Markus Fulde

BASIC-Compiler

für CPC 464/664/6128

Autoren: Fa. GHE Software, Aachen

Aachen, Juni 1988

B A S I C - C O M P I L E R
für CPC 464,664,6128

1.0 Einleitung:

Seit langem vermißt man auf dem reichlichen Softwaremarkt für die CPC's von Schneider einen BASIC-Compiler, der den vollen Sprachumfang des Locomotive-BASICs umfaßt. Dieser Mißstand wurde nun mit dem hier vorliegenden Compiler, der vollständig in Maschinensprache geschrieben wurde, endgültig behoben. Ein kompiliertes Programm ist auf jedem CPC (auch mit vortex-Zusatzhardware) lauffähig. Der Compiler versteht den Sprachumfang des BASIC 1.1 vom CPC 664 und 6128. Programme, die spezielle BASIC 1.1 Befehle enthalten, sind auch auf dem 464 lauffähig, sofern es sich nicht um die Befehle FILL und MASK handelt.

1.1 Hilfe bei Problemen:

"Nichts ist so unvermeidlich wie ein Fehler, für den die Zeit reif ist"

(aus Murphy's Gesetzen, Goldmann TB ISBN 3-442-10046-1)

Aus diesem Grund wird es sich wohl nicht vermeiden lassen, daß der ein oder andere User noch einen Bug in dem Compiler findet.
Sollten Sie also schwerwiegende Probleme bei der Compilierung Ihrer Programme haben, oder meinen, einen Fehler in dem Compiler gefunden zu haben, so gibt es zwei Möglichkeiten für Sie, sich direkt (dies geht am schnellsten) an die Autoren wenden zu können:

telephonisch: Usersprechstunde der Firma GHE, Dienstags abends von 18 - 21 Uhr
0241/162 192

schriftlich: Firma
GHE
z.Hd. Herrn Gunkel
Jülicherstr. 312
5100 Aachen

Bei schriftlichen Anfragen legen Sie bitte eine Diskette mit dem 'Problemprogramm', sowie einer genauen Fehlerbeschreibung und ausreichend Rückporto bei.

2.0 Hardwarevoraussetzungen:

CPC 464, 664 oder 6128 mit einer Diskettenstation

2.1 Lieferunfang:

Folgende Files sollten Sie auf Ihrer Diskette finden:

- BC	.COM	der Compiler
- BC	.DOC	dieses File
- README	.DOC	wenn vorhanden, die neuesten Änderungen und Tips
- DEMO1	.BAS	Graphikdemo
- DEMO2	.BAS	Apfelmännchen
- DEMO3	.BAS	Türme von Hanoi
- DEMO4	.BAS	Landmark Basic CPC
- DEMO4	.BIN	wie oben nur compiliert
- DEMO5	.BAS	Quicksort
- DEMO6	.BAS	Startprogramm für DEMO61, DEMO62 u. DEMO63
- DEMO61	.BAS	Programmierung des Videobaustein 6845
- DEMO62	.BAS	Diskettenmonitor
- DEMO63	.BAS	RAM/ROM Monitor
- DISCMONI	.BIN	Maschinenspracheprogramm für DEMO62

3.0 Allgemeines zur Bedienung:

Kopieren Sie zunächst das Programm BC.COM auf eine CP/M-Systemdiskette.
Da der Compiler bei der Arbeit einiges mit Dateien zu tun hat, wurde als Arbeitsoberfläche CP/M 2.2 bzw. CP/M 3.0 gewählt, da dieses es dem Betriebssystem gestattet, auf mehr als zwei Dateien schnell zuzugreifen. Die compilierten Programme hingegen sind unter BASIC mit:

RUN"name.BIN"

zu starten. Hierbei wird jedoch nicht direkt ein Maschinenprogramm gestartet, was zu Problemen mit den vortex-Stationen führen könnte, sondern es wird dem Interpreter ein BASIC-Programm vorgetäuscht. Dieses besteht aber nur aus einer Zeile, die nur den angehängten Maschinenspracheteil startet.
Das zu compilierende BASIC-Programm wird ganz normal mit:

SAVE"name.BAS"

abgespeichert werden.

Nach dem Betriebssystemwechsel (mit !CPM) geben sie nach dem CP/M Prompt A> folgendes ein:

BC name <ENTER>

Damit wird die Übersetzung des Programms veranlaßt. Nach der Compilierung gibt der Compiler die Zahl der bei der Übersetzung aufgetretenen Fehler an und kehrt ins CP/M zurück. Ein Programm wurde nur dann erzeugt, wenn kein Fehler auftrat. Bei einem oder mehreren Fehlern müßen Sie ins BASIC zurückkehren und das Programm an den entsprechenden Stellen berichtigen und erneut compilieren. Somit läßt sich der Basic Compiler auch recht gut als Syntaxchecker verwenden.
Beim Aufruf des Compilers können noch zwei Parameter angehängt werden, die den erzeugten Programmcode beeinflußen:

L
Mit diesem Parameter werden die Informationen über die Zeilennummern im Binärprogramm unterdrückt. Das Programm wird um 6 Bytes pro Zeile kürzer und ein wenig schneller.

E
Mit diesem Parameter werden die synchronen EVENTS der FIRMWARE nicht mehr abgefragt, die Befehle AFTER, EVERY, ON SQ und alle damit zusammenhängenden Befehle funktionieren nicht mehr, auch wird die Abfrage der ESC-Taste unterdrückt. Hierdurch wird das Programm um 6 Bytes pro Befehl kürzer und noch etwas weiter beschleunigt.
Die Parameter werden beim Aufruf des Compilers hinter dem Namen des zu kompilerenden Programms durch einen Schrägstrich (/) getrennt eingegeben. z.B.:

BC name /LE <ENTER>

Durch die etwas langsame Fließkommaarithmetik im ROM liegt der Geschwindigkeitsgewinn bei der REAL-Arithmetik nur zwischen 50 und 400%, bei Interarithmetik ist das compilierte Programm jedoch bis zu 8x schneller (genaue Werte siehe DEMO4). Insgesamt ergibt sich nach DEMO4 ein 4 bis 5 mal größerer Durchsatz bei dem copilierten Programm gegenüber dem Interpreter (siehe auch bei 'Tips zur Programmierung unter Basic').

3.1 Compiler Fehlermeldungen:

Zunächst können alle vom BASIC bekannten Fehlermeldungen auftreten. Darüber hinaus können durch die Compilerstruktur weitere Fehler auftreten. So ist das Definieren mehrerer Funktion unter der gleichen Variablen unzuläßig und DEFINT,DEFREAL und DEFSTR dürfen nicht an jeder Stelle im Programm stehen (Meldung Variable not found). Also Definitionen an den Anfang des Programms!
Definitionen von Funktionen (DEF FN) müssen physikalisch vor der Benutzung der Funktion stehen (niedrigerere Zeilennummer). Des weiteren können Fehler beim übersetzen auftreten, die durch den Diskettenbetrieb zustande kommen, etwa wenn die Diskette voll ist. Alle weiteren Fehler sind selbsterklärend.
Falls ein compiliertes Programm zu lang wird (ich selber erzeugte mit einer Vorab-Version einmal ein 76k Programm) wird keine Fehlermeldung erzeugt, dieses Programm kann dann wohl nicht mehr geladen und gestartet werden. Betrachten Sie es als Anhalt, daß das Programm, abhängig von aktuellen HIMEM innerhalb Ihres Programmes, z.B. durch MEMORY gesetzt, mit etwa 2-3KByte 'Luft' in den Speicher gehen muß.

3.2 Unterschiede zum Interpreter:

Trotz weitestgehender Kompabilität bestehen noch Unterschiede zum Interpreter. Es folgt eine Übersicht über alle Befehle und ihr Vorhandensein.

Befehle:	Interpreter Basic 1.0	Basic 1.1	Compiler
ABS	x	x	x
AFTER	x	x	x
AND	x	x	x
ASC	x	x	x
ATN	x	x	x
AUTO	x	x	- (2)
BIN$	x	x	x
BORDER	x	x	x
CALL	x	x	x
CAT	x	x	x
CHAIN	x	x	- (2)
CHAIN MERGE	x	x	- (2)
CHR$	x	x	x
CINT	x	x	x
CLEAR	x	x	x
CLEAR INPUT	- (1)	x	x
CLG	x	x	x
CLOSEIN	x	x	x
CLOSEOUT	x	x	x
CLS	x	x	x
CONT	x	x	- (2)
COPYCHR$	- (1)	x	x
COS	x	x	x
CREAL	x	x	x
CURSOR	- (1)	x	x
DATA	x	x	x
DEC$	- (1)	x	x
DEF FN	x	x	x
DEFINT	x	x	x
DEFREAL	x	x	x
DEFSTR	x	x	x
DEG	x	x	x
DELETE	x	x	- (2)
DERR	- (1)	x	x
DI	x	x	x
DIM	x	x	x
DRAW	x	x	x
DRAWR	x	x	x
EDIT	x	x	- (2)
EI	x	x	x
ELSE	x	x	x

END	x	x	x
ENT	x	x	x
ENV	x	x	x
ERASE	x	x	x
ERL	x	x	x
ERR	x	x	x
ERROR	x	x	x
EVERY	x	x	x
EXP	x	x	x
FILL	-	x	x
FIX	x	x	x
FOR	x	x	x
FRAME	- (1)	x	x
FRE	x	x	x
GOSUB	x	x	x
GOTO	x	x	x
GRAPHICS PAPER	- (1)	x	x
GRAPHICS PEN	- (1)	x	x
HEX$	x	x	x
HIMEM	x	x	x
IF	x	x	x
INK	x	x	x
INKEY	x	x	x
INKEY$	x	x	x
INP	x	x	x
INPUT	x	x	x
INSTR	x	x	x
INT	x	x	x
JOY	x	x	x
KEY	x	x	x
KEY DEF	x	x	x
LEFT$	x	x	x
LEN	x	x	x
LET	x	x	x
LINE INPUT	x	x	x
LIST	x	x	- (2)
LOAD	x	x	x (3)
LOCATE	x	x	x
LOG	x	x	x
LOG10	x	x	x
LOWER$	x	x	x
MASK	-	x	x
MAX	x	x	x
MEMORY	x	x	x
MERGE	x	x	- (2)
MID$	x	x	x
MIN	x	x	x
MOD	x	x	x
MODE	x	x	x
MOVE	x	x	x
MOVER	x	x	x

NEW	x	x	x
NEXT	x	x	x
NOT	x	x	x
ON BREAK CONT	- (1)	x	x
ON BREAK GOSUB	x	x	x
ON BRAEK STOP	x	x	x
On ERROR GOTO	x	x	x
ON ** GOSUB	x	x	x
ON ** GOTO	x	x	x
ON SQ GOSUB	x	x	x
OPENIN	x	x	x
OPENOUT	x	x	x
OR	x	x	x
ORIGIN	x	x	x
OUT	x	x	x
PAPER	x	x	x
PEEK	x	x	x
PEN	x	x	x
PI	x	x	x
PLOT	x	x	x
PLOTR	x	x	x
POKE	x	x	x
POS	x	x	x
PRINT	x	x	x
PRINT SPC	x	x	x
PRINT TAB	x	x	x
PRINTT USING	x	x	x
RAD	x	x	x
RANDOMIZE	x	x	x
READ	x	x	x
RELEASE	x	x	x
REM	x	x	x
REMAIN	x	x	x
RENUM	x	x	- (2)
RESTORE	x	x	x
RESUME	x	x	x
RESUME NEXT	x	x	x (3)
RETURN	x	x	x
RIGHT$	x	x	x
RND	x	x	x
ROUND	x	x	x
RUN <name>	x	x	x
RUN <zeilennummer>	x	x	x
SAVE	x	x	x (3)
SGN	x	x	x
SIN	x	x	x
SOUND	x	x	x
SPACE$	x	x	x
SPEED INK	x	x	x
SPEED KEY	x	x	x
SPEED WRITE	x	x	x

SQ	x	x	x
SQR	x	x	x
STOP	x	x	x
STR$	x	x	x
STRING$	x	x	x
SYMBOL	x	x	x
SYMBOL AFTER	x	x	x
TAG	x	x	x
TAGOFF	x	x	x
TAN	x	x	x
TEST	x	x	x
TESTR	x	x	x
TIME	x	x	x
TROFF	x	x	- (2)
TRON	x	x	- (2)
UNT	x	x	x
UPPER$	x	x	x
VAL	x	x	x
VPOS	x	x	x
WAIT	x	x	x
WEND	x	x	x
WHILE	x	x	x
WIDTH	x	x	x
WINDOW	x	x	x
WINDOW SWAP	x	x	x
WRITE	x	x	x
XOR	x	x	x
XPOS	x	x	x
YPOS	x	x	x
ZONE	x	x	x

(1) Diese Befehle sind im Basic 1.0 des CPC 464 nicht vorhanden. User die den Compiler haben, können dennoch diese Befehle in dem zu compilierenden Programm benutzen. Ein Austesten unter Basic mit solchen Befehlen ist natürlich weiterhin nicht möglich.

(2) Die Befehle AUTO, CHAIN, CHAIN MERGE, CONT, DELETE, EDIT, LIST, MERGE, RENUM, TRON, TROFF sind nicht vorhanden.
Sie kann man grob in zwei Gruppen unterteilen:

- o Befehle zur Programmveränderung
- o Befehle zum Nachladen von Programmen

Beide Gruppen sind aus verständlichen Gründen nicht mit einem vernünftigen Aufwand/Nutzen Verhältnis zu implementieren. Die Befehle TRON und TROFF sind unnötig, da diese zur Programmentwicklung benutzt werden und mit dem Compiler eigentlich nur

fehlerfreie Programme übersetzt werden sollten.

(3) Diese Befehle haben einige Bedingungen, die beachtet werden müssen. Weiteres siehe bei Inkompatibilitäten.

Im folgenden eine kurze Beschreibung der zusätzlichen Befehle wie sie den Usern des CPC 464 zu Verfügung stehen:

CLEAR INPUT

Kommando: Löscht alle vorher eingegebenen Eingaben der Tastatur, sowie die, die sich im Tastatur-Speicher befinden.

COPYCHR$

COPYCHAR$(#<Ein/Ausgabegerät>)

Ein Zeichen auf aktueller Cursorposition wird gelesen und einem String zugewiesen. Dieser kann dann weiter verarbeitet werden. Mit <Ein/Ausgabegerät> sind die selben wie beim LOCATE, PRINT oder WINDOW-Befehl gemeint. Wird das gelesene Zeichen nicht erkannt, weil es sich zum Beispiel um Graphik handelt, so wird eine Null ausgegeben.

Beispiel:
```
10 CLS
20 PRINT"Oben links"
30 B$=""
40 FOR I=1 TO 10
50 LOACTE I,1
60 A$ = COPYCHAR$(#0)
70 B$ = B$ + A$
80 NEXT I
90 LOCATE 15,10: PRINT B$
```

Hier steht an der zuletzt genannten Position wieder der Text 'Oben links'.

CURSOR

CURSOR <Systemschalter>,<Benutzerschalter>

Kommando: Dieser Befehl dient dazu, bei Bedarf den Cursor ein und aus zu schalten. Der Parameter der für <Systemschalter> bzw. <Benutzerschalter> übergeben wird ist 0 oder 1. Bei dem Befehl INPUT wird der Systemschalter automatisch ein und bei dem Befehl INKEY$ automatisch aus geschaltet. Will man dies anders, so ist vor dem Befehl dies entsprechend zu setzen.

Beispiel:
```
10 CURSOR 1
20 PRINT"Fargen ?";
30 A$ = INKEY$: IF A$ ="" ThEN 30
40 PRINT A$
50 IF A$="j" OR A$="J" THEN ....
60 CURSOR 0
```

Einer der beiden Schalter kann weggelassen werden. Wird der Parameter nicht angegeben, so ändert sich die Stellung des betreffenden Schalters nicht.

DEC$

DEC$ (<numerischer Ausdruck>,<Formatschablone>)

Kommando: Richtet die Druckausgabe des <numerischen Ausdrucks> unter Verwendung der <Formatschablone> aus. Folgende Formatzeichen werden ausgewertet (siehe auch bei PRINT USING): + - # $ * # , . ^

DERR

Nach dem Aufruf wird der Variablen DERR der letzte Fehlercode des Diskettensystems zurück gegeben. Dieser kann dann weiter verwendet werden um z.B. eine entsprechende Meldung auszugeben, oder eine andere Fehlerroutine auszuführen.

Beispiel:

```
LOAD "name.xyz"

Wenn das File nicht vorhanden ist meldet das System:
NAME.XYZ not found
READY

PRINT DERR
```

146

Die Fehlernummern haben folgende Bedeutung:

0 oder 22	ESC wurde gedrückt
142	Stream ist nicht im richtigen Zustand
143	Ende der Datei erreicht (hard end)
144	Fehlerhaftes Kommando (in der Regel unzutreffender Dateiname)
145	Datei dieses Namens besteht bereits
146	Datei besteht nicht
147	Inhaltsverzeichnis ist voll
148	Diskette ist voll
149	Diskette wurde entnommen, ohne Datei zu schließen
150	Datei nur lesbar
154	Dateiende (soft end, CTRL-Z) erreicht

Diese Meldungen kamen vom AMSDOS. Fehlernummern direkt vom Disk-Controller sind bit-signifikant, wobei Bit 6 bei Diskettenfehlern immer gesetzt ist und Bit 7 = 1 ist wenn AMSDOS schon eine Meldung brachte.

Bit	Bedeutung
0	Adressenbezeichnung fehlt
1	Nicht beschreibbar - Diskette hat Schreibschutz
2	Keine DAten - Sektor unbekannt
3	Laufwerk nicht bereit - keine Diskette im Laufwerk
4	Überlauffehler
5	CRC Fehler - Datenfehler
6	immer 1 wenn Disk-Error
7	auf 1 wenn AMSDOS schon Fehler gemeldet hat

FRAME

FRAME

Funktion: Nach dem Aufruf wird das Schreiben graphischer Symbole auf dem Bildschirm mit dem Strahlrücklauf des Katodenstrahls synchronisiert. Der Effekt ist eine weichere Bewegung und ein Veringern des Flackerns.

GRAPHICS PAPER

GRAPHICS PAPER <Farbstift>

Kommando: Legt fest, auf welchem Farbhintergrund die Zeichnungen auf dem Graphik-Bildschirm erscheinen sollen. Beim Zeichnen durchgehender Linien ist die Hintergrundfarbe unsichtbar.
Die GRAPHICS-PAPER-Farbe bildet den Hintergrund für Zeichen, die unter Anwendung des TAG-Befehls auf den Graphik-Bildschirm geschrieben werden. Sie stellt außerdem die Standardfarbe dar, wenn der Graphik-Bildschirmbereich mit dem Befehl CLS gelöscht wird.

GRAPHICS PEN

GRAPHICS PEN <Farbstift>,<Hintergrundmodus>

Kommando: Bestimmt den <Farbstift> zum Zeichnen von Linien und Punkten. Der <Hintergrundmodus> ist entweder

0: nicht transparenter Hintergrund
oder
1: transparenter Hintergrund

Ansonsten gilt das bei GRAPHICS PAPER gesagte. Einer der beiden Parameter kann weggelassen werden (nicht jedoch das ','). Beim weggelassenen Parameter ändert sich die Einstellung nicht.

ON BREAK CONT

ON BRAK CONT

Funktion: Verhindert ein Unterbrechen des Programms mit der ESC-Taste. Vorsicht, da bei Fehler eventuell das Programm nur mit CTRL-SHIFT-ESC abgebrochen werden kann. Dann ist in der Regel das Programm im Speicher verloren (Ausnahme bildet das Arbeiten von der Ramdisk unter BOS 2.1).
Dieser Befehl wird innerhalb eine Programms wieder mit ON BREAK STOP aufgehoben.

3.3 Inkompatibilitäten

Unterschiede zum Compiler bestehen leider noch bei den folgenden Befehlen:

LOAD, SAVE, RESUME

Bei den Befehlen LOAD und SAVE sind nur Binärdateien erlaubt.
Der Befehl RESUME Zeilennummer darf nicht in ein Unterprogramm zeigen.
Die Befehle RESUME und RESUME NEXT werden zwar ausgeführt, das Ergebnis ist aber sehr unsicher, da im compilierten Programm keine Informationen über die Adresse des nächsten Befehls existieren. Es ist sehr wahrscheinlich, daß das Programm abstürtzt.

3.4 Programmumfang

Beim Compilieren von Programmen werden Sie feststellen, daß diese enorm anwachsen. Dies liegt einerseits daran, daß der BASIC-Interpreter ein Programm in komprimierter Form (in sogenannten Token) ablegt. Schon dadurch wird das Programm etwa doppelt so groß. Zum anderen legt der Compiler vor jedes Programm eine sogenannte "RUN-TIME-Library", eine Programmsammlung also, die vielbenutze Routinen des Compilers enthält. Viele dieser Routinen existieren zwar schon in ähnlicher Form im BASIC-Rom, aber wegen der Systemunabhängigkeit konnte nicht darauf zurückgegriffen werden. Sie kennen dies Art der Compilierung ja auch schon von Turbo Pascal. Auch hier wird eine etwa 11K große Library vorangestellt.

3.5 Der Compiler und Speichererweiterung von vortex

Besitzer einer Speichererweiterung der Firma vortex GmbH und des BOS 2.1 (2.0) steht eine besonders bequeme Möglichkeit zur Verfügung mit dem Compiler zu arbeiten. Hier kann ja die Ramdisk auch unter Basic als Laufwerk MD verwendet werden (umschalten mit !MD). Somit kann die Entwicklung am Interpreterprogramm auf der Ramdisk geschehen. Man sollte sich trotzdem angewöhnen, nach einigen Änderungen am Programm, dieses mit SAVE"laufwerk:name.xyz" auf der Diskette zu sichern. Wenn Sie nun den Compiler starten wollen, brauchen Sie nur ins CP/M zu wechseln. Hier ist die Ramdisk Laufwerk C. Unter der Voraussetzung, daß Sie den Compiler schon auf der Ramdisk haben, können Sie nun den Compiliervorgang wie oben beschrieben zu starten. Treten beim Compilieren Fehler auf, oder will man das compilierte Programm austesten ist man 'ruck zuck' wieder in Basic.

Schwierigkeiten bei der Verwendung der Ramdisk unter Basic treten dann auf, wenn Sie eigene RSX-Befehle verwenden, die den gleichen Namen haben, wie BOS 2.1 sie zur Verfügung stellt, aber eine andere Syntax. Solche Programme kann man nur mit abgeschaltetem BOS entwickeln und austesten. Somit ist der Komfort nun nicht mehr ganz so groß.
Erfahrungen mit anderen Speichererweiterungen (Data Media und dK'Tronics) liegen nicht vor.
Des weiteren wurden die erweiterten Basic-Befehle des BOS 2.1 nicht implementiert. Alle Befehle und Programme die bankübergreifend programmiert wurden, sind so nicht mit dem Compiler bearbeitbar.

3.6 Der Compiler und der CPC 6128 unter CP/M +

Die Compilierung kann sowohl unter CP/M 2.2 als auch unter CP/M+ erfolgen. Das Nutzen der zweiten 64K Bank unter Basic ist mit einem entsprechenden RSX-Befehl oder einem Maschinenspracheprogramm von Ihrem Basic Programm aus möglich.

4.0 Tips zur Programmierung unter Basic

An dieser Stelle noch einige Tips für Ihre BASIC-Programme (ob Sie sie nun compilieren oder nicht):
Verwenden Sie so oft wie möglich die Integer-Arithmetik.
So dauern die folgenden Schleifen interpretiert (Zeiten für die compilierte Version in Klammern):

```
10 FOR i=0 to 10000
20 NEXT i
```

ca. 11.1 sec (8.2 sec)

die Schleife:

```
10 FOR i%=0 to 10000
20 NEXT i%
```

hingegen nur 5.9 sec (1.1 sec).

Der nächste Tip ist mehr ein Hinweis, der sich an Benutzer der vortex-Speichererweiterung richtet: Das BOS 2.1 (2.0) verlangsamt Pro-gramme um etwa 15%. Wenn also darin enthaltene RSX-Befehle nicht im Programm verwendet werden, dann mit !DISBOS die Speicher-erweiterung (und damit das BOS) abschalten (siehe auch DEMO4).

Da die zu compilierenden Programme ca. 15-17 KByte Quellcode nicht überschreiten dürfen, programmieren Sie kurz wie möglich. Dies geht natürlich auf Kosten des Bedienungskomfort.
Eine Möglichkeit auch größere Projekte zu compilieren ist im Ansatz in DEMO6 aufgezeigt. Sie können Ihr Programm in möglichst unabhängige Teile aufsplitten. Gemeinsame Variablen können Sie oberhalb vom HIMEM abgelegen (durch PEEK und POKE wieder auf sie zugreifen) oder über ein BIN-File auf Diskette gespeichert und bei Bedarf wieder geladen werden. Denn bei RUN"name.xyz" wird der Speicher nicht gelöscht. Das neue Programm welches die Variablen an der selben Stelle erwartet, muß nur den selben MEMORY gesetzt haben. So kann man sich das Sichern auf Diskette ersparen (bei unterschiedlichen MEMORY ist Zwischenspeichern auf Diskette sinn-voll).
Den Teilprogrammen ist dann ein Hauptprogramm voranzustellen, welches das Menü enthält und von dem die anderen Programme mit RUN"name.BIN" gestartet werden. Diese Unterprogramme rufen beim Verlassen wieder das Hauptprogramm auf.
Sie können natürlich auch mit den Befehlen OPENOUT,CLOSEOUT,OPENIN und CLOSEIN Daten auf Diskette(Kasette) speichern. So kann man sich auch Variablenspeicher in einem Array vorstellen.

5.0 Die Beispielprogramme

Alle Beispielprogramme liegen aus Platzgründen nur als als Basic-File vor. Ausnahme ist nur DEMO4, welches die Leistungsfähigkeit des Compilers aufzeigt. Die Basic-Quellprogramme sind, soweit wie notwendig, dokumentiert. Weiter Erläuterungen finden Sie bei den entsprechenden Beispielen.

5.1 DEMO1 - Graphik

Hier sind einige Beispiel für die Graphikfähigkeiten der CPC's aufgeführt. Die Anregungen für diese Beispiel entstammen teilweise aus alten Veröffentlichungen. Beachten Sie die Verwendung von Integer-Arithmetik wo immer es ging. Eine weitere Verschnellerung ist durch die Verwendung von SIN, COS etc. nicht durch Berechnung, sondern aus Tabellenwerten machbar.

5.2 DEMO2 - Apfelmännchen

Hier finden Sie den Grundalgorithmus zur Berechnung des Apfelmännchens. Verbesserungen sind durch mehr Komfort bei der Eingabe oder Wahl des nächsten Ausschnittes optisch am letzten Bild denkbar. Hier macht sich die relativ langsame 8-Bit Technik bemerkbar. Auch die Tiefe ist dadurch (durch maximale Genauigkeit bei diesem Basic) beschränkt.

5.3 DEMO3 - Türme von Hanoi

Ein Grundlegendes mathematisches Problem umgesetzt in Basic. Da vollkommen in Integer gehalten ist hier ein großer Geschwindigkeitsgewinn feststellbar. Eignet sich auch recht gut zur Umsetzung in andere Programmiersprachen und Austestung deren Leistungsfähigkeit.

5.4 DEMO4 - Landmark

Hier finden Sie ein Beispiel, wie Sie einzelnen Funktionen Ihres Basic austesten können. Die Performance von 1 wurde eingestellt auf einem CPC 464 mit Diskettenstation und abgeschaltetem BOS. Erweiterung bezüglich z.B. Diskettenzugriffe sind noch denkbar. Hier können Sie auch die Wirkungsweise der Compileroptionen L und E bezüglich der Geschwindigkeit überprüfen. Maximal ist so ein Performance von knapp fünf erreichbar.

5.5 DEMO5 - Quicksort

Ein Algorithmus den Sie auch in Ihren Programmen benutzen können. Die Funktion, Unterschiede und Wirkungsweise von Sortieralgorithmen finden Sie in folgenden Büchern recht gut erklärt:

deutsch: Turbo Pascal Programmbibliothek
Jamsa/Nameroff
McGraw-Hill Verlag
ISBN 3-89028-097-8

und für Programmierfreaks in Englisch:
Algorithms
Robert Sidgewick
Addison-Wesley Verlag
ISBN 0-201-06672-6

5.6 DEMO6 - RUN"name.xyz"

Hier finden Sie ein Beispiel zum Aufruf von drei Unterprogrammen von einem Hauptprogramm aus mit dem RUN-Befehl. Alle Unterprogramme verwenden den MEMORY-Befehl. Unterprogramm 2 (DEMO62) lädt zusätzlich noch ein Maschinenspracheprogramm (DISCMONI.BIN). Die Übernahme von Variablen von einem Umterprogramm zum nächsten wurde hier nicht verwendet.

Unterprogramm 1 (DEMO61):
Hier wird der Videobaustein Ihres Rechners direkt programmiert. Falsche Werte können zm Absturz des Programms führen. Dies liegt aber nicht am Programm sondern an der Ansteuerung des Bausteins. Eine Beschädigung des Monitors oder Rechners ist allerdings ausgeschlossen. Mit 2xESC haben Sie wieder die Defaulteinstellung erreicht.

Unterprogramm 2 (DEMO62):
Hier finden Sie einen einfachen Diskettenmonitor. Er arbeitet auch mit den CPC's und den vortex-Laufwerken ohne Schwierigkeiten zusammen. Die Ramdisk wird nicht unterstützt.

Unterprogramm 3 (DEMO&#):
Hier können Sie sich Ihren Speicherbereich im RAM und ROM anschauen. Auch Erweiterungsroms sind anwählbar (z.B. ROM Nr. 7 Diskettenstation).

6.0 Quellen und weiterführende Literatur

CPC allgemein:

- ROM-Listing CPC 464/664/6128
  von Janneck & Mossakowski
  Markt & Technik Verlag
  ISBN 3-89090-134-4

- CPC 464 intern
  von Brückmann/Englisch/Gerits
  Data Becker Verlag
  ISBN 3-89011-080-0

- CPC 464 inside out
  von s. Huslik
  Huslik Verlag, Augsburg
  ISBN 3-925159-00-2

- CPC 464 Firmware
  Schneider Copmuter Division

- Handbücher des CPC 464, 664 und 6128

Compilerbau:

- Programming Languages
  von Allen B. Tucker
  McGraw-Hill Verlag
  ISBN 0-07-Y66617-2 bzw. 0-07-65416-6

- Compilerbau
  von Niklaus Wirth
  Teubner Verlag
  ISBN 3--519-32338-9

- Compilerbau
  von Aho/Sethi/Ullmann
  Addison-Wesley-Verlag
  ISBN 3-9251188-0-2

Weitere CPC Produkte

Best.-Nr.	Bezeichnung	Preis in DM
201	Copyshop (Hardcopy-Programm) Cass.	59,–
202	Copyshop 3" Disk.	69,–
203	Copyshop (Vortex) 5,25" Disk.	69,–
101	Power-Spiele, 4 Cass.	50,–
102	Power-Spiele, 4 3" Disk.	70,–
103	Starlest, Cass.	24,–
104	Starlest 3" Disk.	29,–
106	Know 3" Disk.	29,–
204	Special Offer No.1 3" Disk.	69,–
205	Special Offer No.2 3" Disk.	69,–
107	Special Offer No.3 3" Disk.	49,–
206	Context CPC Cass.	49,–
207	Context CPC 3" Disk.	59,–
1011	Fantastic Four 3" Disk.	49,–
108	Solid Gold 2 Cass.	35,–
109	Solid Gold 2 Disk. 3"	65,–
110	10 Hit Games 2 Cass.	44,–
111	10 Hit Games 2 Disk. 3"	59,–
112	The world's greatest Cass.	35,–
113	The world's greatest 3" Disk.	49,–
114	6 Computer Hits Cass.	35,–
115	6 Computer Hits 3" Disk.	49,–
116	Clever und Smart Cass.	35,–
117	Clever und Smart 3" Disk.	49,–
118	Driller Cass.	49,–
119	Driller 3" Disk.	59,–
130	Cyrus II Schach Cass.	12,95
131	Bubble Bobble Cass.	35,–
132	Bubble Bobble 3" Disk.	49,–
136	Werewolves 3" Disk.	49,–
137	California Games Cass.	35,–
138	California Games3" Disk.	49,–
139	Buggy Boy Cass.	35,–
140	Buggy Boy 3" Disk.	49,–
141	Combat School Cass.	32,–
142	Combat School 3" Disk	49,–
143	International Karate plus Cass.	35,–
144	International Karate plus 3" Disk.	49,–
149	Champion Ship Sprint	35,–
150	Champion Ship Sprint 3" Disk.	49,–
155	Mah Jong Cass.	35,–
156	Mah Jong 3" Disk	49,–
159	Dan Dare II Mekon's Revenge Cass.	29,–
160	Mekon's Revenge 3" Disk	49,–

Best.-Nr.	Bezeichnung	Preis in DM
525	Joystick	35,–
1101	Bedlam Cass.	35,–
1102	Bedlam 3" Disk	49,–
1103	Cybernoid Cass.	35,–
1104	Cybernoid 3" Disk.	49,–
1105	Druid 2 Cass.	32,–
1106	Druid 2 3" Disk	49,–
1107	Get Dexter!I Cass.	35,–
1108	Get Dexter!I 3" Disk.	49,–
1109	Leaderboard Cass.	35,–
1110	Leaderboard 3" Disk.	49,–
1111	Rampage Cass.	35,–
1112	Rampage 3" Disk.	49,–
1113	Sidearms Cass.	35,–
1114	Sidearms 3" Disk.	49,–
1115	Tetris Cass.	35,–
1116	Tetris 3" Disk.	49,–
1117	Blood Valley Cass.	35,–
1118	Blood Valley 3" Disk.	49,–
1120	Beyond Cass.	38,–
1121	Beyond 3" Disk.	54,–
1122	Zynaps 3" Disk	49,–
161	Space Shuttle Cass.	15,–
163	Boxing Cass.	15,–
165	Ballblazer Cass.	15,–
175	Wintersports Cass.	15,–
179	Spindizzy Cass.	15,–
167	Hacker II Cass.	15,–
177	Tempest Cass.	15,–
169	Star Riders II Cass.	15,–
171	Bib Trouble in Little China Cass.	15,–
173	Sailing Cass.	15,–
181	Ghosbusters 3" Disk.	22,–
162	Space Shuttle 3" Disk.	22,–
164	Boxing 3" Disk.	22,–
184	Little Computer People 3" Disk.	22,–
166	Ballblazer 3" Disk.	22,–
176	Wintersports 3" Disk.	22,–
180	Spindizzy 3" Disk.	22,–
168	Hacker II 3" Disk.	22,–
178	Tempest 3" Disk.	22,–
170	Star Raiders II 3" Disk.	22,–
172	Big Trouble in Little China 3" Disk.	22,–
174	Sailing 3" Disk.	22,–

zu beziehen über

DMV Software

Postfach 250 · 3440 Eschwege

Daten- und Medien-Verlagsges. mbH
Postfach 250, 3440 Eschwege