GENIE IIs

Monitor

Text-Editor

Assembler

TCS
COMPUTER GMBH

Da Sie sich vielleicht an den Umgang mit dem Genie II S ROM in der Grundversion des Genie II S schon gewöhnt haben, möchten wir Ihnen auch zusammen mit der Diskettenstation die Arbeit mit diesem System ermöglichen.
Da das Genie II S ROM aber nur Kassettenroutinen enthält und daher eine Ein-/ Ausgabe auf Diskette nur schwer möglich ist, ist auf der Genie-DOS Master Diskette eine Disk-Version des Programms mit dem Namen NAME/CMD abgespeichert.

In dieser Version sind einige Befehle gegenüber der Grundversion geändert worden.

Der B-Befehl (ursprünglich BASIC) wurde durch den E-Befehl (EXIT) erstetzt. Es wird nun kein BASIC-Warmstart mehr ausgeführt, sondern ein Rücksprung ins DOS (über Adr. 402DH).

Die Befehle Load, Save und View des Monitors fragen den Programmnamen gesondert ab (10 und 11 Bildschirmzeile). Der Name muß immer angegeben werden. Der blinkende Stern und die Anzeige der ein-/ ausgegebenen Bytes sind nun nicht mehr vorhanden.

Auch im Text-Editor mußten einige Änderungen vorgenommen werden. Die minimale Adresse für den Parameter ML ist 5300H, da das DOS, der I/O-Buffer und der Bereich für den Genie-DOS-FCB (File-Control-Block) nicht überschrieben werden dürfen. Aus diesem Grunde ist der verfügbare Platz im Text-Editor auf ca. 32 kByte begrenzt.

! Um eine Bearbeitung von reinen ASCII-Files zu erlauben, wurde der Text-Editor um den Befehl SA (Save-ASCII) erweitert. Ein solcher ASCII-File ist dann für andere Zwecke verwendbar. (z.B. LIST- oder PRINT-Befehl im DOS, LOAD- oder MERGE-Befehl im BASIC, andere Text-Systeme, andere Assembler usw.). Der L-Befehl unterscheidet selbständig zwischen reinen ASCII-Files und Text-Editor Files.

Bei den Befehlen P und PB des Text-Editors wird die auszudruckende Zeile auch im Bildschrim angezeigt und mit dem Zeichen '*' markiert.

Der Assembler speichert ein Programm nun ebenfalls auf Diskette ab. Dabei wird das übliche Maschinenprogramm-Format verwendet, d.h. Sie können ein so assembliertes Programm direkt vom DOS her starten.

Wenn Sie noch Maschinenprogramme auf Kassette haben, dann können Sie diese mit dem Programm VL/CMD auf Diskette kopieren. Um auf Kassette abgespeicherte Texte auf Diskette zu übertragen, verwenden Sie das Programm TCOPY/CMD auf der Genie-DOS Diskette.

Die Fehlermeldung TAPE ERROR wurde durch eine ausführliche Fehlermeldung ersetzt, die das DOS-Error-Overlay ab Adresse 4409H übernimmt. Sie erhalten also die vom DOS her bekannten Fehlermeldungen (z.B. Bauteil nicht erreichbar, Datei nicht in Inhaltsverzeichnis usw.).

Titel Seite

Hierzu ein Beispiel:
Ab Adresse 7000H stehe folgendes Programm.

```
7000 210970      LD    HL,7009H
7003 CD752B      CALL  2B75H
7006 C30070      JP    7000H
7009 2A2A2A20    DEFM  "*** "
700D 00          DEFB  0
```

Sie wollen dieses Programm nun ab Adresse 6000H laufen lassen. Also verschieben Sie es zunächst mit dem M-Befehl des Monitors von 7000H nach 6000H.
Geben Sie ein M 7000 700D 6000 (RETURN),
dann 0 6000 600D 7000 700D 6000 (RETURN).
Es werden alle Adressen zwischen 7000 und 700D die im Bereich von 6000 bis 600D stehen auf die Basis 6000 umgerechnet und im jeweiligen Befehl abgespeichert.
Hinterher sieht das Programm dann folgendermaßen aus:

```
6000 210960      LD    HL,6009H
6003 CD752B      CALL  2B75H
6006 C30060      JP    6000H
6009 2A2A2A20    DEFM  "*** "
600D 00          DEFB  0
```

P xxxx yyyy (RETURN)
PRINT
Druckt den Speicherbereich von xxxx bis yyyy disassembliert aus. Es wird eine Seitenformatierung mit den Parametern PL,RL und ZS des Text-Editors ausgeführt.

Q (xxxx) (RETURN)
DISASSEMBLER
Dieser Befehl dient zur disassemblierten Darstellung von Speicherinhalten (d.h. in Z-80 Mnemonics).
Wenn Sie nur Q eingeben, so wird der Disassembler ab Programmzähler (PC) ein- bzw. ausgeschaltet.
Wenn Sie Q xxxx eingeben, so wird der Speicherbereich ab Adresse xxxx dissassembliert dargestellt.
Die Tasten Pfeil-links und Pfeil-rechts disassemblieren den vorhergehenden bzw. nächsten Befehl.
Die Tasten Pfeil-oben und Pfeil-unten erniedrigen bzw. erhöhen die Adresse um eins (Dies kann z.B. nötig sein, wenn ein Programm fehlerhaft disassembliert wird und eine Korrektur der Anfangsadresse nötig ist).Die Tasten ';' und '-' disassemblieren die vorhergehende bzw. nächste Seite, d.h. acht Befehle vor oder zurück.
BREAK verläßt den Disassembler-Modus.
Der RST 08H Befehl ist normalerweise ein 1-Byte-Befehl, wird jedoch im Basic benutzt, um einen Syntax-Test mit dem folgenden Byte auszuführen.
Dies wurde in diesem Monitor berücksichtigt, um Fehler beim Disassemblieren und Relozieren zu vermeiden.

S (xxxx (yyyy (zzzz (NAME)))) (RETURN)
SAVE
Dieser Befehl dient dazu, einen Speicherbereich im SYSTEM-Format auf Kassette zu schreiben.
Dabei sind xxxx die Anfangsadresse, yyyy die Endadresse und zzzz die Einsprungadresse (Entrypoint).
Der Name kann aus maximal 6 Zeichen bestehen.
Wenn die Parameter nicht angegeben werden, so werden die in der vierten Bildschirmzeile angezeigten Werte angenommen. Sie können also ein Programm mit S (RETURN) sehr einfach kopieren oder mehrmals aufzeichnen.

T (RETURN)
TEXT-EDITOR
Dieser Befehl führt zum Aufruf des Text-Editors (s. Genie II S Text-Editor).

V (xxxx) (RETURN)
VIEW
Dieser Befehl dient zur Anzeige des Namens und der Ladebereiche eines unbekannten SYSTEM-Programmes. Das Band wird gelesen, aber nicht in den Speicher geladen. Ansonsten wirkt der V-Befehl wie der L-Befehl.

X (xxxx (yyyy)) (RETURN)
HEX RECHNEN
Dieser Befehl zeigt die Summe und Differenz der angegebenen Argumente in hexadezimaler und dezimaler Form an. Er ist z.B. bei der Berechnung eines Offsets für den L-Befehl sehr nützlich.

Y (dddd (eeee)) (RETURN)
RECHNEN
Dieser Befehl funktioniert im Prinzip wie der X-Befehl mit dem Unterschied, daß die Argumente mit vier verschieden Zahlen-Basen angegeben werden können. Als Zahlen-Basen sind zugelassen: B-Binär, O-Oktal, D-Dezimal oder H-Hexadezimal. Wenn keine Basis angegeben ist, wird dezimal angenommen. Beipsiel:
Y 12345D 01010101B (RETURN)

Z xxxx yyyy (zz) (RETURN)
ZERO
Dieser Befehl löscht den Speicherbereich von xxxx bis yyyy mit dem Wert zz. Wenn zz nicht angegeben ist, wird der Wert 0 angenommen.

Beachten Sie bei allen Befehlen, daß der Speicherbereich von 0000 bis FFFF nicht gelöscht oder überschrieben werden darf, da dort der Monitor und die anderen Programme liegen ! Wenn Sie diesen Bereich löschen oder überschreiben, so führt das normalerweise zur Zerstörung der Programme und Texte, die im Speicher stehen !

Wenn Sie gleichzeitig einen Text oder ein Assembler-Programm bearbeiten wollen, so vergewissern Sie sich, daß Sie den Bereich des Text-Buffers nicht überschreiben.

Sicherheitshalber sollten Sie den Text-Buffer vorher auf Kassette abspeichern.

Die genaue Aufteilung des Bildschirms und Ihre Bedeutung finden Sie auf der nun folgenden Seite erläutert.

```
Genie II S                                                    Monitor
M 6000 61FF 6400                                                   **
BC: 8820 DE: 4030 HL: D000 AF: C31E F: --HPN- IX: 4015 IY: 3000
BC' 0200 DE' 0134 HL' 3230 AF' 0040 F' -Z---- SP: 43E8 PC: D000
CFFF   00        NOP                          NAME:  VON:  BIS:  ENT:
D000   CDC901  -)CALL  01C9H                   TEST   4400  443F  4410
D003   21003D    LD    HL,3D00H               X:     Y:    X+Y:  X-Y:
D006   222040    LD    (4020H),HL             0001   FFFF  0000  0002
D009   219BD2    LD    HL,0D29BH              00001 65535 00000 00002
D00C   CDD7F3    CALL  0F3D7H
D00F   2AB140    LD    HL,(40B1H)
D012   ED5B5BD1  LD    DE,(0D15BH)
D016   DF        RST   18H                    Tape Error
D000   CDC9 0121 003D 2220:4021 9BD2 CDD7 F32A   ...!.=" `!ö.*...
D010   B140 ED5B 58D1 DFCA:61F1 2AFD 4011 0001   .`.ÄX.f.A.*.`...
D020   1922 58D1 22B1 4011:CEFF 1922 A040 22E8   ."X."."`...".`"..
D030   40C3 61F1 5FB7 0E47:656E 6965 2049 4920   `.A....Genie II
```

So könnte eine Bildschirmdarstellung bei Benutzung des Monitors zum Beispiel aussehen.
Bitte beachten Sie nicht die hier im Beispiel angezeigten Werte, da diese zufällig gewählt und nicht von Bedeutung sind.

Jedoch ist so am besten die Bildschirmaufteilung zu erkennen.

Die erste Bildschirmzeile ist die Kommandozeile.
Der Cursor markiert hier die Eingabeposition.

Die beiden folgenden Zeilen zeigen die Werte der zwischengespeicherten Registerinhalte an. Diese werden beim G-Befehl in die Register übertragen und bei einem Rücksprung über einen Breakpoint oder ein RET zurückgegeben und angezeigt.

In der vierten bis zwölften Zeile ist im linken Teil das Disassembler-Listing, wenn der Disassembler eingeschaltet ist, ab der augenblicklichen Adresse von PC zu sehen.
Die PC-Adresse selbst ist dann mit einem Pfeil (->) markiert.
Im Disassembler-Modus ist kein Pfeil zu sehen, da immer die erste disassemblierte Zeile die aktuelle Adresse anzeigt.

Im rechten Teil sind in den Zeilen vier und fünf die Werte für den Save-, Load- und View-Befehl zu sehen.

Darunter sind die Ergebnisse der Rechen-Befehle X und Y in dezimaler und hexadezimaler Form angezeigt.

In der zwölften Zeile rechts, werden Fehlermeldungen angezeigt (Parameter Error oder Tape Error).

In den letzten vier Zeilen sind 64 Bytes des Speichers in hexadezimaler- und in ASCII-Form angezeigt.

Wenn einmal der Bildschirm durch einen Programmaufruf 'unsauber' sein sollte, so drücken Sie einfach (CLEAR) und Sie sehen wieder klarer.

Dieses Bild gibt nur einen möglichen Zustand des Bildschirms wieder. Das Disassmbler-Listing , die Fehlermeldung und die Ergebnisse der X- und Y-Befehle müssen nicht immer angezeigt werden.

Wie nützlich ein ROM-residenter Monitor beim programmieren in Maschinensprache ist, werden Sie leicht feststellen können.

2. Die Tastaturbelegung im Text-Editor

2.1 Die Zeichentasten

Alle mit Buchstaben, Zahlen oder Sonderzeichen beschrifteten Tasten, erzeugen die aufgedruckten Zeichen. Jedoch sind nicht alle Zeichen auf den Drucker auszugeben.
Die Zeichen '^' und '_' dienen als Steuerbefehle für den Drucker.
Das Zeichen '^' erzeugt beim Ausdruck einen Escape-Code, der bei den meisten Druckern zur Steuerung von verschiedenen Drucker-Funtkionen verwendet wird.
Das Zeichen '_' dient zur Steuerung der Unterstreichung bei EPSON kompatiblen Druckern (dazu gehören STAR, GEMINI usw.).
Beim Drucken wird für ein '_'-Zeichen die Bytefolge
1B 2D 01 bzw. 1B 2D 00 ausgegeben (enstpr.: ESC - 1, ESC - 0).

Wenn Sie nun die Zeichen selbst ausgeben wollen, so benutzen Sie den Steuerbyte-Befehl (P1) G um die Werte einzugeben.
In dieser Anleitung, die auch mit dem Genie II S Text-Editor erstellt worden ist, wurden die Zeichen allerdings nachgemalt da der Typenrad-Drucker sie nicht im Zeichensatz hatte.
Beachten Sie, daß diese Steuer-Zeichen zwar im Bildschirm ein Zeichen belegen, aber nicht beim Ausdrucken.

2.2 Zusammenfassung der Funktionstasten

Die Pfeiltasten haben allein gedrückt die aufgedruckte Funktion.
Also Pfeil-oben bewegt den Cursor in die vorhergehende Zeile, Pfeil-unten in die folgende usw.
Die (CLEAR)-Taste hat die Funktion HOME-Cursor, d.h. sie bewegt den Cursor in die linke, obere Bildschirmecke.
Die (BREAK)-Taste verzweigt, wie auch (P1) A, zur Kommando-Eingabe, die später noch erläutert wird.

Zusätzlich zu den normalen Tastenbelegungen sind eine Reihe von Funktionen über die (P1)-Taste erreichbar. Sie drücken zuerst die (P1))-Taste und dann die angegebene Taste, um die entsprechende Funktion aufzurufen.
Die Funktionen sind:

A - Kommando-Eingabe aufrufen bzw. abbrechen.
B - Block-Anfangsmarker an die Cursorposition.
C - festgelegten Block kopieren.
D - Zeichen an der Cursorposition löschen.
E - Block-Endmarker an die Cursorposition.
F - Find-Kommando wiederholen.
G - Steuerbyte Eingabe.
I - Leerzeichen an der Cursorposition einfügen.
L - festgelegten Block löschen.
N - Eine Seite vorwärts.
O - Eine Seite zurück.
P - System-Parameter anzeigen.
Q - Systemvariablen wiederherstellen.
R - Replace-Kommando wiederholen.
S - Insert-Modus ein-/ausschalten.
T - Leerzeichen ab Cursorposition bis nächstes Zeichen löschen.
U - Umbruch-Modus ein-/ausschalten.
V - festgelegten Block verschieben.
W - vorhergehende Textzeile wiederholen.
X - Zeile ab Cursorposition bis (CR) löschen.
Y - Block-Marker löschen.
Z - Block-Zielmarker an die Cursorposition.

zu c)
Normalerweise, d.h. beim ersten Aufruf des Text-Editors, erscheint beim Druck auf eine Buchstaben-Taste der entsprechende Kleinbuchstabe. Die Großbuchstaben sind über die SHIFT-Tasten zu erreichen. Für die normale Texterstellung ist dies nützlich, da Sie es wahrscheinlich von der Schreibmaschine her so kennen.
Sie können jedoch die Funktion der Schift-Taste im Bezug auf die Buchstaben-Tasten umkehren.
Dies geschieht mittels einmaligem Drücken von (SHIFT) 0.
Danach erscheint beim Druck einer Buchstaben-Taste der Großbuchstabe und die (SHIFT)-Taste erzeugt die Kleinbuchstaben.

zu d)
Die INSERT-Funktion wird mit (P1) S ein bzw. ausgeschaltet.
Der Insert-Modus wird mit dem '*' in der Statuszeile und durch einen größeren Cursor im Text-Fenster dargestellt.
Die Insert-Funktion bewirkt, daß ein nachfolgender Text beim Schreiben nicht überschrieben sondern weitergeschoben wird.

zu e)
Die UMBRUCH-Funktion wird mit (P1) U ein und ausgeschaltet.
Der Umbruch-Modus ist bei der Texterstellung meistens nützlich.
Die Funktion bewirkt, daß ein Wort das über den rechten Rand der Cursorzeile geschrieben würde, in die nächste Zeile übernommen wird.
Wenn Sie allerdings im INSERT-Modus arbeiten, so kann es sein, daß die nächste Zeile auch "voll" ist und dort können überstehende Worte dann nicht in die folgende Zeile übernommen werden, da sonst der ganze folgende Text verschoben würde.

Beim Einfügen sollten Sie also den UMBRUCH-Modus ausschalten.
Meistens werden Sie jedoch einen Text "von oben nach unten" schreiben und dann können Sie den automatischen Umbruch sicher gebrauchen.

zu f)
Der AUTO-REPEAT ist beim ersten Aufruf des Text-Editors immer eingeschaltet. Er bewirkt die automatische Wiederholung einer längere Zeit festgehaltenen Taste. Die Wiederhol-Funktion ist mit der Taste (P2) ein-/ausschaltbar.

1.2 Die Tabulatorzeile

Die Tabulatorzeile zeigt an jeder achten Bildschirmspalte ein Kreuz. Diese Kreuze zeigen die jeweils nächste Cursorposition an, die bei der Eingabe von (SHIFT) (SPACE) erreicht wird.
Der Tabulator wird noch näher erläutert ("2.7 Cursorsteuerung").
Wenn bei der Kommando-Eingabe oder im Assembler Fehler auftreten, so werden diese in der Tabulatorzeile angezeigt bis Sie eine Taste drücken.

1.3 Das Textfenster

Die unteren vierzehn der sechzehn Bildschirmzeilen sind ein Fenster auf den Textspeicher. Der Cursor bleibt immer innerhalb dieses Fensters (außer bei der Kommando-Eingabe).
Wenn Sie mit dem Cursor dieses Fenster nach oben oder unten verlassen wollen, dann wird das Fenster über den nächsten Textausschnitt verschoben. Wenn Sie in der ersten Zeile die Taste Pfeil-oben drücken, dann bleibt der Cursor allerdings in der ersten Textzeile, erscheint aber auf dem ersten Zeichen.

Der CL-Befehl hat sehr schwerwiegende Folgen, und wurde aus diesem Grund nur als Kommando vorgesehen.
Er löscht den Textspeicher ab der momentanen Cursorposition bis zum Text-Ende. Wenn Sie also den kompletten Textspeicher löschen wollen, so geben Sie (SHIFT) ↑ und dann (BREAK) CL (RETURN) ein.

Der CF-Befehl ändert die Blinkfrequenz des Cursors. Mit dem eingegebenen Wert wird der Wert einer Zählschleife mit logisch AND verknüpft. Wenn das Ergebnis ungleich Null ist, dann ist das Cursorzeichen, sonst das Textzeichen zu sehen.
Das heißt bei CF=0 ist kein Cursor zu sehen und bei CF=255 ist das Cursorzeichen immer eingeschaltet.

Der CC-Befehl ändert den ASCII-Wert des Zeichens, das als Cursor verwendet werden soll. Sie können z.B. auch 5FH , 7FH oder 8CH usw. verwenden.

Der DL-Befehl ändert den Wert für eine Zählschleife in der Tastaturabfrage. Er bestimmt die Zeit, die vom Niederdrücken einer Taste bis zum Beginn des AUTO-REPEAT vergeht.
Der Wert 34H ist ein meistens angenehmer Wert und wurde deshalb als Vorbelegung gewählt.

Der F-Befehl sucht ab der derzeitigen Cursorposition nach einem angegebenen Text. Der Text wird nach einem Trennzeichen hinter dem F (z.B. Leerzeichen) eingegeben und mit (RETURN) abgeschlossen. Ein (CR) können Sie mit (SHIFT) (RETURN) und einen Tabulator mit (SHIFT) (SPACE) eingeben.
Wenn der String gefunden wird, so erscheint der Cursor auf dem ersten Zeichen des Strings im Textspeicher.
Andernfalls bleibt er an der Position die er vorher hatte.
F(RETURN) oder (P1) F wiederholen den Befehl mit demselben String ab der neuen Cursorposition.

Der K-Befehl belegt eine der Tasten (P1) 0 bis (P1) 9 mit einem neuen Text. Ein (CR) können Sie mit (SHIFT) (CR) und einen Tabulator mit (SHIFT) (SPACE) eingeben.
Die Belegung der anderen Text-Tasten wird entsprechend verschoben. Wenn der Text zu lang für die 256 reservierten Bytes ist, dann wird die vorherige Belegung beibehalten.
Der Befehl dient zur Vereinfachung der Eingabe von häufig auftretenden Worten oder Phrasen.

Der L-Befehl lädt den nächsten auf Kassette abgespeicherten Text in den Textspeicher. Dabei bleibt der alte Text erhalten und der neue wird hintenan geladen.
Auf diese Weise können Sie meherere Texte zu einem Stück zusammenfügen und komplett bearbeiten.
Wenn der Text nicht innerhalb der Textbuffergrenzen unterzubringen ist, so wird der Ladevorgang abgebrochen und der Text an der zuletzt eingelesenen Position beendet.
Wenn Sie einen neuen Text laden wollen, dann geben Sie vorher (SHIFT) ↑ und (BREAK) CL (RETURN) ein.
Die Belegung der Text-Tasten wird ebenfalls eingeladen.

Der M-Befehl führt zum Aufruf des Monitors. Sie können dort z.B. ein gerade assembliertes Programm ausführen oder ändern und andere Programme laden oder abspeichern.
Wenn Sie den Textspeicher dabei nicht löschen, dann ist der Text bei der Rückkehr mit dem T-Befehl des Monitors noch erhalten.

↑ - Eine Seite zurück.
↓ - Eine Seite vorwärts.
0 - Text: "Meine sehr geehrten Damen und Herren,"(CR)(CR)
1 - Text: "Sehr geehrte Frau "
2 - Text: "Sehr geehrter Herr "
3 - Text: "Herr/Frau/Firma"(CR)(CR)
4 - Text: "Vielen Dank für "
5 - Text: "   mit freundlichen Grüßen"(CR)(CR)
          "   "
6 - Text: "Absender :"(CR)
7 - Text: "Bankverbindung "
8 - Text: "Postfach "
9 - Text: " den 01.01.1984 "

Außerdem gibt es noch einige Funktionen zusammen
mit der (SHIFT)-Taste:

↑ - Cursorposition = erste Textzeile.
↓ - Cursorposition = letzte Textzeile.
← Cursorposition = erstes Zeichen des vorherigen Wortes.
→ Cursorposition = erstes Zeichen des folgenden Wortes.

2.3 Befehle der Kommando-Eingabe

Die (BREAK)-Taste oder (P1) A führen zu einem Aufruf der
Kommando-Eingabe.
Dabei erscheint in der Statuszeile folgendes:

KOMMANDO:.....

Hier können Sie nun verschieden Kommandos ausführen, die
Argumente benötigen oder sicherheitshalber in die Kommandozeile
gelegt wurden.
Bei den Befehlserklärungen bedeutet:

n	ein Byte (z.B. 127, 127D, 7FH, 01111111B oder 177O)
ww	ein Wort (z.B. 6000H, 24576, 24576D usw.)
b	eine Ziffer zwischen 0 und 9.
xxxx	ein String (Zeichenkette) aus Buchstaben, Zahlen oder
yyyy	Sonderzeichen.
)	ein beliebiges Trennungszeichen.

Die Befehle sind:

A, AL od. AP	Assembler aufrufen (s. Genie II S Assembler)
CL	Textspeicher ab Cursorposition löschen.
CF=n	Blinkfrequenz des Cursors ändern.
CC=n	Cursorzeichen ändern.
DL=n	Delay-Wert für Tastatur ändern.
F)xxxx	String ab Cursorposition suchen.
Kb=xxxx	Text-Taste b mit String xxxx belegen.
L	Text von Kassette laden.
M	Monitor aufrufen.
ML=ww	Memory low (untere Speichergrenze) ändern.
MH=ww	Memory high (obere Speichergrenze) ändern.
P	Textspeicher komplett ausdrucken.
PB	festgelegten Block ausdrucken.
PL=n	Papierlänge in Zeilen ändern.
R)xxxx)yyyy	String xxxx durch yyyy ersetzen.
RL=n	Linken Rand für Druckerausgabe ändern.
RP=n	Tastatur Repeat-Frequenz festlegen.
S)(NAME)	Textspeicher auf Kassette abspeichern.
WT=ww	Wait-Zähler für Tastaturabfrage ändern.

Sie laden diesen Brief dann nur von Kassette und geben z.B. ein:
(BREAK)
KOMMANDO:R/$VNAME$/Franz(CR)
und drücken dann sooft wie nötig (P1) R.
Anschließend drücken Sie (SHIFT) ↑ und wieder
(BREAK)
KOMMANDO:R/$NNAME$/Müller(CR)
und entsprechend verfahren Sie mit den anderen **Platzhaltern.**

Diese Methode lohnt sich sicher nur bei längeren **Texten oder** in Assembler Programmen (z.B. Labels umbenennen).

Der R-Befehl dient auch zum **löschen** bestimmter Worte oder Sätze im Text. Dazu geben Sie einfach als zweiten String (hinter dem Trennzeichen) einen Leerstring also nur (CR) ein.

Aber Sie können auch einen Leerstring durch etwas ersetzen, **was** zunächst seltsam erscheinen mag und im Prinzip auch ist.
Sie wollen eine Textstelle z.B. so schreiben:

A U S E I N A N D E R

Sie geben dann einfach den Text "AUSEINANDER" ein und drücken (SHIFT) ← . Dann drücken Sie (BREAK) und R// (CR). Sie setzen also jeweils ein Leerzeichen an die Cursorposition.
(Denn ein Leerstring ist an jeder Textstelle vorhanden !).
Mit den Tasten (P1) R können Sie jetzt das ganze Wort mit Zwischenräumen versehen.

Diese Anwendungen sind nur einige Beispiele für **die vielen** Möglichkeiten, die der Replace-Befehl bietet.

Der RL-Befehl legt die Anzahl der Leerzeichen für den linken Rand bei der Druckerausgabe fest. Der voreingestellte Wert von acht ergibt bei einem Drucker mit 80 Zeichen/Zeile ein mittiges Schriftbild. Wenn Sie einen breiteren Drucker haben, oder an einer bestimmten Stelle des Papiers beginnen wollen, so können Sie den Wert zwischen 0 und 255 frei wählen.

Der RP-Befehl ändert die Repeat-Frequenz des AUTO-REPEAT. Das ist die Zeit die zwischen zwei Wiederholungen einer Taste vergehen soll.

Der S-Befehl speichert den Text aus dem Textbuffer auf Kassette ab. Dabei wird auch die Belegung der Text-Tasten mit abgespeichert. Wenn Sie einen Namen angeben, so darf dieser maximal 16 Zeichen lang sein. Wenn Sie keinen Namen angeben, so wird der letzte benutzte Name verwendet.
Die Abspeicherung beginnt direkt nachdem Sie (RETURN) drücken. Schalten Sie also den Rekorder vorher auf RECORD & PLAY, warten Sie den Bandleerlauf ab und geben Sie dann erst (RETURN) ein.

Der WT-Befehl ändert den Wert für die Zählschleife zur Tastaturentprellung. Wenn Sie wieder erwarten Probleme mit Tastaturprellen haben, so können Sie den Wert entsprechend nach oben oder nach unten anpassen.
Sie sollten den Wert nicht zu klein wählen, da Sie u.U. sonst keine sinnvolle Eingabe mehr machen können. In einem solchen Falle hilft die Funktion (P1) Q die alten Werte wieder herzustellen.

Der ML-Befehl erlaubt es, die untere Grenze des Textbuffers zu ändern. Ein dabei schon vorhandener Text bleibt erhalten und wird entsprechend verschoben. Die **niedrigste**, zulässige Adresse ist **42E8H**. Die **höchste** zulässige Adresse ist **MH -1**. Wenn der schon vorhandene Text nicnt in die neuen Speicher-Grenzen passen würde, so bleibt der alte ML-Wert erhalten und es wird eine Fehlermeldung ausgegeben.

Der MH-Befehl legt die obere Grenze des Textbuffers fest. **Maximale** Adresse ist **CFFFH**. Sie können so z.B. einen Bereich für Daten oder ein anderes Programm freihalten, das nicht von dem eingegebenen Text überschrieben werden soll. MH kann selbstverständlich **minimal ML +1** sein.

Der P-Befehl druckt den Inhalt des Textbuffers aus. Dabei wird ein linker Rand wie mit RL angegeben gelassen. Die Papierlänge PL und die Anzahl der Zeile pro Seite ZS werden zur Formatierung des Ausdrucks verwendet. Wenn Sie die Seiten-formatierung nicht benötigen oder dem Drucker überlassen wollen, so setzen Sie ZS auf denselben Wert wie PL und RL auf Null.

Der PB-Befehl druckt den Text innerhalb des angegebenen Blockes aus. Auch hierbei wird ein linker Rand gelassen und die Seitenformatierung wird vorgenommen.

Der PL-Befehl legt die Anzahl der Zeilen fest, die eine Seite Druckerpapier hat. (Bei 1/6 inch und Traktorpapier sind dies meistens 72 Zeilen). Wenn Sie anderes Papier benutzen oder einen andere Einstellung des Druckers haben (s. Druckerhandbuch), dann können Sie den Wert entsprechend ändern.

Der R-Befehl gestattet es, einen String im Textbuffer durch einen anderen zu ersetzen. Die Strings müssen dabei nicht die-selbe Länge haben und können auch leer sein (vgl. Leerstrings im BASIC). Das Trennungszeichen zwischen R und dem ersten Textstring dient dann als Endmarkierung des ersten Textstrings. Das Zeichen ist beliebig, darf jedoch im ersten String nicht nocheinmal auftauchen.
Sie können mit diesem Befehl z.B. einen Standarbrief von Kassette für die jeweilige Anwendung modifizieren.
Hierzu ein Beipsiel:

```
Herr/Frau/Firma                          Ungenannt & Co.
$VNAME$ $NNAME$                          Waldweg 32
$STRASSE$ $NR$                           5432 Nirgendwo
$PLZ$ $ORT$                              Tel.: 1234/23456

Sehr geehrter Herr $NNAME$

Wir freuen uns Ihnen mitteilen zu können, daß Sie in
unserem Preisausschreiben vom $DATUM$ gewonnen haben.
Wir hoffen, daß Ihnen ihr $GEWINN$ gefallen wird.

                    Mit freundlichen Grüßen

                    Ungenannt & Co.
```

2.6 Die Parameter- und Modus-Funktionen

Die Funktion (P1) P zeigt alle änderbaren System-Variablen auf dem Bildschirm an. Mit einem beliebigen Tastendruck kehren Sie zur normalen Eingabe zurück.

Es bietet sich folgendes Bild:

```
ML= 43EBH
MH= CFFFH
PL= 072
ZS= 066
RL= 008
DL= 34H
RP= 06H
WT= 0500H
CF= 10H
CC= 8FH
```

Die Abkürzungen vor den Parametern sind dieselben, wie sie in der Kommandoeingabe verwendet werden. Sie können mit (P1) P also jederzeit die Werte der Systemvariablen anzeigen lassen.

Die Funktion (P1) Q setzt folgende Parameter auf ihren Anfangswert zurück:

PL=72 ZS=66 RL=8 DL=34H RP=06H WT=500H CF=20H CC=8FH

Sie ist für den Notfall gedacht, wenn Sie einmal zuviele Parameter verändert haben sollten.

2.7 Die Steuerbyte Funktion

Die Funktion (P1) G dient zur Eingabe von Steuerzeichen oder auf dem Bildschirm nicht darstellbaren Zeichen für den Drucker.
Wenn Sie (P1) G gedrückt haben, wird ein Grafikzeichen 'E' gefolgt von zwei Nullen an der Cursorposition eingefügt. Der Cursor blinkt über der linken Null und fordert Sie zur Eingabe einer zweistelligen Hexadzimalzahl auf (0..9,A..F).

Die drei Zeichen im Bildschirm werden bei der Cursorsteuerung wie ein Zeichen behandelt. Deswegen kann in einer Zeile mit einem oder mehreren Steuerbytes, die nicht mit einem (CR) endet die Formatierung beim Drucken mißlingen. Sorgen Sie also immer dafür, daß eine solche Zeile mit einem (CR) abschließt.

Wenn Sie das Steuerbyte mit einer Taste überschreiben, so wird der ASCII-Wert der Taste in dem Byte abgelegt.
Wenn Sie das Steuerbyte löschen, dann verschwinden alle drei Zeichen, also das Grafikzeichen und die beiden Hex-Ziffern.

Mit Hilfe dieser Funktion können Sie alle Werte zwischen 00H und FFH an ihren Drucker schicken und diesen steuern.

2.4 Die Blockmodus-Funktionen

Die Funktionen (P1) B, (P1) C, (P1) E, (P1) L,
(P1) V, (P1) Y und (P1) Z sind Blockmodusfunktionen.
Die Funktion (P1) B legt den Beginn eines Blockes fest.
Die Stelle wird im Text mit '[redacted]' markiert und das Textzeichen
verschwindet "hinter" der Markierung.
Die Funktion (P1) E legt das Ende eines Blockes fest
Die Stelle wird im Text mit '[redacted]' markiert.
Die Funktion (P1) Z legt das Ziel eines Blockes fest.
Die Stelle wird im Text mit '[redacted]' markiert.

Es ist verständlich, daß der Blcokbeginn vor dem Blockende liegen muß, und daß das Ziel nur außerhalb des Blockes sein kann. Alle anderweitigen Eingaben werden daher von vornherein nicht akzeptiert.

Die Funktion (P1) C kopiert den Block von einschließlich der Stelle '[redacted]' bis ausschließlich der Stelle '[redacted]' an die mit '[redacted]' markierte Stelle. Die Marker bleiben erhalten, um den Befehl wiederholen zu können.

Die Funktion (P1) L löscht den Block zwischen '[redacted]' und '[redacted]'. Der Cursor erscheint anschließend an der Position des alten Blockanfangs.

Die Funktion (P1) V verschiebt den Blcok zwischen '[redacted]' und '[redacted]' an die mit '[redacted]' markierte Stelle. Sie führt also die Funktionen (P1) C und (P1) L nacheinander aus.

Die Funktion (P1) Y löscht die gesetzten Blockmarker. Sie können so z.B. nach (P1) C die Marker wieder löschen.

2.5 Die Funktionen für Löschen und Einfügen

Die Funktion (P1) D löscht ein Zeichen an der Cursorposition. Der Rest der Zeile wird 'herangezogen'.

Die Funktion (P1) I fügt ein Leerzeichen an der Cursorposition ein. Der Rest der Zeile wird 'weitergeschoben'.

Die Funktion (P1) T löscht alle Leerzeichen ab der Cursorposition bis zum nächsten Zeichen oder bis zum (CR).

Die Funktion (P1) W wiederholt die vor der Cursorzeile stehende Textzeile und fügt sie vor der Cursorzeile ein.

Die Funktion (P1) X löscht die Zeile ab der Cursorposition bis einschließlich zum nächsten (CR).

Die (RETURN)-Taste fügt auf jeden Fall ein (CR) ein. Das heißt ein Buchstabe wird beim drücken von (RETURN) nicht überschrieben sondern an den Anfang der nächsten Zeile genommen.

3. Die Speicherbelgung des Text-Editors

3.1 Das Textformat im Speicher

Der Text ist generell im ASCII-Format im Speicher abgelegt.
Das heißt ein 'A' entspricht einem 41H oder 65D im Speicher.
Es gibt jedoch einige Ausnahmen.

Die unter 2.6 schon angesprochenen Steuerbytes sind in der Form B7H, n abgespeichert. n ist dabei das angegebene Byte.
Dies ist zur Unterscheidung der Steuerbytes von internen Steuerzeichen, wie (CR) (TAB) usw., notwendig.
Das Byte 09H dient als Tabulator-Zeichen.
Bei der Bildschirmausgabe oder beim Drucken werden hierfür ein bis acht Leerzeichen ausgegeben.
Das Byte 0DH (Carriage Return) dient als Endmarkierung einer logische Zeile. Deren Länge kann größer als eine Bildschirmzeile sein, wenn sie aus mehr als 64 Zeichen besteht oder z.B. Steuerbytes oder Tabulatoren enthält.
Das Byte 03H (entspr. ETX = End Of Text) dient als Markierung des Text-Endes. Es ist das Byte, das bei Kassettenoperationen das Ende des zu ladenden Textes markiert.
Alle anderen Zeichen, die die über Tastatur erreichbar sind, sind mit ihren ASCII-Werten im Speicher abgelegt.

3.2 Das Textformat auf Kassette

Ein Text-File auf Kassette beginnt, wie auch SYSTEM- und CLOAD-Bänder, mit einem Block von 255 mal 00H. Danach folgt ein Synchronisationsbyte mit dem Wert A5H.
Bei SYSTEM-Programmen würde jetzt ein 'U' = 55H folgen und bei BASIC-Programmen dreimal D3H.
Zur Unterscheidung von diesen Bändern beginnt ein Text-File mit 'T' = 54H.

Wenn dieses Byte nicht zu Beginn gefunden wird, wird eine Fehlermeldung angezeigt.
Danach folgt der Name des Textes mit 16 Zeichen im ASCII-Format.
Direkt anschließend sind 256 Bytes abgespeichert, die in den Bereich der Texte für die Text-Tasten geladen werden.
Nun folgt der eigentliche Text, wie er im Speicher stand.
Wenn ein Byte 03H gefunden wird, dem kein Byte B7H vorausging, ist das Textende erreicht.

Wenn Sie während des Ladevorgangs (BREAK) drücken, so wird der Text an der gerade geladenen Stelle beendet und der Ladevorgang abgebrochen. Diese Funktion erlaubt es, wenn auch ein wenig umständlich, nur einen Teil eines Textes zu laden.
Sie ist eigentlich dafür vorgesehen, den Ladevorgang abzubrechen, falls ein falsches Band eingelegt oder ein anderer Fehler begangen wurde.

Als letztes Byte auf der Kassette folgt eine 1-Byte Prüfsumme.
Sie wird durch Addition aller Text-Zeichen mit Ausnahme der Texte für die Text-Tasten und der 03H am Schluß gebildet.
Wenn die beim Laden errechnete und die zum Schluß geladene Prüfsumme nicht übereinstimmen, wird eine Fehlermeldung ausgegeben. Sie müssen dann versuchen, entweder den oder die Fehler zu finden oder, wenn zu viele Fehler vorhanden sind, den Text im Speicher zu löschen und mit einer anderen Lautstärke Einstellung noch einmal zu laden.

2.7 Cursorsteuerung

Der Cursor wird beim ersten Aufruf des Text-Editors durch das Grafikzeichen ' ' dargestellt.
Der Cursor kann mit folgenden Tasten bewegt werden:

←	- Ein Zeichen nach links.
→	- Ein Zeichen nach rechts.
↑	- Eine Zeile zurück.
↓	- Eine Zeile vorwärts.
(SHIFT) ←	- Ein Wort zurück.
(SHIFT) →	- Ein Wort vor.
(P1) ↑ / (P1) O	- Eine Seite zurück.
(P1) ↓ / (P1) N	- Eine Seite vor.
(SHIFT) ↑	- Erste Text-Zeile, erstes Zeichen.
(SHIFT) ↓	- Letzte Text-Zeile, letztes Zeichen.

Ausserdem ändert sich seine Position bei den meisten Kommandos, sowie bei den meisten Funktions-Aufrufen.

Wenn der Cursor in die vorhergehende Bildschirmzeile bewegt wird, so gibt es zwei Möglichkeiten, wo er erscheinen kann. Wenn die Zeile mit einem CR abschließt, so steht der Cursor am Ende der Zeile auf diesem CR. Andernfalls blinkt er genau 64 Zeichen über seiner alten Position.

Wenn der Cursor in die nachfolgende Bildschirmzeile bewegt wird, so gibt es ebenfalls zwei Möglichkeiten. Wenn die Ausgangszeile mit einem CR abschließt, so erscheint der Cursor am Anfang der nächsten Zeile. Andernfalls ist er genau 64 Zeichen unter seiner alten Position.

Diese Eigenschaften können Sie ausnutzen, wenn Sie schnell ans Ende oder an den Anfang einer Zeile möchten.
Um an das Ende einer Zeile zu gelangen, drücken Sie nacheinander die Pfeil-Tasten '↓' und '↑'.
Um an den Anfang zu gelangen, drücken Sie nacheinander die Pfeil-Tasten '↑' und '↓'.
Dadurch wurden zwei zusätzliche Funktionen gespart.

Die Cursorbewegung mit (SHIFT) ← und (SHIFT) → ermöglicht ebenfalls ein schnelles Erreichen der gewünschten Position.
Wenn Sie z.B. ans Ende eines Wortes wollen, so halten Sie solange (SHIFT) → gedrückt, bis Sie auf dem ersten Zeichen des nächsten Wortes sind. Dann können Sie mit ← die Position einfach erreichen.

In diesem Zusammenhang noch ein Hinweis zu den Tabulatoren.
Ein Tabulator besteht aus einem Zeichen.
Dies ist z.B. in Assembler-Programmen platzsparend aber dennoch übersichtlich.

Aber obwohl bis zu 8 Leerzeichen für einen Tabulator ausgegeben werden, können Sie den Cursor nur auf das erste Leerzeichen oder hinter das letzte Leerzeichen des Tabulators bewegen.
Wenn Sie den Tabulator löschen (auf dem ersten Zeichen), dann verschwinden sämtliche Leerzeichen des Tabulators.
Ähnliches gilt ja auch für das Steuerbyte-Zeichen ((P1) G).

Ebenfalls benutzt wird der Bereich des Basic-Input-Buffers ab Adresse 41E8H.
Dort werden die Find- und Replace-Strings von den entsprechenden Befehlen abgelegt.
Wenn Sie ML auf 42E8H setzen, sind zwar weniger als 256 Bytes für den Stack-Bereich frei, aber auch dies reicht im Normalfall noch aus.

Ansonsten sind alle Speicherbereiche von 42E8H bis Memory low, von Memory high bis CFFFH und von 3000H bis 33FFH zu benutzen.

Auf jedenfall sollten Sie, bevor Sie Eingriffe in diese Speicherbereiche vornehmen oder z.B. ein Programm assemblieren und testen wollen, den Text sicherheitshalber vorher abspeichern !
Die Erfahrung zeigt, daß das Warten beim abspeichern auf Kassette insgesamt kürzer ist als die Zeit, die für die neue Eingabe eines Textes oder eines Programmes nötig ist.

Kritische Anmerkungen zum Genie II S Text-Editor

Die Möglichkeiten des Genie II S wurden, was zum Beispiel die Verarbeitungsgeschwindigkeit angeht, voll ausgenutzt.
Dennoch ist z.B. das Schreiben im Insert-Modus am Anfang eines sehr langen Textes nicht mehr mit dem Zehn-Finger-System möglich.
Es wurde auf eine ausgefeilte Art der Textspeicherung, wie sie größere Textsysteme vornehemen, verzichtet.
Auch die Fehlermeldungen mußten auf ein Mindestmaß beschränkt werden, da der Speicherplatz doch sehr knapp war.

Sie müssen bedenken, daß der Z80-Assembler, der Monitor und das Textsystem in nur 12 kByte Speicher Platz finden mußten.
Dafür haben Sie gegenüber anderen Computern den Vorteil, daß alle diese Programme sofort verfügbar sind und nicht erst von Kassette oder Diskette geladen werden müssen !

Wir hoffen, daß mit dieser Anleitung zum Genie II S Text-Editor eine Einführung geglückt ist, die Sie zur vollen Ausnutzung seiner Möglichkeiten befähigt.

3.3 Das Format der Texte für die Text-Tasten

Für die Belegung der Text-Tasten ist ein Speicherbereich von 256 Bytes am Beginn des Programms reserviert.
Der Text, der bei (P1) 0 ausgegeben wird ist der erste im Speicher. Er muß mit einem Byte 00H enden. Danach beginnt der nächste Text für die Taste (P1) 1 usw.
Nach dem letzten Byte 00H für (P1) 9, muß noch ein Byte FFH folgen, das benötigt wird, um bei Benutzung des Kn-Befehls das Ende der Text-Tasten zu finden und den bisher verbrauchten Platz zu berechnen.

Wenn eine Text-Taste keinen Text erzeugen soll, so muß an der entsprechenden Position der Tabelle nur eine 00H stehen.
Sie können also maximal 245 Zeichen auf eine Text-Taste legen (256 minus 1 mal FFH minus 10 mal 00H).
Da in der Kommandozeile die Eingabe von Steuerzeichen außer (CR) und (TAB) nicht möglich ist, müssen Sie, wenn Sie eine Text-Taste damit belegen wollen, dies mit Hilfe des Monitors vornehmen.
Hierzu ein Beispiel :
Sie wollen auf die Taste (P1) 0 ein (ESC) 7 02H, ein (TAB) und den Text "ENDE" ablegen.
Dies sind insgesamt 9 Bytes.
7F 37 B7 02 09 45 4D 44 45
Geben Sie also ein :
(BREAK)
K0=:::::::::(RETURN)
Jetzt legt das Textsystem neun Bytes mit dem Wert 3AH ab und verschiebt die restlichen Belegungen entsprechend.

Drücken Sie nun (BREAK) M (RETURN), um in den Monitor zu gelangen. Dort geben Sie dann H D000 ein, suchen die Doppelpunkte (im ASCII-Dump) und tippen die oben aufgelisteten Bytes ein. (Bei Texten können Sie mit (CLEAR) auch in denASCII-Edit Modus umschalten).
Wenn dies alles getan ist, drücken Sie (BREAK) T (RETURN).
Sie sind dann wieder im Textsystem und können mit (P1) 0 die gewünschte Funktion aufrufen.

Beachten Sie aber immer, daß Sie nicht mehr als 256 Byte verändern dürfen, da anschließend an die Text-Tasten das Programm und Variablen beginnen.

3.4 Die benutzten Adreßbereiche

Der Text-Editor benutzt hauptsächlich die Variablen innerhalb des Bereiches von D000H bis FFFFH.
Dazu gehören z.B. alle Zeiger auf Cursorposition, die Block-Markeradresse, der Programmname usw.
Jedoch liegt der Bereich des Stackpointers (SP-Register) während des Arbeitens mit dem Text-Editor direkt vor der System-Variablen ML (Memory low).
Da 256 Bytes für den Stackpointerbereich genügend sind, wird auch beim Start des Text-Editors ML auf die Adresse hinter der letzten Basic-Feldvariablen plus 256 gesetzt.
Dies ist zu beachten, wenn Sie ML ändern wollen.
Bei der Anleitung zum Gebrauch des Assemblers wird noch auf die benutzbaren Speicherbereiche eingegangen.

BIT	b,B b,C b,D b,E b,H b,L b,(HL) b,A b,(IX+n) b,(IY+n)
RES	b,B b,C b,D b,E b,H b,L b,(HL) b,A b,(IX+n) b,(IY+n)
SET	b,B b,C b,D b,E b,H b,L b,(HL) b,A b,(IX+n) b,(IY+n) (b = 00H 01H 02H 03H 04H 05H 06H 07H)
JP	ww NZ,ww Z,ww NC,ww C,ww PO,ww PE,ww P,ww M,ww (HL) (IX) (IY)
CALL	ww NZ,ww Z,ww NC,ww C,ww PO,ww PE,ww P,ww M,ww
RET	NZ Z NC C PO PE P M
RST	n (n = 00H 08H 10H 18H **20H 28H 30H 38H**)
JR	ww NZ,ww Z,ww NC,ww C,ww
DJNZ	ww
PUSH	BC DE HL AF IX IY
POP	BC DE HL AF IX IY
IM	n (n = 00H **01H 02H**)

LD

	B	C	D	E	H	L	(HL)	A	(IX)	(IY)	(BC)	(DE)	R	I	(ww)	n
B,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
C,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
D,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
E,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
H,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
L,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
(HL),	*	*	*	*	*	*	-	*	-	-	-	-	-	-	-	*
A,	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(IX+n),	*	*	*	*	*	*	-	*	-	-	-	-	-	-	-	*
(IY+n),	*	*	*	*	*	*	-	*	-	-	-	-	-	-	-	*
(BC),	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
(DE),	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
R,	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
I,	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
(nn),	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-

(* = zulässige Adressierungsart, - = unzulässig)

LD	BC,ww BC,(ww) (ww),BC DE,ww DE,(ww) (ww),DE HL,ww HL,(ww) (ww),HL SP,ww SP,(ww) (ww),SP **SP,HL** SP,IX SP,IY IX,ww IX,(ww) (ww),IX IY,ww IY,(ww) (ww),IY
EX	AF,AF' DE,HL (SP),HL (SP),IX (SP),IY
IN	A,(n) B,(C) C,(C) D,(C) E,(C) H,(C) L,(C) A,(C)
OUT	(n),A (C),B (C),C (C),D (C),E (C),H (C),L (C),A

Befehle ohne Argumente

RLCA	RRCA	RLA	RRA	DAA	CPL	SCF	CCF
NEG	EXX	RETI	RETN	EI	DI	NOP	HALT
LDI	LDD	LDIR	LDDR	CPI	CPD	CPIR	CPDR
INI	IND	INIR	INDR	OUTI	OUTD	OTIR	OTDR

Der Assembler wird vom Text-Editor aus mit den Befehlen A, AL oder AP aufgerufen.
Der Befehl A assembliert den Textbuffer, ohne ein Listing im zweiten Durchgang (Pass 2) auszugeben.
Der Befehl AL listet das Programm im zweiten Durchgang auf den Bildschirm und AP druckt es außerdem auf den Drucker aus.

Beim Ausdruck wird eine Seitenformatierung mit den Parametern ZS und PL des Text-Editors vorgenommen (s. ZS- und PL-Befehl).

Der Assembler ist ein Zwei-Pass Assembler.
Im ersten Durchgang wird eine Tabelle aller Symbole (Labels) angelegt, in der die Symbol-Adressen abgelegt werden.
Dabei werden doppelt definierte Symbole erkannt.

Im zweiten Durchgang werden die Werte für die Symbole der Tabelle entnommen und im enstehenden Programm eingesetzt.

Das Programm wird im zweiten Durchgang entweder hinter der Symboltabelle in einem Buffer abgelegt, oder, wenn Sie die Option .OBJECT gewählt haben, direkt ab der Adresse abgelegt, die durch das Origin (ORG-Befehl) festgelegt wurde.

Zunächst nun eine Liste aller zulässigen Befehle
und der mit ihnen zulässigen Adressierungsarten:

Befehl	Adressierungsarten
ADD	A,n A,B A,C A,D A,E A,H A,L A,(HL) A,A
	A,(IX+n) A,(IY+n)
	HL,BC HL,DE HL,HL HL,SP
	IX,BC IX,DE IX,IX IX,SP
	IY,BC IY,DE IY,IY IY,SP
ADC	A,n A,B A,C A,D A,E A,H A,L A,(HL) A,A
	A,(IX+n) A,(IY+n)
	HL,BC HL,DE HL,HL HL,SP
SUB	n B C D E H L (HL) A (IX+n) (IY+n)
SBC	A,n A,B A,C A,D A,E A,H A,L A,(HL) A,A
	A,(IX+n) A,(IY+n)
	HL,BC HL,DE HL,HL HL,SP
AND	n B C D E H L (HL) A (IX+n) (IY+n)
XOR	n B C D E H L (HL) A (IX+n) (IY+n)
OR	n B C D E H L (HL) A (IX+n) (IY+n)
CP	n B C D E H L (HL) A (IX+n) (IY+n)
INC	B C D E H L (HL) A (IX+n) (IY+n)
	BC DE HL SP IX IY
DEC	B C D E H L (HL) A (IX+n) (IY+n)
	BC DE HL SP IX IY
RLC	B C D E H L (HL) A (IX+n) (IY+n)
RRC	B C D E H L (HL) A (IX+n) (IY+n)
RL	B C D E H L (HL) A (IX+n) (IY+n)
RR	B C D E H L (HL) A (IX+n) (IY+n)
SLA	B C D E H L (HL) A (IX+n) (IY+n)
SRA	B C D E H L (HL) A (IX+n) (IY+n)
SLI	B C D E H L (HL) A (IX+n) (IY+n)
SRL	B C D E H L (HL) A (IX+n) (IY+n)
RLD	(HL)
RRD	(HL)

Geben Sie dieses Programm mit dem Text-Editor einmal ein.
Die Kommentare (; Kommentar) brauchen Sie nicht mit einzugeben.

Der Befehl .OBJECT veranlaßt den Assembler, das Programm im zweiten Durchgang ab der Adresse von PC abzulegen (3000H...).
Der Befehl ORG legt die Adresse fest, ab der das Programm stehen soll.

Danach folgen die Definitionen von Labels, die außerhalb des Programms liegen (Fachbezeichnung: EXTERNALS).
Dies ist zur Erhaltung der Übersichtlichkeit bei langen Programmen nützlich und erspart unter Umständen sogar erläuternde Kommentarzeilen.

Wie Sie hier schon sehen, sind die Labels immer von einem Doppelpunkt gefolgt. Dies soll dazu dienen, die Stelle im Text zu finden, an denen ein Label definiert wird, ohne alle Benutzungen des Labels durchsuchen zu müssen.
(Im Texteditor z.B. F/START:²CR³)
Die Länge eines Labels ist nicht begrenzt, Sie können also mehr als sechs Zeichen für ein Label verwenden.
Aus diesem Grunde kann ein Label auch alleine in einer Zeile stehen oder nur von einem Kommentar gefolgt sein.
Auf alle Fälle muß ein Label in der ersten Spalte beginnen.

Das Label CHR: ist ein 8-Bit Label (ein Byte), d.h. die oberen 8 Bit sind null. Wenn Sie z.B. eingeben würden: LD HL,CHR ,
so wäre das Ergebnis in hexadezimaler Form: 21 2A 00.
Andersherum können Sie aber auch z.B. eingeben:
LD A,CLS das Ergebnis wäre : 3E C9.
Es wird also nicht schon bei den Labels zwischen 8 und 16 Bit unterschieden, sondern erst bei den Befehlen selbst.
Wenn Sie jedoch die oberen 8 Bit eines Labels in ein Einzel-Register laden wollen, so können Sie z.B. LD A,CLS>8 eingeben
Das 8 im Argument bedeutet, daß das bisherige Ergebnis achtmal bitweise nach rechts geschoben (d.h. durch 256 geteilt) wird.
Anstelle der 8 könnte auch wiederum ein Label stehen.
Sie können auch einen Ausdruck mit <n nach links schieben.

Doch zurück zum Programm.
Der nächste Befehl ist ein DEFM. Er bewirkt, daß der folgende, in Anführungszeichen oder Apostrophen stehende Text, ab der augenblicklichen Programmzähler-Adresse abgelegt wird.
Die Länge eines solchen Textes darf maximal 255 Zeichen sein.
Wenn Sie einen längeren Text benötigen, so müssen Sie eine neue Zeile mit DEFM beginnen.
Daran anschließend folgt der Befehl DEFB. Er bewirkt, daß das angegebene Byte 00H an der Adresse von PC abgelegt wird.
Der Befehl dient zur Eingabe von Bytes, die mit DEFM nicht abgelegt werden können, weil Sie nicht über die Tastatur erreichbar sind (z.B. 0DH, 00H oder Grafikzeichen)
Der DEFW Befehl hat eine ähnliche Wirkung, aber er legt jeweils zwei Bytes (Doppelbyte oder Wort) in der Form LSB,MSB ab PC im Speicher ab.
Das eigentliche Programm beginnt nun mit der Zeile:

```
BEGIN:  CALL    CLS
```

Das Label BEGIN ist ein INTERNAL, d.h. es liegt innerhalb des Programmes und erhält daher automatisch die Adresse von PC an dieser Stelle. Es wird quasi folgendes ausgeführt:

```
BEGIN:  EQU     $
        CALL    CLS
```

Außerdem werden folgende Pseudobefehle akzeptiert:

ORG	ww	PC auf Adresse ww setzen.
END	(ww)	Ende des Assemblertextes und Entry auf ww setzen (wenn angegeben).
DEFB	n(,n,n,n)	Ein Byte oder eine Liste von Bytes ab PC ablegen.
DEFW	ww(,ww,ww,ww)	Ein Wort oder eine Liste von Worten ab PC ablegen.
DEFS	n(,z)	n Bytes mit dem Wert z ab PC ablegen. (Wenn z fehlt wird 00H angenommen)
DEFM	"abc" o. 'abc'	Einen Text in Anführungszeichen oder Apostrophen ab PC ablegen.
EQU	ww o. n	Einem Label einen Wert zuweisen.
DEFL	ww o. n	Einem Label einen neuen Wert zuweisen.
.OBJECT		Programm ab PC im Speicher ablegen.

Bei den Argumenten bedeutet:

n ein Byte	42, 42D, 2AH, 00101010B, 52O, "*", '*' oder ein Ausdruck: STERN oder "!"+9 usw.
b eine Bitzahl	von 0 ... 7.
ww ein Wort	12345, 12345D, 3039H, 0011000000111001B "09", '09' oder ein Ausdruck: START, $+10 ENDE-4, SCREEN>4 usw.

Es folgt jetzt ein kleines Programm, anhand dessen einige Befehle des Assemblers erläutert werden sollen.

```
; ****************  Programm STERNCHEN
        .OBJECT             ; Im Speicher ablegen
        ORG     3000H       ; 3000H Basisadresse
;
; Label Definitionen
;
CLS:    EQU     01C9H       ; CLS-Routine im ROM
PRTSTR: EQU     2B75H       ; gibt Text ab (HL) bis 00H aus
INCH:   EQU     0049H       ; Tastatur abfragen
SCREEN: EQU     3C00H       ; Bildschirmspeicher Adresse
LENGTH: EQU     400H        ; Bildschirmspeicher Länge
CHR:    EQU     "*"         ; irgenein Zeichen
;
; Mitteilung
;
MESSGE: DEFM    "DRÜCKEN SIE IRGENDEINE TASTE !!"
        DEFB    0
;
; Start des Programms
BEGIN:  CALL    CLS                 ; Bildschirm löschen
        LD      HL,SCREEN           ; HL = Bildschirmspeicher
        LD      DE,SCREEN+1         ; DE = Bildschirmspeicher+1
        LD      BC,LENGTH-1         ; BC = Länge-1
        LD      (HL),CHR            ; erstes Bildschirmzeichen ="*"
        LDIR                        ; (HL) -> (DE) bis BC = 0
        LD      HL,MESSGE           ; HL = Mitteilung
        CALL    PRTSTR              ; ausgeben
        CALL    INCH                ; auf Tastendruck warten
        RET                         ; Zurück zum Hauptprogramm
        END     BEGIN               ; Ende des Programms
```

Nach Ausführung dieser ROM-Routine, soll der Rechner auf einen Tastendruck warten. Die Routine 0049H übernimmt dies und gibt im Akku den Wert der gedrückten Taste zurück.

Anschließend erfolgt ein Rücksprung zum aufrufenden Programm.

Der END-Befehl muß immer der letzte Befehl eines Programm-Textes sein. Der Ausdruck hinter dem END-Befehl ist ein Zeiger auf die Einsprungadresse des Programms.
Wenn kein Argument angegeben ist, so wird die Adresse des ersten ORG Befehls als Einsprungadresse angenommen.
Bei diesem Programm liegt aber ab 3000H ein Text, der nicht als Programm ausgeführt werden soll.
Daher das Argument BEGIN um die Startadresse festzulgegen.

Nach diesem END-Befehl können Sie einen noch beliebige Texte anhängen (z.B. Programmerläuterung oder Ausführungshinweise) die der Assembler nicht beachtet.

Wenn Sie dieses Programm nun eingegeben haben, so können Sie es zunächst auf seine Richtigkeit testen.
Geben Sie im Text-Editor ein: (BREAK) A (CR).
Der Bildschirm wird gelöscht und es erscheint
kurz hintereinander
Assembler

Pass 1

Pass 2

(RETURN) **drücken...**

Wenn Sie nun RETURN drücken, sind Sie wieder im Text-Editor.
Der Cursor ist in der ersten Zeile über dem ersten Zeichen.

Wenn ein Fehler im Programm vorliegt, so wird die Fehlermeldung in der zweiten Bildschirmzeile angezeigt, und der Cursor erscheint in der Fehlerzeile.
Bei den meisten Fehlern ist der Cursor zusätzlich an der fehlerhaften Stelle in der Zeile positioniert.

Wenn Sie keinen Fehler im Programm hatten, dann geben Sie probeweise einmal einen Fehler ein.
Schreiben Sie z.B. CAL anstelle von CALL. In diesem Fall würde die Fehlermeldung "Illegaler Befehl" lauten, und der Cursor wäre über dem ersten Zeichen des fehlerhaften Befehls.

Die möglichen Fehlermeldungen sind :

Illegaler Befehl	Falsche schreibweise in einem Befehl / Label beginnt nicht in der ersten Spalte o.ä.
Fehler im Ausdruck	Die gewählte Adressierungsart ist mit diesem Befehl nicht zulässig/ es wurde eine Klammer, ein Komma o.ä. vergessen oder zuviel eingegeben / ein Label endet nicht mit einem Doppelpunkt usw.

Das $-Zeichen steht im Assembler für den augenblicklichen PC.
Diese Zeilen könnten Sie auch eingeben, es wäre jedoch eine Platzverschwendung.

Anschließend folgen die Vorbelegungen **der Doppelregister** BC, DE und HL für den LDIR Befehl.
Statt der Labels könnten Sie hier **auch verwenden:**

```
LD      HL,3C00H
LD      DE,3C01H
LD      BC,03FFH
```

jedoch mit den Labels SCREEN und LENGTH können Sie dieses Programm auch für einen anderen Z-80 Recher umschreiben, dessen Bildschirmspeicher nicht bei 3C00H beginnt und der eine andere Länge hat, indem Sie nur die Definitionszeilen ändern.
Dies ist besonders wichtig für Veröffentlichungen. Außerdem ist es bei längeren Programmen oft sehr zeitsparend.

Wenn Sie z.B. den Bereich hinter einem Programm als Buffer benutzen wollen und Sie müssen das Programm aus irgendwelchen Gründen verlängern, so müßten Sie dann alle Adressen, die diesen Buffer betreffen, wieder abändern.

Es ist also viel einfacher, wenn Sie ein Label mit dem Namen BUFFER benutzen, und so immer nur eine Wertzuweisung ändern müssen, oder noch einfacher: Sie schreiben das Label BUFFER: vor den END-Befehl in der letzten Zeile.

Noch ein Hinweis zu den Labeldefintionen,
es sind zwar Verschachtelte Definitionen **erlaubt wie z.B.**

```
BASE:   EQU     8000H
BUFF:   EQU     BASE+10H
```

aber Sie dürfen Labels erst dann benutzen, wenn Sie schon definiert sind.
Sonst sind möglicherweise seltsame Ergebnisse zu erwarten.
Ein Beispiel für einen solchen Fehler:

```
START:  DEFS    NEXT-START
NEXT:   ......
```

Der Assembler kann hier die Differenz zwischen NEXT und START nicht richtig berechnen, da erst das Ergebnis dieser Differenz die Adresse von NEXT festlegen würde.
Am besten verwenden Sie in dieser Situation entweder einen eine Zahl als Argument, oder aber schon festliegende Labels.

Der nächste Befehl im Programm ist LDIR. Dieser Befehl kopiert das Byte von der Adresse, auf die HL zeigt, auf die Adresse, auf die DE zeigt. Anschließenden werden HL und DE um eins erhöht und BC um eins erniedrigt. Wenn BC dabei nicht 0 wird, dann wird der Befehl wiederholt.

Hier wird also das Sternchen von 3C00H nach 3C01H kopiert.
Anschließend zeigt HL auf Adresse 3C01H und DE auf 3C02H und das Sternchen wird von 3C01H nach 3C02H kopiert usw.
Diese Methode wird oft verwendet, um einen Speicherbereich zu löschen oder mit einem bestimmten Wert zu belegen.
Anschließend wird HL mit der Adresse eines Textes geladen, um eine ROM-Routine aufzurufen, die ab Adresse HL Zeichen auf den Bildschirm ausgibt, bis ein Byte mit dem Wert 00H erreicht wird.

Ein "Relativer Sprung ist zu groß", wenn die Distanz zwischen Zieladresse und momentaner Adresse größer als 127 ist.
Da der Assembler jedoch Sprünge relativ zur Adresse des gerade assemblierten Befehls berechnet, ist der maximale Offset bei den zwei Byte langen relativen Sprungbefehlen $-125 bis $+129.
Der Befehl JR $ springt also wieder zwei Bytes zurück.

"Kein END-Befehl" ist ein eindeutiger Fehler.
In der letzten Programmzeile muß immer der END Befehl **stehen.**

Die Fehlermeldung "Zuwenig Speicherplatz" kann behebbar sein, wenn die Text-Buffer Grenzen noch nicht maximal ausgenutzt sind. Sie können Sie dann mit den Text-Editor Befehlen ML und MH auf maximale Speicherausnutzung erweitern (s. auch "Text-Editor").
Wenn das schon geschehen ist, oder wenn die Fehlermeldung schon im ersten Durchgang auftritt, so bestehen kaum Chancen das Programm noch zu assemblieren.
Einzige Möglichkeit bleibt dann, Labels zu verkürzen oder wegzulassen, Kommentarzeilen zu entfernen und evtl. mehr Unterprogramme zu benutzen.
Ansonsten können Sie auch versuchen, das Programm in mehrere Teile aufzuteilen und getrennt zu assemblieren.
Dabei gehen natürlich die Symbole aus dem ersten Teil beim zweiten Teil verloren, und müssen umständlich mit
LABEL: EQU xxxx neu definiert werden.

Da Sie maximal 36 kByte Text-Speicher zur Verfügung haben **und** das Verhältnis von Länge des Quellprogramms zu Länge des Objectprogramms ca. 10:1 ist, können Sie knapp 3.5 kByte **lange** Programme noch assemblieren.

Versuchen Sie daher bei längeren Programmen Daten, Bilder, Texte usw. außerhalb des Quell-Programms zu speichern.
Sie können dann ein Programm z.B. mit dem .OBJECT Befehl direkt im Speicher ablegen und anschließend mit dem S Befehl des Monitors zusammen mit den Daten auf Kassette abspeichern.

Die Fehlermeldung "Geschützter Bereich xxxxH !!" tritt nur **auf,** wenn Sie den .OBJECT Befehl verwendet haben und das Programm einen geschützten Speicherbereich überschreiben würde.
Diese Bereiche sind der Bereich des Text-Buffers, der Bereich von 0000H bis FFFFH und die Symboltabelle, die hinter dem Textende beginnt.
Ändern Sie die Adresse des ORG Befehls, oder ändern Sie die Text-Buffer Adressen, um einen größeren freien Speicherbereich zu erhalten.
Beachten Sie, daß vor ML (Memory low) noch 100H Bytes für den Stackpointer-Bereich frei bleiben müssen.

Bevor Sie ein assembliertes Programm ausführen wollen, sollten Sie den Quelltext in jedem Fall auf Kassette abspeichern.
Es ist ja nie ganz sicher, daß das Programm so arbeitet wie es soll und nicht zu einem Kaltstart führt oder den Speicherinhalt überschreibt.

Wenn ein Programm im Bereich von 0000 bis FFFF laufen soll, so können Sie es natürlich nur testen, indem Sie es auf Kassette abspeichern und anschließend mit dem SYSTEM-Befehl vom Basic her wieder einlesen.

Undefiniertes Symbol	Ein nicht definiertes Label wurde in einem Ausdruck benutzt/ eine Zahl beginnt nicht mit einer Ziffer (z.B. FFH statt OFFH).
Doppelt definiertes Symbol	Ein Label wurde zum zweiten mal benutzt.
Relativer Sprung zu groß	Der Offset für einen JR oder DJNZ Befehl ist kleiner als $-125 oder größer als $+129.
Kein END-Befehl	Der END-Befehl fehlt oder wurde versehentlich gelöscht.
Zuwenig Speicherplatz	Die Symboltabelle hat keinen Platz innerhalb der Text-Buffer Grenzen / Programm ist zu lang für die Text-Buffer Grenzen.
Geschützter Bereich xxxxH	Der Bereich xxxxH darf nicht überschrieben werden.

Wie behebt man nun diese Fehler ?
Ein "Illegaler Befehl" ist meist wohl ein Rechtschreibefehler.
Wenn nicht, so vergewissern Sie sich, ob der Befehl in der Tabelle der zulässigen Befehle aufgelistet ist.

Ein "Fehler im Ausdruck" kann verschiedene Ursachen haben.
Am besten einige Beispiele für häufig gemachte Fehler:

```
        LD      A,3A            3AH    wäre richtig
START   CALL    5000H           START: wäre richtig
        RES     (HL)            b,     fehlt
        BIT     4,HL            Doppelregister nicht erlaubt
        LD      (HL),(IX+2)     Adressierungsart nicht erlaubt
        INC     AF              Adressierungsart nicht erlaubt
```

Aber ACHTUNG !!!

```
        LD      A,18            wird akzeptiert als 1 binär.
```

In diesem Fall erhalten Sie keine Fehlermeldung !
Geben Sie also bei solchen Argumenten besonders acht.

Ein "Undefiniertes Symbol" kann auftreten, wenn Sie ein Symbol das Sie benutzen wollen tatsächlich noch nicht definiert haben. Oder Sie haben es in der Definition und im Ausdruck unterschiedlich geschrieben (Groß-/Kleinschrift beachten). Eine weitere Möglichkeit ist, daß eine hexadezimale Zahl größer als 9FH nicht mit einer führenden Null beginnt.
Aber ACHTUNG !!!
Wenn Sie z.B. ein Label mit dem Namen 'BCH' definiert haben, und Sie wollen den Befehl LD A,0BCH ausführen, so führt eine fehlende Null nicht zu einer Fehlermeldung.
Vermeiden Sie daher Symbol-Namen die mit Hexadezimalzahlen, Z-80-Registernamen oder Bedingungs-Flags verwechselbar sind.

Ein "Doppelt definiertes Symbol" liegt vor, wenn zwei Labels die gleiche Länge haben und gleich geschrieben sind.
Aber die Labels START, Start und StArT sind unterschiedlich. Sie können also ein Unterprogramm TEXT: nennen und den auszugebenden Text mit dem Label Text: versehen, ohne daß die Fehlermeldung "Doppelt definiertes Symbol" auftaucht.

Kleinere Programme (bis 1 kByte) können Sie in den Bereich 3000H bis 33FFH assemblieren. Dort ist ein Speicherbereich, der vom Basic nicht benutzt wird und sehr sicher ist.

Wenn Sie aus dem Text-Editor oder dem Monitor ins Basic zurückgehen, so wird der letzte freie Speicher des Basic-Bereiches auf die Adresse von ML -1 gesetzt. Wenn Sie ML auf Adresse 42E8H gesetzt hatten, so kann das Basic nicht mehr richtig arbeiten, da 'weniger als 0 Bytes' Platz vorhanden sind. Also legen Sie entweder ML wieder höher, oder führen Sie einen Kaltstart aus. Dies geht z.B. vom Monitor mit dem Befehl G 0(CR).
Jedoch ist dann der Text-Buffer auch verloren und muß evtl. neu von Kassette geladen werden.

Wenn Sie also im Basic und Text-Editor abwechselnd arbeiten wollen, dann dürfen Sie den Parameter ML nicht kleiner wählen als zu Beginn ausgerechnet wurde. Das Basic-Programm bleibt dann erhalten, solange Sie nicht in diesen Bereich assemblieren oder ihn vom Monitor her überschreiben.

Da das Basic aus einem **EPROM ins RAM 'gebootet'** wird, ist es auch veränderbar. Dazu müssen Sie nur den 'Schreibschutz' für den **unteren 12 kByte** Bereich ausschalten (**BIT 5 auf Port FEH** zurücksetzen). Jedoch müssen Sie dann besonders vorsichtig bei direkten Änderungen sein, da auch der Text-Editor, der Monitor und der Assembler Unterprogramme aus dem Basic-Bereich benutzen.

Wenn Sie also das Basic manipulieren wollen, so ist es noch wichtiger alle Programme auf Kassette zu sichern.

In diesem Zusammenhang ein Hinweis:

Alle Literatur über das Microsoft-Basic des Genie I/II trifft auch im wesentlichen für das Genie II S zu. Es sind lediglich kleine Änderungen vorgenommen worden. Diese sind jedoch im Vergleich mit einem ROM-Listing sehr schnell ersichtlich.

Unter diese Änderungen fallen:
1. Der Text "READY?" - "Mem Size?"
   (CR)(CR)... "READY" - (CR)(CR)"Genie II S"
2. Der NAME Befehl zum Aufruf der Systemerweiterung.
3. Eine automatische Umschaltug auf LOW SPEED bei Kassettenoperationen und zurück auf HIGH SPEED bei MOTOR OFF.

Außerdem wurde die Routine für Zufallszahlen verkürzt, aber gleichzeitig verbessert und beschleunigt. Es entfallen die Multiplikationen, statt dessen wird das R-Register des Z-80 verwendet um Zufallszahlen zu erzeugen.
Der RANDOM-Befehl wäre somit eigentlich überflüssig, wurde aber beibehalten, um die Lauffähigkeit 'alter' Basic-Programme zu garantieren.

Wenn Sie sich noch in die Programmierung des Z-80 einarbeiten wollen, so kann diese Anleitung dabei nicht als Lernhilfe betrachtet werden, da nur die Besonderheiten speziell dieses Assemblers ausführlich erläutert sind.
Wir empfehlen Ihnen daher die einschlägige Literatur über den Z-80 Prozessor und seine Programmierung.
(z.B. Rodnay Zaks, Programmierung des Z-80, SYBEX-Verlag o.ä.).

GENIE IIs

Monitor

Text-Editor

Assembler

TCS
COMPUTER GMBH

Da Sie sich vielleicht an den Umgang mit dem Genie II S ROM in der Grundversion des Genie II S schon gewöhnt haben, möchten wir Ihnen auch zusammen mit der Diskettenstation die Arbeit mit diesem System ermöglichen.
Da das Genie II S ROM aber nur Kassettenroutinen enthält und daher eine Ein-/ Ausgabe auf Diskette nur schwer möglich ist, ist auf der Genie-DOS Master Diskette eine Disk-Version des Programms mit dem Namen NAME/CMD abgespeichert.

In dieser Version sind einige Befehle gegenüber der Grundversion geändert worden.

Der B-Befehl (ursprünglich BASIC) wurde durch den E-Befehl (EXIT) erstetzt. Es wird nun kein BASIC-Warmstart mehr ausgeführt, sondern ein Rücksprung ins DOS (über Adr. 402DH).

Die Befehle Load, Save und View des Monitors fragen den Programmnamen gesondert ab (10 und 11 Bildschirmzeile). Der Name muß immer angegeben werden. Der blinkende Stern und die Anzeige der ein-/ ausgegebenen Bytes sind nun nicht mehr vorhanden.

Auch im Text-Editor mußten einige Änderungen vorgenommen werden. Die minimale Adresse für den Parameter ML ist 5300H, da das DOS, der I/O-Buffer und der Bereich für den Genie-DOS-FCB (File-Control-Block) nicht überschrieben werden dürfen. Aus diesem Grunde ist der verfügbare Platz im Text-Editor auf ca. 32 kByte begrenzt.

Um eine Bearbeitung von reinen ASCII-Files zu erlauben, wurde der Text-Editor um den Befehl SA (Save-ASCII) erweitert. Ein solcher ASCII-File ist dann für andere Zwecke verwendbar. (z.B. LIST- oder PRINT-Befehl im DOS, LOAD- oder MERGE-Befehl im BASIC, andere Text-Systeme, andere Assembler usw.). Der L-Befehl unterscheidet selbständig zwischen reinen ASCII-Files und Text-Editor Files.

Bei den Befehlen P und PB des Text-Editors wird die auszudruckende Zeile auch im Bildschrim angezeigt und mit dem Zeichen '*' markiert.

Der Assembler speichert ein Programm nun ebenfalls auf Diskette ab. Dabei wird das übliche Maschinenprogramm-Format verwendet, d.h. Sie können ein so assembliertes Programm direkt vom DOS her starten.

Wenn Sie noch Maschinenprogramme auf Kassette haben, dann können Sie diese mit dem Programm VL/CMD auf Diskette kopieren. Um auf Kassette abgespeicherte Texte auf Diskette zu übertragen, verwenden Sie das Programm TCOPY/CMD auf der Genie-DOS Diskette.

Die Fehlermeldung TAPE ERROR wurde durch eine ausführliche Fehlermeldung ersetzt, die das DOS-Error-Overlay ab Adresse 4409H übernimmt. Sie erhalten also die vom DOS her bekannten Fehlermeldungen (z.B. Bauteil nicht erreichbar, Datei nicht in Inhaltsverzeichnis usw.).

Hierzu ein Beispiel:
Ab Adresse 7000H stehe folgendes Programm.

```
7000 210970      LD    HL,7009H
7003 CD752B      CALL  2B75H
7006 C30070      JP    7000H
7009 2A2A2A20    DEFM  "*** "
700D 00          DEFB  0
```

Sie wollen dieses Programm nun ab Adresse 6000H laufen lassen. Also verschieben Sie es zunächst mit dem M-Befehl des Monitors von 7000H nach 6000H.
Geben Sie ein M 7000 700D 6000 (RETURN),
dann O 6000 600D 7000 700D 6000 (RETURN).
Es werden alle Adressen zwischen 7000 und 700D die im Bereich von 6000 bis 600D stehen auf die Basis 6000 umgerechnet und im jeweiligen Befehl abgespeichert.
Hinterher sieht das Programm dann folgendermaßen aus:

```
6000 210960      LD    HL,6009H
6003 CD752B      CALL  2B75H
6006 C30060      JP    6000H
6009 2A2A2A20    DEFM  "*** "
600D 00          DEFB  0
```

P xxxx yyyy (RETURN)
PRINT
Druckt den Speicherbereich von xxxx bis yyyy disassembliert aus. Es wird eine Seitenformatierung mit den Parametern PL,RL und ZS des Text-Editors ausgeführt.

Q (xxxx) (RETURN)
DISASSEMBLER
Dieser Befehl dient zur disassemblierten Darstellung von Speicherinhalten (d.h. in Z-80 Mnemonics).
Wenn Sie nur Q eingeben, so wird der Disassembler ab Programmzähler (PC) ein- bzw. ausgeschaltet.
Wenn Sie Q xxxx eingeben, so wird der Speicherbereich ab Adresse xxxx dissassembliert dargestellt.
Die Tasten Pfeil-links und Pfeil-rechts disassemblieren den vorhergehenden bzw. nächsten Befehl.
Die Tasten Pfeil-oben und Pfeil-unten erniedrigen bzw. erhöhen die Adresse um eins (Dies kann z.B. nötig sein, wenn ein Programm fehlerhaft disassembliert wird und eine Korrektur der Anfangsadresse nötig ist).Die Tasten ';' und '-' disassemblieren die vorhergehende bzw. nächste Seite, d.h. acht Befehle vor oder zurück.
BREAK verläßt den Disassembler-Modus.
Der RST 08H Befehl ist normalerweise ein 1-Byte-Befehl, wird jedoch im Basic benutzt, um einen Syntax-Test mit dem folgenden Byte auszuführen.
Dies wurde in diesem Monitor berücksichtigt, um Fehler beim Disassemblieren und Relozieren zu vermeiden.

S (xxxx (yyyy (zzzz (NAME)))) (RETURN)
SAVE
Dieser Befehl dient dazu, einen Speicherbereich im SYSTEM-Format auf Kassette zu schreiben.
Dabei sind xxxx die Anfangsadresse, yyyy die Endadresse und zzzz die Einsprungadresse (Entrypoint).
Der Name kann aus maximal 6 Zeichen bestehen.
Wenn die Parameter nicht angegeben werden, so werden die in der vierten Bildschirmzeile angezeigten Werte angenommen. Sie können also ein Programm mit S (RETURN) sehr einfach kopieren oder mehrmals aufzeichnen.

T (RETURN)
TEXT-EDITOR
Dieser Befehl führt zum Aufruf des Text-Editors
(s. Genie II S Text-Editor).

V (xxxx) (RETURN)
VIEW
Dieser Befehl dient zur Anzeige des Namens und der Ladebereiche eines unbekannten SYSTEM-Programmes.
Das Band wird gelesen, aber nicht in den Speicher geladen. Ansonsten wirkt der V-Befehl wie der L-Befehl.

X (xxxx (yyyy)) (RETURN)
HEX RECHNEN
Dieser Befehl zeigt die Summe und Differenz der angegebenen Argumente in hexadezimaler und dezimaler Form an. Er ist z.B. bei der Berechnung eines Offsets für den L-Befehl sehr nützlich.

Y (dddd (eeee)) (RETURN)
RECHNEN
Dieser Befehl funktioniert im Prinzip wie der X-Befehl mit dem Unterschied, daß die Argumente mit vier verschieden Zahlen-Basen angegeben werden können.
Als Zahlen-Basen sind zugelassen:
B-Binär, O-Oktal, D-Dezimal oder H-Hexadezimal. Wenn keine Basis angegeben ist, wird dezimal angenommen.
Beipsiel:
Y 12345D 010101018 (RETURN)

Z xxxx yyyy (zz) (RETURN)
ZERO
Dieser Befehl löscht den Speicherbereich von xxxx bis yyyy mit dem Wert zz. Wenn zz nicht angegeben ist, wird der Wert 0 angenommen.

Beachten Sie bei allen Befehlen, daß der Speicherbereich von 0000 bis FFFF nicht gelöscht oder überschrieben werden darf, da dort der Monitor und die anderen Programme liegen !
Wenn Sie diesen Bereich löschen oder überschreiben, so führt das normalerweise zur Zerstörung der Programme und Texte, die im Speicher stehen !

Wenn Sie gleichzeitig einen Text oder ein Assembler-Programm bearbeiten wollen, so vergewissern Sie sich, daß Sie den Bereich des Text-Buffers nicht überschreiben.

Sicherheitshalber sollten Sie den Text-Buffer vorher auf Kassette abspeichern.

Die genaue Aufteilung des Bildschirms und Ihre Bedeutung finden Sie auf der nun folgenden Seite erläutert.

```
M 6000 61FF 6400                                                    **
BC: 8820 DE: 4030 HL: D000 AF: C31E F: --HPN- IX: 4015 IY: 3000
BC' 0200 DE' 0134 HL' 3230 AF' 0040 F' -Z---- SP: 43E8 PC: D000
CFFF  00        NOP                          NAME:  VON:  BIS:  ENT:
D000  CDC901  -)CALL  01C9H                  TEST   4400  443F  4410
D003  21003D    LD    HL,3D00H               X:     Y:    X+Y:  X-Y:
D006  222040    LD    (4020H),HL             0001   FFFF  0000  0002
D009  219BD2    LD    HL,0D29BH              00001 65535 00000 00002
D00C  CDD7F3    CALL  0F3D7H
D00F  2AB140    LD    HL,(40B1H)
D012  ED5B5BD1  LD    DE,(0D15BH)
D016  DF        RST   18H                    Tape Error
D000  CDC9 0121 003D 2220:4021 9BD2 CDD7 F32A   ...!.=" `!ö.*...
D010  B140 ED5B 58D1 DFCA:61F1 2AFD 4011 0001   .`.ÄX.f.A.*.`...
D020  1922 58D1 22B1 4011:CEFF 1922 A040 22E8   ."X."."`...".`"..
D030  40C3 61F1 5FB7 0E47:656E 6965 2049 4920   `.A....Genie II
```

So könnte eine Bildschirmdarstellung bei Benutzung des Monitors zum Beispiel aussehen.
Bitte beachten Sie nicht die hier im Beispiel angezeigten Werte, da diese zufällig gewählt und nicht von Bedeutung sind.

Jedoch ist so am besten die Bildschirmaufteilung zu erkennen.

Die erste Bildschirmzeile ist die Kommandozeile.
Der Cursor markiert hier die Eingabeposition.

Die beiden folgenden Zeilen zeigen die Werte der zwischengespeicherten Registerinhalte an. Diese werden beim G-Befehl in die Register übertragen und bei einem Rücksprung über einen Breakpoint oder ein RET zurückgegeben und angezeigt.

In der vierten bis zwölften Zeile ist im linken Teil das Disassembler-Listing, wenn der Disassembler eingeschaltet ist, ab der augenblicklichen Adresse von PC zu sehen.
Die PC-Adresse selbst ist dann mit einem Pfeil (->) markiert.
Im Disassembler-Modus ist kein Pfeil zu sehen, da immer die erste disassemblierte Zeile die aktuelle Adresse anzeigt.

Im rechten Teil sind in den Zeilen vier und fünf die Werte für den Save-, Load- und View-Befehl zu sehen.

Darunter sind die Ergebnisse der Rechen-Befehle X und Y in dezimaler und hexadezimaler Form angezeigt.

In der zwölften Zeile rechts, werden Fehlermeldungen angezeigt (Parameter Error oder Tape Error).

In den letzten vier Zeilen sind 64 Bytes des Speichers in hexadezimaler- und in ASCII-Form angezeigt.

Wenn einmal der Bildschirm durch einen Programmaufruf 'unsauber' sein sollte, so drücken Sie einfach (CLEAR) und Sie sehen wieder klarer.

Dieses Bild gibt nur einen möglichen Zustand des Bildschirms wieder. Das Disassmbler-Listing , die Fehlermeldung und die Ergebnisse der X- und Y-Befehle müssen nicht immer angezeigt werden.

Wie nützlich ein ROM-residenter Monitor beim programmieren in Maschinensprache ist, werden Sie leicht feststellen können.

2. Die Tastaturbelegung im Text-Editor

2.1 Die Zeichentasten

Alle mit Buchstaben, Zahlen oder Sonderzeichen beschrifteten Tasten, erzeugen die aufgedruckten Zeichen. Jedoch sind nicht alle Zeichen auf den Drucker auszugeben.
Die Zeichen '^' und '_' dienen als Steuerbefehle für den Drucker.
Das Zeichen '^' erzeugt beim Ausdruck einen Escape-Code, der bei den meisten Druckern zur Steuerung von verschiedenen Drucker-Funtkionen verwendet wird.
Das Zeichen '_' dient zur Steuerung der Unterstreichung bei EPSON kompatiblen Druckern (dazu gehören STAR, GEMINI usw.).
Beim Drucken wird für ein '_'-Zeichen die Bytefolge
1B 2D 01 bzw. 1B 2D 00 ausgegeben (enstpr.: ESC - 1, ESC - 0).

Wenn Sie nun die Zeichen selbst ausgeben wollen, so benutzen Sie den Steuerbyte-Befehl (P1) G um die Werte einzugeben.
In dieser Anleitung, die auch mit dem Genie II S Text-Editor erstellt worden ist, wurden die Zeichen allerdings nachgemalt da der Typenrad-Drucker sie nicht im Zeichensatz hatte.
Beachten Sie, daß diese Steuer-Zeichen zwar im Bildschirm ein Zeichen belegen, aber nicht beim Ausdrucken.

2.2 Zusammenfassung der Funktionstasten

Die Pfeiltasten haben allein gedrückt die aufgedruckte Funktion. Also Pfeil-oben bewegt den Cursor in die vorhergehende Zeile, Pfeil-unten in die folgende usw.
Die (CLEAR)-Taste hat die Funktion HOME-Cursor, d.h. sie bewegt den Cursor in die linke, obere Bildschirmecke.
Die (BREAK)-Taste verzweigt, wie auch (P1) A, zur Kommando-Eingabe, die später noch erläutert wird.

Zusätzlich zu den normalen Tastenbelegungen sind eine Reihe von Funktionen über die (P1)-Taste erreichbar. Sie drücken zuerst die (P1))-Taste und dann die angegebene Taste, um die entsprechende Funktion aufzurufen.
Die Funktionen sind:

A - Kommando-Eingabe aufrufen bzw. abbrechen.
B - Block-Anfangsmarker an die Cursorposition.
C - festgelegten Block kopieren.
D - Zeichen an der Cursorposition löschen.
E - Block-Endmarker an die Cursorposition.
F - Find-Kommando wiederholen.
G - Steuerbyte Eingabe.
I - Leerzeichen an der Cursorposition einfügen.
L - festgelegten Block löschen.
N - Eine Seite vorwärts.
O - Eine Seite zurück.
P - System-Parameter anzeigen.
Q - Systemvariablen wiederherstellen.
R - Replace-Kommando wiederholen.
S - Insert-Modus ein-/ausschalten.
T - Leerzeichen ab Cursorposition bis nächstes Zeichen löschen.
U - Umbruch-Modus ein-/ausschalten.
V - festgelegten Block verschieben.
W - vorhergehende Textzeile wiederholen.
X - Zeile ab Cursorposition bis (CR) löschen.
Y - Block-Marker löschen.
Z - Block-Zielmarker an die Cursorposition.

zu c)
Normalerweise, d.h. beim ersten Aufruf des Text-Editors, erscheint beim Druck auf eine Buchstaben-Taste der entsprechende Kleinbuchstabe. Die Großbuchstaben sind über die SHIFT-Tasten zu erreichen. Für die normale Texterstellung ist dies nützlich, da Sie es wahrscheinlich von der Schreibmaschine her so kennen.
Sie können jedoch die Funktion der Schift-Taste im Bezug auf die Buchstaben-Tasten umkehren.
Dies geschieht mittels einmaligem Drücken von (SHIFT) 0.
Danach erscheint beim Druck einer Buchstaben-Taste der Großbuchstabe und die (SHIFT)-Taste erzeugt die Kleinbuchstaben.

zu d)
Die INSERT-Funktion wird mit (P1) S ein bzw. ausgeschaltet.
Der Insert-Modus wird mit dem '*' in der Statuszeile und durch einen größeren Cursor im Text-Fenster dargestellt.
Die Insert-Funktion bewirkt, daß ein nachfolgender Text beim Schreiben nicht überschrieben sondern weitergeschoben wird.

zu e)
Die UMBRUCH-Funktion wird mit (P1) U ein und ausgeschaltet.
Der Umbruch-Modus ist bei der Texterstellung meistens nützlich.
Die Funktion bewirkt, daß ein Wort das über den rechten Rand der Cursorzeile geschrieben würde, in die nächste Zeile übernommen wird.
Wenn Sie allerdings im INSERT-Modus arbeiten, so kann es sein, daß die nächste Zeile auch "voll" ist und dort können überstehende Worte dann nicht in die folgende Zeile übernommen werden, da sonst der ganze folgende Text verschoben würde.

Beim Einfügen sollten Sie also den UMBRUCH-Modus ausschalten.
Meistens werden Sie jedoch einen Text "von oben nach unten" schreiben und dann können Sie den automatischen Umbruch sicher gebrauchen.

zu f)
Der AUTO-REPEAT ist beim ersten Aufruf des Text-Editors immer eingeschaltet. Er bewirkt die automatische Wiederholung einer längere Zeit festgehaltenen Taste. Die Wiederhol-Funktion ist mit der Taste (P2) ein-/ausschaltbar.

1.2 Die Tabulatorzeile

Die Tabulatorzeile zeigt an jeder achten Bildschirmspalte ein Kreuz. Diese Kreuze zeigen die jeweils nächste Cursorposition an, die bei der Eingabe von (SHIFT) (SPACE) erreicht wird.
Der Tabulator wird noch näher erläutert ("2.7 Cursorsteuerung").
Wenn bei der Kommando-Eingabe oder im Assembler Fehler auftreten, so werden diese in der Tabulatorzeile angezeigt bis Sie eine Taste drücken.

1.3 Das Textfenster

Die unteren vierzehn der sechzehn Bildschirmzeilen sind ein Fenster auf den Textspeicher. Der Cursor bleibt immer innerhalb dieses Fensters (außer bei der Kommando-Eingabe).
Wenn Sie mit dem Cursor dieses Fenster nach oben oder unten verlassen wollen, dann wird das Fenster über den nächsten Textausschnitt verschoben. Wenn Sie in der ersten Zeile die Taste Pfeil-oben drücken, dann bleibt der Cursor allerdings in der ersten Textzeile, erscheint aber auf dem ersten Zeichen.

Der CL-Befehl hat sehr schwerwiegende Folgen, und wurde aus diesem Grund nur als Kommando vorgesehen.
Er löscht den Textspeicher ab der momentanen Cursorposition bis zum Text-Ende. Wenn Sie also den kompletten Textspeicher löschen wollen, so geben Sie (SHIFT) ↑ und dann (BREAK) CL (RETURN) ein.

Der CF-Befehl ändert die Blinkfrequenz des Cursors. Mit dem eingegebenen Wert wird der Wert einer Zählschleife mit logisch AND verknüpft. Wenn das Ergebnis ungleich Null ist, dann ist das Cursorzeichen, sonst das Textzeichen zu sehen.
Das heißt bei CF=0 ist kein Cursor zu sehen und bei CF=255 ist das Cursorzeichen immer eingeschaltet.

Der CC-Befehl ändert den ASCII-Wert des Zeichens, das als Cursor verwendet werden soll. Sie können z.B. auch 5FH , 7FH oder 8CH usw. verwenden.

Der DL-Befehl ändert den Wert für eine Zählschleife in der Tastaturabfrage. Er bestimmt die Zeit, die vom Niederdrücken einer Taste bis zum Beginn des AUTO-REPEAT vergeht.
Der Wert 34H ist ein meistens angenehmer Wert und wurde deshalb als Vorbelegung gewählt.

Der F-Befehl sucht ab der derzeitigen Cursorposition nach einem angegebenen Text. Der Text wird nach einem Trennzeichen hinter dem F (z.B. Leerzeichen) eingegeben und mit (RETURN) abgeschlossen. Ein (CR) können Sie mit (SHIFT) (RETURN) und einen Tabulator mit (SHIFT) (SPACE) eingeben.
Wenn der String gefunden wird, so erscheint der Cursor auf dem ersten Zeichen des Strings im Textspeicher.
Andernfalls bleibt er an der Position die er vorher hatte.
F(RETURN) oder (P1) F wiederholen den Befehl mit demselben String ab der neuen Cursorposition.

Der K-Befehl belegt eine der Tasten (P1) 0 bis (P1) 9 mit einem neuen Text. Ein (CR) können Sie mit (SHIFT) (CR) und einen Tabulator mit (SHIFT) (SPACE) eingeben.
Die Belegung der anderen Text-Tasten wird entsprechend verschoben. Wenn der Text zu lang für die 256 reservierten Bytes ist, dann wird die vorherige Belegung beibehalten.
Der Befehl dient zur Vereinfachung der Eingabe von häufig auftretenden Worten oder Phrasen.

Der L-Befehl lädt den nächsten auf Kassette abgespeicherten Text in den Textspeicher. Dabei bleibt der alte Text erhalten und der neue wird hintenan geladen.
Auf diese Weise können Sie meherere Texte zu einem Stück zusammenfügen und komplett bearbeiten.
Wenn der Text nicht innerhalb der Textbuffergrenzen unterzubringen ist, so wird der Ladevorgang abgebrochen und der Text an der zuletzt eingelesenen Position beendet.
Wenn Sie einen neuen Text laden wollen, dann geben Sie vorher (SHIFT) ↑ und (BREAK) CL (RETURN) ein.
Die Belegung der Text-Tasten wird ebenfalls eingeladen.

Der M-Befehl führt zum Aufruf des Monitors. Sie können dort z.B. ein gerade assembliertes Programm ausführen oder ändern und andere Programme laden oder abspeichern.
Wenn Sie den Textspeicher dabei nicht löschen, dann ist der Text bei der Rückkehr mit dem T-Befehl des Monitors noch erhalten.

↑ - Eine Seite zurück.
↓ - Eine Seite vorwärts.
0 - Text: "Meine sehr geehrten Damen und Herren,"(CR)(CR)
1 - Text: "Sehr geehrte Frau "
2 - Text: "Sehr geehrter Herr "
3 - Text: "Herr/Frau/Firma"(CR)(CR)
4 - Text: "Vielen Dank für "
5 - Text: "   mit freundlichen Grüßen"(CR)(CR)
          "   "
6 - Text: "Absender :"(CR)
7 - Text: "Bankverbindung "
8 - Text: "Postfach "
9 - Text: " den 01.01.1984 "

Außerdem gibt es noch einige Funktionen zusammen
mit der (SHIFT)-Taste:

↑ - Cursorposition = erste Textzeile.
↓ - Cursorposition = letzte Textzeile.
← - Cursorposition = erstes Zeichen des vorherigen Wortes.
→ - Cursorposition = erstes Zeichen des folgenden Wortes.

2.3 Befehle der Kommando-Eingabe

Die (BREAK)-Taste oder (P1) A führen zu einem Aufruf der
Kommando-Eingabe.
Dabei erscheint in der Statuszeile folgendes:

KOMMANDO:.....

Hier können Sie nun verschieden Kommandos ausführen, die
Argumente benötigen oder sicherheitshalber in die Kommandozeile
gelegt wurden.
Bei den Befehlserklärungen bedeutet:

n	ein Byte (z.B. 127, 127D, 7FH, 01111111B oder 177O)
ww	ein Wort (z.B. 6000H, 24576, 24576D usw.)
b	eine Ziffer zwischen 0 und 9.
xxxx	ein String (Zeichenkette) aus Buchstaben, Zahlen oder
yyyy	Sonderzeichen.
)	ein beliebiges Trennungszeichen.

Die Befehle sind:

A, AL od. AP	Assembler aufrufen (s. Genie II S Assembler)
CL	Textspeicher ab Cursorposition löschen.
CF=n	Blinkfrequenz des Cursors ändern.
CC=n	Cursorzeichen ändern.
DL=n	Delay-Wert für Tastatur ändern.
F)xxxx	String ab Cursorposition suchen.
Kb=xxxx	Text-Taste b mit String xxxx belegen.
L	Text von Kassette laden.
M	Monitor aufrufen.
ML=ww	Memory low (untere Speichergrenze) ändern.
MH=ww	Memory high (obere Speichergrenze) ändern.
P	Textspeicher komplett ausdrucken.
PB	festgelegten Block ausdrucken.
PL=n	Papierlänge in Zeilen ändern.
R)xxxx)yyyy	String xxxx durch yyyy ersetzen.
RL=n	Linken Rand für Druckerausgabe ändern.
RP=n	Tastatur Repeat-Frequenz festlegen.
S)(NAME)	Textspeicher auf Kassette abspeichern.
WT=ww	Wait-Zähler für Tastaturabfrage ändern.

Sie laden diesen Brief dann nur von Kassette und geben z.B. ein:
(BREAK)
KOMMANDO:R/$VNAME$/Franz(CR)
und drücken dann sooft wie nötig (P1) R.
Anschließend drücken Sie (SHIFT) ↑ und wieder
(BREAK)
KOMMANDO:R/$NNAME$/Müller(CR)
und entsprechend verfahren Sie mit den anderen **Platzhaltern.**

Diese Methode lohnt sich sicher nur bei längeren **Texten oder** in Assembler Programmen (z.B. Labels umbenennen).

Der R-Befehl dient auch zum **löschen** bestimmter Worte oder Sätze im Text. Dazu geben Sie einfach als zweiten String (hinter dem Trennzeichen) einen Leerstring also nur (CR) **ein.**

Aber Sie können auch einen Leerstring durch etwas **ersetzen, was** zunächst seltsam erscheinen mag und im Prinzip auch ist.
Sie wollen eine Textstelle z.B. so schreiben:

A U S E I N A N D E R

Sie geben dann einfach den Text "AUSEINANDER" ein und drücken (SHIFT) ← . Dann drücken Sie (BREAK) und R// (CR). Sie setzen also jeweils ein Leerzeichen an die Cursorposition.
(Denn ein Leerstring ist an jeder Textstelle vorhanden !).
Mit den Tasten (P1) R können Sie jetzt das ganze Wort mit Zwischenräumen versehen.

Diese Anwendungen sind nur einige Beispiele für **die vielen** Möglichkeiten, die der Replace-Befehl bietet.

Der RL-Befehl legt die Anzahl der Leerzeichen für den linken Rand bei der Druckerausgabe fest. Der voreingestellte Wert von acht ergibt bei einem Drucker mit 80 Zeichen/Zeile ein mittiges Schriftbild. Wenn Sie einen breiteren Drucker haben, oder an einer bestimmten Stelle des Papiers beginnen wollen, so können Sie den Wert zwischen 0 und 255 **frei** wählen.

Der RP-Befehl ändert die Repeat-Frequenz des AUTO-REPEAT.
Das ist die Zeit die zwischen zwei Wiederholungen einer Taste vergehen soll.

Der S-Befehl speichert den Text aus dem Textbuffer auf Kassette ab. Dabei wird auch die Belegung der Text-Tasten mit abgespeichert. Wenn Sie einen Namen angeben, so darf dieser maximal 16 Zeichen lang sein. Wenn Sie keinen Namen angeben, so wird der letzte benutzte Name verwendet.
Die Abspeicherung beginnt direkt nachdem Sie (RETURN) drücken.
Schalten Sie also den Rekorder vorher auf RECORD & PLAY, warten Sie den Bandleerlauf ab und geben Sie dann erst (RETURN) ein.

Der WT-Befehl ändert den Wert für die Zählschleife zur Tastaturentprellung. Wenn Sie wieder erwarten Probleme mit Tastaturprellen haben, so können Sie den Wert entsprechend nach oben oder nach unten anpassen.
Sie sollten den Wert nicht zu klein wählen, da Sie u.U. sonst keine sinnvolle Eingabe mehr machen können. In einem solchen Falle hilft die Funktion (P1) Q die alten Werte wieder herzustellen.

Der ML-Befehl erlaubt es, die untere Grenze des Textbuffers zu ändern. Ein dabei schon vorhandener Text bleibt erhalten und wird entsprechend verschoben. Die **niedrigste**, zulässige Adresse ist **42E8H**. Die **höchste** zulässige Adresse ist **MH -1**. Wenn der schon vorhandene Text nicnt in die neuen Speicher-Grenzen passen würde, so bleibt der alte ML-Wert erhalten und es wird eine Fehlermeldung ausgegeben.

Der MH-Befehl legt die obere Grenze des Textbuffers fest. **Maximale** Adresse ist **CFFFH**. Sie können so z.B. einen Bereich für Daten oder ein anderes Programm freihalten, das nicht von dem eingegebenen Text überschrieben werden soll. MH kann selbstverständlich **minimal ML +1** sein.

Der P-Befehl druckt den Inhalt des Textbuffers aus. Dabei wird ein linker Rand wie mit RL angegeben gelassen. Die Papierlänge PL und die Anzahl der Zeile pro Seite ZS werden zur Formatierung des Ausdrucks verwendet. Wenn Sie die Seiten-formatierung nicht benötigen oder dem Drucker überlassen wollen, so setzen Sie ZS auf denselben Wert wie PL und RL auf Null.

Der PB-Befehl druckt den Text innerhalb des angegebenen Blockes aus. Auch hierbei wird ein linker Rand gelassen und die Seitenformatierung wird vorgenommen.

Der PL-Befehl legt die Anzahl der Zeilen fest, die eine Seite Druckerpapier hat. (Bei 1/6 inch und Traktorpapier sind dies meistens 72 Zeilen). Wenn Sie anderes Papier benutzen oder einen andere Einstellung des Druckers haben (s. Druckerhandbuch), dann können Sie den Wert entsprechend ändern.

Der R-Befehl gestattet es, einen String im Textbuffer durch einen anderen zu ersetzen. Die Strings müssen dabei nicht die-selbe Länge haben und können auch leer sein (vgl. Leerstrings im BASIC). Das Trennungszeichen zwischen R und dem ersten Textstring dient dann als Endmarkierung des ersten Textstrings. Das Zeichen ist beliebig, darf jedoch im ersten String nicht nocheinmal auftauchen.
Sie können mit diesem Befehl z.B. einen Standarbrief von Kassette für die jeweilige Anwendung modifizieren.
Hierzu ein Beipsiel:

```
Herr/Frau/Firma                          Ungenannt & Co.
$VNAME$ $NNAME$                          Waldweg 32
$STRASSE$ $NR$                           5432 Nirgendwo
$PLZ$ $ORT$                              Tel.: 1234/23456

Sehr geehrter Herr $NNAME$

Wir freuen uns Ihnen mitteilen zu können, daß Sie in
unserem Preisausschreiben vom $DATUM$ gewonnen haben.
Wir hoffen, daß Ihnen ihr $GEWINN$ gefallen wird.

                    Mit freundlichen Grüßen

                    Ungenannt & Co.
```

2.6 Die Parameter- und Modus-Funktionen

Die Funktion (P1) P zeigt alle änderbaren System-Variablen auf dem Bildschirm an. Mit einem beliebigen Tastendruck kehren Sie zur normalen Eingabe zurück.

Es bietet sich folgendes Bild:

```
ML= 43EBH
MH= CFFFH
PL= 072
ZS= 066
RL= 008
DL= 34H
RP= 06H
WT= 0500H
CF= 10H
CC= 8FH
```

Die Abkürzungen vor den Parametern sind dieselben, wie sie in der Kommandoeingabe verwendet werden. Sie können mit (P1) P also jederzeit die Werte der Systemvariablen anzeigen lassen.

Die Funktion (P1) Q setzt folgende Parameter auf ihren Anfangswert zurück:

PL=72 ZS=66 RL=8 DL=34H RP=06H WT=500H CF=20H CC=8FH

Sie ist für den Notfall gedacht, wenn Sie einmal zuviele Parameter verändert haben sollten.

2.7 Die Steuerbyte Funktion

Die Funktion (P1) G dient zur Eingabe von Steuerzeichen oder auf dem Bildschirm nicht darstellbaren Zeichen für den Drucker.
Wenn Sie (P1) G gedrückt haben, wird ein Grafikzeichen'■' gefolgt von zwei Nullen an der Cursorposition eingefügt. Der Cursor blinkt über der linken Null und fordert Sie zur Eingabe einer zweistelligen Hexadzimalzahl auf (0..9,A..F).

Die drei Zeichen im Bildschirm werden bei der Cursorsteuerung wie ein Zeichen behandelt. Deswegen kann in einer Zeile mit einem oder mehreren Steuerbytes, die nicht mit einem (CR) endet die Formatierung beim Drucken mißlingen. Sorgen Sie also immer dafür, daß eine solche Zeile mit einem (CR) abschließt.

Wenn Sie das Steuerbyte mit einer Taste überschreiben, so wird der ASCII-Wert der Taste in dem Byte abgelegt.
Wenn Sie das Steuerbyte löschen, dann verschwinden alle drei Zeichen, also das Grafikzeichen und die beiden Hex-Ziffern.

Mit Hilfe dieser Funktion können Sie alle Werte zwischen 00H und FFH an ihren Drucker schicken und diesen steuern.

2.4 Die Blockmodus-Funktionen

Die Funktionen (P1) B, (P1) C, (P1) E, (P1) L,
(P1) V, (P1) Y und (P1) Z sind Blockmodusfunktionen.
Die Funktion (P1) B legt den Beginn eines Blockes fest.
Die Stelle wird im Text mit '[redacted]' markiert und das Textzeichen
verschwindet "hinter" der Markierung.
Die Funktion (P1) E legt das Ende eines Blockes fest
Die Stelle wird im Text mit '[redacted]' markiert.
Die Funktion (P1) Z legt das Ziel eines Blockes fest.
Die Stelle wird im Text mit '[redacted]' markiert.

Es ist verständlich, daß der Blcokbeginn vor dem Blockende
liegen muß, und daß das Ziel nur außerhalb des Blockes sein
kann. Alle anderweitigen Eingaben werden daher von vornherein
nicht akzeptiert.

Die Funktion (P1) C kopiert den Block von einschließlich
der Stelle '[redacted]' bis ausschließlich der Stelle '[redacted]' an die
mit '[redacted]' markierte Stelle. Die Marker bleiben erhalten, um
den Befehl wiederholen zu können.

Die Funktion (P1) L löscht den Block zwischen '[redacted]' und
'[redacted]'. Der Cursor erscheint anschließend an der Position des
alten Blockanfangs.

Die Funktion (P1) V verschiebt den Blcok zwischen '[redacted]' und
'[redacted]' an die mit '[redacted]' markierte Stelle. Sie führt also die
Funktionen (P1) C und (P1) L nacheinander aus.

Die Funktion (P1) Y löscht die gesetzten Blockmarker. Sie
können so z.B. nach (P1) C die Marker wieder löschen.

2.5 Die Funktionen für Löschen und Einfügen

Die Funktion (P1) D löscht ein Zeichen an der Cursor-
position. Der Rest der Zeile wird 'herangezogen'.

Die Funktion (P1) I fügt ein Leerzeichen an der Cursor-
position ein. Der Rest der Zeile wird 'weitergeschoben'.

Die Funktion (P1) T löscht alle Leerzeichen ab der Cursor-
position bis zum nächsten Zeichen oder bis zum (CR).

Die Funktion (P1) W wiederholt die vor der Cursorzeile
stehende Textzeile und fügt sie vor der Cursorzeile ein.

Die Funktion (P1) X löscht die Zeile ab der Cursorposition
bis einschließlich zum nächsten (CR).

Die (RETURN)-Taste fügt auf jeden Fall ein (CR) ein. Das heißt
ein Buchstabe wird beim drücken von (RETURN) nicht überschrieben
sondern an den Anfang der nächsten Zeile genommen.

3. Die Speicherbelgung des Text-Editors

3.1 Das Textformat im Speicher

Der Text ist generell im ASCII-Format im Speicher abgelegt.
Das heißt ein 'A' entspricht einem 41H oder 65D im Speicher.
Es gibt jedoch einige Ausnahmen.

Die unter 2.6 schon angesprochenen Steuerbytes sind in der Form B7H, n abgespeichert. n ist dabei das angegebene Byte.
Dies ist zur Unterscheidung der Steuerbytes von internen Steuerzeichen, wie (CR) (TAB) usw., notwendig.
Das Byte 09H dient als Tabulator-Zeichen.
Bei der Bildschirmausgabe oder beim Drucken werden hierfür ein bis acht Leerzeichen ausgegeben.
Das Byte 0DH (Carriage Return) dient als Endmarkierung einer logische Zeile. Deren Länge kann größer als eine Bildschirmzeile sein, wenn sie aus mehr als 64 Zeichen besteht oder z.B. Steuerbytes oder Tabulatoren enthält.
Das Byte 03H (entspr. ETX = End Of Text) dient als Markierung des Text-Endes. Es ist das Byte, das bei Kassettenoperationen das Ende des zu ladenden Textes markiert.
Alle anderen Zeichen, die die über Tastatur erreichbar sind, sind mit ihren ASCII-Werten im Speicher abgelegt.

3.2 Das Textformat auf Kassette

Ein Text-File auf Kassette beginnt, wie auch SYSTEM- und CLOAD-Bänder, mit einem Block von 255 mal 00H. Danach folgt ein Synchronisationsbyte mit dem Wert A5H.
Bei SYSTEM-Programmen würde jetzt ein 'U' = 55H folgen und bei BASIC-Programmen dreimal D3H.
Zur Unterscheidung von diesen Bändern beginnt ein Text-File mit 'T' = 54H.

Wenn dieses Byte nicht zu Beginn gefunden wird, wird eine Fehlermeldung angezeigt.
Danach folgt der Name des Textes mit 16 Zeichen im ASCII-Format.
Direkt anschließend sind 256 Bytes abgespeichert, die in den Bereich der Texte für die Text-Tasten geladen werden.
Nun folgt der eigentliche Text, wie er im Speicher stand.
Wenn ein Byte 03H gefunden wird, dem kein Byte B7H vorausging, ist das Textende erreicht.

Wenn Sie während des Ladevorgangs (BREAK) drücken, so wird der Text an der gerade geladenen Stelle beendet und der Ladevorgang abgebrochen. Diese Funktion erlaubt es, wenn auch ein wenig umständlich, nur einen Teil eines Textes zu laden.
Sie ist eigentlich dafür vorgesehen, den Ladevorgang abzubrechen, falls ein falsches Band eingelegt oder ein anderer Fehler begangen wurde.

Als letztes Byte auf der Kassette folgt eine 1-Byte Prüfsumme.
Sie wird durch Addition aller Text-Zeichen mit Ausnahme der Texte für die Text-Tasten und der 03H am Schluß gebildet.
Wenn die beim Laden errechnete und die zum Schluß geladene Prüfsumme nicht übereinstimmen, wird eine Fehlermeldung ausgegeben. Sie müssen dann versuchen, entweder den oder die Fehler zu finden oder, wenn zu viele Fehler vorhanden sind, den Text im Speicher zu löschen und mit einer anderen Lautstärke Einstellung noch einmal zu laden.

2.7 Cursorsteuerung

Der Cursor wird beim ersten Aufruf des Text-Editors durch das Grafikzeichen ' ' dargestellt.
Der Cursor kann mit folgenden Tasten bewegt werden:

←	- Ein Zeichen nach links.
→	- Ein Zeichen nach rechts.
↑	- Eine Zeile zurück.
↓	- Eine Zeile vorwärts.
(SHIFT)	- Ein Wort zurück.
(SHIFT)	- Ein Wort vor.
(P1)↑/(P1) O	- Eine Seite zurück.
(P1)↓/(P1) N	- Eine Seite vor.
(SHIFT) ↑	- Erste Text-Zeile, erstes Zeichen.
(SHIFT) ↓	- Letzte Text-Zeile, letztes Zeichen.

Ausserdem ändert sich seine Position bei den meisten Kommandos, sowie bei den meisten Funktions-Aufrufen.

Wenn der Cursor in die vorhergehende Bildschirmzeile bewegt wird, so gibt es zwei Möglichkeiten, wo er erscheinen kann. Wenn die Zeile mit einem CR abschließt, so steht der Cursor am Ende der Zeile auf diesem CR. Andernfalls blinkt er genau 64 Zeichen über seiner alten Position.

Wenn der Cursor in die nachfolgende Bildschirmzeile bewegt wird, so gibt es ebenfalls zwei Möglichkeiten. Wenn die Ausgangszeile mit einem CR abschließt, so erscheint der Cursor am Anfang der nächsten Zeile. Andernfalls ist er genau 64 Zeichen unter seiner alten Position.

Diese Eigenschaften können Sie ausnutzen, wenn Sie schnell ans Ende oder an den Anfang einer Zeile möchten.
Um an das Ende einer Zeile zu gelangen,
drücken Sie nacheinander die Pfeil-Tasten '↓' und '↑'.
Um an den Anfang zu gelangen, drücken Sie nacheinander die Pfeil-Tasten '↑' und '↓'.
Dadurch wurden zwei zusätzliche Funktionen gespart.

Die Cursorbewegung mit (SHIFT) '←' und (SHIFT) '→' ermöglicht ebenfalls ein schnelles Erreichen der gewünschten Position.
Wenn Sie z.B. ans Ende eines Wortes wollen, so halten Sie solange (SHIFT) '→' gedrückt, bis Sie auf dem ersten Zeichen des nächsten Wortes sind. Dann können Sie mit '←' die Position einfach erreichen.

In diesem Zusammenhang noch ein Hinweis zu den Tabulatoren.
Ein Tabulator besteht aus einem Zeichen.
Dies ist z.B. in Assembler-Programmen platzsparend aber dennoch übersichtlich.

Aber obwohl bis zu 8 Leerzeichen für einen Tabulator ausgegeben werden, können Sie den Cursor nur auf das erste Leerzeichen oder hinter das letzte Leerzeichen des Tabulators bewegen.
Wenn Sie den Tabulator löschen (auf dem ersten Zeichen), dann verschwinden sämtliche Leerzeichen des Tabulators.
Ähnliches gilt ja auch für das Steuerbyte-Zeichen ((P1) G).

Ebenfalls benutzt wird der Bereich des Basic-Input-Buffers ab Adresse 41E8H.
Dort werden die Find- und Replace-Strings von den entsprechenden Befehlen abgelegt.
Wenn Sie ML auf 42E8H setzen, sind zwar weniger als 256 Bytes für den Stack-Bereich frei, aber auch dies reicht im Normalfall noch aus.

Ansonsten sind alle Speicherbereiche von 42E8H bis Memory low, von Memory high bis CFFFH und von 3000H bis 33FFH zu benutzen.

Auf jedenfall sollten Sie, bevor Sie Eingriffe in diese Speicherbereiche vornehmen oder z.B. ein Programm assemblieren und testen wollen, den Text sicherheitshalber vorher abspeichern !
Die Erfahrung zeigt, daß das Warten beim abspeichern auf Kassette insgesamt kürzer ist als die Zeit, die für die neue Eingabe eines Textes oder eines Programmes nötig ist.

Kritische Anmerkungen zum Genie II S Text-Editor

Die Möglichkeiten des Genie II S wurden, was zum Beispiel die Verarbeitungsgeschwindigkeit angeht, voll ausgenutzt.
Dennoch ist z.B. das Schreiben im Insert-Modus am Anfang eines sehr langen Textes nicht mehr mit dem Zehn-Finger-System möglich.
Es wurde auf eine ausgefeilte Art der Textspeicherung, wie sie größere Textsysteme vornehemen, verzichtet.
Auch die Fehlermeldungen mußten auf ein Mindestmaß beschränkt werden, da der Speicherplatz doch sehr knapp war.

Sie müssen bedenken, daß der Z80-Assembler, der Monitor und das Textsystem in nur 12 kByte Speicher Platz finden mußten.
Dafür haben Sie gegenüber anderen Computern den Vorteil, daß alle diese Programme sofort verfügbar sind und nicht erst von Kassette oder Diskette geladen werden müssen !

Wir hoffen, daß mit dieser Anleitung zum Genie II S Text-Editor eine Einführung geglückt ist, die Sie zur vollen Ausnutzung seiner Möglichkeiten befähigt.

3.3 Das Format der Texte für die Text-Tasten

Für die Belegung der Text-Tasten ist ein Speicherbereich von 256 Bytes am Beginn des Programms reserviert.
Der Text, der bei (P1) 0 ausgegeben wird ist der erste im Speicher. Er muß mit einem Byte 00H enden. Danach beginnt der nächste Text für die Taste (P1) 1 usw.
Nach dem letzten Byte 00H für (P1) 9, muß noch ein Byte FFH folgen, das benötigt wird, um bei Benutzung des Kn-Befehls das Ende der Text-Tasten zu finden und den bisher verbrauchten Platz zu berechnen.

Wenn eine Text-Taste keinen Text erzeugen soll, so muß an der entsprechenden Position der Tabelle nur eine 00H stehen.
Sie können also maximal 245 Zeichen auf eine Text-Taste legen (256 minus 1 mal FFH minus 10 mal 00H).
Da in der Kommandozeile die Eingabe von Steuerzeichen außer (CR) und (TAB) nicht möglich ist, müssen Sie, wenn Sie eine Text-Taste damit belegen wollen, dies mit Hilfe des Monitors vornehmen.
Hierzu ein Beispiel :
Sie wollen auf die Taste (P1) 0 ein (ESC) 7 02H, ein (TAB) und den Text "ENDE" ablegen.
Dies sind insgesamt 9 Bytes.
7F 37 B7 02 09 45 4D 44 45
Geben Sie also ein :
(BREAK)
K0=:::::::::(RETURN)
Jetzt legt das Textsystem neun Bytes mit dem Wert 3AH ab und verschiebt die restlichen Belegungen entsprechend.

Drücken Sie nun (BREAK) M (RETURN), um in den Monitor zu gelangen. Dort geben Sie dann H 0000 ein, suchen die Doppelpunkte (im ASCII-Dump) und tippen die oben aufgelisteten Bytes ein. (Bei Texten können Sie mit (CLEAR) auch in denASCII-Edit Modus umschalten).
Wenn dies alles getan ist, drücken Sie (BREAK) T (RETURN).
Sie sind dann wieder im Textsystem und können mit (P1) 0 die gewünschte Funktion aufrufen.

Beachten Sie aber immer, daß Sie nicht mehr als 256 Byte verändern dürfen, da anschließend an die Text-Tasten das Programm und Variablen beginnen.

3.4 Die benutzten Adreßbereiche

Der Text-Editor benutzt hauptsächlich die Variablen innerhalb des Bereiches von 0000H bis FFFFH.
Dazu gehören z.B. alle Zeiger auf Cursorposition, die Block-Markeradresse, der Programmname usw.
Jedoch liegt der Bereich des Stackpointers (SP-Register) während des Arbeitens mit dem Text-Editor direkt vor der System-Variablen ML (Memory low).
Da 256 Bytes für den Stackpointerbereich genügend sind, wird auch beim Start des Text-Editors ML auf die Adresse hinter der letzten Basic-Feldvariablen plus 256 gesetzt.
Dies ist zu beachten, wenn Sie ML ändern wollen.
Bei der Anleitung zum Gebrauch des Assemblers wird noch auf die benutzbaren Speicherbereiche eingegangen.

BIT	b,8 b,C b,D b,E b,H b,L b,(HL) b,A b,(IX+n) b,(IY+n)
RES	b,8 b,C b,D b,E b,H b,L b,(HL) b,A b,(IX+n) b,(IY+n)
SET	b,8 b,C b,D b,E b,H b,L b,(HL) b,A b,(IX+n) b,(IY+n) (b = 00H 01H 02H 03H 04H 05H 06H 07H)
JP	ww NZ,ww Z,ww NC,ww C,ww PO,ww PE,ww P,ww M,ww (HL) (IX) (IY)
CALL	ww NZ,ww Z,ww NC,ww C,ww PO,ww PE,ww P,ww M,ww
RET	NZ Z NC C PO PE P M
RST	n (n = 00H 08H 10H 18H 20H 28H 30H 38H)
JR	ww NZ,ww Z,ww NC,ww C,ww
DJNZ	ww
PUSH	BC DE HL AF IX IY
POP	BC DE HL AF IX IY
IM	n (n = 00H 01H 02H)

LD

	B	C	D	E	H	L	(HL)	A	(IX)	(IY)	(BC)	(DE)	R	I	(ww)	n
B,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
C,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
D,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
E,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
H,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
L,	*	*	*	*	*	*	*	*	*	*	-	-	-	-	-	*
(HL),	*	*	*	*	*	*	-	*	-	-	-	-	-	-	-	*
A,	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(IX+n),	*	*	*	*	*	*	-	*	-	-	-	-	-	-	-	*
(IY+n),	*	*	*	*	*	*	-	*	-	-	-	-	-	-	-	*
(BC),	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
(DE),	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
R,	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
I,	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-
(nn),	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-

(* = zulässige Adressierungsart, - = unzulässig)

LD	BC,ww BC,(ww) (ww),BC DE,ww DE,(ww) (ww),DE HL,ww HL,(ww) (ww),HL SP,ww SP,(ww) (ww),SP SP,HL SP,IX SP,IY IX,ww IX,(ww) (ww),IX IY,ww IY,(ww) (ww),IY
EX	AF,AF' DE,HL (SP),HL (SP),IX (SP),IY
IN	A,(n) B,(C) C,(C) D,(C) E,(C) H,(C) L,(C) A,(C)
OUT	(n),A (C),B (C),C (C),D (C),E (C),H (C),L (C),A

Befehle ohne Argumente

RLCA	RRCA	RLA	RRA	DAA	CPL	SCF	CCF
NEG	EXX	RETI	RETN	EI	DI	NOP	HALT
LDI	LDD	LDIR	LDDR	CPI	CPD	CPIR	CPDR
INI	IND	INIR	INDR	OUTI	OUTD	OTIR	OTDR

Der Assembler wird vom Text-Editor aus mit den Befehlen A, AL oder AP aufgerufen.
Der Befehl A assembliert den Textbuffer, ohne ein Listing im zweiten Durchgang (Pass 2) auszugeben.
Der Befehl AL listet das Programm im zweiten Durchgang auf den Bildschirm und AP druckt es außerdem auf den Drucker aus.

Beim Ausdruck wird eine Seitenformatierung mit den Parametern ZS und PL des Text-Editors vorgenommen (s. ZS- und PL-Befehl).

Der Assembler ist ein Zwei-Pass Assembler.
Im ersten Durchgang wird eine Tabelle aller Symbole (Labels) angelegt, in der die Symbol-Adressen abgelegt werden.
Dabei werden doppelt definierte Symbole erkannt.

Im zweiten Durchgang werden die Werte für die Symbole der Tabelle entnommen und im enstehenden Programm eingesetzt.

Das Programm wird im zweiten Durchgang entweder hinter der Symboltabelle in einem Buffer abgelegt, oder, wenn Sie die Option .OBJECT gewählt haben, direkt ab der Adresse abgelegt, die durch das Origin (ORG-Befehl) festgelegt wurde.

Zunächst nun eine Liste aller zulässigen Befehle
und der mit ihnen zulässigen Adressierungsarten:

Befehl	Adressierungsarten
ADD	A,n A,B A,C A,D A,E A,H A,L A,(HL) A,A
	A,(IX+n) A,(IY+n)
	HL,BC HL,DE HL,HL HL,SP
	IX,BC IX,DE IX,IX IX,SP
	IY,BC IY,DE IY,IY IY,SP
ADC	A,n A,B A,C A,D A,E A,H A,L A,(HL) A,A
	A,(IX+n) A,(IY+n)
	HL,BC HL,DE HL,HL HL,SP
SUB	n B C D E H L (HL) A (IX+n) (IY+n)
SBC	A,n A,B A,C A,D A,E A,H A,L A,(HL) A,A
	A,(IX+n) A,(IY+n)
	HL,BC HL,DE HL,HL HL,SP
AND	n B C D E H L (HL) A (IX+n) (IY+n)
XOR	n B C D E H L (HL) A (IX+n) (IY+n)
OR	n B C D E H L (HL) A (IX+n) (IY+n)
CP	n B C D E H L (HL) A (IX+n) (IY+n)
INC	B C D E H L (HL) A (IX+n) (IY+n)
	BC DE HL SP IX IY
DEC	B C D E H L (HL) A (IX+n) (IY+n)
	BC DE HL SP IX IY
RLC	B C D E H L (HL) A (IX+n) (IY+n)
RRC	B C D E H L (HL) A (IX+n) (IY+n)
RL	B C D E H L (HL) A (IX+n) (IY+n)
RR	B C D E H L (HL) A (IX+n) (IY+n)
SLA	B C D E H L (HL) A (IX+n) (IY+n)
SRA	B C D E H L (HL) A (IX+n) (IY+n)
SLI	B C D E H L (HL) A (IX+n) (IY+n)
SRL	B C D E H L (HL) A (IX+n) (IY+n)
RLD	(HL)
RRD	(HL)

Geben Sie dieses Programm mit dem Text-Editor einmal ein.
Die Kommentare (; Kommentar) brauchen Sie nicht mit einzugeben.

Der Befehl .OBJECT veranlaßt den Assembler, das Programm im zweiten Durchgang ab der Adresse von PC abzulegen (3000H...).
Der Befehl ORG legt die Adresse fest, ab der das Programm stehen soll.

Danach folgen die Definitionen von Labels, die außerhalb des Programms liegen (Fachbezeichnung: EXTERNALS).
Dies ist zur Erhaltung der Übersichtlichkeit bei langen Programmen nützlich und erspart unter Umständen sogar erläuternde Kommentarzeilen.

Wie Sie hier schon sehen, sind die Labels immer von einem Doppelpunkt gefolgt. Dies soll dazu dienen, die Stelle im Text zu finden, an denen ein Label definiert wird, ohne alle Benutzungen des Labels durchsuchen zu müssen.
(Im Texteditor z.B. F/START:²CR³)
Die Länge eines Labels ist nicht begrenzt, Sie können also mehr als sechs Zeichen für ein Label verwenden.
Aus diesem Grunde kann ein Label auch alleine in einer Zeile stehen oder nur von einem Kommentar gefolgt sein.
Auf alle Fälle muß ein Label in der ersten Spalte beginnen.

Das Label CHR: ist ein 8-Bit Label (ein Byte), d.h. die oberen 8 Bit sind null. Wenn Sie z.B. eingeben würden: LD HL,CHR ,
so wäre das Ergebnis in hexadezimaler Form: 21 2A 00.
Andersherum können Sie aber auch z.B. eingeben:
LD A,CLS das Ergebnis wäre : 3E C9.
Es wird also nicht schon bei den Labels zwischen 8 und 16 Bit unterschieden, sondern erst bei den Befehlen selbst.
Wenn Sie jedoch die oberen 8 Bit eines Labels in ein Einzel-Register laden wollen, so können Sie z.B. LD A,CLS>8 eingeben
Das 8 im Argument bedeutet, daß das bisherige Ergebnis achtmal bitweise nach rechts geschoben (d.h. durch 256 geteilt) wird.
Anstelle der 8 könnte auch wiederum ein Label stehen.
Sie können auch einen Ausdruck mit <n nach links schieben.

Doch zurück zum Programm.
Der nächste Befehl ist ein DEFM. Er bewirkt, daß der folgende, in Anführungszeichen oder Apostrophen stehende Text, ab der augenblicklichen Programmzähler-Adresse abgelegt wird.
Die Länge eines solchen Textes darf maximal 255 Zeichen sein.
Wenn Sie einen längeren Text benötigen, so müssen Sie eine neue Zeile mit DEFM beginnen.
Daran anschließend folgt der Befehl DEFB. Er bewirkt, daß das angegebene Byte 00H an der Adresse von PC abgelegt wird.
Der Befehl dient zur Eingabe von Bytes, die mit DEFM nicht abgelegt werden können, weil Sie nicht über die Tastatur erreichbar sind (z.B. 00H, 00H oder Grafikzeichen)
Der DEFW Befehl hat eine ähnliche Wirkung, aber er legt jeweils zwei Bytes (Doppelbyte oder Wort) in der Form LSB,MSB ab PC im Speicher ab.
Das eigentliche Programm beginnt nun mit der Zeile:

```
BEGIN:  CALL    CLS
```

Das Label BEGIN ist ein INTERNAL, d.h. es liegt innerhalb des Programmes und erhält daher automatisch die Adresse von PC an dieser Stelle. Es wird quasi folgendes ausgeführt:

```
BEGIN:  EQU     $
        CALL    CLS
```

Außerdem werden folgende Pseudobefehle akzeptiert:

ORG	ww	PC auf Adresse ww setzen.
END	(ww)	Ende des Assemblertextes und Entry auf ww setzen (wenn angegeben).
DEFB	n(,n,n,n)	Ein Byte oder eine Liste von Bytes ab PC ablegen.
DEFW	ww(,ww,ww,ww)	Ein Wort oder eine Liste von Worten ab PC ablegen.
DEFS	n(,z)	n Bytes mit dem Wert z ab PC ablegen. (Wenn z fehlt wird 00H angenommen)
DEFM	"abc" o. 'abc'	Einen Text in Anführungszeichen oder Apostrophen ab PC ablegen.
EQU	ww o. n	Einem Label einen Wert zuweisen.
DEFL	ww o. n	Einem Label einen neuen Wert zuweisen.
.OBJECT		Programm ab PC im Speicher ablegen.

Bei den Argumenten bedeutet:

n ein Byte	42, 42D, 2AH, 00101010B, 52O, "*", '*' oder ein Ausdruck: STERN oder "!"+9 usw.
b eine Bitzahl	von 0 ... 7.
ww ein Wort	12345, 12345D, 3039H, 0011000000111001B "09", '09' oder ein Ausdruck: START, $+10 ENDE-4, SCREEN>4 usw.

Es folgt jetzt ein kleines Programm, anhand dessen einige Befehle des Assemblers erläutert werden sollen.

```
; ****************  Programm STERNCHEN
        .OBJECT             ; Im Speicher ablegen
        ORG     3000H       ; 3000H Basisadresse
;
; Label Definitionen
;
CLS:    EQU     01C9H       ; CLS-Routine im ROM
PRTSTR: EQU     2B75H       ; gibt Text ab (HL) bis 00H aus
INCH:   EQU     0049H       ; Tastatur abfragen
SCREEN: EQU     3C00H       ; Bildschirmspeicher Adresse
LENGTH: EQU     400H        ; Bildschirmspeicher Länge
CHR:    EQU     "*"         ; irgenein Zeichen
;
; Mitteilung
;
MESSGE: DEFM    "DRÜCKEN SIE IRGENDEINE TASTE !!"
        DEFB    0
;
; Start des Programms
BEGIN:  CALL    CLS                 ; Bildschirm löschen
        LD      HL,SCREEN           ; HL = Bildschirmspeicher
        LD      DE,SCREEN+1         ; DE = Bildschirmspeicher+1
        LD      BC,LENGTH-1         ; BC = Länge-1
        LD      (HL),CHR            ; erstes Bildschirmzeichen ="*"
        LDIR                        ; (HL) -> (DE) bis BC = 0
        LD      HL,MESSGE           ; HL = Mitteilung
        CALL    PRTSTR              ; ausgeben
        CALL    INCH                ; auf Tastendruck warten
        RET                         ; Zurück zum Hauptprogramm
        END     BEGIN               ; Ende des Programms
```

Nach Ausführung dieser ROM-Routine, soll der Rechner auf einen Tastendruck warten. Die Routine 0049H übernimmt dies und gibt im Akku den Wert der gedrückten Taste zurück.

Anschließend erfolgt ein Rücksprung zum aufrufenden Programm.

Der END-Befehl muß immer der letzte Befehl eines Programm-Textes sein. Der Ausdruck hinter dem END-Befehl ist ein Zeiger auf die Einsprungadresse des Programms.
Wenn kein Argument angegeben ist, so wird die Adresse des ersten ORG Befehls als Einsprungadresse angenommen.
Bei diesem Programm liegt aber ab 3000H ein Text, der nicht als Programm ausgeführt werden soll.
Daher das Argument BEGIN um die Startadresse festzulgegen.

Nach diesem END-Befehl können Sie einen noch beliebige Texte anhängen (z.B. Programmerläuterung oder Ausführungshinweise) die der Assembler nicht beachtet.

Wenn Sie dieses Programm nun eingegeben haben, so können Sie es zunächst auf seine Richtigkeit testen.
Geben Sie im Text-Editor ein: (BREAK) A (CR).
Der Bildschirm wird gelöscht und es erscheint
kurz hintereinander
Assembler

Pass 1

Pass 2

(RETURN) drücken...

Wenn Sie nun RETURN drücken, sind Sie wieder im Text-Editor.
Der Cursor ist in der ersten Zeile über dem ersten Zeichen.

Wenn ein Fehler im Programm vorliegt, so wird die Fehlermeldung in der zweiten Bildschirmzeile angezeigt, und der Cursor erscheint in der Fehlerzeile.
Bei den meisten Fehlern ist der Cursor zusätzlich an der fehlerhaften Stelle in der Zeile positioniert.

Wenn Sie keinen Fehler im Programm hatten, dann geben Sie probeweise einmal einen Fehler ein.
Schreiben Sie z.B. CAL anstelle von CALL. In diesem Fall würde die Fehlermeldung "Illegaler Befehl" lauten, und der Cursor wäre über dem ersten Zeichen des fehlerhaften Befehls.

Die möglichen Fehlermeldungen sind :

Illegaler Befehl	Falsche schreibweise in einem Befehl / Label beginnt nicht in der ersten Spalte o.ä.
Fehler im Ausdruck	Die gewählte Adressierungsart ist mit diesem Befehl nicht zulässig/ es wurde eine Klammer, ein Komma o.ä. vergessen oder zuviel eingegeben / ein Label endet nicht mit einem Doppelpunkt usw.

Das $-Zeichen steht im Assembler für den augenblicklichen PC.
Diese Zeilen könnten Sie auch eingeben, es wäre jedoch eine Platzverschwendung.

Anschließend folgen die Vorbelegungen **der Doppelregister** BC, DE und HL für den LDIR Befehl.
Statt der Labels könnten Sie hier **auch verwenden:**

```
        LD      HL,3C00H
        LD      DE,3C01H
        LD      BC,03FFH
```

jedoch mit den Labels SCREEN und LENGTH können Sie dieses Programm auch für einen anderen Z-80 Recher umschreiben, dessen Bildschirmspeicher nicht bei 3C00H beginnt und der eine andere Länge hat, indem Sie nur die Definitionszeilen ändern.
Dies ist besonders wichtig für Veröffentlichungen. Außerdem ist es bei längeren Programmen oft sehr zeitsparend.

Wenn Sie z.B. den Bereich hinter einem Programm als Buffer benutzen wollen und Sie müssen das Programm aus irgendwelchen Gründen verlängern, so müßten Sie dann alle Adressen, die diesen Buffer betreffen, wieder abändern.

Es ist also viel einfacher, wenn Sie ein Label mit dem Namen BUFFER benutzen, und so immer nur eine Wertzuweisung ändern müssen, oder noch einfacher: Sie schreiben das Label BUFFER: vor den END-Befehl in der letzten Zeile.

Noch ein Hinweis zu den Labeldefintionen,
es sind zwar Verschachtelte Definitionen **erlaubt wie z.B.**

```
BASE:   EQU     8000H
BUFF:   EQU     BASE+10H
```

aber Sie dürfen Labels erst dann benutzen, wenn Sie schon definiert sind.
Sonst sind möglicherweise seltsame Ergebnisse zu erwarten.
Ein Beispiel für einen solchen Fehler:

```
START:  DEFS    NEXT-START
NEXT:   ......
```

Der Assembler kann hier die Differenz zwischen NEXT und START nicht richtig berechnen, da erst das Ergebnis dieser Differenz die Adresse von NEXT festlegen würde.
Am besten verwenden Sie in dieser Situation entweder einen eine Zahl als Argument, oder aber schon festliegende Labels.

Der nächste Befehl im Programm ist LDIR. Dieser Befehl kopiert das Byte von der Adresse, auf die HL zeigt, auf die Adresse, auf die DE zeigt. Anschließenden werden HL und DE um eins erhöht und BC um eins erniedrigt. Wenn BC dabei nicht 0 wird, dann wird der Befehl wiederholt.

Hier wird also das Sternchen von 3C00H nach 3C01H kopiert.
Anschließend zeigt HL auf Adresse 3C01H und DE auf 3C02H und das Sternchen wird von 3C01H nach 3C02H kopiert usw.
Diese Methode wird oft verwendet, um einen Speicherbereich zu löschen oder mit einem bestimmten Wert zu belegen.
Anschließend wird HL mit der Adresse eines Textes geladen, um eine ROM-Routine aufzurufen, die ab Adresse HL Zeichen auf den Bildschirm ausgibt, bis ein Byte mit dem Wert 00H erreicht wird.

Ein "Relativer Sprung ist zu groß", wenn die Distanz zwischen Zieladresse und momentaner Adresse größer als 127 ist.
Da der Assembler jedoch Sprünge relativ zur Adresse des gerade assemblierten Befehls berechnet, ist der maximale Offset bei den zwei Byte langen relativen Sprungbefehlen $-125 bis $+129.
Der Befehl JR $ springt also wieder zwei Bytes zurück.

"Kein END-Befehl" ist ein eindeutiger Fehler.
In der letzten Programmzeile muß immer der END Befehl **stehen.**

Die Fehlermeldung "Zuwenig Speicherplatz" kann behebbar sein, wenn die Text-Buffer Grenzen noch nicht maximal ausgenutzt sind.
Sie können Sie dann mit den Text-Editor Befehlen ML und MH auf maximale Speicherausnutzung erweitern (s. auch "Text-Editor").
Wenn das schon geschehen ist, oder wenn die Fehlermeldung schon im ersten Durchgang auftritt, so bestehen kaum Chancen das Programm noch zu assemblieren.
Einzige Möglichkeit bleibt dann, Labels zu verkürzen oder wegzulassen, Kommentarzeilen zu entfernen und evtl. mehr Unterprogramme zu benutzen.
Ansonsten können Sie auch versuchen, das Programm in mehrere Teile aufzuteilen und getrennt zu assemblieren.
Dabei gehen natürlich die Symbole aus dem ersten Teil beim zweiten Teil verloren, und müssen umständlich mit
LABEL: EQU xxxx neu definiert werden.

Da Sie maximal 36 kByte Text-Speicher zur Verfügung haben **und** das Verhältnis von Länge des Quellprogramms zu Länge des Objectprogramms ca. 10:1 ist, können Sie knapp 3.5 kByte **lange** Programme noch assemblieren.

Versuchen Sie daher bei längeren Programmen Daten, Bilder, Texte usw. außerhalb des Quell-Programms zu speichern.
Sie können dann ein Programm z.B. mit dem .OBJECT Befehl direkt im Speicher ablegen und anschließend mit dem S Befehl des Monitors zusammen mit den Daten auf Kassette abspeichern.

Die Fehlermeldung "Geschützter Bereich xxxxH !!" tritt nur **auf,** wenn Sie den .OBJECT Befehl verwendet haben und das Programm einen geschützten Speicherbereich überschreiben würde.
Diese Bereiche sind der Bereich des Text-Buffers, der Bereich von 0000H bis FFFFH und die Symboltabelle, die hinter dem Textende beginnt.
Ändern Sie die Adresse des ORG Befehls, oder ändern Sie die Text-Buffer Adressen, um einen größeren freien Speicherbereich zu erhalten.
Beachten Sie, daß vor ML (Memory low) noch 100H Bytes für den Stackpointer-Bereich frei bleiben müssen.

Bevor Sie ein assembliertes Programm ausführen wollen, sollten Sie den Quelltext in jedem Fall auf Kassette abspeichern.
Es ist ja nie ganz sicher, daß das Programm so arbeitet wie es soll und nicht zu einem Kaltstart führt oder den Speicherinhalt überschreibt.

Wenn ein Programm im Bereich von 0000 bis FFFF laufen soll, so können Sie es natürlich nur testen, indem Sie es auf Kassette abspeichern und anschließend mit dem SYSTEM-Befehl vom Basic her wieder einlesen.

Undefiniertes Symbol	Ein nicht definiertes Label wurde in einem Ausdruck benutzt/ eine Zahl beginnt nicht mit einer Ziffer (z.B. FFH statt OFFH).
Doppelt definiertes Symbol	Ein Label wurde zum zweiten mal benutzt.
Relativer Sprung zu groß	Der Offset für einen JR oder DJNZ Befehl ist kleiner als $-125 oder größer als $+129.
Kein END-Befehl	Der END-Befehl fehlt oder wurde versehentlich gelöscht.
Zuwenig Speicherplatz	Die Symboltabelle hat keinen Platz innerhalb der Text-Buffer Grenzen / Programm ist zu lang für die Text-Buffer Grenzen.
Geschützter Bereich xxxxH	Der Bereich xxxxH darf nicht überschrieben werden.

Wie behebt man nun diese Fehler ?
Ein "Illegaler Befehl" ist meist wohl ein Rechtschreibefehler.
Wenn nicht, so vergewissern Sie sich, ob der Befehl in der Tabelle der zulässigen Befehle aufgelistet ist.

Ein "Fehler im Ausdruck" kann verschiedene Ursachen haben.
Am besten einige Beispiele für häufig gemachte Fehler:

```
        LD      A,3A            3AH    wäre richtig
START   CALL    5000H           START: wäre richtig
        RES     (HL)            b,     fehlt
        BIT     4,HL            Doppelregister nicht erlaubt
        LD      (HL),(IX+2)     Adressierungsart nicht erlaubt
        INC     AF              Adressierungsart nicht erlaubt
```

Aber ACHTUNG !!!

```
        LD      A,1B            wird akzeptiert als 1 binär.
```

In diesem Fall erhalten Sie keine Fehlermeldung !
Geben Sie also bei solchen Argumenten besonders acht.

Ein "Undefiniertes Symbol" kann auftreten, wenn Sie ein Symbol das Sie benutzen wollen tatsächlich noch nicht definiert haben. Oder Sie haben es in der Definition und im Ausdruck unterschiedlich geschrieben (Groß-/Kleinschrift beachten). Eine weitere Möglichkeit ist, daß eine hexadezimale Zahl größer als 9FH nicht mit einer führenden Null beginnt.
Aber ACHTUNG !!!
Wenn Sie z.B. ein Label mit dem Namen 'BCH' definiert haben, und Sie wollen den Befehl LD A,0BCH ausführen, so führt eine fehlende Null nicht zu einer Fehlermeldung.
Vermeiden Sie daher Symbol-Namen die mit Hexadezimalzahlen, Z-80-Registernamen oder Bedingungs-Flags verwechselbar sind.

Ein "Doppelt definiertes Symbol" liegt vor, wenn zwei Labels die gleiche Länge haben und gleich geschrieben sind.
Aber die Labels START, Start und StArT sind unterschiedlich. Sie können also ein Unterprogramm TEXT: nennen und den auszugebenden Text mit dem Label Text: versehen, ohne daß die Fehlermeldung "Doppelt definiertes Symbol" auftaucht.

[redacted]

Kleinere Programme (bis 1 kByte) können Sie in den Bereich 3000H bis 33FFH assemblieren. Dort ist ein Speicherbereich, der vom Basic nicht benutzt wird und sehr sicher ist.

Wenn Sie aus dem Text-Editor oder dem Monitor ins Basic zurückgehen, so wird der letzte freie Speicher des Basic-Bereiches auf die Adresse von ML -1 gesetzt. Wenn Sie ML auf Adresse 42E8H gesetzt hatten, so kann das Basic nicht mehr richtig arbeiten, da 'weniger als 0 Bytes' Platz vorhanden sind. Also legen Sie entweder ML wieder höher, oder führen Sie einen Kaltstart aus. Dies geht z.B. vom Monitor mit dem Befehl G 0(CR).
Jedoch ist dann der Text-Buffer auch verloren und muß evtl. neu von Kassette geladen werden.

Wenn Sie also im Basic und Text-Editor abwechselnd arbeiten wollen, dann dürfen Sie den Parameter ML nicht kleiner wählen als zu Beginn ausgerechnet wurde. Das Basic-Programm bleibt dann erhalten, solange Sie nicht in diesen Bereich assemblieren oder ihn vom Monitor her überschreiben.

Da das Basic aus einem EPROM ins RAM 'gebootet' wird, ist es auch veränderbar. Dazu müssen Sie nur den 'Schreibschutz' für den unteren 12 kByte Bereich ausschalten (BIT 5 auf Port FEH zurücksetzen). Jedoch müssen Sie dann besonders vorsichtig bei direkten Änderungen sein, da auch der Text-Editor, der Monitor und der Assembler Unterprogramme aus dem Basic-Bereich benutzen.

Wenn Sie also das Basic manipulieren wollen, so ist es noch wichtiger alle Programme auf Kassette zu sichern.

In diesem Zusammenhang ein Hinweis:

Alle Literatur über das Microsoft-Basic des Genie I/II trifft auch im wesentlichen für das Genie II S zu. Es sind lediglich kleine Änderungen vorgenommen worden. Diese sind jedoch im Vergleich mit einem ROM-Listing sehr schnell ersichtlich.

Unter diese Änderungen fallen:
1. Der Text "READY?" - "Mem Size?"
   (CR)(CR)... "READY" - (CR)(CR)"Genie II S"
2. Der NAME Befehl zum Aufruf der Systemerweiterung.
3. Eine automatische Umschaltug auf LOW SPEED bei Kassettenoperationen und zurück auf HIGH SPEED bei MOTOR OFF.

Außerdem wurde die Routine für Zufallszahlen verkürzt, aber gleichzeitig verbessert und beschleunigt. Es entfallen die Multiplikationen, statt dessen wird das R-Register des Z-80 verwendet um Zufallszahlen zu erzeugen.
Der RANDOM-Befehl wäre somit eigentlich überflüssig, wurde aber beibehalten, um die Lauffähigkeit 'alter' Basic-Programme zu garantieren.

Wenn Sie sich noch in die Programmierung des Z-80 einarbeiten wollen, so kann diese Anleitung dabei nicht als Lernhilfe betrachtet werden, da nur die Besonderheiten speziell dieses Assemblers ausführlich erläutert sind.
Wir empfehlen Ihnen daher die einschlägige Literatur über den Z-80 Prozessor und seine Programmierung.
(z.B. Rodnay Zaks, Programmierung des Z-80, SYBEX-Verlag o.ä.).