HX-20 HAND-HELD-COMPUTER
Bedienungs-Handbuch

Technologie, die Zeichen setzt.

EPSON

Y20199008004

HX-20 HAND-HELD-COMPUTER
Bedienungs-Handbuch

EPSON

Verehrter Kunde,

wir danken Ihnen für das unserem Hause entgegengebrachte Vertrauen, und beglückwünschen Sie zum Kauf des EPSON Computers HX-20.

Mit dem HX-20 haben Sie ein Gerät gekauft, das Sie unabhängig vom Stromnetz überallhin begleiten und zu Ihren Diensten sein kann. In dem praktischen DIN-A4 Format kann er in jeder Aktentasche untergebracht werden.

In seiner Grundausstattung bietet dieser Computer schon Möglichkeiten, die ihm einen wesentlichen Vorsprung vor seinen Mitbewerbern gibt.

Nun läßt sich der HX-20 aber bis zum vollen Tischcomputer aufrüsten, der jedem Leistungsvergleich mit seinen Konkurrenten standhält. Der Tradition unseres Hauses folgend, sind wir auch mit diesem Gerät bei großer Leistung in bewährter Qualität, erstaunlich günstig.

Natürlich ist es unmöglich, einen Computer auf den Markt zu bringen, der allen Anforderungen gerecht werden kann, die Anwendungspallette ist zu vielfältig. Mit dem HX-20 haben Sie allerdings auch die Möglichkeit, mit HOST-Rechnern zu kommunizieren.

Anhand dieser Dokumentation, welche die Revision 2 ist, hoffen wir Ihnen die Handhabung und den Gebrauch des HX-20 so einfach wie möglich machen zu können. Auch haben wir etliche Vorschläge und Anregungen mit berücksichtigt.

Ihr EPSON-Fachhändler wird Sie auch mit diesbezüglichen Fragen beraten, er verfügt über das entsprechende „KNOW HOW". Ebenso kann er Ihnen die vorhandenen Programme demonstrieren, und eventuelle Software-Wünsche mit berücksichtigen.

Nun wünschen wir Ihnen viel Freude mit Ihrem neuen Begleiter HX-20!

INHALTSVERZEICHNIS

Einführung zum HX-20

BASIC

Kommandos/Befehle

Funktionen

Zusatzinformation zum HX-20

INITIALISIEREN – BETRIEBSBEREITMACHUNG DES HX-20

Der HX-20 muß vor der ersten Inbetriebnahme initialisiert werden, das heißt, mit dem "Kaltstart" muß wie folgt begonnen werden:

Schalten Sie den HX-20 ein, indem Sie den POWER-Schalter (s. Seite 20) auf ON stellen. Nun erscheint folgende Anzeige auf dem Bildschirm:

```
CTRL/§ Initialize
1 MONITOR
2 BASIC
```

Nun drücken Sie die CTRL-Taste, gleichzeitig die SHIFT-Taste und die Taste "3"(§). (Um Sonderzeichen tasten zu können, muß immer die SHIFT-Taste gedrückt werden, wie bei der Schreibmaschine).

Auf dem Display erscheint nun die Anzeige:

```
Enter DATE and TIME
 MMDDYYHHMMSS
=
```

Geben Sie nun das Datum in der Reihenfolge Monat, Tag, Jahr mit jeweils zwei Stellen ein, ebenso die Uhrzeit in der Reihenfolge Stunde, Minute, Sekunde. Die Eingabe mit der RETURN-Taste bestätigen. War die Eingabe fehlerhaft, mit der BREAK-Taste unterbrechen und die Eingabe wiederholen. Der HX-20 ist nun betriebsbereit.

Auf dem Display erscheint das MENU mit der Funktion 1 für den MONITOR und der Funktion 2 für BASIC. Erst jetzt kann eine der beiden Funktionen über die Tasten "1" oder "2" angewählt werden.

EXPANSION-UNIT

Hinweis: Bei Anschluss einer Expansion-Unit müssen zuerst die Programme gesichert werden. Danach muss der HX-20 initialisiert werden (Kaltstart).

Als Besonderheit zu manchen anderen Computern wird der Speicherbereich des HX-20 in fünf Programmspeicher aufgeteilt.

Mit dem Initialisieren werden automatisch diese fünf Programmspeicher und der RAM-File Bereich gelöscht.

In jeden dieser fünf Programmspeicher kann ein BASIC-Programm mit einem TITLE in das MENU aufgenommen werden. Die MENU-Nummern 3 bis 7 kennzeichnen diese Programmbereiche, die durch Drücken einer dieser Tasten sofort das Programm automatisch starten. Das erweiterte MENU wird nach jedem Einschalten auf dem Display angezeigt.

```
CTRL/S Initialize
1 MONITOR
2 BASIC
3 PROG1
```

Sind alle MENU-Namen belegt, werden nach dem Einschalten die TITLE auf dem Display mit Hilfe der SCRN-Taste angezeigt. Anschließend sind die Funktionen 1 bis 4 auf dem LCD sichtbar. Durch Drücken der RETURN-Taste können die Programm-Namen von 4 bis 7 angezeigt werden. Oder mit der SCRN-Taste (Scroll) zurück bzw. mit der SHIFT- und SCRN-Taste vorgescrollt werden.

```
1 MONITOR
2 BASIC
3 PROG1
4 PROG2
5 PROG3
6 PROG4
7 PROG5
```

1.1. Allgemeiner Überblick

Der HX-20 ist ein Personal-Computer, der eine Vielzahl von Applikationen ermöglicht. Von der Minimalkonfiguration mit eingebautem Minidrucker und L.C.D. ist er bis zum leistungsfähigen Mikro-Computer-System aufrüstbar.

Folgende Erweiterungen stehen dem Anwender zur Verfügung:

1. **Fernsehadapter**

   Mit Hilfe dieses Adapters ist es möglich, auf einem Fernseher einen Text wie folgt darzustellen:

   a) schwarz/weiss
      24 Zeilen je 80 Zeichen
   b) Farbe
      24 Zeilen je 40 Zeichen, 4 Grundfarben
   c) Graphik
      480 · 240 Dots
   d) Alle Zeichen wie auf dem HX-20 darstellbar

2. **Terminal Floppy-Disk**

   Die Terminal-Floppy ermöglicht es dem Anwender, Massendaten im schnellen Zugriff zu lesen und zu speichern.

   Technische Daten:

   Extrem schmales 5 1/4 Zoll Doppellaufwerk mit einer Speicherkapazität von 656 Kilobyte.
   Double Side, Double Density
   40 Spuren, 16 Sektoren je Spur
   Übertragungsgeschwindigkeit:
   250 K-Bit/Sek. (MFM), 125 K-Bit/Sek. (FM)

3. **Expansion Unit**

   Mit Hilfe dieser seitwärts aufsteckbaren Erweiterungseinheit besteht die Möglichkeit, die Speicherkapazität des HX-20 von 16 KB RAM auf 32 KB RAM zu verdoppeln.
   Weiterhin stehen zwei freie Steckplätze zur Verfügung, die der Anwender mit entsprechenden, in ROM's abgelegten Programmen bestücken kann.

4. Akustikkoppler

Der EPSON-Koppler CX-20 ermöglicht es dem Anwender, seinen HX-20 mit der Außenwelt zu verbinden, sei es um erfasste Daten zu übertragen oder um ein "Gespräch" von Computer zu Computer zu führen.
Der Koppler ist ebenfalls netzunabhängig, so daß diese Verbindung von jeder Telefonzelle aus zustande kommen kann.
Die Übertragungsgeschwindigkeit beträgt bis zu 300 Baud.

5. Barcode-Lesestift

Dieser praktische Helfer kann über das speziell dafür vorgesehene Interface angeschlossen werden.
Der BARCODE hält fast überall Einzug und ist einfach zu verwenden.
Entsprechende Leseprogramme stehen zur Verfügung.

6. Drucker

Für den Fall, daß der eingebaute Drucker nicht ausreicht, steht als Erweiterung die EPSON-Drucker-Pallette zur Verfügung.
Bei den neueren Modellen stehen zusätzlich 8 internationale Zeichensätze, Underline, Sub- und Superscript zur Verfügung.

7. EPSON Computer

Die EPSON-Computer der QX-Serie eignen sich vorzüglich als HOST-System, da sie die "gleiche Sprache" sprechen. Neben dem erstaunlichen Preis und der reichhaltigen Software haben die Geräte folgende Spezifikation:

256 KByte RAM
2 Floppy-Disk Laufwerke, 5 1/4 Zoll mit je 328 KB Speicherkapazität
Hochauflösende Grafik mit 640 * 400 Punkten
8 Internationale Zeichensätze konfigurierbar

8. ROM-Cartridge

Diese Box ermöglicht es, Programme die in EPROM's abgelegt wurden, zu verwenden.
Die Cartridge wird einfach an der rechten Geräteseite aufgesteckt und ist sofort einsatzbereit.
Sie eignet sich besonders für häufig benutzte Programme.

9. Mikrokassette

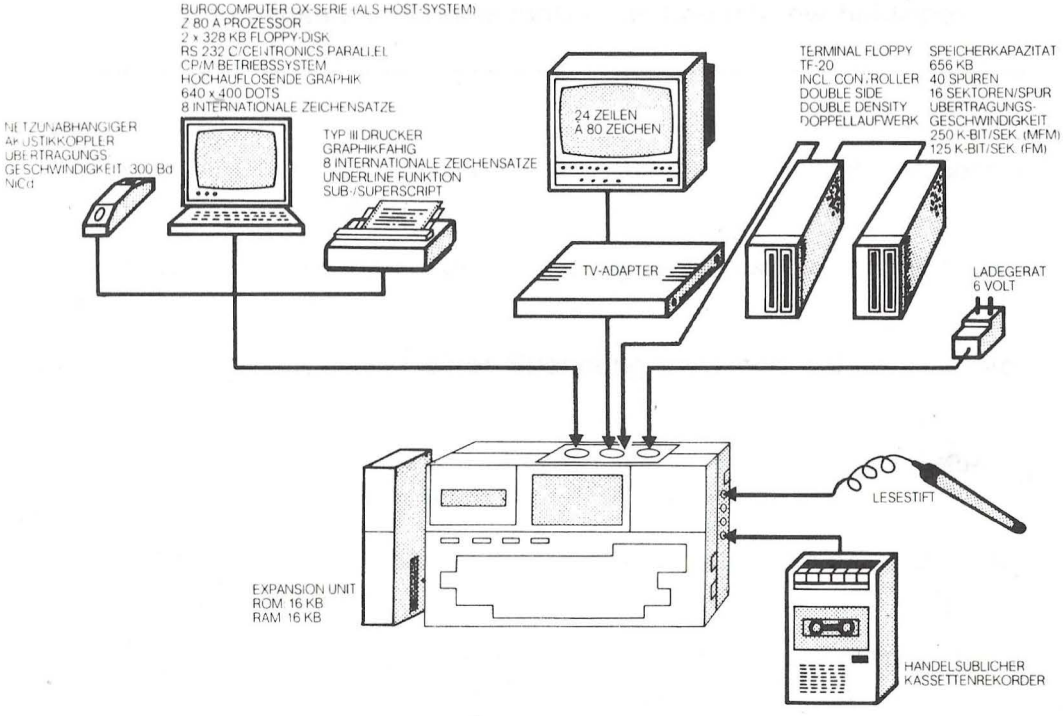
Die Mikrokassette ist eine sehr praktische und wohl am häufigsten eingesetzte Option des HX-20.
Das komplette Kassettenlaufwerk wird einfach auf den Rechner aufgesteckt und ist sofort einsatzbereit.
Dem Anwender steht damit ein preisgünstiger und relativ schneller Massenspeicher zur Verfügung.
Fast alle Programme der EPSON-Software-Bibliothek werden auf diesen Kassetten geliefert.

10. Externer Kassettenrekorder

Ein ebenfalls eingebautes Interface ermöglicht den Anschluß eines handelsüblichen Kassettenrekorders.
Somit steht dem Anwender ein extrem günstiger Massenspeicher zur Aufzeichnung von Programmen und Daten zur Verfügung.

Die im Gerät bereits eingebauten Schnittstellen ermöglichen diese Erweiterungen ohne die sonst üblichen Mehrkosten für die entsprechenden Interfaces.

Zur besseren Übersicht des Geschriebenen alle Erweiterungen in schematischer Darstellung:



1.2. Lieferumfang

Die Grundversion verfügt über folgende eingebaute Peripherie:

- L.C.D.
- Minidrucker

Zum Lieferumfang gehören außerdem:

- AC Adaptor
- Papierrolle
- Farbband
- Bedienungshandbuch in deutscher Sprache

Falls eines der hier genannten Teile fehlt, wenden Sie sich bitte an Ihren EPSON-Vertragshändler.

Wichtige Hinweise:

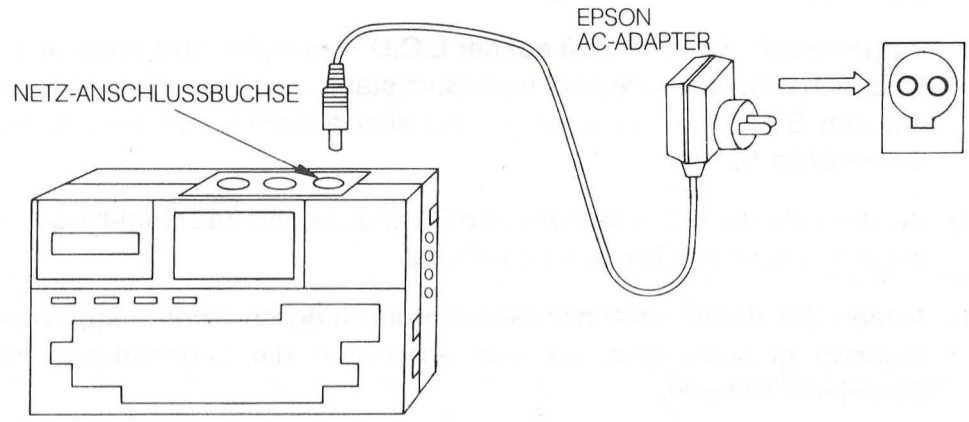
- Bitte achten Sie darauf, daß das Gerät mit der Seriennummer beim EPSON-Vertragshändler registriert wird.
- Sollte das Gerät keine Seriennummer besitzen, so kann es nicht registriert werden und es besteht kein Garantieanspruch.

1.2.1. Laden des Computers

Das Gerät ist vorgeladen und muß, um einsatzbereit zu sein, zuerst voll aufgeladen werden.

Hierzu ist das Netzgerät in die Steckdose und der Ladestecker in die dafür vorgesehene Buchse des HX-20 zu stecken.

Der Computer benötigt für eine volle Aufladung acht Stunden und sollte **nicht länger** am Strom angeschlossen bleiben.



1.2.2. Stromversorgung

Der HX-20 wird durch wiederaufladbare Nickel-Cadmium Batterien permanent mit Strom versorgt.

Diese Hochleistungsbatterien arbeiten mit einer Spannung von 4,5 bis 6,0 Volt.

Die Datensicherheit ist bei einer Spannung zwischen 4,0 und 6,0 Volt gewährleistet.

Der EPSON AC-Adaptor ist in seiner Bauweise genau auf die Batterien abgestimmt und bringt die optimale Ladeleistung.

Wie bereits erwähnt, braucht der HX-20 acht Stunden, um seine Batterien voll aufzuladen.

Achten Sie bitte darauf, daß diese Ladezeit **nicht überschritten** wird, da die Batterien sonst bald schadhaft werden können.

Wenn die Batterien getauscht werden müssen, so sollte dies unbedingt durch den EPSON-Fachhändler geschehen, der über die entsprechenden Ersatzteile und Kenntnisse verfügt.

1.2.3. Netzladung

Prozedur:

(1) Schalten Sie den mit POWER bezeichneten Schalter auf der rechten Geräteseite auf die Stellung OFF.

(2) Vergewissern Sie sich, daß auf der L.C.D.-Anzeige nichts mehr steht. Falls sich wider Erwarten noch ein Buchstabe oder ähnliches zeigt, so drücken Sie den RESET-Knopf, der sich ebenfalls auf der rechten Geräteseite befindet.

(3) Stecken Sie den AC-Adaptor in die Steckdose und das Gegenstück in die entsprechende Buchse des HX-20.

(4) Achten Sie darauf, daß das Gerät, wenn möglich nicht länger als 8 Stunden geladen wird, da sich ansonsten die Lebensdauer der Batterien verringert.

Selbstverständlich ist es auch möglich, das Gerät während des Betriebes aufzuladen, jedoch sollte aus Gründen der Batterielebensdauer darauf verzichtet werden.

Wenn die Batteriespannung unter 4,5 Volt absinkt, erscheint immer wieder die Anzeige

CHARGE BATTERY!

Dieser Aufforderung sollte sofort nachgekommen werden, da ansonsten die Sicherheit der abgespeicherten Daten auf dem Spiel steht.

Ebenso kann es vorkommen, daß auf dem L.C.D. die "unmöglichsten" Zeichen erscheinen. Auch in diesem Fall ist die Batteriespannung abgesunken und das Gerät sollte sofort nachgeladen werden.

Wichtige Hinweise:

- Achten Sie darauf, das Gerät, wenn nötig, nur 8 Stunden zu laden.
- Wenn ein Wechsel der Batterien nötig wird, so lassen Sie diesen durch Ihren EPSON-Fachhändler vornehmen.
- Verwenden Sie zur Netzladung nur den von EPSON mitgelieferten AC-Adaptor.

1.2.4. Betriebsdauer

Da nun so viel von der Leistungsfähigkeit der NiCd-Batterien gesprochen wurde, hier nun die entsprechenden Leistungsmerkmale.

Bei einer Temperatur zwischen 5 und 35 Grad Celsius garantiert EPSON folgende Zeiten bei einem voll aufgeladenen Gerät:

a) reiner L.C.D.-Betrieb : mindestens 24 Stunden

b) nur Druckbetrieb : maximal 1 Stunde

c) nur Mikrokassettenbetrieb : maximal 1 Stunde

1.3. Schalter/Regler

Auf der rechten Geräteseite befinden sich folgende Schalter bzw. Regler:

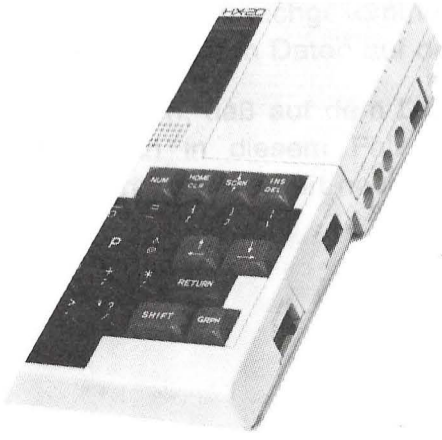
POWER : Mit Hilfe dieses Schalters ist das Gerät ein-(ON) oder auszuschalten.

VIEW ANGLE : Dieser Regler ermöglicht es, die L.C.D.-Anzeige auf den jeweiligen Blickwinkel genau einzustellen. Sollte bei Einschalten des Geräts einmal keine Anzeige zu sehen sein, so ist zuerst der Regler zu drehen. Die Rasterung kann individuell, je nach Blickwinkel und Lichteinfall, eingestellt werden.

MIC/EAR/REM : Diese Interface-Buchsen sind zum Anschluß eines externen Kassettenrekorders bestimmt.

BARCODE : Buchse zum Anschluß eines BARCODE-Lesestift.

RESET : Bei Betätigung dieser Taste wird das System zurückgesetzt und immer ins MENU zurückgesprungen.
Bei einer Programmausführung wird das entsprechende Programm abgebrochen und die Variablenwerte/Feldinhalte gelöscht. Das Programm an sich bleibt jedoch erhalten.



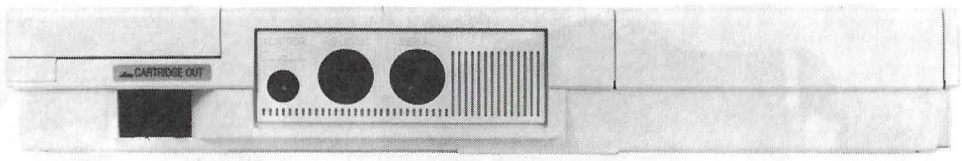
Geräterückseite:

CARTRIDGE OUT	:	Mit Hilfe dieses Schiebers läßt sich die Dummy-Box (das Leergehäuse) auf der rechten Geräteseite entfernen. Wenn das Gehäuseteil entfernt ist, so wird eine Anschlußleiste sichtbar. Sorgen Sie dafür, daß diese Leiste nicht verschmutzt und nicht beschädigt wird, da Sie ansonsten weder Mikrokassette noch ROM-Cartridge, die sich hier alternativ aufsetzen lassen, verwenden können.
ADAPTOR	:	Diese Buchse ist für den Anschluß des AC-Adaptors gedacht.
RS-232C	:	Interface V24-8 PIN Stecker
HIGH SPEED SERIAL	:	Interface-5 PIN Stecker

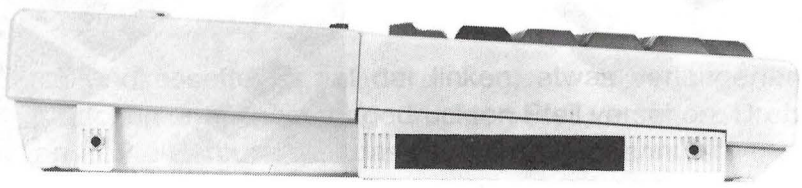
Auf der linken Geräteseite befindet sich eine längliche schwarze Abdeckkappe. Unter dieser Kappe ist der Anschluß für die Expansion-Unit zu erkennen. Entfernen Sie diese Kappe nur, wenn Sie die Expansion-Unit aufstecken und anschrauben müssen.

Eine Beschädigung der hier herausgeführten Steckkontakte (System-Bus) kann unter Umständen den Austausch der Hauptplatine zur Folge haben.

Geräterückseite



linke Geräteseite



1.4. Eingebaute Peripherie

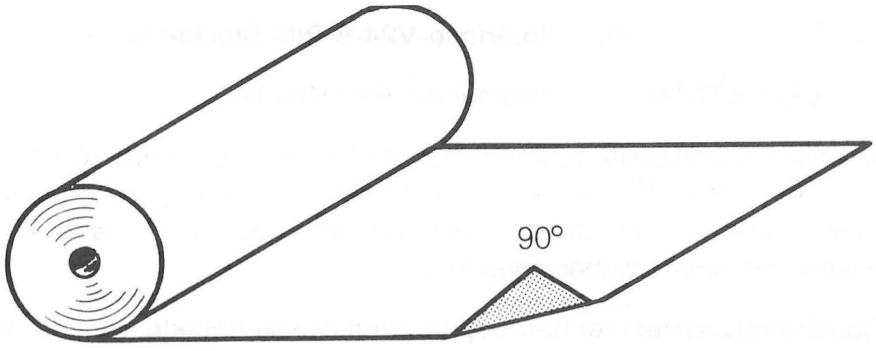
Als eingebaute Peripherie sind Minidrucker, L.C.D. und Tastatur zu verstehen.

1.4.1. Minidrucker

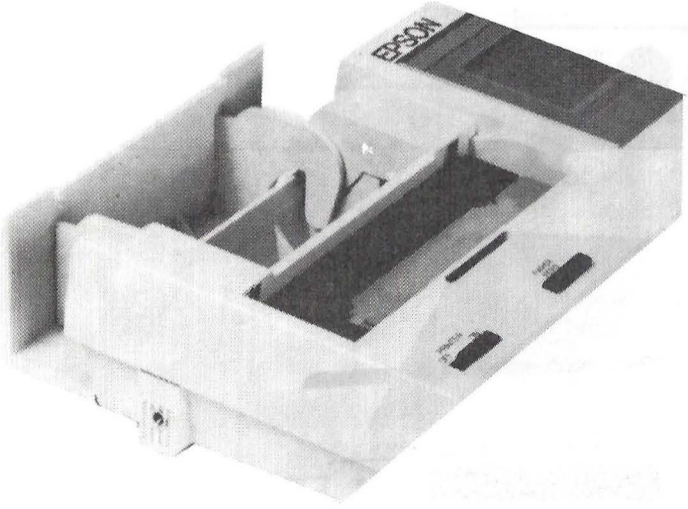
a) Papier einlegen

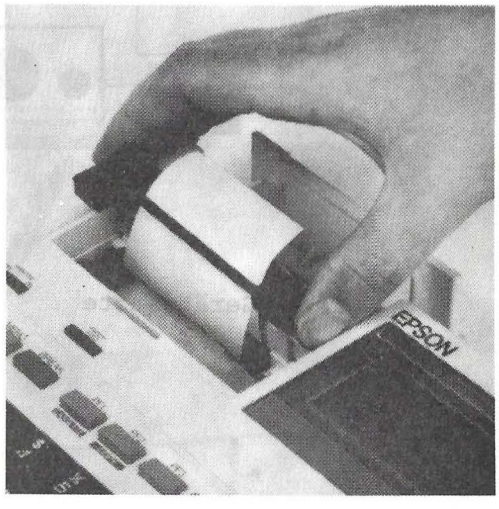
Nehmen Sie die Papierrolle aus der Plastikfolie heraus und wickeln Sie ein circa 5 cm langes Stück ab.

Von diesem Stück falten Sie die Hälfte in einem Winkel von 90 Grad um.



Nun drücken Sie den Deckel des Papierbehälters an der dafür vorgesehenen Leiste mit dem Fingernagel nach hinten.





Nehmen Sie nun die Papierrolle und führen Sie die abgeknickte Stelle unmittelbar unter der Trennleiste ein.

Schieben Sie den Schalter

PRINTER
OFF ON

nach rechts auf ON. Nun drücken Sie die Taste

PAPER FEED

bis das Papier durch die Abreißkante geschoben wird.

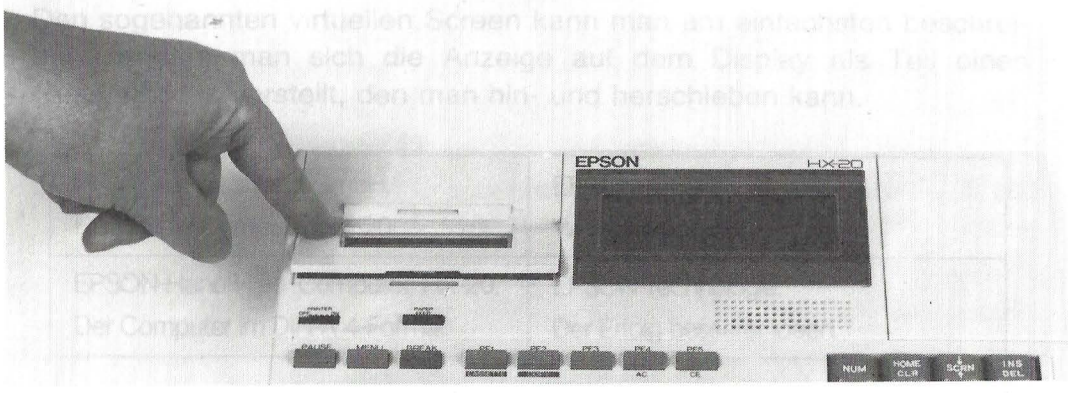
Legen Sie die Papierrolle in den Aufbewahrungsbehälter und schließen Sie den Deckel.

Wenn die Papierrolle zu Ende geht, so erscheint auf der rechten Seite ein roter Streifen als Kennzeichen. Für diesen Fall sollte eine neue Rolle bereitliegen.

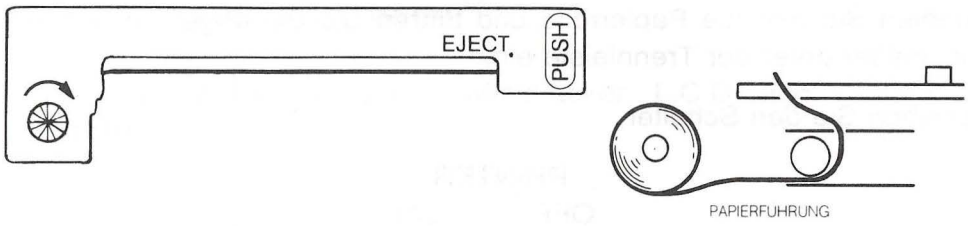
b) Farbband einlegen

Auf dem Gehäuseteil mit der Abreißkante ist PUSH aufgedruckt.

Darüber ist ein geriffeltes Quadrat. Wenn Sie mit dem Finger darauf drücken, so können Sie dieses Teil abheben.



Die Farbbandkassette ist auf der linken, etwas verlängerten Seite mit einem Rädchen und einem aufgedruckten Pfeil versehen. Drehen Sie das Rädchen in Pfeilrichtung, bis das Farbband gespannt ist.



Nun legen Sie das Farbband in dieser Position ein. Dabei ist zu beachten, daß das Papier zwischen Farbband und Kassettengehäuse durchgeführt werden muß.

Auf der rechten Seite der Farbbandkassette ist EJECT und PUSH aufgedruckt. Wenn Sie das Farbband wechseln müssen, drücken Sie mit leichtem Fingerdruck auf die bezeichnete Stelle.

Danach können Sie die Farbbandkassette mühelos herausnehmen.

Nach vollzogenem Wechsel setzen Sie das Gehäuseteil wieder auf das Gerät. Falls sich schmutzige Finger nicht vermeiden ließen, so können wir Sie trösten:

Es ist noch kein Meister vom Himmel gefallen.

1.4.2. LIQUID CRYSTAL DISPLAY (L.C.D.)

Das Display hat eine Darstellung von 4 Zeilen zu je 20 Zeichen. Seine Graphikauflösung beträgt 120 * 32 einzeln ansteuerbare Punkte (Dots).

Jedes Zeichen wird anhand einer 6 * 8 Punktmatrix dargestellt.

6 x 8 PUNKTMATRIX

		1			
	1		1		
1				1	
1				1	
1	1	1	1	1	
1				1	
1				1	

1.4.2.1. Virtueller Bildschirm

Wenn Sie BASIC oder ein Applikationsprogramm benutzen können Sie auf dem Display Text oder Graphik darstellen.

Das L.C.D. selbst zeigt 4 Zeilen zu je 20 Zeichen sichtbar an. Tatsächlich aber kann wesentlich mehr dargestellt werden.

Den sogenannten virtuellen Screen kann man am einfachsten beschreiben, in dem man sich die Anzeige auf dem Display als Teil eines Gesamtbildes vorstellt, den man hin- und herschieben kann.

EPSON Deutschland GmbH. Hand-Held Computer HX-20.	EPSON-Drucker. Weltweit die Nr. 1
EPSON-Hand-Held Computer HX-20. Der Computer im DIN-A 4-Format.	EPSON-Technologie. Der Erfolg besserer Ideen.

Die Dimensionen dieses virtuellen Bildschirmes können programmbedingt variieren; im Normalfall jedoch umfaßt er 8 Zeilen je 40 Zeichen.

Die Anzeige befindet sich dementsprechend außerhalb des virtuellen Bereiches und kann durch entsprechende Tasten direkt oder per Programm hin- und herbewegt werden.

1.4.2.2 Text und Graphik Anzeige

Hier muß bereits etwas dem eigentlichen BASIC-Teil der Beschreibung vorgegriffen werden.

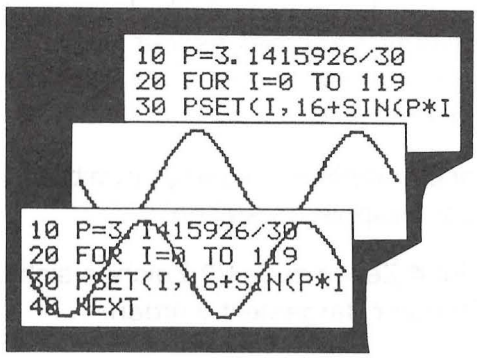
Der HX-20 hat eigentlich zwei Bildschirmzeilen, die unterschiedliche Funktionen haben.

Der erste Teil ist der "Text Screen", der nur Zeichen darstellt.

Der zweite Teil ist der "Graphic Screen", der, wie sein Name aussagt, dazu benutzt wird, um Graphik mittels der entsprechenden BASIC-Kommandos darzustellen.

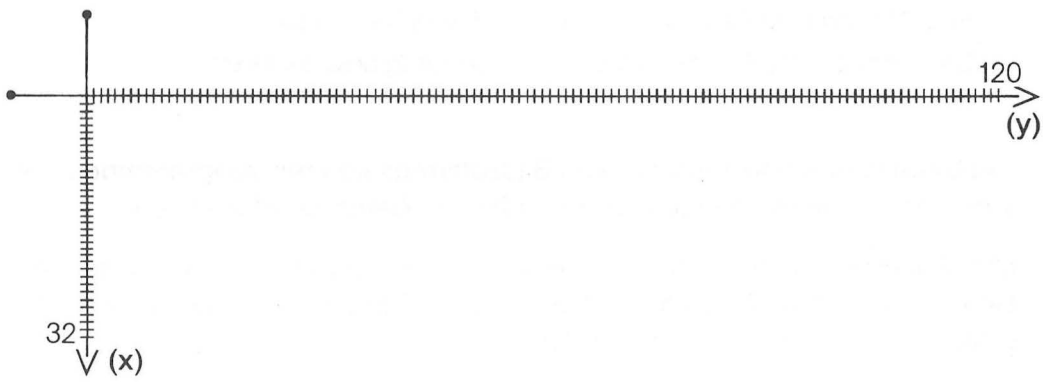
Jeder dieser Bildschirme ist einzeln darstellbar.

EPSON-BASIC jedoch erlaubt auch eine gemeinsame Darstellung von Graphik und Text auf einem Bildschirm, der dann der "Actual-Screen" genannt wird.



Bei einer entsprechenden graphischen Darstellung ist zu beachten, daß das amerikanische Koordinatenkreuz etwas anders als bei uns dargestellt wird.

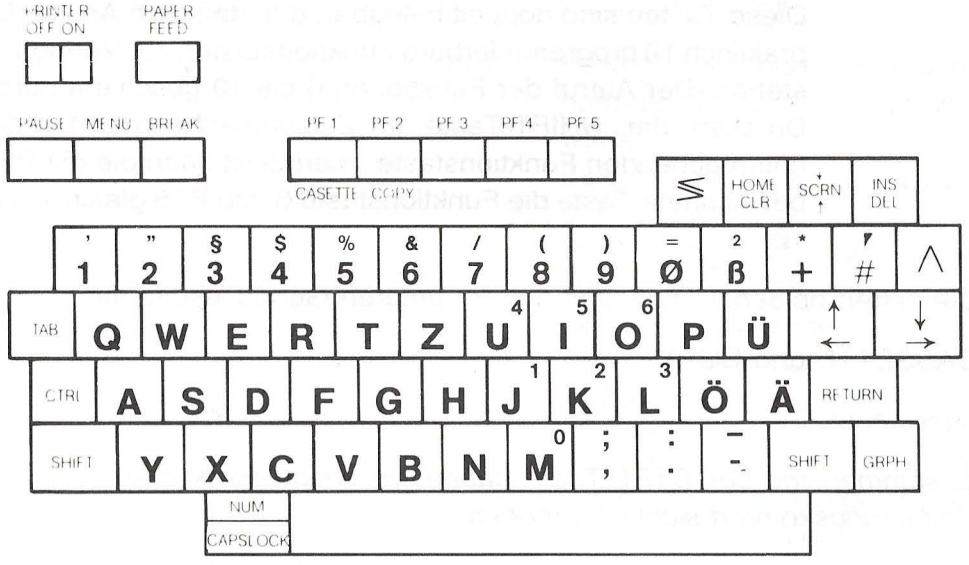
Die folgende Graphik soll Ihnen bei der Umstellung etwas behilflich sein.



1.4.4. Tastatur

Der HX-20 verfügt über eine deutsche Standard-Tastatur.

Diese Tastatur umfaßt eine Vielzahl von Zeichen und Funktionen.



Auf der oberen Leiste befinden sich acht Tasten, die wie folgt bezeichnet sind:

PAUSE : Diese Taste unterbricht eine Programmauflistung oder eine Programmausführung. Dies ist besonders bei Programmauflistungen über das L.C.D. sehr vorteilhaft, da die entsprechenden Programmzeilen ausführlich betrachtet werden können. Die Ausführung geht weiter, wenn irgendeine Taste auf der Tastatur gedrückt wird. Die Tasten 0 ~ 9 ermöglichen unterschiedliche SCROLL-Geschwindigkeiten.

MENU : Ein Druck auf diese Taste bricht jedes laufende Programm ab und kehrt in das MENU zurück. Die Variablenwerte eines Programms werden zurückgesetzt, das Programm selbst bleibt jedoch erhalten.

BREAK : Wird diese Taste gedrückt, so wird jedes Programm unterbrochen und die entsprechende Programmzeile, wo die Unterbrechung stattfand, angezeigt.

Beispiel: BREAK in 40

Ein derart unterbrochenes Programm kann mit dem CONT-Befehl fortgesetzt werden.

Funktionstasten

PF1–PF10 Die nächsten fünf Tasten sind auf der oberen Gehäuseseite mit PF1 bis PF5 bezeichnet und sind Funktionstasten, die im BASIC-Teil genauer beschrieben sind.

Diese Tasten sind doppelt belegbar, d.h. daß dem Anwender praktisch 10 programmierbare Funktionstasten zur Verfügung stehen. Der Aufruf der Funktionen 6 bis 10 geschieht durch Drücken der SHIFT-Taste in Zusammenhang mit der entsprechenden Funktionstaste. Hierbei ist dann die mit PF1 bezeichnete Taste die Funktionstaste 6 und PF5 gleich Taste 10.

Die ersten beiden Tasten sind auf der unteren Gehäuseseite mit

CASSETTE und COPY

bezeichnet.

Zusammen mit der CTRL-Taste haben sie einige Funktionen, die den Bedienungskomfort sichtlich erhöhen.

Hierzu betätigen Sie bei niedergedrückter CTRL-Taste die entsprechende Funktionstaste.

CTRL/PF2 → COPY : Diese Tastenfolge ermöglicht eine Hardcopy, also eine Kopie der L.C.D.-Anzeige auf den Miniprinter, egal ob Graphik oder Text.
Die Hardcopy bezieht sich auf den aktuellen Bildschirm, d.h. die im virtuellen Bereich stehenden Zeichen werden nicht beücksichtigt.

CTRL PF1/ → CASSETTE : Diese Tastenfolge setzt das Mikrokassettenlaufwerk in den manuellen Operationsmodus und zeigt auf dem L.C.D. den aktuellen Wert des Tape-Counters, der zur Steuerung des Kassettenlaufwerkes eine wesentliche Hilfe ist.

Gleichzeitig werden im manuellen Modus die Funktionstasten 1 bis 6 neu belegt und haben folgende Bedeutung:

PF1: Beim Drücken dieser Taste wird das Kassettenband schnell vorwärts gespult. Der aktuelle Wert des Tape-Counters wird permanent auf dem L.C.D. angezeigt.

PF2: Bewegt das Kassettenband mit normaler Geschwindigkeit vorwärts.
Es sollte nicht versucht werden, das Laufwerk vom schnellen Vorwärtslauf ohne Zwischenstop in diesen Modus zu bringen, da dies der Mechanik und vor allem dem Band, längerfristig gesehen, schadet.

PF3: Ein Druck auf diese Taste stoppt das Band, egal ob es sich im schnellen Vor-oder Rücklauf, oder irn normalen Laufmodus befindet.

PF4: Spult das Band im schnellen Rücklauf zurück.

PF5: Beim Drücken dieser Taste, wird der manuelle Operationsmodus verlassen und in die entsprechende Ausführungsart, z.B. Programmkontrolle des Laufwerkes zurückgekehrt.

PF6: (SHIFT +PF1) Mit Hilfe dieser Taste wird der Tape-Counter auf NULL zurückgesetzt.

Die restlichen 4 Funktionstasten (SHIFT +PF2 bis SHIFT +PF5) bleiben ohne Bedeutung.
Auf der Tastatur selbst sind einige Tasten besonders herausgestellt bzw. hervorgehoben. Einige dieser Tasten wurden bereits vorgestellt. Im folgenden werden alle Tasten näher erklärt.

TAB : Beim Drücken dieser Taste wird der Cursor um 8 Stellen nach rechts bewegt.

SHIFT : Diese Taste schaltet die jeweilige Darstellung um, d.h. wenn sich der Rechner z.B. im Darstellungsmodus Großbuchstaben befindet, so wird durch Niederhalten der SHIFT-Taste bewirkt, daß der entsprechend gedrückte Buchstabe in Kleinschrift dargestellt wird.

HOME/CLR : Wird diese Taste gedrückt, so wird der aktuelle Bildschirm gelöscht.
Wird die Taste in Zusammenhang mit der SHIFT-Taste gedrückt, so bleibt die Bildschirmdarstellung erhalten, lediglich der CURSOR wird auf die HOME Position, die sich in der linken oberen Ecke des Displays befindet, gesetzt.

SCRN : Durch das Drücken dieser Taste wird die Bildschirmanzeige um einen Bildschirm nach oben verschoben. Dieses Verschieben wird "Scrollen" genannt. Mit der Tastenfolge SHIFT/SCRN wird die Bildschirmanzeige um einen Bildschirm nach unten gescrollt. Als Beispiel bietet sich an, den BASIC-Modus zu wählen und die daraufhin erscheinende Anzeige nach oben und nach unten zu verschieben.

DEL/INS : Ein Tastendruck löscht das letzte Zeichen vor der aktuellen Cursorposition.
Die Tastenfolge zusammen mit SHIFT ermöglicht das Einfügen von Zeichen in einen bestehenden Text, der dann entsprechend nach rechts verschoben wird.
Diese Funktion bleibt so lange aktiv, bis sie durch erneutes Drücken dieser Tastenfolge oder durch Bewegen der Pfeiltasten abgebrochen wird.

 : Bewegt den CURSOR um ein Zeichen nach links.

 : Bewegt den CURSOR um eine Zeile nach rechts.

Mit der SHIFT-Taste zusammen wird der CURSOR um eine Zeile nach oben bzw. unten verschoben.
Auch hier ist eine maximale Verschiebung um einen Bildschirminhalt nach unten möglich.

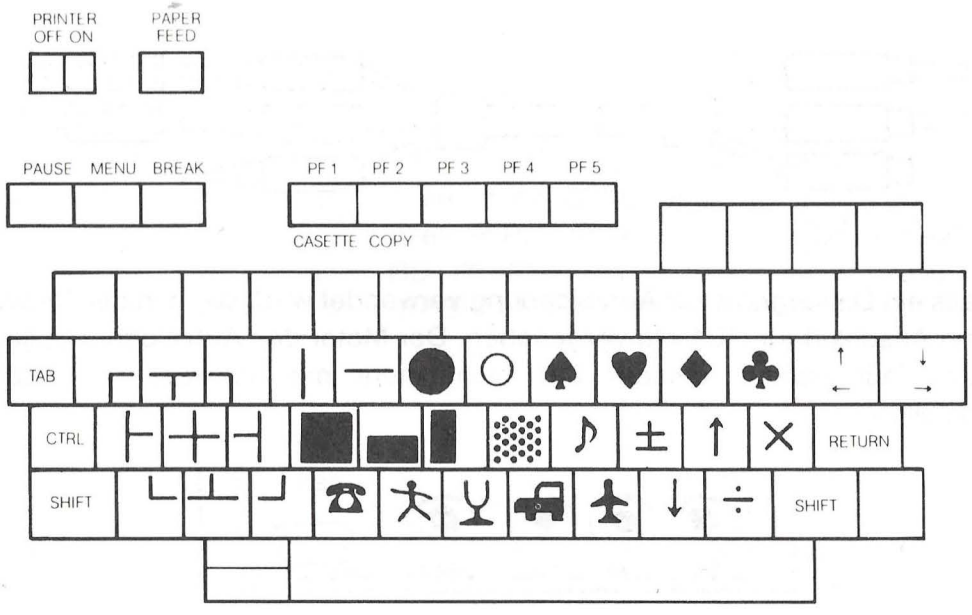
RETURN : Wagenrücklauf oder Zeilenabschluß. Diese Taste muß im Normalfall nach jeder Eingabe gedrückt werden, um den Zeileninhalt an den Computer zur Ausführung zu übergeben.

NUM : Ermöglicht die Eingabe von nur "numerischen" Werten. Alle anderen Eingaben werden nicht berücksichtigt. Diese Prozedur wird nur durch ein abermaliges Drücken dieser Taste wieder aufgehoben.

CAPS LOCK : Setzt den jeweiligen Darstellungsmodus auf Groß- oder Kleinschrift fest.

GRPH : Wird diese Taste gedrückt und festgehalten, so werden auf dem Display die entsprechenden Graphikzeichen dargestellt, mit denen die Tastatur belegt ist.

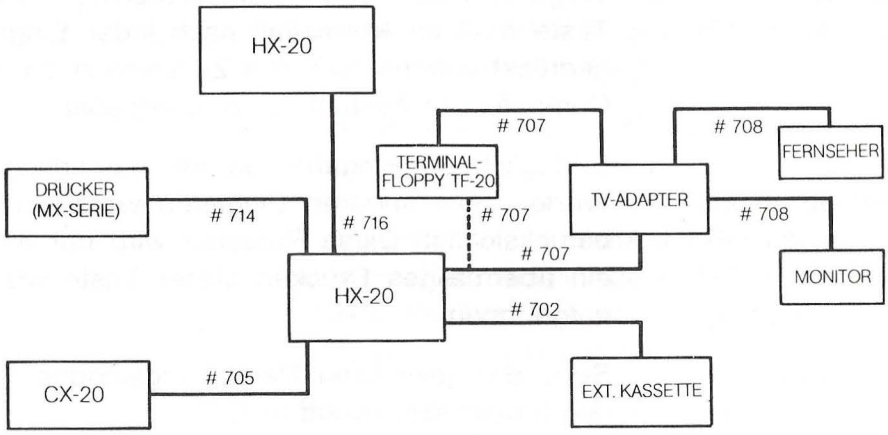
Die Belegung der Tastatur im Graphikmodus ist wie folgt:



1.5. Peripherieanschlüsse

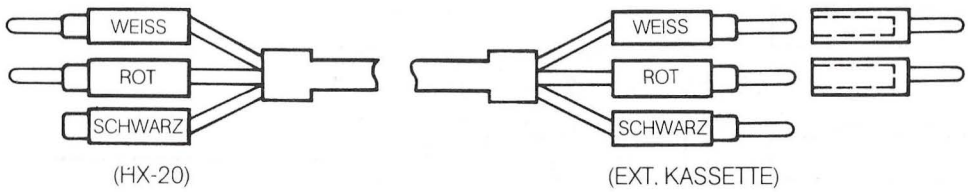
Übersichtsschema

Hier die schematische Übersicht über die von EPSON angebotenen Anschlußkabel mit ihrer Artikelnummer:

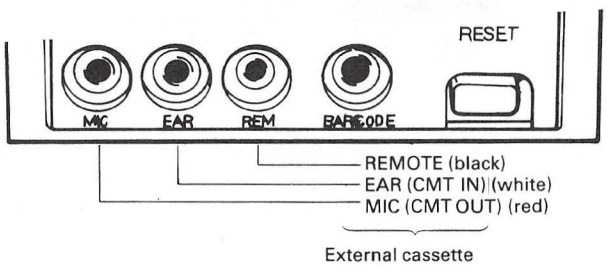


1.5.1. Externer Kassettenrekorder

Der HX-20 verfügt über ein eingebautes Kassetteninterface, das es ermöglicht, einen handelsüblichen Kassettenrekorder anzuschließen. Das Kabel ist wie folgt anzustecken:



Falls ein Diktiergerät zur Aufzeichnung verwendet wird, so ist in der Regel kein Anschluß für REMote vorgesehen. Der Motor des Aufzeichnungsgerätes läßt sich in diesem Fall nicht mehr mit Software an- oder abstellen.



1.5.2. RS-232C Anschlüsse

Über die RS-232C Schnittstelle lassen sich folgende EPSON Geräte anschließen:

- Akustikkoppler CX-20
- Computer der QX-Serie
- EPSON-Drucker

Die Geräte anderer Hersteller lassen sich ebenfalls darüber anschließen, sofern diese über eine RS-232C Schnittstelle verfügen.

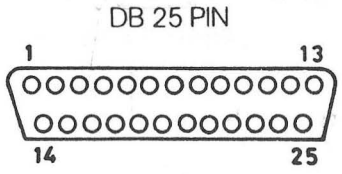
Der Stecker des HX-20 ist ein DIN 8 PIN TSC 4480

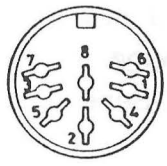
und hat folgende Bedeutung:

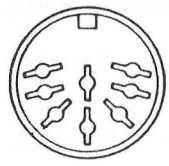
Signal pin No.	Signal Name	Signal	Beschreibung
1	GND	–	Signal Ground
2	TXD	Out	Transmitted data
3	RXD	In	Received data
4	RTS	Out	Request to send
5	CTS	In	Clear to send
6	DSR	In	Data set ready
7	DTR	Out	Data terminal ready
8	CD	In	Carrier detect
E	FG	–	Protective Ground

Wenn das Kabel selbst gelötet wird, so ist besonders darauf zu achten, daß die Steckerbezeichnung genau stimmt, da es ansonsten durchaus möglich ist, daß die Bezeichnung der einzelnen Pole anders ist. Das entsprechende Gegenstück, der Standard DB 25 Stecker ist dem Gerät passend anzuschließen, bzw. bei allen EPSON Kabeln bereits angeschlossen.

Standard DB 25 Stecker







PIN-Belegung DB 25

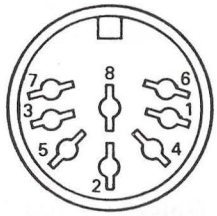
DB-25 connector

Pin No.	Signal name	Colour
1	FG	(Shield)
2	RX	White
3	TX	Red
4		Blue
5		Blue
6	DTR	Green
7	GND	Black
8		Brown
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20	DSR	Yellow
21		
22		
23		
24		
25		

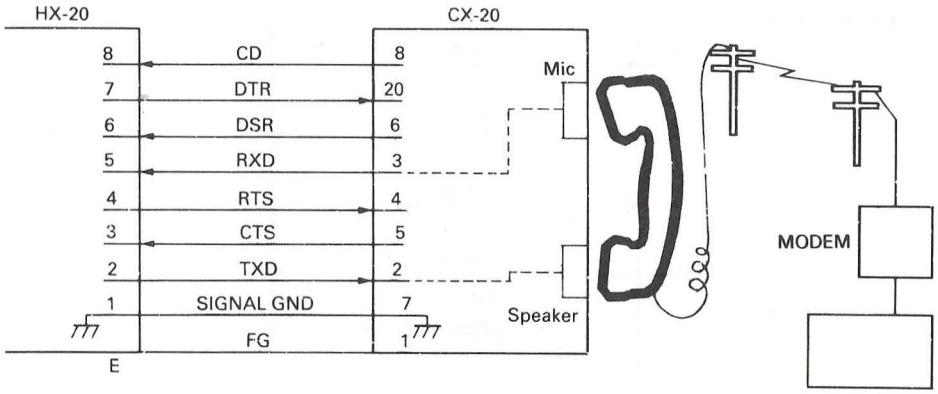
RS-232 C Schnittstelle (Akustik-Koppler)

Interface-Stecker für Datenübertragung zwischen HX-20 und dem Akustik-Koppler oder anderen Geräten.

Verbindung: 8-pin DIN, TCS4480



Signal pin No.	Signal Name	Signal	Beschreibung
1	GND	–	Signal Ground
2	TXD	Out	Transmitted data
3	RXD	In	Received data
4	RTS	Out	Request to send
5	CTS	In	Clear to send
6	DSR	In	Data set ready
7	DTR	Out	Data terminal ready
8	CD	In	Carrier detect
E	FG	–	Protective Ground

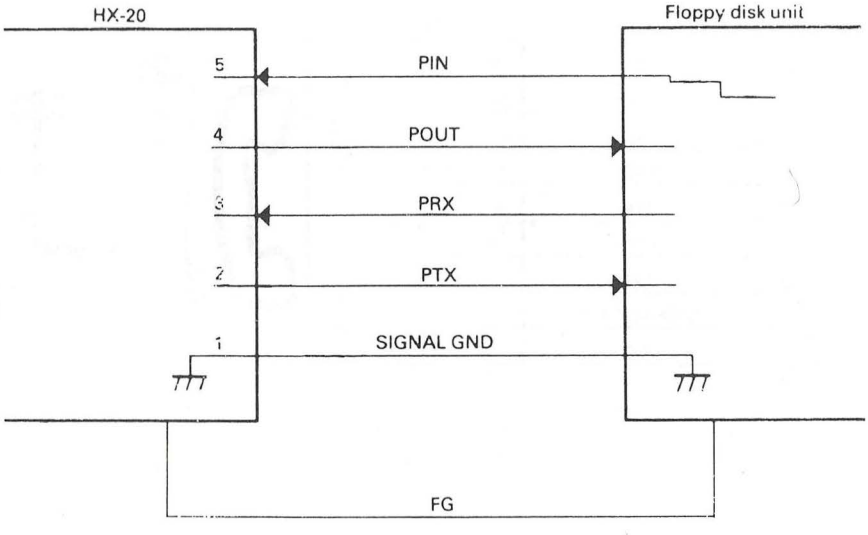


1.5.3. High Speed Serial Interface

Über diese Schnittstelle werden die Floppy-Disk Laufwerke angeschlossen. Der Stecker ist ein DIN 5 PIN mit der Bezeichnung TSC 445Ø

Die PIN-Belegung hat folgende Bedeutung:

Signal pin No.	Signal name	Direction of signal	Description
1	GND	–	Signal Ground
2	PTX	Out	Transmitted data
3	PRX	In	Received data
4	POUT	In	Transmit mode
5	PIN	Out	Receive mode
E	FG	–	Protective Ground



1.5.4. System-Bus

Für eventuelle weitere Anschlüsse, die über den System-Bus vorgenommen werden sollen, auch hier die Belegung:

PIN	SYMB	Bedeutung
1	VB	Direkte Stromversorgung von der Batterie. Unreguliert 1,Ø Amp.
2	-NMI	Active LOW
3	V	(geschaltet) 5Ø mA
4	+5V	(geschaltet)
5	DB7	Data Bit 7
6	DB6	„
7	DB5	„
8	DB4	„
9	DB3	„
1Ø	DB2	„
11	DB1	„
12	DBØ	Data Bit Ø
13	-IOCS	Adresse ØØØØ-ØØ7F I/O Chip select
14	VC	Power for RAM, 4Ø mA
15	ABØ	Adress Bus
16	AB1	“
„	„	„
„	„	„
3Ø	AB15	Adress Bus
31	-RESET	CPU Reset, active low
32	-R/W	Memory read (low) write (high)
33	-RESET	RAM protect bei power off
34	E	von CPU clock
35	ROM EN	low, disable ROM
36	-EXT INT	External interupt
37	GND	Ground
38	GND	Ground
39	CG	Case ground
4Ø	CG	Case ground

1.6. Spezifikationen des HX-20

CPU CMOS, 8 Bit (6301 Main- und Slave-CPU) 614 KHz
ROM CMOS, 32 KByte, intern um 8 KByte erweiterbar
RAM CMOS, 16 KByte, um 16 KByte erweiterbar

Maße:	DIN A 4
	290 mm lang
	215 mm breit
	44 mm hoch
Gewicht:	1,6 kg

Temperatur:

Operation:	5 bis 35 C
Laden:	5 bis 33 C
Datensicherheit:	–5 bis 40 C
Lagerung:	–20 bis 60 C

Feuchtigkeit:

Operation:	10 bis 80 % (ohne Kondensation)
Ausgeschaltet:	10 bis 80 % (ohne Kondensation)

AC Adapter:

Eingangsspannung:	220 Volt +/-10%
Frequenz:	50/60 Hz
Ausgangsspannung:	6 Volt, 600 mA
Verbrauch:	8 Watt

Batterien:

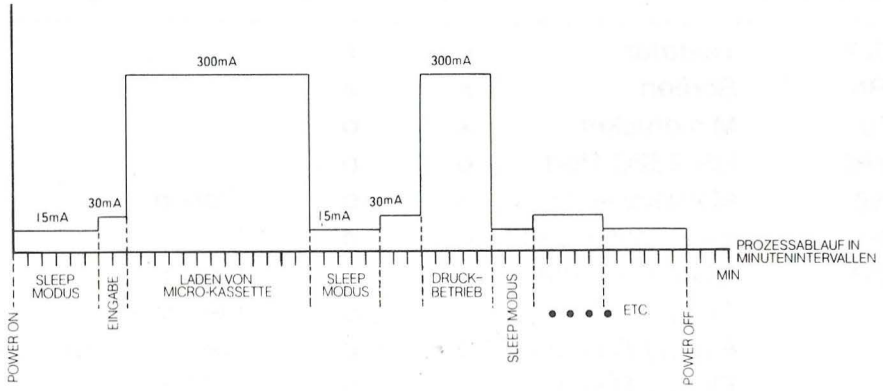
NiCd Batterien:	(SUB C*4) 1100 mAh
Batteriespannung:	4,5–6,0 Volt
Datensicherheit:	4,0–6,0 Volt
Niedrigste Spannung:	4,5 Volt

Aufladung:
Das Gerät lädt sich innerhalb von 8 Stunden voll auf, länger sollte der HX-20 nicht am Netz bleiben.

Leistungsdauer:	min. 24 Std.
nur Minidrucker:	max. 1 Std.
nur Mikrokassette:	max. 1 Std.

1.6.1. Leistungsaufnahme

Zur besseren Übersicht über den Stromverbrauch bei den einzelnen Funktionen folgendes Schema:



1.7. Gerätenamen

Der HX-20 nimmt nur folgende I/O Gerätenamen an:

Geräte Name	Gerät	Input	Output	Bemerkung
KYBD:	Tastatur	o	x	
SCRN:	Screen	x	o	
LPT0:	Minidrucker	x	o	
COM0:	RS-232C Port	o	o	
CAS0:	Mikrokassette	o	o	Option
CAS1:	externe Kassette	o	o	
PAC0:	ROM Cartridge	o	x	Option
A:	Floppy Disk A	o	o	benutzte
B:	Floppy Disk B	o	o	Gerätenamen
C:	Floppy Disk C	o	o	in DISK
D:	Floppy Disk D	o	o	BASIC

Die Gerätenamen LPT0:, COM0:, CAS0:, PAC0: sind mit einer Null einzugeben.

Wird kein Gerätename angegeben, prüft EPSON BASIC automatisch, ob es sich bei dem Gerät um das Mikrokassettenlaufwerk oder die ROM-Cartidge handelt.

Die Bedeutung für Input/Output ist folgende:

o = anwendbar

x = nicht anwendbar

BASIC

BASIC Allgemein

Start der BASIC-Funktion beim HX-20

Nach dem Einschalten des HX-20 erscheint folgendes Bild auf dem Display:

```
CTRL/S Initialize
1 MONITOR
2 BASIC
```

Wird nun die Taste "2" BASIC gedrückt, meldet sich der BASIC-Modus wie folgt:

```
Copyright 1982 by
Microsoft & EPSON
P1:          0 Bytes
2
```

Automatisch steht der Cursor nun im LOGIN Bereich "1". In den ersten Programmbereich kann nun ein BASIC-Programm geschrieben werden.

Der HX-20 kann auch ohne Programm als Rechner in der Funktion "2" BASIC benutzt werden.

Mit der Eingabe PRINT oder statt PRINT "?" können beliebige Rechenoperationen durchgeführt werden. (z.B. ? 450*120 = 54000)

Jedes BASIC-Programm besteht aus Anweisungen (Statements), die Befehle, Kommandos, Funktionen etc. enthalten, die das System anweisen, bestimmte Operationen auszuführen.

Kontrollzeichen

Die Control-Taste (im folgenden als CTRL bezeichnet) kann nur in Verbindung mit einer anderen Taste benutzt werden, um eine bestimmte systemdefinierte Funktion einzuleiten. Diese Funktionen lassen sich programmieren, was eine Vielzahl von nützlichen Möglichkeiten bietet, ein Programm einfacher zu gestalten.

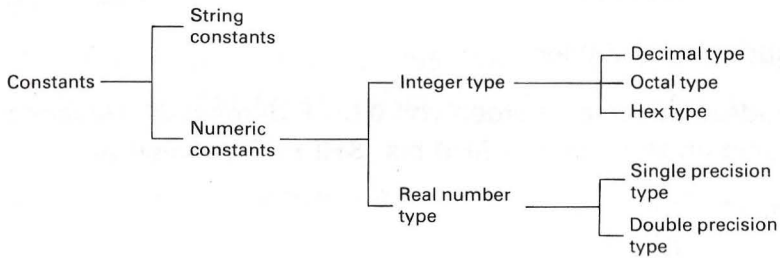
Im folgenden werden diese Zeichen mit dem entsprechenden programmierbaren Wert dargestellt und erläutert.

Hexadezimal CHR$(&Hnn)	Dezimal CHR$(nn)	Bedeutung CTRL/Bedeutung
Ø1	1	CTRL A: Versetzt den physikalischen Schirm nach links auf den virtuellen Schirm.
Ø3	3	CTRL C: Unterbricht den AUTO Modus.
Ø4	4	CTRL D: Rollt die Anzeige um 1Ø Stellen nach links.
Ø5	5	CTRL E: Löscht ab einer bestimmten Position bis zum Zeilenende.
Ø6	6	CTRL F: Versetzt den physikalischen Schirm nach rechts auf den virtuellen Schirm.
Ø8	8	CTRL H: Löscht ein Zeichen, wie Delete oder Backspace
Ø9	9	CTRL I: Horizontaler Tabulator, wie TAB
ØA	1Ø	CTRL J: LINE FEED
ØB	11	CTRL K: Cursor Home Position, wie HOME
ØC	12	CTRL L: Löscht die Anzeige, wie CLS
ØD	13	CTRL M: Carriage Return oder Wagenrücklauf, wie RETURN-Taste
1Ø	16	CTRL P: Rollt das Display nach oben
11	17	CTRL Q: Rollt das Display nach unten
12	18	CTRL R: Schaltet in den Insertmodus um, wie Taste INS
13	19	CTRL S: Rollt die Anzeige um 1Ø Stellen nach rechts.
16	22	CTRL V: Schaltet den Cursor an.
17	23	CTRL W: Schaltet den Cursor ab.
1A	26	CTRL Z: Löscht ab dieser Stelle bis zum Ende des Bildschirms
1C	28	↓→ Versetzt den Cursor nach rechts.
1D	29	↙ Versetzt den Cursor nach links.
1E	3Ø	SHIFT ↑ : Versetzt den Cursor nach oben.
1F	31	SHIFT ↓→ : Versetzt den Cursor nach unten.

Im Text wird CTRL mit dem Exponentialzeichen wie folgt dargestellt:
Beispiel: CTRL D ist dann ^D

Konstanten

EPSON-BASIC unterscheidet grundsätzlich zwei Arten von Konstanten. Das ist einmal die Zeichenketten-Konstante, und zum anderen die numerischen Konstante. Sie lassen sich durch folgendes Schema verdeutlichen:



String-Konstanten

Eine String-Konstante kann aus maximal 255 Zeichen bestehen, inclusive Symbolen. Die Anführungszeichen (") beenden die Stringeingabe. Mit CHR$ können die Anführungszeichen im ASCII-Code angegeben werden.

Beispiel: "EPSON"
"HX-20"

Numerische Konstanten

1. Dezimalzahlen

   Dies sind ganze Zahlen (Integer), die zwischen -32768 und 32767 liegen, und keinen Dezimalpunkt haben. Nur das %-Zeichen kann zusätzlich eingegeben werden.

   Beispiel: 12345
   –9999
   45678%

2. Oktalzahlen

   Oktalzahlen müssen das Vorzeichen Ø oder & haben.

   Oktalzahlen sind im Bereich von &Ø bis &177777.

   Beispiel: &Ø123
   &77777

3. Hexadezimale Zahlen

   Hexadezimalzahlen werden von Ø bis F dargestellt. Hexadezimalzahlen sind im Bereich von &HØ bis &HFFFF definierbar.

   Beispiel: &H1F
   &HABCD

Konstanten mit einfacher Genauigkeit

Konstanten mit einfacher Genauigkeit haben folgende Merkmale:

1. sieben oder weniger Ziffern – oder
2. Exponentialform mit E – oder
3. ein nachfolgendes Ausrufezeichen (!).

Beispiel: 12345.6
–5E–38
234.5!

Konstanten mit doppelter Genauigkeit

Eine doppelt genaue Konstante ist mit 16 Ziffern gekennzeichnet, und hat folgende Merkmale:

1. acht oder mehr Ziffern – oder
2. Exponentialform mit D – oder
3. ein nachfolgendes Nummernzeichenn (#).

Beispiel: 3.121592653
–1.23 D 22
888.8#

Variablen

Variablen sind Namen, die zur Darstellung von Werten in einem BASIC-Programm benutzt werden. Der Wert einer Variablen wird durch den Programmierer bestimmt oder ist das Ergebnis einer Berechnung im Programm. Wird eine Variable bestimmt und ihr kein Wert zugewiesen, so ist ihr Wert generell NULL.

Ein Variablenname kann maximal 255 Zeichen lang sein. Das erste Zeichen muß ein Buchstabe sein. Nur die ersten 16 Zeichen sind die Identifikation.

Variablen stellen entweder einen numerischen Wert oder eine Zeichenkette (String) dar.

Zeichenkettenvariablen (String) werden mit einem Dollarkennzeichen ($) als letztes Zeichen versehen.

Numerische Variablen können ganzzahlige Werte oder Werte mit einfacher und doppelter Genauigkeit sein. Die Deklaration der Zeichen wird als letztes Zeichen angehängt und sieht wie folgt aus:

% ganzzahlige Variablen (Integer)
! Variablen mit einfacher Genauigkeit (single precision)
Variablen mit doppelter Genauigkeit (double precision)
$ String Variable

Wird die Deklaration nicht angegeben, ist es automatisch eine Variable mit einfacher Genauigkeit.

Feldvariable

Eine Feldvariable mit mehreren Elementen kann mit einem Variablennamen versehen werden. Die Dimensionierung und der maximale Wert richtet sich nach der Kapazität des Speichers.

Umwandlungen der Variablentypen

Eine numerische Variable läßt sich nicht in eine String Variable umwandeln. Fehler " Type Mismatch " TM. Numerische Konstanten werden von einem Typ in einen anderen Typ nach folgenden Regeln automatisch umgewandelt:

Beispiele:	A% = 32.34	⇒ 32
	A#=10/3*3#	⇒ 9.999999761581421
	B#=10#/3*3	⇒ 1.0E+01
	A=NOT 123.456	⇒ −124
	P!=3.141592653589793	⇒ 3.141593

Ist eine Variable bestimmt, so wird im Speicher des HX-20 folgender Platz dafür benötigt:

$-String	–255 Zeichen)	3 + Anzahl Zeichen Bytes
%-Integer	(–32768 +32767)	2 Bytes
!-Einfach	(7.1 Zahlen)	4 Bytes
#-Doppelt	(16.8 Zahlen)	8 Bytes

Ausdrücke und Operatoren

Ein Ausdruck können Strings oder numerische Konstanten, Strings oder numerische Variablen, oder eine Kombination von String und numerischen Konstanten sein, die zu einer Operation verbunden werden.

Operationen können mathematische oder logische Berechnungen sein. BASIC teilt diese Operatoren in fünf Gruppen auf:

1. arithmetische
2. vergleichende
3. logische
4. funktionelle
5. String

Arithmetische Operatoren

EPSON BASIC benutzt folgende arithmetischen Operationen:

Operator	Operation	Beispiel
^	Exponentation	A^B
–	Negation	–A
*, /	Multiplikation und Division	A*B, A/B
+, –	Addition und Subtraktion	A+B, A–B

Bei der Berechnung der Ausdrücke wird der Rang der entsprechenden Operation berücksichtigt; Operationen mit höherem Rang werden vor denen mit niedrigerem Rang ausgeführt. Werden Klammern benutzt, so werden die Operationen innerhalb der Klammern als erste ausgeführt.

Operationen mit gleichem Rang werden von links nach rechts ausgeführt. Zwei fortlaufende Operatoren müssen durch Klammern getrennt werden.

EPSON BASIC	Algebraischer Ausdruck
3*X+Y	3X+Y
X/Y–Z	(X/Y)–Z
X^2+Y*3+4	X^2+3Y+4
Y^Y^Z	$(X^Y)^2$
X*(–Y)	X(–Y)

Eine Division durch Null erzeugt einen "Division by zero"-Error, und die Operation wird beendet.

Vergleichsoperatoren

Vergleichsoperatoren vergleichen zwei Werte. Das Ergebnis dieses Vergleichs ist entweder wahr oder nicht wahr. Dieses Ergebnis kann dazu benutzt werden, eine Entscheidung für den weiteren Programmablauf zu treffen.

Operator	Bedeutung	Ausdruck
=	Gleich	X = Y
< >, ><	Ungleich	X<>Y, X><Y
>	Kleiner als	X < Y
>	Größer als	X > Y
<=, =<	Kleiner gleich	X<= Y, X=<Y
>=, =>	Größer gleich	X>= Y, X=>Y

Wenn arithmetische und vergleichende Operatoren in einem Ausdruck kombiniert sind, wird der arithmetische Operator immer zuerst ausgewertet.

Beispiel: X+Y < (T–1)/Z

Dieser Ausdruck ist richtig, wenn der Wert X plus Y kleiner ist, als der Wert T–1 dividiert durch Z.

Logische Operatoren

Logische Operatoren führen mehrere Vergleichsprüfungen aus. Der logische Operator gibt das Ergebnis, das entweder WAHR (nicht NULL) oder UNWAHR (NULL) ist, aus.

In einem Ausdruck werden logische Operatoren nach arithmetischen oder vergleichenden Operationen ausgewertet.

In der folgenden Tabelle sind die logischen Operatoren in ihrer Ausführungsreihenfolge aufgeführt:

NOT (Negation)

X	NOT X
1	Ø
Ø	1

AND (Logical product)

X	Y	X AND Y
1	1	1
1	Ø	Ø
Ø	1	Ø
Ø	Ø	Ø

OR (Logical sum)

X	Y	X OR Y
1	1	1
1	Ø	1
Ø	1	1
Ø	Ø	Ø

XOR (Exclusive-OR)

X	Y	X OR Y
1	1	Ø
1	Ø	1
Ø	1	1
Ø	Ø	Ø

IMP (Implikation)

X	Y	X IMP Y
1	1	1
1	Ø	Ø
Ø	1	1
Ø	Ø	1

EQV (Equivalence)

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

Genau wie bei den Vergleichsoperatoren können sie für Entscheidungen hinsichtlich des weiteren Programmablaufes dienen. Logische Operatoren kann man mit mehreren Vergleichen verknüpfen und die erhaltenen wahren oder unwahren Werte für weitere Entscheidungen benutzen.

Beispiel: 40 IF A<0 AND B=0 THEN 100

In diesem Beispiel ist A negativ, und B 0, es erfolgt ein Sprung zur Zeile 100.

Funktionelle Operatoren

Eine Funktion in einem Ausdruck wird benutzt, um eine vorher festgelegte Operation, die an einem Operanden ausgeführt werden soll, aufzurufen.

EBASIC besitzt solche Funktionen, die fest im System implementiert sind, wie zum Beispiel SQR, SIN, etc. und String Funktionen wie RIGHT$, STR$, etc.

String-Operationen

Auch mit Strings können Operationen durchgeführt werden. Mit einem Pluszeichen (+) lassen sich Strings bis maximal 255 Zeichen verknüpfen.

Beispiel:
```
10 A$="EPSON": B$="HX-20"
20 PRINT A$+B$
RUN
EPSON HX-20
```

Wie numerische Werte, so können auch Strings mit Vergleichsoperatoren versehen werden:

=, <, >, <>, ><, <=, =<, >=, =>

Von jedem String wird ein Zeichen mit dem ASCII-Code verglichen. Werden zwei Strings verglichen, hat die kleinere Codezahl die höhere Priorität. Die Leerzeichen werden beim Vergleich berücksichtigt.

Beispiel:
```
"AAA" = "AAA"
"XYZ" > "XY"
```

Liste der Operatoren

1. Operationen mit Klammern
2. Funktionen
3. Exponentation (^)
4. Negation (–)
5. Multiplikation und Division (*, /))
6. Ganzzahl-Division (\) (im deutschen Zeichensatz Ö)
7. Modulus Integer Division (MOD)
8. Addition und Subtraktion (+,–)
9. Vergleichsoperationen
10. NOT
11. AND
12. OR
13. XOR
14. IMP
15. EQV

Eingabe der BASIC-Befehle und Anweisungen

Wörter in Großbuchstaben müssen wie angezeigt eingegeben werden.

In Spitzklammern (⟨ ⟩) stehende Werte sind vom Anwender einzugeben.

Symbole wie Klammern, Kommas, Semikolons, Gleich, etc. müssen exakt wie beschrieben eingegeben werden.

Wörter oder Zeichen in eckigen Klammern ([ ]) sind wahlfrei, d.h. sie können bei Bedarf benutzt werden.

Wörter, gefolgt von Auslassungszeichen (...), können beliebig wiederholt werden, bis maximal 255 Zeichen.

Wörter, die durch einen vertikalen Strich | getrennt sind, schließen sich gegenseitig aus; es ist eines davon zu wählen.

Eine Falscheingabe kann mit der DEL-Taste einzeln gelöscht oder überschrieben werden. Die INS/DEL mit der SHIFT-Taste ermöglicht das Einfügen von Zeichen.

ABS

FORMAT: ABS (⟨X⟩)

FUNKTION: Gibt den absoluten Wert des numerischen Wertes (X) an.

BEMERKUNG: Der Ausdruck (numerischer Wert) wird mit ABS absolut wiedergegeben.

BEISPIEL:

```
10 PRINT ABS (7*(-5))
 35
```

ASC

FORMAT: ASC (⟨X$⟩)

FUNKTION: Gibt den ASCII-Code eines Zeichens als Dezimalzahl an.

BEMERKUNG: ASC wandelt das erste Zeichen eines Strings (X$) in die dezimale Darstellung des ASCII-Codes um.

Soll nur ein Zeichen umgewandelt werden, muß X$ mit Hochkommas angegeben werden.

Ist der String (X$) Null, erfolgt die Fehlermeldung "FC" (Illegal Function Call).

BEISPIEL:

```
10 X$="EPSON"
20 PRINT ASC(X$)
30 PRINT ASC("D")
40 END
 69
 68
```

ATN

FORMAT: ATN (⟨X = numerischer Ausdruck⟩)

FUNKTION: Ermittelt den Arc Tangens von X.

BEMERKUNG: ATN gibt den Arc Tangens von X in Radiantenform wieder.

Das Resultat der Operation ist im Bereich von -PI/2 bis PI/2.

BEISPIEL:

```
10 X=5
20 PRINT ATN(X)
30 END

 1.3734
```

AUTO

FORMAT: AUTO [⟨Zeilennummer⟩] [, [⟨Schrittgröße⟩]]

FUNKTION: Erzeugt automatisch eine Zeilennummer nach jedem RETURN.

BEMERKUNG: AUTO beginnt mit der angegebenen Zeilennummer und erhöht diese um die angegebene Schrittgröße.

Wird keine Zeilennummer angegeben, so wird automatisch bei Zeilennummer 10 begonnen. Ist keine Schrittgröße genannt, so wird dafür ebenfalls 10 angenommen.

AUTO wird mit ^C (Control-C) ausgeschaltet. Dabei wird die Zeile, in der das Abbruchkommando erfolgte, nicht mehr gespeichert.

BEISPIEL:

`AUTO 100,20`	Beginnt mit Zeilennummer 100 und erhöht jeweils um Schrittgröße 20
`AUTO`	Erzeugt die Zeilennummern 10, 20, 30, usw.
`AUTO 150`	Erzeugt die Zeilennummern 150, 160,usw.

CDBL

FORMAT: CDBL (⟨X⟩)

FUNKTION: Konvertiert X in eine Zahl mit doppelter Genauigkeit.

BEMERKUNG: CDBL konvertiert den Ausdruck X in eine Variable mit doppelter Genauigkeit. X kann eine Ganzzahl oder ein Wert mit einfacher Genauigkeit sein.

BEISPIEL:

```
10 X=50.35
20 PRINT CDBL(X)
30 END
 50.3499984741211
```

CHR$

FORMAT: CHR$ (⟨X⟩)

FUNKTION: Wandelt den ASCII-Wert X in das entsprechende Zeichen um.

BEMERKUNG: Das ASCII-Code Element (X) wird in ein lesbares Zeichen umgewandelt.

X muß im Bereich von 0 bis 255 liegen. Die ersten 32 Steuerzeichen lassen sich nicht darstellen.

BEISPIEL:

```
10 FOR I=33 TO 75
20 PRINT CHR$(I)
30 NEXT I
40 END
!
"
#
$
%
&
'
(
)
*
+
,
-
.
/
0
1
2
3
4
5
6                    A
7                    B
8                    C
9                    D
:                    E
;                    F
<                    G
=                    H
>                    I
?                    J
@                    K
```

CINT

FORMAT: CINT (⟨X⟩)

FUNKTION: Rundet den Wert X ganzzahlig auf oder ab.

BEMERKUNG: CINT rundet die Nachkommastellen des Ausdrucks X auf den ganzzahligen Wert auf oder ab.

X muß im Bereich von -32768 bis 32767 liegen, ansonsten erscheint ein OV (Overflow) Error.

BEISPIEL:

```
10 X=65.78
20 PRINT CINT(X)
30 END

 66
```

CLEAR

FORMAT: CLEAR [⟨Variablenbereich⟩ [,⟨RAM-File Größe⟩]]

FUNKTION: Löscht den Variablen- und den RAM-File-Bereich.

BEMERKUNG: CLEAR löscht den Variablenbereich und ermöglicht eine neue Bereichsdimensionierung. CLEAR schließt alle geöffneten Dateien. Der Variablenbereich ist standardmäßig auf 200 Bytes definiert. Ein zu langer String in einem zu klein definierten Variablenbereich, erzeugt ein "OS" Error (out of String).

Die RAM-File Größe wird mit CLEAR auf Null gesetzt. Die RAM-File Größe liegt standardmäßig bei 256 Bytes.

Mit dem Definieren der RAM-File Größe muß auch der Variablenbereich neu angelegt werden.

BEISPIEL:

```
10 CLEAR 200,650
20 DEFINT U,A,H,R
30 DEFFIL 26,0
```

CLOSE

Format: CLOSE [[#]⟨Dateinummer⟩,[#]⟨Dateinummer⟩...]]

Funktion: Schließen einer Datei /Dateien

Bemerkung: Die Dateinummer ist die Nummer, unter welcher eine Datei geöffnet wurde. CLOSE schließt alle geöffneten Dateien.

Ein STOP hingegen unterbricht die Programmausführung, schließt aber keine Dateien.

Eine geschlossene Datei darf unmittelbar nach Ausführung des CLOSE Befehls wieder geöffnet werden.

Nach CLEAR, LOGIN, NEW, DELETE, WIDTH, LOAD, RUN oder MERGE sind alle Dateien geschlossen.

BEISPIEL:

```
10 OPEN"O",#1,"COM0:(48N
2F)
20 INPUT A$
30 PRINT #1,A$
40 GOTO 20
50 CLOSE #1
60 END
```

CLS

FORMAT: CLS

FUNKTION: Löscht die Text-Anzeige auf dem L.C.D.

BEMERKUNG: CLS löscht nur die Text-Anzeige.
Die Graphik bleibt auf dem Display.

BEISPIEL:

```
10 CLS
20 PRINT"EPSON HX-20"
30 INPUT A$
40 GOTO 10
50 END
```

COLOR

FORMAT: COLOR [⟨Vordergrundfarbe⟩] [, [⟨Hintergrundfarbe⟩] [,⟨Color Set⟩]]

FUNKTION: Spezifiziert die Farbdarstellung im externen Bildschirm.

BEMERKUNG: COLOR benutzt die gewählte Vordergrund- und Hintergrundfarbe für ein externes Display. Es ist möglich, zwischen 8 Farben zu wählen. Die COLOR-Codes sind von Ø bis 3 festgelegt.

BEISPIEL: COLOR Ø,3,Ø

	Farbcodes	
Color set Ø	Ø:	Grün
	1:	Gelb
	2:	Blau
	3:	Rot
Color set 1	Ø:	Weiss
	1:	Cyan
	2:	Magenta
	3:	Orange

Nach dem Warmstart ist standardmäßig eingestellt:

Vordergrundfarbe: 1

Hintergrundfarbe: Ø

Color set : Ø

CONT

FORMAT: CONT

FUNKTION: Fortsetzung eines unterbrochenen Programms.

BEMERKUNG: CONT setzt ein mit der BREAK-Taste oder nach einer STOP-Anweisung unterbrochenes Programm an der gleichen Stelle fort.

BEISPIEL:

```
10 FOR I=1 TO 10
20 I=I+1
30 PRINT I
40 NEXT I
50 END
 2
 4
 6
 8
 10


Break In 30
CONT
 10
```

COPY

FORMAT: COPY

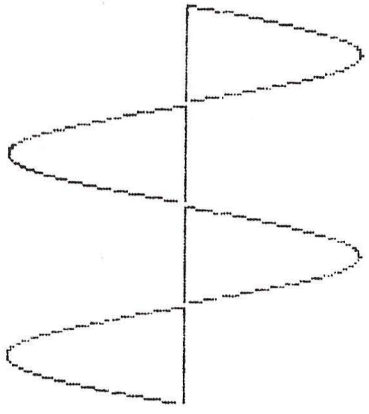
FUNKTION: Kopiert den L.C.D.-Inhalt auf den Minidrucker.

BEMERKUNG: Das COPY-Kommando ohne ein LPRINT Statement druckt den gesamten L.C.D.-Inhalt auf dem eingebauten Minidrucker aus.

Dies gilt sowohl für den Text- als auch für den Graphik-Schirm.

BEISPIEL:

```
10 PI=3.141592
20 DX=PI/64
30 CLS
40 PRINT CHR$(23)
50 LINE(60,31)-(60,0),PS
ET
60 FOR Y=0 TO 63
70 A=SIN(X)*59
80 X=X+DX
90 LINE-(A+60,Y/2),PSET
100 NEXT Y
110 COPY
120 GOTO 30
130 END
```



COS

FORMAT: COS (⟨X⟩)

FUNKTION: Gibt den Cosinus von X wieder.

BEMERKUNG: Das Ergebnis von COS (X) wird in Radiantenform wiedergegeben.

BEISPIEL:

```
10 X=2*COS(.5)
20 PRINT X
30 END
 1.75516
```

CSNG

FORMAT: CSNG (⟨X⟩)

FUNKTION: Konvertiert X in einfache Genauigkeit.

BEMERKUNG: CSNG konvertiert den Ausdruck X von einer Ganzzahl oder einer Zahl mit doppelten Genauigkeit in einen Wert mit einfacher Genauigkeit.

Der Ausdruck X muß im Bereich von -1.70141E+38 bis 1.70141 E+38 liegen.

BEISPIEL:

```
10 X=975.798569
20 PRINT CSNG(X)
30 END
 975.799
```

CSRLIN

FORMAT: CSRLIN

FUNKTION: Gibt Cursorposition in dem virtuellen Bildschirm an.

BEMERKUNG: CSRLIN gibt die vertikale Position des Cursors im virtuellen Bildschirm wieder.

BEISPIEL:

```
10 Y=CSRLIN
20 PRINT Y
30 END
 7
```

CVI, CVS, CVD

FORMAT:

CVI (⟨2-Byte String⟩)
CVS (⟨4-Byte String⟩)
CVD (⟨8-Byte String⟩)

FUNKTION:

Konvertiert einen String in einen numerischen Ausdruck.

BEMERKUNG:

Diese Funktionen rekonvertieren einen String-Typ zurück zu numerischen Werten.

CVI konvertiert einen 2-Byte String zu einer Ganzzahl.

CVS konvertiert einen 4-Byte String zu einem einfach genauen Wert.

CVD konvertiert einen 8-Byte String zu einem doppelt genauen Wert.

Ist die Länge des Strings kleiner als die auszuführende Funktion, erscheint ein FC-(Function Call) Error.

Ist der String länger, wird die Funktion vom Anfang des Strings an nur mit der nötigen Länge ausgeführt.

BEISPIEL:

```
70 FIELD #1,4 AS N$, 12
AS B$,...
80 GET #1
90 Y=CVS(N$)
```

DATA

FORMAT: DATA ⟨Konstantenliste⟩[,⟨Konstanten⟩...]

FUNKTION: Speichert numerische und String Konstanten für die READ-Anweisung.

BEMERKUNG: DATA ermöglicht die Speicherung numerischer oder alphanumerischer Konstanten, die durch Komma getrennt sein müssen. Die DATA-Anweisung kann an beliebigiger Stelle im Programm stehen.

Mit der READ-Anweisung werden die DATA-Konstanten aufgerufen.

Soll die Konstantenliste mehrmals benutzt werden, muß mittels RESTORE zurückgesetzt werden.

BEISPIEL:

```
10 READ A,B,C
20 PRINT A;B;C
30 DATA 1,2,3,4,5,6,7,8,
9
40 READ A,B,C
50 PRINT A;B;C
60 READ A,B,C
70 PRINT A;B;C
80 END

 1  2  3
 4  5  6
 7  8  9
```

DATE$

FORMAT: DATE$ [="MM/DD/YY"]

FUNKTION: Zeigt das aktuelle Datum an.

BEMERKUNG: DATE$ zeigt das internationale Datum in der Reihenfolge: Monat, Tag, Jahr (MM,DD,YY) an.

Nach dem Initialisieren des HX-20 muß das Datum in dieser Reihenfolge eingegeben werden. Es ist dann jederzeit über DATE$ aufrufbar.

Die Eingabe erfolgt in diesem Format: Date$="MM/DD/YY".

BEISPIEL:

```
P3:                0 Bytes
?DATE$
02/21/83
>

DATE$="02/22/83"
?DATE$
02/22/83
>
```

DAY

FORMAT: DAY = n

FUNKTION: Zeigt den Wochentag numerisch an.

BEMERKUNG: Die Wochentage können im HX-20 numerisch abgefragt werden.

Um den aktuellen Tag zu setzen, ist folgende Eingabe nötig: (n steht für den numerischen Wert.

DAY = 1 (Sonntag)
DAY = 2 (Montag)
DAY = 3 (Dienstag)
DAY = 4 (Mittwoch)
DAY = 5 (Donnerstag)
DAY = 6 (Freitag)
DAY = 7 (Samstag)

Der aktuelle Tag wird einmal eingegeben, danach wird er automatisch von der Systemuhr versorgt.

Die Anzeige erfolgt in numerischer Angabe.

BEISPIEL:

```
DAY=2
?DAY
 2
>
```

DEFFIL

FORMAT: DEFFIL ⟨Satzlänge⟩, ⟨Startpunkt⟩

FUNKTION: Definieren von RAM-Files.

BEMERKUNG: Mit DEFFIL wird die Satzlänge in Bytes definiert.

Standardmäßig sind 255 Bytes pro Satz reserviert.

Der Startpunkt giebt an, bei welchem Byte der erste Datensatz anfangen soll. Der Startpunkt muß innerhalb, des mittels CLEAR definierten RAM Bereichs liegen.

Die RAM-Files sind ohne Bezeichnung, sie müssen vom Anwender selbst verwaltet werden.

Folgende Kommandos zerstören das DEFFIL Kommando:

- CLEAR
- LOGIN
- LOAD
- MERGE

BEISPIEL:

```
100 DEFINT I,J,K
110 DEFFIL 2,0
120 FOR I=0 TO 15
130 PUT% I,I
135 PRINT TAB(3);
140 PRINT USING "  & &";
HEX$(I);
150 NEXT I:
160 DEFFIL 2,1
170 FOR J=0 TO 15
180 GET% J,K
185 PRINT TAB(3);
190 PRINT USING "  & &";
HEX$(K);
200 NEXT J
210 END
```

DEF FN

FORMAT: DEF FN ⟨Name⟩ [(⟨Parameterliste⟩)]=⟨Funktion Definition⟩

FUNKTION: Definiert eine vom Anwender bestimmte Funktion

BEMERKUNG: Name muß ein alphabetischer Variablenname sein, der der Funktion zugeordnet wird. Die Parameterliste wird mit ihren Variablennamen von der Funktion aufgerufen, sie müssen durch Komma getrennt sein. Der Funktionsaufruf wird durch die Parameter versorgt. Die Funktionsdefinition kann numerisch, ein String oder beides sein.

Unter dem Namen AB wird der Parameter X mit der Funktion Y*2+2 definiert. Y erhält einen Wert.

BEISPIEL:

```
10 DEFFNA(B,C,R)=SQR(B^2
+C^2-2*B*C*COS(R/180*3.1
415926))
20 INPUT"LÄNGE B=";B
30 INPUT"BREITE C=";C
40 INPUT"HöHE R=";R
50 A=FNA(B,C,R)
60 PRINT "SEITE A =";A
70 GOTO 20
80 END
```

DEFINT /SNG /DBL /STR

FORMAT: DEFINT | DEFSNG | DEFDBL | DEFSTR ⟨Variablentyp⟩ ⟨Variablenbezeichnung⟩

FUNKTION: Variablentypen werden deklariert

BEMERKUNG: DEFINT, DEFSNG, DEFDBL und DEFSTR deklarieren den Variablentyp: Ganzzahl, einfach genau, doppelt genau oder als String. Alle Variablennamen beginnen mit einem Buchstaben und spezifizieren den Typ. Wird kein Typ definiert, gilt standardmäßig für alle Variablen die einfach genaue Definition.

BEISPIEL:

```
10 DEFINT A-D
20 DEFDBL E
30 A=123.456789
40 E=789.123456
50 PRINT A
60 PRINT E
70 END
```

```
 123
 789.123456
```

DEF USR

FORMAT: DEF USR [⟨Nummer⟩] = ⟨Start Adresse⟩

FUNKTION: Startadresse für ein Assembler Unterprogramm.

BEMERKUNG: DEF USR spezifiziert die Startadresse für ein Assembler-Programm. Für Nummer kann ein Wert von Ø -9 eingegeben werden. Wird nichts definiert, gilt die Nummer Ø.

Maximal können 1Ø Unterprogramme aufgerufen werden.

BEISPIEL:

```
100 DEF USR Ø=&HAEC
110 DEF USR 1=&HA4Ø
18Ø PRINT USR1(24ØØ)
22Ø PRINT USRØ(Ø)
```

DELETE

FORMAT: DELETE [⟨Zeilennummer⟩][-[⟨Zeilennummer⟩]]

FUNKTION: Löschen von Programmzeilen

BEMERKUNG: DELETE löscht Programmzeilen spezifiziert.

Von Start Programmzeile bis Ende Programmzeile werden alle dazwischen liegenden Statements gelöscht.

Nach Ausführung des Befehls erfolgt immer die Rückkehr in den Kommando-Modus von BASIC.

Eine Eingabe ohne Zeilenummer löst FC-Error (Illegal Function Call) aus.

BEISPIEL:

```
P3:          59 Bytes
DELETE 20
DELETE 50-80
>
```

DIM

FORMAT: DIM ⟨Variable⟩ (Index)[,⟨Variable⟩ (Index),...]

FUNKTION: Dient zum Dimensionieren eines Feldes.

BEMERKUNG: Alle dimensionierten Felder werden einem Variablennamen zugeordnet.

Die einzelnen Elelmente unterscheiden sich nur durch den beigefügten Index. Wird kein Index beigefügt, so wird er automatisch auf 1Ø gesetzt.

DIM setzt den spezifizierten Feldbereich auf Null oder auf Blank.

Die Überschreitung des Index löst ein BS-Error (Bad subscript) aus.

Jedes Feld-Element kann nach seinem Inhalt abgefragt werden.

BEISPIEL:

```
10 CLS
20 DIM A(20)
25 ERASE A
30 DIM A(20,10)
35 ERASE A
40 DIM A$(A,B)
50 FOR I=0 TO 20
70 PRINT I
80 NEXT I
90 END
```

DSKF Floppy Disk

FORMAT: DSKF ("Gerätename:")

FUNKTION: Zeigt den freien Speicherbereich der Diskette an.

BEMERKUNG: Der freie Speicherbereich des spezifischen Laufwerkes wird in KBytes angezeigt.

Vor dem Aufruf von DSKF muß das RESET-Kommando aufrufen werden.

Der Gerätename kann A, B, C, oder D sein.

BEISPIEL:

```
P1:          0 Bytes
?DSKF("A:")
 240
>
```

DSKI$ Floppy Disk

FORMAT: DSKI$ (⟨Gerätename⟩, ⟨Spur Nr.⟩, ⟨Sektor Nr.⟩)

FUNKTION: Liest Daten direkt von der Diskette.

BEMERKUNG: DSKI$ liest einen logischen Sektor der spezifizierten Spur im spezifizierten Laufwerk und gibt jeweils einen 128-Byte langen String aus.

BEISPIEL:

```
P1:          0 Bytes
DSKI$"A:",8,10,A$

>
```

DSKO$ Floppy Disk

FORMAT: DSKO$ ⟨Gerätename⟩, ⟨Spur Nr.⟩, ⟨Sektor Nr.⟩, ⟨String Ausdruck⟩

FUNKTION: Schreibt Daten direkt auf die Diskette.

BEMERKUNG: DSKO$ ist ein Disketten-Ausgabe-Kommando, das einen String in einen spezifizierten Bereich auf die Diskette schreibt. Die Daten werden in einer Einheit von 128 Bytes in einen Satz geschrieben. Physikalisch wird er in die erste oder zweite Hälfte des Single Sektors (128 Byte) geschrieben.

Nur 128 Byte des Strings werden akzeptiert.

Der Gerätename kann A, B, C, oder D sein.

Die Spur-Nr. ist eine Zahl von 0 bis 39.

Die Sektor-Nr. ist eine Zahl von 1 bis 64.

BEISPIEL:

```
10 CLEAR 500
20 A$="123456"
30 A$=LEFT$-(A$+STRING$(
1))
40 CHR$(0)),128)
50 DSKO$"A:",8,10,A$
60 END
```

END

FORMAT: END

FUNKTION: Schließt alle Dateien und beendet das Programm.

BEMERKUNG: END beendet einen Programmablauf, schließt alle geöffneten Dateien und kehrt nach Ausführung in den Kommando-Modus von BASIC zurück. Ein END-Statement kann mehrmals im Programm vorkommen. Ein END-Statement am Programmende ist nicht zwingend erforderlich. In diesem Fall wird jedoch keine Datei geschlossen.

BEISPIEL:

```
10 FOR I=1 TO 10
20 PRINT"HALLO"
30 END
```

EOF

FORMAT: EOF (⟨Datei-Nummer⟩)

FUNKTION: Prüft das Datei-Ende.

BEMERKUNG: Die Datei-Nummer muß die Nummer sein, unter der die Datei geöffnet wurde. EOF fragt nach dem Datei-Ende der spezifizierten Datei. Bei erreichtem Ende gibt EOF den Code -1 aus, ansonsten Ø. Das Datei-Ende kann mit einer Bedingung abgefragt werden, die entweder ausgeführt wird (Code -1 wahr) oder bei ELSE fortgesetzt wird (Code Ø unwahr).

Wird die spezifizierte Datei über RS-232C angesprochen ("COMØ:), gibt EOF "-1" aus, wenn der Buffer leer ist, und "Ø" wenn der Buffer geladen ist. Die EOF-Funktion gibt für die Übertragung zur Tastatur immer "Ø" zurück.

BEISPIEL:

```
10 OPEN "I",#1,"COMØ:(28
N2B)
20 IF EOF(1) THEN GOTO 1
ØØ
30 INPUT #1,A$
40 PRINT A$
100 CLOSE
```

ERASE

FORMAT: ERASE ⟨Liste von Feldvariablen⟩

FUNKTION: Eliminiert Felder in einem Programm.

BEMERKUNG: ERASE eliminiert früher dimensionierte Felder, spezifiziert in der Liste von Feldvariablen. Nach ERASE können die Felder mit dem DIM-Statement neu dimensioniert werden. Wird ein Feld redimensioniert ohne vorausgegangen ERASE-Befehl, erfolgt ein DD-Error. (Doppelte Definition)

BEISPIEL:

```
10 D=10
20 FOR I=1 TO 10
30 A(I)=11-I
40 NEXT I
50 GOSUB 200
60 ERASE A
70 D=20
80 DIM A(20)
90 FOR I=1 TO 20
100 A(I)=I
110 NEXT I
120 GOSUB 200
130 END
200 FOR I=1 TO D
210 PRINT USING"###";A(I
);
220 NEXT I
230 RETURN
```

```
 10  9  8  7  6  5  4  3
  2  1  1  2  3  4  5  6
  7  8  9 10 11 12 13 14
 15 16 17 18 19 20
```

ERR und ERL

FORMAT: ERR ERL

FUNKTION: Anzeige Fehlercode und Zeilennummer.

BEMERKUNG: Wird eine Error-Unterroutine eingegeben, so drückt ERR den Code des Fehlers aus und ERL die Zeilennummer, in der der Fehler aufgetaucht ist. Wird die Fehlerroutine im Direktmodus eingegeben, hat ERL den Wert 65 535.

BEISPIEL:

```
10 ON ERROR GOTO 80
20 CLS
30 INPUT "NR.VON 1 BIS 9
":A
40 IF(A<1)OR (A>9) THEN
ERROR 200
50 PRINT A
60 ERROR 210
70 END
80 IF ERR=200 THEN PRINT
 "WIEDERHOLEN":RESUME 30
90 IF ERL=60 THEN PRINT"
TEST":RESUME 70


NR.VON 1 BIS 9? 80
WIEDERHOLEN
NR.VON 1 BIS 9? 3
_3
```

ERROR

FORMAT: ERROR ⟨Ganzzahlwert⟩

FUNKTION: Fehler lassen sich simulieren.

BEMERKUNG: Ein ERROR-Befehl ohne vorausgegangenem ON ERROR GOTO setzt den Ganzzahlwert in die Fehlermeldung um und wird angezeigt. Das Programm wird gestoppt. Der Ganzzahlwert muß im Bereich von 1 bis 255 liegen. Ein Error-Code der nicht spezifiziert ist, wird als UP-Error angezeigt. In allen Fällen wird die Zeilennummer ausgewiesen.

BEISPIEL:

```
120 ERROR 2
RUN
SN Error In 120
>
```

EXEC

FORMAT: EXEC [⟨Startadresse⟩]

FUNKTION: Startpunkt für ein Assembler-Unterprogramm.

BEMERKUNG: Mit EXEC wird von einem BASIC-Programm in ein Assembler-Unterprogramm zu der angegebenen Startadresse gesprungen. Das Maschinenprogramm muß im Speicher geladen sein. Wird keine Startadresse angegeben, ist die Startadresse des LOADM-Kommandos oder ein vorausgegangener EXEC Befehl maßgebend.

Das BASIC-Programm ist so lange inaktiv, bis das Assembler-Programm ausgeführt ist. Der RAM-FILE-Bereich und das BASIC-Programm sind währenddessen nicht geschützt. Nach Ausführung des Maschinen-Programms wird das BASIC-Programm nach dem EXEC-Statement fortgesetzt.

BEISPIEL

```
P3:          0 Bytes
NEW
250 EXEC &H0C00
_
```

EXP

FORMAT: EXP (⟨X⟩)

FUNKTION: Gibt "E"(2.71828) hoch X an.

BEMERKUNG: Das Argument "X" muß das Resultat einer Exponentialfunktion sein. Ist "X" größer als 88.Ø2969, wird ein OV-Error (Overflow) angezeigt.

BEISPIEL:

```
10 N=4
20 FOR I=1 TO N
30 READ A(I)
40 A=A(I)
50 PRINT EXP(A)
60 NEXT I
70 DATA 3,62.487,42,58.6
412
80 END
```

```
 20.0855
 1.37328E+27
 1.73927E+18
_2.93459E+25
```

FIELD

Floppy Disk

FORMAT: FIELD [#] ⟨Datei Nr.⟩, ⟨Feldlänge⟩. AS ⟨String⟩

FUNKTION: Definiert ein Feld in einer Random Datei.

BEMERKUNG: Bevor Daten in eine Random-Datei geschrieben oder gelesen werden können, muß ein FIELD-Statement vorausgehen. Datei Nr. ist die Nummer, unter der die Datei geöffnet wurde, Feldlänge ist die Länge der Stringvariablen. Die Gesamtlänge darf nicht mehr als 128 Bytes betragen.

BEISPIEL:

```
10 OPEN "R",#1,TEST.DAT"
20 FIELD #1,8 AS A$,6 AS
 B$
30 RSET A$="EP"
40 LSET B$="SON"
50 PRINT A$+B$
60 PUT #1,1
70 CLOSE
80 END
```

FILES

FORMAT: FILES ["⟨Gerätename⟩"]

FUNKTION: Zeigt alle Dateinamen des betreffenden Gerätes an.

BEMERKUNG: Auf dem Display werden alle Dateien des angegebenen Gerätes nach folgenden Kriterien angezeigt: Dateiname, Dateityp, Klassifikation und das Aufzeichnungsformat.

Als Klassifikation wird auf dem Display nur eine Kennziffer mit folgender Bedeutung angezeigt:

Ø = BASIC Programm
1 = Daten
2 = Assembler Programm

Beim Aufzeichnungsformat gilt:

A = ASCII
Format B = Binärformat

Wird kein Gerätename angegeben, werden die Dateien des angeschlossenen Gerätes angezeigt. Die Mikro-Kassette wird bis zum Bandende gelesen, mit der BREAK-Taste kann dieser Vorgang unterbrochen werden.

BEISPIEL:

```
FILES"CAS0:"
FILES"A:"

>
```

FILNUM

Floppy Disk

FORMAT: FILNUM 〈 Nr. des FCB 〉

FUNKTION: Spezifiziert die Nummer der Datei-Kontroll-Blocks (FCB) für eine Disketten-Datei.

BEMERKUNG: FILNUM spezifiziert die Datei-Nummern, die simultan während eines BASIC-Programms geöffnet werden können. Ist eine Disketten-Datei geöffnet, kann jeder FCB 143 Bytes lang sein.

FILNUM reserviert im voraus die Bereichsgröße der FCBs. Die Nummer der FCB kann 1 bis 15 sein.

Wenn FILNUM ausgeführt ist, sind alle Dateien geöffnet.

BEISPIEL:

```
STAT
P1:              0 Bytes
FILNUM 1
_
```

FIX

FORMAT: FIX (⟨X⟩)

FUNKTION: Gibt die Ganzzahl von X wieder.

BEMERKUNG: FIX gibt die Ganzzahl von X bei einem positiven und negativen Wert mit dem gleichen Wert wieder. Der Unterschied zu INT liegt darin, daß FIX nicht die nächst niedrigere Zahl bei einem negativen X -Wert wiedergibt.

BEISPIEL:

```
10 FOR I=1 TO 5
20 READ A(I)
30 A=A(I)
40 PRINT FIX(A);
50 NEXT I
60 DATA 1.23,4.5,0,-6.6,
-9.3
70 END


RUN
 1  4  0 -6 -9
>
```

FOR...TO...STEP – NEXT

FORMAT: FOR ⟨Variable⟩=⟨X⟩ TO ⟨Y⟩[STEP ⟨Z⟩]

FUNKTION: Wiederholung der Anweisungen in einer Schleife.

BEMERKUNG: Variable "X" ist ein Zähler, dem ein bestimmter Grundwert zugewiesen wird. "Y" ist der Endwert, den dieser Zähler erreichen soll. STEP "Z" ist die Schrittgröße, die bis zum Endwert hochzählt. Ist keine Schrittgröße angegeben, wird automatisch 1 angenommen. Die Schleife wird mit der NEXT-Anweisung geschlossen. Ist der Endwert erreicht, wird das Programm nach NEXT fortgesetzt. Die Argumente X, Y, und Z können sich auch aus Berechnungen im Programm ergeben und lassen sich als Variable darstellen. Der Endwert "Y" kann auch negativ sein, in diesem Fall wird rückwärts gezählt.

Programmierte Schleifen ermöglichen mehrmals die Wiederholung eines Programmabschnittes zwischen FOR und NEXT. Programm-Schleifen können ineinander geschachtelt werden, nur ist darauf zu achten, daß immer zuerst die innere Schleife mit NEXT abgeschlossen werden muß.

```
10 FOR T=1 TO 1000
20 T=T+1
30 NEXT T
```

```
10 FOR I=1 TO 10
20 FOR K=1 TO 20
30 NEXT K
40 NEXT I
```

```
10 FOR I=1 TO 10
20 FOR J=1 TO 20
30 NEXT I
40 NEXT J
```

FRE

FORMAT: FRE(⟨Ausdruck⟩)

FUNKTION: Gibt die freien Bytes aus.

BEMERKUNG: Ist der Ausdruck numerisch, gibt FRE die Anzahl der freien Bytes im BASIC-Textbereich zurück.
Ist der Ausdruck ein String, gibt FRE die freien Bytes im BASIC String-Bereich zurück.

BEISPIEL:

```
P2:          0 Bytes
PRINT FRE(0)
 13711
>


NEW
PRINT FRE("A$")
 200
>
```

FRMAT Floppy Disk

FORMAT: FRMAT [⟨ Gerätename ⟩]

FUNKTION: Formatiert eine Diskette.

BEMERKUNG: Mit FRMAT wird eine Diskette in dem angegebenen Laufwerk formatiert.

Der Gerätename kann A, B, C, oder D sein. Fehlt der Gerätename, wird B angenommen. Mit der Frage: "Are you sure?" läßt sich das Kommando nochmals bestätigen. Wird "Y" eingegeben, beginnt der Formatierungsvorgang, mit "N" kann das Kommando wieder gewechselt werden. Es ist möglich, gebrauchte Disketten nochmals zu formatieren.

BEISPIEL:

```
NEW
STAT
P3:          0 Bytes
FRMAT"A:"  _
```

GCLS

FORMAT: GCLS

FUNKTION: Löscht den Graphik-Schirm.

BEMERKUNG: GCLS löscht nur die Graphik auf dem L.C.D. oder auf dem Display. Wird der Text-Schirm zusammen mit dem Graphik-Schirm benutzt, wird nur der Graphik-Schirm gelöscht.

BEISPIEL:

```
10 CLS
20 PI=3.141592
30 DX=PI/64
40 LINE(60,31)-(60,0),PS
ET
45 GCLS
50 PRINT"HALLO"
60 END
```

GET%

FORMAT: GET% ⟨Satznummer⟩, ⟨Variablen Name⟩ [,⟨Variablen Name⟩...]

FUNKTION: Liest Datensätze in den RAM-FILE-Bereich.

BEMERKUNG: Das GET%-Kommando liest Daten mit der angegebenen Satznummer in die Variablen, spezifiziert durch die Liste der Variablennamen.

Bevor ein GET% Kommando angesprochen werden kann, muß der Satz mittels DEFFIL definiert worden sein.

Jeder mit GET% angesprochene Satz muß mit PUT% in der Speicherdatei abgelegt sein.

Satznummer	Nummer von Ø-255
Variable	Jede in BASIC erlaubte Variable. Gesamt maximal 255 Zeichen.

BEISPIEL:

```
10 CLEAR 100,1000
20 DEFFIL 20,100
30 INPUT"NAME1:",A$
40 PUT%1,A$
50 GET%1,D$
60 PRINT D$
70 END
```

GET FLoppy Disk

Format: GET (#) Dateinummer (Satznummer)

Funktion: Lesen einer Aufzeichnung von einer Diskettendatei mit wahlfreiem Zugriff in einen Puffer mit wahlfreiem Zugriff.

Bemerkungen: (Dateinummer) ist die Nummer, unter der die Datei eröffnet wurde. Wenn die Aufzeichnungsnummer weggelassen wird, wird die nächste Aufzeichnung (nach dem letzten GET-Befehl) in den Puffer gelesen. Die größtmögliche Aufzeichnung ist 32767.

```
10 OPEN"R",#1,"FILE"
20 FIELD #1,20 AS N$,4 A
S A$,8 AS P$
30 INPUT"DIGIT";CODE%
40 IF CODE% ="0" THEN GO
TO 90
50 GET #1,CODE%
60 PRINT N$
70 PRINT USING"$$###.##"
;CVS(A$)
80 PRINT P$;PRINT
90 GOTO 30
100 CLOSE
110 END
```

GOSUB...RETURN

FORMAT: GOSUB ⟨Zeilennummer⟩
RETURN

FUNKTION: Sprung in ein Unterprogramm und zurück.

BEMERKUNG: GOSUB springt in ein BASIC-Unterprogramm und kehrt mit RETURN zurück. Die Zeilennummer ist die erste Zeile des Unterprogramms.

Ein Unterprogramm ist eigentlich ein selbstständiges Programm, das aber so geschrieben ist, daß es von einem anderen Programm aufgerufen werden muß, und nur dann ausgeführt wird.

Im BASIC kann eine beliebige Zahl von Unterprogrammen aufgerufen werden, die ihrerseits eine beliebige Zahl von Anweisungen haben, jedoch muß der letzte Befehl eines Unterprogramms die RETURN-Anweisung sein.

Unterprogramme können ebenfalls Unterprogramme aufrufen, d.h. sie können geschachtelt sein; es muß jedoch immer wieder zum aufrufenden Programm zurückgekehrt werden.

BEISPIEL:

```
10 INPUT B$
20 GOSUB 400
30 PRINT B$
40 GOTO 10
400 L=LEN(B$)
410 IF L=20 THEN B$=LEFT
$(B$,20)
420 NU$="......
430 REST=20-L
440 B$=LEFT$(B$,L)+RIGHT
$(NU$,REST)
450 RETURN
```

GO TO/GOTO

FORMAT: GO TO ⟨Zeilennummer⟩, oder GOTO ⟨Zeilennummer⟩

FUNKTION: Sprung zu der angegebenen Zeilennummer.

BEMERKUNG: Mit einer GOTO-Anweisung erfolgt ein unbedingter Sprung zu der angegebenen Zeilennummer. Es ist kein Rücksprung erforderlich.

GO TO und GOTO haben dieselbe Funktion.

BEISPIEL:

```
10 CLS
20 INPUT "TEXT: ",A$
30 GOTO 10
40 END
```

HEX$

FORMAT: HEX$ (⟨X⟩)

FUNKTION: Gibt den hexadezimalen Wert von X wieder.

BEMERKUNG: HEX$ konvertiert den Ausdruck X zu einem hexadezimalen Wert.

X muß im Bereich zwischen -32 768 und 65 535 liegen. Vor der Umwandlung in HEX$ wird X auf- oder abgerundet.

BEISPIEL:

```
10 PRINT "DEZ  HEX"
20 FOR I = 7 TO 16
30 PRINT USING"###";I;
40 PRINT USING"&    &";""
,HEX$(I)
50 NEXT I
60 END
  7   7
  8   8
  9   9
 10   A
 11   B
 12   C
 13   D
 14   E
 15   F
 16   10
```

IF...THEN...ELSE/IF...GOTO...ELSE

FORMAT:

IF ⟨Ausdruck⟩
[THEN [⟨Statement⟩ / ⟨Zeilennummer⟩] / GOTO ⟨Zeilennummer⟩] [ELSE [⟨Statement⟩ / ⟨Zeilennummer⟩]]

FUNKTION:

Der weitere Programmablauf wird von dem Resultat des Ausdrucks abhängig gemacht.

BEMERKUNG:

IF setzt die Programmausführung abhängig von der Bedingung des Ausdrucks fort.

Wenn das Resultat nicht Null ist, (wahr), wird der THEN oder GOTO Befehl ausgeführt.

THEN kann eine Zeilennummer oder ein oder mehrere Statement(s) sein.

Nach GOTO muß immer eine Zeilennummer folgen.

Ist das Resultat Null (unwahr), folgt die ELSE (ansonsten) Ausführung, falls angegeben.

Andernfalls fährt das Programm mit dem nächsten Statement fort.

Der Ausdruck kann eine Zeichenkette, eine Variable, ein arithmetischer Ausdruck, eine Konstante oder ein String sein.

BEISPIEL:

```
100 IF X$="ABC" THEN 220

100 IF X=1 THEN 150

100 IF X$0Y$ THEN U$=X$+
"XYZ":GOTO 220

100 IF A>10 THEN A=0 ELS
E 100
```

INKEY$

FORMAT: INKEY$

FUNKTION: Fragt ein Zeichen von der Tastatur ab.

BEMERKUNG: INKEY$ erlaubt nur die Eingabe eines Zeichens über die Tastatur. INKEY$ liest das Zeichen im Tastatur-Puffer und gibt es über ein Zeichen-String aus. Alle Zeichen, die nicht in der "Zeichen Code"-Tabelle" enthalten sind, werden ignoriert.

BEISPIEL:

```
10 A$=INKEY$
20 INPUT"EINGABE:",A$
30 IF A$="M" THEN 40 ELS
E 10
40 PRINT "OK"
50 END
```

```
10 A$=INKEY$
20 IF A$=""THEN 30 ELSE
10
30 PRINT"FERTIG"
40 END
```

INPUT

FORMAT: INPUT ["⟨TEXT⟩"], [⟨Liste von Variablen⟩]

FUNKTION: Eingabe über die Tastatur während des Programmablaufs.

BEMERKUNG: INPUT unterbricht den Programmablauf und wartet auf die Eingabe über die Tastatur.

TEXT, vor den Variablen geschrieben, kann eine Bemerkung sein, um dem Anwender die Eingabe zu verdeutlichen.

Fällt dieser Text weg, erscheint als Eingabeaufforderung ein Fragezeichen ?.

Bei der Eingabe ist darauf zu achten, daß kein Komma enthalten ist, ansonsten erscheint auf dem L.C.D. "? Redo", und die Eingabe muß wiederholt werden.

Wird eine Variablenliste gesetzt, so sind die einzelnen Variablen durch Komma zu trennen.

Bei einer Zeichenkette ist darauf zu achten, daß im Text kein Komma vorkommt, da ansonsten der nachfolgende Text nicht mehr berücksichtigt wird.

BEISPIEL:

```
10 INPUT"WERT: ",A
20 INPUT"TEXT: ",A$
30 INPUT"WERTE:",A,B,C

WERT: 2305
TEXT: GUTEN TAG
WERTE:25,75,150
>
```

INPUT#

FORMAT: INPUT # ⟨Dateinummer⟩, [⟨Variablenliste⟩]

FUNKTION: Liest Daten von einer geöffneten Datei und überträgt sie in die Variablen.

BEMERKUNG: Die Dateinummer ist die Nummer, unter der die Datei geöffnet wurde.

Die Variablenliste sind die in der Datei befindlichen Variablen, die in das Programm eingelesen werden sollen. Sie werden der Reihe nach so gelesen, wie sie eingegeben wurden. Numerische Werte, Leerzeichen, Returns und Zeilenvorschübe werden ignoriert.

Sind alle Daten eingelesen, und es erfolgt ein INPUT# Befehl, erscheint ein IE Error (Input past End).

BEISPIEL:

```
10 OPEN"I",#1,"COM0:(48N
2B)
20 INPUT #1,A$
30 PRINT A$
40 GOTO 20
50 CLOSE
60 END
```

INPUT$

FORMAT: INPUT$ (⟨X⟩ [,[#] ⟨Dateinummer⟩])

FUNKTION: Liest einen String von X Zeichen von der Tastatur oder aus der angegebenen Datei.

BEMERKUNG: INPUT$ liest einen String von Zeichen in der Anzahl X aus der spezifizierten Datei.

Wird keine Dateinummer angegeben, können die Zeichen in der Länge X über die Tastatur eingegeben werden. INPUT$ wartet bis der String mit der angegebenen Zahl X übereinstimmt.

INPUT$ liest alle Zeichen, mit Ausnahme von BREAK und Control "C".

BEISPPEL:

```
10 REM INHALT EINER DATE
I IN HEX
20 OPEN"I",1,"DATA"
30 IF EOF(1) THEN 60
40 PRINT HEX$(ASC(INPUT$
(1,#1)));
50 GOTO 30
60 PRINT
70 END
```

INSTR

FORMAT: INSTR ([⟨I⟩,] ⟨String 1⟩, ⟨String 2⟩)

FUNKTION: Gibt die Position eines Strings in einem anderen String an.

BEMERKUNG: INSTR sucht den String 2 im String 1 und gibt die Position des Ereignisses wieder. Wird er nicht gefunden, ist die Position Ø.

Mit I kann die Position gesetzt werden, ab der gesucht werden soll. I muß im Bereich von Ø-255 liegen.

Ist I nicht gesetzt, wird ab Beginn von String 1 gesucht. Wenn String 2 Null ist, gibt INSTR den gleichen Wert von I zurück.

BEISPIEL:

```
10 X$="ABCDEB"
20 Y$="B"
30 LPRINT INSTR(X$,Y$),I
NSTR(4,X$,Y$)
40 END
 2             6
```

INT

FORMAT: INT (⟨X⟩)

FUNKTION: Gibt den größten Integerwert von X wieder.

BEMERKUNG: INT gibt den größten Integerwert, der gleich oder kleiner als X ist, wieder.

BEISPIEL:

```
10 FOR I=1 TO 5
20 LPRINT INT(RND*100);
30 NEXT
40 END
```

```
 59  20  55  63  10
```

```
10 FOR I=1 TO 6
20 READ A(I)
30 PRINT INT(A(I));
40 NEXT I
50 DATA 3,-8,4.7,7.2,-1.
8,-6.1
60 END
```

```
 3 -8  4  7 -2 -7
```

KEY

FORMAT: KEY ⟨Key Nummer⟩, ⟨String⟩

FUNKTION: Definiert die Funktionstasten.

BEMERKUNG: Die Funktionstasten beim HX-20 sind frei belegbar. Zusammen mit der SHIFT-Taste stehen 10 Funktionstasten zur freien Verfügung.

Sie können als Programmierhilfe oder als Programmstart dienen.

Die Länge des Strings kann maximal 15 Zeichen sein.

BEISPIEL:

```
P1:                61 Bytes
KEY 1,"STAT"
KEY 2,"RUN"+CHR$(13)
>
```

KEY LIST/KEY LLIST

FORMAT: KEY LIST/KEY LLIST

FUNKTION: Anzeige oder Ausdruck der Funktionstasten.

BEMERKUNG: KEY LIST zeigt die Belegung der einzelnen Funktionstasten auf dem Display an und zwar die Tastennummer mit dem dazugehörenden Text.

Das ^M Zeichen bedeutet, daß das Programm sofort gestartet wird.

KEY LLIST druckt die Belegung der Funktionstasten auf dem eingebauten Minidrucker aus.

BEISPIEL:

```
PF1  AUTO
PF2  LIST^M
PF3  LLIST^M
PF4  STAT
PF5  RUN^M
PF6  ?DATE$:?TIME$^M
PF7  LOAD
PF8  SAVE
PF9  TITLE
PF10 LOGIN
```

KILL Floppy-Disk

FORMAT: KILL 〈 Dateiname 〉

FUNKTION: Löscht eine Datei oder Programm von der Diskette.

BEMERKUNG: Der Datei-oder Programmname muß auf der Diskette vorhanden sein.

KILL sollte nur bei geschlossenen Dateien angewandt werden.

KILL kann für ein BASIC-Programm, Daten oder ein Maschinensprache-Programm zum Löschen benutzt werden.

BEISPIEL:

```
STAT
P1:          0 Bytes
KILL"PROG1"
>
```

LEFT$

FORMAT: LEFT$ (⟨ String⟩, ⟨ I ⟩)

FUNKTION: Zeigt von links beginnend einen String an.

BEMERKUNG: LEFT$ gibt von links beginnend eine Zeichenreihe in der Länge von I aus. I darf nur im Bereich von Ø-255 liegen. Ist I größer als die Zeichen im String, wird nur die tatsächliche Länge des Strings angezeigt.

Wird I nicht angegeben, gibt LEFT$ keine Zeichen aus.

BEISPIEL:

```
10 A$="HX-20"
20 B$=LEFT$(A$,2)
30 LPRINT B$
40 END
```

```
HX
```

LEN

FORMAT: LEN (⟨String⟩)

FUNKTION: Gibt die Länge des Strings aus.

BEMERKKNG: LEN gibt die gesamte Länge des Strings aus. Auch Steuerzeichen, Controlzeichen, Blanks, die nicht ausgedruckt werden können, werden berücksichtigt.

BEISPIEL:

```
10 A$="EPSON"
20 LPRINT LEN(A$)
30 END
 5
```

LET

FORMAT: [LET] ⟨Variable⟩ = ⟨Ausdruck⟩

FUNKTION: Der Variablen kann ein Wert zugeordnet werden.

BEMERKUNG: Diese Anweisung dient dazu, einer Variablen einen bestimmten Wert zuzuweisen.

Der Befehl ist eine Option, der nicht verwendet werden muß, oftmals aber zur Übersichtlichkeit eines Programmlistings beiträgt.

BEISPIEL:

```
10 LET MWST=13
ODER
10 MWST=13
>
```

LINE

FORMAT: LINE [(⟨X1,Y1⟩)]-(⟨ X2,Y2 ⟩), |PSET / PRESET| [,⟨Farbe⟩]

FUNKTION: Zieht eine Linie zwischen zwei Punkten.

BEMERKUNG: Mit diesem Kommando wird eine Linie von Punkt (X1,Y1) zu Punkt (X2,Y2) auf dem Graphik-Schirm gezogen.

Auf dem externen Bildschirm kann der Graphik-Schirm farbig dargestellt werden.

Mit SCREEN wird der Graphik-Schirm auf dem externen Display angesprochen.

Der Ausdruck PSET zeichnet die Linie.

Mit PRESET kann diese Linie durch Benutzung der Vordergrundfarbe wieder gelöscht werden.

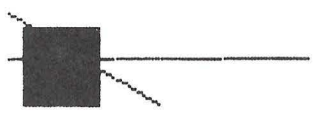
Die Punktauflösung auf dem L.C.D.ist folgende:

X = Ø bis 119
Y = Ø bis 31

Mit dem COPY-Befehl können die Bildschirminhalte auf dem eingebauten Minidrucker ausgedruckt werden.

BEISPIEL:

```
10 CLS
20 FOR I=5 TO 30
30 LINE(5,I)-(30,I),PSET
40 NEXT I
50 LINE(0,0)-(50,30),PSE
T
60 LINE(0,15)-(100,15),P
SET
70 COPY
80 END
```



LINE INPUT

FORMAT: LINE INPUT ["⟨Text ⟩";] ⟨Stringvariable⟩

FUNKTION: Eingabe einer Zeile in eine Stringvariable.

BEMERKUNG: LINE INPUT gestattet die Eingabe bis zu 255 Zeichen über die Tastatur ohne irgendwelche Beschränkungen. D.h. Kommas, Doppelpunkte, Semikolon etc. sind erlaubt. Die eingegebenen Zeichen werden als Stringvariable dargestellt.

TEXT dient als Hinweis für den Anwender.

Jede Eingabe wird mit der RETURN-Taste beendet.

LINE INPUT erlaubt eine Unterbrechung der Eingabe mit der BREAK-Taste.

BASIC geht in den Kommando-Modus zurück.

Mit CONT kann das Programm fortgesetzt werden.

BEISPIEL:

```
10 LINE INPUT"ZEICHEN IN
PUT: ";A$
20 LPRINT A$
30 END

"EPSON HX-20"
```

LINE INPUT#

FORMAT: LINE INPUT# ⟨Dateinummer⟩, ⟨Stringvariable⟩

FUNKTION: Liest eine Zeile von einer Datei und überträgt sie in eine Stringvariable.

BEMERKUNG: LINE INPUT# überträgt eine Zeile ohne Abgrenzung (max. 255 Zeichen) von einer sequentiellen Datei in eine Stringvariable.

Die Dateinummer ist die Nummer, unter der die Datei mit OPEN geöffnet wurde.

Die Stringvariable ist der Variablenname, in den die Zeile übertragen wird.

BEISPIEL:

```
10 OPEN"O",#1,"DATEN"
20 LINE INPUT;A$
30 PRINT #1,A$
40 CLOSE #1
45 WIND
50 OPEN"I",#1,"DATEN"
60 LINE INPUT #1,A$
70 CLOSE #1
80 PRINT A$
90 END
```

LIST/LLIST

FORMAT: LIST [⟨von Zeilennummer⟩] [[⟨ bis Zeilennummer⟩]]
LLIST [⟨von Zeilennummer⟩] [-[⟨ bis Zeilennummer⟩]]

FUNKTION: Ausgabe eines Programmlistings.

BEMERKUNG: LIST zeigt die Programmzeilen des Speicherbereiches an, der mit LOGIN angewählt wurde. LIST ohne weitere Angabe listet das komplette Programmm aus. LIST mit Angabe der Startzeilennummer zeigt nur diese Zeile an. LIST mit Angabe der Startzeilennummer und der Endzeilennummer listet das Programm von und bis zu diesen Nummern aus.

Bei Eingabe der – von bis – Zeilennummer darauf achten, daß sie in aufsteigender Reihenfolge erfolgt.

Das Listen kann mit der Pause-Taste gestoppt werden und mit der RETURN-Taste wieder fortgesetzt werden.

Mit der BREAK-Taste erfolgt ein Abbruch.

LLIST gibt das Programmlisting auf dem eingebauten Minidrucker aus. Die Pause-Taste bleibt beim Drucken ohne Funktion.

Ansonsten ist LLIST identisch mit LIST.

BEISPIEL:

`LIST`	Gesamtprogramm Ausdruck
`LIST 100`	Ausdruck Zeile 100
`LIST 100-200`	Ausdruck Zeilen 100-200
`LIST -100`	Ausdruck Zeile 0 bis 100

LIST/SAVE

FORMAT: LIST ⟨String⟩

FUNKTION: Ausgabe eines Programms auf Kassette.

BEMERKUNG: Dieses LIST-Kommando hat mit der eingebauten Kassette dieselbe Funktion wie das SAVE-Kommando.

Die Aufzeichnung des Strings erfolgt im ASCII-Format.

LIST, mit String definiert, muß immer in Gänsefüßchen stehen, gefolgt von einem Komma.

BEISPIEL:

```
P1:              0 Bytes
LIST",A$
>
```

LIST"COM0:"

FORMAT: LIST "COM0: [(⟨BLPSC⟩)]" [, [⟨Zeilennummer⟩]]–[⟨Zeilennummer⟩]]]

FUNKTION: Programmlisting über RS-232C.

BEMERKUNG: LIST "COM0: hat diesselbe Funktion wie LIST, nur ist die RS-232C Schnittstelle angeschlossen.

Die Details über die BLPSC-Spezifikation sind unter OPEN COM0:" näher erläutert.

BEISPIEL:

`LIST"COM0:(48N22)`	komplettes Listing
`LIST"COM0:(48N2F)",80`	List Zeile 80
`LIST"COM0:(48N2F)",50-90`	List Zeile 50 bis 90

LOAD

FORMAT: LOAD [⟨Programmname⟩ [,R]]

FUNKTION: Lädt ein Programm oder eine Datei in den Speicher.

BEMERKUNG: Mit dem LOAD-Kommando wird ein Programm oder eine Datei in den dafür vorgesehenen Speicher geladen.

Der jeweilige Speicherbereich wird mit LOGIN n vorbesetzt. Mit STAT werden die Speicherbereiche angezeigt.

Wird kein Programmname (Dateibezeichnung) angegeben, wird das erste aufgefundene Programm geladen.

LOAD mit Angabe eines Programmnamens liest so lange, bis der spezifizierte Name gefunden wurde, danach wird das entsprechende Programm geladen.

LOAD schließt alle geöffneten Dateien und löscht alle Variablen.

Mit der Option R werden keine Dateien geschlossen, das Programm wird sofort nach dem Ladevorgang gestartet.

```
>
P1:          0 Bytes
LOAD"CAS0:PROG1",R
```

LOAD "COM0:"

FORMAT: LOAD "COM0: [(⟨BLPSC⟩)]"

FUNKTION: Lädt über das RS-232 Interface.

BEMERKUNG: Mit diesem LOAD-Kommando können Programme oder Daten über das RS-232-Interface geladen werden. Die Programme müssen im ASCII-Format (SAVE,A) abgespeichert sein.

Die (BLPSC) Spezifikation steht detailliert unter OPEN "COM0:".

BEISPIEL:

```
NEW
STAT
P1:           0 Bytes
LOAD"COM0:(68N2F)",R
```

LOADM

FORMAT: LOADM [⟨Programmname⟩][, [Startadresse][,R]]

FUNKTION: Laden eines Programms in Maschinensprache.

BEMERKUNG: Ein Maschinenprogramm, das über den Monitor erstellt wurde, wird mit LOADM geladen.

Wird kein Programmname angegeben, so wird das erste aufgefundene Programm eingeladen.

Der Ladevorgang beginnt mit der Startadresse. Mit der Option R wird nach dem Laden das Programm sofort gestartet. Ist R nicht angegeben, wird nach dem Ladevorgang in den BASIC-Kommando-Modus zurückgekehrt. Mit EXEC und der Startadresse kann in die Programmausführung gesprungen werden.

BEISPIEL:

```
P1:          0 Bytes
LOADM"CAS0:TEST"
LOADM"CAS0:TEST",R
_
```

LOAD?

FORMAT: LOAD? [⟨Programmname⟩]

FUNKTION: Prüft die aufgezeichneten Programme.

BEMERKUNG: LOAD? prüft, ob das abgespeicherte Programm oder Daten korrekt aufgezeichnet wurden. Das im Speicher stehende Programm wird dabei nicht berücksichtigt und nicht gelöscht.

BEISPIEL:

```
STAT
P1:            0 Bytes
LOAD?"CAS0:PROG1"
_
```

LOC

Floppy Disk

FORMAT: LOC [(Datei Nummer)]

FUNKTION: Gibt die laufende Satznummer wieder.

BEMERKUNG: Wurde die spezifizierte Datei im "R"-Modus geöffnet, wird der zuletzt mit GET oder PUT benutzte Satz wiedergegeben. Bei einer im "I"-Modus geöffneten Datei wird die Nummer des logischen Sektors gelesen, sobald er ausgegeben werden kann. Ein OPEN im "O"-Modus schreibt die Nummer des Sektors sofort nach der Ausführung zurück.

BEISPIEL:

```
200 IF LOC(1) > 50 THEN
STOP
```

LOCATE

FORMAT: LOCATE ⟨X⟩, ⟨Y⟩, [,⟨Cursor Schalter⟩]

FUNKTION: Setzt den Cursor auf den Bildschirm.

BEMERKUNG: LOCATE positioniert den Cursor an den bestimmten Punkt auf dem virtuellen Schirm.

Es muß X und Y definiert sein. X gilt für die horizontale Koordinate, Y für die vertikale Koordinate.

Der Cursor kann nicht außerhalb des virtuellen Schirms, dessen Grenzen mit WIDTH festgelegt werden, positioniert werden.

Der Cursor-Schalter ist nicht aktiviert mit "0", und aktiviert mit "1".

Vorbesetzt ist 0.

BEISPIEL:

```
10 CLS
20 A$="ABCDE"
30 LOCATE 5,2
40 PRINT A$
50 LOCATE 10,3
60 PRINT A$
70 LOCATE 15,4
80 PRINT A$
90 END

     ABCDE
          ABCDE
               ABCDE
>
```

LOCATES

FORMAT: LOCATES ⟨X⟩, ⟨Y⟩

FUNKTION: Setzt den Cursor im physikalischen Schirm.

BEMERKUNG: LOCATES bewegt den physikalischen Schirm mit den spezifizierten Koordinaten X und Y für die linke, obere Ecke im virtuellen Schirm. X ist für horizontal, Y für vertikal.

Der physikalische Schirm kann die Abgrenzung des virtuellen Schirms nicht verlassen.

Die Standard-Position des Cursors ist 0,0.

Nach dem LOCATE-Befehl kann mit LOCATES diese Position wieder erreicht werden.

BEISPIEL:

```
10 CLS
20 A$="HX-20"
30 LOCATE 3,1
40 PRINT A$
50 LOCATES 0,0
60 PRINT A$
70 END
```

LOF

FORMAT: LOF (⟨Dateinummer⟩)

FUNKTION: Gibt die Größe der spezifizierten Datei wieder.

BEMERKUNG: Die spezifizierte Dateinummer muß mit OPEN "Input" geöffnet sein. Befindet sich diese Datei in der ROM-Cartridge, gibt LOF die verbleibende Länge der Datei in Bytes wieder. Ist die spezifizierte Datei über RS-232C definiert, gibt LOF die Anzahl Bytes wieder, die im Buffer gespeichert sind.

BEISPIEL:

```
10 OPEN"O",#1,"COM0:(48N
2B)"
20 INPUT A$
30 PRINT #1,A$
40 IF LOF(1)=0 THEN 40
50 GOTO 20
60 CLOSE
70 END
```

LOG

FORMAT: LOG (⟨X⟩)

FUNKTION: Gibt den natürlichen Logarithmus von X wieder.

BEMERKUNG: LOG gibt den natürlichen Logarithmus von X wieder.

X muß größer als Null sein.

BEISPIEL:

```
10 A=2:B=15:C=21.8
20 LPRINT LOG(A),LOG(B),
LOG(C)
30 END
```

```
 .693147        2.70805
 3.08191
```

LOGIN

FORMAT: LOGIN ⟨Speicherbereich⟩[, R]

FUNKTION: Anwählen des entsprechenden Speicherbereichs.

BEMERKUNG: Für Applikationsprogramme in BASIC stehen dem Anwender fünf Speicherbereiche zur Verfügung, die separat gespeichert werden können.

Das LOGIN-Kommando spezifiziert den entsprechenden Programmbereich mit einer Zahl von 1-5. Die Option R bedeutet, daß das Programm sofort nach dem Sprung in den Bereich gestartet wird.

Alle Kommandos, die die Programmausführung oder die Modifikation betreffen, (NEW, LIST, LOAD, SAVE, etc.), gelten nur für diesen angewählten Bereich. Sobald ein LOGIN-Bereich verlassen wird, sind alle Variablen gelöscht.

In jedem Speicherbereich kann ein unterschiedliches Programm stehen, jedoch ist ein Austausch von Variablen nicht möglich.

BEISPIEL:

```
LOGIN2
P2:              0 Bytes
>
```

LPRINT /LPRINT USING

FORMAT: LPRINT [⟨Ausdruck⟩ ,⟨Liste von Ausdrücken⟩...]]

FUNKTION: Ausgabe von Daten auf dem eingebauten Minidrukker.

BEMERKUNG: LPRINT gibt die Daten auf dem Drucker aus.

LPRINT USING druckt die Daten formatgerecht aus. Wird eine Liste von Daten ausgedruckt, muß zwischen den Ausdrücken ein Komma, Semikolon, oder ein Blank stehen.

LPRINT und LPRINT USING sind ansonsten identisch mit PRINT und PRINT USING.

BEISPIEL:

```
10 A$="1234567"
20 B$="ABCDEFG"
30 LPRINT USING"!";A$
40 LPRINT USING"&";A$
50 LPRINT USING "ö  ö";A
$;B$
60 LPRINT USING "ö
  ö";A$;B$
70 END
```

```
1
1234567
1234ABCD
1234567   ABCDEFG
```

LSET, RSET

Floppy-Disk

FORMAT: LSET ⟨ String Variable ⟩ = ⟨ String ⟩ RSET ⟨ String Variable ⟩ = ⟨ String ⟩

FUNKTION: Speichert Daten links-oder rechtsbündig in den Random-Datei-Buffer.

BEMERKUNG: Um Daten in den Random-Datei-Buffer zu speichern, muß LSET oder RSET benutzt werden.

Ist die Länge des Strings kürzer als die Stringvariable in dem FIELD-Statement, füllt LSET linksbündig, und RSET rechtsbündig den Buffer mit Blanks auf.

Die Überlänge fällt weg.

Bevor ein numerischer Wert in den Random-Buffer abgespeichert werden kann, muß er zu einem String konvertiert worden sein.

LSET oder RSET kann für Strings benutzt werden, die keinem FIELD-Statement zugewiesen worden sind.

BEISPIEL:

```
150 LSET A$=MKS$(AMT)
160 LSET D$=DESC($)
```

```
150 LSET A$=MKS$(AMT)
160 LSET D$=DESC($)
210 A$=SPACE$(20)
220 RSET A$=N$
```

MEMSET

FORMAT: MEMSET [⟨niedrigste RAM Adresse für BASIC⟩]

FUNKTION: Speicherbereich für ein Maschinenprogramm.

BEMERKUNG: MEMSET setzt einen Bereich für ein Maschinenprogramm fest.

Dieser Bereich läßt sich von BASIC nicht überschreiben.

Standardmäßig ist der niedrigste Bereich 1 Byte höher als der Systembereich.

Nach dem Kaltstart steht die Speicheradresse auf &H0A40. Auch beim Warmstart bleibt diese Adresse erhalten.

MEMSET ohne Angabe einer Adresse setzt automatisch auf diesen Standardwert zurück.

MEMSET löscht den Variablenbereich.

BEISPIEL:

```
MEMSET &H0D00
MEMSET 4000
MEMSET
Ok
```

MERGE

FORMAT: MERGE [⟨Programmname⟩ [, R]]

FUNKTION: Mischt das spezifizierte Programm mit dem Programm im Speicher, ohne daß der momentane Speicherinhalt gelöscht wird.

Das externe Programm muß im ASCII-Code-Format zur Verfügung stehen (SAVE, A). Haben mehrere Programmzeilen die gleiche Zeilennummer, werden die des geladenen Programms berücksichtigt (höhere Priorität).

Ist kein Programmname definiert, wird das zuerst gelesene Programm geladen.

Mit der Option R wird nach dem Merge-Vorgang das Programm sofort gestartet.

BEISPIEL:

```
STAT
P1:          0 Bytes
MERGE"CAS0:PROG1"_
```

MERGE "COM0:"

FORMAT: MERGE "COM0: [(⟨BLPSC⟩)" [,R]]

FUNKTION: Führt MERGE über das Interface RS-232C aus.

BEMERKUNG: Dieses Kommando ist dasselbe wie MERGE, nur wird das zu ladende Programm über RS-232C eingelesen.

Die Details über die BLPSC Spezifikation sind unter OPEN "COM0:" näher erläutert.

BEISPIEL:

```
STAT
P1:          0 Bytes
MERGE"COM0:(68N2F),R
```

MID$

FORMAT: MID$ (⟨String⟩, ⟨I⟩, ⟨J⟩)

FUNCTION: Gibt eine beliebige Anzahl von Zeichen eines Strings wieder.

BEMERKUNG: MID$ gibt einen String in der Länge von J Zeichen, ab Position I wieder.

Der Wert von J und I muß zwischen Ø und 255 liegen.

Wird J nicht angegeben, werden von der Position I ausgehend, alle rechtsbündigen Zeichen ausgegeben. Ist der String kleiner als I, wird MID$ nicht ausgeführt.

BEISPIEL:

```
10 A$="DEUTSCHLAND"
20 M$=MID$(A$,8,4)
30 LPRINT M$
40 END
LAND
```

MKI$, MKS$, MKD$ Floppy Disk

FORMAT:

MKI$ (⟨ Ganzzahl ⟩)
MKS$ (⟨ einfache Genauigkeit ⟩)
MKD$ (⟨ doppelte Genauigkeit ⟩)

FUNKTION:

Konvertiert numerische Daten zu einem String.

BEMERKUNG:

Alle gespeicherten Daten in einem Random-Datei-Buffer müssen ein String sein.

Numerische Daten müssen zu einem String konvertiert werden.

MKI$ konvertiert einen 2-Byte String, MKS$ konvertiert einen 4-Byte String, MKD$ konvertiert einen 8-Byte String.

BEISPIEL:

```
90 AMT=(K+T)
100 FIELD #1,8 AS D$, 20
 AS N$
110 LSET D$ = MKS$(AMT)
120 LSET N$ = A$
130 PUT #1
```

MON

FORMAT: MON

FUNKTION: Sprung von einem BASIC-Programm in den System-MONITOR.

BEMERKUNG: MON verzweigt von einem BASIC-Programm in den Systemmonitor. MON wird benutzt, um die Programmkontrolle von BASIC in Maschinensprache zu übertragen. Die Monitor-Kommandos haben folgende Bedeutung:

S = Adresse setzen
D = Adresse anzeigen
X = aktuelle Register-Anzeige
B = Ende Monitor
GO= zurück zum MONITOR
R = Lädt ein Maschinenprogramm
V = Prüft ein Maschinenprogramm
W = Sichert ein Maschinenprogramm
A = Spezifiziert die Adressen für R,V,W

Automatisch angezeigt werden:

A = Register A
B = Register B
X = Index, Register X
C = Condition Code Register
S = Stack Pointer
P = Programm Counter

Der Cursor wird links oben mit einem Strich angezeigt.

Der Monitor nimmt nur ein gültiges Kommando an, andernfalls verzweigt MON automatisch in das BASIC Programm.

BEISPIEL:

```
-G A3B5 _

 A=00 B=6E   X=AB1C
 C=C8 S=5764 P=A3B5

STAT
P1:      -      0 Bytes
MON
>
```

MOTOR

FORMAT: MOTOR [⟨ON/OFF⟩]

FUNKTION: Ein- bzw. Ausschalten des Motors bei einem externen Kassettenrecorder.

BEMERKUNG: MOTOR kann bei einem an den HX-20 angeschlossenen Kassettenrecorder den Motor ein-oder ausschalten.

Ist der Remote Schalter auf OFF, kann er mit ON automatisch mit einem Ton eingeschaltet werden.

BEISPIEL:

```
P1:          0 Bytes
MOTOR ON
MOTOR OFF
>
```

NAME

Floppy Disk

FORMAT: NAME ⟨ alter Dateiname ⟩, AS ⟨ neuer Dateiname ⟩

FUNKTION: Wechselt den Dateinamen.

BEMERKUNG: NAME wechselt den Datei-oder Programmnamen auf der Diskette. Der alte Datei-Programmname muß auf der Diskette existieren.

Der neue Name darf auf der Diskette noch nicht vorhanden sein.

NAME sollte nicht bei geöffneten Dateien angewandt werden.

BEISPIEL:

```
STAT
P1:          0 Bytes
NAME"PRG1" AS "PRG2"
```

NEW

FORMAT: NEW

FUNKTION: Löscht den Speicher und die Variablen.

BEMERKUNG: NEW löscht den gesamten aktuellen Speicherbereich, der mit LOGIN definiert wurde.

NEW sollte vor der Erstellung eines neuen Programms eingegeben werden, um das alte BASIC-Programm in diesem Bereich zu löschen. Nach NEW wird zum BASIC-Modus verzweigt.

Wurde das Programm mit einem Titel geschützt, erfolgt nach NEW die Anzeige "PP Error" (Protect Program). Das NEW-Kommando schließt alle geöffneten Dateien.

BEISPIEL:

```
P5:             1427 Byt
NEW
STAT
P5:             0 Bytes
```

OCT$

FORMAT: OCT$ (⟨X⟩)

FUNKTION: Gibt den oktalen Wert einer Dezimalzahl wieder.

BEMERKUNG: OCT$ konvertiert die Dezimalzahl von (X) zu einem oktalen Wert und gibt diesen als String zurück. Der Wert von (X) darf nur im Bereich von-32768 bis 65535 liegen. (X) wird vorher zu einer Ganzzahl gerundet.

BEISPIEL:

```
10 A=32675:B=9:C=8
20 A$=OCT$(A):B$=OCT$(B)
:C$=OCT$(C)
30 LPRINT A$,B$,C$
40 END

77643          11
10
```

ON ERROR GOTO

FORMAT: ON ERROR GOTO ⟨Zeilennummer⟩

FUNKTION: Fängt einen eventuellen Fehler auf, und verzweigt zu der angegebenen Zeilennummer.

BEMERKUNG: Tritt während des Programmablaufs ein Fehler auf, verzweigt ON ERROR GOTO zu der angegebenen Zeilennummer, und fährt an dieser Stelle mit dem Programmablauf fort.

ON ERROR GOTO sollte als erste Zeilennummer angegeben werden, damit das ganze Programm überwacht wird, und kein Programmabbruch erfolgt. Es wird kein Fehler angezeigt.

Mit einem Error-Unterprogramm, kann die Fehlerbehandlung dann gezielt erfolgen. Um sämtliche Fehler abzufangen und angezeigt zu bekommen, empfiehlt es sich, ein ON ERROR GOTO 0 anzugeben.

BEISPIEL:

```
10 ON ERROR GOTO 70
20 INPUT"WERT:",A
30 IF A<0 THEN ERROR 70
40 INPUT"MENGE:",B
50 IF B<0 THEN ERROR 80
60 GOTO 10
70 REM *** ERROR ***
80 IF ERR=255 AND ERL=30
 GOTO 100
90 IF ERR=250 AND ERL=50
 GOTO 100
100 PRINT"EINGABE FALSCH
"
110 RESUME 20
120 END

WERT:10
MENGE:-5
EINGABE FALSCH
WERT:_
```

ON...GOTO/ON...GOSUB

FORMAT: ON ⟨Ausdruck⟩ | GOSUB / GOTO | ⟨Zeilennummer⟩, [⟨Zeilennr.⟩]

FUNKTION: Sprung zu einer definierten Zeilennummer.

BEMERKUNG: ON...GOSUB/GOTO sind berechtigte Sprunganweisungen, die bei erfüllter Bedingung zu der angegebenen Zeilennummer verzweigen.

Der Ausdruck muß im Bereich von 0-255 sein.

Ist der Ausdruck negativ, erscheint ein FC (Illegal function call) Error. Ist der Ausdruck 0 oder größer als die angegebene Zeilennummer, wird der Befehl ignoriert, und das Programm beim nächsten Statement fortgesetzt.

In diesem Fall erscheint kein Error.

Bei ON...GOSUB muß die definierte Zeilennummer die erste Zeilennummer des Unterprogramms sein.

BEISPIEL:

```
100 INPUT"DATE/TIME (D/T
)";A$
110 I=INSTR("DT",A$)
120 ON I GOSUB 300,200
130 PRINT:GOTO 100
200 LPRINT TIME$:RETURN
300 LPRINT DATE$:RETURN
310 END

01/12/83
14:22:10
```

OPEN

FORMAT: OPEN "Modus", [#] ⟨Dateinummer⟩, ⟨Dateibezeichnung⟩

FUNKTION: Öffnet eine Datei für I/O Optionen.

BEMERKKNG: OPEN eröffnet auf dem spezifizierten Gerät eine Datei mit der angegebenen Dateinummer.

Der Modus kann entweder "I" für Input, oder "O" für Output sequentiell sein.

Die Dateinummer kann nur eine Ganzzahl von 1 bis 16 sein. Ist eine Dateinummer schon geöffnet, kann sie nicht nochmals geöffnet werden.

BEISPIEL:

```
10 INPUT A$
20 OPEN"I",#1,"CAS0:PRIM
.BAS"
30 FOR I=1 TO 10
40 LINE INPUT #1,A$
50 NEXT I
60 CLOSE #1
70 END
```

OPEN "COM0:"

FORMAT: OPEN "⟨Modus⟩", [#] ⟨Dateinummer⟩, "COM0:[(⟨BLPSC⟩)]"

FUNKTION: Spezifiziert und öffnet den RS-232C-Port.

BEMERKUNG: Das OPEN "COM0:" Kommando ist das gleiche wie das normale OPEN, nur wird hier die RS-232C Schnittstelle spezifiziert angesprochen.

Die Übertragungsspezifikation (BLPSC) läßt sich über die Software wie folgt steuern.

BEISPIEL:

```
10 OPEN"O",#1,"COM0:(48N
2F)
```

Bedeutung der BLPSC Spezifikation:

B (Baud Rate)
0 = 110 bps
1 = 150 bps
2 = 300 bps
3 = 600 bps
4 = 1200 bps
5 = 2400 bps
6 = 4800 bps (Standard)

L (Wortlänge)
7 = 7 Bit Wort
8 = 8 Bit Wort (Standard)

P (Parity)
N = No Parity Check (Standard)
E = Even Parity Check
O = Odd Parity Check

S (Stop Bits)
1 = 1 Stop Bit
2 = 2 Stop Bits (Standard)

C (Kontroll-Leitungen)

Die aktiven Kontroll-Leitungen stellen sich durch eine Hexadezimal-Zahl von Ø bis F dar – verbunden zu vier binären Bits, wovon ein Bit mit der Kontroll-Leitung korrespondiert. In der folgenden Darstellung wird die aktive Signal-Linie mit "o" dargestellt, und die inaktive Linie mit "x". Beim RTS-Signal sagt das "+"-Zeichen, daß das positive Signalpotential aktiv ist, und das "-"-Zeichen, daß das negative Signalpotential aktiv ist.

Kontroll-Leitung / Hexadezimal-Zahl	CTS	DSR	RTS	CD
0	○	○	–	○
1	○	○	–	×
2	○	○	+	○
3	○	○	+	×
4	○	×	–	○
5	○	×	–	×
6	○	×	+	○
7	○	×	+	×
8	×	○	–	○
9	×	○	–	×
A	×	○	+	○
B	×	○	+	×
C	×	×	–	○
D	×	×	–	×
E	×	×	+	○
F	×	×	+	×

Nach dem Warmstart ist standardmäßig eingestellt:

Baudrate:	4800 bps
Wortlänge:	8 Bit
Parity:	No parity check
Stop Bit:	2 Stop Bits
CTS:	Ignoriert
DSR:	Aktiv
RTS:	+ Potential ist aktiv
CD:	Ignoriert

OPTION BASE

FORMAT: OPTION BASE [0 | 1]

FUNKTION: Deklariert einen Minimalwert für die Feldvariable.

BEMERKUNG: OPTION BASE wird benötigt, um einen Minimalwert für Feldvariablen von Indizierungen zu deklarieren, der entweder 0 oder 1 sein kann.

Der Ausgangswert ist 0. Wird OPTION BASE 1 ausgeführt, ist das niedrigste Feldindex 1. OPTION BASE kann nicht gewechselt werden, während RUN oder CLEAR ausgeführt wird.

BEISPIEL:

```
130 OPTION BASE 1:DIM A(
9)
```

PCOPY

FORMAT: PCOPY ⟨X⟩

FUNKTION: Kopiert ein BASIC-Programm in einen anderen Programmbereich.

BEMERKUNG: PCOPY kopiert ein BASIC Programm von einem LOGIN n Bereich in einen anderen Programmbereich.

Der Ausdruck X kann ein LOGIN Bereich von 1 bis 5 sein.

Der Programmspeicher, in den kopiert wird, muß Ø Bytes haben, und darf nicht mit einem Title geschützt sein, ansonsten wird ein PP Error (Protect Program) angezeigt.

Ist das zu kopierende Programm zu lang, erscheint ein OM Error (Out of Memory).

In beiden Fällen wird der Kopiervorgang nicht ausführt.

BEISPIEL:

```
LOGIN1
P1:                187 Byte
PCOPY 4
>
```

PEEK

FORMAT: PEEK (⟨Adresse⟩)

FUNKTION: Liest ein Byte von der spezifizierten Adresse.

BEMERKUNG: PEEK liest ein Byte von der spezifizierten Adresse.

Die (Adresse) muß im Bereich von Ø bis 65535 liegen (&HØ bis &HFFFF). Die Adresse wird zu einer Ganzzahl gerundet. Die Adressen &HØ bis &H4D sind reserviert für INPUT/OUTPUT. Werden mit PEEK unerlaubte Adressen gelesen, erscheint ein FC Error (Illegal Funktion Call). Für diese Adressen muß in &H7E &H8Ø geschrieben werden.

BEISPIEL:

```
10 A=PEEK(&H0C00)
```

POKE

FORMAT: POKE ⟨Adresse⟩, ⟨numerischer Ausdruck⟩

FUNKTION: Schreibt ein Byte in die spezifizierte Adresse.

BEMERKUNG: Das POKE Kommando schreibt ein Daten-Byte (8 Bits), spezifiziert im numerischen Ausdruck, in die vorgesehene Adresse.

Der numerische Ausdruck muß eine Ganzzahl im Bereich von Ø bis 255 sein. (&HØ bis &HFF).

Die Adresse muß eine Zwei-Byte-Ganzzahl im Bereich von Ø bis 65535 sein. (&HØ bis &HFFFF).

POKE überschreibt die Speicherstelle. Die Adresse &HØ bis &H4D ist für INPUT/OUTPUT reserviert.

Die Adressen &H4E bis &HA3F mit POKE angesprochen, erzeugen keinen Fehler.

POKE und PEEK sind sehr nützlich zum optimalen Ausnutzen von Speicherbereichen, Laden von Assembler-Unterroutinen, sowie deren Versorgung untereinander mit Ergebnissen und sonstigen Parametern.

Mit dem POKE + EXEC-Statement können verschiedene Zeichen landesspezifisch gesetzt werden. Anstatt die DIP-Switches umzuschalten, genügt es, die Adresse &H7F zu verändern.

Charakter/Land	Eingabe Daten
U.S.A.	&HØ
Italien	&H11
Schweden	&H12
Dänemark	&H13
England	&H14
Deutschland	&H15
Frankreich	&H16
Spanien	&H17

BEISPIEL:

```
10 POKE &H7F,&HØ
20 EXEC &HFF6A

10 POKE &H7F,Ø
20 EXEC &HFF6A
```

POINT

FORMAT: POINT (⟨horizontaler Punkt⟩, ⟨vertikaler Punkt⟩)

FUNKTION: Gibt den Status des spezifizierten Punktes auf dem Graphik-Schirm wieder.

BEMERKUNG: POINT prüft, ob der spezifizierte Punkt horizontal oder vertikal auf dem Graphik-Schirm gesetzt ist.

Auf dem L.C.D. hat der gesetzte Punkt den Status 1, ist der Punkt nicht gesetzt, den Status 0.

Auf dem externen Display mit Farbmonitor, wird der Farbcode mit einer Zahl zwischen 0 und 3 wiedergegeben.

BEISPIEL:

```
100 CLS:DEFINT A-Z
110 DIM P(17,7)
120 LPRINT"ABC"
130 FOR Y=0 TO 7
140 FOR X=0 TO 17
150 P(X,Y)=POINT(X,Y)
160 NEXT X,Y
170 FOR Y=0 TO 7
180 LPRINT
190 FOR X=0 TO 17
200 IF P(X,Y) THEN A$="■
"ELSE A$=" "
210 LPRINT A$;
220 NEXT X,Y
230 END
```

POS

FORMAT: POS (⟨Wert⟩)

FUNKTION: Gibt die horizontale Position des Cursors auf dem virtuellen Schirm wieder.

BEMERKUNG: Der Wert der Zahl muß im Bereich von Ø bis 16 liegen. Wird Ø angegeben, gibt POS die horizontale Position des Cursors auf dem virtuellen Schirm wieder.

Wird eine Zahl zwischen 1 und 16 angegeben, gibt POS die Anzahl Zeichen im Buffer der geöffneten Datei wieder, die Integer-Zahl ist die Dateinummer.

So läßt sich auch die horizontale Position des Druckerkopfes einstellen.

BEISPIEL:

```
10 IF POS(X) 60 THEN PRI
NT CHR$(13)
20 END
```

PRESET

FORMAT: PRESET (⟨ X ⟩, ⟨Y⟩)

FUNKTION: Löscht einen Punkt auf dem Graphik Schirm.

BEMERKUNG: PRESET löscht horizontal und vertikal die gesetzten Punkte auf dem Graphik-Schirm.

BEISPIEL: PRESET (40,25)

PRINT/LPRINT

FORMAT: PRINT | LPRINT [⟨Ausdruck⟩ [; | ,⟨Liste von Ausdrücken⟩]]

FUNKTION: Ausgabe von Daten auf dem L.C.D. oder Drucker.

BEMERKUNG: PRINT zeigt die spezifizierten Daten auf dem L.C.D. an, während LPRINT diesselben Daten auf dem eingebauten Drucker ausdruckt. Es können Strings oder numerische Werte ausgegeben werden.

Wird eine Liste von Ausdrücken ausgegeben, muß ein Komma, Semikolon, oder Blank den Ausdruck trennen.

Das (?) ersetzt den Befehl PRINT.

Der PRINT Befehl (?), kann im Direktmodus auch für Berechnungen, Rechenoperationen usw. benutzt werden.

Ausgabe-Formate: Steht zwischen den Ausdrücken ein Semikolon oder ein Blank, werden diese ohne Zwischenraum angezeigt, bzw. ausgedruckt.

Komma trennt die Ausdrücke.

BASIC teilt eine Zeile in 14-stellige Zonen auf, und gibt den Ausdruck in dieser Form wieder.

Ist ein String länger als das spezifizierte Display, wird automatisch in der nächsten Zeile fortgesetzt.

BEISPIEL:

```
10 PRINT "EPSON"
20 PRINT A$,A,C$,D
30 PRINT A$+"HX-20"
40 PRINT A$;B$;C$
50 END

EPSON
                0
                0
HX-20
```

PRINT USING/LPRINT USING

FORMAT: |PRINT / LPRINT| USING 〈"Formatangabe"〉 |;/,| [〈Ausdruckliste〉]

FUNKTION: Gibt Strings oder numerische Werte in einem spezifizierten Format aus.

BEMERKUNG: PRINT USING/LPRINT USING bringt einen angegebenen Ausdruck in die spezifizierte Formatangabe.

Bei PRINT USING erfolgt die Ausgabe auf dem Display, bei LPRINT USING auf dem eingebauten Minidrucker.

String-Felder

!	Spezifiziert nur das erste Zeichen im String und gibt es aus.
ö	Spezifiziert, daß der String mit (n+2) Zeichen (n Blanks) ausgegeben wird. Ist der String länger als das Feld, werden die zusätzlichen Zeichen ignoriert. Ist das Feld länger, wird es rechtsbündig mit Leerzeichen aufgefüllt.
&	Dient zur Ausgabe von einem String mit variabler Länge. Das &-Zeichen gibt an, daß der String in seiner exakten Länge ausgegeben werden soll.

Numerische Felder

#	Dieses Nummernzeichen steht für die Position einer Ziffer und gibt jeden numerischen Wert rechtsbündig aus.
.	Ein Dezimalpunkt kann an jeder Feldposition stehen.
+	Ein Pluszeichen am Anfang oder Ende eines formatierten Strings kennzeichnet, einen positiven oder negativen Wert.
-	Ein Minuszeichen am Ende des formatierten

Strings gibt einen negativen Wert aus. Am Anfang eines Strings sind ein oder mehrere Minuszeichen eine Formatierungs-Anweisung.

** Zwei Sterne am Anfang der Formatierung lassen am Feldanfang zwei Blanks frei.

$$ Zwei Dollarzeichen werden links vom formatierten Wert ausgegeben. Das Exponentialformat kann nicht mit Dollarzeichen benutzt werden.

**$ Diese Kombination am Feldanfang hat den gleichen Effekt wie zwei Dollars oder zwei Sterne, ein Blank und ein Dollarzeichen werden zusätzlich ausgedruckt.

, Ein Komma links vom Dezimalpunkt in einem formatierten String gibt jedes dritte Zeichen mit einem Komma aus. Rechts vom Dezimalpunkt wird das Komma am Feldende ausgegeben.

^^^^ Nach der Formatierung mit dem Nummernzeichen bedeuten vier Exponentialzeichen das Exponentialformat.

– Formatiert die Zeichen als Literale.Der Unterstrich formatiert das nächste Zeichen wie ein Zeichen ohne Formatfunktion.

% Ist der Ausgabewert länger als das spezifizierte Feld, wird das Prozentzeichen vor die Wertausgabe gedruckt.

Bemerkung: Die formatierten Zeichen zeigen den unterschiedlichen ASCII-Code.

USASCII	Frankreich	Deutschl.	England	Dänemark	Schweden	Italien	Spanien
#	#	#	£	#	#	#	Pt
$	$	$	$	$	¤	$	$
\	ç	Ö	\	ø	Ö	\	Ñ
^	^	^	^	^	Ü	^	^

PRINT#

FORMAT: PRINT # 〈 Dateinummer 〉, [〈 Ausdruck 〉...]

FUNKTION: Schreibt Daten in eine sequentielle Datei.

BEMERKUNG: Die Dateinummer ist die Nummer, unter der die Datei geöffnet wurde.

Ausdruck ist ein numerischer Wert oder ein String, der in die Datei geschrieben wird.

Die Daten werden in demselben Format ausgegeben, mit Komma, Semikolon etc., wie sie geschrieben wurden.

BEISPIEL: PRINT #1,A,B

PRINT # USING

FORMAT: PRINT # 〈 Dateinummer 〉, USING 〈"Format-String"〉; [〈Ausdruck〉 [,〈Liste von Ausdrücken〉...]]

FUNKTION: Schreibt Strings oder numerische Werte in eine sequentielle Datei, mit einem spezifizierten Format.

BEMERKUNG: PRINT # USING schreibt formatierte Strings oder numerische Ausdrücke in die angegebene Datei.

Die gelesenen Daten werden im gleichen Format ausgegebenen wie sie in der Datei abgelegt wurden.

BEISPIEL:

```
10 A(1)=123
20 A(2)=12.34
30 A(3)=123.456
40 A(4)=.12
50 FOR I=1 TO 4
60 PRINT#USING"###.##";A
(I)
70 NEXT I
80 FOR I=1 TO 4
90 PRINT#USING "###.##
";A(I);
100 NEXT I
110 END
```

```
123.00
 12.34
123.46
  0.12
123.00    12.34   123.46
  0.12
```

PSET

FORMAT: PSET (⟨X⟩, ⟨Y⟩) [,⟨Farbe⟩]

FUNKTION: Setzt spezifizierte Punkte auf den Graphik-Schirm.

BEMERKUNG: PSET setzt den spezifizierten Punkt auf den Graphik-Schirm.

X ist die horizontale Koordinate, Y die vertikale Koordinate.

BEISPIEL:

```
100 CLS:PI=3.14159
110 DEFINT X,Y
120 FOR I=0 TO 2*PI STEP
 PI/54
130 X=COS(I)*15+64
140 Y=SIN(I)*15+16
150 PSET(X,Y)
160 NEXT:COPY
170 END
```

PUT%

FORMAT: PUT% ⟨Satznummer⟩, ⟨Variable⟩[⟨Variable]⟩...

FUNKTION: Schreibt variable Sätze in ein RAM-File.

BEMERKUNG: PUT% schreibt variable Werte in den angegebenen Satz.

String-Variable müssen am Ende der Variablenliste stehen, und nur ein String pro Satz ist erlaubt.

BEISPIEL:

```
10 DEFFIL 20,0
20 A=1.2:B%=3:C$="HALLO"
30 PUT%0,A,B%,C$
40 FOR I=0 TO 2
50 LPRINT
60 GET%0,A,B%,C$
70 LPRINT A;B%;C$
80 NEXT I
90 END
```

```
 1.2  3 HALLO

 1.2  3 HALLO

 1.2  3 HALLO
```

PUT

Floppy Disk

FORMAT: PUT (#) Dateinummer (Satznummer)

FUNKTION: Speicherung einer Datei aus einem Puffer mit wahlfreiem Zugriff auf eine Diskettendatei mit wahlfreiem Zugriff.

BEMERKUNG: (Dateinummer) ist die Nummer, unter der die Datei geöffnet wurde. Wenn Satznummer weggelassen wird, erhält die Aufzeichnung die nächste verfügbare Satznummer (nach dem letzten PUT). Die größtmögliche Satznummer ist 32767.

BEISPIEL:

```
10 OPEN"R",#1,"FILE"
20 FIELD#1,20 AS N$,4 AS
 A$,8 AS P$
30 INPUT"DIGIT";CODE%
40 INPUT"NAME";X$
50 IF X$ ="" THEN GOTO 1
20
60 INPUT"WERT";AMT
70 INPUT"PHON";TEL$:PRIN
T
80 LSET N$=X$
90 LSET A$=MKS$(AMT)
100 LSET P$=TEL$
110 PUT#1,CODE%
120 CLOSE
130 END
```

RANDOMIZE

FORMAT: RANDOMIZE [⟨Ausdruck⟩]

FUNKTION: Generiert den Zufallsgenerator.

BEMERKUNG: RANDOMIZE verändert die Reihenfolge der Randomnummern mit dem eingegebenen Ausdruck und generiert den Zufallsgenerator. Ist der Ausdruck nicht gesetzt, wird von EBASIC eine Nummer verlangt, die den Zufallsgenerator aktiviert.

Der Ausdruck muß im Bereich von-32768 bis 32767 liegen.

RANDOMIZE kann nicht im Direktmodus angesprochen werden.

BEISPIEL:

```
10 CLS
20 RANDOMIZE 4
30 FOR I=1 TO 3
40 PRINT RND(I)
50 NEXT I
60 END
```

```
 .835997
 .899615
 .876296
```

READ

FORMAT: READ ⟨Variable⟩ [, ⟨ Variable⟩...]

FUNKTION: Liest Daten von der DATA-Anweisung in die angegebenen Variablen.

BEMERKUNG: Eine READ-Anweisung kann nur in Verbindung mit dem DATA-Statement erfolgen. Die Variable kann numerisch oder ein String sein, der Datentyp muß identisch mit der READ-Anweisung sein.

Stimmen die Variablentypen nicht überein, erfolgt ein SN-Error (Syntax).

Ist die Liste von Variablen in der READ-Anweisung länger als die DATA-Statements, erscheint ein OD-Error (Out of Data).

BEISPIEL:

```
10 READ X,Y
20 A=X+Y
30 PRINT A
40 END
50 DATA 50,25

 75
```

REM

FORMAT: REM [⟨Bemerkung⟩]

FUNKTION: Kommentar, Information im Programm.

BEMERKUNG: Die REM-Anweisung dient ausschließlich zum Kommentieren eines Programms. Die Information nach REM wird vom Programm ignoriert. Für REM kann auch ein Hochkomma (') stehen.

BEISPIEL:

```
10 REM ***********
20 REM T E S T
30 REM ***********
40 ' AUTOR X,Y,Z
```

RENUM

FORMAT: RENUM [⟨Neue Zeilennummer⟩] [, [⟨Alte Zeilennummer⟩] [, [⟨Schrittgröße⟩]]

FUNKTION: Neue Nummerierung der Programmzeilen.

BEMERKUNG: Mit Angabe einer neuen Zeilennummer wird das Programm neu numeriert.

Die alte Zeilennummer ist Ausgangspunkt für die neue Numerierung.

Die Schrittgröße ist automatisch auf 10 eingestellt.

Wird nur RENUM angegeben, beginnen die Programmzeilen mit 10 und werden in 10er Schritten automatisch durchnumeriert. Die Programmzeilen dürfen nicht größer als 64000 sein.

BEISPIEL:

```
RENUM
RENUM 1000,3
RENUM 5,3,5
>
```

RESET

Floppy Disk

FORMAT: RESET [⟨Gerätename⟩]

FUNKTION: Gibt einen Kommandowechsel auf der Disk frei.

BEMERKUNG: Das Diskettensystem beim TF-20 schützt Daten vor dem automatischen Schreiben. Erscheint ein "Disk write Error", gibt RESET diesen Schutz frei. Der Gerätename kann "A, B, C, oder D'' sein.

Fehlt die Angabe des Gerätenamens, wird "A" angenommen. Vor RESET sollte ein CLOSE stehen, damit alle Dateien geschlossen sind.

Eine Diskette muß sich bei Ausführung von RESET im Laufwerk "A" befinden.

BEISPIEL:

```
P1:                0 Bytes
CLOSE
RESET
>                  _
```

RESTORE

FORMAT: RESTORE [⟨Zeilennummer⟩]

FUNKTION: Wiederholt die DATA-Anweisung mehrmals.

BEMERKUNG: Sollen die in der DATA-Anweisung abgelegten Werte mehrmals mit READ eingelesen werden, ist dazu RESTORE erforderlich. Wird keine Zeilennummer angegeben, wird die erste DATA-Anweisung wieder eingelesen.

BEISPIEL:

```
10 READ A,B
20 C=A+B
30 PRINT C
40 RESTORE
50 GOTO 10
60 END
70 DATA 35,50

 85
 85
 85
 85
 85
 85
```

RESUME

FORMAT: RESUME [|NEXT / ⟨Zeilennummer⟩|]

FUNKTION: Setzt das Programm nach einem Fehler fort.

BEMERKUNG: Mit RESUME kann spezifiziert im Programm nach einem Fehler fortgefahren werden.

In der Fehlerroutine wird die entsprechende Zeilennummer mit RESUME aufgerufen.

Es können drei Formate von RESUME aufgerufen werden:

RESUME	Fortsetzen des Programms nach dem Fehler.
RESUME NEXT	Fortsetzen des Programms bei dem nächsten Statement nach dem Fehler.
RESUME nn	Fortsetzen des Programms bei den angegebenen Zeilennummer nn.

BEISPIEL:

```
10 ON ERROR GOTO 100
.
.
.
1000 IF ERR=53 AND E
     RL=70 THEN OPEN
     "O",#1,"COPIE":
     RESUME 500
.
.
.
1100 ON ERROR GOTO 0
```

RIGHT$

FORMAT: RIGHT$ (⟨X$⟩, ⟨I⟩)

FUNKTION: Gibt die rechten I Zeichen des Strings X$ aus.

BEMERKUNG: Der Wert von I muß im Bereich von 0 bis 255 liegen.

Ist I größer als die Gesamtanzahl Zeichen von X$ wird der Befehl nicht ausgeführt.

BEISPIEL:

```
10 CLS
20 A$="EPSON DEUTSCHLAND
"
30 B$="**********"
40 V=24
50 L=LEN(A$)
60 PRINT "LÄNGE IST: ";L
70 FOR I=1 TO L
80 PRINT LEFT$(A$,I)
90 NEXT I
100 CLS
110 FOR I=1 TO 11
120 PRINT RIGHT$(A$,I)
130 NEXT I
140 CLS
150 FOR Z=1 TO 4
160 PRINT MID$(A$,5,3)
170 NEXT Z
180 FOR I=1 TO 11
190 IF I>5 THEN 200 ELSE
 210
200 LPRINT LEFT$(A$,5);:
GOTO 220
210 LPRINT LEFT$(A$,I);
220 LPRINT TAB(7);MID$(B
$,8,2);
230 LPRINT TAB(V-I);RIGH
T$(A$,I)
240 NEXT I
```

```
E      **                D
EP     **               ND
EPS    **              AND
EPSO   **             LAND
EPSON  **           HLAND
EPSON  **          CHLAND
EPSON  **         SCHLAND
EPSON  **        TSCHLAND
EPSON  **       UTSCHLAND
EPSON  **      EUTSCHLAND
EPSON  **     DEUTSCHLAND
```

RND

FORMAT: RND [(⟨X⟩)]

FUNKTION: Eine Zufallszahl wird erzeugt.

BEMERKUNG: RND gibt eine Random-Zahl zwischen 0 und 1 aus. Die erzeugte Zufallszahl hängt von (X) wie folgt ab:

Ist der Wert (X) negativ, wird eine neue Reihenfolge von Random-Nummern generiert.

Ist der Wert (X) 0, bleibt die zuletzt generierte Random-Nummer bestehen.

Ist der Wert (X) positiv, wird die nächste Random-Nummer in der Reihenfolge generiert.

Wird (X) nicht angegeben, wird die nächste Random-Nummer in der Reihenfolge generiert. Die gleiche Reihenfolge gilt für das RUN oder CLEAR Kommando, wenn kein RANDOMIZE Befehl vorausging.

BEISPIEL:

```
100 DEFINT N,B
110 DIM B(120)
120 INPUT"Wiederholung";
A
130 CLS
140 FOR I=0 TO A
150 N=RND*120
160 B(N)=B(N)+1
170 PSET (N,31-B(N))
180 NEXT

RUN
Wiederholung? 1800
_
```

RUN

FORMAT: RUN [⟨Zeilennummer⟩] oder
RUN ⟨Dateiname⟩ [,R]

FUNKTION: Start der Programmausführung.

BEMERKUNG: RUN startet ein im Speicher befindliches Programm.

Wird eine Zeilennummer angegeben, beginnt der Programmstart ab diesem Statement.

RUN mit Angabe eines Dateinamens lädt ein Programm in den Speicher.

Das RUN-Kommando löscht alle Variablen und schließt alle geöffneten Dateien.

Mit der Option "R" wird das Programm ausgeführt, die Dateien bleiben geöffnet.

BEISPIEL:

```
RUN
RUN 1000
RUN "CAS0:PRGR3"
RUN "CAS0:TEST",R
```

RUN "COM0:"

FORMAT: RUN "COM0: [(⟨BLPSC⟩)]"[, R]

FUNKTION: Lädt ein Programm über das Interface RS-232C.

BEMERKUNG: Mit RUN "COM0:" wird ein Programm in den Speicher über das Interface RS-232C geladen.

Dieses Kommando ist ansonsten dasselbe wie RUN.

Das zu ladende Programm über RS-232C muß im ASCII-Format sein. Die Spezifikation BLPSC ist unter OPEN "COM0:" näher erläutert.

BEISPIEL:

```
RUN "COM0:(48N2F)"
```

SAVE

FORMAT: SAVE ⟨Gerätename:Programmname ⟩ [, | A | V |]

FUNKTION: Sichert ein EBASIC Programm auf dem angegebenen Gerät.

BEMERKUNG: Dieses Kommando wird zum Sichern eines EBASIC-Programms auf dem spezifizierten Gerät benutzt. Das Programm kann mehrmals gesichert werden.

Die Option "A" bedeutet, daß das Programm im ASCII-Format gesichert wird.

Wird "A" nicht angegeben, erfolgt die Sicherung im Binär-Format.

Um MERGE benutzen zu können, ist die Programmsicherung im ASCII-Format erforderlich.

Mit der Option "V" wird ein Programm auf der Mikrokassette nach SAVE geprüft. Bei einem anderen Gerät wird "V" ignoriert.

BEISPIEL:

```
Microsoft & EPSON
P1:          0 Bytes
SAVE "CAS0:PROG4",A
_
```

SAVE "COM0:"

FORMAT: SAVE "COM0: [(⟨BLPSC⟩)]",A

FUNKTION: Sichert ein Programm über das RS-232C Interface.

BEMERKUNG: Wird"A" nicht angegeben, wird im Binär-Format gesichert.

SAVE"COM0:" ist derselbe Befehl wie SAVE, nur daß das Programm über RS-232C gesichert wird.

Die Spezifikation (BLPSC) wird unter OPEN "COM0:" näher erläutert.

BEISPIEL:

```
Copyright 1982 by
Microsoft & EPSON
P1:          0 Bytes
SAVE"COM0:(48N2F)",A
```

SAVEM

FORMAT: SAVEM ⟨Programmname⟩, ⟨Basis-Adresse⟩, ⟨Top-Adresse⟩, ⟨Startadresse⟩ [, V]

FUNKTION: Sichert ein Assemblerprogramm.

BEMERKUNG: SAVEM sichert ein Programm in Maschinensprache auf dem spezifizierten Gerät.

Die Basis-Adresse ist die Anfangsadresse des Programms, Top-Adresse die Endadresse des Maschinenprogramms.

Die Option "V" mit der Mikrokassette bedeutet, daß nach dem Sichern das Programm ''verify''-gelesen wird.

Bei einem anderen Gerät wird V ignoriert.

Das Programm wird mit LOADM ab der Startadresse geladen.

BEISPIEL:

```
SAVEM"CAS0:PROG1",&H
0B00,&H0C00,&H0B00
```

SCREEN

FORMAT: SCREEN ⟨Text⟩, ⟨Graphik Modus⟩

FUNKTION: Spezifiziert den Text oder Graphik-Modus.

BEMERKUNG: SCREEN setzt bei dem LCD oder externen Display den Text und den Graphik-Modus.

Der Text-Modus wird nur mit Ø oder 1 spezifiziert.

Mit Ø wird auf dem LCD der Text-Modus (physikal. Screen) gesetzt, mit 1 auf dem externen Display.

Der Graphik-Modus muß zwischen Ø und 2 liegen. Der Graphik-Modus auf dem LCD-Schirm wird mit Ø spezifiziert, 1 setzt den Graphik-Modus bei dem externen Display auf 384 x 192 Punkte.

Wird 2 definiert, können 384 x 192 Punkte angesprochen werden.

Nach dem Warmstart sind automatisch auf dem LCD der Text- und der Graphik-Schirm gesetzt.

Folgende Kombinationen können gesetzt werden:

SCREEN Ø,Ø	Text- und Graphik-Modus auf dem LCD.
SCREEN Ø,1	Text-Modus auf dem LCD, Graphik auf dem externen Display.
SCREEN Ø,2	Text-Modus auf dem LCD, Graphik-Modus auf dem externen Display (384 x 192) Punkte.
SCREEN 1,Ø	Text-Modus auf dem externen Display, Graphik-Modus auf dem LCD.

BEISPIEL:

```
Microsoft & EPSON
P1:          Ø Bytes
SCREEN Ø,1
_
```

SCROLL

FORMAT: SCROLL [⟨Geschwindigkeit⟩] [, [⟨Modus] [, ⟨Spaltenscroll X⟩,⟨Zeilenscroll Y⟩]]

FUNKTION: Der physikalische Schirm wird gescrollt.

BEMERKUNG: Mit SCROLL kann die LCD-Anzeige mit der spezifizierten Geschwindigkeit von Ø bis 9 eingestellt werden. 9 ist die höchste, Ø die niedrigste Scrollgeschwindigkeit.

Der Modus kann mit Ø oder 1 definiert werden. Im Modus Ø ist kein horizontales Scrollen möglich.

Ist der Cursor außerhalb des physikalischen Schirms, kann er nur durch einen Ausgabe-Befehl in die Ausgangs-Position gebracht werden.

Im Modus 1 ist horizontales Scrollen über die gesamte Breite des LCD-Schirms möglich.

Standardmäßig ist der Modus auf Ø gesetzt.

Mit Spaltenscroll X von 1 bis 20 auf dem LCD läßt sich die Anzahl Zeichen scrollen wie mit der CTRL + Cursor-Taste.

Mit dem Zeichenscroll lassen sich die Zeilen, wie mit der SCRN-Taste, von 1 bis 4 auf dem LCD, und 1 bis 16 auf dem externen Display scrollen.

BEISPIEL:

```
SCROLL 9,0,10,4
SCROLL 9,0,16,16

>
```

SGN

FORMAT: SGN (⟨X⟩)

FUNKTION: Gibt den Vorzeichenwert von X aus.

BEMERKUNG: Ist X positiv, ist SGN = 1.
Ist X Null, ist SGN = 0.
Ist X negativ, ist SGN = -1.

BEISPIEL:

```
10 A=2.8:B=23:C=-2.8
20 LPRINT SGN(A),SGN(B),
SGN(C)
30 END
```

```
 1             1
-1
```

SIN

FORMAT: SIN (⟨X⟩)

FUNKTION: Gibt den Sinus von X aus.

BEMERKUNG: X wird in Radiantenform dargestellt und mit einfacher Genauigkeit wiedergegeben. SIN gibt den Sinus von X aus.

BEISPIEL:

```
PRINT SIN (.3)
 .29552
-

PRINT SIN(3.1415926/
2)
1
```

SOUND

FORMAT: SOUND ⟨Ton⟩, ⟨Dauer⟩

FUNKTION: Erzeugt einen spezifizierten Ton.

BEMERKUNG: SOUND erzeugt einen Ton über den Piezoelektrik-Lautsprecher, im Bereich von Ø bis 56.

Ton definiert die Höhe des Klangs, in folgender Bedeutung: Ø kein Ton bzw. Pause.

Pause 1 bis 28 vier Oktaven auf der Tonskala, von C (do) bis B (ti).

13 ist gleich 88Ø Hz.

29 bis 56 Halbton zwischen 1 bis 28

Dauer kann im Bereich von Ø bis 255 definiert werden. Der Lautsprecher ertönt für ein Zeitintervall der Länge ⟨Dauer⟩ multipliziert mit 0,1 Sekunden.

BEISPIEL:

```
10 FOR I=1 TO 10
20 READ A,B
30 SOUND A,B
40 NEXT I
50 END
60 DATA 3,4,7,5,94,11,2,
25,7,2,4,9,7,2,3,8,8,5,4
,2
```

SPACE$

FORMAT: SPACE$ (⟨X⟩)

FUNKTION: Gibt einen String mit X Leerzeichen aus.

BEMERKUNG: SPACE$ gibt einen String mit X Leerzeichen aus. X muß im Bereich von Ø bis 255 liegen.

BEISPIEL:

```
10 FOR X=1 TO 3
20 A$=SPACE$(X)
30 PRINT A$;X
40 NEXT X
50 END
```

```
 1
  2
   3
```

SPC

FORMAT: SPC (⟨X⟩)

FUNKTION: Gibt X Leerzeichen aus.

BEMERKUNG: SPC gibt die in X spezifizierte Anzahl Leerzeichen aus.

SPC kann nur in Verbindung mit PRINT oder LPRINT benutzt werden. X muß im Bereich von Ø bis 255 liegen.

BEISPIEL:

```
10 PRINT "EPSON";SPC(5);
"HX-20"

EPSON     HX-20
```

SQR

FORMAT: SQR (⟨X⟩)

FUNKTION: Berechnet die Quadratwurzel von X.

BEMERKUNG: SQR berechnet die Quadratwurzel von X.
X muß größer als Null sein.

BEISPIEL:

```
10 INPUT"WERT:",A
20 X=SQR(A)
30 PRINT X
40 END

RUN
WERT:250
 15.8114
>
```

STAT

FORMAT: STAT [ALL] [⟨Speicherbereich Nr.⟩]

FUNKTION: Zeigt den Status der Speicherbereiche an.

BEMERKUNG: STAT ALL zeigt die freien und belegten Bytes der 5 Speicherbereiche, des RAM-FILE Bereichs, MEMSET und die gesamten freien Bytes an.

Mit der Pause-Taste kann gestoppt werden, die RETURN-Taste setzt die Anzeige wieder fort.

STAT mit Angabe der Speicherbereich-Nr. zeigt die belegten Bytes dieses Bereich an.

BEISPIEL:

```
STAT3
P3:         487 Bytes
_


STAT ALL
P1:         123 Bytes
P2:           0 Bytes
P3:           0 Bytes
P4:           0 Bytes
P5:           0 Bytes
RAM FILE    256 Bytes
MEMSET     2624
12761 Bytes Free_
```

STOP

FORMAT: STOP

FUNKTION: Unterbrechung des Programmablaufs.

BEMERKUNG: STOP kann an jeder beliebigen Stelle des Programms stehen.

Sofort wird das Programm mit Angabe der Zeilennummer unterbrochen, und BASIC kehrt in den Kommando-Modus zurück. STOP schließt keine geöffneten Dateien.

Mit dem CONT-Befehl kann das Programm fortgesetzt werden.

BEISPIEL:

```
STOP

Break in 270
CONT_
```

STR$

FORMAT: STR$ (⟨X⟩)

FUNKTION: Wandelt X in einen String um.

BEMERKUNG: STR$ konvertiert den Wert von X zu einem String. Für den Wert X können alle Typen von numerischen Konstanten benutzt werden.

BEISPIEL:

```
10 A$="DM"
20 B$=STR$(26.75)
30 PRINT B$;"",A$
40 END

RUN
 26.75          DM

>
```

STRING$

FORMAT: STRING$ (⟨X⟩, | ⟨ String ⟩ / ⟨ num. Ausdruck ⟩ |)

FUNKTION: Gibt einen spezifizierten String aus.

BEMERKUNG: STRING$ gibt einen String von Zeichen als String oder numerischen Ausdruck in der Anzahl von X aus. Bei Angabe eines Strings wird nur das erste Zeichen in der Länge von X ausgegeben. Der numerische Ausdruck wird im ASCII-Code ausgegeben. Der numerische Ausdruck muß im Bereich von 0 bis 255 liegen.

BEISPIEL:

```
10 PRINT STRING$ (4,"M")

RUN
MMMM
Ok

10 PRINT STRING$ (6,65)

RUN
AAAAAA

Ok
```

SWAP

FORMAT: SWAP ⟨Variable 1⟩, ⟨Variable 2⟩

FUNKTION: Austausch zweier Variablen

BEMERKUNG: Mit SWAP kann der Wert von zwei gleichen Variablentypen getauscht werden.

Es können verschiedene Variablentypen sein: Integer, single Precision, double Precision, oder Strings.

Sind die beiden Variablentypen nicht identisch, erscheint ein TM Error (Type Mismatch). Ist der Variable 2-Typ eine einfache Variable, und der Wert ist nicht definiert, erscheint ein FC Error (Illegal function call).

BEISPIEL:

```
10 CLS
20 A$="EPSON":B$="GMBH"
30 SWAP A$,B$
40 PRINT A$,B$
50 END

GMBH            EPSON
```

SYSGEN FLOPPY DISK

FORMAT: SYSGEN

FUNKTION: Erstellt eine neue Systemdiskette.

BEMERKUNG: Bevor SYSGEN gestartet werden kann, muß im Laufwerk "A" eine Systemdiskette eingelegt sein und im Laufwerk "B" eine neue Diskette.

Nach dem Aufruf SYSGEN wird die Frage "Are you sure?" auf dem Display gestellt, um das Kommando mit Y bestätigen zu lassen, oder mit N nochmals zu wechseln.

SYSGEN kopiert nur den Systembereich und die Programme, die mit SYS beginnen. Jede Disk kann so als Systemdiskette konfiguriert werden.

Mit dem RESET-Kommando wird die Anzeige "Disk write protected" gelöscht.

BEISPIEL:

```
SYSGEN_
```

TAB

FORMAT: TAB (⟨I⟩)

FUNKTION: Positioniert einen Ausdruck um I-Blanks nach rechts.

BEMERKUNG: TAB kann nur mit PRINT oder LPRINT benutzt werden. Es wird von links ausgehend zu dem Ausdruck I positioniert.

BEISPIEL:

```
10 PRINT TAB(6);"HX-20"
20 END

      HX-20
```

TAN

FORMAT: TAN (⟨X⟩)

FUNKTION: Gibt den Tangens von X an.

BEMERKUNG: TAN gibt den Tangens des Ausdrucks X in Radiantenform an. X wird in einfacher Genauigkeit dargestellt.

BEISPIEL:

```
10 X=100
20 PRINT TAN(X)
30 END

-.587217
```

TAPCNT

FORMAT: TAPCNT

FUNKTION: Bandzähler für die Mikrokassette.

BEMERKUNG: TAPCNT setzt den Bandzähler der Mikrokassette. Der Zähler kann im Bereich von −32768 bis 32767 gesetzt werden. Mit WIND kann an die entsprechende Stelle vor- oder zurückgespult werden.

BEISPIEL:

```
100 TAPCNT=0
110 OPEN "O",#1,"CAS0:TE
ST"
120 FOR I=1 TO 10
130 PRINT#1,I;I*I
140 NEXT
150 CLOSE
160 WIND 0
170 OPEN "I",#1,"CAS0:TE
ST"
180 IF EOF(1) THEN 220
190 INPUT#1,A,B
200 PRINT A,B
210 GOTO 180
220 CLOSE
230 END
```

oder:
PRINT TAPCNT

TIME$

FORMAT: TIME$="HH:MM:SS"

FUNKTION: Ausgabe der aktuellen Uhrzeit.

BEMERKUNG: Die Uhrzeit kann jederzeit für das entsprechende Land gesetzt oder abgefragt werden.

HH gilt für Stunde, MM für Minute, und SS für die Sekunden.

Die Uhrzeit wird durch einen Clock-Baustein versorgt und kann jederzeit abgefragt werden.

BEISPIEL:

```
P1:          0 Bytes
PRINT TIME$
12:45:39
>

TIME$ = "18:45:00"
>
```

TITLE

FORMAT: TITLE ⟨Programmname⟩

FUNKTION: Hinterlegt einen Programmnamen im Menu.

BEMERKUNG: In jedem LOGIN-Bereich kann ein Programm mit einem max. 8-stelligen Programmnamen in das Menu aufgenommen werden.

Nach dem Einschalten oder mit dem STAT-Kommando wird der Inhalt ersichtlich.

Ist ein LOGIN-Bereich mit einem Titel versehen, kann er nicht mit NEW oder LOAD überschrieben werden (PP Error) Protect Programm.

Ein Titel kann nur wieder gelöscht werden, indem TITLE ohne Programmname eingegeben wird.

BEISPIEL:

```
TITLE "PROGR. 1"
TITLE""
>
```

TRON/TROFF

FORMAT: TRON
TROFF

FUNKTION: Steuert den Trace-Modus

BEMERKUNG: Das TRON-Statement kann im Direkt- oder Indirekt-Modus ein- bzw. ausgeschaltet werden.

Bei der Programmausführung wird jeweils die aktuelle Zeilennummer angezeigt.

Mit TROFF wird dieser Anzeige-Modus wieder ausgeschaltet.

BEISPIEL:

```
10 FOR I=1 TO 5
20 PRINT I
30 NEXT I
40 END
TRON
RUN_
```

USR

FORMAT: USR [⟨Ziffer⟩] (⟨Argument⟩)

FUNKTION: Aufruf eines Assembler-Unterprogramms.

BEMERKUNG: USR ruft ein mit DEF USR definiertes Unterprogramm mit dem angegebenen Argument auf.

Bevor die USR-Funktion aufgerufen werden kann, muß ein Maschinenprogramm im Speicher stehen, und die Startadresse mit DEFUSR definiert sein.

Maximal 10 USR-Funktionen können gesetzt werden.

Mit der angegebenen Ziffer von 0 bis 9 wird das jeweilige Unterprogramm aufgerufen.

Wird keine Ziffer angegeben, wird USR 0 angenommen. Mit ⟨Argument⟩ kann von einem BASIC-Programm ein Argument in das Assemblerunterprogramm transferiert werden.

(siehe auch Seite 224)

BEISPIEL:

```
10 DEF USR0=&H0A40
20 ON ERROR GOTO 160
30 OPEN "O",#1,"COM0:"
40 OPEN "I",#2,"COM0:"
50 A$ = "RS232 TEST STRI
NG"
60 REM
70 PRINT USR0(3600)
80 PRINT TIME$
90 PRINT #1,A$
100 INPUT #2,B$
110 PRINTUSR0(0)
120 LPRINT "EMPFANGEN ";
TIME$
130 LPRINT B$
140 B$ = ""
150 GOTO 60
160 PRINT TIME$
170 LPRINT "FEHLER : ";E
RR;" IN ZEILE ";ERL
180 RESUME 60
```

VAL

FORMAT: VAL (⟨X$⟩)

FUNKTION: Gibt den numerischen Wert von X$ wieder.

BEMERKUNG: Das erste Zeichen von X$ sollte ein: +,-,&,., oder eine Ziffer sein, ansonsten wird Ø ausgegeben. In Hexadezimal kann nur Ø bis F, und in Octal nur Ø bis 7 angegeben werden, alles andere wird ignoriert.

BEISPIEL:

```
10 A$="12345"
20 B$="67890"
30 LPRINT A$+B$
40 LPRINT VAL(A$)+VAL(B$
)
50 END

1234567890
 80235
```

VARPTR

FORMAT: VARPTR (⟨Variablenname⟩)

FUNKTION: Gibt die Adresse des 1. Variablenzeichen aus.

BEMERKUNG: Jede Art von Variablen (numerische Strings oder Feldvariablen) können unter Variablenname spezifiziert werden.

Ein Wert muß erst mit einem Variablennamen hinterlegt werden, dann kann ein VARPTR Befehl folgen.

Die Ausgabe-Adresse ist eine Ganzzahl zwischen −32768 und 32767.

Zu einer negativen Adresse muß 65536 addiert werden, um den wahren Wert zu erhalten.

BEISPIEL:

```
10 A=250
20 X=VARPTR(A)
30 LPRINT X
40 END

 2732
```

WHILE...
WEND

Floppy Disk

FORMAT: WHILE 〈 Ausdruck 〉 WEND

FUNKTION: Führt die dazwischen liegenden Befehle in einer Schleife aus.

BEMERKUNG: Ist die Kondition des Ausdrucks wahr, werden die Befehle bis WEND ausgeführt.

BASIC kehrt automatisch zu WHILE zurück, und wiederholt den Vorgang so lange, bis die Kondition unwahr ist. Danach wird die Programmausführung bei WEND fortgesetzt. Die WHILE...WEND Schleife kann den gesamten Programmspeicher umfassen.

WHILE ohne WEND erzeugt einen WE Error, WEND ohne WHILE einen WH Error.

BEISPIEL:

```
10 A=1
20 WHILE A
30 PRINT"GUTEN TAG"
40 A=A+1
50 IF A=5 THEN END
60 WEND
70 END
```

WIDTH

FORMAT: WIDTH ⟨Zeichenanzahl Zeile⟩, ⟨Anzahl Zeilen⟩ [⟨Scroll Rand⟩]

FUNKTION: Setzt die Größe des virtuellen Schirms.

BEMERKUNG: WIDTH spezifiziert die Größe des virtuellen Bildschirms.

Die Zeichenanzahl pro Zeile kann von 2Ø bis 255, und die Anzahl der Zeilen von 4 bis 255 gesetzt werden.

Die Größe des Bildschirms ist von der Speicherkapazität abhängig.

Wird WIDTH zu groß definiert, erscheint ein OM Error (Out of Memory). Mit dem Scroll-Rand läßt sich links und rechts ein Rand einstellen, so daß sich der Cursor immer auf dem Display befindet.

Nach dem Warmstart ist WIDTH mit 4Ø, 8, 3 auf dem LCD gesetzt, und WIDTH 4Ø, 37, 5 auf dem externen Display.

Wenn WIDTH ausgeführt ist, sind alle Dateien geschlossen, die Variablen und die mit DEF definierten Felder gelöscht.

BEISPIEL:

```
5 WIDTH "LPTØ:",1Ø
1Ø INPUT;A$
2Ø LPRINT A$
3Ø END

EPSON DEUT
SCHLAND GM
BH DÜSSELD
ORF
```

WIDTH ⟨ Gerätename ⟩

FORMAT: WIDTH ["LPT0:", "COM0:"] ⟨ Anzahl Zeichen ⟩

FUNKTION: Setzt die Anzahl Zeichen beim Drucker.

BEMERKUNG: Mit WIDTH wird die maximale Anzahl von Zeichen angegeben, die ausgedruckt werden sollen. Normalerweise ist das Semikolon gleichbedeutend mit dem Ende einer Druckzeile, mit WIDTH werden nur die gewünschten Anzahl Zeichen ausgedruckt, die im Bereich von 1 bis 255 liegen müssen. Dies gilt für PRINT, LPRINT und PRINT#.

NOTIZ Der Gerätename der RS-232C Schnittstelle ist spezifiziert mit "COM0:". Um von dem HX-20 aus mit einem externen Drucker oder Terminal zu kommunizieren, ist dieser Gerätename erforderlich. Ist die Kommunikation fehlerhaft, kann mit der BREAK-Taste der Vorgang unterbrochen werden.

Der externe Drucker ist standardmäßig auf 80 Druckzeichen eingestellt. Um mehr Zeichen drukken zu können, ist vor den ersten PRINT-Befehl WIDTH "COM0:", 255 zu setzen.

BEISPIEL:

```
10 WIDTH 20,25,5
```

WIND

FORMAT: WIND [⟨ Bandzähler ⟩]

FUNKTION: Kontrolliert den schnellen Vor- und Rücklauf der Mikrokassette.

BEMERKUNG: WIND steuert den Vor- und Rücklauf der Mikrokassette. Der Bandzähler muß im Bereich von-32768 bis 32767 liegen. Er kann über TAPCNT abgefragt werden.

Wurde der Bandzähler auf der Kassette mit abgespeichert, kann er mit WIND abgefragt werden. WIND ohne Angabe spult an den Bandanfang zurück und setzt TAPCNT auf Ø

BEISPIEL:

```
10 WIND 300
WIND
WIND 300
```

Programm zum Ausdruck der EPSON-BASIC-BEFEHLE

```
100 CLS:SOUND 16,3
110 S$=STRING$(20,133)
120 PRINTS$
130 PRINT"   BEFEHLS-LIS
TE"
140 PRINTS$
150 FOR T=1 TO 500:NEXT
T
160 CLS:SOUND 20,3
170 H$="0123456789ABCDEF
"
180 PRINT"AUF MINI-PRINT
ER - 1"
190 PRINT"AUF BILD-SCHIR
M  - 2"
200 W$=INPUT$(1):W=VAL(W
$)
210 IF W=1 THEN 220 ELSE
 240
220 OPEN "O",#1,"LPT0:"
230 GOTO 250
240 OPEN "O",#1,"SCRN:"
250 INPUT"Von Adresse $"
,X$
260 IF X$="0" THEN ADR=3
2945:GOTO 280
270 GOSUB 410:ADR=X
280 CLS:PRINT#1,"EPSON -
 BASIC-BEFEHLE":PRINT#1
290 BF$="":FL=0:GOSUB 38
0
300 IF ADR>33539 THEN 40
0
310 BYT=PEEK(ADR):ADR=AD
R+1
320 IF BYT>90 THEN BYT=B
YT-128:FL=1
330 IF BYT=92 THEN BF$="
ö = CHR$(92)":GOTO 360
340 BF$=BF$+CHR$(BYT)
350 IF FL=1 THEN 360 ELS
E 310
360 GOSUB 370:GOTO 290
370 PRINT#1,"$";ADR$;"-
";BF$:RETURN
380 ADR$=RIGHT$("0000"+H
EX$(ADR)+" ",5)
390 RETURN
400 PRINT#1:PRINT#1,"End
e":END
410 X=0:L=LEN(X$):FOR A=
0 TO L-1
420 C$=MID$(X$,A+1,1)
430 FOR D=1 TO 16:IF C$=
MID$(H$,D,1) THEN 450
440 NEXT D:RETURN
450 X=X*16+D-1:NEXT A:RE
TURN
```

```
$80B1 - END
$80B4 - FOR
$80B7 - NEXT
$80BB - DATA
$80BF - DIM
$80C2 - READ
$80C6 - LET
$80C9 - GO
$80CB - RUN
$80CE - IF
$80D0 - RESTORE
$80D7 - RETURN
$80DD - REM
$80E0 - '
$80E1 - STOP
$80E5 - ELSE
$80E9 - TRON
$80ED - TROFF
$80F2 - SWAP
$80F6 - DEFSTR
$80FC - DEFINT
$8102 - DEFSNG
$8108 - DEFDBL
$810E - DEFFIL
$8114 - ON
$8116 - LPRINT
$811C - LLIST
$8121 - RENUM
$8126 - ERROR
$812B - RESUME
$8131 - AUTO
$8135 - DELETE
$813B - DEF
$813E - POKE
$8142 - PRINT
$8147 - CONT
$814B - LIST
$814F - CLEAR
$8154 - OPTION
$815A - RANDOMIZE
$8163 - WHILE
$8168 - WEND
$816C - NEW
$816F - ERASE
$8174 - LOADM
$8179 - LOAD?
$817E - SAVEM
$8183 - SAVE
$8187 - LOAD
$818B - MERGE
$8190 - OPEN
$8194 - CLOSE
$8199 - LINE
$819D - SCROLL
$81A3 - SOUND
$81A8 - MON
$81AB - FILES
$81B0 - MOTOR
$81B5 - PUT
$81B8 - GET
$81BB - LOCATES
$81C2 - LOCATE
$81C8 - CLS
$81CB - KEY
$81CE - WIDTH
$81D3 - PSET
$81D7 - PRESET
$81DD - COPY
$81E1 - EXEC
$81E5 - WIND
$81E9 - GCLS
$81ED - SCREEN
$81F3 - COLOR
$81F8 - LOGIN
$81FD - TITLE
$8202 - STAT
$8206 - PCOPY
```

```
$820B - MEMSET
$8211 - BASE
$8215 - TAB(
$8219 - TO
$821B - SUB
$821E - FN
$8220 - SPC(
$8224 - USING
$8229 - USR
$822C - ERL
$822F - ERR
$8232 - OFF
$8235 - ALL
$8238 - THEN
$823C - NOT
$823F - STEP
$8243 - +
$8244 - -
$8245 - *
$8246 - /
$8247 - ^
$8248 - AND
$824B - OR
$824D - XOR
$8250 - EQV
$8253 - IMP
$8256 - MOD
$8259 - ö = CHR$(92)
$825A - >
$825B - =
$825C - <
$825D - SGN
$8260 - INT
$8263 - ABS
$8266 - FRE
$8269 - POS
$826C - SQR
$826F - LOG
$8272 - EXP
$8275 - COS
$8278 - SIN
$827B - TAN
$827E - ATN
$8281 - PEEK
$8285 - LEN
$8288 - STR$
$828C - VAL
$828F - ASC
$8292 - CHR$
$8296 - EOF
$8299 - LOF
$829C - CINT
$82A0 - CSNG
$82A4 - CDBL
$82A8 - FIX
$82AB - SPACE$
$82B1 - HEX$
$82B5 - OCT$
$82B9 - LEFT$
$82BE - RIGHT$
$82C4 - MID$
$82C8 - INSTR
$82CD - VARPTR
$82D3 - STRING$
$82DA - RND
$82DD - TIME
$82E1 - DATE
$82E5 - DAY
$82E8 - INKEY$
$82EE - INPUT
$82F3 - CSRLIN
$82F9 - POINT
$82FE - TAPCNT

Ende
```

BASIC-KOMMANDOS
zum Ansprechen der Gerätenamen

Device / Kommando	KYBD:	SCRN:	LPT0:	COM0:	CAS0:	CAS1:	PAC0:
LOAD	×	×	×	○	○	○	○
LOADM	×	×	×	×	○	○	○
LOAD?	×	×	×	×	○	○	×
RUN "<file descriptor>"	×	×	×	○	○	○	○
MERGE	×	×	×	○	○	○	○
FILES	×	×	×	×	○	○	○
INPUT#	○	×	×	○	○	○	○
INPUT$	○	×	×	○	○	○	○
EOF	–	×	×	○	○	○	○
LOF	–	×	×	○	–	–	○
SAVE	×	○	○	○	○	○	×
SAVEM	×	×	×	×	○	○	×
LIST	×	○	○	○	○	○	×
PRINT# (USING)	×	○	○	○	○	○	×
POS	×	○	○	○	–	–	×
OPEN mode	I	O	O	I/O	I/O	I/O	I

Die Bedeutung von x oder ○ ist folgende:

○: kann benutzt werden
x: kann nicht benutzt werden
–: Das Kommando wird nicht berücksichtigt

ZUSATZINFORMATIONEN ZUM HX-20

DATEIVERARBEITUNG
RAM-Files

Der RAM-File Bereich (Random) ist eine der vielen Applikationen die EPSON-BASIC bietet. Der RAM-File Bereich ist ein sogenannter Festspeicherbereich. Mit Verwendung von CMOS Bausteinen, die permanent unter Strom stehen und einen geringen Verbrauch haben, wird diese Speichermöglichkeit gegeben.

Möglichkeiten der RAM-Files:

- Daten können im Direktzugriff angelegt werden.
- Daten können frei gewechselt werden.
- Die Daten bleiben nach dem Ausschalten erhalten.
- Die Daten können von beliebig vielen Programmen benutzt werden.
- Eine schnelle Zugriffszeit ist gewährleistet verglichen mit den I/O Übertragungsroutinen von und zu den Peripheriegeräten.

Im HX-20 können gleichzeitig fünf Programme abgespeichert werden. Zu dem RAM-File Bereich haben alle Programmbereiche einen Zugriff. Der Anwender muß den RAM-File Bereich selbst verwalten.

Mit dem DEFFIL Statement wird der Speicherplatz des ersten Satzes definiert und für die gesamten Sätze reserviert, die anschließend mit PUT% geschrieben, oder mit GET% gelesen werden.

Die Länge eines Satzes wird anhand der Variablentypen und der Stringlänge spezifiziert. Ein String kann nur nach numerischen Variablen benutzt werden.

Die Variablentypen haben verschiedene Längen von Bytes.

Die Aufteilung ist wie folgt:

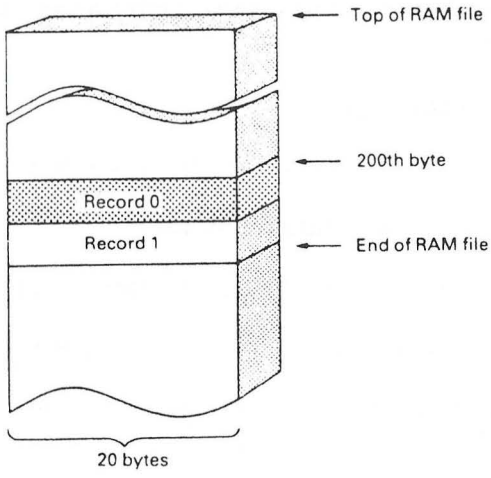
Integer (Ganzzahl) Variable:2 Bytes, Single Precision (einfach genau):4 Bytes, Double Precision (doppelt genau):8 Bytes, String Variablen sind nicht definierbar.

Wird beispielsweise eine Integervariable geschrieben und als einfach oder doppelt genaue Variable gelesen, hat der gelesene Wert einen anderen Inhalt.

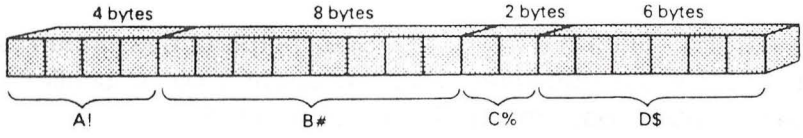
Darstellung der Speicherung der verschiedenen Variablentypen.

Beispiel:
```
100 DEFFIL 20,200,, „
200 PUT% 0, A!,B#,C%,D$. .
330 GET% 0, E!,F#,G%,H$. .
400 END
```

Die Satzlänge in diesem Beispiel teilt sich wie folgt auf:



Die Platzbelegung im RAM-File sieht folgendermaßen aus:



Die Ausgabe des Programms ist folgende:

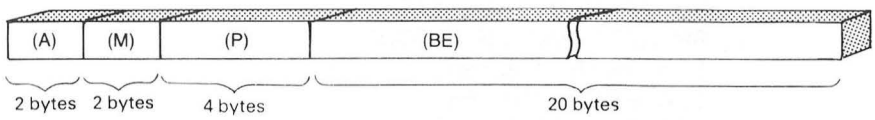
E1=A!, F#=B#
G%=C%, H$=D$

Erstellen einer Datei im RAM-FILE.

Im folgenden Beispiel erstellen wir eine Artikel-Datei:

Zuerst muß die Satzlänge festgelegt werden. In diesem Beispiel setzt sich der Satz zusammen aus Satznummer, Artikel-Nr. Menge, Preis und Bezeichnung. Die Werte Artikel-Nr. und Menge sind Ganzzahl-Variablen, der Preis eine einfach genaue Variable, und die Bezeichnung soll ein String mit 20 Zeichen sein.

Artikel-Nr.	2 Bytes	(A)
Menge	2 Bytes	(M)
Preis	4 Bytes	(P)
Bezeichnung	20 Bytes	(BE)
Gesamt	28 Bytes	



Nun wird die gesamte Dateigröße festgelegt. Sie setzt sich zusammen aus Satzlänge mal Anzahl Sätze. Hier nehmen wir im Beispiel 50 Sätze mit der Satzlänge von 28 Bytes. Nun rechnen wir:

28 X 50 1400 Bytes

Diese 1400 Bytes werden im RAM-File Bereich mit dem CLEAR-Kommando reserviert. In diesem Beispiel wird die Satznummer automatisch abgespeichert, es braucht nur die Artikel-Nr. Menge, Preis und die Bezeichnung geschrieben werden. Wird die Bezeichnung mit mehr als 20 Zeichen eingegeben, werden die überzähligen Zeichen ohne Fehlermeldung ignoriert. Mit DEFINT und DEFSNG werden die Variablen deklariert.

Beispiel:

```
10 CLEAR 200,1400
20 DEFINT U,A,H
30 DEFSNG R   .
40 DEFFIL 28,0
50 IF U>49 THEN 140
60 INPUT"SATZNR. ";U
70 IF U=0 THEN END
80 INPUT"ARTNR. ";A
90 INPUT"MENGE ";H
100 INPUT"PREIS ";R
110 INPUT"BEZEICH ";NA$
120 PUT% U,A,H,R,NA$
130 GOTO 50
140 END
```

Verarbeiten der Daten im RAM-File

Mit diesem Programm werden die zuvor geschriebenen Daten aus dem RAM-File gelesen. Mit Eingabe der Satznummer können die Sätze von 0 bis 50 gezielt gelesen werden. Jeder Satz kann unverändert, oder nach Veränderung mit einem Update zurückgeschrieben werden.

Diese beiden Beispielprogramme könnten z.B. einmal für die Stammdatenerfassung, und zum anderen für die Stammdatenänderung Gültigkeit haben.

Beispiel:

```
10 CLEAR 200,1400
20 DEFINT U,A,H
30 DEFSNG R
40 DEFFIL 28,0
50 PRINT
60 INPUT"SATZNR." ;U
70 IF U>50 THEN END
80 IF U=0 THEN END
90 GET% U,A,H,R,NA$
100 PRINT"ARTNR." ;A
110 PRINT"MENGE ";H
120 PRINT"PREIS ";R
130 PRINT NA$
140 PRINT"UPDATE J / N "
;
150 IF INPUT$(1)<>"J" GO
TO 50
160 PRINT
170 INPUT"ARTNR. ";A
180 INPUT"MENGE ";H
190 INPUT"PREIS ";R
200 INPUT"BEZEICH ";NA$
210 PUT% U,A,H,R,NA$
220 GOTO 50
```

Sequentielle Dateien

Sequentielle Dateien können im Gegensatz zu den RAM-File-Dateien nur in logischer Reihenfolge nacheinander verarbeitet werden. Die Sätze werden wie bei einer Musikkassette abgelegt. Die Dateigröße richtet sich nach dem Speicherplatz des HX-20. Sequentielle Dateien lassen sich sehr einfach erstellen. Die Zugriffszeit ist gegenüber den RAM-File Dateien erheblich länger, da jeder Satz gelesen wird.

Vorteile von sequentiellen Dateien sind:

- die Datei kann so groß wie der Speicherplatz des HX-20 sein.
- einfache Dateierstellung

Nachteile der sequentiellen Dateien:

- Die Sätze können nur nacheinander eingelesen werden.
- lange Zugriffszeit
- Änderungen oder Neuzugänge in eine bestehende Datei sind ziemlich aufwendig.

Dateihandhabung:

Um mit einer sequentiellen Datei arbeiten zu können, sind folgende Prozeduren nötig:

- Datei öffnen. (Mit dem OPEN # Befehl)
- Input der Daten von der Datei zum Speicher – mit dem INPUT# Statement werden die Daten gelesen.
- Output der Daten vom Speicher in die Datei – mit dem PRINT# Statement werden die Daten geschrieben.
- Datei schließen. (Mit dem CLOSE # Befehl)

Jede geöffnete Datei muß mit einer eigenen Nummer versehen sein. Maximal zulässig sind 15 geöffnete Dateien. Jede Datei muß vor Beendigung des Programms wieder geschlossen werden.

Alle Daten, die mit dem PRINT# Statement geschrieben wurden, und mit INPUT# gelesen werden sollen, müssen in der gleichen Reihenfolge und

mit dem gleichen Variablentyp versehen sein, ansonsten erscheint ein TM Error (Type Mismatch). Nach den I/O Operationen muß immer ein CLOSE oder ein END Statement folgen. Die Datenübertragung zwischen dem HX-20 und einem Peripheriegerät erfolgt über einen Buffer, der nur 255 Bytes aufnimmt. Die Schreib Operation wird erst ausgeführt, wenn die 255 Bytes im Buffer stehen. Fehlt das CLOSE oder END Statement, stehen die Daten nur im Buffer, werden aber nicht geschrieben.

Erstellen einer sequentiellen Datei mit Abspeichern auf Mikrokassette:

In diesem Beispiel wird ein Adressen-Programm erstellt. Der Satz besteht aus Name, Branche, Straße, Ort. Abgespeichert werden die Daten auf der eingebauen Mikrokassette mit "CAS0: und auf einer externen Kassette mit"CAS1:

Beispiel:

```
10 OPEN"O",#1,"CAS0:ADR"
20 PRINT
30 INPUT"NAME ";N$
40 IF N$="" THEN 130
50 INPUT"BRANCHE ";B$
60 INPUT"STRASSE ";S$
70 INPUT"PLZ/ORT ";O$
80 PRINT#1,N$
90 PRINT#1,B$
100 PRINT#1,S$
110 PRINT#1,O$
120 GOTO 20
130 CLOSE
140 END
```

Die Daten werden immer dann auf das Band geschrieben, wenn der Buffer mit 255 Bytes gefüllt ist. Wird bei der Eingabe des Namens nur RETURN gedrückt, wird die Datei geschlossen, und das Programm beendet. Durch Drücken der BREAK-Taste wird das Programm abgebrochen, die Daten im Buffer nicht aufgezeichnet, und die Datei nicht geschlossen.

Lesen von sequentiellen Dateien

Mit diesem Programm wird die eingegebene Adressendatei gelesen, und nach gleichen Orten selektiert ausgegeben. Dabei wird das Dateiende mit EOF abgefragt.

Beispiel:

```
10 OPEN "I",#1,"CAS0:ADR
"
20 PRINT"STADT ? "
30 INPUT A$
40 LPRINT" *** STADT ";A
$;"***"
50 IF EOF(1) THEN GOTO 1
40
60 INPUT #1,N$,B$,S$,O$
70 IF A$<>O$ GOTO 50
80 LPRINT
90 LPRINT N$
100 LPRINT B$
110 LPRINT S$
120 LPRINT O$
130 GOTO 50
140 LPRINT " *** ENDE **
* "
150 CLOSE
160 END
```

Korrektur oder Änderungen in sequentiellen Dateien

Bei einer sequentiellen Datei können einzelne Felder im Satz nicht verändert und zurückgeschrieben werden. Um Änderungen vornehmen zu können, muß eine neue Datei erstellt werden, die alte Datei gelesen und verändert, und in die neue Datei geschrieben werden.

Nur wenn die gesamte Datei im Speicher des HX-20 aufgenommen werden kann, d.h. eine kurze Datei ist, können die Daten wie bei den RAM-Files verändert werden.

Mit Hilfe einer externen Kassette läßt sich eine Änderung vereinfacht vornehmen. Die alte Datei ist auf der externen Mikrokassette ("CAS1:), die neue auf der eingebauten ("CAS0:).

Beispiel:

```
10 CNT=TAPCNT
20 OPEN"I",#1,"CAS1:ADR"
30 OPEN "O",#2,"CAS0:WOR
K"
40 IF EOF(1) GOTO 190
50 INPUT #1, N$,B$,S$,O$
60 PRINT N$
70 PRINT B$
80 PRINT S$
90 PRINT O$
100 PRINT"ÄNDERN  J / N
? "
110 IF INPUT$(1)<>"J" GO
TO 170
120 PRINT
130 INPUT"NAME " ;N$
140 INPUT"BRANCHE ";B$
150 INPUT"STRASSE ";S$
160 INPUT"PLZ/ORT ";O$
170 PRINT #2,N$,B$,S$,O$
180 GOTO 40
190 CLOSE
200 WIND CNT
210 PRINT"REWIND KASSETT
E"
220 PRINT
230 PRINT"IF
240 PRINT "IF OK,INPUT"J
"
250 OPEN "O",#1,"CAS1:AD
R"
260 OPEN "I",#2,"CAS0:WO
RK"
270 IF EOF(2) GOTO 310
280 INPUT#1,N$,B$,S$,O$
290 PRINT#2,N$,B$,S$,O$
300 GOTO 270
310 CLOSE
320 END
```

Option Mikrokassette

Das Mikrokassettenlaufwerk ist eine sehr praktische Option des HX-20. Es umfasst die Kassettenfunktionen wie REWIND, schneller und langsamer Vorlauf, und STOP. Vom BASIC-Kommando-Modus aus sind diese Funktionen manuell wie folgt zu steuern:

- Seitlichen Schieber des Kassettenlaufwerks öffnen.
- Mikrokassette mit der offenen Seiten nach unten einlegen und Klappe schließen.
- Durch Drücken der CTRL-Taste, und gleichzeitig PF 1, erscheint der Bandzähler auf dem LCD. Der Bandzähler kann im Bereich von-32767 bis 32767 liegen.

Nun haben die Funktionstasten folgende Bedeutung:

- PF 1 – schneller Vorlauf
- PF 2 – langsamer Vorlauf
- PF 3 – Band STOP
- PF 4 – REWIND
- PF 5 – Ende der manuellen Kassettenfunktion. Zurück zum Ausgangspunkt.
- PF 6 – RESET zu 00000

(Shift + PF 1)

Während die Mikrokassette benutzt wird, ist der eingebaute Minidrucker nicht betriebsbereit.

Es dürfen keine manuellen Kassettenfunktionen betätigt werden, während die Mikrokassette über das Programm angesteuert wird.

Sichern, Check und Laden der Programme über Kassette.

Für die eingebaute Mikrokassette "CAS0:'' benutzen, für die externe Kassette "CAS1:''.

SAVE

⟨Dateinamen⟩ [,A] [,V]

BASIC muß im Kommando-Modus, in dem LOGIN-Bereich, der gesichert werden soll, stehen.

Durch Eingabe des WIND-Kommandos und Drücken der RETURN-Taste wird das Band an den Anfang gespult, und auf 00000 gestellt.

Nun über die Tastatur SAVE und den Programmnamen mit, A eingeben. A bedeutet, daß das Programm im ASCII-Format abgespeichert wird. Wichtig für eine nachfolgende MERGE Operation!

Das Band beginnt sich zu drehen, die LED Anzeige auf dem Mikrokassettenlaufwerk leuchtet auf. Nach Beendigung des SAVE-Vorgangs erscheint der Cursor wieder auf dem Display.

```
P1:          0 Bytes
SAVE"CAS0:PROG1",A

>
```

Check-Operation mit LOAD?

REWIND Operation mit WIND und der RETURN-Taste eingeben. Das Band wird an den Anfang gespult. Nun über die Tastatur LOAD? "CASØ:Programmname" eingeben. Wird das Programm gefunden, beginnt die Aufzeichnungsprüfung. Bei korrekter Aufzeichnung erscheint der Cursor wieder auf dem Display, ansonsten ein I/O Error.

```
STAT
P1:          Ø Bytes
LOAD?
Searching
```

Laden eines Programms über Mikrokassette.

Das Band mit REWIND (WIND-Kommando eingeben, mit der RETURN-Taste bestätigen) an den Anfang zurückspulen.

Den gewünschten LOGIN-Bereich anwählen. Der Cursor muß auf dem Display erscheinen. Nun über die Tastatur LOAD "CASØ:Programmname" eingeben. Das Band beginnt sich zu drehen. Nach dem Ladevorgang erscheint der Cursor wieder auf dem Display.

Wird das Programm nicht gefunden, oder es erscheint eine Fehlermeldung auf dem LCD, Vorgang wiederholen.

```
P1:          Ø Bytes
LOAD"CASØ:PROG1"
Searching
_
```

Handhabung von externen Kassettenrecordern

Der HX-20 ist standardmäßig mit einem Interface für eine externe Kassette ausgerüstet. An der rechten Geräteseite befinden sich drei Steckbuchsen, die mit MIC, EAR, und REM bezeichnet sind, was für Eingang, Ausgang und Fernsteuerung steht.

Der REMote-Stecker kann in den meisten Fällen weggelassen werden, da er nur den Motor des Kassettenlaufwerks steuert.

Wird ein Kassettenrecorder angeschlossen, müssen die Stecker an beiden Geräten (HX-20 und Kassettenrecorder) übereinstimmen. Der MIC-Stecker ist meistens rot und der EAR Stecker weiß.

Der günstigste Aufzeichnungspegel muß herausgefunden werden, da die meisten Geräte unterschiedlich sind.

Um Programme laden, sichern, oder prüfen zu können, müssen folgende Konditionen eingestellt sein:

- BASIC muß angewählt sein.
- BASIC muß im Kommando-Modus stehen.

Der Gerätename des externen Kassettenrecorders ist "CAS1:''.

Um ein Programm auf die externe Kassette zu schreiben, ist folgende Eingabe nötig:

SAVE"CAS1:Programmname", A

Das "A" steht für nachfolgende MERGE-Operationen, bedeutet ASCII-Format.

Um ein Programm von der externen Kassette zu laden, ist folgende Eingabe nötig:

LOAD"CAS1:Programmname"

Soll die Aufzeichnung des Programms auf der externen Kassette geprüft werden, ist folgende Eingabe nötig:

LOAD?"CAS1:Programmname"

Diese Kommandos unterscheiden sich nur im Gerätenamen von der eingebauten Mikrokassette.

Programme in Maschinensprache

In Maschinensprache oder in Assembler geschriebene Programme können in EPSON BASIC mit der USR und EXEC Funktion aufgerufen werden. Mit USR können Daten wie ein Argument benutzt werden.

Ist ein Bit in Maschinensprache falsch, können die Programme, auch die BASIC Programme, zerstört werden.

Die in Maschinensprache geschriebenen Programme sind in einem separaten Speicher untergebracht, der die Daten und die BASIC-Programme schützt. Im HX-20 liegt der MEMSET-Bereich vor den Programmspeichern für BASIC. Mit dem MEMSET-Kommando wird der Programmbereich definiert. Der MEMSET-Bereich muß ein Byte höher sein als die Endadresse des Maschinenprogramms. Erst dann kann das Binär-Programm geladen werden. Standardmäßig ist der MEMSET-Bereich auf 2624 Bytes festgelegt.

Schreiben und Laden von Programmen in Maschinensprache

Kurze Programme können mit dem POKE-Befehl in den MEMSET Bereich geschrieben werden. Mit der MONITOR-Funktion wird das Programm direkt in den Speicher geschrieben und geladen. Ein geschriebenes Programm in Maschinensprache auf Kassette wird mit LOADM "CAS0: Programmname" wie ein BASIC-Programm geladen und mit SAVEM gesichert.

Programme, die auf einem CMOS-Baustein im ROM-Sockel gespeichert sind, können wie ein Dienstprogramm im HX-20 benutzt werden. In diesem Fall wird die Programm-Ladezeit erheblich verkürzt.

USR Funktion

Das Format der USR Funktion, um ein Maschinenprogramm aufzurufen, ist folgendes:

USR (⟨ Zahl ⟩) (⟨ Argument ⟩)

Zahl kann eine Ganzzahl von Ø bis 9 sein, die mit dem DEFUSR-Befehl übereinstimmen muß. Das Argument kann ein numerischer oder ein String-Ausdruck sein. Die USR Funktion hat ein Argument. Der Akkumulator A enthält den Wert des spezifizierten Argumenttyps.

Der Wert in dem Akkumulator hat folgende Bedeutung:

2 (Integer) Ganzzahl
3 String
4 Single precision (einfache Genauigkeit)
8 Double precision (doppelte Genauigkeit)

Ist der Argumenttyp numerisch, zeigt der Wert in dem Indexregister die Adresse des "floating-point accumulators" an, wo das Argument gespeichert ist.

Wenn die USR-Funktion zu BASIC zurückkehrt, müssen die Daten in demselben Format an der gleichen Speicherstelle wie das Argument abgespeichert werden. Für die Rückkehr in das BASIC-Programm wird das Unterprogramm RTS (&H39) gebraucht. Der Wert des Stack-Pointers für die Rückkehr in das BASIC-Programm muß identisch mit dem Aufruf der USR-Funktion sein.

EXEC-Statement

Unterprogramme in Maschinensprache können mit dem EXEC-Statement wie folgt aufgerufen werden:

EXEC ⟨Adresse⟩

Wurde ein Maschinensprache-Programm mit LOADM,R in den Speicher geladen, oder die Startadresse im Programm spezifiziert, muß die Adresse bei EXEC nicht angegeben werden.

EXEC führt das Programm nur ab der spezifizierten Adresse aus, hat aber keine Funktion, um Befehle zu durchlaufen. Werden Variablen von BASIC und vom Maschinenprogramm mit EXEC aufgerufen, werden sie direkt in den Speicher geschrieben und gelesen. Mit PEEK und POKE wird die spezifizierte Adresse im Maschinenprogramm von BASIC aus angesprochen.

VARPTR-Funktion

Mit der VARPTR-Funktion kann geprüft werden, auf welcher Adresse die spezifizierte Variable im Speicher steht. VARPTR wird benutzt, um die mit EXEC aufgerufene Variable in das Maschinenprogramm zu übertragen, oder zwei oder mehr Variable während der USR-Funktion zu durchlaufen.

Bevor VARPTR benutzt werden kann, muß die Variable mit dem Argument spezifiziert und der Wert zugewiesen sein.

Die Ausgabe der VARPTR Funktion ist die Top-Adresse der spezifizierten variablen Daten.

MEMORY MAP

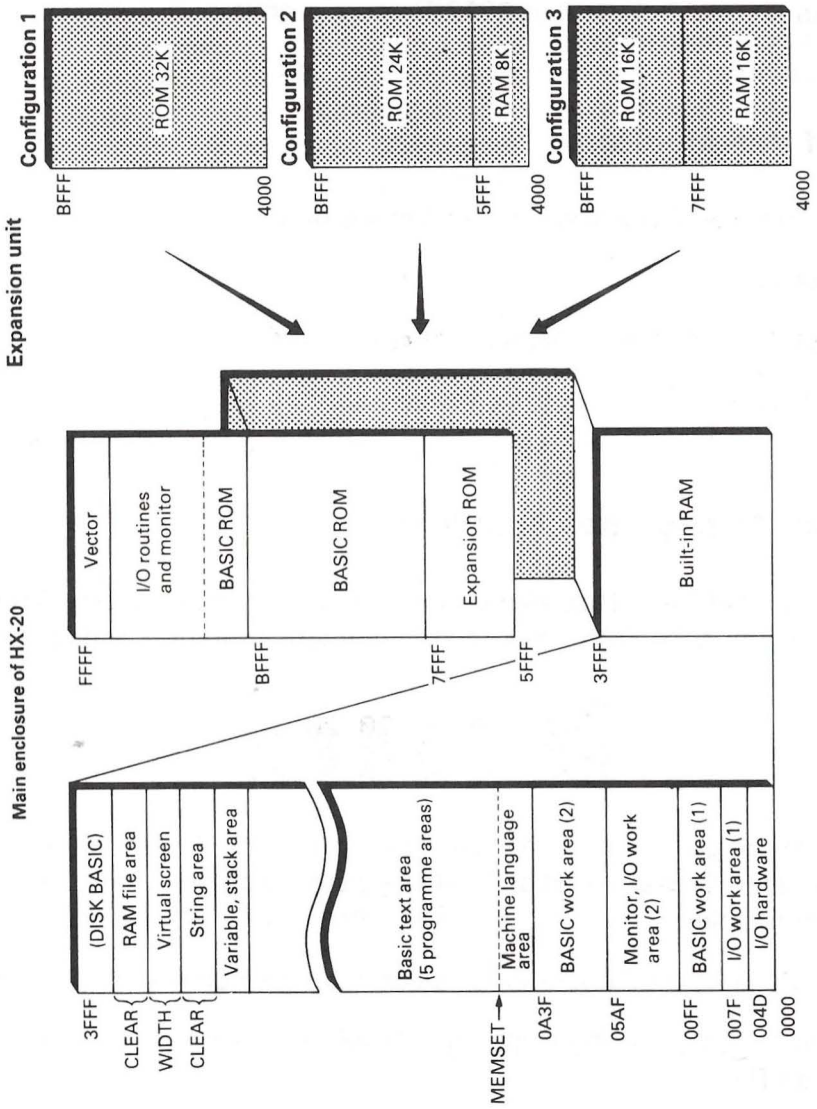
Der Speicher des HX-20 ist in folgende Bereiche aufgeteilt:

Adresse			
0000	bis	004D	I/O Ports
004E	bis	007F	RAM. I/O Routinen und Workbereich.
0080	bis	00FF	RAM. Workbereich für BASIC.
0100	bis	04AF	RAM. I/O Routinen Workbereich und Buffer.
04B0	bis	0A3F	RAM. Workbereich für BASIC.
0A40	bis	3FFF	RAM.
4000	bis	5FFF	Nicht benutzt.
6000	bis	7FFF	ROM. (ROM 5) Option 8K Byte ROM.
8000	bis	9FFF	ROM. (ROM 4) BASIC-Interpreter.
A000	bis	BFFF	ROM. (ROM 3) BASIC-Interpreter.
C000	bis	CFFF	ROM. (ROM 2) BASIC-Interpreter.
D000	bis	DFFF	ROM. MENU, MONITOR und Virtual Screen.
E000	bis	FFFF	ROM. (ROM 1) I/O Routinen.

Der ROM-Inhalt ist wie folgt belegt:

Adresse			
FFD0	bis	FFD1	Adresse der Tastatur.
FFD2	bis	FFD3	Datenadresse des Buffers im Minidrucker.
FFD4	bis	FFD5	Datenadresse des I/O Buffers der externen Mikrokassette.
FFD6	bis	FFD7	Datenadresse des I/O Buffers der eingebauten Mikrokassette.
FFD8	bis	FFD9	Datenadresse I/O Buffer RS-233C
FFDA	bis	FFDB	Topadresse des physikalischen Screen-Buffers.
FFDC	bis	FFDD	Topadresse der I/O Routinen-Buffer.
FFDE	bis	FFDF	Datenadresse für die Scroll-Geschwindigkeit auf dem virtuellen Schirm.
FFE0	bis	FFE1	Topadresse des Headersatzes auf der externen Mikrokassette.
FFE2	bis	FFE3	Adresse des Headersatzes auf der eingebauten Mikrokassette.

Memory Map


Configuration 1
ROM 32K
BFFF
4000
Configuration 2
ROM 24K
RAM 8K
BFFF
5FFF
4000
Configuration 3
ROM 16K
RAM 16K
BFFF
7FFF
4000
Expansion unit
Main enclosure of HX-20
Vector
I/O routines and monitor
BASIC ROM
BASIC ROM
Expansion ROM
Built-in RAM
FFFF
BFFF
7FFF
5FFF
3FFF
(DISK BASIC)
RAM file area
Virtual screen
String area
Variable, stack area
Basic text area (5 programme areas)
Machine language area
BASIC work area (2)
Monitor, I/O work area (2)
BASIC work area (1)
I/O work area (1)
I/O hardware
3FFF
CLEAR
WIDTH
CLEAR
MEMSET
0A3F
05AF
00FF
007F
004D
0000

MONITOR

Der HX-20 kann nicht nur in BASIC programmiert werden, sondern auch in Assembler-Maschinensprache. Im HX-20 ist ein 6301 Prozessor, der aufwärts kompatibel mit den 6800 Prozessoren ist.

Aufruf des Monitors

Der Monitor kann wie folgt aufgerufen werden:

- über das MENU mit der Funktion 1
- über das MON Kommando von BASIC aus.
- über einen TRAP-Interrupt vom Programm aus.

Display-Anzeige des Monitors

Die Daten auf dem Display im Monitor sind immer im physikalischen Schirm. Nach dem Aufruf des Monitors erscheint folgende Anzeige:

```
-D
0A40: 00 0E 20 20 20
0A45: 20 20 20 20 20
0A4A: 00 00 00 00 00
```

A und B zeigen den Inhalt des Akkumulators A und B an, X die Indexregister, C das Condition Code Register, S den Stack Pointer, P den Programmzähler.

Das "-" Zeichen ist die Cursor-Position für eine Kommando-Eingabe.

Wird ein Programm-Interrupt mit TRAP erzeugt, zeigt das Display folgendes Bild:

```
-
Trap!
 A=00 B=00   X=2300
 C=F0 S=4AE8 P=2300
```

Kommando Eingabe Typen

S	Set	Verändern des Speicherinhalts.
D	Dump	Anzeige des Speicherinhalts.
G	Go	Programmausführung

X	Examine	Verändern der Indexregister
R	Read	Laden von Programmen oder Daten von einem externen Speicher in den HX-20
W	Write	Schreibt den Speicherinhalt in einen externen Speicher.
V	Verify	Prüfen der Ausgabedaten.
A	Adress	Spezifiziert den Speicherbereich beim Laden von externen Programmen oder beim Sichern auf einen externen Speicher.
K	Key	Spezifiziert die Daten für die automatische Tastatureingabe, wenn der Powerschalter auf ON steht.
B	Back	Rückkehr zur Ausgangsposition vor dem Aufruf des Monitors.

Eingabe der Kommandos

Die Monitor-Kommandos können von der 2. bis zur 20. Spalte der Cursor-Anzeige folgend, eingegeben werden. Die Eingabe kann nur in der ersten Zeile erfolgen. Das erste Zeichen ruft das entsprechende Kommando auf, danach können ein oder mehrere Argumente folgen.

Die Taste INS/DEL wird nur als Delete Taste (&H08) akzeptiert, alle anderen Codes aus der Tabelle sind Eingabe-Kommandos. Zwischen dem Kommando und dem Argument kann ein Leerzeichen stehen.

Beschreibung der Kommandos

S Set Kommando

Dieses Kommando zeigt die Speicheradresse an und dient zum Austauschen von Speicherwerten.

-S 1000 49

Nach Eingabe des Kommandos S muß die Adresse mit 4 Stellen oder weniger in Hexadezimal eingegeben werden. Mit der RETURN-Taste muß die Eingabe bestätigt werden. Auf dem Display erscheint nun der Cursor

neben der Anzeige des Speicherinhalts. Nun kann der Inhalt verändert werden, die Eingabe in Hexadezimal mit 2 Stellen. Nach Auslösen mit der RETURN-Taste erscheint die nächste Adresse automatisch. Der neue Inhalt wird jeweils in den Speicher zurückgeschrieben. Soll das Kommando geändert werden, genügt die Eingabe eines Punktes (.) und die RETURN-Taste. Oder mit der INS/DEL Taste bis zur Position von "S" gehen, und ein neues Kommando eingeben.

D DUMP-Kommando

Dieses Kommando zeigt nach Eingabe einer Adresse den Speicherinhalt an.

D 1000

Die Speicheradresse wird hexadezimal mit max. 4 Stellen eingegeben. Nach Drücken der RETURN-Taste wird der Speicherinhalt mit 15 Bytes angezeigt. Um die nächsten Adressen angezeigt zu bekommen, genügt das Drücken der RETURN-Taste.

Die Adresse 0000 bis 004D ist für I/O Routinen reserviert, und für die Kommandos "D" und "S" geschützt.

G Go-Kommando

Go setzt den Programmzähler auf den eingegebenen Wert, und startet von dieser Adresse aus das Programm.

G A3B5

Wird von BASIC aus in den MONITOR gesprungen, kann mit dem Kommando "G" und der Eingabe der Programmadresse wieder in das BASIC-Programm zurückgesprungen werden. Oder vom Monitor aus in das MENU zurück.

X Examine-Kommando

Dieses Kommando zeigt den Registerinhalt an, und ermöglicht eine Veränderung.

X A=00

Nur das Kommando X eingeben und mit der RETURN-Taste bestätigen. Soll der Accumulator A verändert werden, den Wert hexadezimal eingeben, und RETURN drücken. Der neue Inhalt wird in der dritten Zeile angezeigt. Das Betätigen der RETURN-Taste zeigt alle Register A, B, X, C, S, P und wieder mit A beginnend an. Das Aufheben des Kommandos erfolgt mit der Eingabe eines Punktes (.), oder mit der DEL-Taste an die Position von X.

TRAP

Der HX-20 verzweigt automatisch in die MONITOR TRAP-Funktion, wenn ein nicht definiertes Kommando für die 6301 CPU angesprochen wird.

```
_
TrAP!
 A=00 B=00   X=2300
 C=F0 S=4AE8 P=2300
```

Durch Drücken der RESET-Taste kann in das MENU zurückgekehrt werden.

A Adress-Kommando

Mit dem Kommando "A" wird der Speicherbereich für die Eingabe eines Programms oder Daten definiert, die mit dem Kommando "R" eingelesen werden. Das gleiche gilt für die Ausgabe mit dem "W"-Kommando.

A RETURN

Folgende Eingaben werden mit dem ''A''-Kommando gesetzt:

A	T=0000	Anfangs-Adresse (Top address)
A	L=0000	End-Adresse (Last address)
A	O=0000	Startpunkt (Offset value)
A	E=0000	Start-Adresse (Entry Point)

Soll kein Wert verändert werden, nur die RETURN-Taste drücken. Das Kommando kann durch Punkt (.) Eingabe und RETURN wieder verlassen werden.

W Write-Kommando

Daten oder Programme werden mit diesem Kommando auf ein externes Speichermedium geschrieben.

W M,Programmname.Typ RETURN

Folgende Gerätenamen müssen angegeben werden:

M: eingebautes Mikrokassettenlaufwerk
C: externes Mikrokassettenlaufwerk

Bevor ein Programm ausgelagert werden kann, muß zuvor mit dem Kommando "A" der Speicherbereich definiert werden, mit Angabe der Anfangsadresse, Endadresse und der Startadresse. Wird der Startpunkt auch definiert, muß die Anfangsadresse und der Startpunkt addiert werden.

Der Programmname kann max. 8 alphanumerische Zeichen, umfassen und der Dateityp ein String mit 3 alphanumerischen Zeichen sein. War die Aufzeichnung in Ordnung, erscheint auf dem Display ein "ok".

R Read-Kommando

Mit diesem Kommando werden Programme oder Daten von der Mikrokassette oder der ROM-Cartridge in den Speicher des HX-20 eingelesen.

Das zu ladende Programm kann nur ein Programm in Maschinensprache sein. Es muß mit SAVEM oder mit dem "W" Kommando gesichert worden sein.

Bevor das "R"-Kommando aktiviert werden kann, müssen mit dem "A"-Kommando die Adressen T, L, O und E gesetzt werden.

Die Gerätenamen für den externen Speicher lauten wie folgt:

M: eingebautes Mikrokassettenlaufwerk
C: externes Mikrokassettenlaufwerk
P: ROM Cartridge.

Das "R"-Kommando wird folgendermaßen eingegeben:

R M,Programmname.Typ,R RETURN
oder **R** M,Programmname.Typ RETURN

mit der Option R wird das Programm sofort nach dem Einlesen ab der Anfangsadresse gestartet. War der Lesevorgang fehlerhaft, erscheint auf dem Display "Error".

Dateiname, Programmname und Typ

Der Programm-Dateiname kann max. 8 alphanumerische Zeichen haben, das erste Zeichen muß ein Buchstabe sein. Der Typ muß ein String mit 3 alphanumerischen Zeichen sein. Wird die Option R angegeben, muß der Dateityp vorhanden sein. Es kann anstelle des Programmnamens auch ein "*.*" angegeben werden. In diesem Fall werden alle folgenden Zeichen akzeptiert.

R M,*.*

Werden nur die ersten Buchstaben des Programmnamens eingegeben, muß ein "*" dahinter stehen.

R M,AB*.BIN,R

V Verify-Kommando

Dieses Kommando prüft die Aufzeichnung auf ein externes Speichermedium.

V M,Programmname.Typ RETURN

Der Gerätename kann folgender sein:

M: eingebautes Mikrokassettenlaufwerk
C: externes Mikrokassettenlaufwerk

Das Verify-Kommando vergleicht nicht den Speicherinhalt im HX-20 mit dem Programm oder den Daten auf der Kassette, sondern prüft nur die Aufzeichnung.

Folgende Fehlermeldungen können aus den Kommandos "W", " R", und "V" resultieren:

Fehlercode	Erläuterung
00	Normaler Datentransfer
01	End of File
80	Die Mikrokassette ist nicht angeschlossen
81	Eingabe-Fehler
82	Das Programm wurde nicht gefunden
83	Fehlerhaftes Programm oder Daten
86	Falsches Datenformat
8B	Ein undefiniertes Programm wurde gefunden

8C	Der Speicherbereich wurde nicht definiert
91	Ausgabefehler
99	Falscher Gerätename
A0	Die ROM Cartridge ist nicht vorhanden
A1	Falscher Programmname (ROM Cartridge)
A4	Fehler im Header in der ROM Cartridge
A5	Falsche Header Adresse in der ROM Cartridge

K Key-Kommando

Mit diesem Kommando kann eine automatische Tastatureingabe erzielt werden, sobald der POWER Schalter auf ON steht. Mit dem Einschalten des HX-20 wird die Eingabe gestartet.

Maximal 17 String Zeichen können eingegeben werden. Die Funktionstasten werden mit zwei Zeichen gezählt. Die RETURN-Taste ist ein Zeichen, sie wird auf dem Display als CR dargestellt.

Die Eingabe wird durch Drücken der Tasten CTRL, SHIFT und "3"(§) beendet. (Wie initialisieren)

K2PRINT"HX-20"

Nach dem Einschalten des HX-20 erscheint folgendes Bild:

```
P1:       0 Bytes
PRINT"HX-20"
HX-20
>
```

Soll das "K"-Kommando nicht wirksam werden, die BREAK-Taste während des Einschaltens drücken.

Um das Kommando zu wechseln, muß eingegeben werden:

K CTRL §

B Back-Kommando

Durch dieses Kommando gelangt man in den Aufruf-Modus des Monitors.

B RETURN

Das "B"-Kommando kehrt in das MENU zurück, wenn der MONITOR mit der Funktion "1" aufgerufen wurde. In den BASIC-Modus, wenn das MON-Kommando eingegeben wurde.

RS-232C Serielle Kommunikation

Der HX-20 verwendet eine im folgenden beschriebene asynchrone serielle Kommunikations-Schnittstelle.

Bei der asynchronen Übertragung wird ein Startbit (Ø) vorab gesendet, um anzuzeigen, daß die Übertragung begonnen hat. Wie Sie in der Abbildung sehen, ist der Level auf ''1'' gesetzt, wenn keine Daten vorhanden sind. Sobald Daten angekündigt werden, sinkt der Level auf ''Ø'' ab.

Dem Startbit folgen die Datenbits des zu übertragenden Bytes, wobei mit dem niedrigsten Bit (Bit Ø) begonnen wird. Den Datenbits folgen ein bzw. zwei Stop-Bits, um anzuzeigen, daß die Übertragung beendet ist und um den Level wieder auf ''1'' zu setzen. Die Übertragungsgeschwindigkeit für jedes einzelne Bit hängt von der gewählten Bitrate ab. So entspricht z. B. die Übertragungsgeschwindigkeit eines Bits bei einer Bitrate von 3ØØ Bits pro Sekunde 3,3 Millisekunden.

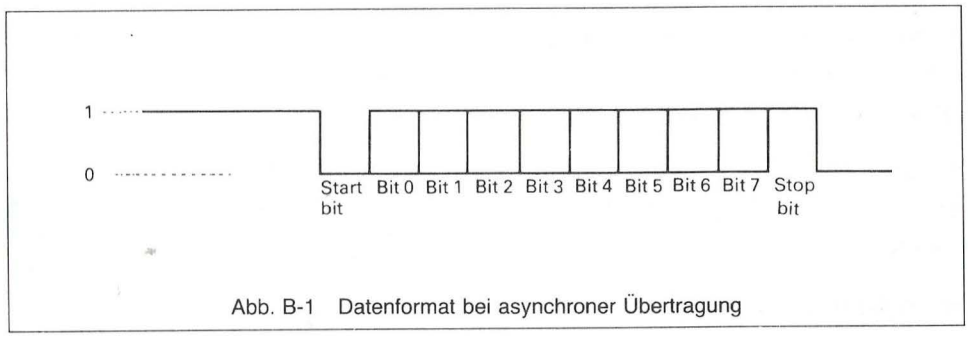


Abb. B-1 Datenformat bei asynchroner Übertragung

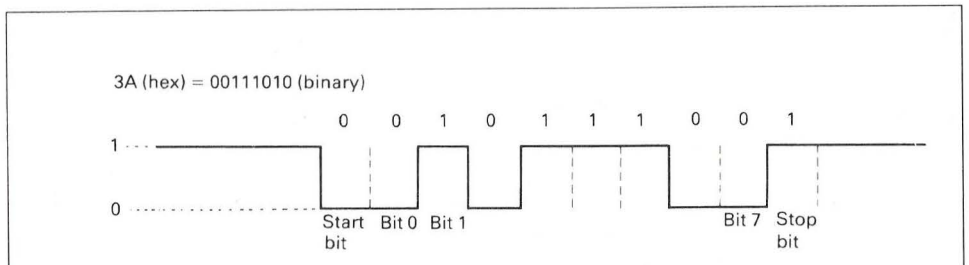


Abb. B-2 Beispiel für Übertragung des 8-Bit-Wortes 3A mit einer Übertragungsrate von 3ØØ Bps.

Der Spannungspegel variiert von –8V(–3 bis –15) bei Level ''1'' auf +8V(+3 bis +15) bei Level Ø (s. Abb. B-3).

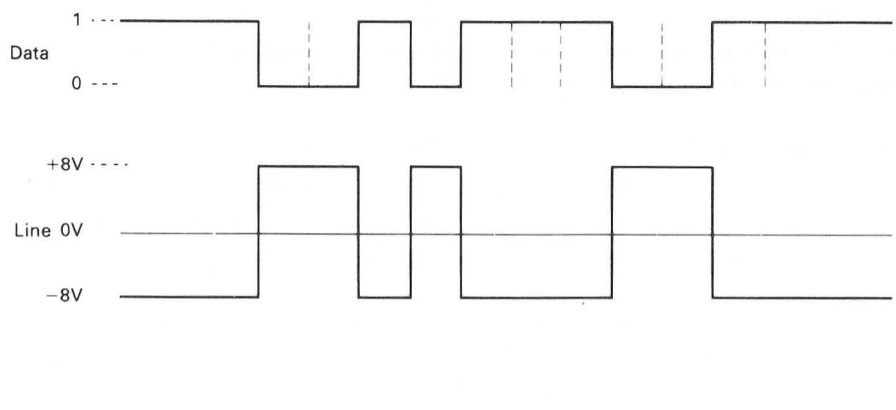


Abb. B-3 Abhängigkeit von Datensignalen und Spannungspegel

Einschränkungen bei der seriellen Kommunikation mit HX-20

Da der HX-20 als batteriegeladenes Gerät konstruiert ist, wird die serielle Schnittstelle erst dann mit der benötigten Spannung versorgt, wenn eine Kommunikation ausgeführt wird.

Deshalb wird das Signal für einen kurzen Augenblick unstabil, nämlich dann, wenn bei Beginn der Kommunikation der Stromverbrauch kurzfristig ansteigt.

Die folgende Abbildung zeigt dies deutlich:

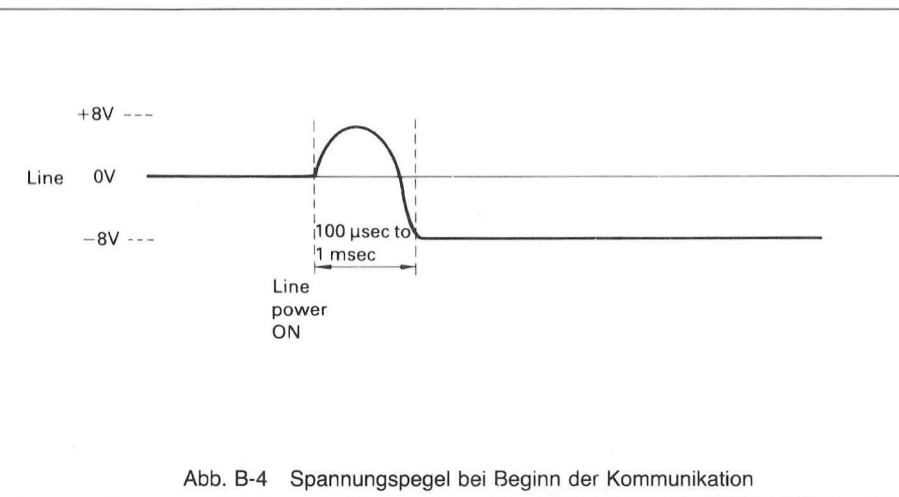


Abb. B-4 Spannungspegel bei Beginn der Kommunikation

Wenn die RS-232C Schnittstelle dazu benutzt wird, serielle Daten zu übertragen, so steigt TXD (Transmit Data) an, wie es in der Abbildung dargestellt wird.

Dies kann zu einem nicht ganz einwandfreien Empfang bei der entsprechenden Empfangseinrichtung führen, ist jedoch abhängig von der gewählten BIT-Rate oder anderen Zuständen.

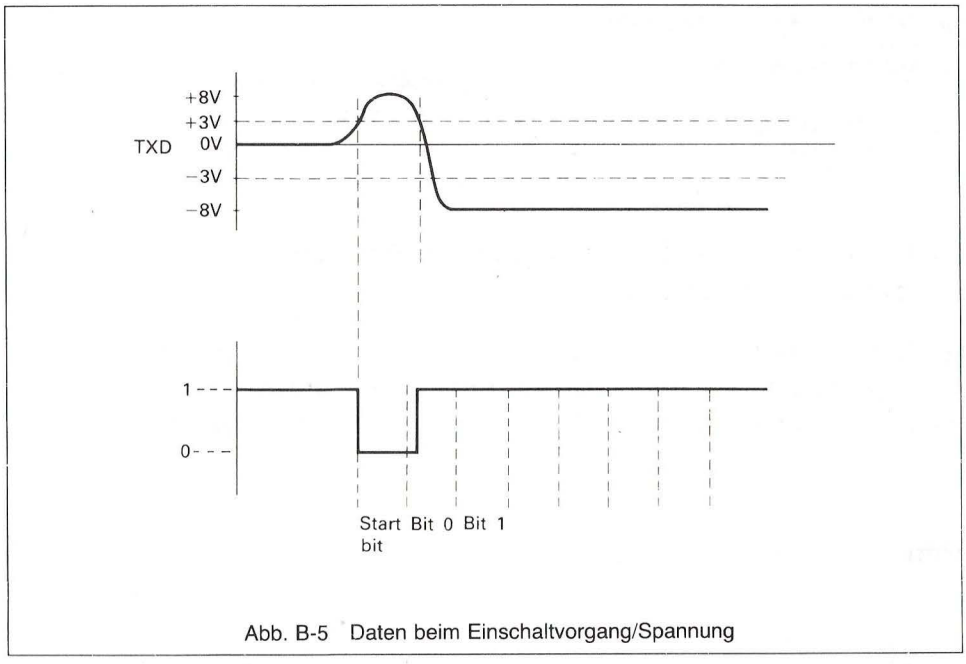


Abb. B-5 Daten beim Einschaltvorgang/Spannung

So kann sich das eine, etwas verstümmelte BIT zum Beispiel beim Auslisten eines Programmes auf einen externen Drucker als ein nichtgewünschtes Zeichen auswirken. (Benutzung des BASIC-Kommandos LIST ,,COMØ'').

Wenn Programme von einem HX-2Ø zu einem anderen HX-2Ø transferiert werden (A nutzt LIST ,,COMØ'', B nutzt LOAD ,,COMØ''), so kann diesem Problem durch entsprechende Gegenmaßnahmen vorgebeugt werden.

Gegenmaßnahmen

a) Benutzen Sie eine BIT-Rate, die langsam genug ist, dieses eine BIT zu ignorieren.

b) Versorgen Sie die Schnittstelle kurz vor der Kommunikation mit Strom, um den instabilen Moment zu vermeiden.
c) Stimmen Sie Sende- und Empfangseinheit aufeinander ab.

Folgende Reihenfolge sollte eingehalten werden, um die Schnittstelle vor der Übertragung mit Strom zu versorgen:

Sendeeinheit	Empfangseinheit
(1) OPEN "I", #1, "COMØ:(28N2B)" (Line power ON)	
(2) WIDTH "COMØ:",255	
	(3) LOAD"COMØ:(28N2B)" (Start programme reception)
(4) LIST "COMØ:(28N2B)" (Start programme transmission)	
(5) CLOSE #1	

Eine Prozedur, um Sender und Empfänger aufeinander abzustimmen, wird im folgenden Beispiel aufgeführt.

Die Sendeeinheit schickt das Zeichen „A" (zur Synchronisation), worauf die Empfangseinheit ebenfalls mit einem „A" antwortet, wenn sie das Synchronisationszeichen der Empfangseinheit empfangen hat.

Eröffnen für Datentransfer

Sendeeinheit	Empfangseinheit
10 OPEN "I",#1,"COMØ:(68N2B)"	10 OPEN "O",#2,"COMØ:(68N2B)"
20 OPEN "O",#2,"COMØ:(68N2B)"	20 OPEN "I",#1,"COMØ:(68N2B)"
30 PRINT #2,"A";	30 IF LOF(1)=0 THEN 30
40 FOR I=1 TO 300: NEXT I	40 A$=INPUT$(LOF(1),1)
50 IF LOF(1)=0 THEN 30	50 IF A$<>"A" THEN 30
60 A$=INPUT$ (LOF(1),1)	60 PRINT #2,"A";
70 IF A$<>"A" THEN 30	

Eine andere Methode ist, Sender und Empfänger untereinander auf den jeweiligen Status überprüfen zu lassen.

Dazu das folgende Beispiel:

Sendeeinheit	Empfangseinheit
10 OPEN "I",#1,"COM0:(68N2B)"	10 OPEN "O",#2,"COM0:(68N2B)"
20 A$=INPUT$(1) (Waits for key input)	20 A$=INPUT$(1) (Waits for key input)
30 OPEN "O",#2,"COM0:(68N2B)"	30 OPEN "I",#1,"COM0:(68N2B)"

Hier beginnt die Übertragung dann, wenn beide, Sender und Empfänger eine Taste betätigt haben.

Der Spannungswechsel der auftritt, wenn der Strom eingeschaltet wird, kann RTS (Request to Send) und DTR (Data Terminal Ready) in gleicher Weise beeinflussen.

Alle Operationen der SLAVE-CPU (Neben-CPU) werden mittels Kommandos von der MAIN-CPU (Haupt-CPU) kontrolliert.

Aus diesem Grund können manche Operationen nicht simultan ausgeführt werden.

So wird zum Beispiel ein Dateneingang auf der RS-232C-Schnittstelle durch einen Interrupt unterbrochen, wenn ein Datenausgang zu einem Drucker stattfindet.

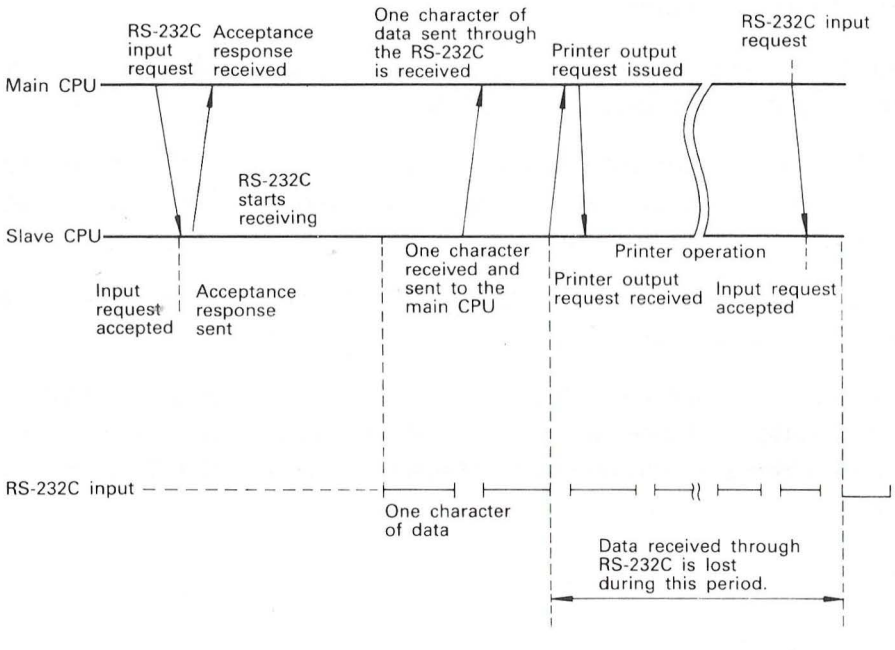


Abb. B-6 Verlust von Daten bei Empfang durch RS-232C

Simultane I/O-Operationen

Nr.	MAIN-CPU Interrupt		MAIN-CPU	SLAVE-CPU
1			MC-Input	
2			MC-Output	
3			Ext.Kass. Input	
4			Ext.Kass. Output	
5	Keyboard input interrupt	Batterie voltage interrupt etc.	L.C.D. Input	Lautsprecher Outp.
6			L.C.D. Output	Printer Output
7			L.C.D. Input	RS-232C Input
8			RS-232C Output	Lautsprecher Outp.
9			RS-232C Output	Printer Output
10			RS-232C Output	RS-232C Input
11			Hihg-speed-serial	Keine Operation

Simultan durchführbare Operationen sind in der vorstehenden Liste aufgeführt.

Die möglichen Kombinationen sind jeweils in der gleichen Zeile nebeneinanderstehend aufgeführt.

So sind zum Beispiel während eines Inputs von der Mikrokassette, unter Nummer 1 aufgeführt, beide CPU's besetzt.

Unter Nummer 11 der aufgeführten Liste wird gezeigt, daß die SLAVE-CPU während einer Operation über die High-speed-Serial-Schnittstelle keinerlei Operationen ausführt.

Bei dieser Kommunikation ist die MAIN-CPU beschäftigt und greift u.a. auf die SLAVE-CPU zurück.

Eine Ausnahme gibt es allerdings: Wenn die SLAVE-CPU eine Ausgabe auf den Sound-Lautsprecher von ca. 10 Sekunden startet, so kann die High-speed-Serial-Kommunikation starten, ohne daß der Ton unterbrochen wird.

ASCII-Code-Tabelle

USASCII

Hex. No.	Binary No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Hex. No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	128	144	160	176	192	208	224	240
1	0001	1 17	17	! 33	1 49	A 65	Q 81	a 97	q 113	129	145	161	177	193	209	225	241
2	0010	2 18	18	" 34	2 50	B 66	R 82	b 98	r 114	130	146	162	178	194	210	226	242
3	0011	3 19	19	# 35	3 51	C 67	S 83	c 99	s 115	131	147	163	179	195	211	227	243
4	0100	4 20	20	$ 36	4 52	D 68	T 84	d 100	t 116	132	148	164	180	196	212	228	244
5	0101	5 21	21	% 37	5 53	E 69	U 85	e 101	u 117	133	149	165	181	197	213	229	245
6	0110	6 22	22	& 38	6 54	F 70	V 86	f 102	v 118	134	150	166	182	198	214	230	246
7	0111	7 23	23	' 39	7 55	G 71	W 87	g 103	w 119	135	151	167	183	199	215	231	247
8	1000	8 24	24	(40	8 56	H 72	X 88	h 104	x 120	136	152	168	184	200	216	232	248
9	1001	9 25	25	) 41	9 57	I 73	Y 89	i 105	y 121	137	153	169	185	201	217	233	249
A	1010	10 26	26	* 42	: 58	J 74	Z 90	j 106	z 122	138	154	170	186	202	218	234	250
B	1011	11 27	27	+ 43	; 59	K 75	[91	k 107	{ 123	139	155	171	187	203	219	235	251
C	1100	12 28	28	, 44	< 60	L 76	\ 92	l 108	¦ 124	140	156	172	188	204	220	236	252
D	1101	13 29	29	- 45	= 61	M 77	] 93	m 109	} 125	141	157	173	189	205	221	237	253
E	1110	14 30	30	. 46	> 62	N 78	^ 94	n 110	~ 126	142	158	174	190	206	222	238	254
F	1111	15 31	31	/ 47	? 63	O 79	_ 95	o 111	127	143	159	175	191	207	223	239	255

ENGLAND

Hex. No.	Binary No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0 16	16	SP 32	0 48	@ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1 17	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2 18	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♣ 146	162	178	194	210	226	242
3	0011	3 19	19	£ 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♥ 147	163	179	195	211	227	243
4	0100	4 20	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♦ 148	164	180	196	212	228	244
5	0101	5 21	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6 22	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	ª 150	166	182	198	214	230	246
7	0111	7 23	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	▬ 151	167	183	199	215	231	247
8	1000	8 24	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	↑ 152	168	184	200	216	232	248
9	1001	9 25	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	↓ 153	169	185	201	217	233	249
A	1010	10 26	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	→ 154	170	186	202	218	234	250
B	1011	11 27	27	+ 43	; 59	K 75	[91	k 107	{ 123	▒ 139	← 155	171	187	203	219	235	251
C	1100	12 28	28	, 44	< 60	L 76	\ 92	l 108	| 124	■ 140	↔ 156	172	188	204	220	236	252
D	1101	13 29	29	- 45	= 61	M 77	] 93	m 109	} 125	▪ 141	× 157	173	189	205	221	237	253
E	1110	14 30	30	. 46	> 62	N 78	^ 94	n 110	~ 126	▬ 142	÷ 158	174	190	206	222	238	254
F	1111	15 31	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

FRANCE

Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	16	SP 32	0 48	à 64	P 80	' 96	p 112	+ 128	○ 144	160	176	192	208	224	240
1	0001	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♣ 145	161	177	193	209	225	241
2	0010	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♠ 146	162	178	194	210	226	242
3	0011	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♥ 148	164	180	196	212	228	244
5	0101	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	22	& 38	6 54	F 70	V 86	f 102	v 118	┼ 134	à 150	166	182	198	214	230	246
7	0111	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	┴ 151	167	183	199	215	231	247
8	1000	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	ê 152	168	184	200	216	232	248
9	1001	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	ù 153	169	185	201	217	233	249
A	1010	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	↑ 154	170	186	202	218	234	250
B	1011	27	+ 43	; 59	K 75	[91	k 107	é 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	28	, 44	< 60	L 76	¥ 92	l 108	ù 124	▌ 140	↓ 156	172	188	204	220	236	252
D	1101	29	- 45	= 61	M 77	] 93	m 109	è 125	▐ 141	× 157	173	189	205	221	237	253
E	1110	30	. 46	> 62	N 78	^ 94	n 110	¨ 126	▀ 142	÷ 158	174	190	206	222	238	254
F	1111	31	/ 47	? 63	O 79	_ 95	o 111	● 127	● 143	± 159	175	191	207	223	239	255

GERMANY

Hex. No.	Binary No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♣ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♥ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♦ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	ª 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	↑ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	↓ 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	→ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	← 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Ä 91	k 107	ä 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ö 92	l 108	ö 124	▌ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Ü 93	m 109	ü 125	▐ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	ß 126	▬ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

DENMARK

Hex. No.	Binary No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	╪ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♣ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♠ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♥ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	┼ 134	ª 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	╞ 135	┘ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	╔ 136	¤ 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	╗ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┌ 138	╝ 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Æ 91	k 107	æ 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ø 92	l 108	ø 124	▌ 140	→ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Å 93	m 109	å 125	▐ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	▀ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

SWEDEN

Hex. No.	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000		16	SP 32	0 48	@ 64	P 80	` 96	p 112	128	144	160	176	192	208	224	240
1	0001		17	! 33	1 49	A 65	Q 81	a 97	q 113	129	145	161	177	193	209	225	241
2	0010		18	" 34	2 50	B 66	R 82	b 98	r 114	130	146	162	178	194	210	226	242
3	0011		19	# 35	3 51	C 67	S 83	c 99	s 115	131	147	163	179	195	211	227	243
4	0100		20	¤ 36	4 52	D 68	T 84	d 100	t 116	132	148	164	180	196	212	228	244
5	0101		21	% 37	5 53	E 69	U 85	e 101	u 117	133	149	165	181	197	213	229	245
6	0110		22	& 38	6 54	F 70	V 86	f 102	v 118	134	150	166	182	198	214	230	246
7	0111		23	' 39	7 55	G 71	W 87	g 103	w 119	135	151	167	183	199	215	231	247
8	1000		24	(40	8 56	H 72	X 88	h 104	x 120	136	152	168	184	200	216	232	248
9	1001		25	) 41	9 57	I 73	Y 89	i 105	y 121	137	153	169	185	201	217	233	249
A	1010		26	* 42	: 58	J 74	Z 90	j 106	z 122	138	154	170	186	202	218	234	250
B	1011		27	+ 43	; 59	K 75	Ä 91	k 107	ä 123	139	155	171	187	203	219	235	251
C	1100		28	, 44	< 60	L 76	Ö 92	l 108	ö 124	140	156	172	188	204	220	236	252
D	1101		29	- 45	= 61	M 77	Å 93	m 109	å 125	141	157	173	189	205	221	237	253
E	1110		30	. 46	> 62	N 78	Ü 94	n 110	ü 126	142	158	174	190	206	222	238	254
F	1111		31	/ 47	? 63	O 79	_ 95	o 111	127	143	159	175	191	207	223	239	255

ITALY

Hex. No.	Binary No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	@ 64	P 80	ù 96	p 112	┼ 128	ö 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	à 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	↑ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	è 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	↓ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	→ 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	[91	k 107	ä 123	▒ 139	← 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	\ 92	l 108	ò 124	▉ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	] 93	m 109	è 125	▌ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	ì 126	▐ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	ß 127	● 143	± 159	175	191	207	223	239	255

SPAIN

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hex. No.																	
0	0000	0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♣ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♥ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♦ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	ª 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	º 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	¿ 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	⌐ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	¬ 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	[91	k 107	{ 123	▒ 139	½ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ñ 92	l 108	| 124	■ 140	¼ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	] 93	m 109	} 125	▪ 141	« 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	▬ 142	» 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	⌂ 127	● 143	± 159	175	191	207	223	239	255

ASCII-Code-Tabelle

Zeichen	Dezimal	Binär	Zeichen	Dezimal	Binär	Zeichen	Dezimal	Binär
NUL	0	0000000	+	43	0101011	V	86	1010110
SOH	1	0000001	,	44	0101100	W	87	1010111
STX	2	0000010	-	45	0101101	X	88	1011000
ETX	3	0000011	.	46	0101110	Y	89	1011001
EOT	4	0000100	/	47	0101111	Z	90	1011010
ENQ	5	0000101	0	48	0110000	[	91	1011011
ACK	6	0000110	1	49	0110001	\	92	1011100
BEL	7	0000111	2	50	0110010	]	93	1011101
BS	8	0001000	3	51	0110011	^	94	1011110
HT	9	0001001	4	52	0110100	_	95	1011111
LF	10	0001010	5	53	0110101	`	96	1100000
VT	11	0001011	6	54	0110110	a	97	1100001
FF	12	0001100	7	55	0110111	b	98	1100010
CR	13	0001101	8	56	0111000	c	99	1100011
SO	14	0001110	9	57	0111001	d	100	1100100
SI	15	0001111	:	58	0111010	e	101	1100101
DLE	16	0010000	;	59	0111011	f	102	1100110
DC1	17	0010001	<	60	0111100	g	103	1100111
DC2	18	0010010	=	61	0111101	h	104	1101000
DC3	19	0010011	>	62	0111110	i	105	1101001
DC4	20	0010100	?	63	0111111	j	106	1101010
NAK	21	0010101	@	64	1000000	k	107	1101011
SYN	22	0010110	A	65	1000001	l	108	1101100
ETB	23	0010111	B	66	1000010	m	109	1101101
CAN	24	0011000	C	67	1000011	n	110	1101110
EM	25	0011001	D	68	1000100	o	111	1101111
SUB	26	0011010	E	69	1000101	p	112	1110000
ESC	27	0011011	F	70	1000110	q	113	1110001
FS	28	0011100	G	71	1000111	r	114	1110010
GS	29	0011101	H	72	1001000	s	115	1110011
RS	30	0011110	I	73	1001001	t	116	1110100
US	31	0011111	J	74	1001010	u	117	1110101
SP	32	0100000	K	75	1001011	v	118	1110110
!	33	0100001	L	76	1001100	w	119	1110111
"	34	0100010	M	77	1001101	x	120	1111000
#	35	0100011	N	78	1001110	y	121	1111001
$	36	0100100	O	79	1001111	z	122	1111010
%	37	0100101	P	80	1010000	{	123	1111011
&	38	0100110	Q	81	1010001	|	124	1111100
'	39	0100111	R	82	1010010	}	125	1111101
(	40	0101000	S	83	1010011	~	126	1111110
)	41	0101001	T	84	1010100	DEL	127	1111111
*	42	0101010	U	85	1010101			

Programm zum Auflisten der ASCII-Codes vom HX-20

```
10 REM ***************
20 REM
30 REM ASCII - CODE
40 REM
50 REM ***************
60 REM
70 LPRINT "ASCII-CHARACT
ER CODES "
80 LPRINT " "
90 LPRINT " DEC      HEX
      CHR"
100 FOR I=32 TO 90
110 LPRINT TAB(1)(I);TAB
(12) HEX$(I);TAB(23)CHR$
(I)
120 NEXT I
130 FOR J=91 TO 159
140 LPRINT TAB(1)(J);TAB
(12)HEX$(J);TAB(23)CHR$(
J)
150 NEXT J
160 END
```

ASCII-CHARACTER CODES

DEC	HEX	CHR
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z

ASCII-CHARACTER CODES

DEC	HEX	CHR
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	¦
125	7D	}
126	7E	~
127	7F	\
128	80	┼
129	81	┴
130	82	┬
131	83	┤
132	84	├
133	85	─
134	86	│
135	87	┌
136	88	┐
137	89	└
138	8A	┘
139	8B	▒
140	8C	▉
141	8D	▊
142	8E	▋
143	8F	●
144	90	○
145	91	♠
146	92	♥
147	93	♦
148	94	♣
149	95	♪
150	96	♫
151	97	♭
152	98	♯
153	99	¥
154	9A	₧
155	9B	↑
156	9C	↓
157	9D	×
158	9E	÷
159	9F	±

FEHLERMELDUNGEN

Fehler Code	Fehler Nr.	Meldung	Bemerkung
/Ø	11	*Division by Zero* Division durch Null	Nicht erlaubte Division durch Null.
AO	52	*File already open* Datei bereits geöffnet	Einer bereits geöffneten Datei ist ein sequentieller Ausgabe-Modus OPEN eingesetzt worden. Oder ein KILL wurde für eine Datei angegeben, welche offen ist.
BD	58	*Bad Data In File* Falsches Daten-Format	Eine Anweisung oder ein Befehl mit einem falschen Format.
BF	51	*Bad File Mode* Falscher Datei-Modus	Es wurde versucht, PUT,GET,LOF mit einer seqentiellen Datei zu benutzen, um eine Random Datei zu laden, oder ein OPEN wurde mit einem anderen Dateimodus als I, O oder R benutzt.
	7Ø	*Bad File Structure* Falscher Datei-Aufbau	Die Datei-Struktur ist zerstört
BN	5Ø	*Bad File Number* Falsche Datei-Nummer	Eine Anweisung oder ein Befehl beziehen sich auf eine Datei mit einer Dateinummer, welche nicht eröffnet ist, oder sich außerhalb der Dateinummern befindet, welche bei der Initialisierung bestimmt wurden. (1 bis 16).
BO	61	*Buffer Overflow*	Überlauf im internen Buffer
	69	*Bad Record Number* Falsche Satz Nummer	Die Satznummer bei PUT oder GET war grösser als 32737 oder gleich Null.

Fehler Code	Fehler Nr.	Meldung	Bemerkung
BS	9	*Subscript out of range* Index außerhalb des Bereiches	Falsche Indexnummer, oder Index außerhalb der Felddimension. Die DIM-Anweisung ist zu groß.
CN	17	*Can't Continue* Fortsetzung nicht möglich	Versuch, ein Programm fortzusetzen nach: einem Fehler, nach einer Unterbrechung, oder wenn das Programm nicht existiert.
DD	10	*Duplicate Definition* Doppelte Definition	Doppelte DIM oder DEF-Anweisung im gleichen Programm sind nicht erlaubt.
	64	*Directory Full* Alle Dateien belegt	Es wurde versucht, mehr Dateien als möglich anzulegen.
	66	*Disk Full* Diskette voll	Der gesamte Speicherbereich der Diskette ist benutzt.
	71	*Drive not ready* Laufwerk nicht bereit	Keine Diskette im Laufwerk
DS	56	*Direct Statement in File* Direkte Anweisung in der Datei	Beim Laden einer Datei im ASCII-Format, wird eine direkte Anweisung festgestellt. Das Laden wird beendet.
DD	60	*Device Unavailable* Gerät nicht vorhanden	Ein nicht vorhandenes Gerät wurde angesprochen.
	72	*Disk Write Protected* Disketten Schreibschutz	Es wurde eine Anweisung auf eine schreibgeschützte Diskette gegeben.
	67	*File already exists* Datei bereits vorhanden	Der Dateiname, welcher in einer NAME-Anweisung bestimmt wurde, ist identisch zu einem bereits auf der Diskette verwendeten.

Fehler	Fehler Code	Meldung Nr.	Bemerkung
FC	5	Illegal Function Call Unerlaubter Funktions-Aufruf	Einer mathematischen oder einer Zeichenkettenfunktion wurde ein Parameter zugewiesen, welcher außerhalb des Bereiches liegt. Ein FC-Fehler kann auch als Ergebnis auftreten von: 1. negativem/zu langem Index 2. negativem Argument bei LOG 3. negativem Argument für SQR 4. negativer Mantisse mit einem nicht ganzzahligen Exponenten 5. Aufruf an eine USR-Funktion, ohne Startadresse 6. falsches Argument für MID$, LEFT$,RIGHT$,INP,OUT,WAIT, PEEK,POKE,TAB,SPC,STRING$, SPACE$,INSTR,ON....GOTO
FD	55	Bad File Descriptor Falscher Datei-Descriptor	Eine Datei mit einem falschen Gerätenamen, falschen Dateinamen, wurde aufgerufen.
FN	23	*FOR without NEXT* FOR ohne NEXT	Es fehlt die NEXT-Anweisung nach einem vorausgegangenen FOR.
	68	*Field Overflow* Bereichsüberschreitung	Eine FIELD-Anweisung versucht mehr Bytes als für die Aufzeichnung einer Random-Datei bestimmt sind, zuzuweisen.
ID	12	*Illegal Direct* Unerlaubter direkter Modus	Eine Anweisung, welche im direkten Modus unerlaubt ist, wurde als ein direkter Modus-Befehl eingegeben.
IE	54	*Input past end* Eingabe nach Ende	Eine INPUT-Anweisung erfolgt nach Ende der Datei. Mit EOF ist das Datei-Ende aufzufinden.

Fehler Code	Fehler Nr.	Meldung	Bemerkung
IO	53	Device I/O Error Geräte-I/O-Fehler	Kassetten Lese/Schreibfehler, RS-232C oder High Speed Serial mit Parity-Fehler.
IU	59	*Device in Use* Gerät in Betrieb	Mehr als eine OPEN-Anweisung für ein Gerät.
LS	15	*String too long* String zu lang	Die String-Anweisung ist zu lang.
MO	22	*Missing Operand* Fehlender Operand	Ein Ausdruck enthält einen Operator, auf welchen kein Operand folgt.
NE	63	*File not Found* Datei nicht vorhanden	Eine LOAD, KILL oder OPEN-Anweisung, bezieht sich auf eine nicht vorhandene Datei
NF	1	*NEXT without FOR* NEXT ohne FOR	Eine Variable in einer NEXT-Anweisung entspricht nicht einer vorher ausgeführten, unpaarigen FOR-Anweisungs-Variablen.
NO	57	*File not OPEN* Datei nicht geöffnet	Eine Datei-Anweisung, ohne vorausgegangenem OPEN.
NR	19	*No Resume* Kein Resume	Eine Fehlersuch-Routine wurde eingegeben, ohne daß sie eine RESUME-Anweisung enthält.
OD	4	*Out of Data* Außerhalb des Datenbereichs	Eine READ-Anweisung ohne Data-Anweisung.
OM	7	*Out of Memory* Außerhalb des Speicherbereichs	Das Programm ist zu groß, hat zu viele FOR-Schleifen oder GOSUBS, zu viele Variablen oder Ausdrücke die zu kompliziert sind.
OS	14	Out of String Space Außerhalb des String-Bereichs	Variablen von Zeichenketten überschreiten den zugewiesenen String-Bereich. Mit CLEAR kann man mehr Raum erzeugen, oder die Größe und Anzahl der Zeichenketten reduzieren.

Fehler Code	Fehler Nr.	Meldung	Bemerkung
OV	6	*Overflow* Bereichs-überschreitung	Das Ergebnis einer Berechnung ist zu groß, um im Zeichen-format von BASIC dargestellt zu werden. Bei Überschreitung ist das Ergebnis Null, die Ausführung wird ohne Fehlermeldung fortgesetzt.
PP	62	*Protected Program* geschütztes Programm	Mit der TITLE-Anweisung kann ein Programm geschützt werden.
	73	*Read Error*	Lesefehler
RG	3	*Return without Gosub* Return ohne Gosub	Es wurde eine Return-Anweisung aufgefunden, ohne vorangegangene un-paarige GOSUB-Anweisung.
RW	20	*Resume without Error* Resume ohne Fehler	Es wurde eine RESUME-Anweisung festgestellt, bevor eine Fehlersuch-Routine eingegeben wurde.
SN	2	*Syntax Error* Regel-Fehler	Es tritt eine Zeile auf, welche eine unrichtige Reihenfolge von Zeichen enthält, (z.B. unpaarige Klammern, falschen Befehl, oder Anweisung, oder fal-sche Punktierung).
ST	16	*String Formula* too complex String Formel zu kompliziert	Ein Zeichenkettenausdruck ist zu lang oder zu kompliziert. Der Ausdruck muß in kleinere Ausdrücke unterteilt werden.
	65	*Too many Open* Disk Files Zu viele OPEN-Dateien	Es wurde versucht, mit OPEN eine neue Datei zu erzeugen, obwohl alle 255 Abrufeingänge belegt sind.

Fehler Code	Fehler Nr.	Meldung	Bemerkung
TM	13	*Type Mismatch* Schreibfehler	Einem Stringvariablennamen wurde ein numerischer Wert zugeordnet, oder umgekehrt; einer Funktion, welche auf ein numerisches Argument wartet, wurde ein Stringargument zugewiesen oder umgekehrt.
UF	18	*Undefined User Function* Unbestimmte Anwenderfunktion	*Eine USR-Funktion wurde auf*-gerufen, bevor die Funktions-abstimmung (DEF-Anweisung) gegeben war.
UL	8	*Undefined Line Number* Unbestimmte Zeilen-Nummer	Eine Zeilenangabe in einem GOTO,GOSUB,IF....THEN....ELSE oder DELETE ist auf eine nicht existieren-de Zeile bezogen.
UP	21 26-49 75-255	*Unprintable Error* Nicht ausdruckbarer Fehler	Für die bestehende Fehler-bedingung ist keine Fehler-meldung vorhanden. Dies wird gewöhnlich durch einen FEHLER mit einem unbe-stimmten Fehlercode verursacht.
	74	*Write Error* Schreibfehler	Schreibfehler
WE	24	*WHILE without WEND* WHILE ohne WEND	Es wurde WHILE ohne WEND festgestellt.
WH	25	*WEND without WHILE* WEND ohne WHILE	Es wurde WEND ohne voraus ge-gangenes WHILE festgestellt.

Tabelle der reservierten Wörter

ABS
ALL
AND
ASC
ATN
AUTO
BASE
CDBL
CHR$
CINT
CLEAR
CLOSE
CLS
COLOR
CONT
COPY
COS
CSNG
CSRLIN
CVI,CVS,CVD
DATA
DATE
DAY
DEF
DEFDBL
DEFFIL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DSKF
DSK$
DSKØ$
ELSE
END
EOF
EQV
ERASE
ERL
ERR
ERROR
EXEC
EXP
FILES
FILNUM
FIX
FN
FOR
FRE
GCLS
GET
GO
HEX$
IF
IMP
INKEY$
INPUT
INSTR
INT
KEY
LEFT$
LEN
LET
LINE
LIST
LLIST
LOAD
LOAD?
LOADM
LOCATE
LOCATES
LOF
LOG
LOGIN
LPRINT
LSET
MEMSET
MERGE
MID$
MKI$, MKS$
MKD$
MOD
MON
MOTOR
NAME
NEW
NEXT
NOT
OCT$
OFF
ON
OPEN
OPTION
OR
PCOPY
PEEK
POINT
POKE
POS
PRESET
PRINT
PSET
PUT
RANDOMIZE
READ
REM
RENUM
RESET
RESTORE
RESUME
RETURN
RIGHT$
RND
RSET
RUN
SAVE
SAVEM
SCREEN
SCROLL
SGN
SIN
SOUND
SPACE$
SPC
SQR
STAT
STEP
STOP
STR$
STRING$
SUB
SWAP
SYSGEN
TAB
TAN
TAPCNT
THEN
TIME
TITLE
TO
TROFF
TRON
USING
USR
VAL
VARPTR
WEND
WHILE
WIDTH
WIND
XOR

EPSON

Technologie, die Zeichen setzt.

EPSON Deutschland GmbH · Zülpicher Strasse 6 · 4000 Düsseldorf 11 · Telefon (0211)56030

Printed in Japan 88.03-1.5