

INHALT =====	Seite =====
I. EINFUEHRUNG .....	2
II. LADEN .....	3
III. SYNTAX DER DOKUMENTATION .....	5
IV. LABEL DEFINITION .....	6
V. LABEL AUFRUFE .....	9
VI. ABHAENGINER LABEL AUFRUF .....	10
VII. VERZWEIGUNG DURCH ZEICHENKETTEN .....	11
VIII. PARAMTER UEBERGABE IN BASIC .....	11
IX. MASCHINENSPRACHE PARAMETER UEBERGABE .	12
X. SUCH BEFEHLE .....	15
XI. TABELLEN BEFEHLE .....	16
XII. SYSTEM INFORMATIONS BEFEHLE .....	19
XIII. FEHLERMELDUNGEN .....	20
XIV. BEISPIEL PROGRAMM "DEMO" .....	ANHANG 1
XV. BEISPIEL PROGRAMM "ARRAY" .....	ANHANG 2

MACRO-BASIC ist ohne Einschränkungen verwendbar in Verbindung mit LEVEL II BASIC, Microsoft und PCS LEVEL 3 BASIC und den DISC-BASIC Versionen von TRSDOS, NEWDOS 21, NEWDOS 40 und NEWDOS 50. Bereits unter LEVEL II BASIC mit 16k Speicher und Cassette arbeitet MACRO-BASIC problemlos.

## II.

LADEN VON MACRO-BASIC

Auf der 1. Seite der Cassette sind 2 Kopien von MACRO-BASIC, auf der Ruckseite ein dokumentiertes Beispiel Programm abgespeichert. MACRO-BASIC wird geladen und initialisiert wie ein Basic-Programm. Sie initialisieren Basic in der bewunschten Version (LEVEL II, DISK BASIC, LEVEL III etc.) wie gewohnt mit einer MEMORY-SIZE Eingabe und laden MACRO-BASIC mit CLOAD bzw. LOAD mit anschliessendem RUN oder lediglich RUN in DISK-BASIC. MACRO-BASIC verschiebt sich dann unter die in MEMORY SIZE angegebene obere Speichergrnze, schuetzt sich selbststendig, zeigt seine Eingangsmeldung und geht in den Befehlsmodus. Sie koennen dann die Zusatzbefehle verwenden.

Im Befehlsmodus wird in MACRO-BASIC nicht mehr das Wort READY, sondern nur noch der Prompt > dargestellt, so dass Sie eine Bildschirmzeile mehr im Befehlsmodus verfuegbar haben.

Speicherplaetze oberhalb von MEMORY SIZE werden von MACRO-BASIC nicht benutzt. Die nur fuer die Initialisierung benoetigten Speicherplaetze werden nach der Initialisierung wieder freigegeben, so dass waehrend der Ausfuehrungsphase nur ca. 1700 Bytes belegt bleiben.

Fuer Benutzer, die eingese Maschinensprache Programme an den NAME oder LINE Vektor geknuerft haben, besteht die Moeslichkeit, diese Routinen wie gewohnt anzuseringen, wenn VOR der Erstinitialisierung von MACRO-BASIC diese Vektoren spaendert worden sind und die Programme im Speicher hoeher als MACRO-BASIC liegen.

**DER VERSUCH, ZEILEN VON MACRO-BASIC ZU EDITIEREN  
ODER IN IRGENDWEINER FORM ZU VERAEENDERN  
FUEHRT ZU INITIALISIERUNGS FEHLERN !!!**

Dabei wird der Speicherinhalt zerstoert und in fast allen Faellen der Fehler gemeldet. Der Computer MUSS dann ueber den RESET Knopf oder Ein- und Ausschalten neu gestartet werden.

II/A. CASSETTE

Wenn Sie mit Expansion-Interface unter einem Basic arbeiten, dass die Echtzeit Uhr unterstuetzt, muessen Sie zunaechst

COMD "T"

eingeben, um die Uhr abzustellen.

## I.

EINFUEHRUNG

MACRO-BASIC ist ein Satz von Zusatzbefehlen zum LEVEL II Basic des TRS-80, der

**PROGRAMMVERZWEIGUNGEN MIT SYMBOLISCHEN NAMEN  
UND PARAMETER UEBERGABEN**

ermoeslicht. Damit bieten sich dem Programmierer in Basic Vorteile, die sonst erst in hoeher entwickelten Compiler Sprachen wie PASCAL oder ELAN zur Verfuegung stehen.

\* Anstelle von abstrakten Zeilennummern werden bei Benennen und Unterprogramm Aufrufen benannte FUNKTIONSBLOCKE mit ihren PARAMETERN aufgerufen.

\* Bei Verzweigungen, die abhaengig vom Wert eines AUSDRUCKS oder vom INHALT EINER STRING-VARIABLEN vorgenommen werden, sind SPFUEHRE UND UNTERPROGRAMM-AUFRUFE und MASCHINENPROGRAMM-AUFRUFE mit oder ohne Parameteruebergabe beliebig MISCHBAR.

\* Die UEBERSICHT ueber den Programmablauf wird insbesondere bei laengeren Programmen durch Programm SELBSTDOKUMENTATION in den Verzweigungsbefehlen erheblich vereinfacht, was durch ERWEITERTE KOMMENTIERUNGSTECHNIK zusätzlich unterstuetzt wird.

\* Neue REFERENZBEFEHLE fuer Bildschirm und Drucker-Ausgabe ermoeslichen sofortiges AUFFINDEN JEDES DEFINIERTEN LABELS die DARSTELLUNG ALLER DEFINIERTEN LABEL und das Auffinden BELIEBIGER ZEICHENKETTEN mit LEERZEICHEN IGNORIERUNG.

\* Auch nach UMNUMMERIERUNGEN sind Verzweigungsziele noch ohne laenges Suchen erkennbar.

\* Label sind in der LAENGE nur durch die in Basic erlaubte Zeilenlaenge besaenzt.

\* Mischen von BUCHSTABEN, ZAHLEN und mehreren SONDERZEICHEN fuer ein Label sowie die Verwendung von Basic Befehlsworten innerhalb eines Labels ist zulassig.

\* Die Befehls-SYNTAX ist ANSCHAULICH und erlaubt die Verwendung von Leerzeichen zwischen Befehl, Trennzeichen und Label.

## III.

## SYNTAX DER DOKUMENTATION

MACRO-BASIC verwendet die Befehle LINE, NAME, DEF und USRTABX in Verbindung mit verbindlichen Sonderzeichen als Trennzeichen. Die Verwendung dieser Befehle ohne die Trennzeichen entspricht der üblichen Verwendung dieser Befehle in der jeweils verwendeten Basic Version.

In der Syntax-Beschreibung seien folgende Vereinbarungen:

<b>GROSSBUCHSTABEN</b>	Befehlswort, muss wie gezeigt geschrieben werden
@ \$ !	verbindliche Trennzeichen
# * : , ( )	verbindliches Leerzeichen, sonstige Leerzeichen koennen beliebig fehlen
LC	numerischer Ausdruck
EXPR	Zeichenketten Ausdruck
EXPR#	Text, der vom Benutzer formuliert wird
kleinbuchstaben	Text innerhalb der Klammer nach Wahl
< >	sonstige Leerzeichen beliebig
####	Zeilennummer
<...>	Wiederholung in der vorgenannten Form nach Wahl
<#>	weiterer Basic Programtext nach Wahl
<komment>	Kommentartext nach Wahl
<BREAK>	BREAK-Taste
<SHIFT @>	Shift Taste und @-Taste gleichzeitig.

Sie laden unter Basic mit CLOAD wie gewohnt. Bei richtiger Lautstaerke-Einstellung (zwischen 4 und 6) erscheinen jetzt die geordneten Sterne, von denen der rechte langsam blinkt. Wenn keine Sterne erscheinen oder keiner blinkt, versuchen Sie es mit einer anderen Lautstaerke-Einstellung. Nach kurzer Zeit erfolgt die READY-Meldung. Sie starten das Programm jetzt wie ein Basic Programm mit RUN. MACRO-BASIC bringt jetzt seine Einlesemeldung und verzweigt in den Befehlsmodus.

Es ist empfehlenswert, eine Sicherungskopie des erfolgreich geladenen MACRO-BASIC zu machen. Sie koennen dazu MACRO-BASIC nach Eingabe von CMD "T!" wie ein normales Basic Programm mit SAVE abspeichern und mit CLOAD? prueflesen. Das Programm darf vor dem Abspeichern nicht editiert oder gestartet worden sein!!!

Wenn eine Version bei keiner Lautstaerke-Einstellung ladbar sein sollte, versuchen Sie die 2. Version auf der gleichen Seite der Datencassette zu laden. Sollte auch diese nicht ladbar sein, koennen Sie gegen Ruecksendung der Originalcassette ein Ersatzexemplar erhalten.

## II.E. DISKETTE

Um MACRO-BASIC von Cassette auf Diskette zu uebertragen, laden Sie es zunaechst wie unter Cassette beschrieben. Sie koennen es nun als Basic File mit SAVE "MACRO" auf Diskette speichern. BEVOR sie es editiert oder initialisiert haben. Die Abspeicherung darf nicht im ASCII-Format erfolgen. Beim Laden von Diskette koennen Sie unter NEWDOS oder NEWDOS 80 den Einzelzeilen-Einstieg benutzen, z.B.:

```
BASIC 4, 43000, RUN "MACRO"
```

oder

```
BASIC RUN "MACRO"
```

Die Zahl der Files oder die MEMORY-SIZE-Grenze kann beim Laden von der beim Abspeichern abweichen. Die Echtzeit-Uhr muss nach der Initialisierung wieder mit

```
CMD "R"
```

aktiviert werden, wenn Interrupts erwuenscht sind. Wenn Basic ueber CMD "S" oder Reset verlassen wurde, ist der Programtext noch mit BASIC 4 verfuegbar und kann abgezeichnet werden. MACRO-BASIC muss jedoch nach Speichern des Textes neu geladen werden.

Die Definition muss als ERSTER Befehl in der Zeile stehen.

Fuer die Bildung von Labeln gelten folgende Einschränkungen:

1. Die Basic Schlüsselworte ELSE DATA REM sowie ' sind innerhalb von Labeln nicht zulaessig.
2. Die Zeichen @, \$ und !, die als Trennzeichen fuer Verzweigungsarten dienen, sind dann nicht als Bestandteile von Labeln zulaessig, wenn Label in einer LINE ON Verzweigung verwendet werden.
3. LZ ( und ; werden als Label-Schlusszeichen aufgefasst, koennen daher nicht innerhalb eines Labels stehen.
4. Anführungszeichen " sind innerhalb von Labels nur zulaessig, wenn sie paarweise innerhalb eines Labels auftreten.

Um Risiken zu vermeiden, sollte man daher innerhalb eines Labels folgende Zeichen oder Zeichenketten grundsatzlich nicht verwenden:

@ \$ ! ' ( ) " LZ ELSE DATA REM

Die Verwendung anderer Basic-Schlüsselworte (z.B. INPUT oder PRINT) und Sonderzeichen wie . + , oder > sowie von Zahlen ist innerhalb von Labeln in beliebiger Reihenfolge zulaessig. Dabei erhoert die Verwendung von Schlüsselworten innerhalb von Labeln sogar die Verarbeitungs-Geschwindigkeit und spart Speicherplatz sowie Tabellen-Speicherplatz im Tabellen-Modus.

Fuer den auf das Label oder die Variablenliste folgenden wahlweisen Kommentartext gelten folgende Einschränkungen:

1. Die Verwendung von ; und ELSE innerhalb des Kommentars ist nicht zulaessig, da der darauf folgende Text als Basic Befehl interpretiert wuerde.
2. Falls auf den Kommentartext innerhalb der Definitionszeile weitere Befehle folgen, gelten die aehnliche Einschränkungen wie bei Labeln: DATA REM und ' sind unzulässig, " duerfen nur paarweise verwendet werden.
3. Mit ( darf der Kommentar nur beginnen, wenn er auf eine Variablen- bzw. Typ-Liste folgt. Wenn der Kommentar unmittelbar auf das Label folgt, ist ein verbindliches Leerzeichen erforderlich, um das Label Ende zu kennzeichnen. Zeilenvorschuebe und Kommata sind zulaessig.

Wird dasselbe Label innerhalb eines Programms mehrfach definiert, so ist nur die Definition aeltlich, die dem Programmstart am naechsten ist. Eine Pruefung auf mehrfache Definition erfolgt nicht. In Zweifelsfaellen kann der Textsuchbefehl NAME #) string zur Pruefung von Doppeldefinitionen verwendet werden.

## 10.

### ----- LABEL DEFINITION

Die Definition eines Labels bewirkt, dass die Programmzeile, in der das Label definiert wird, mit diesem Label als ihrem Namen aufgerufen werden kann. Die Definition eines Labels fuer Maschinensprache Programmteile bewirkt, dass ein Maschinensprache Programm, dessen Startadresse in der Label-Definitionszeile bestimmt wird, mit seinem Namen aufgerufen werden kann.

Eine Label-Definition besteht aus folgenden Teilen:

1. der Zeilennummer mit dem unmittelbar oder nur durch Leerzeichen von ihr getrennten Befehlswort NAME,
2. einem Trennzeichen, mit dem zwischen einem Basic-Label und einem Maschinensprache Label unterschieden wird,
3. dem symbolischen Namen (Label), mit dem die Zeile bzw. das Maschinensprache Programm benannt werden soll,
4. wahlweise einer Liste von Zielvariablen bzw. Zielarten, wenn Parameter in Empfang genommen werden sollen,
5. wahlweise einem Kommentartext,
6. wahlweise weiterem Programmtext.

Damit erfolgt die Definition eines Labels fuer eine Basic Programmzeile in folgender Form:

```
%%NAME # label (( var-liste )) (LZ komm ) : prog
```

Diese Definition bewirkt, dass bei einem Aufruf des Labels zu dem Programmtext verzweigt wird, der auf das naechste Befehls-Trennzeichen folgt, also : oder Zeilenende. Das Label kann wahlweise mit

LZ ( : oder Zeilenende

abgeschlossen werden. Bei Abschluss eines Labels mit ( wird eine Variablen-Liste erwartet, die mit ) abgeschlossen werden muss (z.B. unter Parameter Uebergabe). Wenn ein Kommentartext folgen soll, kann er unmittelbar nach dem Abschluss Zeichen der Variablen-Liste ) oder - nach einem verbindlichen Leerzeichen - auf das Label selbst folgen. Wenn bei der Abarbeitung des Programms ohne Label Aufruf auf eine Label Definition gestossen wird, werden Label, Variablen-Liste und Kommentartext ueberlesen und an der darauffolgende Programmtext abgearbeitet.

## U.

LABEL-AUFRUFE

Ein mit Label gekennzeichnete Programmabschnitt kann als Sprungziel (entsprechend GOTO %%%%) oder als Unterprogramm (entsprechend GOSUB %%%%), ein mit Label definiertes Maschinenprogramm entsprechend der USR Funktion aufgerufen werden. Der Aufruf besteht aus folgenden Teilen:

1. wahlweise einer Zeilennummer,
2. wahlweise Programmtext (z.B. IF .. THEN Anweisungen),
3. dem aufrufenden Befehl LINE,
4. dem Verzweigungstyp Trennzeichen @ \$ oder !,
5. dem Label, das im Programm fuer die Stelle definiert wurde, zu der verzweigt werden soll,
6. einer Parameter-Liste, wenn das Label mit Parametern definiert wurde,
7. einem wahlweisen Kommentarfeld,
8. wahlweise weiterem Programmtext, der nach Unterprogramm- oder Maschinenprogramm-Aufrufen abgearbeitet wird oder wenn die Bedingung fuer einen Sprung Befehl nicht erfuehlt ist.

Die Syntax fuer einen SPRUNG-Befehl ist:

```
<%%> <#> LINE @ label << param-liste >><LZ komm > <:#>
```

Die Syntax fuer einen Unterprogramm Aufruf unterscheidet sich davon nur in dem Trennzeichen \$ statt @ :

```
<%%> <#> LINE $ label << param-liste >><LZ komm> <:#>
```

Ein Maschinenprogramm Aufruf benutzt als Trennzeichen ! :

```
<%%> <#> LINE ! label << var-liste >> <komm> <:#>
```

Der nach dem Kommentar wahlweise moegliche weitere Programmtext muss unbedingt mit einem : abgetrennt werden. Als Label interpretiert werden nur die Zeichen bis zum ersten Label Schlusszeichen, also LZ ( : oder Zeilenende. Der Kommentar bei Label Aufruf und Definition kann unterschiedlich sein.

IN 4. LABEL FUEER MASCHINENSPRACHE ROUTINEN

Die Definition von Labeln fuer MASCHINENSPRACHE Routinen erfolgt ueber die NAME-Anweisung mit dem Trennzeichen ! in folgendem Format:

```
%%>> NAME ! label < adrexx > << var-liste >>< komm >
```

Der Ausdruck adrexx wird als Eiserungs Adresse des Maschinenprogramms aufgefasst. Zulaessig sind hierbei Ausdruecke, die einen Integer Wert erzeben oder Konstanten, die den Bereich von 1 bis 65529 nicht ueberschreiten (wie die Eingabe bei MEMORY SIZE ). Fuer hohe Adressen sind negative Zahlen also nicht erforderlich (anders als bei DEFUSR in BASIC-BASIC). Wenn der Ausdruck, der die Adresse definiert, Variable enthaelt, wird der beim Aufruf resultierende Wert der Variablen bei der Adress-Berechnung benutzt. Das hat den Vorteil, dass mit der gleichen Definitions-Zeile verschiedene Eiserungs-punkte in ein Programm erfolgen koennen, birgt bei unkontrolliertem Einsatz von Variablen das Risiko eines undefinierten Einstiegs. Sicherheitshalber sollten daher weniger erfahrene Programmierer nur Konstanten benutzen.

Die Definitionszeile darf keine weiteren Basic Befehle enthalten. Sie kann an beliebiger Stelle im Programm stehen und wird wie eine Kommentar Zeile ueberlesen, bis ein Aufruf des Maschinenprogramms erfolgt. Erst in diesem Moment erfolgt die Adressberechnung und der Vergleich der Variablenliste der aufrufenden Zeile mit ihrer Tyelliste.

Die Definition muss als erster Befehl in einer Zeile stehen und ist nur gueltig, solange die Definitions-Zeile im Programmtext steht. Eine mehrfach Definition ist nicht moeglich, da nur die im Programmtext am weitesten zu Beginn liegende Definition ausgewertet wird.

Es ist zulaessig, das gleiche Label fuer Basic Zeilen und Maschinensprache-Aufrufe zu verwenden, ohne dass Fehler auftreten. Bis zu 10 Parameter aller Variablen-Typen koennen an das Maschinenprogramm uebergeben werden (z.B.). Die Regeln ueber zulaessige Zeichen innerhalb des Labels und des Kommentars gelten wie bei Basic Label Definitionen.

## VII.

VERZWEIGUNG DURCH ZEICHENKETTEN

Verzweigungen koennen abhaengig vom Inhalt einer String-Variablen oder eines String-Ausdrucks erfolgen, der als Ergebnis einen String hat. Dieser String muss aus einem Verzweigungstyp Trennzeichen @ \$ oder ! als erstem Zeichen und einem Label bestehen und kann eine Parameter-Liste in Klammern sowie abschliessenden Kommentartext beinhalten. Der fuer diese Verzweigungstechnik als Argument benutzte String muss den Syntax-Anforderungen des LINE @, LINE \$ oder LINE ! Befehls ohne das beginnende Schluesselwort LINE und ohne vorherigen und auf den Kommentar folgenden Programtext entsprechen.

Dieser Aufruf hat folgendes Format:

```
<%%> <par:> LINE USING ( expr$ ) <komm> <:par>
```

Diese Verzweigungstechnik ermoeglicht es, umfangreiche verteilte Strukturen mit wechselnden Variablen in den Parameter-Übersetzen komfortabel zu programmieren, oder String Arrays zur Ablaufsteuerung in modular aufgebauten Programmen heranzuziehen.

## VIII.

PARAMETER UEBERTRAGUNG IN BASIC

MACRO-BASIC ermoeglicht die Uebersgabe einer Gruppe von Parametern an Unterprogramme und Sprungziele im aufrufenden LINE \$ bzw. LINE @ Befehl, wenn in der Definitionszeile des auferufenen Labels eine Ziel-Variablen Liste auf das Label folgt, die im Aufbau der Parameter Liste in der nach dem aufrufenden Label entspricht.

Die Zuweisung der Werte aus der Parameter Liste an die Variablen aus der Variablen Liste erfolgt nach den selben Regeln, die fuer LET bzw. = Zuweisungen in Basic erlaubt sind:

1. Typumwandlungen in den Typ der Zielvariablen erfolgen bei der Uebersgabe von numerischen Ausdruecken.
2. Als Uebersgabe Parameter koennen Konstanten oder Ausdruecke verwendet werden.
3. Als Zielvariablen sind alle in Basic zulaessigen Variablen Namen moeslich.

Die Rueckverzweigung nach Abschluss des Unterprogramms erfolgt wie gewohnt mit RETURN. Die Rueckverzweigung aus einem Maschinenprogramm darf nur ueber einer FET - Anweisung in Maschinensprache. Der Rueckprung verzweigt zu dem naechsten Befehl nach dem auf den Label-Aufruf folgenden ; oder Zeilenende, ein eventueller Kommentar wird uebersprungen.

Die Verwendung der Befehle GOTO %%%% und GOSUB %%%% sowie A=USR(X) ist wie gewohnt neben LINE @ label LINE \$ label und LINE ! label zulassig.

## UI.

ABHAENGINGER LABEL AUFRUF

Entsprechend den Basic Anweisungen  
ON expr GOTO %%%%, ... , %%%% bzw.  
ON expr GOSUB %%%%, ... , %%%%

gibt es in MACRO-BASIC eine vom Wert eines Ausdrucks (expr) abhaengige Verzweigung, in der Sprung- und Unterprogramm-Anweisungen und Maschinenprogramm-Aufrufe GENISCHT werden koennen und die Parameter Uebersaben und Kommentartext-Einschuebe in der Verzweigungs-Liste erlaubt. Bei dieser Verzweigungstechnik wird zunaechst der numerischer Ausdruck expr als Integer Wert n berechnet und der n-te Verzweigungsbefehl in einer auf den Ausdruck folgenden Verzweigungs-Liste ausgefuehrt:

```
<%%> <par: > LINE ON expr ,  
$ label1 << param-liste1 >> <komm1>  
@ label2 << param-liste2 >> <komm2>  
! label3 << var-liste3 >> <komm3>  
<...>  
<:par>
```

Die Auswertung von expr entspricht der in dem genannten ON expr GOTO Befehl (vgl. LEVEL II Handbuch S. 46 f.). Verbindliches Trennzeichen ist das Komma nach expr.

Die Label in der Verzweigungs-Liste werden bis zum LE ; oder Zeilenende Trennzeichen ausgewertet. Jedes @ ! oder ! Zeichen wird als Labelkennungszeichen bewertet, darf also nie auch ; nicht in den Kommentaren auftreten! Grundsatzlich sind auch bei den Kommentaren die in Latein unzulassigen Zeichenfolgen oder Sonderzeichen nicht gestattet (s.o.). Klammern in Kommentaren sind nicht zulassig. Zusätzliche Kommas oder Zeilenvorschuebe sind NACH dem das Label abschliessenden LE erlaubt, aber nicht verbindlich. Ein ; oder Zeilenende wird als Abschluss der Liste aufgefasst.

Das Zeichen @ in dieser Liste ist ein Dummy-Zeichen, das darauf hinweist, das eine Adresse uebergeben wird. Die fortlaufende Zaehlung hinter @ dient der besseren Uebersicht bei laengeren Typ-Listen. Diese Zaehlung muss mit 0, nicht 1, beginnen, kann bis maximal 9 gehen und muss unmittelbar auf @ folgen.

Bei Eintritt in das Maschinenprogramm ist das

Z-FLAG gesetzt, wenn KEINE PARAMETER uebertragen wurden, das A-REGISTER enthaelt die ZAHL der uebergebenen Parameter.

Wenn Parameter uebergeben wurden, zeigt

das IV-REGISTER auf eine Variablentyp und Adress-TABELLE, das B-REGISTER ist 0, das C-REGISTER enthaelt die TYP-Nummer der Variable, das HL-REGISTER deren ADRESSE im VARPTR Format (vgl. LEVEL II Handbuch S. 8/8). Die uebrigen Register und der alternative Registersatz sind undefiniert. Alle Register duerfen veraendert werden. Fuer das Maschinenprogramm sind 14 Stack-Ebenen = 28 Byte gesichert.

Die Variablen Typ-Nummer wird durch die Zahl der fuer den Typ benoetigten Daten Bytes bestimmt:

Typ	Variable:
2	Integer
3	String
4	einfache Genauigkeit
8	doppelte Genauigkeit

Sicherheitshalber sollte jedes empfangende Maschinenprogramm am Anfang eine Pruefroutine enthalten, die prueft, ob Zahl und Typen der Parameter dem entsprechen, was das Programm verarbeitet. Bei Fehlern kann das Programm mit

JP 1997H ;SN-ERR Routine

verlassen werden.

Wenn MACRO-BASIC unter DISK-BASIC verwendet wird, kann das auferufene Maschinenprogramm in den meisten DOS Versionen mit

CALL 4400H ;Einsties ins Debug Programm

besonnen werden, bis die Verzahnung von Programm und Variablen zuverlaessig programmiert ist. Die Register entsprechen dann dem oben beschriebenen Einsties Format.

Die Syntax fuer eine Zielzeile mit zuehoerigem Aufruf ist:

```

NAME # label ( var1 , var2 , <...> , varn ) <komm> <:fer>
<:ferr> <:fer:> LINE # label ( exprs , expr2 , <...> , exprn )
<komm> <:fer>

```

Fuer einem Sprung Befehle muss anstelle des # ein @ stehen. Die Parameter Uebergabe in einer LINE ON Verzweigung erfolgt ebenso.

## IX.

## PARAMETER UEBERGABE AN MASCHINENPROGRAMME

Anders als bei der Parameter Uebergabe in Basic koennen an Maschinensprache Programme lediglich vor dem Aufruf des Maschinenprogramms definierte Variable aller Typen, jedoch keine Konstanten oder Ausdruecke uebergeben werden. Durch diese Technik koennen dem Maschinenprogramm die ADRESSEN der Variablen im VARPTR Format uebermittelt werden. Das Maschinenprogramm seinerseits kann ueber diese Variablen Werte an das Basic Programm uebergeben.

Um das Risiko, das die Uebermittlung vom Typ her nicht erwarteter Variabler an das Maschinenprogramm birgt, zu verringern, muss in der Definitions Zeile fuer das Maschinenprogramm ein Typ-Liste fuer alle Variablen, die uebergeben werden koennen in folgendem Format aufgebaut werden:

```

NAME ! label ( adrexp ) ( @0 t0 , @1 t1 , <...> , @9 t9 )
<komm>

```

Dieser Liste entspricht folgende Aufrufzeile:

```

<:ferr> <:fer:> LINE ! label ( var0 , var1 , <...> , var9 ) <komm>
<:fer>

```

t0 bis t9 sind dabei die Variablen Typ Zeichen des LEVEL II BASIC (vgl. LEVEL II Handbuch S. 1/3 und 1/4), also:

```

! fuer Integer Variable
! fuer einfache Genauigkeit
! (normal Fall bei Eintritt in Basic)
# fuer doppelte Genauigkeit
# fuer Zeichenkette

```

X.

SUCH BEFEHLE

Um ein schnelles Auffinden von Labeln und Programmteilen zu ermöglichen, enthält MACRO-BASIC Suchbefehle fuer Zeilen, in denen Label definiert wurden, und einen Textsuchbefehl. Diese Befehle stehen fuer Drucker und Bildschirm-Ausgabe zur Verfuegung.

Alle Befehle dieser Gruppe sind wahlweise im Befehlsmodus oder unter Programmkontrolle ausfuehrbar. Bei Ausfuehrung unter Programmkontrolle koennen weitere Anweisungen in der gleichen Zeile folgen, bei Ausfuehrung im Befehlsmodus wird unmittelbar nach Abarbeitung eines Referenzbefehls in den Befehlsmodus zurueckverzweigt.

Alle Suchbefehle speichern die jeweils letzte ausgearbeitete Zeile als "laufende Zeile" fuer LIST . oder EDIT . ab. Suchbefehle, die mehrere Zeilen ausgeben, koennen mit <SHIFT > vorbeisehend angehalten und mit <BREAK> abgebrochen werden.

Die Suchbefehle verwenden den NAME Befehl in Verbindung mit einem Ausgabe-Typ-Kennzeichen. Dieses Ausgabe-Typ-Kennzeichen ist fuer alle Bildschirm Ausgaben ein # und fuer alle Drucker Ausgaben ein \*.

Die Syntaxformen sind:

<%%> <pr> NAME # label

Bildschirm Listing der Zeile, in der das Basic Label label definiert wurde.

<%%> <pr> NAME ##

Bildschirm Listing aller Zeilen, in denen Basic Label definiert wurden. In diesem Format muessen beide # unmittelbar OHNE Leerzeichen aufeinander folgen.

<%%> <pr> NAME # ! label

Bildschirm Listing der Zeile, in der das Maschinensprache Label label definiert wurde.

<%%> <pr> NAME ## !

Bildschirm Listing aller Zeilen, in denen Maschinensprache Label definiert wurden. (Anm.: In Vers. 1.0 nur im Programmmodus implementiert!)

Die Eintraesse in der Typ/Adresse Tabelle sind nur fuer so viele Eintraesse gueltig, wie im A-Register angegeben. Die uebrigen Werte haben keine Bedeutung. Die TABELLE hat folgendes Format:

ADRESSE	INHALT	zustandem in REGISTER
IV+0	Zahl der Parameter	A
IV+1	Parameter 0: Typ	C
IV+2	LSB der Adresse im UAPTR Format	L
IV+3	MSB der Adresse	R
IV+4	Parameter 1: Typ	
IV+5	LSB der Adresse im UAPTR Format	
IV+6	MSB der Adresse	
IV+7	Parameter 2: Typ	
IV+8	LSB der Adresse im UAPTR Format	
IV+9	MSB der Adresse	
IV+10	Parameter 3: Typ	
IV+11	LSB der Adresse im UAPTR Format	
IV+12	MSB der Adresse	
IV+13	Parameter 4: Typ	
IV+14	LSB der Adresse im UAPTR Format	
IV+15	MSB der Adresse	
IV+16	Parameter 5: Typ	
IV+17	LSB der Adresse im UAPTR Format	
IV+18	MSB der Adresse	
IV+19	Parameter 6: Typ	
IV+20	LSB der Adresse im UAPTR Format	
IV+21	MSB der Adresse	
IV+22	Parameter 7: Typ	
IV+23	LSB der Adresse im UAPTR Format	
IV+24	MSB der Adresse	
IV+25	Parameter 8: Typ	
IV+26	LSB der Adresse im UAPTR Format	
IV+27	MSB der Adresse	
IV+28	Parameter 9: Typ	
IV+29	LSB der Adresse im UAPTR Format	
IV+30	MSB der Adresse	



Die Wahl Moeglichkeit zwischen den beiden Arbeitsweisen wurde bereitgestellt, um

1. bei kleineren Systemen keinen zusaetzlichen Tabellen Speicherplatz zu belegen und um
2. waehrend der Phase der Programmeditierung die Zeilenentnahme im Programmmodus durchfuehren zu koennen (ohne Tabellenaufbau Zeit) und dann bei das fertiggestellte Programm mit der hoeheren Verarbeitungsgeschwindigkeit des Tabellen Modus ausfuehren zu koennen.

Nach der Initialisierung befindet sich MACRO-BASIC im Programm Modus, d.h. es beansprucht keinen Tabellenplatz.

Die Umschaltung auf den Tabellen Modus erfolgt, indem Bytes fuer die Tabelle bereitgestellt werden ueber die Anweisung:

```
<%%>> <pr> DEF USRTAB( arg ) <:pr>
```

Dabei ist das Argument arg entweder ein numerischer Ausdruck, der einen im Integer Bereich liegenden Wert ausser 0 ergibt oder das Wort LEN. Wenn LEN als Argument verwendet wird, wird genau der Speicherplatz fuer die Tabelle bereitgestellt, der fuer eine Tabelle aller ausenblicklich definierten Label benoetigt wird. Anderenfalls wird die in arg definierte Bytezahl als Tabellenplatz bereitgestellt.

Das Zuruecksetzen in den Programm Modus erfolgt ueber den Befehl

```
<%%>> <pr> DEF USRTAB( 0 ) <:pr>
```

Dabei muss 0 als Konstante gesetzt werden. Eine Variable, deren Wert 0 ist, fuehrt zu einer FC-Fehlermeldung.

Beide System Befehle erlauben innerhalb des Wortes USRTAB( mit seiner offenen Klammer keine Leerzeichen.

Wenn die System Befehle aufgerufen werden, hat das ein teilweises CLEAR zu Folge: Die Inhalte von Variablen sind seloescht, die Variablen Typ Vereinbarungen bleiben jedoch erhalten. Ausserdem sind die Werte der letzten RETURN Stack-Ebene noch verfuesbar, so dass von einer eventuellen Unterprogramm-Ebene, in der der Befehl steht, noch zurueckverweist werden kann. Der ueber CLEAR nn bereitgestellte Zeichenkatten Speicherplatz wird in seinem Umfang nicht seloendert.

```
<%%>> <pr> NAME # > string
```

Bildschirm Listins aller Zeilen, in denen der nach > stehende Suchtext string im Programmtext befunden wird.

Dabei werden Leerzeichen im Suchtext wie im Programmtext beim Veraleich nicht beruecksichtigt. Das Suchtext Ende ist das Ende der Basic Zeile, in der der Suchbefehl steht bzw. bei Aufruf im Befehlsmodus alles, was bis zum Einabeabschluss auf > folgt. Als Programmtext wird der Text durchsucht, wie er im Listins erscheint, also nicht unterschieden zwischen Text in "" oder REM-Zeilen und komprimiert abgespeicherten Basic Schlusselworten. Waehrend des Suchlaufs erscheint in der oberen rechten Ecke des Bildschirms ein blinkender Block.

Fuer Ausgabe auf einen Zeilendrucker wird bei ansonsten gleicher Syntax in allen Such Befehlen \* statt # verwendet.

## XI.

### TABELLEN BEFEHLE

MACRO-BASIC arbeitet intern mit 2 unterschiedlichen Arbeitsweisen, die vom Benutzer ueber System Befehle festgelegt werden koennen: PROGRAMM MODUS und TABELLEN MODUS.

Im Programm Modus erfolgt die Suche nach einem in Verweisungen oder Referenz-Befehlen angesprochenem Label unmittelbar im Basic Programmtext, der beginnend mit der 1. Zeile zeilenweise durchsucht wird, bis das passende Label befunden wird.

Im Tabellen Modus wird nach der Einabe jeder Programmzeile und nach jeder Editierung eine dem letzten Stand entsprechende free-format Label Tabelle aufgebaut, die neben alle definierten Label mit der zuehoerigen Adresse und einen Trennzeichen enthaelt. Bei der Suche nach einem Label wird in diesem Fall lediglich die Tabelle durchsucht, was eine Steigerung der Verarbeitungsgeschwindigkeit um das doppelte bis dreifache (je nach Programmtext) zur Folge hat. Bei der Eintragung in die Tabelle werden im Label enthaltenen Basic Schlusselworte auf ein Byte komprimiert (das Label INPUTLIST z.B. beansprucht also nur 2 Byte in der Tabelle). Die Laenge der Tabelle entpricht der Laenge aller definierten Label \* Anzahl der Label mal 3 Bytes + 1 Byte. Bei einer durchschnittlichen Label-Laenge von 6 Byte ergibt sich bei 100 Labeln eine Tabellen Laenge von 901 Byte.

## XII.

SYSTEM INFORMATIONEN BEFEHLE

Um ueber die augenblicklichen Arbeitsbedingungen des Systems Informationen zu erhalten, gibt es die Funktion

USRTAB(func)

wobei func einer folgender Befehle sein muss:

LEN FRE CLEAR # !

Der Wert, den diese Funktion liefert, kann einer Variablen zugewiesen werden oder unmittelbar auf Bildschirm oder Drucker auszugeben werden, z.B.:

? USRTAB(FRE) oder A=USRTAB(CLEAR)

Folgende Funktionen sind moeglich:

USRTAB(LEN)

ergibt die von dem Programm benoetigte Tabellen Laenge.

USRTAB(FRE)

ergibt den noch nicht belegten Tabellenplatz. Eine negative Zahl zeigt an, dass man sich im Programm Modus befindet.

USRTAB(CLEAR)

ergibt den mit DEF USRTAB(ars) bereitgestellten Tabellen Platz.

USRTAB(#)

ergibt die Zahl der definierten Basic Label

USRTAB(!)

ergibt die Zahl der definierten Maschinensprache Label.

Sinnvoll ist die Technik der Umschaltung von Tabellen- auf Programm-Modus besonders bei Programm Chainina, wo sie eine optimierte Speichernutzung ermoeglicht, wenn jedes Programm im Chainina mit DEF USRTAB(LEN) beginnt und vor dem Aufruf des folgenden Programms mit DEF USRTAB(0) abgeschlossen wird.

Wenn im Tabellen Modus mehr Label definiert wurden, als im reservierten Tabellen Platz gespeichert werden koennen, erfolgt eine OM-ERROR (Out of Memory Error) Meldung, der ein \* voraussetzt. Die Zeile wurde in diesem Fall fehlerfrei entlaesensgenommen, die Verzweigungen arbeiten aber nicht mehr sicher. Sie muessen in diesem Fall entweder in den Programm-Modus wechseln oder angemessenen Tabellenplatz bereitstellen.

Die Label Tabelle wird jeweils in den Memory-Schutz Bereich mit einbezogen. Sie liegt im Speicher zwischen MACRO-BASIC und dem String Bereich. Bei Verwendung des Tabellenmodus muss MACRO-BASIC im Speicher UNTER allen anderen eventuell verwendeten Maschinensprache Programmen liegen!

**OM-ERROR**

Der zur Verfassung stehende Speicherplatz war fuer einen Textsuch-Befehl nicht ausreichend.

Bei Einsprung in ein Maschinenprogramm ist nicht ausreichend Speicherplatz fuer 14 Stack Ebenen verfuesbar.

**\* OM-ERROR**

Der bereitgestellte Label-Tabellen-Platz reichte nicht aus

**MO-ERROR**

Ein Sprung oder Referenz Befehl wurde ohne Argument (label) verwendet

**TM-ERROR**

Das Argument von LINE USING (expr\$) ergab keine Zeichenkette.

Der Ausdruck bei LINE ON expr, ergab keinen numerischen Wert

Die Adress-Berechnung fuer Maschinenprogramme ergab keinen numerischen Wert.

Bei Parameter Uebersaben treten Typ-Widerspruche zwischen der empfangenden Variablen-Liste und der uebergebenden Parameter-Liste auf.

Die Typen-Liste einer Maschinprogramm Label-Definition stimmt nicht mit der Variablen-Liste beim Aufruf ueberein (Inteser und einfache Genauigkeit werden als unterschiedliche Typen aufgefasset!)

**FC-ERROR**

Eine vorher nicht verwendete Variable sollte an ein Maschinprogramm uebergeben werden.

Eine Maschinprogramm Einsprung Adresse wurde mit 0 definiert.

Es wurde versucht, mit einem Ausdruck statt einer Konstanten 0 Bytes Tabellen Platz bereitzustellen

Der NAME Befehl wurde ohne definiertes Trennzeichen benutzt.

## XIII.

## FEHLER MELDUNGEN

Die Basic Befehle ON ERROR GOTO und RESUME haben keine Entsprechung in MACRO-BASIC und muessen weiterhin mit Zeilennummern verwendet werden.

Wenn Fehler bei Parameteruebertrasungen auftreten, wird als Fehlerzeile immer die aufrufende Zeile angegeben.

Wenn bei der Abarbeitung von Programmen, die in MACRO-BASIC beschrieben sind, Fehler erkannt werden, wird in folgende Fehler Routinen des ROM verzweigt:

**UL-ERROR**

Ein nicht definiertes Label wurde ueber einen Referenzbefehl oder Verzweigungsbefehl angesprochen.

**SN-ERROR**

Bei einer LINE @ expr, ... Verzweigung wurde kein Komma nach expr verwendet.

Bei einer LINE USING (expr\$) Verzweigung fehlten Klammern.

Bei Parameter Uebersaben waren die Argumente nicht in Klammern eingeschlossen oder nicht durch Komma getrennt.

Bei einer Typliste in einer Maschinensprache Label-Definition wenn die Typ-Zaehlung nicht mit 0 war nicht fortlaufend oder nicht numerisch, oder Leerzeichen wurden zwischen @ und lfd. Nummer.

Bei USRTAB() Befehlen wurden unzulassige Leerzeichen verwendet.

Ein nicht zulassiges Trennzeichen wurde verwendet.

**OD-ERROR**

Bei der Parameteruebersabe stimmte die Parameter Anzahl in der aufrufenden Zeile nicht mit der in der Definitionszeile ueberein.

**OU-ERROR**

Das Argument von expr in der abhaengigen Verzweigung war groesser als 255.

Bei einer Parameter-Uebersabe wurde ein fuer die Zielvariable zu grosser Wert uebertragen.

```

450 'KOMMENTARTEXT KANN ALSO ENTWEDER
460 '- NACH LEERZEICHEN AUF EIN LABEL FOLGEN
470 '- DIFEKT ODER NACH LEERZEICHEN AUF EINE PARAMETER
480 ' LISTE FOLGEN
490 'PARAMETER LISTEN SIND WAHLWEISE MOEGLICH
500 '- UNMITTELBAR NACH DEM LABEL
510 '- ODER DURCH LEERZEICHEN GETRENNT
520 -----
530 NAME $ SEITE: PRINT: INPUT "-----> WEITER ???":A$
540 LINE ! SEITE BEMERKUNG: RETURN
550 'DIE LETZTE ZEILE RUFT DAS MASCHINENPROGRAMM "SEITE"
560 'AUF, TRENNZEICHEN IST "!", DAS LABEL "SEITE"
570 'WURDE ALSO FUER EIN MASCHINENPROGRAMM (MIT "!")
580 'UND EIN UNTERPROGRAMM (MIT "$") VERWENDET
590 NAME $ INFO UEBER LABEL ODER TEXTE: PRINT
600 'LABEL IST NUR "INFO" !
610 PRINT "I N F O"
620 '==> ALLE SUCH BEFEHLE WERDEN MIT "*" STATT "#" AUF DEN
630 ' DRUCKER AUSGEGEBEN !
640 PRINT "SUCHE NACHE DER DEFINITION VON
'BEISPIEL1":PRINT
650 NAME # BEISPIEL1 'LISTET ZEILE "BEISPIEL1"
660 'LABELENDE DURCH DAS LEERZEICHEN NACH "BEISPIEL1"
670 LINE$ SEITE
680 PRINT "LISTE ALLER LABEL:":PRINT
690 NAME ##
700 LINE $ SEITE
710 PRINT "LISTE ALLER MASCHINENPROGRAMM-LABEL:":PRINT
720 NAME ##!
730 'DAS "!"-ZEICHEN WIRD ALS KENNZEICHEN FUER
740 'MASCHINENSPRACHE-AUFRUFE UND -LABEL VERWENDET
750 LINE $SEITE
760 PRINT "BEREITGESTELLTER TABELLEN PLATZ:":
770 PRINT USRTAB(CLEAR) "BYTES"
780 PRINT "WENN 0 BYTES BEREITGESTELLT WURDEN, ARBEITET"
790 PRINT "DAS SYSTEM OHNE TABELLE": PRINT
800 PRINT "FUER EINE TABELLE WUERDEN":
810 PRINT USRTAB(LEN) "BYTES BENOETIGT": PRINT
820 PRINT USRTAB(FRE) "BYTES DES BEREITGESTELLTEN ":
830 PRINT "TABELLENPLATZES SIND FREI."
840 PRINT "OHNE BEREITGESTELLTEN TABELLENPLATZ IST ":
850 PRINT "DIESE ZAHL NEGATIV": PRINT
860 PRINT USRTAB(0) "BASIC-LABEL WURDEN DEFINIERT":
PRINT
870 PRINT USRTAB(1) "MASCHINENSPRACHE LABEL WURDEN DEF."
880 ' "USRTAB(0) MUSS OHNE LEERZEICHEN GESCHRIEBEN WERDEN!
890 LINE$ SEITE: RETURN 'LABEL SCHLUSSZEICHEN ":
900
910 NAME $ TEXTSUCHE BEISPIELE: PRINT "DIESER TEXT WIRD
GESUCHT:"
920 NAME #> INFO
930 'ALLES WAS AUF "#>" FOLGT BIS ZUM ZEILEN ENDE IST
940 'SUCHTEXT. DIE GLEICHEN ZEILEN WERDEN GEZEIGT MIT:
950 NAME #> I N F O
960 'LEERZEICHEN IM TEXT WERDEN ALSO IGNORIERT
970 'WENN DER TEXTSUCHBEFEHL IM PROGRAMM STEHT, WIRD DIE
980 'GERADE AKTIVE SUCHBEFEHLS ZEILE SELBER NICHT GEZEIGT
990 LINE$ SEITE: PRINT "AUFRUFE DES UP 'SEITE':"
1000 NAME#> LINE$SEITE
1010 PRINT: PRINT "BITTE ACHTEN SIE AUF DIE LEERZEICHEN !"

```

```

*****
** DEMO **
*****

```

Das MACRO-BASIC Programm "DEMO" und das Assembler Programm "APPAY" wurden der Dokumentation beifuegt, um die dort detailliert beschriebenen Moeglichkeiten zu veranschaulichen. Die Zeilen 1240 bis 1570 von "DEMO" beziehen sich auf "APPAY". "DEMO" befindet sich auf der Fueckseite der Cassette mit MACRO-BASIC. Sie koennen es als erste Einfuehrung in den Befehlsatz der Spracherweiterung benutzen.

## PROGRAMM LISTING

=====

```

100 'DEMO
110 CLS: PRINT "BEISPIELPROGRAMM FUER MACRO-BASIC"
120 PRINT: "(C) KLAUS DAMM 1981"
130 PRINT@ 576, "DIESES PROGRAMM IST ZUM LESEN DES
PROGRAMMTEXTES"
140 PRINT "GEDACHT, DER AUSFUEHRLICH KOMMENTIERTE BEISPIELE
FUER"
150 PRINT "DIE NEUEN BEFEHLE VON MACRO-BASIC ENTHAELT."
160 PRINT: PRINT: INPUT "WEITER ": F$
170 CLEAR 350 'ZEICHENKETTENPLATZ RESERVIEREN
180 DEF USRTAB(0) 'PROGRAMM MODUS STATT TABELLEN MODUS
190 LINE @ START 'ENTSPRICHT GOTO 1590 "NAME $ START"
200 'TRENNZEICHEN FUER SPRUNGANWEISUNG IST "@"
210 -----
220 'BEISPIEL UNTERPROGRAMME
230 NAME $ BEISPIEL1 'LABEL DEFINITION OHNE PARAMETER
PRINT "BEISPIEL1"
240
250 RETURN 'ABSCHLUSS WIE NACH GOCUB AUFRUF
260 NAME $ BEISPIEL2(I,X$)
270 PRINT "BEISPIEL2: UNTERPROGRAMM MIT 2 PARAMETERN"
280 PRINT "1. PARAMETER: "; I, "2. PARAMETER: "; X$
290 RETURN
300 NAME $ BEISPIEL3 (I) BEMERKUNG: PRINT "BEISPIEL3:"
310 PRINT "UNTERPROGRAMM MIT EINEM PARAMETER: "; I
320 RETURN 'DER TEXT "BEMERKUNG" IN BEISPIEL3 IST
330 'WAHLWEISE MOEGLICHER KOMMENTAR, DER MIT
340 'EINEM : ABGESCHLOSSEN WIRD, DANACH
350 'FOLGT AUSFUEHRBARER PROGRAMMTEXT
360 NAME $ BEISPIEL4 BEMERKUNG: PRINT "BEISPIEL4": RETURN
370 'DER TEXT "BEMERKUNG" IST KOMMENTAR, DER DURCH EIN
380 'LEERZEICHEN VOM LABEL "BEISPIEL4" ABGETRENNT WIRD
390 NAME $ BEISPIEL5(X$) BEMERKUNG: PRINT "BEISPIEL5"
400 PRINT "PARAMETER: "; X$: RETURN
410 'LEERZEICHENVERWENDUNG ZWISCHEN NAME $ UND LABEL
420 'IST BELIEBIG, DAS LABEL HEISST "BEISPIEL5"
430 'LABEL ABSCHLUSS IST DIE PARAMETER-LISTE "(X$,I)"
440

```

```

1620 *AUFRUF DES UNTERPROGRAMMS "INFO" OHNE PARAMETER
1630 *"IRGEND EIN KOMMENTAR" MIT LEERZEICHEN ABGETRENNT
1640 *TRENNZEICHEN FUER UNTERPROGRAMM AUFRUF IST "$"
1650 LINE# TEXTSLICE NACH ZEICHENKETTEN IM PROGRAMM-PRINT"OK"
1660 *UNTERPROGRAMM NAME IST "TEXTSUCHE", BIS ZUM ":" IST
1670 *KOMMENTARTEXT, DANACH PROGRAMMTEXT
1680 PRINT
1690 PRINT "JETZT FOLGEN BEISPIELE FUER UNTERPROGRAMM-AUFRUFE"
1700 PRINT "MIT PARAMETERN": PRINT
1710 INPUT "GEBEN SIE EINE BEISPIEL ZEICHENKETTE EIN": Z$
1720 INPUT "GEBEN SIE EINE BEISPIEL ZAHL EIN": Z: PRINT
1730 LINE$ BEISPIEL2(Z,Z$) BEMERKUNG: PRINT "ZURUECK"
1740 *UNTERPROGRAMM NAME IST "BEISPIEL2"
1750 *Z WIRD AN I UND Z$ AN X$ UEBERGEBEN
1760 *"BEMERKUNG" KANN WEGFALLEN
1770 PRINT: PRINT "JETZT WERDEN AUSDRUECKE UEBERGEBEN:"
1780 LINE$ BEISPIEL2 (Z*2, Z$+"---"+Z$): PRINT "ZURUECK"
1790
1800 NAME$ MENU: LINE$ SEITE
1810 PRINT, "*** M E N U ***": PRINT
1820 PRINT "<1> BIS <5> AUFRUF VON UNTERPROGRAMMEN"
1830 PRINT "<6> AUFRUF DES MASCHINENPROGRAMMS 'SEITE'"
1840 PRINT "<7> SPRUNG ZUM PROGRAMMTEIL 'ARRAY'"
1850 PRINT "<8> VERZWEIGUNGEN NACH ZEICHENKETTEN TESTEN"
1860 PRINT "<9> PROGRAMM ENDE": PRINT
1870 INPUT "WAELHEN SIE NUN BITTE PROGRAMM TEILE AN ": A$: PRINT
1880 *JETZT ERFOLGT VON EINEM NUM. WERT ABHAENIGE VERZWEIGUNG:
1890
1900 LINE ON VAL(A$), $BEISPIEL1 OHNE PARAMETER
    $BEISPIEL2(123,"TESTWERT") MIT PARAMETERN
    $BEISPIEL3 (100+VAL(A$)) AUSDRUCK ALS PARAMETER
    $BEISPIEL4 $BEISPIEL5("TEXT")
    $SEITE MASCH. PRGRM
    $ARRAY $ZEICHENKETTEN ENDE
1910 *DER AUSDRUCK WIRD MIT "," ABGESCHLOSSEN, ES FOLGT EINE
1920 *VERWEIGUNGSLISTE, IN DER AUFRUFE MIT UND OHNE PARAMETER
1930 *GEMISCHT WERDEN KOENNEN.
1940 *TEILWEISE WURDEN KOMMENTARE EINGEFUEGT, ZEILENVORSCHUEBE
1950 *SIND EBENFALLS MOEGLICH, ALLE LABEL WERDEN MIT "(" ODER
1960 *LEERZEICHEN ABGESCHLOSSEN.
1970 *VOR JEDEM LABEL STEHT EIN VERZWEIGUNGSTYP-TRENNZEICHEN,
1980 *ALSO "$" FUER UNTERPROGRAMM, "@" FUER SPRUNG
1990 *UND "!" FUER MASCHINENPROGRAMM.
2000 *ALLE VERZWEIGUNGSTYPEN SIND IN EINER LISTE
2010 *GEMISCHT MOEGLICH !
2020 LINE @ MENU
2030
2040
-----
2050 NAME $ZEICHENKETTEN DEMONSTRATION: LINE!SEITE
2060 PRINT "BEISPIEL FUER VERZWEIGUNG NACH STRING INHALT": PRINT
2070 Z$="(C) KLAUS DANN 1981"
2080 UP$="BEISPIEL2" LABEL FUER VERZWEIGUNG
2090 FA$="(3,Z$)" *3 WIRD ERSTER, Z$ WIRD 2. PARAMETER
2100 LINE USING ("$"+UP$+PA$) BEMERKUNG: PRINT
2110 *EIN STRING AUSDRUCK WIRD AUS FOLGENDEN TEILEN GEBILDET:
2120 *EINEN VERZWEIGUNGSTYP ZEICHEN, HIER "$"
2130 *EINEM STRING, DER DAS LABEL ENTHAELT
2140 *GGF. EINEM STRING, DER EINE PARAMETER LISTE ENTHAELT
2150 *DIE STRING VERKNUEPFUNG KANN IM AUFRUFENDEN BEFEHL
2160 *EFFOLGEN WIE OBEN.

```

```

1020 LINE#SEITE:RETURN
1030
-----
1040 *BEISPIELE FUER MASCHINENPROGRAMM DEFINITIONEN
1050 NAME ! SEITE (457) BEMERKUNG
1060 *ALLE LEERZEICHEN IN DER LERZTEN ZEILE BELIEBIG !
1070 *"!" KEINZEICHEN FUER MASCHINENSPPACHE DEFINITION
1080 *"SEITE" IST LABEL
1090 *(457) IST DIE ADRESSE IN DEZIMAL (ROM CODE "CLS")
1100 *"BEMERKUNG" IST KOMMENTAR
1110 *PARAMETER WURDEN NICHT UEBERGEBEN
1120 NAME ! DUMMY (457) (00%, 01$) BEMERKUNG=
1130 *DIESES BEISPIEL UEBERGIBT 2 PARAMETER UEBER DAS
1140 *IV-REGISTER AN DAS MASCHINENPROGRAMM "DUMMY"
1150 *DIE VARIABLEN, DEREN ADRESSEN ALS PARAMETER UEBER-
1160 *GEBEN WERDEN, MUESSEN VOR DEM AUFRUF EINEN INHALT
1170 *ZUGEWIESEN BEKOMMEN HABEN
1180 NAME $ DUMMY-CALL (4%, X$): LINE ! DUMMY (4%, X$): RETURN
1190 *"DUMMY-CALL" IST EIN BASIC UNTERPROGRAMM
1200 *ES LIST WERTE IN DIE VARIABLEN A$ UND X$ EIN
1210 *UND RUFT DAS MASCHINENPROGRAMM "DUMMY" AUF
1220 *A$ UND X$ WERDEN AN "DUMMY" UEBERGEBEN
1230
-----
1240 ***** SCHNELLE ARRAY BEFEHLE *****
1250 *FUELLEN EINES ARRAY MIT EINER KONSTANTEN "FILL"
1260 *KOPIEREN EINES ARRAYS IN EINEN ANDEREN "COPY"
1270 *DAS ASSEMBLER PROGRAMM DAZU LIEGT ALS SOURCE LISTING
1280 *DER DOCUMENTATION BEI, SO DASS ES LEICHT ERWEITERT
1290 *WERDEN KANN !
1300 *!!!!!!!!!!!! DIE AUFRUFENDEN ZEILEN DUEFFEN SIE NUR
1310 *!!!!!!!!!!!! BENUTZEN, WENN SIE DAS BEISPIEL
1320 *!!!!!!!!!!!! PROGRAMM "ARRAY.CMD" AUF 0F200 ASSEMBLIERT
1330 *!!!!!!!!!!!! IN DEN SPEICHER GELADEN HABEN.
1340 *!!!!!!!!!!!! AENDERN SIE GGF. DIE ADRESSE IN DEN NAECHSTEN
1350 *!!!!!!!!!!!! ZEILEN, WENN IHR SYSTEM KLEINER IST
1360 NAME ! FILL (61952) (00%, 01%, 02%)
1370 NAME ! COPY (-3584) (00%, 01%, 02%, 03%)
1380
1390 *"FILL" UND "COPY" BEGINNEN BEIDE BEI 0F200H
1400 *DIESE ADRESSE IST POSITIV DEZIMAL 61952, NEGATIV -3584
1410 *BEIDE ADRESS-DEFINITIONEN SIND MOEGLICH
1420
1430 NAME $ ARRAY BEISPIELE: CLEAR 350 *NICHT FEDIENSIONIEREN
1440 *"ARRAY" NICHT ALS UNTERPROGRAMM AUFRUFEN WG. CLEAR !
1450 DIM A% (250), B% (250) *ZWEI INTEGER ARRAYS SCHAFFFEN
1460 U%=5 *U% IST INTEGER FUELLWERT, HIER 5
1470
1480 *AUFRUFBEISPIEL FUER "FILL" UND "COPY", ("!" ENTFERNEN!)
1490 *LINE! FILL (A%(0), A%(250), U%) FUELLE ARRAY A% MIT U%
1500 *GEFUELLT WERDEN ELEMENT 0 BIS 250
1510 *LINE! COPY (B%(10), B%(100), A%(0), A%(90))
1520 *DIE ELEMENTE 0 BIS 90 VON ARRAY A% WERDEN KOPIERT
1530 *IN DIE ELEMENTE 10 BIS 100 VON ARRAY B%
1540 FOR I=1 TO 250: PRINT B%(I):+NEXT
1550 LINE$ SEITE: LINE@ MENU
1560 *RUECKSPRUNG WG. CLEAR VOR DER DIMENSIONIERUNG
1570
1580
-----
1590 NAME $ START: CLS
1600 *BEISPIELE FUER UNTERPROGRAMM-AUFRUFE
1610 LINE# INFO IRGEND EIN KOMMENTAR

```

```
*****
**                **
**   ARRAY       **
**                **
*****
```

Das folgende Assembler Programm ARRAY demonstriert, wie leistungsstark die Parameter-Übergabe von Maschinensprache Programmen in MACRO-BASIC genutzt werden kann und wie die Parameter von dem aufgerufenen Maschinenprogramm in Empfang genommen werden können.

ARRAY dient zum Kopieren und Füllen von numerischen Arrays. Beide Funktionen werden mit der gleichen Startadresse angesprochen und durch die Zahl der Parameter unterschieden.

Geben Sie das folgende Assembler Programm in Ihren EDTASM ein. Ersetzen Sie dann in

```
00200 PAGE    EQU    0F200H
```

die Start und Einsprung Adresse durch die Adresse, an die Sie das Programm lesen möchten und assemblieren Sie es mit der geänderten Adresse. Tragen sie die geänderte Adresse in dem MACRO-BASIC Programm "DEMO" in Zeile 1360 und 1370 ein und entfernen Sie dort die "" Zeichen in Zeile 1490 und 1510. Wenn Sie mit Diskette arbeiten können Sie anfangs die Zeile

```
01250 START0 CALL    4400H ;FUER TEST: DEBUG MIT DISK
```

im Assembler Listing stehen lassen und die Parameterübergabe im Single-Step verfolgen. Für eine Arbeitsversion und unter LEVEL II löschen Sie bitte diese Assembler Zeile, bevor Sie das Programm assemblieren. Sollten Sie keinen Assembler haben, kann ich Ihnen auf Wunsch gegen einen Unkostenbeitrag von DM 12,- eine nach Ihren Wünschen assemblierte Version als SYSTEM-Band auf Cassette zusenden.

Mit einem Disk-System laden die unter das Maschinenprogramm unter DOS, initialisieren dann DISK-BASIC mit der Startadresse als Memory-Size und geben dann "RUN MACRO" ein. Laden Sie "DEMO", ändern Sie ggf. die Adresse in Zeile 1360 und 1370. Jetzt können Sie Array anwenden, um Arrays zu füllen oder zu kopieren.

Mit einem Cassetten-System assemblieren Sie mit

```
05600      END    1A19H      ;BASIC READY
```

und Speichern Sie das Maschinenprogramm als System-Band ARRAY. Drucken Sie RESET und geben als MEMORY SIZE die Startadresse ein (s.o.). Laden Sie MACRO BASIC mit CLOAD, starten Sie es mit RUN und laden dann unter MACRO BASIC das Beispielprogramm "DEMO". Ändern Sie die Zeilen 1360, 1370, 1490 und 1510 wie

```
2170 *DER STRING KANN AUCH VORHER GEBILDET WERDEN:
2180 U$=#BEISPIEL3 (9)"
2190 LINE USING( U$)
2200 LINE USING("#SEITE") RUFT UEBER STRING-KONSTANTE-AUF
2210 *AUCH MASCHINENPROGRAMME KOENNEN SO AUFGERUFEN WERDEN:
2220 LINE USING ("! SEITE"): RETURN
2230 *DURCH STRING VERARBEITUNGSTECHNIKEN LASSEN SICH MIT DIESEM
2240 *VERZWEIGUNGSMECHANISMUS FUER EINEN FORTGESCHRITTENEREN
2250 *PROGRAMMIERER MACRO-SPRACHEN AUFBAUEN, INDEM
2260 *STRING-ARRAYS MITEINANDER VERKNUEPFT WERDEN UND DIE
2270 *ABLAUFSTEUERUNG UEBERNEHMEN
2280 *
2290 NAME $ ENDE :CLS
2300 CLS: PRINT," ..... WENN SIE DIESES BEISPIEL-PROGRAMM"
2310 PRINT"ZU MACRO BASIC KENNENGELERNT HABEN, KOENNEN SIE SICH
SICHER"
2320 PRINT"VORSTELLEN, WIEVIELE STUNDEN ARBEIT ES GEKOSTET HAT,
DIESE"
2330 PRINT"ERWEITERUNG VON BASIC ZU ENTWICKELN UND
AUSZUTESTEN, "
2340 PRINT
2350 PRINT"WENN ES IHNEN GEFAEHLT, EMPFEHLEN SIE DAHER BITTE
IHREN"
2360 PRINT"FREUNDEN UND BEKANNTEN, ES ALLEIN ODER ZU ZWEIT BEI
MIR"
2370 PRINT"ZU KAUFEN, GEBEN SIE ES BITTE NUR IN AUSNAHMEN
WEITER !"
2380 PRINT"DER PREIS IST DOCH AUCH FUER EINEN HOEBYISTEN
TRAGBAR ... "
2390 PRINT
2400 PRINT"MACRO-BASIC IST BEWUSST NICHT GEGEN KOPIEREN
GESCHUETZT, "
2410 PRINT"DAMIT SIE SICH SICHERUNGSKOPIEN MACHEN KOENNEN, "
2420 LINE# SEITE: PRINT"DAS WARS .... "
2430 PRINT
2440 PRINT"ICH HOFFE, DIESES BEISPIEL PROGRAMM HILFT IHNEN, "
2450 PRINT"DEN EINSTIEG IN EINE KOMFORTABLE UND
LEISTUNGSSTARKE"
2460 PRINT"ERWEITERUNG VON BASIC ZU FINDEN, ": PRINT
2470 PRINT"EINE SYSTEMATISCHE UERTIEFENDE DARSTELLUNG GIBT
IHREN"
2480 PRINT"DIE AUSFUEHRLICHE DOKUMENTATION, ": PRINT
2490 PRINT"SOLLTEN SIE DENNOCH FRAGEN HABEN, KOENNEN SIE SICH"
2500 PRINT"IN ZWEIFELSFAELEN AN MICH DIREKT WENDEN:"
2510 PRINT: PRINT"KLAUS DAMM, REHWEG 8, 5354 WEILERSWIST 2"
2520 PRINT:PRINT," ... VIEL SPASS !"
2530 END
```

```

02650 LD H,(IV+12)
02700 LD L,(IV+11) :=>QUELL ARRAY END
02750 LD D,(IV+9)
02800 LD E,(IV+8) :=>QUELL ARRAY START
02850 SBC HL,DE :HL=QUELL BYTE COUNT - TP
02900 ADD A,L
02950 LD L,A :ADD LSB OF LEN
03000 LD A,H
03050 ADC A,0 :CORRECT MSB
03100 LD H,A :HL=CORRECT LEN
03150 SBC HL,BC :CHK IF DEST LEN=SRC LEN
03200 POP HL :RESTORE DESTINATION
03250 RET NZ :ERROR IF COUNT NOT EQU
03300 EX DE,HL :HL=>QUELL ARRAY START
03350 :DE=>ZIEL ARRAY START
03400 LDIR :MOVE SOURCE TO DEST
03450 POP HL :WIPE OUT ERROR EXIT
03500 RET
03550
03600 FILL CP 3 :CHECK IF STRING
03650 RET Z : YES: ERROR
03700 POP HL :WIPE OUT ERROR EXIT
03750 LD H,(IV+9)
03800 LD L,(IV+8) :=>SOURCE UAP
03850 PUSH BC :ARRAY LEN
03900 LD B,0
03950 LD C,A :TYPE =COUNT FOR LDIR
04000 EXX :SET UP LOOP COUNT IN 2
04050 POP HL :GET ARRAY LEN
04100 RRA :SHIFT OUT BIT 0=0
04150 RRA :AT LEAST 1 SHIFT REQU
04200
04250 DIULP SRL H :MSB/2
04300 RR L :LSB/2 MIT CARRY
04350 RRA :MORE DIV. REQU ?
04400 JR C,DIULP :DO IF SO
04450 :ELSE: HL=COUNT
04500
04550 MULP EXX :=>MOVE PARAM
04600 PUSH BC :VAR. LEN
04650 PUSH HL :SOURCE PNT
04700 LDIR :MOVE INTO ARRAY
04750 POP HL
04800 POP BC :RESTORE COUNT & SOURCE
04850 EXX
04900 DEC HL
04950 LD A,H
05000 OR L
05050 JR NZ,MULP
05100 RET
05150
05200 FARERR LD HL,ERRMG
05250 CALL NGOUT
05300 JP SNERR
05350
05400 ERRMG DEFB 00H
05450 DEFH 'ARRAY PARAMETER
05500 NOP :TERMINATOR
05550
05600 END 1A19H :BASIC READY

```

beschrieben. Mit einem 16K System muessen Sie je nach Startadresse ggf. einige Remark Zeilen loeschen oder die Arrays kleiner dimensionieren, um zureichend Speicherplatz zu haben.

## ASSEMBLER LISTING

=====

```

00100 :ARRAY/ASS AF/CMD U.08.07.81/2
00150
00200 PAGE EQU 0F200H
00250
00300 NGOUT EQU 28A7H
00350 SNERR EQU 1997H
00400
00450
00500 :COPY ARRAY
00550 :FORMAT COMPATIBEL ZU MACRO-BASIC
00600 :ENTRY: 4 PARAMETER, TYPGLEICH
00650 :PARAM 1: ZIEL ARRAY ANFANG
00700 : 2: ENDE
00750 : 3: QUELL ARRAY ANFANG
00800 : 4: ENDE
00850
00900 :FILL ARRAY
00950 :ENTRY: 3 PARAMETER, TYPGLEICH
01000 :PARAM 1: ZIEL ARRAY ANFANG
01050 : 2: ENDE
01100 : 3: QUELL VARIABLE
01150
01200 CPB PAGE
01250 STARTO CALL 4400H :FUER TEST: DEBUG MIT DISK
01300 START LD DE,FARERR
01350 PUSH DE :SET ERROR EXIT
01400 CP 3 :CHECK # OF PARAM
01450 JR NZ,COPY
01500 LD IX,FILL :CORRECT EXIT
01550 JR FARCHK :CHECK PAR. 1 & 2
01600 COPY CP 4
01650 RET NZ :EFF IF NOT 2 OR 4 PAR
01700 LD IX,COPY0 :SET EXIT FOR COPY
01750 FARCHK LD A,(IV+1) :1. TYP
01800 CP (IV+4) :2. TYP
01850 RET NZ
01900 CP (IV+7) :3. TYP
01950 RET NZ
02000 LD D,(IV+6) :MSB OF FAR. 2 ADP
02050 LD E,(IV+5) : & LSB
02100 EX DE,HL :HL=ZIEL ARRAY END
02150 :DE=ZIEL ARRAY START
02200 SBC HL,DE :HL=BYTE COUNT - TYP
02250 LD C,A :BC=TYP
02300 ADD HL,BC :HL=BYTE COUNT
02350 LD B,H
02400 LD C,L :BC=BYTE COUNT
02450 JP (IX) :GO TO COPY0 OR FILL
02500 COPY0 CP (IV+10) :CHECK TYPE 4
02550 RET NZ
02600 PUSH DE :SAVE ZIEL ARRAY START

```

MACRO-BASIC  
V. 2.2  
Copyright (c) Klaus Damm 1982

Dokumentation Teil 2:

Erweiterungen gegenüber Version.1.x

INHALT

=====

- I Zusammenfassung: V.2.x gegenüber V.1.x
- II Verwendung von '.'
- III 'WHILE'
- IV BASE:  
Basis-Adresse für Benutzer-Eingriffe
- V Schnittstelle für benutzerdefinierte Basic-Befehle
- VI AUTOSTART:  
Schnittstelle für Start eines Anschlußprogramms  
nach Initialisierung von MACRO-BASIC
- VII Tonausgabe über Cassetten-Interface
- VIII Sonstiges



## I ERWEITERUNGEN GEGENÜBER V.1.x

MACRO-BASIC Version 2.x ist voll abwärts kompatibel zu Vorfassungen. Jedoch wurden einige Erweiterungen vorgenommen, um die Handhabung für den Anwender noch komfortabler zu gestalten:

- \* Es ist nun durchweg möglich, anstelle des Schlüsselwortes 'LINE' das Sonderzeichen '.' zu verwenden. Programme werden dadurch noch übersichtlicher.
- \* Um strukturierte Programmierung weiter zu unterstützen, wurde eine WHILE Schleifenanweisung implementiert.
- \* Zusätzlich zu dem Maschinensprache Interface 'LINE ! Label' wurde eine Benutzerschnittstelle zum LEVEL II Interpreter geschaffen, der es dem erfahrenen Programmierer ermöglicht, eigene neue Basic Schlüsselworte zu implementieren.
- \* MACRO-BASIC übergibt nach der Initialisierung eine Bezugsadresse-BASE, die für Änderungen an MACRO-BASIC durch den Anwender verwendet werden kann.
- \* Besonders für die Erstellung von MACRO-BASIC Anwenderprogrammen unter Disk-Basic ist die Möglichkeit von Vorteil, ab Version 2.2 nach der Initialisierung von MACRO-BASIC ein Anschlußprogramm starten zu können, ohne das in den Befehlsmodus verzweigt wird.
- \* Eine Tonausgabe-Routine ermöglicht Unterstützung von Bildschirmmeldungen und ermöglicht die Komposition eigener Melodien.

### Hinweise:

MACRO-BASIC V.1.0 benutzte die OUT OF DATA Fehlermeldung, wenn bei der Parameterübergabe an Verzweigungsziele Programmierfehler entdeckt wurden. Diese Fehlerbehandlung konnte ein unkontrolliertes Verhalten zur Folge haben, wenn DATA Statements im Programm standen. Daher wurde sie in Folgeversionen durch die Fehlermeldung ILLEGAL FUNCTION CALL ersetzt.

Die Anweisung NAME ##! <Liste aller Maschinensprache-Label> arbeitet ab V.2.x sowohl im Tabellen- als auch im Programmmodus.

Die Funktion USRTAB(#) (Zahl der definierten Label) ergibt in V.2.2 nicht wie vorgesehen die Zahl der Label, sondern immer 0. Systemfehler treten dadurch nicht auf.

## II '.' STATT 'LINE'

Das Wort 'LINE' kann sowohl in Verbindung mit MACRO-BASIC Befehlen als auch in Verbindung mit den Graphik-Befehlen von LEVEL III BASIC (PCS oder Microsoft) durch einen Punkt '.' (ASCII 46 dezimal) ersetzt werden.

So vereinfacht sich z.B. der Unterprogrammaufruf mit Parametern

```
LINE $ AUSGABE ("TEXT"+B$)
```

zu

```
.$ AUSGABE ("TEXT"+B$)
```

In wenigen Fällen wird diese Kurzschreibweise allerdings nur dann akzeptiert, wenn sie unmittelbar oder durch Leerzeichen getrennt auf das Trennzeichen ':' folgt.

Das ist z.B. der Fall im Befehlsmodus und nach 'THEN'.

In diesen Fällen lautet das obige Beispiel also:

```
:.$ AUSGABE ("TEXT"+B$)
```

bzw.

```
IF A=3 THEN :.$ AUSGABE ("TEXT"+B$)
```

Bei der Benutzung von LINE anstelle von '\$' entfällt die Notwendigkeit, einen ':' einzufügen.

In der folgenden Dokumentation der Erweiterungen von V.2.2 gegenüber V.1.1 wird durchweg '\$' statt LINE verwendet. In allen Fällen ist LINE gleichwertig möglich.

### III WHILE

Mit Hilfe der WHILE Anweisung kann ein Schleifenkörper abhängig von einer Abbruchbedingung wiederholt durchlaufen werden. Der Schleifenkörper muß in der MACRO-BASIC Implementierung dieser Kontrollstruktur ein mit 'NAME\$ label' definiertes Unterprogramm sein.

Die WHILE Anweisung besteht aus einem Bedingungsteil und einem Unterprogrammaufruf (Schleifenkörper). Sie arbeitet wie folgt:

- Vor Aufruf des Unterprogramms wird geprüft, ob die Bedingung wahr ist.
- Wenn die Bedingung wahr ist, wird das Unterprogramm (ggf. mit Parametern) aufgerufen.
- Nach Durchlaufen des Unterprogramms, das wie gewohnt mit RETURN abgeschlossen wird, erfolgt eine erneute Bedingungsprüfung.
- Dieser Vorgang wird so lange wiederholt, bis die Bedingung nicht mehr wahr ist.

Dieses Verfahren bietet im Vergleich zur FOR NEXT Schleife in vielen Fällen erhebliche Vorteile:

- Im Unterschied zur FOR NEXT Schleife wird der Schleifenkörper also nicht wenigstens einmal durchlaufen !
- Während bei der FOR NEXT - Konstruktion lediglich ein Zählergrenzwert als Bedingung geprüft werden kann, lassen sich in der WHILE Konstruktion beliebige Bedingungen oder Verknüpfungen von Bedingungen verwenden.

Im Unterschied zur GOTO Schleife:

```
%%%% IF cond THEN GOSUB ZZZZ: GOTO %%%%
```

muß in der in MACRO-BASIC WHILE Anweisung der Rücksprung zur Bedingungsprüfung nicht programmiert werden. Vor und nach der WHILE-Anweisung können wahlweise weitere Programmanweisungen in derselben Zeile stehen. Die Anweisung ist auch im Befehlsmodus möglich.

Die Formulierung dieser Konstruktion in MACRO-BASIC mit WHILE hat folgende Syntax:

```
%%%%< bgr> :: . WHILE cond , $ label <(param) komm> <: bgr>
```

Dem Schlüsselwort 'WHILE' muß ein Punkt '.' oder das Schlüsselwort 'LINE' vorausgehen. Das Trennzeichen '::' (2 Doppelpunkte) vor '.' oder 'LINE' ist verbindlich, auch wenn keine weiteren Anweisungen vor 'WHILE' in der Zeile verwendet werden!

cond kann jede Bedingung sein, die in einer IF Anweisung verwendet werden kann.

Der Aufruf des Schleifenkörpers erlaubt sämtliche Formen, wie sie für Unterprogrammaufrufe unter MACRO-BASIC zulässig sind, also nach dem Trennzeichen '\$' eine wahlweise Parameterliste und wahlweiser Kommentartext, der bei fehlender Parameterliste durch ein verbindliches LZ von Label getrennt werden muß.

#### IV BASE

MACRO-BASIC ist bei der Initialisierung selbst relocierend. Um für Änderungen durch den Benutzer und Ausnutzung der Benutzer-Schnittstellen-Decodierung eine Bezugsadresse für die tatsächliche Lage im Speicher zu haben, wird von MACRO-BASIC eine Bezugsadresse BASE an den Anwender übergeben. Diese Adresse ist an den Speicherstellen

```
16526 = 408EH (least significant Byte) und  
16527 = 408FH (most significant Byte)
```

abgelegt. Sie bleibt dort solange unverändert, wie ein Benutzerprogramm keine USR-Funktion definiert.

BASE kann mit folgendem Programmsegment in eine Basic Integer Variable eingelesen werden:

```
100 BASE% = 0  
110 AB = VARPTR (BASE%)  
120 U = 16526  
130 POKE AB, PEEK (U)  
140 POKE AB+1, PEEK (U+1)  
150 PRINT BASE%
```

Da Basic Integer Variable mit Vorzeichen interpretiert, sind die Adressen im Expansion-Interface dezimal negativ.

## V BENUTZERDEFINIERBARE BEFEHLE

Macro-Basic ermöglicht die Verwendung von '...' als Code für den Interpreter, die Zeichen bis zum nächsten ':' bzw. Zeilenende als einen vom benutzerdefinierten Basic Befehl aufzufassen, der in einem vom Benutzer erstellten Maschinenprogramm interpretiert und ausgeführt wird.

Im Leerzustand wird durch den '...' die Einfügung eines Kommentars zwischen zwei Programmweisungen in folgender Form ermöglicht:

```
%%>>> pgr: .. Kommentar : pgr
```

Der Kommentar wird dadurch erkannt, daß er durch zwei Punkte '...' eingeleitet wird.

Für **erfahrene** Assembler-Programmierer bietet MACRO-BASIC außer der Möglichkeit, Maschinensprache Unterprogramme mit '.' label aufzurufen ab Version 2.2 die Möglichkeit, die Sprache nach eigenen Vorstellungen zu erweitern, ohne selber eine Schnittstelle zum Interpreter entwickeln zu müssen.

Ein benutzerdefinierter Befehl wird von Basic aus mit der Syntax:

```
<%%>>> <pgr :> .. usercmd <komm> <pgr>
```

oder

```
<%%>>> <pgr :> LINE. usercmd <komm> <pgr>
```

aufgerufen.

Die vom Benutzer für Erweiterungen vorgesehenen Befehle und deren Parameter werden durch usercmd repräsentiert. Falls mehrere Erweiterungen vorgesehen sind, muß die Erweiterungsroutine zunächst die Dekodierung von usercmd vornehmen.

MACRO-BASIC vollzieht die Vordekodierung der Benutzerschnittstelle '...', übernimmt die Registersicherung und verzweigt an die Adresse, die in

```
BASE+1 + 0BECH (least significant Byte) und  
BASE+1 + 0BEDH (most significant Byte)
```

steht.

Das HL Registerpaar zeigt bei Eintritt in die dort indizierte Routine auf das erste Zeichen von usercmd nach eventuell führenden LZ.

Die Benutzerroutine muß den Stack bei Verlassen im Eingangszustand rückübergeben und mit RET abschließen.

Die Ausdekodierung des Befehls und der Parameter sowie die Sicherstellung eines lokalen Stack- und Speicherbereichs liegen bei der Ausnutzung dieser Möglichkeit voll in der Verantwortung des Programmierers. Dieser muß bei der Dekodierung berücksichtigen, daß Teile von usercmd bei der Zeilenentgegennahme zu 'Tokens' komprimiert wurden.

Wer derartige Eingriffe vornimmt, sollte sich dessen bewußt sein, daß nachlässiges Arbeiten einen Systemabsturz nach sich ziehen kann.

Als Hilfestellung sei auf das vorzügliche umfangreiche Buch von James Parhour "MICROSOFT BASIC DECODED & other Mysteries for the TRS-80" (ISBN 0 - 936200 -01 -4) verwiesen (u.a. erhältlich bei DATA BECKER in Düsseldorf), das neben einem disassemblierten ROM-Listing Informationen über benutzbare ROM-Routinen und Beispiele zum Eingriff in den Interpreter enthält.

## VI AUTOSTART

Mit der Einführung von AUTOSTART wurde die Anwender von MACRO-BASIC die Möglichkeit geschaffen, unmittelbar nach der Initialisierung von MACRO-BASIC eine Anweisung ausführen lassen zu können, ohne daß vorher in den Befehlsmodus verzweigt wird.

Wie in der Dokumentation zu V.1.0 beschrieben, wird MACRO-BASIC als in Basic verpackter Maschinencode geliefert, der nicht editiert werden kann, ohne Initialisierungsfehler zur Folge zu haben.

Ab Version 2.2 kann davon abweichend

Zeile 250 (die letzte Programmtextzeile)

des MACRO-BASIC Code-Programms durch den Anwender geändert werden. Diese Zeile wird nach der Initialisierung interpretiert, als ob sie im Befehlsmodus eingegeben worden wäre.

Das so geänderte MACRO-BASIC kann vom Anwender als Basic Programm abgespeichert werden, bevor es mit RUN gestartet wurde.

Diese Erweiterung wurde im wesentlichen dazu entwickelt, den unmittelbaren Aufruf eines Folgeprogramms zu ermöglichen.

Dieses Folgeprogramm kann in einem System mit Disketten gestartet werden, indem Zeile 250 lautet:

```
250 RUN "filename</ext><<d>"
```

In einem Cassettensystem kann ein Folgeprogramm mit dem Namen n geladen werden durch:

```
250 CLOAD "n"
```

Ein derartiges Folgeprogramm bietet die Möglichkeit, über die Benutzung von BASE wie oben beschrieben Eingriffe in MACRO-BASIC vorzunehmen, um z.B. eine Extend-Routine zu laden und zu initialisieren oder um Maschinenprogramme zu laden, die mit !label aufgerufen werden sollen.

Eine andere sinnvolle Möglichkeit besteht darin, ein Unterprogrammpaket als Folgeprogramm einzuladen, das immer wieder benutzt werden soll.

Für Programmieren von Anwenderprogrammen unter MACRO-BASIC bietet sich die Möglichkeit (diskettenunterstützt) in Zeile 250 das Anwenderprogramm unter einem dummy-Filenamen aufzurufen und den File, der MACRO-BASIC mit der geänderten Zeile 250 enthält, mit dem Namen des Anwenderprogramms zu versehen, so daß für den Programm-Anwender MACRO-BASIC bis auf die Initialisierungsmeldung "transparent" bleibt.

Über die Möglichkeit des DOS AUTO Befehls lassen sich durch AUTOSTART nun auch unter MACRO-BASIC ohne Verwendung des DOS CHAINings vollständig selbstbootende Programmsysteme aufbauen.

Die Fehlerbehandlung bei der Abarbeitung von Zeile 250 nach Initialisierung von MACRO-BASIC entspricht der des Interpreters im Befehlsmodus.

Für die Änderungen von Zeile 250 gelten folgende Einschränkungen, deren Nichtbeachtung zu Initialisierungsfehlern führt:

Zeile 250 **muß** im MACRO-BASIC Code Programm stehen, darf also nicht gelöscht werden, ohne ersetzt zu werden.

Die Länge des Programmtexes in Zeile 250 darf in der durch den Interpreter nach Entgegennahme komprimierten Form 30 Zeichen nicht überschreiten. Basic Schlüsselworte werden beim Komprimieren auf maximal 2 Bytes komprimiert.

### VII TONAUSGABE

Ab Version 2.2 wurde MACRO-BASIC um einen Software-Tongenerator erweitert, der über die Cassetten Schnittstelle (Port OFFH) Rechteckschwingungen erzeugt. Die Cassetten-Buchse des TRS-80 kann über ein Kabel mit DIN-Steckern an den Tonband-Eingang eines Audio-Verstärkers geführt werden, so daß die Töne über Lautsprecher hörbar werden. Bei Verwendung eines Video-Genies mit eingebautem Cassetten Recorder muß der Cassetten-Anschluß nach außen geführt werden, um diese Möglichkeit nutzen zu können. Nehmen Sie dazu im Zweifelsfall Kontakt zu Ihrem Händler auf.

Der Befehl zur Ansprache des Tongenerators ermöglicht die Erzeugung von 36 definierten Tönen im Abstand von Halbtonschritten und Pausen. Es können 256 verschiedene Werte für die Tondauer verwendet werden. Im Unterschied zu anderen Software-Tongeneratoren ist die Dauer des Tons unabhängig von der gewählten Tonhöhe und der Ton nicht durch Knacken unterbrochen.

Die Syntax des Tongenerator Aufrufs lautet

```
<TTTT> <pgr :> . TON ( th , d ) <komm> < : pgr>
```

Die Tonhöhe wird über den Parameter th , die Tondauer über d angegeben. Beide Parameter müssen im Bereich von 0 und 255 liegen. Sie können als Konstante, numerische Variable oder numerische Ausdrücke übergeben werden. Der Variablentyp (Integer, Single oder Double) ist freigestellt. Aufgrund der zulässigen Wertebereiche sind jedoch Integer Variable aus Geschwindigkeitsgründen von Vorteil.

Die Zuordnung von th zu Halbtonschritten wurde wie folgt festgelegt:

	c	cis d	d	dis e	e	f	fis g	g	gis a	a	ais h	h
		des		es			ges		as		b	
tief	1	2	3	4	5	6	7	8	9	10	11	12
mittel	13	14	15	16	17	18	19	20	21	22	23	24
hoch	25	26	27	28	29	30	31	32	33	34	35	36

Die Töne sind aufgrund von Rundungsfehlern geringfügig verstimmt und etwas tiefer als üblich.

Wenn als *th* der Wert 0 übergeben wird, wird das als Pausenzeichen interpretiert. Die Länge der Pause wird wie die eines Tons durch *d* angegeben. Werte von 37 bis 255 werden ohne Fehlermeldung akzeptiert und ergeben undefinierte Tonhöhen.

Der Tongenerator interpretiert einen *d* Wert von 255 als ca. 1 sec. Kleinere Werte als kürzere Tondauern.

Großere Werte als 255 für *th* und *d* ergeben die Fehlermeldung ILLEGAL FUNCTION CALL.

Gerauschartige Effekte lassen sich mit Hilfe der RND-Funktion für Tonhöhe bei sehr kurzen Dauern erzeugen, z.B.:

```
FOR I=1 TO 300: .TON (RND*36).0: NEXT
```

Bei Verwendung von MACRO-BASIC unter Disk-Basic ist zu beachten, daß die Tonausgabe-Routine bei jedem Aufruf Interrupts abschaltet, die falls gewünscht vom Anwenderprogramm mit CMD "R" wieder aktiviert werden können.

Für Assembler-Programmierer, die die aus dem aufrufenden MACRO-BASIC Befehl decodierten Parameter für Tonhöhe und Tondauer anders auswerten möchten besteht die Möglichkeit, die Tonausgabe-Routine zu ändern. Sie liegt im Speicher bei

BASE+1 + 088CH bis BASE+1 + 08E8H.

Dieser Bereich kann durch ein Patch-Programm nach der Initialisierung von MACRO-BASIC (z.B. durch ein mit AUTOSTART aufgerufenes Programm) überschrieben werden.

Das Parameterformat bei Eintritt in die Routine ist:

C-Register: Tonhöhe                      D-Register: Tondauer.

Alle Register einschließlich des alternativen Registersatzes können ohne Sicherung verwendet werden. Eine geänderte Tonausgabe-Routine wird mit RET abgeschlossen.

## VIII SONSTIGES

MACRO-BASIC wurde sorgfältig getestet, bevor es auf den Markt gebracht wurde. Sollten dennoch Fehler auftreten, wenden Sie sich bitte unmittelbar an den Autor. Er wird versuchen, den Fehler möglichst schnell zu beheben. Eine weitergehende Haftung kann nicht übernommen werden.

Anwenderprogrammierer, die die MACRO-BASIC-Möglichkeiten zur nationalen Programmenstellung nutzen wollen, können einen entsprechenden Lizenzvertrag zu günstigen Staffelpreisen bei dem Autor abschließen.

Klaus Damm, Rehweg 8, D-5354 Weilerswist 2, Tel. (02254) 7086

TRS-80, Newdos 40 und Newdos 80 sind eingetragene Warenzeichen der Tandy-Corporation bzw. von Apparat Inc.