sound Data Blöcke für jeden der drei Tonkanäle und den Rauschkanal reserviert. Jeder Block benötigt 10 Bytes. Obiger Befehl benötigt daher 320 Bytes. (8 * 4 Kanäle * 10 Blocklängen).

Der Buffer wird am oberen Speicherende unterhalb der Systemvariablen angelegt.

Kurzform: SB.

SGN(Zahl)

Produziert einen Wert abhängig vom Vorzeichen der angegebenen Zahl.

Eine positive Zahl ergibt +1
Eine negative Zahl ergibt -1
Eine Null ergibt 0

Beispiel:

```
SGN (-2.5)= -1
SGN ( 2.5)=  1
SGN (   0)=  0
```

Kurzform: SG.

SIN(Winkelwert in rad)

Ergibt den Sinus des in rad angegebenen Winkels.

Kurzform: SI.

SOUND Parameterliste

Der Effekt des Tonbefehls wird von der Anzahl der angegebenen Parameter bestimmt.

SOUND Kanal,Frequenz,Volumen (3 Parameter)

Kanäle = 0,1,2 oder 3
0-2 sind die eigentlichen Tonkanäle
3 ist der Rauschkanal

SOUND Kanal,Frequenz,Vol.,Frequ.Incr.,Vol.Incr.,Dauer,Modus
(7 Parameter)
Die ersten 3 Parameter gleichen dem ersten Beispiel.

Nach je 1/64 Sek. erhöht der Computer die Frequenz und die Lautstärke (Vol.) um die vom Increment (Incr.) vorgegebenen Werte und zwar so lange, wie der Ton durch den Dauer-Parameter anhält. Die Zeiteinheit für "Dauer" ist ebenfalls 1/64 Sek., d.h., daß der Wert 64 = 1 Sekunde Ton bedeutet.

Der Modus kann entweder 0 oder 1 sein.

Bei Modus 0 werden die Frequenz- und Lautstärkenparameter des Sound Buffer vom nächsten SOUND-Befehl so übernommen, wie sie vorgefunden wurden.

Bei Modus 1 werden die spezifizierten Frequenz- und Volumewerte in den Sound Buffer übertragen, um den entsprechenden Tonkanal zu initialisieren.

Kurzform: SO.

SPK$ (Screen Peek)

Ermittelt den ASCII-Code des Zeichens auf der Kursorposition und incrementiert diese Position.

Beispiel:

```
10 CLS
20 FOR I=32 TO 64
30 PRINT CHR$(I);
40 NEXT
50 CSR 0,0
60 FOR I=32 TO 64
70 LET A$(I)=SPK$
80 NEXT
90 PRINT
100 PRINT A$
```

Die Zeichen werden vom Bildschirm gelesen, in einem Feld gespeichert und dann noch einmal ausgedruckt.

Nachfolgend noch ein Beispielprogramm für den direkten Ausdruck eines Textscreens auf einem Panasonic Drucker:

```
10 REM ** SCREEN COPY TEXT MODUS
20 LPRINT CHR$(27);"Q";CHR$(39);:REM Rechter Rand des
Druckers wird auf Spalte 39 gesetzt!
30 DIM A$(780)
50 CSR 0,0
60 FOR I=1 TO 780
70 LET A$(I)=SPK$
80 NEXT
90 LPRINT A$
```

Kurzform: SPK.

SPRITE n,pat,xp,yp,xs,ys,Farbe

n bestimmt die Spritenummer 1-32

pat bestimmt das Muster 0-127 (Größe 0)
0- 31 (Größe 1)

xp bestimmt die x-Lage der Mitte des Sprites

yp bestimmt die y-Lage der Mitte des Sprites (Skalen -4095 bis +4095)
0,0 ist die Lage links unten im Bildschirm, d.h. wie bei PLOT.

xs bestimmt die horizontale Geschwindigkeit (Bereich -128 bis +127), wobei eine 1 das Sprite 1/8 Pixel je Zyklus gemäß CTLSPR 0 bewegt.

ys bestimmt die vertikale Geschwindigkeit (Bereich -128 bis +127).

Farbe bestimmt die Spritefarbe (0-15)

Anm. Spritekoordinaten werden absolut berechnet. Fenster-Dimensionierungen (VS) werden ebensowenig beachtet wie ein falscher Modus.

Kurzform: S.

SQR(Zahl)

Ergibt die Quadratwurzel von Zahl.

Kurzform: SQ.

STEP

Siehe FOR

Kurzform: STE.

STOP

Hält den Programmablauf an. Sofern das Programm nicht zwischenzeitlich verändert wurde, bewirkt CONT die Programmfortsetzung.

Beispiel:

```
10 REM Lange Pause Programm (1 Min 30 Sek.)
20 CLOCK "000000"
30 PRINT "START"
40 IF TIME$="000130" THEN STOP
50 GOTO 40
```

Kurzform: ST.

STR$(Zahl)

Wandelt (Zahl) in einen String mit einem vorangehenden Leerzeichen um.

Beispiel:

```
STR$(2+2)=" 4"
```

Beachten Sie bitte, daß STR$ einen String ergibt und daher nicht in numerischen Ausdrücken verwendet werden kann.

Kurzform: STR.

TAN(Winkelwert in rad)

Ergibt den Tangens des angegebenen Winkels.

Kurzform: TA.

THEN

Siehe IF

Kurzform: T.

TIME$

Ergibt die Zeit der Echtzeituhr nach folgendem Schema:

Std Std Min Min Sek Sek

Std Std gibt die Anzahl der verstrichenen Stunden seit Initialisierung durch die CLOCK-Anweisung an. Der Stundenzähler geht bis 99, um dann bei Null neu anzufangen.

Beispiel:

```
10 CLOCK "000000"
20 CSR 10,10
30 PRINT TIME$
40 GOTO 20
```

Siehe CLOCK

Kurzform: TI.

TO

Siehe FOR

USR(Adresse)

USR überträgt die Programmsteuerung direkt auf die angegebene Speicheradresse. Dies ist die übliche Art, um Maschinencode vom BASIC aus aufzurufen. Da der MTX-Assembler bereits diese Funktion in BASIC einbindet, entfällt USR für die meisten Anwendungen.

Bei der Rückkehr ins BASIC ist der USR-Wert der des BC-Registerpaars.

Wenn Sie folgendes Programm assemblieren und mit USR aufrufen, wird eine '100' ausgedruckt.

```
        10 CODE

8007    LD BC,100
800A    RET

        Symbols:

        20 PRINT USR (32775)
```

32775 ist der Dezimalwert von Hex 8007.

VAL("String")

Ergibt den numerischen Wert von String.

Kurzform: VA.

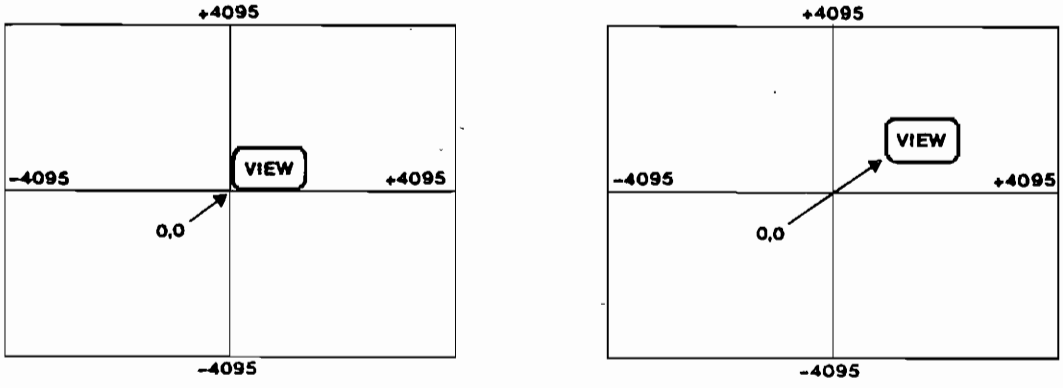
VERIFY "Name"

Verifiziert ein auf Band befindliches Programm mit dem im Arbeitsspeicher. Leider befindet sich hier ein Fehler im ROM, der ein ordnungsgemäß abgespeichertes Programm eventuell als falsch erkennen kann, obwohl es wieder einladbar ist. Dieser Fehler läßt sich unter Umständen durch ein Vorab-VERIFY eines anderen Programms, gefolgt von dem richtigen VERIFY umgehen.

Kurzform: VE.

VIEW Richtung,Entfernung

Im Grafik-Modus kann der Bildschirm als ein Fenster auf den Sprite-Ebenen betrachtet werden. Dieses Fenster liegt gewöhnlich wie folgt.



Der VIEW-Befehl positioniert das Fenster innerhalb der in Bild 2 angegebenen Grenze ohne die Lage der Sprites zu ändern.

Richtung = 0 - 7
Entfernung = 0 - 255

Kurzform: VI.

VS n

Dieser Befehl wählt ein Fenster (Virtual Screen, VS) aus den vom Befehl CRVS generierten Fenstern aus. Der Computer schaltet automatisch auf den vorbestimmten Bild-Modus (Grafik oder Text).

SOFTWARE-Anhänge

ANHANG 1

ASCII-CODES

ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC
NUL	00	0	/	2F	47		5E	94
SOH	01	1	0	30	48		5F	95
STX	02	2	1	31	49		60	96
ETX	03	3	2	32	50	a	61	97
EOT	04	4	3	33	51	b	62	98
ENQ	05	5	4	34	52	c	63	99
ACK	06	6	5	35	53	d	64	100
BEL	07	7	6	36	54	e	65	101
BS	08	8	7	37	55	f	66	102
HT	09	9	8	38	56	g	67	103
LF	0A	10	9	39	57	h	68	104
VT	0B	11	:	3A	58	i	69	105
FF	0C	12	;	3B	59	j	6A	106
CR	0D	13	<	3C	60	k	6B	107
SO	0E	14	=	3D	61	l	6C	108
SI	0F	15	>	3E	62	m	6D	109
DLE	10	16	?	3F	63	n	6E	110
DC1	11	17	@	40	64	o	6F	111
DC2	12	18	A	41	65	p	70	112
DC3	13	19	B	42	66	q	71	113
DC4	14	20	C	43	67	r	72	114
NAK	15	21	D	44	68	s	73	115
SYN	16	22	E	45	69	t	74	116
ETB	17	23	F	46	70	u	75	117
CAN	18	24	G	47	71	v	76	118
EM	19	25	H	48	72	w	77	119
SUB	1A	26	I	49	73	x	78	120
ESC	1B	27	J	4A	74	y	79	121
FS	1C	28	K	4B	75	z	7A	122
GS	1D	29	L	4C	76		7B	123
RS	1E	30	M	4D	77		7C	124
US	1F	31	N	4E	78		7D	125
space	20	32	O	4F	79		7E	126
!	21	33	P	50	80	DEL	7F	127
"	22	34	Q	51	81			
#	23	35	R	52	82			
$	24	36	S	53	83			
%	25	37	T	54	84			
&	26	38	U	55	85			
'	27	39	V	56	86			
(	28	40	W	57	87			
)	29	41	X	58	88			
*	2A	42	Y	59	89			
+	2B	43	Z	5A	90			
,	2C	44		5B	91			
-	2D	45		5C	92			
.	2E	46		5D	93			

ANHANG 2

EINIGE NÜTZLICHE CONTROL- UND ESCAPE-KOMBINATIONEN

KONTROLL-KOMBINATIONEN

CTL	Dn	Bestimmt die Hintergrundfarbe (n)
CTL	E	Löscht bis Zeilenende
CTL	Fn	Bestimmt die Vordergrundfarbe (n)
CTL	G	Glocke
CTL	H	Rückstellung, Kursor links
CTL	I	Tabuliert den nächsten Block mit 8 Spalten
CTL	J	Zeilenvorschub, Kursor nach unten (LF)
CTL	K	Kursor nach oben
CTL	L	Löscht den Bildschirm, Kursor zur Grundstellung
CTL	M	Zeilenvorschub, Cursor ganz nach links (CR)
CTL	W	Tab zurück
CTL	Y	Kursor vorwärts
CTL	Z	Kursor zur Grundstellung
CTL	(Eckige Klammer zu)	Page-Modus
CTL	(Schrägstr.v. links o. nach rechts u.)	Scroll-Modus
CTL	^	Kursor ein
CTL	_	Kursor aus

ESCAPEKOMBINATIONEN

ESC	S	Tastaturblockade
ESC	B0	Amerikanischer Zeichensatz
ESC	B1	Englischer Zeichensatz
ESC	B2	Französicher Zeichensatz
ESC	B3	Deutscher Zeichensatz
ESC	B4	Schwedischer Zeichensatz
ESC	B5	Spanischer Zeichensatz
ESC	I	Fügt eine Leerzeile an der Kursorstellung ein
ESC	J	Löscht augenblicklich Kursorzeile
ESC	K	Dupliziert eine Zeile
ESC	Xc	Simuliert CONTROL-Zeichen c

ANHANG 3

FEHLERMELDUNGEN

Params	Falsche oder unvollständige Parameterangabe für einen Befehl oder eine Funktion.
Mistake	Ein Fehler wurde begangen, der sich aus dem Kontext erklären lassen müßte.
A	Ein Punkt befindet sich außerhalb des Fensters (VS).
SE.A	Fenstertyp nicht in der Tabelle vorhanden.
SE.B	Nicht zulässige ESC-Kombination.
SE.C	Befehl für dieses Gerät nicht zulässig.
SE.D	Das angewählte Fenster (VS) ist nicht definiert worden.
SE.E	Unzulässiger UDG/UDG Typ.
Symbol?	Ein Symbol fehlt (z.B. "=", "TO", "THEN", ",", etc.)
Not numeric	Eine Zahl wird erwartet.
Boolean?	Ein Wahrheitswert wird erwartet.
Mismatch	Unzulässige Relation zwischen verschiedenen Wertetypen.
BK	Bandfehler bei LOAD oder SAVE.
No data	Keine DATA-Zeile oder keine Seite bei NODDY.
Overflow	Zahl ist zu groß.
Div/0	Division durch 0.
Out of range	Zahl befindet sich außerhalb eines zulässigen Bereichs.
No space	Speicherplatzmangel für: ein Feld, ein Programm, Verknüpfungen, eine komplexe Berechnung.
Subscript	Eine Index ist außerhalb des zulässigen Wertes oder es gibt zu viele.
Gosub	Es gibt zu viele Gosubs (mehr als 34).
Undefined	Eine nicht vorhandene Variable wird benutzt.

Array exists	Ein Feld dieser Art gibt es schon.
NO FOR	Es existiert kein FOR für ein vorhandenes NEXT.
No call	Es gibt kein entsprechendes GOSUB für ein RETURN.
No line	Es wird auf eine nicht vorhandene Zeile Bezug genommen.

ANHANG 4

DAS NUMERISCHE TASTENFELD

Das numerische Tastenfeld wurde für Anwenderprogramme entwickelt. Beachten Sie bitte die Break-Taste oben rechts. Sie können entscheiden, ob diese Taste aktiviert werden soll oder nicht. Um das numerische Tastenfeld anzuwenden, müssen Sie die Shift-Taste und eine der Zahlen drücken. In diesem Modus wird die Break/9-Taste eine 9 und keinen Programmabbruch ergeben.

Es gibt aber auch eine Sperre für diese Taste. Die Sperre besteht aus einem Bit im Keyboard Flag. Ist dieses Bit gesetzt, können die Zahlen ohne Shift benutzt werden; Break jedoch wird hier noch aus Sicherheitsgründen die 9 übersteuern. Um die 9 jetzt noch ohne Shift zu nutzen, muß das Break-Bit der Interrupt Flags INTFFF ausgeschaltet werden.

```
10        POKE 64145,132
20        POKE 64862,13
30        PRINT INKEY$:GOTO 30
```

ANHANG 5

SYSTEM VARIABLEN

FA52	CTRBADR	Kontrollpuffer für Sound
FA7A	STKTOP	Oberes Ende des Stacks
FA7A	LSTPG	Anzahl von 32k Seiten
FA7B	VARNAM	Anfang der Variablennamen
FA7D	VALBOT	Anfang der Variablenwerte
FA7F	CALCBOT	Anfang des Kalkulatorstacks
FA81	CALCST	Oberes Ende des Kalkulatorstacks
FA83	KBDBUF	Adresse des Keyboard Puffers
FA85		Syntax der User-Routine
FA89	USER	BASIC-User-Sprung
FA8C		
FA8F	IOPL	Liste der Peripherie
FA90	REALBY	Panel Breakpoint
FA91	KBFLAG	Siehe weiter hinten
FA92	STKLIM	Oberes Ende des freien Speicherplatzes
FA94	SYSTOP	Oberes Ende der zu SAVEnden Variablen
FA96	SSTACK	Adresse des Maschinenstacks
FA98	USERINT	Siehe weiter hinten
FA9B	NODLOC	
FA9E	FEXPAND	Panel Erweiterung
FAA1	USERNOD	Noddy Erweiterung
FAA4	NBTOP	Oberes Ende von Noddy
FAA6	NBTPG	" " " "
FAA7	BASTOP	Oberes Ende der augenblicklichen BASIC-Seite
FAA9	BASTPG	
FAAA	BASBOT	Anfang von BASIC
FAAC	BASTPO	Oberes Ende jeder BASIC-Seite
FACC	ARRTOP	Oberes Ende jedes Feldes
FACF	BASELIN	
FAD1	BASLNP	
FAD2	PAGE	Augenblickliche Seiten-Konfiguration
FAD3	CRNTPG	Augenblickliche BASIC-Seite
FAD4	PGN1	
FAD5	PGN2	
FAD6	PGTOP	
FAD8	GOSTACK	GOSUB-Stack
FB41	GOPTR	
FB43	GOSNUM	Anzahl der verschachtelten GOSUBs
FB45	CTYSLT	Tastenfeld-Konfiguration
FB46	DATAAD	Data-Zeiger
FB48	DATAPG	
FB49	DESAVE	

** Bis hierhin werden die Systemvariablen bei jedem SAVE mit auf Kassette gespeichert.

FB4B	START	Keyboard-Puffer
FD48	STACK	Maschinenstack
FD48	SETCALL	
FD4B	RICHJL	
FD4E	USRRST	Restart 38
FD51	USERIO	Siehe weiter hinten
FD54	USERROR	Error-Anzeiger
FD57	CLOCK	Real Time Clock
FD5E	INTFFF	Siehe weiter hinten
FD5F	CASBAUD	Kassettenbaudrate
FD60	MIDVAL	
FD61	RETSAVE	Anfangsadresse für automatisches Laden
FD65	VAZERO	
FD67	VERIF	
FD68	TYPE	
FD69	CONTFLG	
FD6A	CONTAD	Adresse der Zeile nach
FD6C	CONTPG	STOP oder BREAK
FD6D	ASTACK	
FD6F	TMPHL	
FD71	TMPA	
FD73	STACCT	
FD75	PRORPL	Siehe weiter hinten
FD76	IOPR	Siehe weiter hinten
FD77	AUTOIN	Increment für automatische Zeilennumerierung
FD79	AUTOST	
FD7B	AUTOCT	
FD7C	LASTKY	Letzte gedrückte Taste
FD7D	LASTASC	ASCII-Code der letzten gelesenen Taste
FD7E	LASTDR	
FD7F	RNSEED	
FD81	BREAK	
FD82	COMMAND	Adresse des ersten auszuführenden Befehls
FD84	ERRPOS	Position des Syntax Errors
FD86	FLAGS1	
FD87	ITYPE	Verwendet vom Assembler und Panel
FD89	MAFD	
FD8B	MBCD	
FD8D	MDED	
FD8F	MHLD	
FD91	MAF	Verschiedene
FD93	MBC	kurzfristige
FD95	MDE	Speicheradressen
FD97	MHL	zum
FD99	MIX	Festhalten
FD9B	MIY	von
FD9D	MSP	Z80
FD9F	MPC	Registern
FDA1	MEMPOINT	
FDA3	WCHJUMP	
FDA5	POINTERR	
FDA6	DADD	
FDA8	ASBYTE	
FDA8	INDEX	
FDAA	DBYTE	

FDAC	LINKER	
FDAD	EDIT	
FDAE	LENGTH	
FDAF	DETYPE	
FDBO	DTYPE	
FDB1	DISAD	
FDB3	DPROG	
FDB5	LABTABL	
FDB7	APROG	
FDB9	ENDTAB	
FDBB	COMMENT	
FDBC	COMAD	
FDBE	ADLABEL	
FDC1	INDEXLAB	
FDC3	DATALAB	
FDC6	DBLABEL	
FDC7	BASEM	
FDC9	CURLAB	
FDCC	ACC1	Verwendet für mathematische Funktionen
FDF2	INTTAB	NODE Interrupt Tabelle
FE02	GASH	Verwendet für Sound
FE04	TEMP	
FE14	CHAN	
FE16	FREQ	
FE18	VOL	
FE1A	WKAREA	VS Arbeitsbereich
FE3F	BSSTR	
FE4B	SPEED	Sprite Geschwindigkeit
FE4C	SPBASE	
FE4D	MVDIST	Move Distance
FE4E	NOSPR	Spriteanzahl
FE4F	DLSPNO	Anzahl der kreisenden Sprites
FE50	PLSPNO	Anzahl der PLOT-Sprites
FE51	MVNO	
FE52	DELSPR	
FE53	VCOUNT	Zähler für Kursor-Flash
FE54	VDPSTS	Kopie der VDP Statusregister
FE55	SPRTBL	Kontrollpuffer für Sprites
FF55	SMBYTE	Konfiguration der Spritegrüßen
FF56	LENLO	
FF57	LENHI	
FF58	VINTFG	Sprite Interrupt Flag: Falls 0, kann ohne Gefahr zum Screen geschrieben werden
FF59	CHPTR	Zeichensatzzeiger

Virtuelle Screens Kontrollen - siehe weiter hinten

FF5D	SCRNO	FFB7	SCRN6
FF6C	SCRN1	FFC6	SCRN7
FF7B	SCRN2	FFD5	VS TYPE TABLE - Screen Unterfunktionen
FF8A	SCRN3	FFED	Virtuelles Screen
FF99	SCRN4	FFEE	OVERLAY
FFA8	SCRN5		

VIRTUELLE SCREENS - Byte Format für jedes Screen

Byte	Inhalt
1	Screen Art, Auto Scroll, Kursor Flash, Page Modus
2	Aktuelle Print-Position in dem Screen
3	Zweites Byte der Print-Position
4	Absolute obere linke Ecke
5	Zweites Byte der oberen linken Ecke
6	Größe der Screen in Zeichen
7	Zweites Byte der Screengröße
8	Zeilenbreite des Physical Scren
9	Hält Kursorzeichen
10	Farbe für Umrandung, Paper und Ink
11	Print Farben: Ink, Paper; Print Attribute
12	Zweites Byte der Print Farben
13	Nonprint Farben: Ink, Paper; Nonprint Attribute
14	Zweites Byte der Nonprint Farben
15	Scroll-Zähler

INTFFF (FD5E Hex)

Der MTX erzeugt jede 1/64 Sekunde einen Interrupt. Damit der Anwender möglichst einfach diese Interrupts verwenden kann, gibt es einen Interrupt Flag (INTFFF) und eine USERINT Position. Das Interrupt Flag gibt an, welche der vorhanden Routinen bei jedem Interrupt aufgerufen werden sollen. (0 = Kein Aufruf, 1 = Aufruf).

Bit 0 - Sound
    1 - Break Taste
    2 - Keyboard Auto Repeat (Automatische Wiederholung der gedrückten Taste nach einiger Zeit)
    3 - Sprite Bewegungen und blinkender Kursor
    4 - USER
    5 - USER
    6 - USER

Wenn irgendwelche der USER-Bits gesetzt sind, erfolgt ein Sprung zur USERINT Adresse.

PRORPL Wenn PRORPL = 1 ist, erfolgt eine Ausgabe zu der in IOPL spezifizierten Peripherie.
Wenn PRORPL = 0 ist, erfolgt eine Ausgabe zu der in IOPR spezifizierten Peripherie.

IOPR 0 = Screen
IOPL 1 = Centronics
2 = RS232 A

KBDFLG (FA91 Hex 64145 Dezimal)

Bit 7 - Alpha Lock
    5 - Page/Scroll Modus
    2 - Numerisches Tastenfeld (Nur Zahlen)

CTYSLT (FB45 Hex 64325 Dezimal)

Diese Systemvariable wählt die Tastaturabfrage und kann auch durch die zwei Schalter auf der Platine initialisiert werden.

ANHANG 6

FUNKTIONSTASTEN

Das Funktionstastenfeld kann verwendet werden, um eine Funktion mit nur einem Tastendruck einzugeben. Es sind 8 Tasten mit der Bezeichnung F1 bis F8 vorhanden.

Geben Sie dieses Programm ein:

```
10 PRINT ASC(INKEY$)
20 GOTO 10
```

Wenn Sie dieses Programm starten und eine Funktionstaste drücken, erscheint der entsprechende ASCII-Code. Das gleiche gilt für die Tasten mit Shift.

F1	128	REM	SHIFT und F1	136	CRVS
F2	129	CLS	SHIFT und F2	137	CLEAR
F3	130	ASSEM	SHIFT und F3	138	CLOCK
F4	131	AUTO	SHIFT und F4	139	ATTR
F5	132	BAUD	SHIFT und F5	140	COLOUR
F6	133	VS	SHIFT und F6	141	INK
F7	134	CONT	SHIFT und F7	142	CSR
F8	135	USER	SHIFT und F8	143	DATA

Falls erwünscht, können mit der GENPAT-Anweisung die Zeichen der Funktionstasten, die meist nur Leerzeichen oder Kästchen darstellen, sinngemäß neudefiniert werden.

Ein Beispiel:

```
10 GENPAT 1,130,0,96,144,144,240,144,148,0
```

ergibt das Zeichen 'A.' auf F3, die die ASSEM-Anweisung enthält.

ANHANG 7

FARBTABELLE

0	Transparent
1	Schwarz
2	Grün
3	Hellgrün
4	Dunkelgrün
5	Hellblau
6	Dunkelrot
7	Cyan
8	Rot
9	Hellrot
10	Dunkelgelb
11	Hellgelb
12	Dunkelgrün
13	Dunkelrot
14	Grau
15	Weiß

ANHANG 8

SOUND TABELLE 1

Frequenz = 4000000/32*n (Wobei n der jeweilige Wert ist.)

Direkter Befehl	SBUF	Resultat (Hz)
10	80	12500
20	160	6250
30	240	4166
40	320	3125
50	400	2500
60	480	2083
70	560	1785
80	640	1562
90	720	1388
100	800	1250
110	880	1136
120	960	1041
130	1040	961
140	1120	892
150	1200	833
160	1280	781
170	1360	735
180	1440	694
190	1520	657
200	1600	625
210	1680	595
220	1760	568
230	1840	543
240	1920	520
250	2000	500
260	2080	480
270	2160	462
280	2240	446
290	2320	431
300	2400	416
310	2480	403
320	2560	390
330	2640	378
340	2720	367
350	2800	357
360	2880	347
370	2960	337
380	3040	328
390	3120	320
400	3200	312
420	3360	297
440	3520	284
460	3680	271
480	3840	260
500	4000	250
520	4160	240

Direkter Befehl	SBUF	Resultat (Hz)
540	4320	231
560	4480	223
580	4640	215
600	4800	208
620	4960	201
640	5120	195
660	5280	189
680	5440	183
700	5600	178
720	5760	173
740	5920	168
760	6080	164
780	6240	160
800	6400	156
820	6560	152
840	6720	148
860	6880	145
880	7040	142
900	7200	138
920	7360	135
940	7520	132
960	7680	130
980	7840	127
1000	8000	125
1020	8160	122

SOUND TABELLE 2

RAUSCHEN

DC (Periodisches Rauschen)	SB	R
0	0	Shift rate = 7812.5 Hz
1	8	Shift rate = 3906.25 Hz
2	16	Shift rate = 2604.17 Hz
3	24	Shift rate = CHANNEL 2
Hintergrundrauschen		
4	32	Shift rate = 7812.5 Hz
5	40	Shift rate = 3906.25 Hz
6	48	Shift rate = 2604.17 Hz
7	56	Shift rate = CHANNEL 2

SOUND TABELLE 3

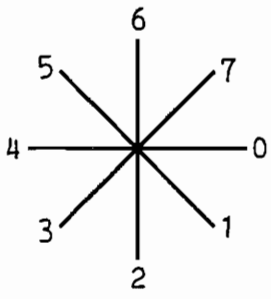
LAUTSTÄRKE

Direkter Befehl	SBUF	Resultat (DB)
0	0	OFF
1	16	-28
2	32	-26
3	48	-24
4	64	-22
5	80	-20
6	96	-18
7	112	-16
8	128	-14
9	144	-12
10	160	-10
11	176	- 8
12	192	- 6
13	208	- 4
14	224	- 2
15	240	- 0

ANHANG 9

ABSOLUTE RICHTUNGEN

Einige Grafik-Befehle, inklusive MVSPR und VIEW, verwenden Richtungsparameter, um eine von acht Richtungen zu spezifizieren. Diese sind in dem nachfolgenden Diagramm illustriert.



Technische Anhänge der MTX-Serien

TECHNISCHE ANHÄNGE DER MTX SERIEN

1. Einführung

Allgemeine Beschreibung

Die MTX 500 Serie umfaßt hochwertige Personalcomputer mit einer Datenbreite von 8 Bit, die sowohl in ROM-Version als auch in Diskettenversion betrieben werden können. Der Z80A Microprozessor und der TMS 9918A Videoprozessor sind die Grundbausteine des Systems, die es ermöglichen das Gerät sowohl in der preiswerten ROM-Version mit Farb-TV-Ausgang als auch in der Diskettenversion mit dem Betriebssystem CP/M, dann mit 80 Zeichen/Zeile und Farbmonitorausgang zu betreiben.

Der RAM-Bereich kann in der Grundversion mit 32k oder 64k bestückt werden, ist aber jederzeit auf bis zu 512k RAM erweiterbar.

Weitere 16k sind schon in der Grundversion ausschließlich für den Videospeicher reserviert. Weitere 24k ROM enthalten das MTX-Basic, den System-Monitor, zusätzliche Sprachen und Hilfsprogramme. Als Standard enthält bereits die Grundversion ein Kassetteninterface (Übertragungsrate 2400 Baud; softwaremäßig in den Systemvariablen änderbar), die Tastatur, ein Cartridge Port, zwei Spielreglereingänge, ein Druckerinterface (Centronics kompatibel), frei wählbare parallele I/O Ports, Farb-TV-Ausgang mit Tonübertragung, Farb- und Schwarzweiß-Monitorausgang und einen Audioausgang. Weitere Zusatzplatinen (nicht im Grundgerät enthalten) liefern zwei unabhängige RS 232 Schnittstellen mit einer gepufferten Buserweiterung, eine 80 Zeichen Farbkarte, ein Floppy Disc System, Pseudo Discs mit schnellem RAM-Zugriff und eine Ansteuerung für ein Winchester-Laufwerk.

Die Tastatur umfaßt 79 hochwertige Tasten (keine Gummi- oder Folientasten) und ist auf einer Stahlplatte montiert, die direkt mit dem Aluminiumgehäuse verbunden ist. Ein Aluminiumgehäuse wurde gewählt, um gute Wärmeableitung und Schutz, Fremdstrahlung und Eigenabstrahlung garantieren zu können.

2. Technische Daten

Hardware

Gehäuse

Das Gehäuse besteht aus zwei schwarz eloxierten Aluminiumschalen, die an der Vorderkante klappbar sind und an der Rückseite durch eine schwarze Kunststoffschiene verbunden werden. Die Seitenteile sind ebenfalls aus Aluminium und können durch Lösen von je drei Schrauben leicht entfernt werden. Durch die Wahl des Gehäuses ist eine gute Wärmeableitung für den Spannungsstabilisator gesichert.

Abmessungen in mm: Breite 488 Tiefe 202 Höhe 56
Gewicht in kg : 2,6

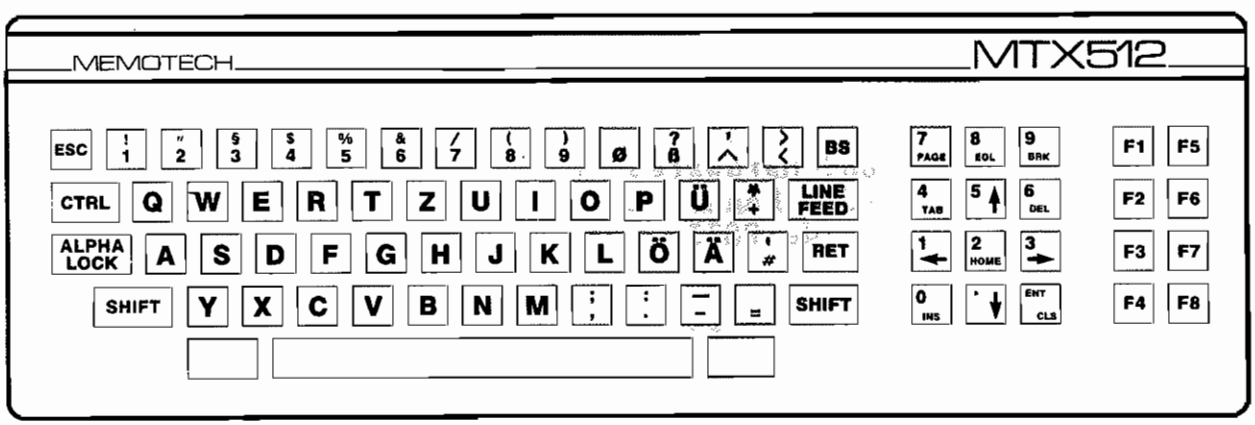
Tastatur

Die Tastatur ist auf einem 1-mm-starken Stahlblech montiert, das direkt an dem oberen Gehäuseteil festgeschraubt ist. Dadurch wird jegliche Belastung für die Platinen vermieden. Die 79 Tasten sind durch eine Extra-Platine miteinander verbunden. Die Verbindung zur Hauptplatine stellt ein steckbares Flachkabel dar.

Für den deutschen Markt sind die Tasten nach deutscher Norm angeordnet (kein QWERTY). Im einzelnen gibt es eine 57-tastige Schreibmaschinentastatur mit Standardabständen und Standardneigung.

Desweiteren gibt es eine abgesetzte numerische Tastatur, die auch zur Kursorsteuerung sowie für Editierfunktionen verwendet werden kann (12 Tasten). Ferner umfaßt die Tastatur 8 Funktionstasten auf denen in Zusammenarbeit mit SHIFT 16 Funktionen definiert sind. Zwei unbeschriftete Tasten, die rechts und links der großen SPACE-Taste angebracht sind, lösen bei gleichzeitigem Drücken einen System-Reset aus.

Alle alpha-numerischen Tasten sind selbstverständlich mit einer AUTOREPEAT-Funktion belegt.



CPU-Platine

In der unteren Gehäuseschale ist die CPU-Platine montiert. Sie enthält folgende Baugruppen:

Die Zilog Z80 CPU mit einer Taktfrequenz von 4 MHz.
24k ROM Bereich mit folgenden Programmen:

> MTX Basic mit MTX Logo Grafikbefehlen.
> MTX Noddy zur einfachen Bildschirmgestaltung.
> Monitorprogramm mit Z80 Assembler/Disassembler, sowie Ausgabe aller Z80 Register, Speicher und Programmlisting, einschließlich der Möglichkeit, diese zu verändern.

Videoprozessor mit extra RAM Speicher (VRAM).
Benutzer RAM mit 32k (MTX 500) oder 64k (MTX 512). Dieser Bereich steht jeweils im vollen Umfang dem Benutzer zur Verfügung.
Video Platine zur Erzeugung der TV und Tonsignale.
Eine Echtzeituhr.
Einen Zeichensatz (alpha-numerisch) mit Klein- und Großschreibung, freidefinierbare Zeichen und Sprites. Als Tastenfeld sind alle international geläufigen lieferbar.

Erweiterungsplatinen

Bis zu zwei Erweiterungsplatinen können im Hauptgehäuse untergebracht werden, und zwar RAM-, ROM- oder Schnittstellen-Platinen.

Speicherplatinen:

Der RAM-Bereich kann durch entsprechende Platinen um 32k, 64k, 128k oder 256k auf bis zu 512k erweitert werden.

Schnittstellen-Platinen:

Die Platine enthält zwei vollständig unabhängige RS232 Schnittstellen (Übertragungsrate bis 19200 Baud), die über ein vollständiges Handshakesystem und Modemleitungen verfügen. Außerdem wird über diese Platine der Diskettenbus betrieben (notwenig für Diskettenbetrieb). Mit dieser Platine können dann sowohl Floppydiscs (FDX) als auch Harddiscs (HDX) angesteuert werden. Zudem kann mit ihr ein MTX Node/Ringsystem aufgebaut werden.

Node/Ringsystem:

Dies ist ein Soft- und Hardwaresystem, mit dem ein MTX Netzwerksystem aufgebaut werden kann. Es beruht auf dem Z80 Interruptsystem und benötigt beide RS232 Schnittstellen der Schnittstellenplatine.

Die Kombinationsmöglichkeiten der Erweiterungsplatinen ist der nachstehenden Tabelle zu entnehmen:

	RAM Platinen				Schnittstellen-Platine
	32k	64k	128k	256k	
32k		*	*	*	*
64k	*	*	*	*	*
128k	*	*	*	*	*
256k	*	*	*	*	*
Schnittst.Platine	*	*	*	*	

* = gemeinsam einsetzbar

ROM Erweiterungen

Über einen Cartridge Port oder den Discbus sind folgende Erweiterungen möglich:

MTX Pascal
MTX Forth
Node System Software
Geschäfts-, Lern- und Spielsoftware

Anzeige

Über Farbfernseher oder Video Monitor können 24 Zeilen zu 40 Zeichen dargestellt werden. Dies ist jedoch durch eine 80-Zeichen-Farbkarte erweiterbar (diese Karte wird für FDX und HDX Systeme benötigt). Besonderheiten der Darstellung sind vollständige Bildschirmseiten sowie reine Textdarstellung (40 x 24 Zeichen) und gemischte Text-Grafikdarstellung (32 x 24 Zeichen und 256 x 192 Punkte in 16 Farben).

Besonderheiten der Grafik

Es können bis zu 32 unabhängige Sprites durch den Anwender definiert werden, zu denen dann zusätzlich ein Muster- und Hintergrundbild erzeugt werden kann. Die Erzeugung dieser Bilder wird durch entsprechende Befehle sehr vereinfacht.

Input/Output

Im Grundgerät sind folgende Ein- und Ausgänge vorhanden:

Kassetten Interface (Übertragungsrate programmierbar über Systemvariable; normal bis 2400 Baud)
Frei benutzbarer paralleler Input/Output Port
Zwei Joystick Anschlüsse mit Standart-Pinbelegung
Softwaremäßig steuerbarer Vierkanalton - drei Stimmen und Hintergrundgeräusch mit Ausgabe über Fernsehlautsprecher oder Audioausgang.
Monitorausgang mit Synchronimpulsen (1V Spitze)
Cartridge Port
Parallele Druckerschnittstelle (für Centronics kompatible Drukker).

Als Erweiterung stehen zur Verfügung:

Schnittstellenplatine (SSP) mit zwei RS232 Schnittstellen und Disketteninterface Bus.
Drucker mit parallel (Centronics) oder serieller RS232 Schnittstelle.

Stromversorgungseinheit

Eingang: 220 VAC 50 Hz
Ausgang: 22.5 VAC mit 1A mit Abgriffen bei 9V und 18V.
Abmessungen in mm: Breite 92 Tiefe 110 Höhe 70
Gewicht in kg: 1,0

Das Netzteil ist doppelt isoliert und hat an der Seite einen Kippschalter, der bei eingeschaltetem Gerät beleuchtet ist. Der Transformator ist im Gehäuse vibrationsfrei und gedämpft befestigt, so daß vom Netzteil keine Störungen ausgehen können. Ein- und Ausgangskabel sind mit einer Zugentlastung versehen. Der Ausgang wird über eine 6-poligen DIN Stecker (240 Grad) mit dem Hauptgerät verbunden. Das Gehäuse ist aus Sicherheitsgründen verschweißt.

MTX Diskettensystem

Der MTX kann als Floppy-Disc- (FDX) und Winchester-Disc-System (HDX) betrieben werden.

Beide Systeme benötigen die Schnittstellen Platine im Hauptgerät und mindestens 64k RAM. Für beide Systeme gelten die folgenden Daten:

Das Disc-System ist in einem gesonderten 19" schwarz eloxiertem Aluminiumgehäuse untergebracht. Das Gehäuse enthält ein Einschubsystem, daß folgende Karten aufnehmen kann:

Eine Computer-Erweiterungskarte
Eine 80 Zeichen Farbkarte
Vier Silicon Disc Speichersätze (Pseudofloppy)
Eine Floppycontroler-Karte
Ein integriertes Netzteil, das auch das Hauptgerät versorgt (Eingangsspannung 220VAC, 50Hz bei deutscher Version)
Einen Parallelport zur Buserweiterung
Einbaumöglichkeit für zwei 5 1/4" Laufwerke.
Zusätzlich kann ein Akkupuffer eingebaut werden.
Im Preis für die FDX und HDX Systeme sind die Lizenzkosten für CP/M 2.2 und das CP/M 2.2 Betriebssytem selbst enthalten.

80 Zeichen Farbkarte

Notwendig für den Disc-Betrieb ist die 80 Zeichen Farbkarte, mit der dann alle entsprechenden Softwarepakete des CP/M wie z.B. Colour Wordstar voll lauffähig sind. Zur Zeit ist bereits eine Vielzahl entsprechender Software auf dem Markt erhältlich (CP/M Software).

Die 80 Zeichen Farbkarte verfügt über folgende Ein- und Ausgänge:

RGB Monitorausgang mit wahlweiser positiver oder negativer Synchronisation, schwarz/weiß Videoausgang mit negativer Synchronisation (1V Spitze zu Spitze)
Lichtgriffeleingang
Einkanal Ton
Das Ausgabeformat beträgt 80 Zeichen bei 24 Zeilen maximal, und im Grafikmodus stehen 160*96 Punkte zur Verfügung.
Für die Ausgabe stehen zwei Zeichensätze zu je 96 Zeichen mit echten Unterlängen zur Verfügung.
Ein 4k ROM enthält Grafikzeichen.
Ferner ist die Baugruppe vollständig Teletext kompatibel.
Schnelle und störungsfreie Manipulation des Bildschirms ist möglich (mit ca. 25000 Baud).
Durch ein ROM ist es möglich, sämtliche CP/M typische Monitorsteuerungen durchzuführen, als da wären:

- Vollständige Kursorkontrolle
- Vektor Plot und Punkt Plot
- Sämtliche Editierfunktionen mit Bildschirm-Dump
- Kontrolle aller Attribute für Farb- und Schwarzweißausgabe.

Das Silicon Disc (Pseudo Floppy)

Die Silicon Discs sind schnelle RAM Platinen mit 1/4 oder 1 Megabyte Speicherkapazität, die vom CP/M wie eines der Laufwerke 0 bis 13 verwaltet werden. Bis zu 4 Silicon Discs können im FDX oder HDX Gehäuse untergebracht werden (mit 1 bis 8 Megabytes pro Einschub). Die einzelnen Kontroller können bis zu 4 Pseudodiscs ansteuern, wodurch sich bei maximal 4 Einschubmöglichkeiten eine Gesamtspeicherkapazität von 32 Megabytes mit schnellem Zugriff ergibt. Dies ist unabhängig von den Diskettenlaufwerken (5 1/4" oder 8"), von denen bis zu 4 Stück angesteuert werden können. Damit können Sie Ihren MTX Computer so erweitern, daß für fast alle Aufgaben ausreichend Speicherplatz zur Verfügung steht. Die Besonderheit der Pseudodiscs liegt darin, daß sie etwa 5 mal schneller als ein Winchesterlaufwerk und sogar 50 mal schneller als ein Diskettenlaufwerk sind.

Das System arbeitet dadurch so schnell und effektiv, daß es durchaus mit 16 und 32 Bit Rechnern verglichen werden kann, und dabei noch erhebliche Einsparungen im Hardwarebereich gegenüber diesen Rechnern zu erkennen sind. Ebenfalls wird der Verwaltungsaufwand (Hard- und Software) erheblich vereinfacht.

Zudem ermöglichen die Pseudodiscs es, viele Programme (Database, Textverarbeitung, Compilation) mit nur einem Laufwerk äußerst effektiv zu benutzen.

Ferner entstehen beim Einsatz der Pseudodiscs keine laufenden Kosten (Anschaffung neuer Disketten) und es treten auch keine mechanischen Abnutzungserscheinungen auf.

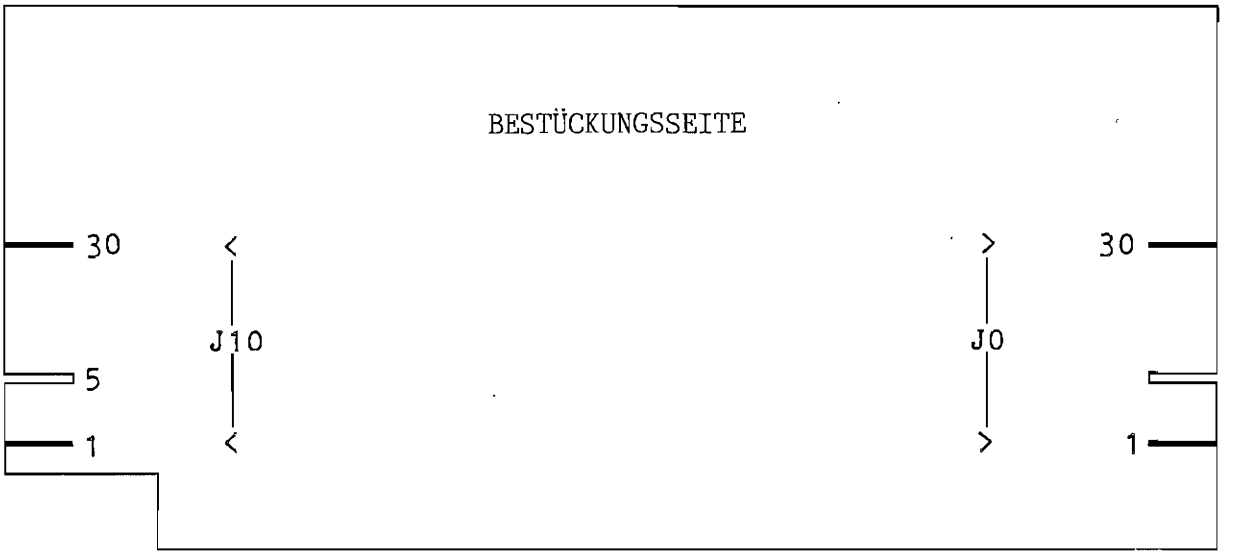
Floppykontroller-Karte

Auf dieser Platine kommt der vollständige Western Digital 1719 IC-Satz zum Einsatz, mit dem fast alle CP/M Floppydrives angesteuert werden können (0-13 Laufwerke). Dadurch können alle Laufwerke mit 5 1/4" oder 8", ein oder doppelseitig und mit Single- oder Double-Density angesprochen werden, falls sie über einen SASI (Shugart) Anschluß verfügen. Es besteht auch die Möglichkeit, gleichzeitig unterschiedliche Laufwerke mit SASI Anschluß an einem Gerät zu betreiben.

Der 1719 IC-Satz in Verbindung mit einem DMA-Kontroller sorgt für eine schnelle Übertragung der Daten in beide Richtungen, so daß auch bei großen Speicherkapazitäten die Standzeiten minimal werden.

Über einen Steckverbinder können externe Floppies angeschlossen werden.

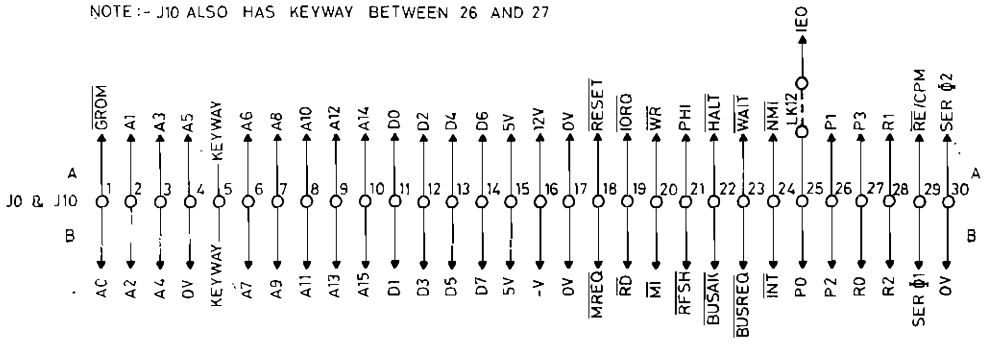
3. Der MTX Systembus



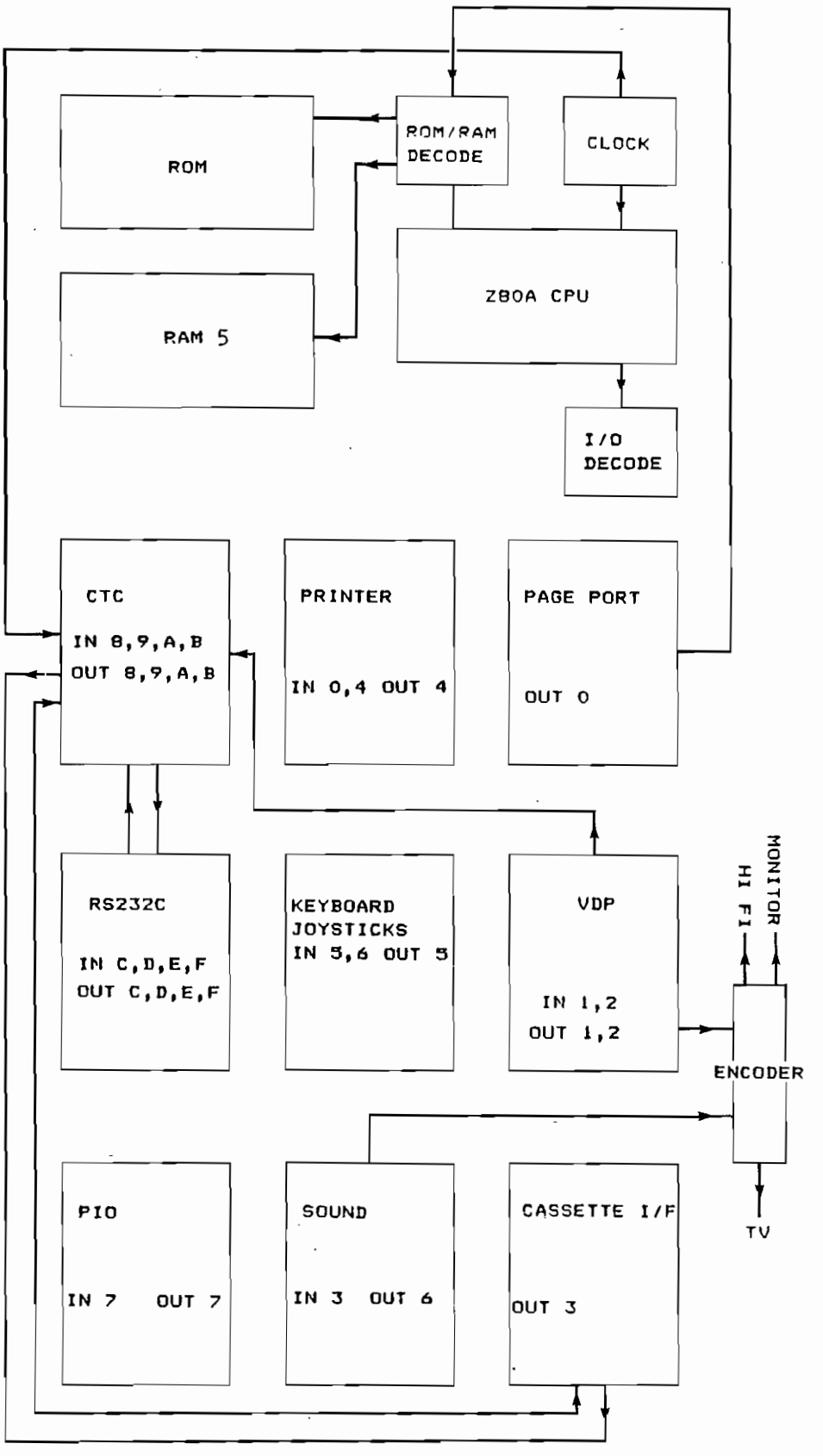
Der Systembus stellt den gesamten Z80 Bus zur Verfügung, ebenso die einzelnen Spannungen, den ROMenable-Anschluß (GROM), die ROM-banking Adressen (RO -R2), die RAMbanking Adressen (PO -P3) und die seriellen Taktsignale 01 und 02.

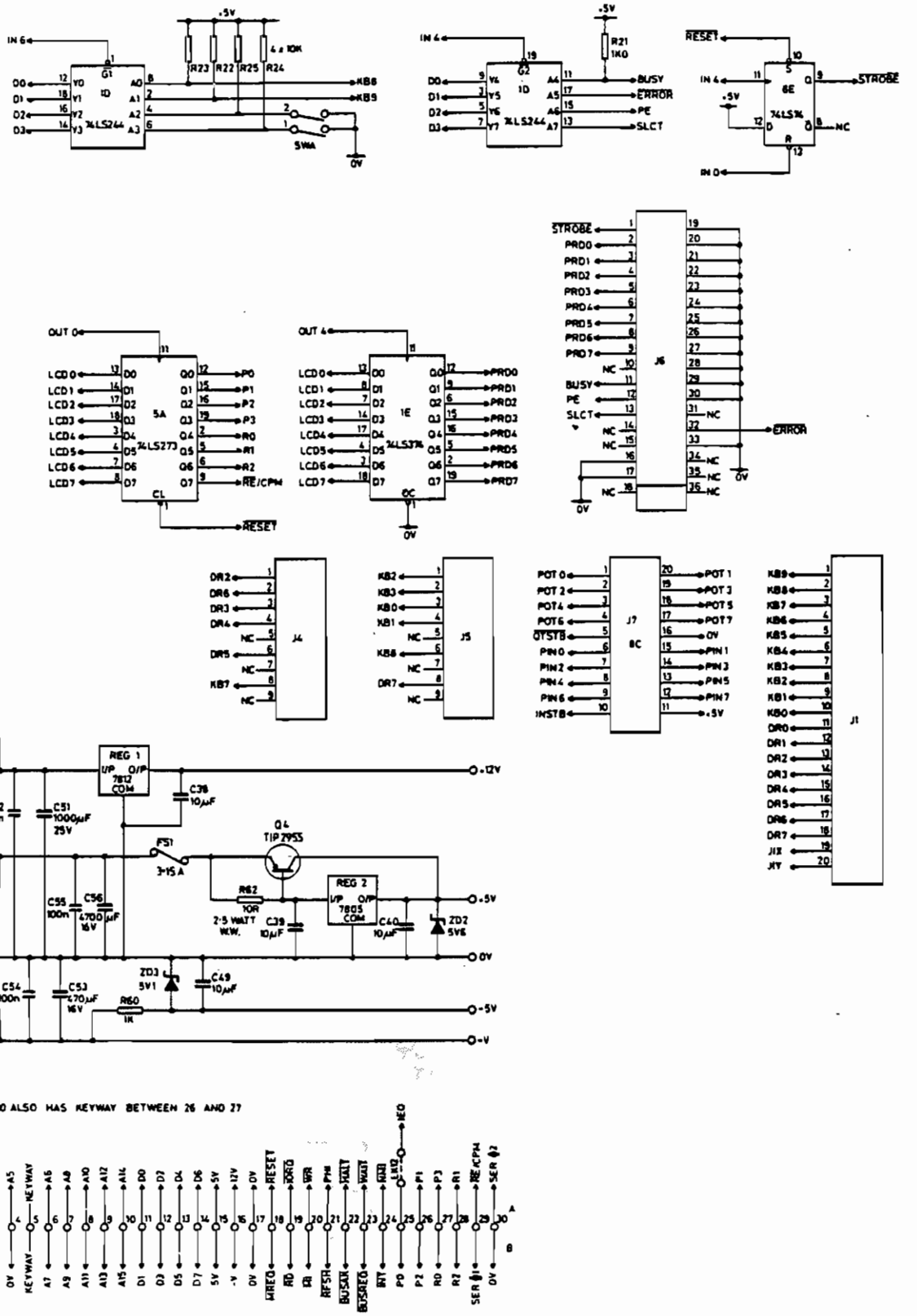
Alle Anschlüsse sind von außerhalb des Gehäuses über einen direkten Steckverbinder (60polig - 2 x 30; 2,54 mm Raster) an den Kontakten J10 abgreifbar. Mit einem gleichen Stecker können die Signale auch im Gerät an den Kontrakten J0 abgegriffen werden. (Wichtig!! Die Kontaktreihen J0 und J10 sind nicht identisch, sondern spiegelbildlich. Damit können Sie Ihre Erweiterungen nicht wahlweise an eine der Kontaktreihen anschließen. Lesen Sie deshalb bei fertigen Erweiterungsplatinen die Einbauanleitung besonders sorgfältig)!

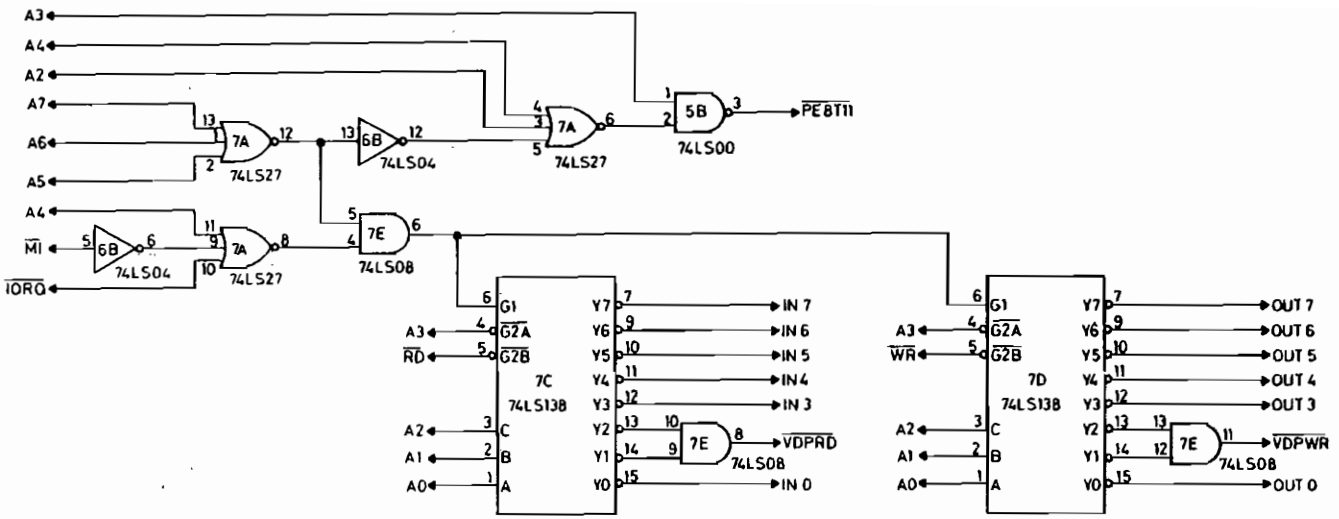
Die folgende Figur zeigt Ihnen die Anschlußbelegung der Kontaktreihen. 'A' ist dabei die Bestückungsseite und 'B' die Lötseite der Platine.



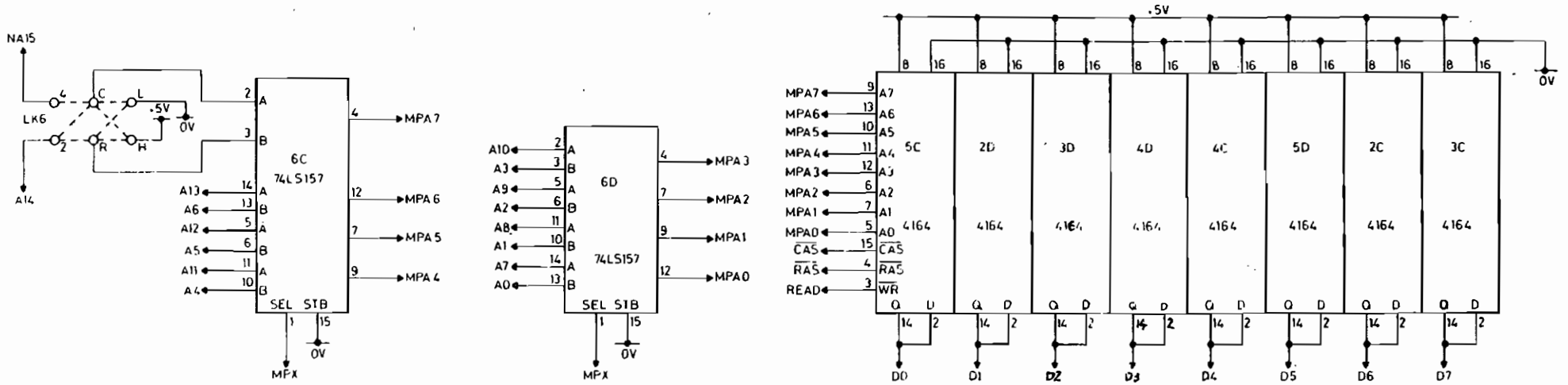
4. Blockdiagramm des MTX Grundgeräts

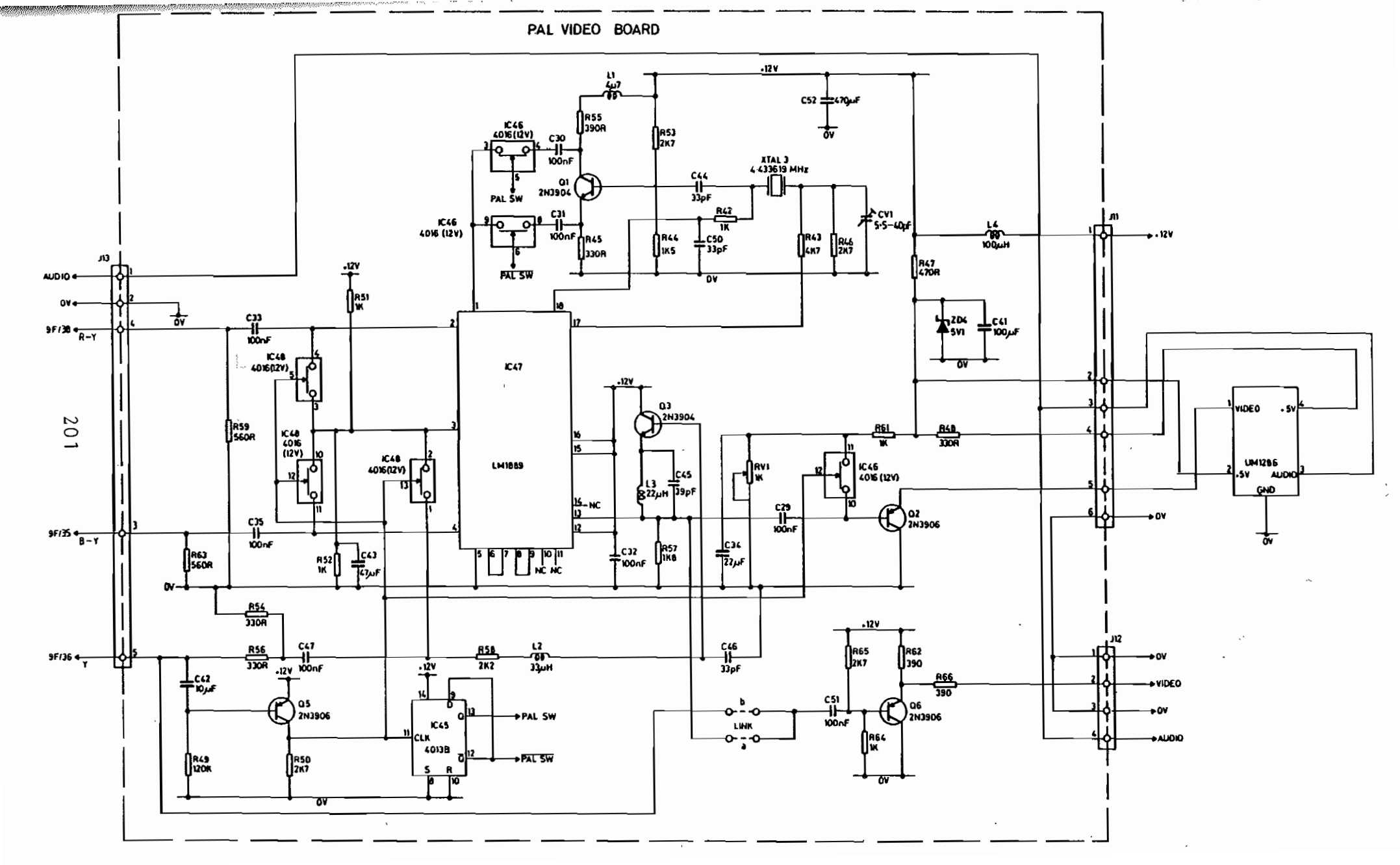


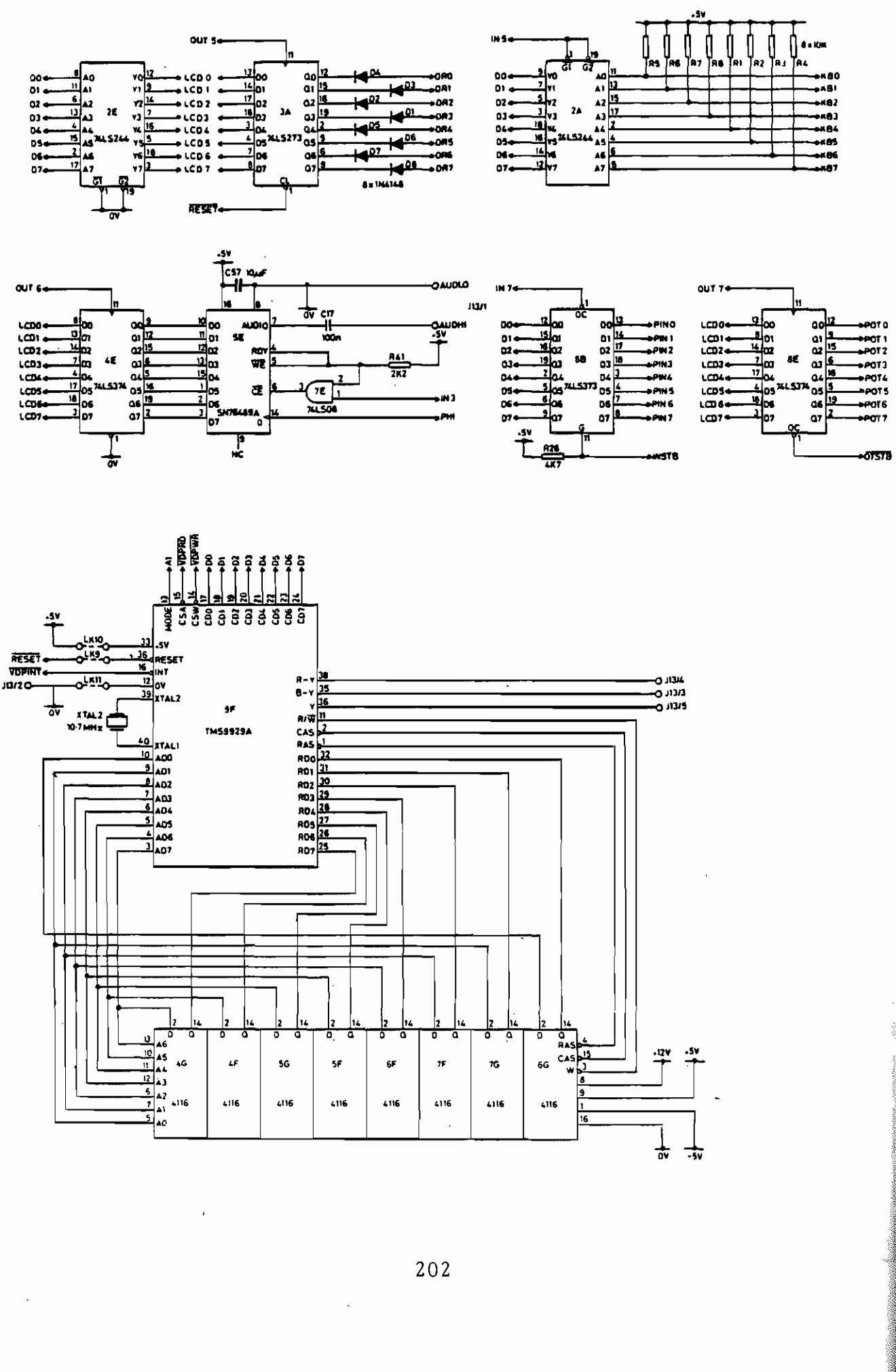

IN 6
+5V
4 x 10K
R23
R22
R25
R24
G1
1D
74LS244
KB8
KB9
SW4
0V
IN 4
R21
1K0
G2
BUSY
ERROR
PE
SLCT
RESET
STROBE
6E
74LS74
NC
IN 0
OUT 0
OUT 4
LCD0
LCD1
LCD2
LCD3
LCD4
LCD5
LCD6
LCD7
5A
1E
74LS273
CL
RESET
P0
P1
P2
P3
R0
R1
R2
RE/CPM
PRD0
PRD1
PRD2
PRD3
PRD4
PRD5
PRD6
PRD7
PRD
J6
ERROR
J4
J5
J7
8C
POT 0
POT 1
POT 2
POT 3
POT 4
POT 5
POT 6
POT 7
PIN 0
PIN 1
PIN 2
PIN 3
PIN 4
PIN 5
PIN 6
PIN 7
J1
D19
1N4004
D14
1N4004
REG 1
7812
C38
10µF
C52
100n
C51
1000µF
25V
J9/1
J9/2
J9/3
J9/4
J9/5
+12V
D15
1N5402
D17
1N5402
FS1
3·15 A
Q4
TIP 2955
R62
10R
2·5 WATT
W.W.
C39
10µF
REG 2
7805
C40
10µF
ZD2
5V6
+5V
C55
100n
C56
4700µF
16V
0V
ZD3
5V1
C49
10µF
D16
1N4002
C54
100n
C53
470µF
16V
R60
1K
-5V
D18
1N4002
-V
NOTE :- J10 ALSO HAS KEYWAY BETWEEN 26 AND 27
J10
A
B
GROM
A1
A3
A5
KEYWAY
A6
A8
A10
A12
A14
D0
D2
D4
D6
5V
12V
0V
RESET
XORQ
WR
PHI
HALT
WAIT
NMI
INT
REQ
P0
P1
P2
P3
RD
R1
R2
RE/CPM
SER ϕ1
SER ϕ2
A0
A2
A4
0V
KEYWAY
A7
A9
A11
A13
A15
D1
D3
D5
D7
5V
-V
0V
WREQ
RD
WR
RFSH
BUSAK
BUSREQ
INT
P0
SER ϕ1
0V

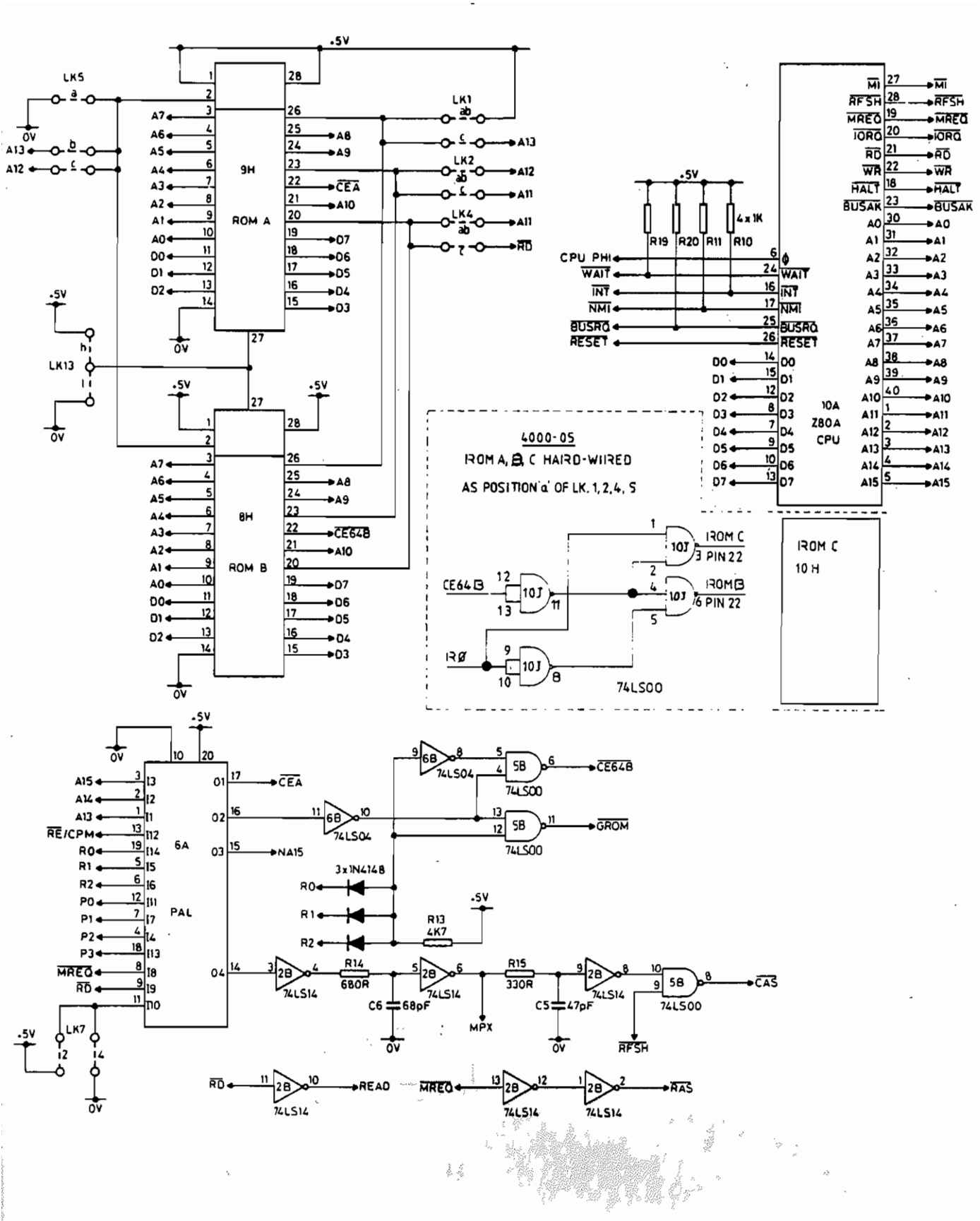

A3
A4
A2
A7
A6
A5
A4
MI
IORQ
7A
74LS27
6B
74LS04
7A
74LS27
7A
74LS27
5B
74LS00
PEBT11
7E
74LS08
6B
74LS04
7A
74LS27
G1
G2A
G2B
7C
74LS138
C
B
A
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
IN 7
IN 6
IN 5
IN 4
IN 3
7E
74LS08
VDPRD
IN 0
RD
WR
7D
74LS138
OUT 7
OUT 6
OUT 5
OUT 4
OUT 3
7E
74LS08
VDPWR
OUT 0

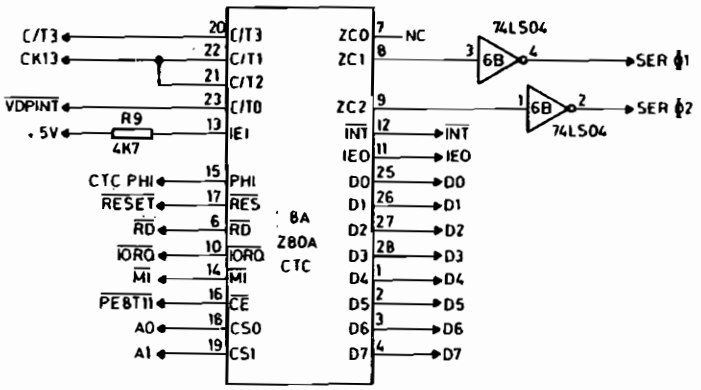
200

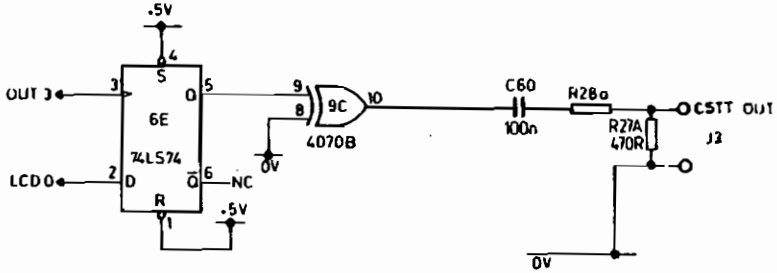

NA15
LK6
A14
C
L
R
H
+5V
0V
6C
74LS157
SEL
STB
MPX
MPA 7
MPA 6
MPA 5
MPA 4
A13
A6
A12
A5
A11
A4
6D
74LS157
MPA 3
MPA 2
MPA 1
MPA 0
A10
A3
A9
A2
A8
A1
A7
A0
+5V
0V
5C
2D
3D
4D
4C
5D
2C
3C
4164
MPA7
MPA6
MPA5
MPA4
MPA3
MPA2
MPA1
MPA0
CAS
RAS
READ
WR
D0
D1
D2
D3
D4
D5
D6
D7

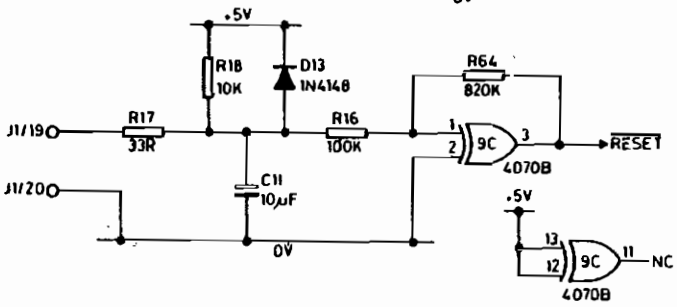

PAL VIDEO BOARD
+12V
0V
L1 4µ7
C52 470µF
R55 390R
R53 2K7
IC46 4016 (12V)
C30 100nF
PAL SW
Q1 2N3904
C44 33pF
XTAL 3 4.433619 MHz
IC46 4016 (12V)
C31 100nF
R45 330R
R44 1K5
C50 33pF
R42 1K
R43 4K7
R46 2K7
CV1 5-5-40pF
L4 100µH
R47 470R
ZD4 5V1
C41 100µF
J11
J13
AUDIO
0V
9F/38 R-Y
9F/35 B-Y
9F/36 Y
R51 1K
C33 100nF
IC48 4016(12V)
IC47
LM1889
R59 560R
IC48 4016 (12V)
IC48 4016(12V)
Q3 2N3904
L3 22µH
C45 39pF
R61 1K
R48 330R
RV1 1K
IC46 4016 (12V)
C29 100nF
Q2 2N3906
C35 100nF
R63 560R
R52 1K
C43 47µF
C32 100nF
R57 1K8
C34 22µF
R54 330R
R56 330R
C47 100nF
R58 2K2
L2 33µH
C46 33pF
C42 10µF
Q5 2N3906
IC45 4013B
PAL SW
PAL SW
R49 120R
R50 2K7
LINK
R65 2K7
R62 390
R66 390
C51 100nF
Q6 2N3906
R64 1K
J12
0V
VIDEO
0V
AUDIO
VIDEO
+5V
UM1286
+5V
AUDIO
GND
201

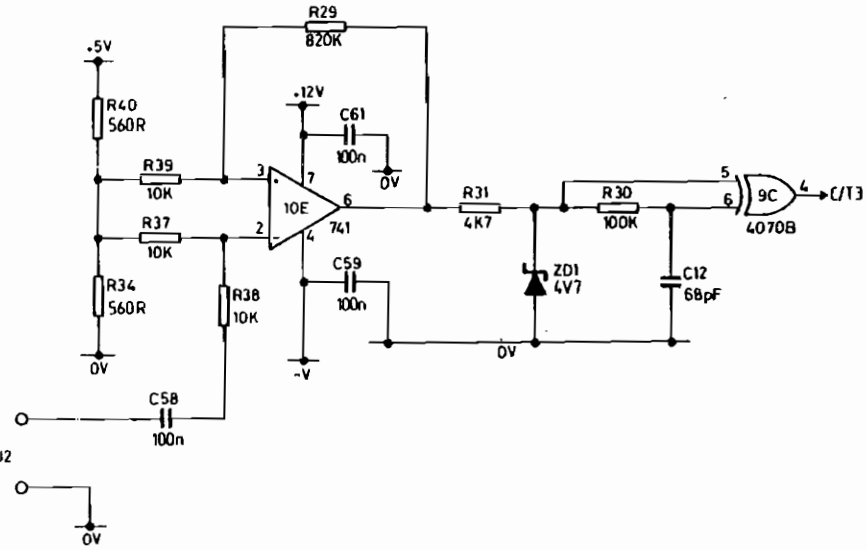


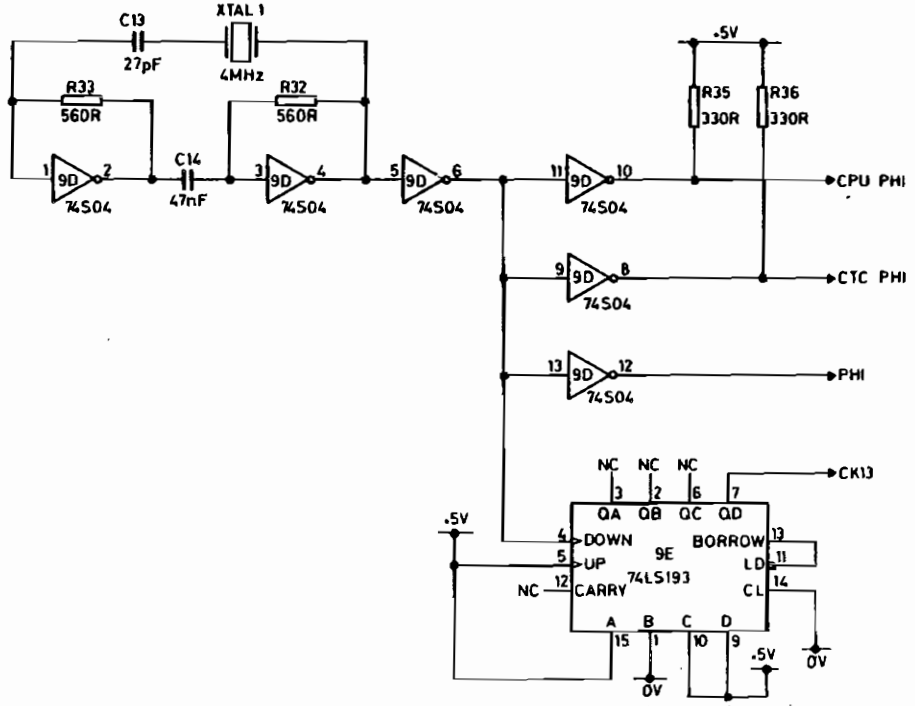

+5V
LK5
LK1
LK2
LK4
LK13
LK7
9H
ROM A
8H
ROM B
4000-05
IROM A, B, C HARD-WIRED
AS POSITION 'a' OF LK. 1, 2, 4, 5
74LS00
IROM C
10 H
10A
Z80A
CPU
4x1K
R19
R20
R11
R10
CPU PHI
WAIT
INT
NMI
BUSRQ
RESET
6A
PAL
3x1N4148
R13
4K7
R14
680R
R15
330R
C6 68pF
C5 47pF
74LS14
74LS04
74LS00
MPX
RFSH
READ
RAS
CAS
CEA
CE64B
GROM
NA15


C/T3
CK13
VDPINT
+5V
R9
4K7
CTC PHI
RESET
RD
IORQ
M1
PEBT11
A0
A1
8A
Z80A
CTC
ZC0
ZC1
ZC2
INT
IEO
D0
D1
D2
D3
D4
D5
D6
D7
NC
74LS04
SER Φ1
SER Φ2


+5V
OUT 3
LCD0
6E
74LS74
NC
4070B
C60
100n
R28a
R27A
470R
CSTT OUT
J3
0V


+5V
R18
10K
D13
1N4148
R64
820K
J1/19
R17
33R
R16
100K
C11
10µF
J1/20
RESET
4070B
NC
0V

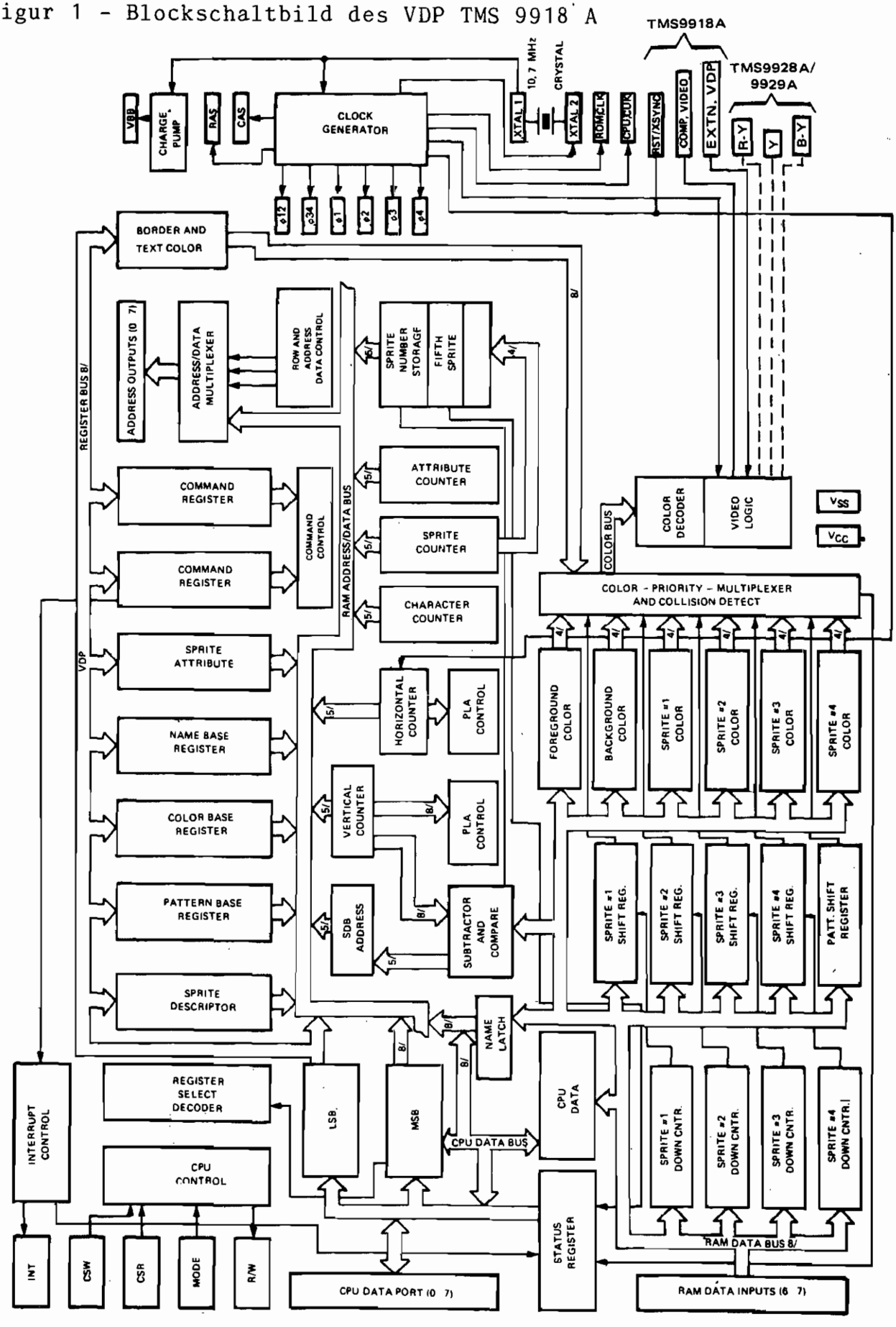

R29
820K
+5V
+12V
R40
560R
C61
100n
R39
10K
R37
10K
R34
560R
R38
10K
10E
741
R31
4K7
R30
100K
C59
100n
ZD1
4V7
C12
68pF
C/T3
4070B
C58
100n
J2
0V
-V


C13
27pF
XTAL 1
4MHz
R33
560R
R32
560R
C14
47nF
74S04
+5V
R35
330R
R36
330R
CPU PHI
CTC PHI
PHI
CK13
NC
QA
QB
QC
QD
DOWN
UP
CARRY
9E
74LS193
BORROW
LD
CL
A
B
C
D
0V

6. Der Videoprozessor TMS 99XX A

Je nach Vertriebsbereich kann der MTX Computer mit dem TMS 9918 A (England), dem TMS 9928 A (Nord Amerika) oder dem TMS 9929 A (Europa) ausgerüstet werden. Der Videoprozessor (VDP) ist ein I/O Baustein mit zwei Ports (Mode = 0 für Port 1 und Mode = 1 für Port 2). Der VDP mischt die Farbsignale, fügt das Audiosignal hinzu und gibt das Ausgangssignal an den HF Modulator weiter. Die Schaltungsdetails variieren je nach Bestimmungsland.

Figur 1 - Blockschaltbild des VDP TMS 9918 A



CPU Interface Kontrollsignale

Die Art und Richtung der Datenübertragung wird durch die Leitungen CSW, CSR und MODE kontrolliert. CSW zeigt an, daß von der CPU zum VDP geschrieben werden soll. Im aktiven Zustand (low) werden die 8 Datenbits (D7 - D0) zum VDP übertragen. CSR steuert den Datentransfer vom VDP zur CPU. Auch hier ist der aktive Zustand 'low', in dem dann die Daten (D7 - D0) zur CPU übertragen werden. CSR und CSW sollten niemals gleichzeitig aktiv sein, da der VDP dann Daten ausgibt und gleichzeitig undefinierbare Werte liest.

MODE bestimmt das Ziel oder den Ausgangspunkt eines Transfers, und ist normalerweise mit einer der niederwertigen CPU-Adreßleitungen verbunden.

Schreiben von der CPU in ein VDP Register

Der VDP verfügt über acht reine Schreibregister (Auswertung dieser erfolgt durch den VDP) und ein Leseregister, daß den Status des VDP enthält. Die Schreibregister steuern die VDP-Operationen und die Aufteilung des Videorams (VRAM). Das Statusregister steuert den Interrupt, zeigt die Überlappung der Sprites und das fünfte Sprite-Statusflag an.

Jedes der 8 Schreibregister kann durch 2 8-Bit Datentransfers von der CPU aus geladen werden. Tabelle 1 zeigt das benötigte Format für diese Datentransfers. Das erste übertragene Byte enthält die Daten, das zweite Byte gibt das Ziel an. Das höchstwertige Bit (D7) dieses zweiten Bytes muß dabei auf '1' sein. D6 bis D3 müssen '0' sein und die drei niederwertigsten Bits (D0 - D2) geben das Zielregister an. Der Mode-Eingang muß bei der Übertragung dieser beiden Bytes auf 'high' liegen.

Zum Umschreiben der internen Systemregister (keine VDP Register) nach der Übertragung eines Datenbytes, muß das Statusregister gelesen werden, damit die interne Logik den nächsten Datentransfer als solchen erkennt, und ihn nicht mit einer Registerzuweisung verwechselt. Dieses Problem tritt insbesondere bei Interruptbetriebenen Systemen auf. Jedoch sollte immer, wenn der Status des VDP nicht sicher bekannt ist, dieser Lesevorgang durchgeführt werden. Man beachte auch, daß die CPU Adresse beim Schreiben in ein VDP Register zerstört wird.

Schreiben von der CPU in das VRAM

Die CPU schreibt in das VRAM über den VDP, wobei ein 14-stelliges, sich selbst erhöhendes internes Adreßregister benutzt wird. Zur Übertragung der jeweiligen Adresse werden deshalb zwei 8 Bit Ladebefehle benötigt. Ein dritter 8 Bit Transfer überträgt dann die Daten in das adressierte VRAM Byte. Danach wird das Adreßregister automatisch erhöht. Bei Übertragung mehrerer Datenbytes nacheinander (die auch nebeneinander angezeigt werden sollen), braucht deshalb die Adreßausgabe nur einmal erfolgen. Bei der Übertragung des zweiten Adreßbytes muß darauf geachtet werden, daß D7 '0' und D6 '1' ist. Das MODE-Signal muß während der Adreßübertragung auf 'high' und bei der Datenübertragung auf 'low' sein. CSW muß bei allen Übertragungen aktiv (low) sein. (Siehe Tabelle 1)

Lesen des VDP Statusregisters

Mit einem einzigen 8 Bit Lesebefehl kann der Inhalt des VDP Statusregisters durch die CPU gelesen werden. MODE muß in diesem Fall 'high' sein und CSR muß aktiviert werden (low).

Lesen aus dem VRAM

Gegenüber vielen anderen Computern, die eine eigene VDP mit gesondertem VRAM haben, ist es beim MTX möglich, dieses VRAM auszulesen. Dabei liest die CPU die Daten über den VDP, ähnlich wie beim Schreiben. Auch hier werden zunächst die beiden Adreßbytes übertragen und dann das Datenbyte gelesen. Wie beim Schreiben müssen beim Lesen aufeinanderfolgender Daten die Adressen nur einmal übertragen werden. Der Unterschied zum Schreiben besteht im wesentlichen darin, daß zunächst Daten (hier die Adressenbytes) zum VDP übertragen werden müssen. Dies erfolgt genauso wie beim Schreiben in das VRAM, nur müssen D6 und D7 des zweiten Adreßbytes '0' sein. Nach diesem Transfer erfolgt ein Lesen von dem VDP zur CPU. Dazu muß CSW inaktiv und CSR aktiv (low) werden. Während des Adreßtransfers muß MODE auf '1' und beim Lesen der Daten auf '0' sein. Die Ausführungszeit beträgt dabei 8 Mikrosek. um die Daten vom VRAM zum VDP zu transferieren und 3 Mikrosek. nach der Adreßübertragung. Hinzu kommen noch die Transferzeiten zwischen CPU und VDP.

TABELLE 1 - CPU/VDP DATENTRANSFER

OPERATION		MSB 7	6	5	4	BIT 3	2	1	LSB 0	CSW	CSR	MODE
WRITE TO VDP REGISTER												
Byte 1	Data Write	D7	D6	D5	D4	D3	D2	D1	D0	0	1	1
Byte 2	Register Select	1	0	0	0	0	RS2	RS1	RS0	0	1	1
WRITE TO VRAM												
Byte 1	Address set up	A7	A6	A5	A4	A3	A2	A1	A0	0	1	1
Byte 2	Address set up	0	1	A13	A12	A11	A10	A9	A8	0	1	1
Byte 3	Data Write	D7	D6	D5	D4	D3	D2	D1	D0	0	1	0
READ FROM VDP REGISTER												
Byte 1	Data Read	D7	D6	D5	D4	D3	D2	D1	D0	1	0	1
READ FROM VRAM												
Byte 1	Address set up	A7	A6	A5	A4	A3	A2	A1	A0	0	1	1
Byte 2	Address set up	0	0	A13	A12	A11	A10	A9	A8	0	1	1
Byte 3	Data Read	D7	D6	D5	D4	D3	D2	D1	D0	1	0	0

Der VDP Interrupt

Der VDP INT Ausgang wird benutzt, um nach jedem Ausgabedurchlauf einen Interrupt zu erzeugen (alle 1/50 oder 1/60 sek., je nach Land). Der INT Ausgang ist aktiv, wenn das Interrupt Enable (IE) bit im VDPregister 1 auf '1' und das F-Bit des Statusregisters ebenfalls auf '1' ist. Der Interrupt wird gelöscht, wenn das Statusregister gelesen wird.

Initialisierung des VDP

Der VDP wird extern bei jedem RESET initialisiert. Dabei muß der RESET Eingang für mindestens 3 Mikrosek. auf 'low' gehen. Dieser RESET synchronisiert alle Taktsignale mit ihrer negativen Flanke, setzt die Horizontal- und Vertikalzähler auf einen definierten Anfangszustand und löscht die VDP-Register 1 und 0. Der Bildschirm wird dabei automatisch gelöscht, da das Blankregister im VDP-Register 1 auf '0' gesetzt wird. Trotz des gelöschten Bildschirms versorgt der VDP das VRAM weiterhin mit Refreshsignalen, damit dieser Zustand erhalten bleibt. Nur während des eigentlichen Resetimpulses unterbleibt der Refreshvorgang.

VDP/VRAM Schnittstelle

Der VDP kann 16k Bytes des VRAM über 14 Adreßleitungen adressieren. Der VDP holt Werte aus dem VRAM um das Videobild mit Inhalt zu generieren. Zudem dient der VDP als Zwischenspeicher für alle Datentransfers zwischen CPU und VRAM. Ferner erzeugt der VDP die für das dynamische VRAM notwendigen Refreshsignale.

VRAM Interface Kontrollsignale

Das VDP-VRAM-Interface besteht aus zwei undirektionalen 8 Bit Datenleitungen und drei Steuerleitungen. Der Datentransfer vom VRAM zum VDP erfolgt über den VRAM Lesebus (RDO-RD7), während der Transfer vom VDP zum VRAM für Daten und Adressen über den VRAM Adreß/Datenbus (ADO-AD7) erfolgt. Die VRAM Reihenadresse wird bei aktivem RAS (low) und die VRAM Spaltenadresse bei aktivem CAS (low) übertragen. Der Datentransfer zum VRAM erfolgt, wenn R/W aktiv (low) ist.

Die Schreibregister des VDP

Die 8 Schreibregister des VDP werden in Tabelle 2 dargestellt. Register 0 und 1 enthalten Flags, die den VDP in unterschiedliche Betriebsarten versetzen. Die Register 2 bis 6 enthalten Werte, mit denen Startpositionen und Unterteilungen im VRAM festgelegt werden. Register 7 wird für den Bildhintergrund und die Textfarbe verwendet.

VDP REGISTER

REGISTER	MSB<—Bit—>LSB 0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	M3	EV
1	4/16K	BLANK	IE	M1	M2	0	SIZE	MAG
2	0	0	0	0	NAME TABLE BASE ADDRESS			
3	<——COLOUR TABLE BASE ADDRESS——>							
4	0	0	0	0	0	PATTERN GENERATOR BASE ADDRESS		
5	0	<——SPRITE ATTRIBUTE TABLE BASE ADDRESS——>						
6	0	0	0	0	0	SPRITE PATTERN GENERATOR BASE ADDRESS		
7	<——TEXT COLOUR 1——>				TEXT COLOUR 0/BACKDROP COLOUR			
STATUS READ ONLY	F	5S	C	<——FIFTH SPRITE NUMBER——>				

Im folgenden werden die einzelnen VDP-Register beschrieben:

REGISTER 0 enthält zwei VDP-Kontrollbits. Alle weiteren Bits sind für spätere Modifikationen reserviert und müssen auf '0' gesetzt werden.

Bit6 M3 (Modebit 3)
Bit7 Externes Video enable/disable
'1' enables externen Videoinput
'0' disables externen Videoinput

REGISTER 1 enthält acht VDP-Kontrollbits.

Bit0 4/16k Auswahl
'0' für 4k RAM Operationen
'1' für 16k RAM Operationen (MTX) Operationen)
Bit1 BLANK enable/disable
'0' löscht den aktiven Displaybereich
'1' erhält den aktiven Displaybereich
Beim Löschen wird nur noch die Hintergrundfarbe ausgegeben.
Bit2 IE (Interrupt Enable)
'0' disables VDP-Interrupt
'1' enables VDP-Interrupt
Bit3,4 M1, M2 (Modebits 1 und 2)
M1, M2 und M3 bestimmen den Operationsmodus des VDP:

M1	M2	M3	Modus
0	0	0	Grafik1 Modus
0	0	1	Grafik2 Modus
0	1	0	Mehrfarb. Modus
1	0	0	Text-Modus

Bit5 Reserviert
Bit6 Größe (bestimmt die Größe der Sprites)
'0' wählt Spritegröße 0 (8x8 Bits)
'1' wählt Spritegröße 1 (16x16 Bits)
Bit7 MAG (Vergrößerung der Sprites)
'0' wählt MAG0 (1x)
'1' wählt MAG1 (2x)

REGISTER 2 definiert die Basisadresse des Unterblocks, der die Namenstabelle enthält. Sein Wert kann zwischen 0 und 15 liegen. Der Inhalt dieses Registers bildet die obersten 4 Bits der 14-Bit-Adresse der Namenstabelle; somit ist die Basisadresse dieser Tabelle gleich dem Inhalt von Register 2 mal '400h'.

REGISTER 3 definiert die Basisadresse des Farbtabellenunterblocks. Der Bereich dessen Inhalts kann zwischen 0 und 255 liegen. Der Inhalt dieses Registers definiert die oberen 8 Bits der 14-Bit-Adresse der Farbtabelle; somit ist die Basisadresse dieser Tabelle der Inhalt des Registers 3 mal '40h'.

REGISTER 4 definiert die Basisadresse des Musters, Text oder Mehrfarbgenerator Unterblocks. Seine Werte dürfen zwischen 0 und 7 liegen. Dieses Register legt damit die oberen 3 Bits der 14-Bit Adresse dieser Blöcke fest; somit ist die Basisadresse durch den Inhalt dieses Registers mal '800h' festgelegt.

REGISTER 5 definiert die Basisadresse des Unterblocks für die Sprite Attribute. Möglich sind Werte zwischen 0 und 127. Der Inhalt dieses Registers bestimmt die oberen 7 Bits der entsprechenden 14-Bit Adresse; somit ist diese Basisadresse durch Register 5 mal '80h' definiert.

REGISTER 6 definiert die Basisadresse des Generatorunterblocks für die Spritemuster. Wie bei Register 4 kann der Wert zwischen 0 und 7 liegen. Entsprechend werden die oberen Bits der Basisadresse festgelegt und man kann sie auf gleiche Weise berechnen.

REGISTER 7 enthält in den oberen 4 Bits den Farbcode für Farbe 1 im Textmodus. Die unteren 4 Bits enthalten den Farbcode für Farbe 0 im Textmodus und die Hintergrundfarbe für alle Modusarten. Die Farbcodes sind der Tabelle 4 zu entnehmen.

Das Statusregister

Der VDP hat ein einziges 8 Bit Statusregister, das durch die CPU abgefragt werden kann. Dieses Register enthält ein interruptabhängiges Flag, das Spriteübereinstimmungsflag, das fünfte Spriteflag und die Nummer dieses Sprites, falls es existiert. Das Format des Registers (Bitaufteilung) ist der Tabelle 2 zu entnehmen. Eine Erläuterung des Inhalts erfolgt später. Dieses Statusregister kann jederzeit gelesen werden, wenn man sich Klarheit über die F, C und 5S Statusbits verschaffen will. Jedoch ist zu beachten, daß bei jedem Lesen das Interruptflag F gelöscht wird. Deshalb sollte dieses Register nur gelesen werden, wenn ein VDP Interrupt bevorsteht.

Das Interruptflag (F)

Das F-Statusflag im Statusregister wird am Ende des letzten Rasterimpulses der letzten aktiven Displayzeile auf '1' gesetzt. Es wird auf '0' gelöscht, wenn das Statusregister gelesen wird oder der VDP einen Systemreset erhält. Falls das IE-Bit im VDP Register 1 aktiv ist ('1'), so ist der VDP Interruptausgang (INT) solange aktiv (low), wie das F-Statusflag auf '1' ist.

Übereinstimmungsflag (C)

Das C-Statusflag im Statusregister wird auf '1' gesetzt, falls zwei oder mehr Sprites sich überlappen. Überlappung tritt auf, wenn mindestens zwei Sprites auf dem Bildschirm einen oder mehrere identische Punkte haben. Transparente Farbsprites werden dabei ebenso berücksichtigt, wie diejenigen, die ganz oder teilweise außerhalb der Bildfläche liegen. Sprites jenseits des Spriteattributbegrenzers (D016h) werden nicht berücksichtigt. Das Löschen des Registers auf '0' erfolgt unter den gleichen Bedingungen wie beim F-Flag.

Fünftes Spriteflag (5S) und seine Nummer

Das 5S Statusflag wird auf '1' gesetzt, falls 5 oder mehr Sprites auf einer Linie liegen (Linien von 0 bis 192) und der Wert der Rahmenflags '0' ist. Die Löschbedingungen sind die gleichen wie bei F- und C-Flag. Die Nummer des fünften Sprites in einer Linie wird durch die unteren 5 Bits des Statusflags angegeben. Sie erscheint dort, sobald 5S aktiv ('1') wird, und bleibt bis zum Löschen des 5S-Flags erhalten. Zu beachten ist, daß das Setzen des 5S-Flags keinen Interrupt erzeugt.
Der VDP arbeitet mit 262 Zeilen pro Rahmen und etwa 60 Rahmen pro Sekunde bei nicht unterbrochenem Arbeitsmodus.

TABELLE 3

Parameter	Punkt Taktzyklen	
Horizontal	Muster/Mehrfarb	Text
horizontal aktive Anzeige	256	240
rechter Rand	15	25
rechts löschen	8	8
horizontale Sync.	26	26
links löschen	2	2
Farbbruch	14	14
links löschen	8	8
linker Rand	13	19
	342	342
Vertikal	Linien	
vertikal aktive Anzeige	192	
unterer Rand	24	
unten löschen	3	
vertikale Sync.	3	
oberes löschen	13	
oberer Rand	27	
	262	

Videodisplay Modi

Der VDP erzeugt ein Bild auf dem Bildschirm, das am besten durch einzelne Bildebenen, die übereinander gelegt werden können, dargestellt werden kann. Figur 2 zeigt die Bedeutung der einzelnen Ebenen. Objekte, auf Ebenen näher zum Betrachter, haben die höhere Priorität. In dem Fall, daß zwei Belegungen auf unterschiedlichen Ebenen den gleichen Punkt ansprechen, wird die Belegung der höherwertigen Ebene zur Bilderzeugung benutzt. Damit die Belegung einer bestimmten Ebene auf jeden Fall zur Bilderzeugung dient, müssen alle Ebenen vor dieser Ebene an diesem Punkt transparent sein. Die ersten 32 Ebenen können jeweils einen Sprite enthalten. (Sprites sind vom Anwender definierbare Bilder, deren Position im Bild durch vertikale und horizontale Koordinaten im VRAM festgehalten werden.) Die einzelnen Ebenen sind bis auf die Sprites selber, transparent. Da die Koordinaten der Sprites Punktkoordinaten sind, können die Sprites sehr einfach und genau über den Bildschirm bewegt werden. Sprites sind in folgenden Dimensionen verfügbar: 8x8 Punkte, 16x16 Punkte und 32x32 Punkte. Hinter den Spriteebenen liegt die Musterebene, die für Text oder Grafikbilder vorgesehen ist, die durch Text, Grafik1, Grafik2 oder Mehrfarbmodus erzeugt werden. Hinter der Musterebene liegt die Hintergrundebene, die größer als die anderen Ebenen ist, so daß auf dem Bildschirm ein Rahmen um diese erscheint. Die letzte und niederrangigste Ebene ist die externe Videoebene. Das in ihr gespeicherte Bild wird durch den externen Videoeingang erzeugt.

Der Hintergrund besteht aus einer einzigen Farbe, die für den Rahmen und die durchgängigen Lücken in den einzelnen Ebenen verwendet wird. Diese Hintergrundfarbe wird in Register 7 des VDP gespeichert. Wenn das Hintergrundfarbenregister den Transparent-Code enthält, wird der Hintergrund automatisch als 'schwarz' gewählt, und der externe Videomodus ist nicht aktiviert.

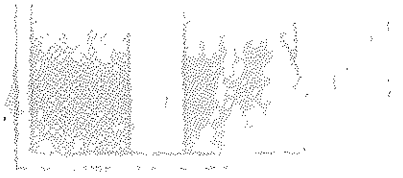
Die 32 Spriteebenen werden in den Mehrfarb- und Grafik-Modi mitverwendet. Im Textmodus werden sie nicht angesprochen und sind automatisch transparent. Jeder Sprite kann eine Fläche von 8x8, 16x16 oder 32x32 Punkten in seiner Ebene bedecken. Jede nicht bedeckte Stelle der Ebene ist transparent. Auch können die Sprites selbst oder Teile von ihnen transparent sein. Sprite 0 befindet sich auf der hochrangigsten Ebene und Sprite 31 auf der direkt vor der Musterebene. Wannimmer ein Punkt einer Spriteebene transparent ist, kann an dieser Stelle die Farbe der nächsten Ebene, falls nicht auch transparent, gesehen werden. Ist ein Punkt jedoch nicht transparent, so geht die Information der nachfolgenden Ebenen an dieser Stelle verloren.

Bei der Zusammensetzung von Bildern aus einzelnen Sprites muß beachtet werden, daß sich in jeder horizontalen Linie höchstens 4 Sprites befinden dürfen. Befinden sich weitere Sprites in einer Linie, werden sie für diese Linie oder Teile von ihnen das Übereinstimmungsflag (C) nicht aktivieren. Die Kontrolle über eine solche Überfüllung einer Linie mit Sprites erfolgt mit dem 5S-Flag und der Nummer des fünften Sprites in einer Linie (VDP Statusregister).

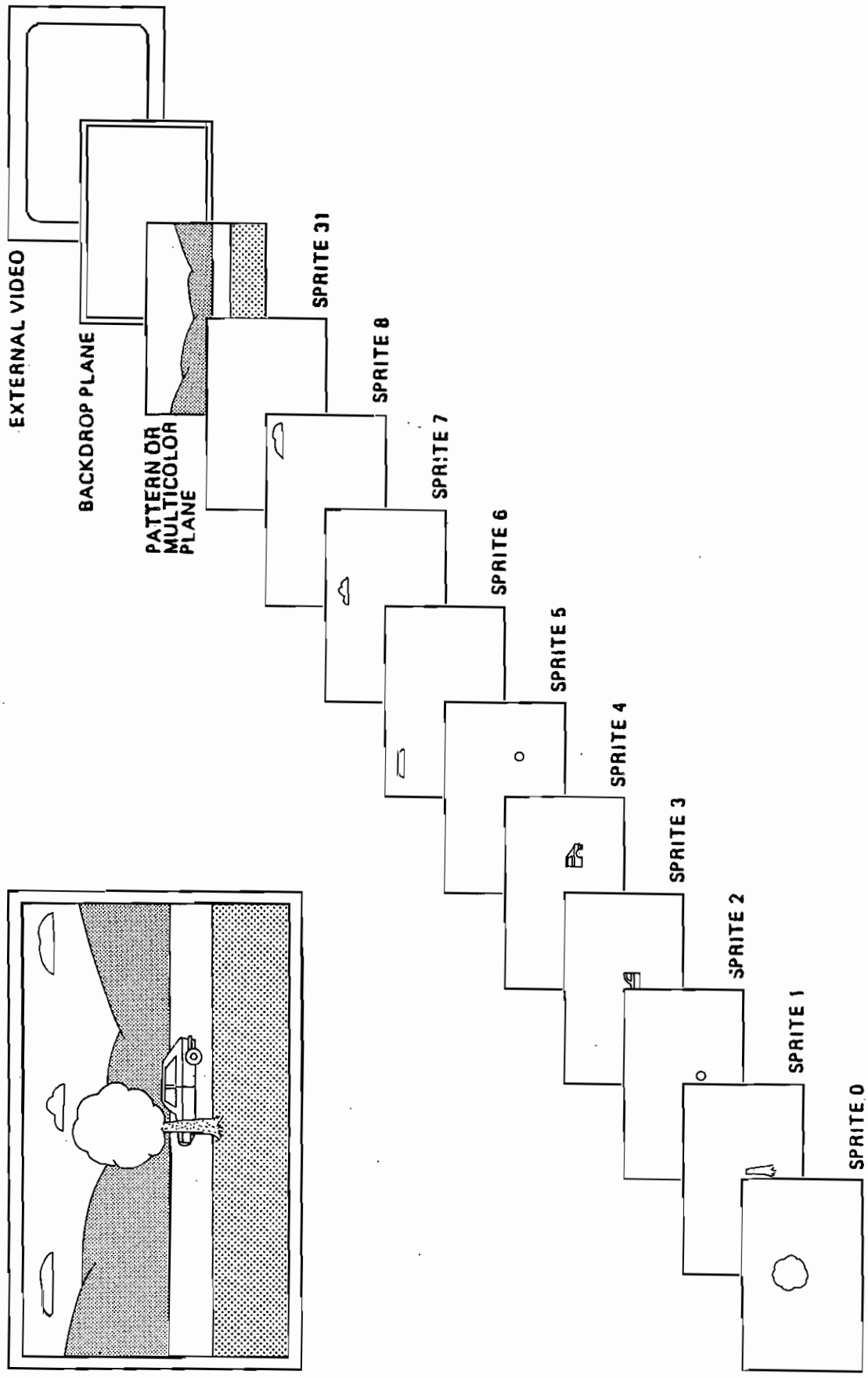
Die Musterebene wird anders als die Spriteebenen, auch noch im Text-Modus angesprochen. Sie wird zur Darstellung der Grafikmuster von Charakterzeichen benutzt. Falls ein Punkt dieser Ebene nicht transparent ist, wird die Hintergrundfarbe automatisch durch die Musterebenenfarbe ersetzt, und nur, wenn hier ein Punkt transparent ist, kann auf dem Bildschirm die Hintergrundfarbe erscheinen.

Der VDP verfügt über drei Farbvideobild Modi und einem Textmodus, mit denen die Musterebene gestaltet werden kann: Grafik1 Modus, Grafik2 Modus, Textmodus und Mehrfarbmodus. Grafik1 und Grafik 2 bewirken dabei, daß die Ebene in Felder von 8x8 Punkten, den Musterpositionen, aufgeteilt wird. Da das ganze Bild aus 256x192 Punkten besteht, ergeben sich 32x24 dieser Felder in diesem Modi. Im Grafik1 Modus können 256 dieser 768 Positionen mit je zwei Farben pro Feld definiert werden. Im Grafik2 Modus können durch ein spezielles Überlagerungsschema alle 768 Felder definiert werden. In diesem Modus können dann pro Felderzeile zwei Farben für eine Felddefinition gewählt werden. Damit können alle 15 Farben und transparent in einer einzigen Feldposition verwendet werden. Im Textmodus wird die Ebene in Felder von 6x8 Punkten zerlegt, die man als Textpositionen bezeichnet. In der Ebene befinden sich 40x24 solcher Positionen. Im Textmodus können Sprites nicht zur Darstellung gebracht werden, und nur zwei Farben sind für den ganzen Bildschirm möglich. Im Mehrfarbmodus wird die Ebene in 64x48 Positionen aufgeteilt, von denen jede 4x4 Punkte groß ist. Pro Feld ist hier eine Farbe erlaubt.

Die VDP-Register definieren die Basisadressen für mehrere Unterblöcke im VRAM. Diese Unterblöcke enthalten Tabellen, die zur Erzeugung des Bildes auf dem Sichtgerät notwendig sind. Die Mustern amentabelle, die Mustergeneratortabelle und die Spritegeneratortabelle werden benutzt, um die Bilder der Sprites sichtbar zu machen. Der Inhalt dieser Tabelle muß durch den Mikroprozessor erzeugt werden. Bewegung kann erreicht werden, wenn die Werte im VRAM im Echtzeitbetrieb verändert werden.



Figur 2 - VDP Bildebenen



Der VDP kann 15 Farben und transparent darstellen (s. Tabelle 4). Auf Schwarz-/Weißgeräten erscheinen hier 8 unterschiedliche Grautönungen. Der Helligkeitsgrad ist ebenfalls aus der Tabelle zu ersehen; 0,00 bedeutet dabei schwarz und 1,00 weiß. Falls einmal alle Ebenen in einem Punkt transparent sein sollten, erscheint dieser Punkt in der Anzeige als schwarz.

Tabelle 4 - Farbtabelle

Farbe (HEX)	Farbe	Helligkeit (DC Wert)	Farbgrad (AC Wert)
0	transparent	0.00	-
1	schwarz	0.00	-
2	mittel grün	0.60	0.60
3	hell grün	0.80	0.53
4	dunkel blau	0.47	0.73
5	hell blau	0.67	0.60
6	dunkel rot	0.53	0.53
7	cyan	0.80	0.73
8	mittel rot	0.67	0.73
9	hell rot	0.80	0.73
A	dunkel gelb	0.87	0.53
B	hell gelb	1.00	0.40
C	dunkel grün	0.47	0.60
D	magenta	0.60	0.47
E	grau	-	-
F	weiß	1.00	-

Grafik 1 Modus

Der VDP ist im Grafik 1 Modus, wenn die Bits M1, M2 und M3 in den VDP Registern 1 und 0 den Wert '0' haben. Im Grafik 1 Modus wird die Musterebene in 32 Spalten und 24 Reihen aufgeteilt. Jede dieser Musterpositionen besteht aus 8x8 Punkten. Im VRAM wird eine Tabelle angelegt, um diese Musterebene zu generieren. Insgesamt 2848 Bytes VRAM werden benötigt für den Masternamen, die Farben und die Generatortabelle. Jedoch kann Speicherplatz eingespart werden, wenn nicht alle Musterpositionen belegt werden müssen. Dadurch können die einzelnen Tabellen näher zusammenrücken und im VRAM entsteht Platz für weitere Speicheraufgaben.

Die Mustergenerator-Tabelle enthält ein Verzeichnis von Mustern, die in den einzelnen Musterpositionen angezeigt werden können. Diese Tabelle ist 2048 Bytes lang und in 256 Muster aufgeteilt, von denen jedes 8 Bytes (8x8 Bit) umfaßt. Jede '1' in diesen Bytes stellt Farbe 1 dar, und jede '0' die zweite Farbe 0.

Zur Auswahl einer Musterdefinition aus der Generatortabelle wird ein 8 Bit Mustername benötigt. Die Basisadresse der Generatortabelle liegt bei einer 2k Grenze des VRAM. Die Basisadresse der Musterdefinitionstabelle wird durch Register 4 des VDP bestimmt. Dieses Register legt die oberen drei Bits der Tabelle fest. Die folgenden 8 Bits werden durch den Namen der ausgewählten Musterdefinition festgelegt und die letzten 3 Bits der Adresse enthalten die Reihennummer in der Musterdefinition.

Für jede der 8x8 Musterdefinitionen von den 256 möglichen werden 8 Bytes benötigt. Das erste Byte definiert dabei die erste Zeile des Musters, das zweite die zweite Zeile usw. Das erste Bit eines solchen Bytes definiert dann die erste Spalte des Musters, das zweite die zweite Spalte usw. Dabei wird zwischen zwei Farben ausgewählt. Eine '1' repräsentiert dabei Farbe 1 und eine '0' die Farbe 0. Farbe 1 wird dabei durch die oberen 4 Bits des Farbregisters (VDP Register 7) und Farbe 0 durch die unteren festgelegt. Als Beispiel hierfür soll folgende Grafik dienen:

Zeile/Byte	Spalte 0 1 2 3 4 5 6 7	Bit 0 1 2 3 4 5 6 7
0	* * * * *	0 1 1 1 1 1 0 0
1	* *	0 0 0 0 1 1 0 0
2	* *	0 0 0 0 0 1 1 0
3	* * * *	1 1 0 0 0 1 1 0
4	* *	0 0 0 0 1 1 0 0
5	* *	0 0 0 0 1 1 0 0
6	* * * * *	0 1 1 1 1 1 0 0
7	*	0 0 0 1 0 0 0 0
	(Muster)	(Musterdefinition)

Die Farbe wird in der Musterfarbtabelle abgelegt, die 32 Einstiegspunkte hat, von denen jeder 1 Byte groß ist. Jedes dieser Bytes speichert zwei Farben (Farbe 1 in den oberen 4 Bits und Farbe 0 in den unteren). Das erste dieser Bytes definiert die Farben für die Muster 0 bis 7, das zweite die Muster 8 bis 15 usw. Die Farbwerte in Hexform sind Tabelle 4 zu entnehmen. Somit können in diesem Grafikmodus 32 unterschiedliche Farbpaare in einem Bild dargestellt werden.

Die Musternamentabelle wird in einem 768 Byte großen Block im VRAM ab einer 1k Grenze abgespeichert. Die Basisadresse wird durch 4 Bits des VDP Registers 2 festgelegt. Dies bestimmt die obersten 4 Bit der 14-Bit Adresse. Die anderen 10 Bits werden durch den Zeilen- und Spaltenzähler festgelegt.

Jeder Wert in der Musternamentabelle bezieht sich auf eine Definition in der Mustergeneratortabelle. Die oberen 5 Bits legen dabei die Farbgruppe des jeweiligen Musters fest. Es gibt 32 Farbgruppen zu je acht Mustern. Für alle Muster einer solchen Gruppe sind nur zwei Farben möglich. Die Farbtabelle ist ebenfalls im VRAM abgespeichert, umfaßt 32 Bytes und beginnt an einer 64-Byte-Grenze. Die Basisadresse wird dabei durch Register 3 des VDP bestimmt. Dieses bildet die oberen 8 Bits der 14-Bit-Adresse im VRAM. Das nächste Bit der Adresse ist dann '0' und die restlichen 5 Bits sind gleich den oberen 5 Bits entsprechend der Namenstabelleneingaben.

Da die Basisadressen all dieser Tabellen durch VDP Register festgelegt werden, können durch Ändern dieser Registerwerte sehr schnell und einfach neue Grafikbilder erzeugt werden. Dies ist besonders bei der Darstellung von Bewegungen sehr vorteilhaft.

Wenn die Mustergeneratortabelle mit einem Mustersatz geladen ist, kann durch Veränderung der Musternamenstabelle auf einfache Weise eine Veränderung des angezeigten Bildes erreicht werden. Doch auch durch Änderung des Mustersatzes können solche Veränderungen des Bildes hervorgerufen werden. Die Durchführung beider Methoden ist sehr einfach, da alle notwendigen Tabellen im VRAM enthalten sind, und bei jeder Bildausgabe diese Tabellen neu interpretiert werden.

Bei Ausgabe von Text soll zunächst der Zeichensatz in den Mustergenerator geladen werden. Dabei erweist es sich als praktisch, wenn die Musternummer mit dem 8-Bit-Wert des geladenen Zeichens übereinstimmt (z.B. für 'A' wählt man die Musternummer 4116). Als nächstes lädt man die Musterfarbtabelle mit den gewünschten Werten. Um nach diesen Vorbereitungen einen Text anzeigen zu können, müssen nun nur noch die 8-Bit-Werte der gewünschten Zeichen in der geplanten Reihenfolge in der Musternamenstabelle abgespeichert werden.

Außer Text können aber auch Grafikbilder mit der Musterebene zur Anzeige gebracht werden. Um einen Gegenstand darzustellen, der nicht größer als 8x8 Punkte ist, muß nur ein einziges Musterfeld definiert werden. Größere Abbildungen müssen durch mehrere Musterfelder zusammengesetzt werden. Dazu muß das Bild in Felder von 8x8 Punkten zerlegt werden. Man beachte, daß für die Darstellung von schnellen Bewegungen nur die Einstiegspunkte in einzelne Tabellen verändert werden müssen.

Tabelle 5 - Musterfarbentabelle

Byte Nr.	Muster Nr.
0	0..7
1	8..15
2	16..23
3	24..31
4	32..39
5	40..47
6	48..55
7	56..63
8	64..71
9	72..79
10	80..87
11	88..95
12	96..103
13	104..111
14	112..119
15	120..127
16	128..135
17	136..143
18	144..151
19	152..159
20	160..167
21	168..175
22	176..183
23	184..191
24	192..199
25	200..207
26	208..215
27	216..223
28	224..231
29	232..239
30	240..247
31	248..255

Insgesamt werden 2848 Bytes des VRAM für die Abspeicherung der Muster-, Namens- und Generatortabelle benötigt. Weniger Platz wird gebraucht, wenn nicht alle Musterfelder definiert werden. Die einzelnen Speicherblöcke können dann kleiner werden, und man hat durch die VDP Register die Möglichkeit, die einzelnen Bereiche enger zusammenzuschieben. Der freiwerdende Platz kann dann für andere Aufgaben genutzt werden.

Grafik 2 Modus

Der VDP befindet sich im Grafik 2 Modus, wenn die Bits M1 und M2 '0' sind und M3 '1' ist. Der Grafik 2 Modus ist ähnlich aufgebaut wie der Grafik 1 Modus, nur stehen für die einzelnen Musterfelder mehr Definitionsmöglichkeiten zur Verfügung. Die Folge ist, daß jedes der 768 (32x24) Musterfelder einen eigenen Zugang zum Mustergenerator haben muß. Auch können hier mehr Farbkombinationen in den einzelnen Musterfeldern erzielt werden. Hier stehen nämlich, für jedes Byte eines 8x8 Musters, zwei Farben zur Verfügung. Mit diesen Möglichkeiten steigt natürlich der Platzbedarf im VRAM erheblich und kann bis zu 12k Bytes bei voller Ausnutzung des Grafik 2 Modus betragen.

In gleicher Weise wie im Grafik 1 Modus ist im Grafik 2 Modus die Musternamentabelle aufgebaut. Da im Grafik 1 Modus die Musternamen nur 8 Bit lang sind, können maximal 256 verschiedene Musterdefinitionen verwendet werden. Im Grafik 2 Modus hingegen, wird der Bildschirm in drei gleichgroße Teile von je 256 Musterpositionen aufgeteilt, und ebenso die Mustergeneratortabelle, von der jeder der drei Teile 2048 Bytes umfaßt. Musterdefinitionen im ersten Teil entsprechen dabei dem oberen Drittel des Bildschirms und entsprechend der zweite und der dritte Teil dem zweiten und dritten Drittel des Bildschirms. Die Musternamenstabelle ist ebenfalls in drei Teile zu je 256 Bytes aufgeteilt, wobei auch hier Namen aus der ersten Tabelle den Musterdefinitionen aus dem ersten Teil der Mustergeneratortabelle entsprechen. In gleicher Weise verläuft die Entsprechung in den zweiten und dritten Teilen. Somit werden, falls die 768 Musterfelder nur einmal festgelegt sind, alle 8-Bit Musternamen dreimal verwendet, und zwar einmal in jedem Teil der Musternamentabelle. Die Mustergeneratortabelle beginnt an einer 8k-Grenze des VRAM und kann deshalb nur im unteren oder oberen Teil des 16k Bereichs liegen. Die Wahl erfolgt dabei durch das höchstwertigste Bit des VDP Registers 4. Alle niederwertigeren Bits müssen auf '1' gesetzt werden.

Die Farbtabelle ist ebenfalls 6144 Bytes lang und in drei gleiche Blöcke zu je 2048 Bytes aufgeteilt. Jeder Einstieg in diese Tabelle ist 8 Bytes lang, so daß zu jedem Byte in einem Musterfeld zwei Farben definiert werden können. Das Adressierungsschema ist hier das gleiche wie bei der Mustergeneratortabelle, nur daß statt Register 4 das Register 3 des VDP verwendet wird.

Der Mehrfarbmodus

Der VDP befindet sich im Mehrfarbmodus, wenn M1 und M3 '0' sind und M1 '1' ist. Dieser Modus erzeugt eine Ebene mit 64x48 Farbfeldern. Jedes dieser Felder besteht aus 4x4 Punkten. Als Farbe für jedes dieser Farbfelder kann eine der 15 möglichen oder transparent gewählt werden.

Somit können hier alle Farben bei einer Auflösung von 4x4 Punkten Anwendung finden. Die Hintergrund- und die Spriteebene werden durch diesen Modus nicht beeinflußt.

Die Mehrfarbnamentabelle ist die gleiche wie im Grafikmodus und hat 768 Einstiegspunkte. Die Namen zeigen jetzt aber nicht mehr auf eine Farbtabelle, sondern die Farbe wird jetzt von der Mustergeneratortabelle übermittelt. Die Namen zeigen hier auf ein 8-Byte Segment des VRAM in der Mustergeneratortabelle.

Gebraucht werden von diesem 8-Byte Segment nur 2 Bytes. Diese beiden definieren zusammen 4 Farben, die auf ein 8x8-Punkt-Muster verteilt werden. Die obersten 4 Bits des ersten Bytes definieren dabei das obere linke Viertel und die unteren 4 Bits das obere rechte Viertel des 8x8-Punkte Feldes. Entsprechend werden durch das zweite Byte die beiden unteren Viertel definiert.

Die Lage dieser 2 Bytes in dem 8-Byte Segment, die von dem Musternamen angesprochen werden, ist abhängig von der Bildschirmposition, die angesprochen werden soll. Für Namen in der oberen Zeile des Bildes (Name 0 bis 31), sind die beiden Bytes die ersten beiden in dem jeweiligen 8-Byte Segment. Die Namen 32 bis 63 benutzen dann das zweite und dritte Byte, die Namen von 64 bis 95 das vierte und fünfte Byte und die Namen von 96 bis 127 die letzten beiden Bytes des 8-Byte Segments. Dieses Aufteilungsprinzip gilt auch für die weiteren Teile des Bildes.

Die Darstellung des VRAM-Inhaltes auf dem Bildschirm wird dadurch vereinfacht, daß Namen mehrfach in der Namenstabelle verwendet werden. Da die Byteserien, die im 8-Byte Segment verwendet werden, sich alle 4 Zeilen wiederholen, kann alle 4 Zeilen in der gleichen Spalte der gleiche Name verwendet werden. Dadurch definiert dann jedes 8-Byte Segment 2x8 Musterquadrate auf dem Bildschirm als eine geradeaus-vorwärts-Übersetzung des 8-Byte VRAM Segments, auf das der gemeinsame Name zeigt.

Bei einem Aufbau in dieser Weise werden weiterhin 768 Bytes für die Namenstabelle und 1536 Bytes als Farbinformation in der Mustergeneratortabelle benötigt (24 Reihen x 32 Spalten x 8 Bytes/Musterposition). Zusammen werden also 1728 Bytes des VRAM belegt; es muß jedoch beachtet werden, daß die Basisadressen bei den 1k- und 2k-Übergängen des VRAM liegen, und die Tabellen somit nicht gepackt werden können.

Der Text-Modus

Der VDP befindet sich im Text-Modus, wenn M2 und M3 '0' sind und M1 '1' ist. Im Text-Modus wird der Bildschirm in 24 Reihen zu 40 Spalten aufgeteilt. Jede Textposition ist 6 Punkte breit und 8 Punkte hoch. Die benötigten Tabellen zur Erzeugung der Musterebene sind die Musternamentabelle und die Mustergeneratortabelle. Es ist möglich, bis zu 256 verschiedene Muster zu erzeugen. Die Musterdefinitionen sind in der Mustergeneratortabelle im VRAM gespeichert und können jederzeit geändert werden. Ebenfalls im VRAM ist die Musternamentabelle enthalten, in der die Musterdefinition aller 960 Musterfelder enthalten sind. Zu beachten ist, daß Sprites im Text-Modus nicht verfügbar sind.

Die Reihe bei den Namen der einzelnen Muster beginnt oben links auf dem Bildschirm mit Nummer 0, wird dann zeilenweise fortgesetzt (Zeile 1 endet mit Nummer 39), und endet mit Nummer 959 rechts unten auf dem Bildschirm.

Wie im Grafik-Modus enthält die Mustergeneratortabelle ein Verzeichnis der Textmuster (normalerweise alphanumerische Zeichen, es können aber auch eigene Muster im 6x8 Raster gewählt werden), die in den Textpositionen angezeigt werden können. Die Generatortabelle ist 2048 Bytes lang und in 256 Textmuster zu je 8 Bytes aufgeteilt. Da jedes Muster in diesem Modus nur 6 Punkte breit ist, werden die beiden niederwertigsten Bits der einzelnen Bytes ignoriert. Jedes dieser 8-Byte Segmente definiert ein Muster, wobei eine '1' im Byte die Farbe 1 definiert und eine '0' Farbe 2. Die ausgewählten Farben werden durch das VDP Register 7 festgelegt.

Da die Reihenfolge der Musternamen mit der Reihenfolge der ausgegebenen Zeichen übereinstimmt, muß der Mustername nur den entsprechenden Wert zur Mustergeneratortabelle enthalten. Bei 960 Bildschirmpositionen muß die Musternamentabelle somit 960 Bytes lang sein. Sie wird als zusammenhängender Block ab einer 1k Grenze im VRAM abgespeichert. Die Startadresse dieser Namenstabelle wird durch 4 Bits des VDP Registers 2 bestimmt. Sie legen die oberen 4 Bits der 14-Bit Adresse fest. Die weiteren 10 Bits zeigen auf einen der 960 Speicherplätze im VRAM, in dem einer der Musternamen abgelegt ist. Die Namenstabelle ist zeilenweise organisiert. Gegenüber den anderen, bereits besprochenen Betriebsarten des VDP, kann im Text-Modus für die ganze Seite nur zwischen zwei Farben gewählt werden.

Wie der Name schon sagt, ist dieser Modus insbesondere für die Textverarbeitung vorgesehen. Man muß sich nun entscheiden, ob man den Text im Grafik-Modus mit 32 Zeichen/Zeile und einer Vielzahl von Farbmöglichkeiten, oder im Text-Modus mit 40 Zeichen/Zeile und nur zwei Farben darstellen will. Die Entscheidung hierüber muß vom jeweils anstehenden Problem abhängig gemacht werden. Jedoch kann man mit einiger Programmiersorgfalt erreichen, daß der gleiche Zeichensatz (Mustergeneratortabelle) sowohl im Text- als auch im Grafik 1 Modus verwendet werden kann.

Dabei ist dann besonders darauf zu achten, daß die letzten beiden Bits jedes Bytes in der Mustergeneratortabelle den Wert '0' haben. Ein Umschalten vom Text- in den Grafikmodus bewirkt dann, daß die Abstände zwischen den Zeichen vergrößert werden (2 Punkte) und die Anzahl der Zeichen von 40 auf 32 Zeichen/Zeile reduziert wird.

Wie im Grafikbereich kann auch bei der Wahl des Text-Modus die Anzeige durch Änderung der Namenstabelle oder der Generatortabelle erreicht werden.

Der Speicherplatzbedarf für die Mustergeneratortabelle beträgt bei 256 Mustern 2048 Bytes, die im VRAM ab einer 2k Grenze abgelegt werden. Die Basisadressen dieser Tabelle wird durch das VDP Register 4 festgelegt, das die obersten 3 Bits der 14-Bit VRAM Adresse festlegt. Die nächsten 8 Bits entsprechen dem 8-Bit Namen der ausgewählten Musterdefinition. Die letzten 3 Bits der VRAM Adresse entsprechen der Zeilennummer in der Musterdefinition.

Für jedes der 256 möglichen Zeichen von 6x8 Punkten wird ein Block von 8 Bytes benötigt. Die Aufteilung der einzelnen Bytes für die Bilderzeugung ist dabei die gleiche wie im Grafik-Modus, nur daß die letzten beiden Bits jeweils nicht verwendet werden. Wegen der Umschaltmöglichkeit zwischen Grafik- und Text-Modus wird empfohlen, diese auf '0' zu setzen. Wie im Grafik-Modus kann mit den '0' und '1' Werten zwischen zwei Farben gewählt werden. Da hier jedoch für den ganzen Bildschirm nur zwei Farben zur Verfügung stehen, braucht keine gesonderte Farbtabelle angelegt werden, sondern zur Farbbestimmung wird jeweils das Farbregister 7 des VDP herangezogen. Eine '0' in einem Definitionsbit wählt dabei die Farbe 0 (untere 4 Bits des Reg. 7).

Für Generator- und Namenstabelle werden somit, bei Ausnutzung aller Darstellungsmöglichkeiten, 3005 Bytes im VRAM benötigt. Auch hier kann in vielen Fällen Platz eingespart werden, insbesondere dann, wenn nicht alle 256 möglichen Musterdefinitionen gebraucht werden. Der freiwerdende Platz kann dann für weitere Tabellen verwendet werden, was besonders wichtig ist, wenn ein Umschalten zwischen Text- und Grafik-Modus gewünscht wird.

Die Sprites

Das Videobild kann auch durch 32 Spriteebenen aufgebaut werden, die in ihrer Priorität über den anderen Ebenen liegen. Der besondere Einsatzpunkt dieser Sprites liegt in der Darstellung von sich überlappenden Bildern und der gleichmäßigen Darstellung von Bewegungen. Das Besondere für die Bewegungsdarstellung ist dabei, daß die Lage des Sprites im Bild durch ein einziges Koordinatenpaar festgelegt wird. Benutzt werden dabei die Koordinaten des links oben liegenden Punktes des Spritemusters. Zur Darstellung der Bewegung brauchen dann nur diese Koordinaten verändert werden. Ein Nachteil der Sprites liegt darin, daß sie nicht im Text-Modus verwendet werden können, jedoch kann hier die Texteinblendung mit 32 Zeichen/Zeile durch den Grafik-Modus erreicht werden. Um das Übereinanderlegen der einzelnen Ebenen zu erleichtern, wird automatisch dafür gesorgt, daß die Ebenen außerhalb der Sprites transparent sind.

Die Unterblöcke des VRAM, die zur Spriteerzeugung gebraucht werden, sind die Spriteattribut- und die Spritegeneratortabelle. Diese Tabellen ähneln den Mustertabellen. Die Attributtabelle definiert dabei die Lage des Sprites und die Generatortabelle die Form des Sprites. Die Spriteformate sind der Tabelle 6 zu entnehmen.

Da 32 Spriteebenen zur Verfügung stehen, gibt es auch 32 Einstiegspunkte in der Spriteattributtabelle. Diese 32 Blöcke bestehen aus jeweils 4 Bytes. Die Einstiegspunkte sind nach den Ebenen geordnet, so daß der erste Einstieg zur Spriteebene 0 gehört. Mit den 32 Blöcken zu 4 Bytes ergibt sich für die Spriteattributtabelle eine Länge von 128 Bytes. Die Tabelle kann an einer 128-Byte-Grenze des VRAM beginnen. Die Basisadresse der Tabelle wird durch das VDP Register 5, das die oberen 5 Bits der 14-Bit Adresse festlegt, bestimmt. Die nächsten 5 Bits der VRAM Adresse entsprechen dann der Nummer der Spriteebene und die letzten 2 Bits bestimmen die einzelnen Teile der 32 Blöcke. Die 4 Bytes eines Blocks bestimmen die Position (2 Bytes), den Musternamen und die Farbe des Sprites.

Tabelle 6 - Spritemuster Format

Größe	Vergrößerung (MAG)	Fläche (in Punkten)	Auflösung	Bytes/Muster
0	0	8x8	Einzelpunkt	8
1	0	16x16	Einzelpunkt	32
0	1	16x16	2x2 Punkte	8
	1	32x32	2x2 Punkte	32

Dabei definieren die ersten beiden Bytes eines Blocks die Position (linke obere Ecke) des Sprites im Bild. Das erste legt hierbei die Entfernung von der oberen Kante des Bildes fest. Es ist so definiert, daß der Wert 1 den Sprite an die obere Grenze bringt, wo er den Hintergrund berührt. Das zweite Byte bestimmt die waagerechte Verschiebung von der linken Kante des Bildes. Der Wert 0 bringt hier den Sprite an die linke Kante des Hintergrundes. Bei all diesen Koordinatenbestimmungen muß unbedingt darauf geachtet werden, daß der linke obere Punkt des Sprites der Bezugspunkt ist.

Wenn ein Sprite sich mit dem Hintergrundrahmen überlappt, werden nur die Punkte des Sprites angezeigt, die innerhalb des Rahmens liegen. Dies hat für den Anwender den Vorteil, daß er Sprites auf dem Rahmen bewegen kann, ohne daß sie im Bild angezeigt werden. Auf dem oberen oder unteren Rand befindet sich der Sprite bei Werten zwischen -31 und 0 bzw. 191 und 207 im ersten Byte des Blocks. Um Sprites auf der rechten Seite des Rahmens bewegen zu können, muß das zweite Byte den Wert 255 enthalten. Für eine Bewegung auf dem linken Teil des Rahmens, muß ein Bit des vierten Bytes verwendet werden.

Das dritte Byte der Spriteattributtabelle enthält den Wert, der auf die Spritegeneratortabelle zeigt. Dieser 8-Bit Wert entspricht dem Musternamen im Grafik-Modus.

Byte 4 enthält dann die Farbdefinition des Sprites. Da je Ebene nur eine Farbe möglich ist, werden von diesem Byte nur die unteren 4 Bits für die Farbdefinition verwendet. Das höchstwertige Bit dieses Bytes kann zum Verschieben des Sprites auf den linken Rand verwendet werden. Hat dieses Bit den Wert '1', so wird der Sprite um 32 Punkte nach links verschoben. Bei der Wahl eines entsprechenden Wertes im zweiten Byte des Blocks kann also der linke Rand erreicht werden.

Die Spritegeneratortabelle ist maximal 2048 Bytes lang, und beginnt an einer 2k Grenze des VRAM. Sie ist in 256 Blöcke zu 8 Bytes aufgeteilt. Das dritte Byte in jedem Attributblock bestimmt, welcher dieser 8-Byte Blöcke angesprochen werden soll. Für die '1' Werte in diesen 8 Bytes wird dann die im vierten Byte des Attributblocks bestimmte Farbe gewählt. Für '0' Werte wird automatisch der Transparent-Modus gewählt. Die Basisadresse des Spritegeneratorblocks wird durch Register 6 des VDP festgelegt, wobei durch dieses Register die oberen 3 Bits der 14-Bit Adresse bestimmt werden. Die nächsten 8 Bits werden durch den Spritenamen (3. Byte im Attributblock) festgelegt. Die letzten 3 Bits entsprechen dann der Zeilennummer des Spritemusters (Auswahl zwischen den 8 Bytes). Diese Verteilung ist bei der Wahl von Größe = 1 ein wenig modifiziert, im Prinzip aber gleich.

Es gibt jedoch bei der Wahl der Lage von Sprites eine kleine Einschränkung. So dürfen nicht mehr als 4 Sprites in einer waagerechten Linie liegen (dies bezieht sich auf den Bezugspunkt der Sprites). Befinden sich 5 oder mehr Sprites in einer Linie, so wird dies durch das Statusregister des VDP signalisiert. In einem solchen Fall wird das 5S Bit gesetzt (1) und die Nummer der Ebene, in der das fünfte Sprite liegt, wird im Statusregister ausgegeben. Das fünfte und möglicherweise weitere Sprites in der Linie können nicht auf dem Bildschirm dargestellt werden. Dabei wird von der vorderen Ebene, mit der größten Priorität, aus gezählt.

Werden größere Sprites als solche mit 8x8 Punkten benötigt, kann dies durch Veränderung der Bits Größe und Vergrößerung (MAG) im VDP Register 1 erreicht werden. Maximal sind Sprites mit 32x32 Punkten erzeugbar. Eine Übersicht über die Möglichkeiten gibt Tabelle 6. Zu beachten ist, daß bei der Wahl von MAG als '1' die Auflösung von einem Punkt auf ein Feld von 2x2 Punkten herabgesetzt wird. Dies ist notwendig, damit für alle Kombinationen der Bits die gleiche Generatortabelle verwendet werden kann.

Neben der 5S-Kontrolle überprüft der VDP die einzelnen Sprites auch auf Übereinstimmung. Übereinstimmung tritt auf, wenn zwei Sprites an der gleichen Position des Bildschirms die gleiche Form haben. Dadurch wird die Darstellung nicht beeinflußt (bis auf die unterschiedliche Priorität der Ebenen), aber das C-Bit im Statusregister des VDP wird auf '1' gesetzt und kann von der CPU ausgewertet werden.

Die Spritedarstellung wird beendet, falls der Wert '206' (D016) in dem vertikalen Positionierungsbyte eines der Sprites in der Attributtabelle findet. Dies erlaubt es einem, die Spriteattributtabelle auf eine minimale Größe zu kürzen. Zudem kann ein Teil der Spriteebenen, oder auch alle, durch Änderung eines einzigen Bytes ausgeblendet werden.

Bei Ausnutzung aller Spritemöglichkeiten wird ein Speicherbereich von 2176 Bytes im VRAM benötigt. Wie in den anderen Betriebsarten, kann dieser Bereich bei Einschränkung der Möglichkeiten verkleinert werden und die beiden Tabellen können zusammengeschoben werden.

Betrachtet man den Speicherbedarf für die einzelnen Betriebsarten, so stellt man fest, daß bei einem Speicherangebot von 16k VRAM es durchaus möglich ist, sogar den speicherintensiven Grafik 2 Modus mit dem Spritemodus zu kombinieren. Daraus ergibt sich eine Vielzahl von Darstellungsmöglichkeiten, über die selbst Rechner gehobeneren Preises nur sehr selten verfügen. Neben den Hardwaremöglichkeiten bietet der MTX, durch die Aufteilung in einzelne Bildebenen, zudem den Vorteil einer einfachen Softwaresteuerung.

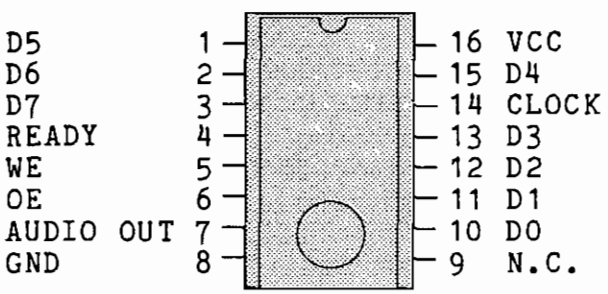
7. Der Tongenerator

Bei dem im MTX verwendeten Tonprozessor handelt es sich um den Texas Instruments Chip SN 76489 A. Dieses Bauteil hat hier folgende Ein- und Ausgabemöglichkeiten:

Über Port 6 der CPU werden die Daten ausgegeben (OUT (6),Data)
Über Port 3 der CPU erfolgt der Strobe für die Datenübernahme (IN (3))

Um die Daten also zum Tongenerator zu übertragen, muß über Port 6 die gewünschte Date ausgegeben werden und ein Dummy-Lesebefehl (der gelesene Wert wird von der CPU nicnt verwendet) besorgt dann die Übernahme in die Register des Tongenerators. Damit der Baustein genügend Zeit zur Verarbeitung der Daten hat, darf der nächste Übernahmeimpuls erst nach 32 Taktzyklen erfolgen.

Das folgende Bild gibt eine Übersicht der Anschlußbelegung des SN 76489 A (Ansicht von oben).



Kurzbeschreibung

Der SN 76489 A ist ein komplexer digitaler Tongenerator in I'2 L/Bipolar Technologie. Er enthält einen preiswerten Ton-und Geräuschgenerator, der speziell für die Kombination mit Mikroprozessoren entwickelt wurde. Das Bauteil wird über einen Datenbus von der Zentraleinheit aus angesprochen. Die wesentlichen Anschlußdaten sind:

Vcc	5V + -10%
VOH (Pin 4)	5,5V max.
IO1 (Pin 4)	2mA max.
Arbeitstemperatur	0 Grad - 70 Grad C

FUNKTIONSWEISE

1) Der Tongenerator

Jeder Tongenerator besteht aus einem frequenzerzeugenden und einem verstärkenden Teil. Die Frequenzerzeugung benötigt eine 10-Bit Dateninformation (F9 - F0), um die Hälfte der Periode der gewünschten Frequenz (n) festzulegen. F9 ist dabei das höchstwertige und F0 das niederwertigste Bit. Die Information wird in einen 10-stufigen Tonzähler geladen, der mit einer Frequenz von N/16 (N = Eingangstaktfrequenz) herabgezählt wird. Wenn dieser Zähler den Wert '0' hat, wird ein Überlaufsignal erzeugt. Dieses Signal schaltet das Frequenzflipflop und lädt den Zähler mit dem vorgegebenen Wert. Die Ausgangsfrequenz läßt sich deshalb nach folgender Formel bestimmen:

$f=N / 32n$ mit N=Taktfrequenz in Hz und n=10-Bit-Binärzahl

Der Ausgang des Frequenzflipflops wird an einen 4-stufigen Verstärker (Verstärkung < 1) weitergeleitet. Die Abhängigkeit der Verstärkung von dem übermittelten Verstärkungswort ist aus Tabelle 1 zu entnehmen. Dabei sei darauf hingewiesen, daß auch andere Bitkombinationen möglich sind, und somit eine maximale Verstärkung von -28 db erreicht werden kann.

Tabelle 1 -Verstärkungskontrolle-

Bitposition				Verstärkung in db
A3	A2	A1	A0	(Abschwächung)
0	0	0	1	-2
0	0	1	0	-4
0	1	0	0	-8
1	0	0	0	-16
1	1	1	1	aus

2) Der Geräuschgenerator

Der Geräuschgenerator besteht aus einer Rauschquelle und einem Verstärker. Die Rauschquelle ist ein Schieberegister mit einer EXOR-Netzwerk-Rückkopplung. Die Rückkopplung ist notwendig, damit das Schieberegister nicht in einer Nullstellung hängen bleibt.

Tabelle 2 -Rausch-Rückkopplungskontrolle-

FB	Funktion
0	periodisches Rauschen
1	weißes Rauschen

Bei jeder Änderung des Rauschkontrollregisters wird das Schieberegister gelöscht. Das Schieberegister wird mit einem Viertel der Rate weitergeschoben, die durch die beiden NF-Bits festgelegt wird. Die festen Schieberaten werden durch den Eingangstakt bestimmt.

Tabelle 3 -Rauschgenerator-Frequenz-Kontrolle-

Bits NF1	NF0	Schieberate
0	0	N/512
0	1	N/1024
1	0	N/2048
1	1	Tongenerator "3 Ausgabe

Auch die Verstärkung des Rauschgenerators ist softwaremäßig steuerbar. Eine Übersicht über die Ansteuerungsmöglichkeiten gibt Tabelle 4.

3) Ausgangspuffer/Verstärker

Der Ausgangspuffer ist ein konventioneller Operationsverstärker in Addierschaltung. Er addiert die drei Tonsignale und das Rauschsignal. Der Ausgangsstrom kann maximal 10mA betragen.

4) Interface von der CPU zum SN 76489 A

Das Interface zwischen CPU und Tonprozessor besteht aus 8 Datenleitungen und 3 Steuerleitungen (WE, CE und READY). Jeder Tongenerator benötigt eine 10-Bit Information zur Frequenzbestimmung und eine 4-Bit Information zur Festlegung der Verstärkung. Für eine Frequenzfestlegung wird deshalb eine 2-Byte und für die Verstärkungsfestlegung eine 1-Byte Datenübertragung benötigt.

Falls nur ein Tongenerator angesprochen wird, braucht, nach der Übertragung der ersten drei Bytes (Frequenz und Verstärkung), nur noch das zweite Byte der Frequenzbestimmung übertragen werden, da die Registernummer im Chip gespeichert wird. Dadurch können die oberen 6 Bits für die Frequenzbestimmung schnell verändert werden.

5) Kontrollregister

Der SN 76489 A verfügt über 8 interne Register, die die Ton und Rauscherzeugung kontrollieren. Bei allen Datenübertragungen zum Chip muß das erste Byte einen 3-Bit-Wert enthalten, der bestimmt, welches Register angesprochen werden soll. Eine Übersicht der Kontrollregister-Codes gibt Tabelle 4.

Tabelle 4 -Registeradressierungsbits-

R2	R1	R0	Ausgewähltes Register
0	0	0	Tone 1 Frequenz
0	0	1	Tone 1 Verstärkung
0	1	0	Tone 2 Frequenz
0	1	1	Tone 2 Verstärkung
1	0	0	Tone 3 Frequenz
1	0	1	Tone 3 Verstärkung
1	1	0	Rauschkontrolle
1	1	1	Rauschverstärkung

6) Datenformat

Die Bitbelegung für die einzelnen Register ist den folgenden Tabellen zu entnehmen.

Frequenzfestlegung (2-Byte Transfer):

D7			-				D0	
1	R2	R1	R0	F3	F2	F1	F0	(erstes Byte)
0	X	F9	F8	F7	F6	F5	F4	(zweites Byte)

Rauschquelle:

D7			-			D0
1	R2	R1	R0	X	FB	NF1	NF0

Verstärkung:

D7			-			D0
1	R2	R1	R0	A3	A2	A1	A0

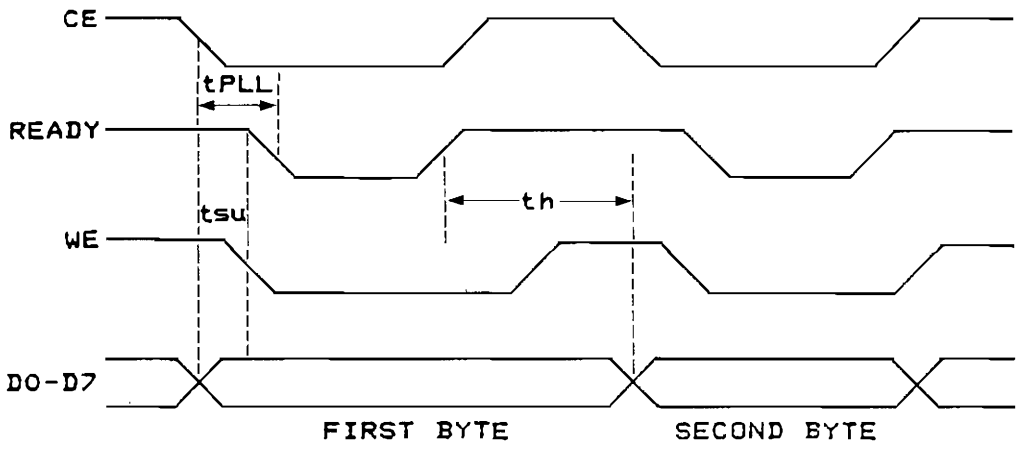
7) Datenformat

Die CPU spricht den SN 76489 A an, indem die CE-Leitung aktiv wird (low). Solange CE nicht aktiv ist, können keine Daten übertragen werden. Ist CE aktiv, so werden bei einem Strobesignal an WE (high nach low) die Daten im angesprochenen Register abgelegt. Dazu muß aber die gewünschte Date zu diesem Zeitpunkt anliegen.

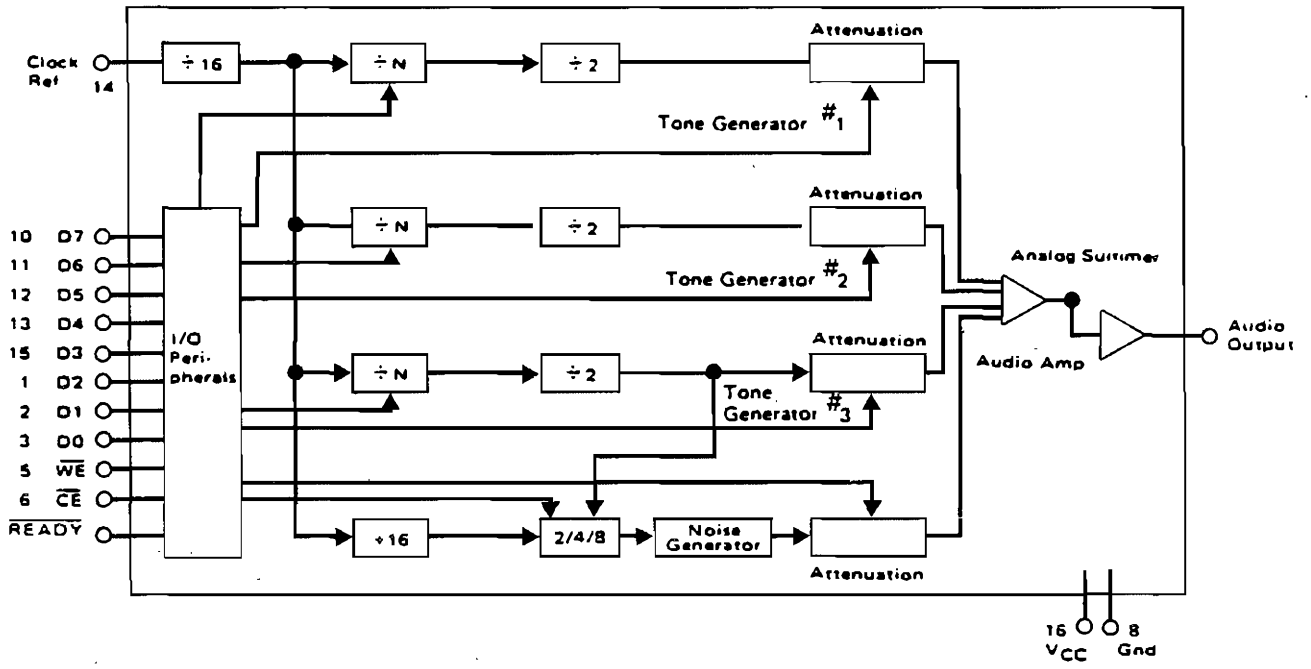
Zum Laden der Werte in die internen Register braucht der Chip etwa 32 Taktzyklen. Der READY-Ausgang (offener Kollektor) wird zur Synchronisation des Mikroprozessors beim Transfer verwendet. Dazu geht der Ausgang in den 'low'-Zustand, sobald CE aktiv ist, und kehrt erst dann in den 'high'-Zustand (über Pullupwiderstand) zurück, wenn der Datentransfer mit Abspeicherung vollendet ist.

Der Zeitablauf eines Transfers kann aus Figur 1 entnommen werden.

Figur 1 -Datentransfer Zeitdiagramm-



Figur 2 -Blockschaltbild des SN 76489 A-



Beschreibung des Blockdiagramms

Die Schaltung besteht aus drei programmierbaren Tongeneratoren, einem programmierbaren Rauschgenerator, einem Frequenzteiler, 4 getrennten Verstärkern (Abschwächer) und einem summierenden Verstärker mit Ausgangspuffer. Über ein 8-Bit parallel Interface kann das IC Daten vom Computer aufnehmen, speichern und mit ihnen die einzelnen Schaltungsgruppen steuern.

8. Die MTX 500 Speicheraufteilung

Die seitenweise-strukturierte Speicheraufteilung (Banking) des MTX Computers wurde gewählt, um folgende Betriebsarten realisieren zu können.

1) ROM BASED (RELCPMH = 0)

ROMs belegen den Speicherbereich von 0 bis 3FFFh. Die unteren 8k Monitor ROM stehen dabei in jeder Bank zur Verfügung (0 - 1FFFh). Die je 8k ROMs zwischen 2000h und 3FFFh stehen jeweils in den einzelnen Banks zur Verfügung. Der MTX verfügt über 8 Banks (0 - 7), die über die Leitungen R0, R1 und R2 des Bankport Registers aktiviert werden. Somit können hier 8x8k ROM angesprochen werden. Noch größere Kapazität steht bei den RAMs zur Verfügung (512k) da hier zwischen 16 Banks (0 - F) gewählt werden kann. Die Auswahl zwischen den RAM-Banks erfolgt über die Leitungen P0, P1, P2 und P3, die durch das entsprechende Bankport Register erzeugt werden. Dabei ist der Bereich von C000h bis FFFFh (16k) für alle RAM-Banks gemeinsam, so daß 32k RAM im Banking betrieben werden können. Der gemeinsame ROM und RAM Bereich für alle Banks hat den Vorteil, daß es ohne große Probleme möglich ist, von einer Bank in die andere umzuschalten, da gemeinsame Monitorfunktionen und Übergabevariablen in diesen Bereich abgelegt sind oder abgelegt werden können. Ein weiterer Vorteil des MTX liegt darin, daß ROM-Banking und RAM-Banking voneinander unabhängig sind.

Für die Grundgeräte MTX 500 und MTX 512 wurde folgende Verteilung des internen RAMs gewählt:

MTX 500 (32k): RAM befindet sich im Bereich von 8000h bis FFFFh in Bank 0

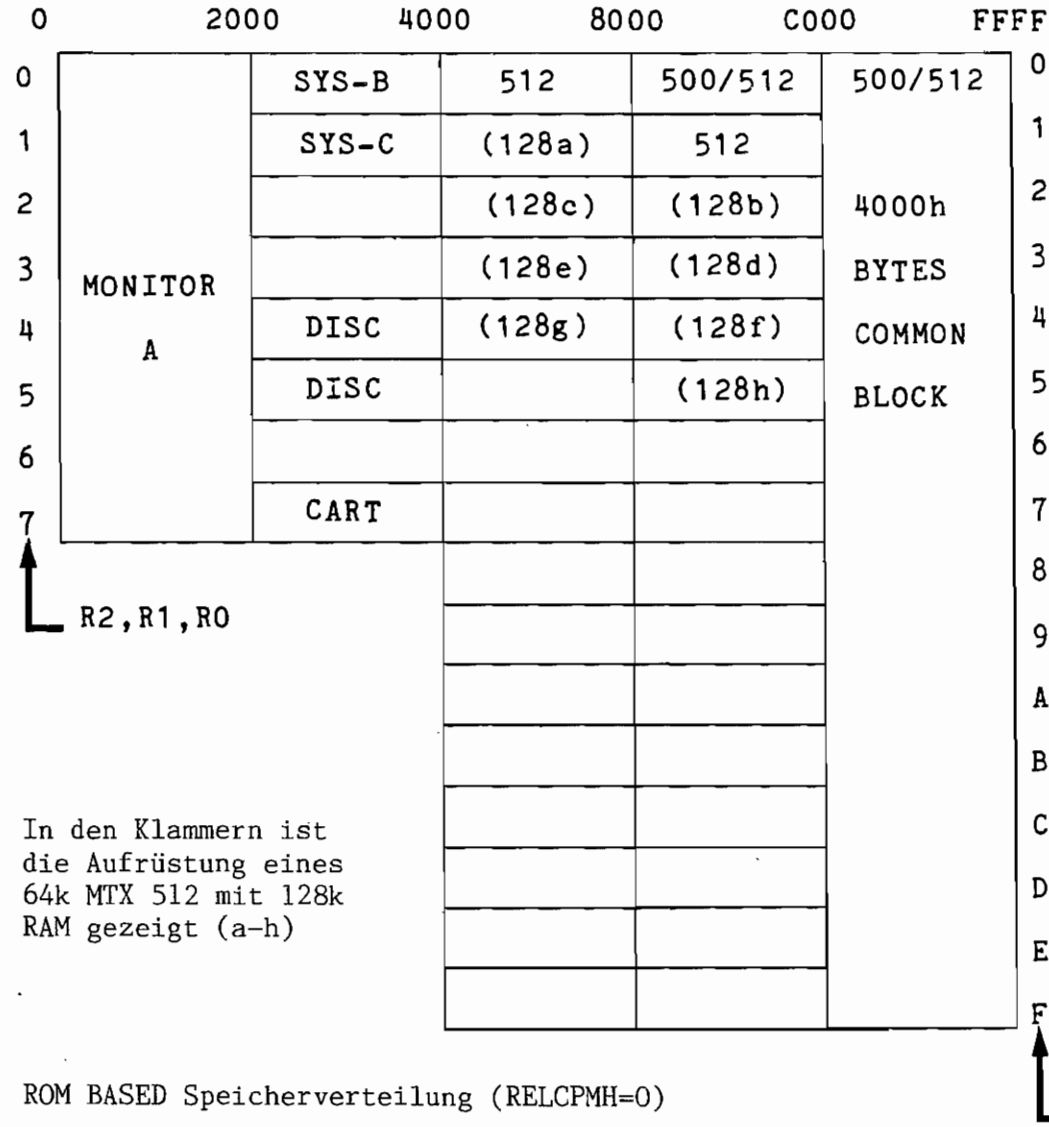
MTX 512 (64k): 48k RAM befindet sich im Bereich von 4000h bis FFFFh in Bank 0
16k RAM befindet sich im Bereich 8000h bis BFFFh in Bank 1.

2) RAM BASED (RELCPMH = 1)

In dieser Betriebsart sind alle ROMs abgeschaltet und es können bis zu 16 Banks mit 48k eingerichtet werden. Sie liegen zwischen 0h und BFFFh, und werden durch P0 bis P3 aktiviert. Im Bereich von C000h bis FFFFh liegen 16k RAM, die in jeder Bank zur Verfügung stehen. Man findet hier natürlich ähnliche Vorteile wie oben aufgeführt.

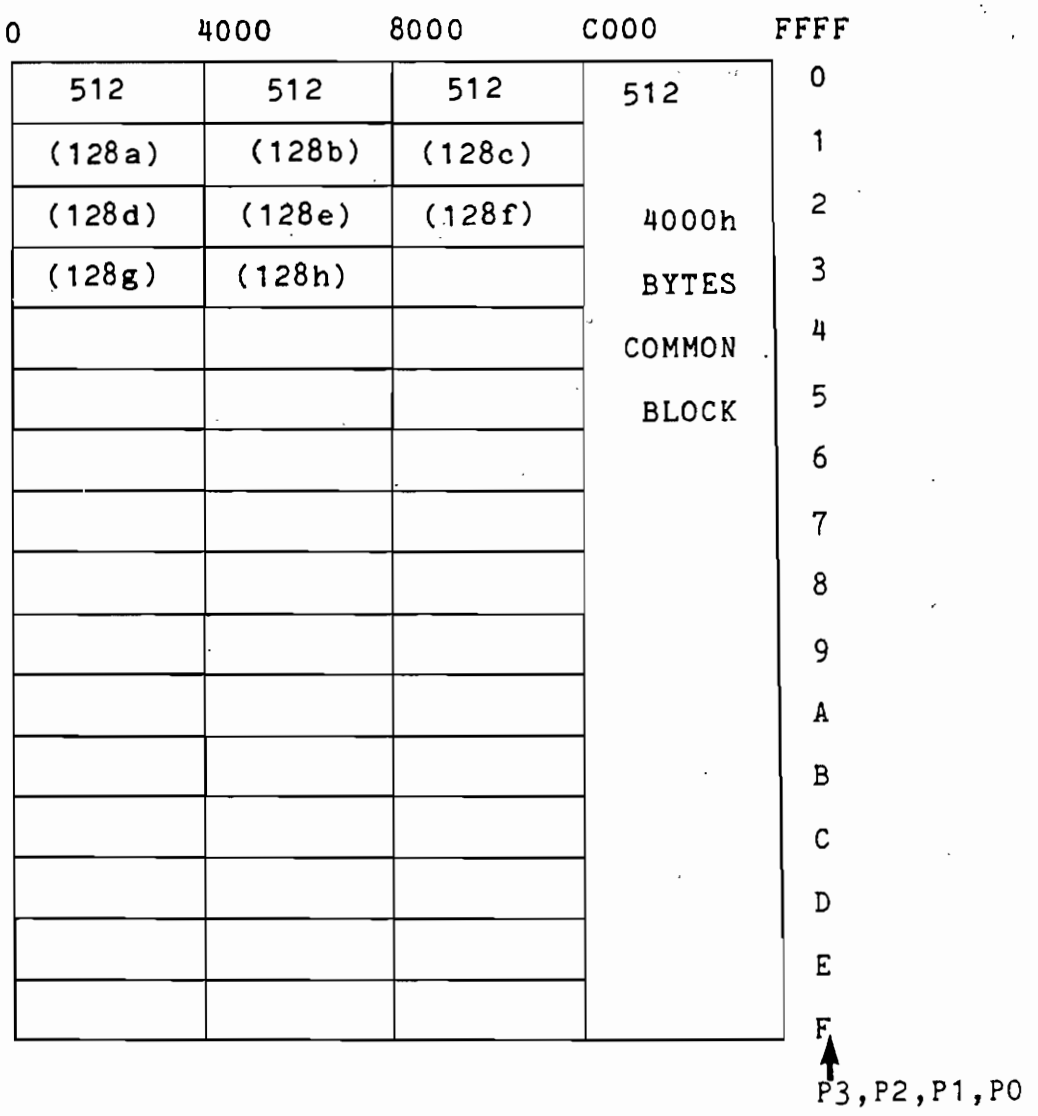
Da in diesem Modus kein Betriebssystem vorliegt, kann er praktisch nur im Zusammenhang mit einem Diskettenbetriebssystem betrieben werden (CPM). Die folgende Tabelle gibt eine Übersicht des Bankport Registers, das über OUT(0),DATA angesprochen werden kann.

D7	D6	D5	D4	D3	D2	D1	D0
RELCPMH	R2	R1	R0	P3	P2	P1	P0



In den Klammern ist die Aufrüstung eines 64k MTX 512 mit 128k RAM gezeigt (a-h)

ROM BASED Speicherverteilung (RELCPMH=0)



In den KLammern ist die Aufrüstung eines 64k MTX 512 mit 128k RAM gezeigt (a-h)

RAM BASED Speicherverteilung (RELCPMH=1)

9. Input/Output Adressen

In diesem Abschnitt wird aufgeführt, unter welchen I/O-Adressen einzelne Register oder Funktionen angesprochen werden. (Da die weiteren I/O Adressen für mögliche Erweiterungen noch nicht feststehen, sollten eigene Erweiterungen zum MTX mit variablen I/O Adressen versehen werden.)

00h
Input
IN(0) erzeugt das Druckerstrobe Signal. Durch diesen Befehl geht der Strobeausgang auf '0' Mit IN(4) oder dem CPU Reset geht der Ausgang wieder auf '1'. Damit Strobe während eines Interrupts nicht zu lange auf '0' bleiben kann, sollte in jedem Interrupt Strobe zurückgesetzt werden.

Output
OUT(0),d definiert die Bankadressen von RAM und ROM, und wählt zwischen RAM BASED und ROM BASED Modus. (Belegung der einzelnen Bits siehe 8. MTX Speicherverteilung.)

01h
Input
IN(1),d bewirkt ein Lesen des VDP (mode = 0) und bildet mit 02h zusammen zwei R/W Ports für den VDP. Details sind der Erläuterung zum TMS 9918 zu entnehmen. Man beachte jedoch, daß die Adressleitung A1 der CPU mit dem Mode Input verbunden ist.

Output
OUT(1),d bewirkt ein Schreiben in den VDP (mode = 0)

02h
Input
IN(2),d bewirkt Lesen des VDP (mode = 1).

Output
OUT(2),d bewirkt ein Schreiben in den VDP (mode = 1).

03h
Input
IN(3) wird als Strobesignal für den Tongenerator verwendet. Nachdem die Daten über OUT(6) anstehen, muß sofort ein Strobe erzeugt werden. Für die Übernahme braucht der Tongenerator ca. 32 Taktzyklen. In dieser Zeit darf kein neuer Strobeimpuls erfolgen. Die eingelesenen Daten bei IN(3) werden nicht verwertet.

Output
OUT(3),d gibt in D0 Daten an den Kassettenrekorder zur Speicherung aus (MIC). Das von D0 erzeugte Signal wird abgeschwächt (-20dB*VCC) und durch einen Tiefpaßfilter geleitet. Durch mehrfache Benutzung von OUT(3) wird ein serielles Signal erzeugt.

04h
Input
IN(4),d fragt den Betriebszustand des Druckers an der Centronics-Schnittstelle ab. Die einzelnen Bits haben dabei folgende Bedeutung:

D0	Busy	(aktiv '1')
D1	Error	(aktiv '0')
D2	PE	(aktiv '1')
D3	SLCT	(aktiv '1')

Output
OUT(4),d gibt parallel die 8-Bit Daten an den Drucker. Wenn der Drucker empfangsbereit ist (IN(4)), soll ca. 1 Mikrosek. nach Anstehen der Daten mit IN(0) der Strobe erzeugt werden. Nach wiederum ca. 1 Mikrosek. soll der Strobeimpuls durch IN(4) gelöscht werden.

05h
Input
IN(5),d wird verwendet, um die 8 niederwertigsten Bits der 10-Bit Empfangsleitung der 8x10 Tastaturmatrix zu lesen.

Output
OUT(5),d versort über einen Zwischenspeicher die 8 Treibleitungen der 8x10 Tastaturmatrix.

06h
Input
IN(6),d ergänzt IN(5). Die letzten beiden Empfangsleitungen werden über D2 und D3 gelesen.

Output
OUT(6),d gibt über einen Zwischenspeicher Daten an den Tongenerator, die dort durch ein Strobesignal (IN(3)) übernommen werden.

07h
Input
IN(7),d liest über eine PIO den freien parallel Input Port. Daten können in der PIO bis zum Lesen durch ein INSTB-Signal (aktiv '0') zwischengespeichert werden.

Output
OUT(7),d gibt über die PIO Daten an den parallelen Output Port. Die Daten werden in der PIO bis zur nächsten Ausgabe abgespeichert. Der Ausgang ist ein Tri-State Ausgang, der mit OTSTB gesteuert wird.

08,09,0A,0Bh
Unter diesen 4 Portadressen kann der Z80A CTC angesprochen werden. (Beachten Sie das Datenblatt des CTC.)

Die Verteilung der einzelnen Ports ist folgende:

	Input	Output
08 ch1	VDPINT	keine Verbindung
09 ch2	4MHz/13	DART ser. clock 0
0A ch3	4MHz/13	DART ser. clock 1
0B ch3	CSTTE edge	keine Verbindung

0C,0D,0E,0Fh
Unter diesen Adressen werden 4 Schreib/Leseports des DARTs angesprochen. (Beachten Sie das Datenblatt des DART.)

Die Bedeutung der einzelnen Ports ist folgende:

0C chA	Daten
0D chB	Daten
0E chA	Kontrollwerte
0F chB	Kontrollwerte

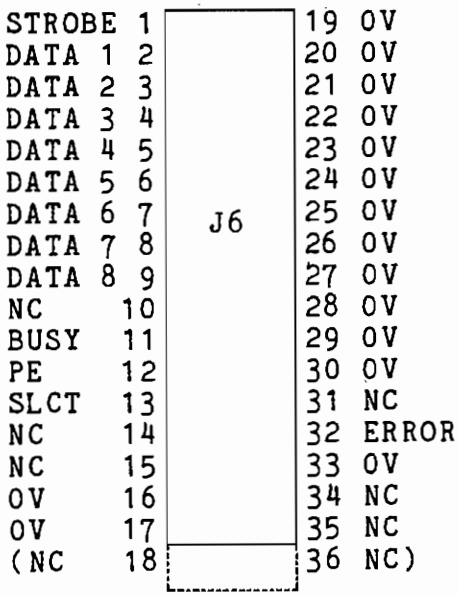
Die Ports 10h bis 1Eh sind zur Zeit nicht benutzt, jedoch ist 1Fh für eine Steuerung eines Kassettenrekorders vorgesehen. Die Portadressen 20h bis FFh werden in der Disketteneinheit für externe Nutzung zur Verfügung gestellt.

10. Das parallele Druckerinterface

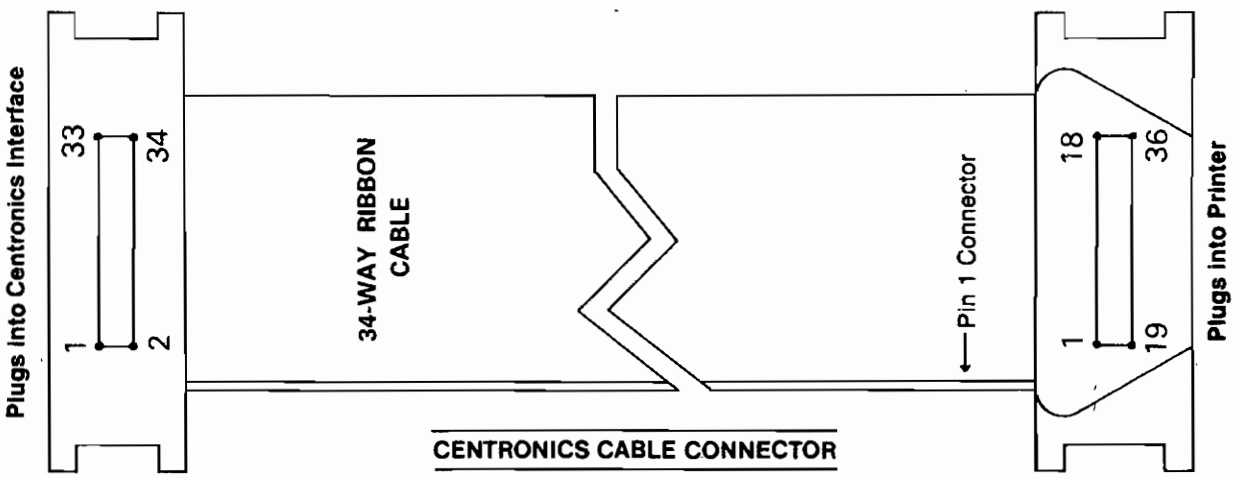
Der MTX Computer verfügt bereits in der Grundversion über einen Druckerport. Dieser ist Centronics kompatibel, und wird mit einer 34poligen Steckerleiste (17+17 im 2,54 Raster) an der Rückseite aus dem Gehäuse herausgeführt.

Die Pinbelegung ist der folgenden Zeichnung zu entnehmen.

Passend zum Grundgerät wird der Drucker DMX80 mit Parallelschnittstelle angeboten, jedoch kann auch jeder andere Drucker mit Centronics kompatibler Schnittstelle verwendet werden.



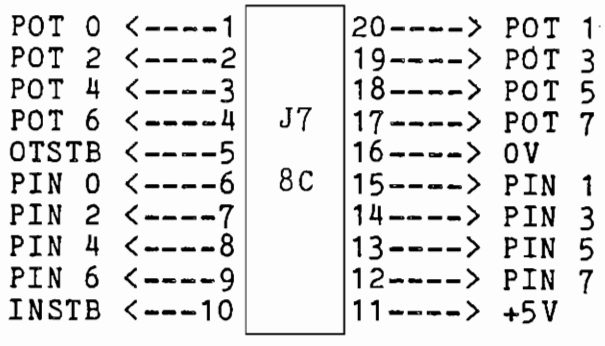
MTX500 Series Centronics Type Parallel Printer I/F Connector
34-Way (17+17) Right Angle Header Plug



11. Der parallele I/O-Port

Über einen 20-Pin DIL-Sockel wird ein 8-Bit Eingangs- und ein 8-Bit Ausgangsport im Grundgerät zur Verfügung gestellt. Bei den Anschlüssen handelt es sich um eine TTL kompatible PIO. Die Eingänge können mit INSTB getaktet werden (aktiv 'low'), und die Ausgänge können mit OTSTB in den TRI-State Status versetzt werden (Tri-State bei 'high'). Zusätzlich liefert der Anschluß Masse und +5V Leitungen, wobei jedoch die +5V Leitung nur mit 20mA belastet werden darf.

Die Pinbelegung der Schnittstelle ist dem folgenden Bild zu entnehmen:



MEMOTECH

Memotech Limited, Witney, Oxon OX8 6BX