M A S K E N G E N E R A T O R 2.0

FÜr SCHNEIDER CPC 464

Herausgegeben von
vortex Computersysteme Vertriebs GmbH

(C)1985 R. Kindermann

Benutzerhandbuch

Vervielfältigung und Weitergabe - auch auszugsweise- dieses Handbuches und der dazugehörigen Programm-Diskette bedürfen der vorherigen schriftlichen Zustimmung des Autors.

vortex übernimmt keine Garantie, was die Eignung der Programme für bestimmte Anwendungen betrifft.

Alle Änderungen vorbehalten.

Neuenstadt im Februar 1986.

Alle Vertriebsrechte liegen bei vortex Computersysteme Vertriebs GmbH, Klingenberg 13, 7106 Neuenstadt 5 .

I N H A L T S V E R Z E I C H N I S

1. Einleitung

Auf der Diskette befinden sich mehrere Programme:

```
ADRESSEN.BAS
ADRESSEN.MSK
AMSDOS  .BAS
DIALOG  .BAS
DIALOG-R.BAS
INITX   .BAS
MASKGEN .BAS
MASKGEN .OVR
VDOS1   .BAS
VDOS2   .BAS
XBASIC  .RSX
XBASIC-R.RSX
XMASK   .RSX
```

Davon werden nicht alle zum Erstellen und Verwenden einer Bildschirmmaske benötigt. Die genaue Beschreibung der einzelnen Programme findet sich in den entsprechenden Kapiteln.
WICHTIG: Falls der Maskengenerator mit einem anderen Diskettenbetriebssystem als VDOS 1.0 benutzt werden soll, muß das Programmpaket erst entsprechend installiert werden. Sonst ist es möglich, daß der Rechner 'abstürzt'. Für diese Installation sind die drei Programme AMSDOS.BAS, VDOS1.BAS und VDOS2.BAS auf der Diskette. Sie sind in Kapitel 7 beschrieben.
Der Maskengenerator erzeugt aus dem erstellten Bild einen komprimierten Code, der dann abgespeichert wird. Außerdem sind in der entstehenden Datei sämtliche Informationen über die erlaubten Eingabefelder enthalten. Durch den Aufruf der Dialogfunktion 'Maske laden' wird das Bild wieder geladen und die Eingabefelder sind definiert.
Das Programm ADRESSEN ist ein Beispielprogramm für die Verwendung des Maskengenerators und Dialogs. Es ist ausführlich kommentiert und ungeschützt. Auch die darin verwendetete Maske ADRESSEN.MSK befindet sich auf der Diskette und kann mit MASKGEN geladen und bearbeitet werden.
Mit INITX läßt sich die Befehlserweiterung unabhängig vom Maskengenerator oder Dialog initialisieren.
Die Datei XBASIC.RSX enthält die Befehlserweiterung selbst.
Das Programm DIALOG ist auch ungeschützt und läßt sich einfach mit MERGE, CHAIN oder CHAIN MERGE in bestehende Programme einfügen.

ACHTUNG: Sämtliche Programme auf der Diskette unterliegen dem Copyright von Robert Kindermann, Zweigstr. 10, 8039 Puchheim-Bhf.
Dennoch darf das Programm DIALOG-R.BAS von einem rechtmäßigen Benutzer des Programmpakets 'Maskengenerator 2.0' in eigene Programme eingebunden und verkauft werden. Voraussetzung dafür ist, daß das entstehende Programm geschützt abgespeichert wird (durch Angabe von SAVE <name>,p). Die für DIALOG-R notwendige Datei

XBASIC-R.RSX darf dann selbstverständlich auch kopiert werden.

Es wird empfohlen diese Anleitung gründlich zu studieren, um mit den Möglichkeiten des Maskengenerators und der vorhandenen Hilfsprogramme vertraut zu werden. Besonders sei hier auf Kapitel 6 (Beispielprogramm ADRESSEN) und das Programm ADRESSEN selbst hingewiesen. Auch ist es empfehlenswert die Maske ADRESSEN in den Maskengenerator zu laden und dort eingehend anhand dieser Maske alle Befehle des Maskengenerators zu testen.

2. Maskengenerator 2.0

Das Programm wird mit RUN "MASKGEN" gestartet. Dabei wird die verschiebliche Befehlserweiterung XMASK.RSX unterhalb von HIMEM in den Speicher geladen. Falls der Speicher durch einen vorherigen MEMORY Befehl zu klein sein sollte, stoppt der Maskengenerator und das System muß vor einem Neustart zurückgesetzt werden.
Ansonsten meldet sich das Programm mit einem leeren Bildschirm und der Systemzeile. Der Bilschirm ist in Anwenderbereich und Systembereich aufgeteilt. Die Zeilen 1-22 beinhalten den Anwenderbereich, die letzten drei Zeilen sind der Systembereich. In den Systemzeilen stehen wichtige Informationen über den Zustand des Programms (z.B. Cursor, Copycursor, oder Art des eingegebenen Befehls).

2.1 Editierbefehle

Zum Erstellen einer Bildschirmmaske sind verschiedene Editierbefehle verfügbar (Tabelle 1). Die meisten sind an die Grundfunktionen des CPC464 gehälten, wenn auch einige Einschränkungen aufgehoben sind. So läßt sich der Cursor immer beliebig im gesamten Anwenderbereich positionieren. Die Bewegung des Cursor geschieht dabei durch die normalen Cursortasten. Die Position des Cursors wird in der Systemzeile angezeigt. Im folgenden sollen für die Cursortasten die Abkürzungen <oben>, <unten>, <rechts> und <links> verwendet werden.
Im Gegensatz zum normalen Basicbefehl EDIT wird ein Zeichen nicht in den Text eingefügt, sondern es wird der alte Text überschrieben. Dies ist besonders hilfreich beim Erstellen von Masken in Spaltenschreibweise (Tabellen o.ä.). Das Einfügen eines Zeichens geschieht mit CTRL-E. Dabei wird der Text nach dem Cursor um ein Zeichen nach rechts verschoben, an der Stelle des Cursors wird ein Leerzeichen eingefügt. Jetzt kann durch Überschreiben des Leerzeichens ein beliebiges Zeichen in den Text gebracht werden.
Mit dem Copycursor lassen sich einfach Texte kopieren. Dabei sind keine Einschränkungen zu beachten. Mit SHIFT <oben>, <unten>, <rechts> und <links> läßt sich der Copycursor bewegen. Die COPY-

Taste kopiert dann ein Zeichen vom Copycursor an die Position des Cursors. Danach werden Cursor und Copycursor um ein Zeichen nach rechts verschoben. Dadurch können beliebig lange Texte mit der Wiederholfunktion der Tastatur kopiert werden.
Der Copycursor bleibt auch nach ENTER noch auf dem Bildschirm. Die Abschaltung geschieht durch CTRL-D. Außerdem wird er bei Aufruf eines externen Befehls (siehe 2.2 Sonstige Befehle) abgeschaltet.
Die ENTER-Taste stellt den Cursor nur auf den Anfang der nächsten Zeile, veranlasst sonst aber keine Aktion.
Zur einfachen und schnellen Erstellung von Tabellen und anderen Masken in Spaltenschreibweise ist eine Tabulatorfunktion realisiert worden. Durch CTRL-TAB kann ein Tabulatorstop gesetzt bzw. gelöscht werden. Mit TAB wird der Cursor auf den nächsten Stop gestellt. In der Systemzeile werden alle Tabulatorstops angezeigt.
Mit CTRL-<links> wird der Cursor an den Anfang, mit CTRL-<rechts> an das Ende der Zeile gestellt.
Ganze Zeilen können mit CTRL-<oben> bzw. CTRL-<unten> gelöscht bzw. eingefügt werden. Beim Löschen ist die neu entstehende Zeile leer. Durch das Einfügen einer Zeile fällt die unterste Zeile des Anwenderbereichs weg.
Einzelne Zeichen lassen sich wie im normalen EDIT-Befehl löschen. Mit CLR wird das Zeichen unter dem Cursor gelöscht, und der restliche Text wird um ein Zeichen nach links verschoben. Genauso verhält es sich bei DEL, nur wird das Zeichen vor dem Cursor gelöscht. Dabei gibt es eine Einschränkung zu beachten. Die Tasten CLR, DEL und auch CTRL-E beziehen sich immer nur auf eine Zeile. Es ist damit nicht möglich z.B. zwei Zeilen zu einer einzigen zusammenzufassen oder eine neue Zeile einzufügen. Dies muß mit dem Copycursor und CTRL-<oben> oder CTRL-<unten> gemacht werden.

Folgende Funktionen stimmen mit den Steuerzeichen des CPC464 überein:

Mit CTRL-L wird der Bildschirm gelöscht (Anwenderbereich).
CTRL-Q löscht eine Zeile von Beginn bis zur Cursorposition. Das Zeichen unter dem Cursor wird auch gelöscht.
CTRL-R löscht eine Zeile von der Cusorposition bis zum Ende. Das Zeichen unter dem Cursor wird auch gelöscht.
CTRL-S löscht den Bildschirm bis zur Cursorposition.
CTRL-T löscht den Bildschirm ab der Cursorposition.
CTRL-X vertauscht Hintergrund- und Schreibfarbe.
CTRL-^ stellt den Cursor in die obere linke Ecke (Cursor Home).
Weitergehende (externe) Befehle werden durch CTRL-C eingeleitet.

Der Editiermodus wird verlassen, in der Systemzeile erscheint 'Eingabe' und das Programm wartet auf einen Kennbuchstaben für den Befehl. Falls der Kennbuchstabe nicht erkannt wird, erscheint die Fehlermeldung 'Falsche Eingabe' und es wird in den Editiermodus zurückgekehrt.
Die einzelnen Kennbuchstaben sind im nächsten Kapitel beschrieben.

Editierbefehle:
==========

Taste	Wirkung	Bemerkung
<rechts>, <links>, <oben>, <unten>	bewegen Cursor	gesamter Anwenderbereich
SHIFT-<rechts>, <links>, <oben>, <unten>	bewegen Copycursor	gesamter Anwenderbereich
COPY	Zeichen kopieren	
CTRL-D	Copycursor löschen	
CTRL-<oben>	Zeile löschen	
CTRL-<unten>	Zeile einfügen	
CLR, DEL	Zeichen löschen	CLR löscht Zeichen unter Cursor, DEL vor Cursor
CTRL-E	Zeichen einfügen	
CTRL-<links>	Cursor an Anfang der Zeile	
CTRL-<rechts>	Cursor an Ende der Zeile	
TAB	Tabulator	
CTRL-TAB	Tabulator setzen bzw. löschen	
ENTER	Cursor an Anfang der nächsten Zeile	
CTRL-L	Bildschirm löschen	
CTRL-Q	Zeile bis Cursor löschen	
CTRL-R	Zeile ab Cursor löschen	
CTRL-S	Bildschirm bis Cursor löschen	
CTRL-T	Bildschirm ab Cursor löschen	
CTRL-X	Schreibfarben vertauschen	
CTRL-^	Cursor in die linke obere Ecke	
CTRL-C	Externen Befehl aufrufen	

Tabelle 1: Zusammenfassung der Editierbefehle

2.2 Sonstige Anweisungen

Nachdem der Editiermodus mit CTRL-C verlassen worden ist, kann im Befehlsmodus ein Kennbuchstabe für den jeweiligen Befehl eingegeben werden. Bei der Eingabe des Kennbuchstabens wird zwischen Groß- und Kleinschrift unterschieden.

l - Maske laden

Mit l kann eine Maske geladen werden. Der Dateiname, der anzugeben ist, kann eine Laufwerkskennung besitzen, die mit einem Doppelpunkt vom Namen getrennt ist (A:<name> oder B:<name>). Falls anstelle des Dateinamens nichts (leeres Feld) eingegeben wird, erfolgt ein Rücksprung in den Editiermodus, ohne eine Maske zu laden.

s - Maske speichern

Mit s kann die gerade auf dem Bildschirm befindliche Maske abgespeichert werden. Der Dateiname, der anzugeben ist, kann eine Laufwerkskennung besitzen, die mit einem Doppelpunkt vom Namen getrennt ist (A:<name> oder B:<name>). Falls anstelle des Dateinamens nichts (leeres Feld) eingegeben wird, erfolgt ein Rücksprung in den Editiermodus, ohne die Maske zu speichern.

i - Inhaltsverzeichnis

Mit i kann das Inhaltsverzeichnis aller Masken auf einem Laufwerk erstellt werden. Dabei sollte keine Maske mehr in Bearbeitung sein, da der Bildschirm gelöscht wird. Nach Auswahl des Laufwerks wird das Inhaltsverzeichnis und der verbleibende Speicherplatz auf der Diskette auf den Bildschirm gebracht. Dies bleibt solange stehen, bis eine Taste gedrückt wird.

f - Fenster Definition

Durch den Aufruf von f wird ein Eingabefenster definiert. Dazu erscheint in der Systemzeile folgende Definitionsleiste:

```
NR: 0  SP: 1  ZE: 1  LE: 1  PL: 0  CHAR:B  CTRL:  E    XCHAR:
```

Außerdem wird das Fenster in seiner momentanen Position und Länge auf dem Bildschirm invertiert angezeigt. Bei Aufruf dieser Funktion wird automatisch das nächste freie Fenster in die Definitionsleiste geladen. Die einzelnen Felder haben folgende Bedeutung:

NR ist die jeweilige Nummer des Fensters, beginnend mit Nummer Null. Die maximale Fensternummer ist 255.

SP,ZE legen die Spalte und die Zeile fest, in der das Fenster beginnen soll. Diese Werte entsprechen der Cursorposition beim Aufruf von f. Dadurch läßt sich sehr einfach der Beginn eines Fensters festlegen. Die maximale Spalte ist 80, die maximale Zeile ist 22.

LE legt die Länge eines Fenster fest. Die maximale Länge ist 80.

PL legt die Plausibilitätskontrolle fest. Falls in PL null steht, wird keine Kontrolle durchgeführt, ansonsten wird in Abhängigkeit von PL ein Unterprogramm aufgerufen. Damit lassen sich leicht Sonderfunktionen für verschiedene Felder realisieren (z.B. Kürzel, siehe Kapitel 5).

CHAR legt fest, welche Zeichen als Eingabe zulässig sind. Dabei kann zwischen verschiedenen Gruppen von Zeichen ausgewählt werden. Die Gruppen werden durch folgende Buchstaben abgekürzt:

B - Nur Buchstaben erlaubt
G - Kleinbuchstaben werden in Großbuchstaben verwandelt
Z - Nur Ziffern erlaubt
A - Alle Zeichen erlaubt
S - Leerzeichen erlaubt
U - Eingabefeld unterstreichen

Diese Gruppen können in beliebiger Reihenfolge und Kombination gewählt werden. Die Reihenfolge wird allerdings der Ordnung BGZASU angepaßt. Falls keine dieser Gruppen erlaubt ist, werden nur Sonderzeichen als Eingabe akzeptiert (siehe XCHAR). Falls auch keine Sonderzeichen definiert sind, ist es unmöglich eine Eingabe in diesem Feld zu machen!

CTRL legt die Tasten fest, mit denen ein Verlassen des Eingabefeldes möglich ist. Dabei bedeutet:

O - Cursor nach oben
U - Cursor nach unten
E - Enter
C - CLR
B - ESC

Achtung! Falls keine Aussprungtaste erlaubt ist, kann das Eingabefeld nicht verlassen werden!

XCHAR legt die Sonderzeichen fest, die außer den in CHAR definierten Zeichen erlaubt sind. Falls in CHAR kein Leerzeichen erlaubt ist, können zehn Sonderzeichen definiert werden, sonst nur neun.

Es sei nochmals auf Kapitel 6 hingewiesen, wo anhand eines Beispiels die Möglichkeiten der Fenster Definition besser erklärt werden können.

F - Fenster anzeigen

Mit F werden sämtliche Fenster auf dem Bildschirm invertiert.

d - Fenster ausdrucken

Mit d werden die Fensterdefinitionen auf den Drucker ausgegeben (siehe Beispielausdruck im Anhang).

L - Fenster löschen

Mit L wird eine Fensterdefinition gelöscht. Die darauffolgenden Fenster werden um eine Position nach unten verschoben, d.h. die Fensternummer ändern sich.

E - Fenster einfügen

Mit E kann ein Fenster eingefügt werden. Die Fenster verschieben sich ab der Position, an der eingefügt wird, um eins nach oben. Die Fensternummern erhöhen sich also um eins.

k - Fenster kopieren

Mit k können Fenster, die in einer Zeile liegen, auf n neue Zeilen kopiert werden. Dazu werden die Grenzen der Zeile (Fensternummern) und die Anzahl der Zeilen, die erstellt werden sollen, eingegeben. In den enstehenden Zeilen ändert sich nur die Zeilennummer in der Fensterdefinition, sonst bleiben alle Parameter der Orginalzeile erhalten.

t - Tastatur belegen

Mit t können einzelne Tasten umbelegt werden (ähnlich KEY DEF). Dazu muß die Tastennummer, ihr Normalwert und ihr SHIFT-Wert eingegeben werden. Der CTRL-Wert der Taste lässt sich nicht ändern. Die Taste wird wiederholbar.

x - Erweiterungsstring belegen

Mit x kann einer von 32 Erweiterungsstrings definiert werden (ähnlich KEY). Dazu muß zuerst das Token für den Erweiterungsstring (128-159) und dann der String selbst eingegeben werden.

X - Verdeckte Zeichen einschalten

Da Zeichen im Bereich von 128 bis 159 als Erweiterungsstring interpretiert werden, bietet X die Möglichkeit, auf einen Erweiterungsstring das entsprechende Zeichen zu legen. Damit ist das Zeichen dann verfügbar, da in einem Erweiterungsstring keine Erweiterungszeichen berücksichtigt werden.

q - Befehlsmodus verlassen

Mit q wird in den Editiermodus zurückgekehrt, ohne daß ein Befehl ausgeführt wird.

Q - Maskengenerator verlassen

Mit Q wird der Maskengenerator beendet.

Externe Befehle:
==========

Kennbuchstabe	Befehl
l	Maske laden
s	Maske speichern
i	Inhaltsverzeichnis aller Masken
f	Fenster Definition
F	Fenster anzeigen (invertieren)
L	Fenster löschen
E	Fenster einfügen
d	Fensterdefinitionen ausdrucken
k	Fensterzeile kopieren
t	Tastatur belegen
x	Erweiterungsstring definieren
X	Verdeckte Zeichen ermöglichen
q	Rückkehr in den Editiermodus (Befehlsabbruch)
Q	Verlassen des Maskengenerators

Tabelle 2: Zusammenfassung der externen Befehle

3. DIALOG 2.1

Das Unterprogrammpaket DIALOG 2.1 dient zur Verwendung der im Maskengenerator definierten Bildschirmmasken und Eingabefelder. Es stehen fünf Funktionen für den Anwender zur Verfügung (Tabelle 3):

1 - DIALOG initialisieren
2 - Maske laden
3 - DIALOG aufrufen
4 - Eingabefelder löschen
5 - Eingabe in Systemzeilen

Es gibt für alle Funktionen nur eine Einsprungstelle auf Zeile 65000. Die Auswahl der entsprechenden Funktion geschieht durch die Variable d.fn%. Sie muß mit der entsprechenden Funktionsnummer belegt werden. Nach Belegung aller Übergabeparameter wird die Funktion mit GOSUB 65000 aufgerufen. Dadurch ist es möglich mit allen Erweiterungen und Änderungen kompatibel zu bleiben. Im Anwenderprogramm müssen keine umständlichen Zeilenänderungen vorgenommen werden. Der gesamte DIALOG verwendet nur Variablennamen, die mit 'd.' beginnen. Das Anwenderprogramm sollte solche Namen vermeiden, um das einwandfreie Zusammenspiel zwischen Anwenderprogramm und DIALOG sicherzustellen.
Für jedes Eingabefeld werden zwei Strings bereitgestellt. In d.win$(x) werden die Fensterdefinitionen (Format siehe SCREEN INPUT, Neue Basicbefehle) gespeichert, die zugehörigen Daten in d.dat$(x). Dabei hat d.dat$(x) genau die Länge, die durch d.win$(x) festgelegt worden ist.
Im folgenden sind die Funktionen ausführlich beschrieben.

3.1 Dialogfunktionen

DIALOG initialisieren

Funktionsnummer : 1
Eingangsparameter: keine
Ausgangsparameter: keine
Beschreibung : Diese Funktion muß vor Aufruf aller anderen Funktionen angewählt werden. Es werden sämtliche Variable des Dialogs mit ihren Standardwerten belegt. Außerdem wird die benötigte Befehlserweiterung geladen und aufgerufen. Dem für Diskettenoperationen benötigten 2k-Puffer wird ein fester Platz zugewiesen. Das erspart lange Wartezeiten durch eine GARBAGE COLLECTION vor jedem Laufwerkszugriff. Danach ist es leider nicht mehr möglich mit SYMBOL AFTER den Speicherbereich

für selbst definierte Zeichen zu vergrößern. Deswegen gibt es einen Aussprung vor Belegung des 2k-Puffers. Es wird die Zeile 64000 als Unterprogramm aufgerufen. Darin kann der Anwender seine eigenen Varialen und andere Größen initalisieren. Dort kann auch mit SYMBOL AFTER mehr Speicherplatz für selbstdefinierte Zeichen reserviert werden.
Dabei sollte jedoch beachtet werden, daß die Befehlserweiterung die Zeichen von 240-246 mit den deutschen Umlauten belegt. Nach einem SYMBOL AFTER Befehl werden aber diese Zeichen auf ihre Standardwerte zurückgesetzt, d.h. sie müssen neu belegt werden. Eine Tabelle mit diesen Werten befindet sich im Anhang (Tabelle 6).

Hinweise : Der Bildschirmmodus wird auf MODE 2 gestellt. Diese Einstellung darf nicht verändert werden. Alle Fenster haben vor Aufruf der Anwenderinitialisierung volle Bildschirmgröße. Danach bleiben die Fenster #1 bis #7 unverändert, Fenster #0 wird nochmals auf volle Bildschirmgröße eingestellt. Diese Einstellung darf nicht verändert werden.
Die selbstdefinierbaren Zeichen beginnen standardmäßig ab Zeichen 240. Die Zeichen 240-246 sind für die deutschen Umlaute belegt. Sie dürfen nicht geändert werden.

Maske laden

Funktionsnummer : 2
Eingangsparameter: d.name$
Ausgangsparameter: d.oben%, d.dat$(d.oben%-1), d.win$(d.oben%-1)
Beschreibung : Mit dieser Funktion kann eine im Maskengenerator erstellte Maske wieder geladen werden. Dazu muß in d.name$ der Name der Maske übergeben werden. Die Kennung .MSK, die der Maskengenerator erzeugt, wird auch vom DIALOG automatisch erzeugt. Der Maskenname kann eine Laufwerkskennung haben (A:<name> oder B:<name>). Die Variable ist vom DIALOG auf 10 Leerzeichen vorbelegt (8 Zeichen für Namen, 2 Zeichen für Laufwerk). Dadurch läßt sich auch bei häufigem Aufruf dieser Funktion ein Vollschreiben des Speichers vermeiden (siehe Zuweisung durch MID$, Hinweise). Falls ein unkorrekter Dateiname übergeben worden ist, oder ein sonstiger Diskettenfehler auftrat, wird die Operation abgebrochen und der Fehleransprung aufgerufen (siehe Anwenderausprünge).
Ansonsten wird die Maske auf den Bildschirm

gebracht und d.dat$(x) und d.win$(x) mit den entsprechenden Werten belegt. Der Variablen d.oben% wird die Anzahl der Fenster zugewiesen. Da die Fenster von null an durchnummertiert sind, sind auch d.dat$ und d.win$ nur von null bis d.oben%-1 definiert (für d.oben%>0!).

Hinweise: Der Bildschirm wird gelöscht und auf MODE 2 gestellt. Alle Bildschirmfenster werden auf volle Bildschirmgröße gestellt. Desvegen existiert der Anwenderaussprung in Zeile 64250. Dort können alle vom Anwender benötigten Fenster reinitialisiert werden. Fenster #0 muß volle Bildschirmgröße behalten. Auf die Variable d.win$(x) darf nur lesend zugegriffen werden. Die Länge der Variablen d.dat$ und d.dat$(x) ist fest eingestellt und darf nicht geändert werden. Dies wird durch die Zuweisung über MID$ erreicht:

MID$(d.dat$,1)=<neuer String>

Dabei wird d.dat$ nicht neu angelegt, sondern nur überschrieben. Seine Länge ändert sich dabei nicht. Von d.dat$ und <neuer String> werden nur Zeichen bis zum Ende des Kürzeren von beiden berücksichtigt, d.h. falls <neuer String> länger ist, werden die restlichen Zeichen nicht gewertet, falls er kürzer ist, werden die restlichen Zeichen von d.dat$ nicht geändert. Mit MID$(d.dat$,1)=d.sp255$ kann d.dat$ auf Leerzeichen gelöscht werden. d.sp255$ ist ein String mit 255 Leerzeichen. Analog geschieht die Zuweisung über MID$ auf ein Feldelement d.dat$(x).
Es ist nicht zulässig den Wert von d.dat$ oder d.dat$(x) auf andere Art zu ändern! Dies führt zu einem Fehler in der DIALOG Funktion 3!
Der Wert von d.oben% darf nicht geändert werden.

Dialog aufrufen

Funktionsnummer : 3
Eingangsparameter: d.min%, d.max%, d.ein%
Ausgangsparameter: d.ex%, d.aus%
Beschreibung : Diese Funktion fasst mehrere Eingabefelder zu einer Eingabemaske zusammen. Dadurch wird die Eingabe und Bearbeitung der Fenster einer Maske durch den Anwender ermöglicht. Jede Eingabe in ein Fenster ist in dem entsprechenden d.dat$(x) abgespeichert. Fehlende Stellen werden mit Leerzeichen aufgefüllt. Die

verschiedenen Editiermöglichkeiten innerhalb des Eingabefeldes sind in Abschnitt 4.1 Neue Basicbefehle bei SCREEN INPUT beschrieben. Die Nummer des ersten Eingabefeldes wird in d.ein% übergeben. Dort wird dann eine Eingabe erwartet. Danach führt der DIALOG je nach Aussprungtaste unterschiedliche Aktionen durch (nur falls die Aussprungtaste auch als solche definiert wurde):

<oben> : Die Eingabe schaltet auf ein Feld vor dem jetzigen Eingabefeld um. Dabei wird d.ein% um eins erniedrigt. Falls d.min% unterschritten wird, erfolgt ein Rücksprung zum aufrufenden Programm.

<unten>: Die Eingabe schaltet auf ein Feld nach dem jetzigen Eingabefeld um. Dabei wird d.ein% um eins erhöht. Falls d.max% überschritten wird, erfolgt ein Rückspung zum aufrufenden Programm.

ENTER : siehe <unten>.

CLR : Alle vorgenommenen Änderungen in diesem Eingabefeld werden rückgänig gemacht. Danach wird nochmals in dieses Eingabefeld gesprungen. d.ein% bleibt unverändert.

ESC : Es wird keine Aktion durchgeführt. Falls der Benutzer das Programm nicht abbricht (oder eine ON BREAK GOSUB Routine durchgeführt wird), springt der DIALOG nochmals ins gleiche Eingabefeld. Der Cursor steht dann allerdings am Anfang des Feldes.

Achtung: Falls ESC nicht als Aussprungtaste zugelassen ist, kommt es zu folgendem Verhalten des DIALOGS. Während der Eingabe mit SCREEN INPUT wird das Drücken der ESC-Taste vom Betriebssystem lediglich registriert, es kann aber nicht die entsprechende Aktion ausgeführt werden (Anhalten des laufenden Programms, Cursor darstellen und auf zweites ESC warten). Dies wird erst nach Verlassen des Eingabefelds durchgeführt. Falls ESC während der Eingabe gedrückt wurde, wird der DIALOG vom Betriebssystem nach dem SCREEN INPUT Befehl angehalten, der Benutzer muß dann ESC oder eine andere Taste drücken. Dies kann manchmal verwirrend sein, da das Warten auf

die zweite ESC-Taste und der erste Druck auf die ESC-Taste zeitlich sehr weit auseinander liegen können.

Hinweise : Durch den Aufruf dieser Funktion wird kein Speicher für Stringverarbeitung belegt. Auch nach wiederholtem Aufruf ist keine GARBAGE COLLECTION erforderlich. Der für d.dat$ und d.dat$(x) benötigte Speicherplatz wurde bereits beim Masken laden reserviert. Im übrigen gelten die an die Zuweisung von d.dat$ und d.dat$(x) gestellten Forderungen (siehe Dialogfunktion 2, Hinweise).

Eingabefelder löschen

Funktionsnummer : 4
Eingangsparameter: keine
Ausgangsparameter: keine
Beschreibung : Mit dieser Funktion werden alle d.dat$(x) auf Leerzeichen gesetzt. Außerdem werden am Bildschirm sämtliche Fenster gelöscht.
Hinweise : Diese Routine erfüllt alle an die Zuweisung von d.dat$ und d.dat$(x) gestellten Forderungen (siehe Dialogfunktion 2, Hinweise).

Eingabe in Systemzeilen

Funktionsnummer : 5
Eingangsparameter: d.sys1$, d.sys2$, d.len%, d.char%, d.ctrl%, d.xchar$
Ausgangsparameter: d.dat$
Beschreibung : Diese Funktion ermöglicht die Eingabe in Systemzeilen (Zeile 24 und 25). Dabei wird in d.sys1$ bzw. d.sys2$ der Text, der in Zeile 24 bzw. 25 erscheinen soll, übergeben. Das Eingabefenster beginnt automatisch nach d.sys2$, d.h. die Länge von d.sys2$ ist ausschlaggebend. In d.len% wird die Länge des Eingabefeldes übergeben. Die beiden Variablen d.char% und d.ctrl% sind bitsignifikant. Sie definieren die erlaubten Zeichen und Ausprungtasten. Das genaue Format ist in Abschnitt 4.1 bei SCREEN INPUT beschrieben. In d.xchar$ werden Sonderzeichen, die außer den in d.char% definierten Zeichen zulässig sind, übergeben. Die Eingabe wird in d.dat$ an das aufrufende Programm übergeben.
Hinweise : Das Fenster #0 wird auf volle Bildschirmgröße gestellt. Die Cursorposition wird nicht verändert. Von d.sys1$ und d.sys2$ werden nur die

ersten achtzig Zeichen ausgegeben. Von d.xchar$ werden nur die ersten zehn Zeichen berücksichtigt. d.xchar$ wird bei der Initialisierung des DIALOGs mit zehn Leerzeichen belegt. Es wird nicht überprüft, ob das Fenster in die Bildschirmzeile passt. Außerdem wird nicht geprüft, ob in d.len% eine sinnvolle Länge übergeben wird (d.len%<256). Die Systemzeilen sind nach Verlassen der Funktion gelöscht.

DIALOGfunktionen

d.fn%	Funktion	Bemerkung
1	DIALOG initialisieren	Sämtliche DIALOG-Variable werden mit ihren Standardwerten belegt.
2	Maske laden	Eine Maske anzeigen und alle Fensterstrings laden.
3	Dialog aufrufen	Eingabe in definierte Felder.
4	Eingabefelder löschen	Sämtliche Fenster der Maske löschen.
5	Eingabe in Systemzeilen	Eingabe in den Zeilen 24 und 25 (Systemzeilen).

Tabelle 3: Dialogfunktionen

3.2 Anwenderaussprünge

Die drei wichtigsten Anwenderaussprünge wurden schon bei den DIALOG-Funktionen erläutert. Zur Vollständigkeit werden sie hier nochmals erwähnt.

Zeile 64000: Initialisierung aller Anwendervariablen und -größen, im Speziellen eine SYMBOL AFTER Anweisung.

Zeile 64250: Reinitialisierung aller Fenster nach dem Laden einer Maske.

Zeile 64500: Anwenderfunktionen, werden in Abhänigkeit von d.pl% aufgerufen. Dadurch ist es dem Anwender des DIALOGs leicht möglich, eigene Funktionen in das Unterprogrammpaket DIALOG 2.1 einzubinden, die dann in Abhänigkeit von der Größe PL des Eingabefeldes ausgeführt werden.

Es existiert ein weiterer Aussprung auf Zeile 64750. Diese Zeile wird angesprungen, wenn innerhalb des DIALOGs ein Fehler aufgetreten ist. Dieser Fehler kann bei richtiger Übergabe aller Parameter an den DIALOG nur ein Diskettenfehler sein. Die entsprechende Fehlernummer wird in d.derr% übergeben. Die genaue Beschreibung der Diskettenfehlermeldungen befindet sich im Abschnitt 4.2. Die normale, vom Basic erzeugte Fehlermeldung wird in d.err%, die Zeile, in der der Fehler auftrat, in d.line% übergeben. Dieser Aussprung wird über GOTO angesprungen. Dadurch ist ein fehlerfreier Ablauf der Funktion RESUME gewährleistet (kein Stacküberlauf durch GOSUB ohne RETURN).

3.3 DIALOG-R 2.1

Das Unterprogrammpaket DIALOG-R 2.1 besitzt diesselben Schnittstellen und Parameter wie DIALOG 2.1. Lediglich überflüssige Kommentare und Leerzeichen wurden entfernt. Außerdem wird als Befehlserweiterung nicht XBASIC sondern XBASIC-R verwendet. Der Unterschied besteht darin, daß in XBASIC-R die Routinen fehlen, die zur Umwandlung der neuen Schlüsselwörter (SCREEN, GET und DERR) in Token und deren Rückwandlung in Texte gebraucht werden. Das hat zur Folge, daß bei einer Eingabe unter XBASIC-R die neuen Schlüsselwörter nicht mehr erkannt werden. Außerdem ist es unmöglich, Zeilen zu LISTen, in denen die neuen Schlüsselwörter verwendet wurden. Da beides zur Laufzeit eines Programms nicht nötig ist, ist diese Erweiterung nach Entwicklung und Austesten eines Programms (mit XBASIC) völlig ausreichend. Dabei ist der teilweise Schutz vor LIST sicher auch für den Anwender interessant.
Dieses Unterprogrammpaket darf in Anwenderprogramme eingebunden und mit verkauft werden, falls das resultierende Programm geschützt gespeichert worden ist und auch nur so verkauft wird. Dazu kann auch die Datei XBASIC-R.RSX mitkopiert werden (siehe Einleitung).

4. XBASIC 2.2

Um einen Maskengenerator sinnvoll zu verwirklichen, wurden auf dem CPC464 einige Befehlserweiterungen implementiert (Tabelle 5). Diese Erweiterung ist nicht als sogenannte RSX im System vorhanden, sondern verwendet die BASIC-Indirections um die neuen Schlüsselwörter einzubinden. Es werden neue Token dafür vergeben. Außer den neuen Befehlen wurden einige Unzulänglichkeiten des Betriebssystems umgangen. Es ist unter XBASIC möglich, Strings direkt in externen Befehlen zu verwenden, d.h. !dir,"*.com" gibt keine Fehlermeldung (wie CPC 664). Außerdem ist es möglich, die Diskettenfehler mit ON ERROR GOTO abzufangen. Dazu wurden zwei neue Fehlermeldungen 'File not open' und 'Broken' eingebunden. Die Diskettenfehlermeldungen sind im übernächsten Abschnitt beschrieben.

4.1 Neue Basicbefehle

Es wurden zwei neue Schlüsselwörter als Befehle eingeführt: SCREEN und GET. Damit stehen fünf neue Befehle zur Verfügung:

```
SCREEN <dateiname>
SCREEN INPUT <fensterstr>
SCREEN LOCATE <fensterstr>
SCREEN CLEAR <fensterstr>
GET <str>,<länge>(,<ende>)
```

Sämtliche SCREEN-Befehle verwenden das Fenster #0. Es sollte auf volle Bildschirmgröße gestellt sein.
Die Befehlserweiterung kann unabhängig vom Maskengenerator und DIALOG mit RUN "INITX" eingeschalten werden.

SCREEN <dateiname>

Dieser Befehl eröffnet die Datei <dateiname>. Dabei muß es sich um eine Maskendatei handeln. Danach wird die Maske geladen und auf dem Bildschirm angezeigt. Das Eingabegerät ist nach diesem Befehl wie durch OPENIN <dateiname> eröffnet, da im Anschluß an die Bildschirmmaske die Informationen für die definierten Fenster kommen (für den Fall, daß die Maske mit dem Maskengenerator erstellt worden ist). Das erste Byte gibt die Anzahl der Fenster an. Danach kommen die einzelnen Fensterstrings, wobei das erste Byte vor jedem String dessen genaue Länge angibt.

SCREEN INPUT <fensterstr>

Dieser neue Befehl zur Eingabe stellt umfangreiche Möglichkeiten für eine formatierte Eingabe zur Verfügung. Dabei ist in <fensterstr> die Information gespeichert, die zur Definition eines Eingabefeldes nötig ist. <fensterstr> ist folgendermaßen aufgebaut (Tabelle 4):

```
CHR$(sp)+CHR$(ze)+CHR$(le)+CHR$(char)+CHR$(ctrl)+CHR$(pl)+xchar$
```

Dabei bedeuten:

ze die Zeile, in der das Eingabefeld beginnt
sp die Spalte, in der das Eingabefeld beginnt
le die Länge des Eingabefeldes
char definiert die Zeichen, die als Eingabe erlaubt sind. Zeichen, die nicht erlaubt sind, können auch nicht in das Feld eingegeben werden. Diese Definition geschieht bit-signifikant, wobei die einzelnen bits folgende Bedeutung haben:

Bit	Wert	Bedeutung, falls Bit=1
0	1	Buchstaben erlaubt (mit Umlauten)
1	2	Umwandlung in Großschrift
2	4	Ziffern erlaubt (0-9, ohne '.')
3	8	Alle Zeichen erlaubt
4	16	Leerzeichen (nur Maskengenerator)
5	32	Unterstreichen (nur Maskengenerator)
6	64	keine
7	128	Feld unterstreichen (bei Eingabe)

Die Bits 4 und 5 werden nur vom Maskengenerator verwendet, SCREEN INPUT ignoriert sie. Falls das Leerzeichen als Eingabe zulässig sein soll, muß es als Sonderzeichen definiert sein (siehe xchar).

ctrl definiert die Tasten, mit denen die Eingabe verlassen werden kann. Diese Größe ist auch bitsignifikant, dabei bedeuten:

Bit	Wert	Bedeutung, falls Bit=1
0	1	<oben> (Cursortaste nach oben)
1	2	<unten> (Cursortaste nach unten)
2	4	ENTER
3	8	ESC
4	16	CLR
5	32	keine
6	64	keine
7	128	keine

Falls das entsprechende Bit in ctrl auf eins gesetzt ist und die zugehörige Taste während der Eingabe gedrückt wird, wird die Eingabe verlassen. Unerheblich von der Aussprungtaste steht das Ergebnis der Eingabe immer in d.dat$.

pl wird nur vom Maskengenertator benutzt. SCREEN INPUT ignoriert diesen Wert.

xchar legt die Sonderzeichen fest, die außer den char definierten Zeichen erlaubt sind. Die Länge von xchar ist unbegrenzt. Dennoch ist es sinnvoll xchar so kurz wie möglich zu halten, damit nicht unnötig viel Speicherplatz für einen Fensterstring belegt wird.

Die Länge von d.dat$ muß größer der im Fensterstring definierten Länge (LE) sein. Die Fensterlänge (LE) muß größer null sein. Außerdem muß die Zeile (ZE) im Bereich von 1 bis 25, die Spalte (SP) im Bereich von 1 bis 80 liegen. Es wird nicht empfohlen, Fenster zu definieren, die über eine Zeile hinausgehen. Die Mindestlänge von 6 Zeichen für einen Fensterstring darf nicht unterschritten werden (auch wenn nicht alle Definitonen benötigt werden). Falls eine dieser Forderungen nicht erfüllt ist, bricht SCREEN INPUT mit ERROR 5: 'Improper argument' ab.
Während der Eingabe stehen folgende Editiermöglichkeiten inner-

halb des Eingabefeldes zur Verfügung:

<rechts>,<links> bewegen Cursor innerhalb des Fensters

DEL löscht Zeichen vor dem Cusor, der restliche Text im Fenster wird um ein Zeichen nach links verschoben.

TAB An der Position des Cursors wird ein Leerzeichen eingefügt (wie CTRL-E im Maskengenerator).

Neu eingegebene Zeichen werden nicht automatisch eingefügt, sondern überschreiben den evtl. schon vorhandenen Text.
Beispiele für Fensterstrings befinden sich im Anhang.

SCREEN LOCATE <fensterstr>

SCREEN LOCATE stellt eine andere Form des BASIC-Befehls LOCATE dar. Der Cursor wird in Zeile ZE und Spalte SP gestellt.

SCREEN CLEAR <fensterstr>

Löscht auf dem Bildschirm ein Fenster. Dazu wird der Cursor zuerst auf Zeile ZE und Spalte SP gestellt. Danach werden soviele Leerzeichen ausgegeben, wie in <fensterstr> definiert ist (LE).

Aufbau eines Fensterstrings <fensterstr>

Byte	Kürzel	Bedeutung
1	SP	Spalte, in der das Fenster beginnt
2	ZE	Zeile, in der das Fenster beginnt
3	LE	Länge des Fensters
4	CHAR	Erlaubte Zeichen
Bit	0	Buchstaben erlaubt
	1	Umwandlung in Großschrift
	2	Ziffern erlaubt (0-9, ohne '.')
	3	Alle Zeichen erlaubt
	4	Leerzeichen (nur Maskengenerator)
	5	Unterstreichen (nur Maskengenerator)
	6	keine Bedeutung
	7	Feld unterstreichen (bei Eingabe)

5	CTRL	Erlaubte Aussprungtaste
Bit	0	<oben>
	1	<unten>
	2	ENTER
	3	ESC
	4	CLR
	5	keine Bedeutung
	6	keine Bedeutung
	7	keine Bedeutung
6	PL	Wert für Anwenderfunktionen, siehe DIALOG
7-	XCHAR	Sonderzeichen, zusätzlich zu CHAR erlaubt

Tabelle 4: Aufbau eines Fensterstrings

GET <str>,<länge>,(<ende>)

Mit GET ist es möglich eine bestimmte Anzahl Zeichen von der Diskette zu lesen. Dabei arbeitet GET ähnlich dem BASIC-Befehl INPUT #9. Der eingelesene String wird in <str> abgespeichert. Dabei werd die in <länge> angegebenen Anzahl Zeichen gelesen (falls in der eröffneten Datei vorhanden). Optional kann zusätzlich ein Endezeichen angegeben werde. <ende> entspricht dem ASCII-Wert dieses Endezeichens. Dann wird entweder bis zum Erreichen der festegelegten Länge <länge> oder bis zum Erreichen des Endezeichens <ende> (meistens CR=0DH) gelesen.
Beispiele für die Verwendung von GET befinden sich im Anhang.
Außer diesen neuen Befehlen wurden zwei neue Funktionen eingeführt. Die eine (DERR) dient zur Abfrage der Diskettenfehlermeldung (nächster Abschnitt), die andere (CAT) dient zum Feststellen des freien Speicherplatzes auf der Diskette, d.h. PRINT CAT ergibt den freien Speicherplatz auf der Diskette. CAT läßt sich auch in anderen Ausdrücken verwenden (frei=CAT, IF CAT < 20 THEN). CAT bezieht sich auf das gerade angewählte Laufwerk. Dabei muß nach jedem Diskettenwechsel mindestens einmal auf die neue Diskette durch OPENIN, OPENOUT oder CAT (BASIC-Befehl, nicht die neue Funktion!) zugegriffen worden sein. Ansonsten ist der von CAT gelieferte Wert bedeutungslos.

Zusammenfassung der neuen Befehle

Befehl	Art	Bemerkung
SCREEN <dateiname>	B	eröffnet eine Maske und bringt sie auf den Bildschirm
SCREEN INPUT <fensterstr>	B	Eingabe in ein Fenster

SCREEN LOCATE <fensterstr>	B	Cursor an Anfang des Fensters stellen
SCREEN CLEAR <fensterstr>	B	Fenster am Bildschirm löschen
GET <str>,<länge>,(<ende>)	B	definierte Anzahl Zeichen von der Diskette lesen
|<extern>,<str>	B	In externen Befehlen kann ein String direkt übergeben werden (z.B. |era,"*.bak")
DERR	F	Diskettenfehlernummer
CAT	F	Freier Speicheplatz auf der Diskette

(B: Befehl, F: Funktion)

Tabelle 5: Zusammenfassung der XBASIC Befehlserweiterungen

4.2 Diskettenfehlermeldungen

Um auch Diskettenfehler mit der ON ERROR GOTO Anweisung abfangen zu können, wurde die Fehlermeldungsroutine des BASIC-Interpreters erweitert. Die beiden neuen Fehlermeldungen heißen:

```
31  File not open
32  Broken
```

Der Fehler 'File not open' erklärt sich selbst. Fehler 32 tritt immer dann auf, wenn ein Fehler von den Diskettenroutinen gemeldet wird, der bei den Cassettenroutinen nicht auftreten kann. Z.B.:

```
OPENIN ""

falsche Eingabe
Broken
Ready
```

Immer dann, wenn Fehler 32 aufgetreten ist, steht in DERR die Fehlernummer, die von der entsprechenden Diskettenroutine zurückgegeben worden ist. Mögliche Fehlermeldungen sind:

```
16  falsche Eingabe
17  Die Datei existiert bereits
18  Die Datei existiert nicht
19  Das Inhaltsverzeichnis ist voll
20  Die Diskette ist voll
21  Die Diskette wurde gevechselt, obwohl noch Dateien
    offen waren
22  Die Datei ist nur lesbar
```

Außerdem werden noch Fehlermeldungen vom FDC (Floppy Disc Controller) durchgereicht. Die einzelnen Bits bedeuten:

Bit	Bedeutung
0	Die Adressmarke wurde nicht gefunden
1	Die Diskette ist schreibgeschützt
2	Der Sektor kann nicht gefunden werden, oder seine ID-Marke kann nicht fehlerfrei gelesen werden
3	Das Laufwerk meldet nicht READY, d.h. es ist entweder kein Laufwerk angeschlossen oder es ist keine Diskette im Laufwerk
4	Das TRACK0 Signal wird auch nach 77 STEP Impulsen nicht vom Laufwerk gemeldet (Hardware Fehler)
5	Prüfsummenfehler im Daten- oder ID-Feld

Eigentlich sollte bei den FDC-Fehlernummer das Bit 6 gesetzt sein, um diese von den anderen zu unterscheiden. Leider geschieht das nicht. Dadurch ist es nicht immer möglich anhand der Fehlernummer die Ursache eindeutig festzustellen.
Falls Bit 7 gesetzt ist, wurde der Fehler dem Anwender bereits mitgeteilt (tritt bei ON ERROR GOTO nicht auf).
Die Möglichkeit Diskettenfehler abzufangen hat auch einige Nachteile, die beachtet werden sollten. So ist es zum Beispiel nicht möglich eine Datei ordnunggsgemäß zu schließen, falls diese schreibgeschützt ist und schon ein Schreibversuch gestartet wurde, der mit ON ERROR GOTO abgefangen worden ist. Die einzige Möglichkeit die Datei zu schließen besteht darin, mit !ATTRIBUT den Schreibschutz zu entfernen und dann mit CLOSEOUT die Datei zu schließen. Dann wird allerdings die alte Datei überschrieben. Falls das nicht erwünscht ist, kann auf die Diskette nichts mehr geschrieben werden, da bei jedem CLOSEOUT ein 'File already open' Fehler vom BASIC-Interpreter erzeugt wird.
Weitere Einschränkungen sind im Kapitel 6 beschrieben.

4.3 XBASIC-R

XBASIC-R ermöglicht dieselben Erweiterungen wie auch XBASIC. Allerdings fehlen die Routinen zur Umwandlung der neuen Schlüsselwörter in Token und Rückwandlung der Token in die entsprechenden Texte (siehe DIALOG-R). XBASIC-R ist also ein reines RUN TIME Modul, andere Anwendungen sind nicht sinnvoll.

5. Beispielprogramm ADRESSEN

Dieses Beispiel zeigt die Verwendung von MASKGEN und DIALOG. Zuerst wurde eine Maske erstellt, die alle für eine Adressenverwaltung notwendigen Informationen darstellen und aufnehmen kann. Diese Maske heißt ADRESSEN.MSK und ist auf der Diskette vohanden. Hier soll anhand einiger Felder gezeigt werden, wie die Fenster definiert sind:

Anrede-Feld:

```
NR:4  SP:12  ZE:7  LE:15  PL:1  CHAR:B  SU  CTRL:OUE C  XCHAR:.-
```

Spalte und Zeile ergeben sich aus der Position von 'Anrede:'. Die Länge wurde mit 15 relativ hoch gewählt, damit alle denkbaren Anrede eingebbar sind. Die Plausibilitätskontrolle wurde für diese Feld auf eins gesetzt. Dadurch ist es dann im Programm ADRESSEN möglich, eine Kürzelfunktion für die häufigen Anreden Frau, Herr und Firma einzubinden. Als Zeichen wurden nur Buchstaben (B) und das Leerzeichen (S) erlaubt. Das Feld wird bei der Eingabe unterstrichen (U). Als Aussprungtasten wurden <oben> (O), <unten> (U), ENTER (E) und CLR (C) erlaubt. Da in der Anrede Abkürzungen (z.B. Dr.) und zusammengesetzte Wörter vorkommen können wurden '.' und '-' als Sonderzeichen erlaubt.

Wohnort-Feld:

```
NR:7  SP:12  ZE:11  LE:30  PL:0  CHAR:B  SU  CTRL:OUE C  XCHAR:.-
```

Spalte und Zeile ergeben sich wieder aus der Position von 'Wohnort:'. Auch hier wurde die Länge mit 30 sehr hoch gewählt, damit auch alle Orte erfasst werden können. Für dieses Feld existiert keine Anwenderfunktion (PL=0). Zeichen, Aussprungtasten und Sonderzeichen sind wie im Anrede-Feld definiert.

Telefon-Feld:

```
NR:11  SP:12  ZE:14  LE:13  PL:0  CHAR: Z SU  CTRL:OUE C  XCHAR:/
```

Die Spalte und Zeile ergibt sich wieder aus der Position von 'Telefon:'. Mit einer Länge von 13 ist das Feld bestimmt für alle Telefonnummern ausreichend (Vorwahl max. 5 Ziffern, ein Trennzeichen '/', Rufnummer dann max. 7 Zeichen). Auch in diesem Feld wurde keine Anwenderfunktion ermöglicht. Als Zeichen wurden diesmal nur Ziffern und das Leerzeichen (zum Überschreiben) erlaubt. Die Aussprungtasten sind wie in den beiden vorherigen Feldern definiert. Als Sonderzeichen ist das Trennzeichen zwischen Vorwahl und Rufnummer erlaubt ('/').
Das Programm ADRESSEN.BAS zeigt das Zusammenspiel zwischen DIALOG und dem Anwenderprogramm. Es ist in folgende Abschnitte eingeteilt:

10 - 99	Initialisierung
100 - 999	Hauptschleife (Befehlseingabe und Aufruf)
1000 - 6999	Befehle #1 bis #6
50000 - 50140	Anwenderfunktion #1 (Kürzel)
64000 - 64999	Anwenderaussprünge DIALOG
65000 -	DIALOG

Da im Listing ausführliche Kommentare enthalten sind, wird hier nur noch auf die Grobstruktur eingegangen.
Nach der Initialisierung des Systems wird in der Hauptschleife auf die Eingabe eines Befehls gewartet. Danach wird je nach eingegebener Befehlsnummer das entsprechende Unterprogramm aufgerufen. Zur Befehlseingabe wird die DIALOG-Funktion 5 verwendet. Hier wird deutlich, wie ein Systemfenster definiert wird. In d.sys1$ und d.sys2$ wird der auszugebende Text abgespeichert. Als Eingabe sind nur Ziffern zulässig, das Feld ist unterstrichen (d.char%=4+128). Als Aussprungtaste ist lediglich ENTER definiert (d.ctrl%=4).
Der erste Befehl (Zeile 1000-1999) zeigt die Verwendung der DIALOG-Funktion 3. Dabei werden d.ein%, d.min% und d.max% so gesetzt, daß eine Eingabe in das Uhrzeit- und Datumsfeld nicht möglich ist.
Eine interessante Möglichkeit der Plausibilitätskontrolle zeigt die erste (und einzige) Anwenderfunktion in Zeile 50000-50140. Diese Routine wird nur aufgerufen, wenn PL des aktuellen Fensters den Wert 1 hat. Dort wird dann geprüft, ob in das Feld eines der Kürzel f, F oder h eingegeben wurde. Falls ja wird es durch das entsprechende Wort Frau, Firma oder Herr ersetzt. Dabei muß die Zuweisung an d.dat$ über MID$ erfolgen, um dessen Länge nicht zu verändern (siehe Hinweise zu DIALOG-Funktion 3).
Eine weitere Besonderheit zeigen die Zeilen 64350 und 64360. Dort wird nach jedem Aufruf der DIALOG-Funktion 2 (Maske laden) das Datum und die Uhrzeit auf den Bildschirm gebracht. Falls in diesem Programm mehrere Masken Verwendung finden sollen, müßten alle die Fenster #0 und #1 als Datums- und Uhrzeitfeld definiert haben.
Mit RUN "ADRESSEN" wird das Programm gestartet. Es empfiehlt sich, alle Funktionen diese Programmes auszuprobieren, um einen Eindruck von den Fensterdefinitionen zu erhalten. Dabei können dann auch eigene Änderungen in der Maske ADRESSEN.MSK getestet werden.

6. Hinweise zur Benutzung

Die Befehlserweiterungen XBASIC und XBASIC-R sind beide verschieblich. Sie benötigen beim Initialisieren 3811 (XBASIC) bzw. 3623 (XBASIC-R) Bytes Speicher. Danach (also zur Laufzeit) werden nur noch 1619 bzw. 1525 Bytes belegt. Um die Erweiterungen in den Speicher zu bringen, muß die Ladeadresse hinter LOAD angegeben werden (siehe DIALOG). Im Dialog ist die Ladeadresse so gewaehlt, daß die Erweiterungen direkt unterhalb von HIMEM im Speicher stehen. Es ist jedoch auch möglich, die Erweiterung an

jeden beliebigen Platz im Speicher zu bringen. Dazu dient folgendes Programm (z.B.):

```
10 MaxAdr=39999
20 '
30 ' Beispiel: höchste benutzte Adresse = 39999
40 '
50 HMem=MaxAdr-3811          :'fuer XBASIC
60 MEMORY HMem               :'Speicher fuer Initialisierung
70 LOAD "XBASIC",HMem+1      :'Erweiterung in diesen Bereich laden
80 CALL HMem+1               :'Erweiterung initialisieren
90 MEMORY HMem+2192          :'2192:=3811-1619
100 '
110 ' Die letzte durch XBASIC belegte Speicherzelle ist jetzt
120 ' 39999
130 '
140 END
```

Die Speicherreservierung MEMORY HMem+2192 darf erst nach der Initialisierung der Routine (durch CALL HMem+1) erfolgen.
Um die neuen Befehle in den BASIC-Interpreter einzubinden werden die BASIC Indirections ab OAC01H verwendet. Beim initialisieren der Erweiterung werden diese Vektoren überschrieben. Es wird nicht geprüft, ob vielleicht schon eine andere Erweiterung diese Vektoren verwendet. Daher sollte eine andere Erweiterung, die diese Vektoren benutzt, vorher abgeschaltet werden.
Um die Diskettenfehler abzufangen werden in der Sprungtabelle des Betriebssystems einige Einträge überschrieben. Diese Einträge sind:
DISC IN OPEN, DISC IN CHAR, DISC IN DIRECT, DISC OUT OPEN, DISC OUT CHAR, DISC OUT DIRECT, DISC OUT CLOSE.
Das hat zur Folge, das beim Aufruf dieser Routinen die Fehlermeldungen, die eigentlich vom BASIC-Interpreter erzeugt werden sollte, jetzt sofort erzeugt werden. Das bedeutet u.a. das die definierten Schnittstellen nicht mehr den normalen Spezifikationen genügen. Beim Betrieb mit dem BASIC-Interpreter sind aber keine Schwierigkeiten zu erwarten. Falls aber andere Programme (außer BASIC) direkt auf diese Sprünge zugreifen, sollte zur Vermeidung unliebsamer Zwischenfälle die Erweiterung vorher abgeschalten werden (System Reset).
Außerdem wird beim Aufruf einer der oben genannten Routinen vorher noch TXT OUTPUT überschrieben, falls eine ON ERROR GOTO Anweisung im BASIC Programm vorhanden ist. Dadurch ist es möglich die Fehlermeldungen abzufangen, die das Diskettenbetriebssystem produziert. Allerdings ist es dann nicht mehr möglich während den Diskettenroutinen durch den Aufruf von TXT OUTPUT irgendwelche Ausgaben zu machen. Auch diese Änderung funktioniert mit dem eingebauten BASIC und dem Diskettenbetriebssytem einwandfrei, sollte jedoch bei Verwendung durch andere Programme abgeschalten sein. Da die BASIC-Indirections nur im BASIC 1.0 existieren und außerdem Unterprogramme im ROM des BASIC-Interpreters aufgerufen werden, ist diese Erweiterung nicht auf den CPC 664 übertragbar. Außerdem wird einmal in das ROM des Diskettenbetriebssystems gesprungen (neue Funktion CAT), d.h. auch an VDOS 1.0 ist diese Version gebunden (allerdings nur CAT, alle anderen Erweiterungen

laufen auch unter einem anderen Betriebssytem).
Als letztes wurde noch der Eintrag für CAS TEST EOF geändert. Die Änderung bewirkt, daß die BASIC-Funktion EOF bei reinen Text-Dateien richtig funktioniert.

7. AMSDOS, VDOS 1.0 oder VDOS 2.0 ?

Nachdem für den CPC 464 verschiedene Diskettenbetriebssysteme existieren, muß der Maskengenerator bzw. die BASIC Befehlserweiterung an das jeweilige Betriebssystem angepaßt werden. Dazu dienen die Programme AMSDOS.BAS, VDOS1.BAS und VDOS2.BAS. Wie aus dem Namen ersichtlich, installieren sie den Maskengenerator auf den Betrieb mit AMSDOS, VDOS 1.0 oder VDOS 2.0. Das entsprechende Programm wird mit RUN <Programmname> gestartet. Die Installation wird dann automatisch durchgeführt. Dazu müssen sich die drei Dateien XMASK.RSX, XBASIC.RSX und XBASIC-R.RSX auf der Diskette befinden!
Z.B.: Durch

```
RUN "VDOS2"
```

wird der Maskengenerator auf das Diskettenbetriebssystem VDOS 2.0 eingestellt.
Bei der Auslieferung ist das Programmpaket auf VDOS 1.0 eingestellt.
Unter AMSDOS und VDOS 1.0 sind alle Möglichkeiten des Maskengenerators wie oben beschrieben vorhanden. Unter VDOS 2.0 ist die neue Funktion CAT gesperrt.
Weitere Unterschiede sind:

- Diskettenfehler werden nicht abgefangen, da diese Aufgabe bereits von VDOS 2.0 übernommen wird.
- Die Vektoren CAS IN OPEN bis CAS CATALOG werden nicht überschrieben.
- Das EOF Problem wird nicht behandelt.

A N H A N G

I. Beispiele

Um die Befehle, die im folgenden besprochen werden, auf dem Rechner zu implementieren, sollte die Befehlserweiterung mit RUN "INITX" gestartet werden.

GET <str>,<länge>,(<ende>)

Dieser Befehl dient dazu, von der Diskette Strings zu lesen, die nicht durch die Steuerzeichen CR und LF beendet werden, oder in denen diese Steuerzeichen vorkommen. Dies ist zum Beispiel bei einem Fensterstring der Fall. Dabei wird zuerst die Länge des Strings und dann der String selbst auf die Diskette geschrieben.
Z.B.: Der String
SetWindow$=CHR$(26)+CHR$(1)+CHR$(13)+CHR$(10)+CHR$(25)
soll nach dieser Methode gespeichert werden. Dies wird durch folgende Anweisungen erreicht:

```
... : PRINT #9, LEN(SetWindow$);Setwindow$; : ...
```

Danach steht auf der Diskette (## ist nächstes freie Byte, die Zahlen sind Hex angegeben):

```
... 05 1A 01 0D 0A 19 ...
                      ##
```

Gelesen wird der String durch die Anweisungen:

```
... : GET Laenge$,1 : GET SetWindow$,ASC(Laenge$) : ...
```

Dabei wandert der Zeiger auf der Diskette folgendermaßen:

```
        Zeiger                              Programm

... 05 1A 01 0D 0A 19 ...          ... :
    ##

... 05 1A 01 0D 0A 19 ...          GET Laenge$,1 :
       ##

... 05 1A 01 0D 0A 19 ...          GET  SetWindow$,ASC(Laenge$)
                      ##           : ...
```

Mit dieser Methode werden die Fensterstring von MASKGEN bzw. DIALOG geschrieben bzw. gelesen.

SCREEN INPUT <fensterstring>

Hier sollen zwei einfache Eingabefelder definiert werden. Zuerst ein Feld, in das eine Kennziffer eingegeben werden soll. Diese Kennziffer soll folgendermaßen aufgebaut sein:

MMJJVVNN

Dabei ist MM das Geburtsmonat, JJ das Geburtsjahr, VV die ersten beiden Buchstaben des Vornamens und NN die des Nachnamens. Die Buchstaben sollen immer groß geschrieben sein, damit zwischen Groß- und Kleinschrift nicht unterschieden werden muß. Auf dem Bildschirm soll in Zeile 8 'Kennziff:' stehen. Damit ist ZE auf 8 und SP auf 12 festgelegt. Die Länge des Eingabefeldes ist durch 'MMJJVVNN' auf 8 festgelegt. Als Zeichen müssen Buchstaben und Ziffern erlaubt sein, wobei die Buchstaben automatisch in Großschrift gewandelt werden sollen. Um die Übersichtlichkeit zu erhöhen, soll das Feld unterstrichen sein. Dadurch ergibt sich CHAR 1+2+4+128=135. Als Aussprungtaste soll lediglich ENTER definiert sein, d.h. CTRL ergibt sich zu 4. Sonderzeichen werden keine benötigt und PL wird auf 0 gesetzt, da es von SCREEN INPUT sowieso ignoriert wird. Damit ergibt sich der Fensterstring zu:

```
Bsp1$=CHR$(12)+CHR$(8)+CHR$(8)+CHR$(135)+CHR$(4)+CHR$(0)
```

Mit d.dat$=" ":SCREEN INPUT Bsp1$ lassen sich jetzt Eingaben in das Feld machen. Diese Eingabe steht dann in d.dat$. Eine Zeile tiefer steht auf dem Bildschirm 'Geschlecht:'. In diesem Feld sollen nur zwei Buchstaben erlaubt sein, W und M. Dadurch ist ZE mit, SP mit 12 und LE mit 1 festgelegt. Da keine Gruppe von Zeichen erlaubt ist und ein Unterstreichen nicht sinnvoll ist (LE=1!), wird CHAR=0. Als Aussprungtaste soll ENTER, <oben> und <unten> definiert sein. Damit ergibt sich CTRL zu 1+2+3=6. Als Sonderzeichen muß W und M definiert sein, PL ist wieder null. Damit ergibt sich der Fensterstring zu:

```
Bsp2$=CHR$(12)+CHR$(9)+CHR$(1)+CHR$(0)+CHR$(6)+CHR$(0)+"wm"
```

Mit d.dat$=" ":SCREEN INPUT Bsp2$ lassen sich Eingaben in das

zweite Feld machen.

II. Listing/Fensterausdruck

```
10 '
20 ' Beispiel: ADRESSVERWALTUNG
25 '
29 '*************************************************************
30 '
40 ' --- DIALOG initialisieren, Maske ADRESSEN laden und anzeigen
45 '
50 d.fn%=1:GOSUB 65000
70 d.fn%=2:MID$(d.name$,1)="ADRESSEN":GOSUB 65000
98 '
99 '*************************************************************
100 '
110 ' Hauptschleife
120 '
130 WHILE NOT exit%
135 '
140 ' --- Systemzeilen belegen, anzeigen und Eingabe abwarten
150 ' --- Nur Ziffern erlauben, Laenge auf 2 setzen und ENTER als
160 ' --- einzige Aussprungtaste definieren, Feld unterstreichen
165 '
170  d.sys1$="1 Eingabe  2 Löschen  3 Suchen  4 Datum  5 Uhr  6 Ende"
180  d.sys2$="Eingabe " : case%=0
190  d.char%=4+128 : d.ctrl%=4 : d.len%=2
210  WHILE (case%<1) OR (case%>6)
220   d.fn%=5 : GOSUB 65000 : case%=VAL(d.dat$)
230  WEND
235 '
240 ' --- solange wiederholen, bis eine zulaessige Funktion
250 ' --- gewaehlt ist und dann Funktion aufrufen
255 '
260  ON case% GOSUB 1000,2000,3000,4000,5000,6000,7000
300 WEND
305 '
310 ' --- Programm beenden
315 '
320 MODE 2 : PRINT "Ende." : END
998 '
999 '*************************************************************
1000 '
1010 ' Funktion 1, Eingabe
1020 '
1030 ' --- Alte Eintraege loeschen, Datum und Uhrzeit schreiben
1040 ' --- Einsprung, maximal und minimal erlaubtes Feld setzen
1050 ' --- DIALOG aufrufen
1055 '
1060 d.fn%=4 : GOSUB 65000 : GOSUB 64250
1070 d.ein%=2 : d.min%=2 : d.max%=d.oben%-1
1080 d.fn%=3 : GOSUB 65000
1100 RETURN
```

```
1998 '
1999 '************************************************************
2000 '
2010 ' Funktion 2, Loeschen
2020 '
2030 ' --- Hier kann eine Funktion zum Loeschen beliebiger
2040 ' --- Eintraege, die moeglicherweise auf der Diskette
2050 ' --- abgespeichert sind, eingefuegt werden
2060 '
2070 RETURN
2998 '
2999 '************************************************************
3000 '
3010 ' Funktion 3, Suchen
3020 '
3030 ' --- siehe Funtion 2, Loeschen (Zeile 2000-2060)
3040 '
3050 RETURN
3998 '
3999 '************************************************************
4000 '
4010 ' Funktion 4, Datum
4020 '
4030 ' --- DIALOG nur mit Datum und Uhrzeitfeld (Nr. 0,1) aufrufen
4040 '
4050 d.ein%=0 : d.min%=0 : d.max%=1
4060 d.fn%=3 : GOSUB 65000
4070 RETURN
4998 '
4999 '************************************************************
5000 '
5010 ' Funktion 5, Uhr
5020 '
5030 ' --- Hier kann eine Funktion eingfuegt werden, die die
5040 ' --- aktuelle Uhrzeit auf dem Bildschim in dem dafuer
5045 ' --- vorgesehenen Feld anzeigt
5050 '
5060 RETURN
5998 '
5999 '************************************************************
6000 '
6010 ' Funktion 6, Ende
6020 '
6030 ' --- exit% auf TRUE setzen
6040 '
6050 exit%=-1
6060 RETURN
6998 '
6999 '************************************************************
50000 '
50010 ' PL=1, Anrede-Feld
50020 '
50030 ' --- Hier wird eine Kuerzelfunktion ermoeglicht
50040 ' --- Jedes
50050 ' ---          h wird in Herr
```

```
50060 ' ---          f wird in Frau
50070 ' ---          F wird in Firma
50080 ' --- verwandelt.
50090 '
50100 IF MID$(d.dat$,1,10)="h          " THEN MID$(d.dat$,1)="Herr"
50110 IF MID$(d.dat$,1,10)="f          " THEN MID$(d.dat$,1)="Frau"
50120 IF MID$(d.dat$,1,10)="F          " THEN MID$(d.dat$,1)="Firma"
50130 SCREEN LOCATE d.win$(d.ein%):PRINT MID$(d.dat$,1,5)
50140 RETURN
64000 '
64010 ' *** Dialog Ansprung Anwender Initialisierung ***
64020 '
64030 ' --- Hier koennen alle vom Anwender benoetigten Variablen
64040 ' --- initialisiert werden. Ausserdem ist hier ein SYMBOL
64050 ' --- AFTER Befehl erlaubt, um eigene Zeichen zu definieren.
64060 ' --- In diesem Fall wird nur das Datum festgesetzt
64065 '
64070 datum$="20.07.85":uhrzeit$="00:00:00"
64080 RETURN
64250 '
64260 ' *** Dialog Ansprung Anwender Maskeninitialisierung ***
64270 '
64280 ' --- Dieses Unterprogramm wird nach jedem Masken laden
64290 ' --- aufgerufen. Da durch das Laden einer Maske saemtliche
64300 ' --- Fenster auf volle Bildschirmgroesse gestellt werden,
64310 ' --- sollten hier alle Fensterdefinitionen stehen.
64320 ' --- In diesem Fall wird das aktuelle Datum mit auf den
64330 ' --- Bildschirm gebracht.
64340 '
64350 SCREEN LOCATE d.win$(0):MID$(d.dat$(0),1)=datum$
64355 PRINT datum$;
64360 SCREEN LOCATE d.win$(1):MID$(d.dat$(1),1)=uhrzeit$
64365 PRINT uhrzeit$;
64370 RETURN
64500 '
64510 ' *** Dialog Ansprung Anwender Funktionen ***
64520 '
64530 d.perr%=0:d.off%=0:d.aus%=0
64540 '
64550 ' --- Diese Routine wird nach jeder Eingabe im Dialog
64560 ' --- aufgerufen. In der Variablen d.pl% steht die
64570 ' --- aktuelle Plausibilitaet. Dadurch koennen in
64580 ' --- Abhaengigkeit vom Fenster unterschiedlich Unter-
64590 ' --- programme aufgerufen werden. Falls in d.pl% Null
64595 ' --- steht sollte keine Aktion ausgefuehrt werden.
64600 '
64610 ON d.pl% GOSUB 50000
64620 RETURN
64750 '
64760 ' *** Dialog Ansprung Fehler ***
64770 '
64780 ' --- Jeder Fehler im Dialog wird an diese Stelle
64790 ' --- weitergereicht. Kann bei Einhaltung der
64800 ' --- Uebergabeparameter nicht eintreten. Dabei steht in
64810 ' --- d.err% die Fehlernummer, in d.derr% die Disketten-
```

```
64815 ' --- fehlernummer und in d.line die Zeilennummer
64820 '
64830 MODE 2:PRINT"DIALOG: Fehler:";d.err%;" in Zeile";d.line
64835 RESUME 64840
64840 END
```

FENSTERDEFINITIONEN Maske: ADRESSEN

Nummer	Spalte	Zeile	Länge	Plaus.	Zeichen	Aussprung	Sonderzeichen
NR: 0	SP: 9	ZE: 1	LE: 8	PL:0	CHAR: Z SU	CTRL: UE C	XCHAR:.
NR: 1	SP: 9	ZE: 2	LE: 8	PL:0	CHAR: Z SU	CTRL:OUE C	XCHAR::
NR: 2	SP:57	ZE: 7	LE: 8	PL:0	CHAR: Z SU	CTRL:OUE C	XCHAR::
NR: 3	SP:57	ZE: 8	LE: 5	PL:0	CHAR: Z SU	CTRL:OUE C	XCHAR:
NR: 4	SP:12	ZE: 7	LE:15	PL:1	CHAR:B SU	CTRL:OUE C	XCHAR:.-
NR: 5	SP:12	ZE: 8	LE:25	PL:0	CHAR:B SU	CTRL:OUE C	XCHAR:.-
NR: 6	SP:12	ZE: 9	LE:20	PL:0	CHAR:B SU	CTRL:OUE C	XCHAR:.-
NR: 7	SP:12	ZE:11	LE:30	PL:0	CHAR:B SU	CTRL:OUE C	XCHAR:.-
NR: 8	SP:57	ZE:11	LE: 4	PL:0	CHAR: Z SU	CTRL:OUE C	XCHAR:
NR: 9	SP:12	ZE:12	LE:25	PL:0	CHAR:B SU	CTRL:OUE C	XCHAR:.-
NR:10	SP:57	ZE:12	LE: 4	PL:0	CHAR:B Z SU	CTRL:OUE C	XCHAR:
NR:11	SP:12	ZE:14	LE:13	PL:0	CHAR: Z SU	CTRL:OUE C	XCHAR:/
NR:12	SP:12	ZE:16	LE:69	PL:0	CHAR: A U	CTRL:OUE C	XCHAR:
NR:13	SP:12	ZE:17	LE:69	PL:0	CHAR: A U	CTRL:OUE C	XCHAR:

Anzahl der Fenster: 14 Belegter Speicherplatz: 437 bytes

III. Tabellen

Editierbefehle:

Taste	Wirkung	Bemerkung
<rechts>,<links>, <oben>,<unten>	bewegen Cursor	gesamter Anwenderbereich
SHIFT-<rechts>, <links>,<oben>, <unten>	bewegen Copycursor	gesamter Anwenderbereich
COPY	Zeichen kopieren	
CTRL-D	Copycursor löschen	
CTRL-<oben>	Zeile löschen	
CTRL-<unten>	Zeile einfügen	

CLR, DEL	Zeichen löschen	CLR löscht Zeichen unter Cursor, DEL vor Cursor
CTRL-E	Zeichen einfügen	
CTRL-<links>	Cursor an Anfang der Zeile	
CTRL-<rechts>	Cursor an Ende der Zeile	
TAB	Tabulator	
CTRL-TAB	Tabulator setzen bzw. löschen	
ENTER	Cursor an Anfang der nächsten Zeile	
CTRL-L	Bildschirm löschen	
CTRL-Q	Zeile bis Cursor löschen	
CTRL-R	Zeile ab Cursor löschen	
CTRL-S	Bildschirm bis Cursor löschen	
CTRL-T	Bildschirm ab Cursor löschen	
CTRL-X	Schreibfarben vertauschen	
CTRL-^	Cursor in die linke obere Ecke	
CTRL-C	Externen Befehl aufrufen	

Tabelle 1: Zusammenfassung der Editierbefehle

Externe Befehle:

Kennbuchstabe	Befehl
l	Maske laden
s	Maske speichern
i	Inhaltsverzeichnis aller Masken
f	Fenster Definition
F	Fenster anzeigen (invertieren)
L	Fenster löschen
E	Fenster einfügen

d	Fensterdefinitionen ausdrucken
k	Fensterzeile kopieren
t	Tastatur belegen
x	Erweiterungsstring definieren
X	Verdeckte Zeichen ermöglichen
q	Rückkehr in den Editiermodus (Befehlsabbruch)
Q	Verlassen des Maskengenerators

Tabelle 2: Zusammenfassung der externen Befehle

DIALOGfunktionen

d.fn%	Funktion	Bemerkung
1	DIALOG initialisieren	Sämtliche DIALOG-Variable werden mit ihren Standardwerten belegt.
2	Maske laden	Eine Maske anzeigen und alle Fensterstrings laden.
3	Dialog aufrufen	Eingabe in definierte Felder.
4	Eingabefelder löschen	Sämtliche Fenster der Maske löschen.
5	Eingabe in Systemzeilen	Eingabe in den Zeilen 24 und 25 (Systemzeilen).

Tabelle 3: Dialogfunktionen

Aufbau eines Fensterstrings <fensterstr>

Byte	Kürzel	Bedeutung
1	SP	Spalte, in der das Fenster beginnt
2	ZE	Zeile, in der das Fenster beginnt
3	LE	Länge des Fensters
4	CHAR	Erlaubte Zeichen

Bit	0	Buchstaben erlaubt
	1	Umwandlung in Großschrift
	2	Ziffern erlaubt (0-9, ohne '.')
	3	Alle Zeichen erlaubt
	4	Leerzeichen (nur Maskengenerator)
	5	Unterstreichen (nur Maskengenerator)
	6	keine Bedeutung
	7	Feld unterstreichen (bei Eingabe)
5	CTRL	Erlaubte Aussprungtaste
Bit	0	<oben>
	1	<unten>
	2	ENTER
	3	ESC
	4	CLR
	5	keine Bedeutung
	6	keine Bedeutung
	7	keine Bedeutung
6	PL	Wert für Anwenderfunktionen, siehe DIALOG
7	XCHAR	Sonderzeichen, zusätzlich zu CHAR erlaubt

Tabelle 4: Aufbau eines Fensterstrings

Zusammenfassung der neuen Befehle

Befehl	Art	Bemerkung
SCREEN <dateiname>	B	eröffnet eine Maske und bringt sie auf den Bildschirm
SCREEN INPUT <fensterstr>	B	Eingabe in ein Fenster
SCREEN LOCATE <fensterstr>	B	Cursor an Anfang des Fensters stellen
SCREEN CLEAR <fensterstr>	B	Fenster am Bildschirm löschen
GET <str>,<länge>,(<ende>)	B	definierte Anzahl Zeichen von der Diskette lesen
!<extern>,<str>	B	In externen Befehlen kann ein String direkt übergeben werden (z.B. !era,"*.bak")
DERR	F	Diskettenfehlernummer
CAT	F	Freier Speicherplatz auf der Diskette

Tabelle 5: Zusammenfassung der XBASIC Befehlserweiterungen

Symboltabelle der Umlaute

Zeichen	Werte (Reihenfolge wie im SYMBOL Befehl)
240	90,60,102,102,126,102,102,0
241	186,108,198,198,198,108,56,0
242	102,0,102,102,102,102,60,0
243	72,0,120,12,124,204,118,0
244	36,0,60,102,102,102,60,0
245	68,0,102,102,102,102,62,0
246	56,108,108,108,102,102,108,96

Tabelle 6: Symboltabelle der Umlaute