Das ist der C 128

Mehr als ein C 64 und mehr als ein CP/M-Computer – der C 128 ist angetreten, die Marktführerschaft von Commodore im Homecomputer-Markt zu sichern.

Nicht jeder Computer, der PC heißt, ist ein Personal Computer. Der C 128 PC zum Beispiel ist noch eine ganze Menge mehr. Auf ihm laufen nämlich neben Wordstar und dBase II auch noch Summer Games und Ghost Busters. Er vereinigt damit zwei für Commodore-Homecomputer ungewöhnliche Eigenschaften in sich: Er ist voll kompatibel zu einem anderen Commodore Computer, nämlich zum Commodore 64 , und er ist dank seines Z80A-Zweitprozessors uneingeschränkt CP/M-fähig.

Eigentlich besteht der C 128 daher aus drei Computern auf einer Platine, nämlich einem C 64, dann natürlich einem C 128 und schließlich noch einer CP/M-Maschine. Dementsprechend existieren drei grundsätzliche Betriebsarten, in denen der C 128 jeweils in einer anderen internen Hardware-Konfiguration läuft. Nach dem Einschalten ohne Steckmodul und CP/M-Diskette im angeschlossenen Laufwerk befindet sich der Computer im normalen C 128-Modus. Die Einschaltmeldung auf dem für den C 128 entwickelten 1901-Monitor versetzt jeden C 64-Besitzer in ehrfürchtiges Staunen: »122365 Bytes Free« heißt es da; gewiß nicht alltäglich für einen 8-Bit-Computer. Der Bildschirm hat im Textbetrieb eine Organisation von 40 Zeichen mal 25 Zeilen. Darüber hinaus kann im C 128- und im CP/M-Modus auch eine Darstellung mit 80 Zeichen mal 25 Zeilen (auf RGB- oder monochromem Monitor) gewählt werden.

Der C 128 kann die 16 Grundfarben des C 64 darstellen: in den Betriebsarten C 64 und C 128 dazu auch noch die bekannten 8 Sprites. Für den C 64-kompatiblen Betrieb ist das Basic 2.0 enthalten, für den CP/M-Betrieb wird das Betriebssystem CP/M 3.0 plus von der Diskette geladen.

Das Gehäuse des C 128 ist im modernen und eleganten ultraflachen Styling gehalten und hebt sich wohltuend vom klobigen Einerlei der VC 20/C 64/C 16-Gehäuse ab. Die Tastatur macht insgesamt einen soliden Eindruck (Bild 1).

Zusätzlich zur normalen Schreibmaschinentastatur gibt es zur schnellen Zahleneingabe einen numerischen Tastenblock mit 14 Tasten.

Der deutsche Zeichensatz wird durch die Umschalttaste ASCII/DIN erreichbar. Die Belegung der Tasten entspricht dann den DIN-Normen. Die Bezeichnungen sind bereits auf den Tastenkappen aufgedruckt. Durch diesen Zeichensatz ergibt sich teilweise eine fünffache Belegung einzelner Tasten. Doch diese Tatsache wird durch speziell auf den deutschen Markt zugeschnittene Textverarbeitungsprogramme und andere professionelle Software gerechtfertigt.

Auffällig sind vier farblich abgesetzte Viererblocks von Tasten oberhalb von Schreibmaschinentastatur und Ziffernblock. Ganz rechts handelt es sich dabei um die von allen Commodore-Heimcomputern bekannten Funktionstasten, die nach dem Einschalten mit Basic-Befehlen belegt sind (Tabelle 1). Im C 64-Modus können die Funktionstasten jedoch aus Gründen der Kompatibilität nur in der gewohnten Art und Weise vom Anwender-Programm abgefragt werden.

Links neben diesem Block finden sich die vier Cursortasten. Die restlichen acht Sondertasten sind mit zum Teil sehr speziellen, aber nützlichen Funktionen belegt.

Die ESC-Taste hat die Bedeutung wie beim C 16/Plus 4. Gefolgt von einer Buchstabentaste führt ESC eine Reihe von Sonderfunktionen wie Bildschirm-Scrollen, zeilenweises Löschen oder Cursor positionieren aus (Tabelle 2).

Die Alt-Taste hat im normalen Betrieb keine Bedeutung, kann aber bei entsprechender Programmierung von Anwenderprogrammen aus benutzt werden, um anderen Tasten eine neue Bedeutung zu geben (ähnlich wie bei der Control-Taste).

Die Line-Feed-Taste wirkt wie »Shift Return« beim C64; der Cursor springt an den Anfang der nächsten Bildschirmzeile, ohne daß ein Befehl ausgeführt würde. Durch Drükken der Taste »No Scroll« wird ein Listing angehalten; bei beliebigem Tastendruck geht's dann weiter.

F1	GRAPHIC	Grafik-Modus ein
F2	DLOAD"	Programm laden
F3	DIRECTORY	Directory anzeigen
F4	SCNCLR	Bildschirm löschen
F5	DSAVE"	Programm speichern
F6	RUN	Programm starten
F7	LIST	Programm listen
F8	MONITOR	Maschinensprache-monitor aktivieren

Tabelle 1. Die Funktionstastenbelegung

Taste	Funktion
A	Insert-Modus ein
B	Untere rechte Ecke eines Windows definieren
C	Insert-Modus aus
D	Bildschirmzeile löschen
E	Cursor-Blinkmodus aus
F	Cursor-Blinkmodus ein
G	Akustisches Signal aus
H	Akustisches Signal ein
I	Neue Bildschirmzeile einfügen
J	Cursor an Zeilenanfang setzen
K	Cursor an Zeilenende setzen
L	Bildschirm-Scrolling ein
M	Bildschirm-Scrolling aus
N	Normal-Modus 80-Zeichen Bildschirm
O	Insert,-Anführungs,- und Invers-Modus aus
P	Bildschirmzeile bis Cursor-Position löschen
Q	Bildschirmzeile ab Cursor-Position löschen
R	Invers-Modus 80-Zeichen-Bildschirm
S	Block-Cursor ein
T	Obere linke Ecke eines Windows definieren
U	Strich-Cursor ein (nur bei 80 Zeichen)
V	Rollt Bildschirm um eine Zeile nach oben
W	Rollt Bildschirm um eine Zeile nach unten
X	Umschaltung 40/80 Zeichen und zurück
Y	Voreingestellte Tabulatorstops setzen
Z	Alle Tabulatorstops löschen
@	Bildschirm ab Cursor-Position löschen

Tabelle 2. Die ESC-Funktionen beim C128

Die Help-Taste ist eine sinnvolle Hilfe bei der Fehlersuche. Falls ein Programm mit Fehlermeldung abbricht, reicht ein Druck auf die Help-Taste, um die fehlerhafte Zeile am Bildschirm aufzulisten. Der Teil der Zeile, in dem der Fehler aufgetreten ist, wird dabei blinkend dargestellt.

Sehr wichtig ist die 40/80 Zeichen-Taste. Wie bei »Shift-Lock« handelt es sich um eine einrastende Taste. Je nachdem, in welcher Stellung sich diese Taste beim Einschalten oder bei einem Reset befindet, geht der C 128 entweder in den 40- oder in den 80 - Zeichen-Modus. Die 80-Zeichen-Darstellung ist natürlich nur im C 128-Modus oder unter CP/M möglich. Sind 80 Zeichen gewählt, dann wird der vom C 64 übernommene VIC II Video Chip, der auch beim C 128 für Bildschirmdarstellung, Grafik und Sprites sorgt, einfach abgeschaltet.

Doch keine Angst, der Bildschirm bleibt nicht dunkel, denn jetzt übernimmt ein vorsorglich eingebauter zweiter Video-Chip mit dem prosaischen Namen 8563 die Kontrolle. Dieser Baustein ist nämlich im Gegensatz zu dem in erster Linie aus Kompatibilität zum C 64 eingebauten VIC II in der Lage, 80 Zeichen pro Zeile zu kontrollieren.

An Anschlüssen verfügt der C 128 über einen User-Port, einen Datasetten-Port, einen Modulatorausgang, einen Audio-Eingang, einen Video-Ausgang, einen digitalen RGB-Ausgang, einen seriellen Port zum Anschluß der Commodore-Peripherie-Geräte sowie über zwei Joystick-Ports (Bilder 2 und 3).

Die ebenfalls neuen Diskettenlaufwerke 1570/1571 wie auch das beim Modell C 128/D integrierte Laufwerk ist in den Betriebsarten C 64 und C 128 kompatibel mit der Floppy 1541. Durch doppelseitige Benutzung der Diskette ergibt sich allerdings bei der 1571 eine Speicherkapazität von maximal 360 KByte (formatiert) pro Diskette. Das Double-Density, Double-Sided (doppelte Aufzeichnungsdichte – beidseitige Aufzeichnung)-CP/M-Format erlaubt eine Speicherkapazität von bis zu 410 KByte. Die Übertragungsrate zwischen Diskettenlaufwerk und C 128 beträgt im C 64-Modus wie gewohnt 300 Byte pro Sekunde. Im C 128-Modus beträgt die Übertragungsrate bereits immerhin 1500 Zeichen pro

Bild 2. Der PC 128 von hinten gesehen. Von rechts nach links: User-Port, RGB-Ausgang, Fernseher, Composite, Video, Serieller Port, Datasetten-Anschluß, Expansions-Port für Steckmodule

Bild 3. Anschlüsse und Schalter an der rechten Seite: Netzteilanschluß, Einschaltknopf, Reset-Taster, Joystick-Ports 2 und 1

Sekunde, liegt also etwa im »Hypra-Load«-Bereich.

Super-Basic 7.0

Beim Basic-Interpreter zeigt sich der C 128 und der C 128D ohne Zweifel von einer seiner stärksten Seiten: Das Basic 7.0 enthält alle Befehle und Funktionen der Basic-Versionen 2.0 (C 64), 3.5 (C 16 und Plus/4) und 4.0 (CBM 80xx). Damit stehen bereits leistungsfähige Grafikbefehle wie DRAW, BOX oder CIRCLE sowie viele Diskettenkommandos zur Verfügung. Doch damit nicht genug. Zusätzlich enthält das 7.0-Basic eine Reihe spezieller Befehle zur Steuerung von Sprites und zur einfachen Programmierung des Synthesizer-Bausteins (SID). Die zusätzlich zum 2.0-Basic vorhandenen Befehle und Funktionen sind in Tabelle 3 beschrieben.

Schon eine erste, oberflächliche Betrachtung dieser Tabelle läßt eine neue Dimension der Basic-Programmierung erahnen. Endlose DATA-Orgien und wüster GOTO-Dschungel gehören mit diesem Basic endgültig der Vergangenheit an. Formatierte Zahlenausgabe mittels PRINT USING ist dabei ebenso selbstverständlich wie Befehle zur Abfrage von Joystick, Lightpen und Paddels.

Mit WINDOW läßt sich ein Bildschirmfenster definieren, auf das sich anschließend alle PRINT- und INPUT-Befehle beziehen. Der Befehl mit dem beziehungsreichen Namen SLEEP läßt den C 128 denn auch tatsächlich für die angegebene Zeit schlafen: »SLEEP 5« hält das Programm fünf Sekunden lang an. So spart man sich das umständliche Hantieren mit leeren FOR...Next-Schleifen für oftmals sinnvolle Verzögerungen im Programmablauf. Zeiten zwischen einer Sekunde und 18 Stunden (!) sind programmierbar, womit sich die Frage aufwirft, wer seinen Computer während eines Programmes wohl für mehr als eine Minute anhalten will.

Eine Reihe von Befehlen dient ausschließlich der bequemeren Programmentwicklung: AUTO gibt bei der Programmeingabe automatisch die Zeilennummern vor, mit TRON kann in der Testphase eines Programmes eine Trace-Funktion eingeschaltet werden. Es werden dann auf dem Bildschirm die Zeilennummern der gerade abgearbeiteten Basic-Zeilen angezeigt. Dies bewährt sich insbesondere bei Fehlern, in der Programmlogik.

AUTO	Automatische Zeilennummerierung
APPEND	Öffnet eine sequentielle Datei zum Datenanfügen
BACKUP	Kopiert eine komplette Diskette
BANK	Wählt Speicherbank für PEEK, POKE und SYS
BEGIN...BEND	Faßt mehrere Basic-Zeilen zu einem Block zusammen
BOOT	Lädt und startet CP/M von Diskette
BOX	Zeichnet Rechtecke
BSAVE	Speichert beliebige Speicherbereiche auf Floppy
BUMP	Liefert bei Sprite-Kollisionen die Sprite-Nummer
CATALOG	Listet Inhaltsverzeichnis der Diskette
CHAR	Fügt Text in hochauflösende Grafik ein
CIRCLE	Zeichnet Kreise, Ellipsen und Vielecke
COLLECT	Löscht offene Dateien und reorganisiert Diskette
COLLISION	Dient zur Sprite-Kollisions-Abfrage
COLOR	Setzt Farben für Text und Grafik
CONCAT	Verbindet zwei sequentielle Dateien miteinander
COPY	Kopiert eine Disketten-Datei
DCLEAR	Schließt alle Kanäle zur Diskettenstation
DCLOSE	Schließt Kanal zur Diskettenstation
DEC	Dezimalwert einer Hexadezimalzahl
DELETE	Löscht einen Zeilenbereich aus dem Programm
DIRECTORY	Disketteninhaltsverzeichnis (wie CATALOG)
DLOAD	Lädt ein Programm von Diskette
DOPEN	Öffnet Kanal zur Diskettenstation
DO...LOOP	Programmschleife LOOP springt immer zu DO zurück.
DRAW	Setzt Punkte und zeichnet Linien
DSAVE	Speichert ein Programm auf Diskette
DS	Ergibt den Fehlerstatus des Diskettenlaufwerks
DS$	Enthält Fehlerstatus der Floppy im Klartext
DVERIFY	Überprüft Programmspeicherung auf Disk
EL	Enthält Zeilennummer bei Auftreten eines Fehlers
ELSE	Alternative bei IF-THEN, falls Bedingung nicht erfüllt
ENVELOPE	Definiert Hüllkurve für Synthesizer
ER	Liefert den Code des zuletzt aufgetretenen Fehlers
ERR$	Liefert Fehlermeldung im Klartext
EXIT	Dient zum Verlassen einer DO...LOOP-Schleife
FAST	Schaltet auf doppelte Geschwindigkeit (2MHz Takt)
FETCH	Holt Daten aus beliebiger Speicherbank (RAM-Floppy)
FILTER	Setzt den Klangfilter-Parameter für den SID
GETKEY	Wartet auf Tastendruck
GO64	Schaltet in den C64-Modus
GRAPHIC	Wählt Grafik-Modus aus
GSHAPE	Schreibt ein Shape aus einem String auf den Bildschirm
HEADER	Dient zum Formatieren von Disketten
HELP	Listet nach Fehlermeldung die Fehlerzeile am Bildschirm auf
HEX$	Wandelt Dezimalzahlen in Hexadezimal-Strings
INSTR	Ergibt Position eines Teilstrings in einem anderen String
JOY	Fragt Joystickposition ab
KEY	Dient zur Belegung der Funktionstasten
LOCATE	Positioniert den Grafik-Cursor
MID$	Ermöglicht jetzt auch Wertzuweisung an Teilstrings
MONITOR	Ruft den eingebauten Maschinensprache-Monitor auf
MOVESPR	Bewegt ein Sprite über den Bildschirm
PAINT	Füllt einen Bereich der hochauflösenden Grafik aus
PEN	Fragt Lightpen ab
PLAY	Spielt die in einem String abgelegte Tonfolge
POINTER	Ergibt die Adresse einer Variablen im Speicher
POT	Fragt Paddles ab
PRINT USING	Erlaubt formatierte Zahlenausgabe
PUDEF	Definiert Steuerzeichen für PRINT USING.
RCLR	Liefert gewählten Farbcode für Text und Grafik
RECORD	Positioniert Schreib-/Lesezeiger bei relativen Dateien
RENAME	Dient zum Umbenennen von Diskettendateien
RENUMBER	Numeriert das Basic-Programm neu
RESTORE	Setzt DATA-Zeiger auf beliebige Zeilennummer
RESUME	Rückkehr aus einer Fehlerbehandlungsroutine
RGR	Liefert die Nummer des eingestellten Grafikmodus
RREG	Weist Variablen die Werte der Prozessorregister zu
RSPRCOLOR	Liefert den aktuellen Code des Mehrfarben-

	modus für Sprites
RSPPOS	Liefert Position und Geschwindigkeit eines Sprites
RSPRITE	Ergibt je nach Parameter alle Sprite-Attribute
RWINDOW	Liefert Parameter des eingestellten Bildschirmfensters
SCALE	Maßstabswahl bei hochauflösender Grafik
SCNCLR	Löscht Text- oder Grafikbildschirm
SCRATCH	Löscht eine Diskettendatei
SSHAPE	Speichert ein Shape in eine Stringvariable
SLEEP	Hält die Programmausführung für eine wählbare Zeit an
SLOW	Schaltet von 2 MHz auf 1 MHz Takt zurück
SOUND	Erzeugt Toneffekte mit wählbarer Frequenz und Dauer
SPRCOLOR	Setzt Mehrfarben-Modus-Farben für Sprites
SPRDEF	Ruft den integrierten Sprite-Editor auf
SPRITE	Setzt Sprite-Attribute
SPRSAV	Speichert ein Sprite in einem String oder umgekehrt
STASH	Überträgt Daten in eine Speicherbank (RAM-Floppy)
SWAP	Tauscht Daten zwischen zwei Speicherbänken aus
TEMPO	Setzt Abspieltempo für PLAY-Anweisung
TRAP	Verzweigt im Fehlerfall zu einer Fehlerbehandlungsroutine
TROFF	Schaltet Programmablaufverfolgung (Trace) aus
TRON	Schaltet Trace ein
UNTIL	Setzt Bedingung für DO...LOOP fest (DO UNTIL...)
VOL	Setzt Lautstärke für die SOUND-Anweisung
WHILE	Setzt Bedingung für DO...LOOP fest (DO WHILE..)
WIDTH	Setzt die Strichstärke für alle Grafikbefehle
WINDOW	Definiert ein Bildschirmfenster
XOR	Liefert die Exklusiv-Oder-Verknüpfung zweier Werte

Tabelle 3. Diese Befehle sind im Basic 7.0 dazugekommen. Die Befehle von Basic 2.0 (C64/VC 20) sind nicht aufgeführt, aber dennoch voll im Basic 7.0 integriert.

Eine falsch gesetzte IF-Abfrage wird damit zum Beispiel schnell erkannt – man sieht ja, wohin das Programm springt. Zu Testzwecken kann TRON natürlich auch im Programm verwendet werden. Am Anfang eines »verdächtigen« Programmteils fügt man einfach den TRON-Befehl ein, am Ende dieses Abschnittes wird die Trace-Funktion mit TROFF wieder außer Betrieb gesetzt.

Der RENUMBER-Befehl dient zum Umnumerieren des gesamten Programms oder auch nur einzelner Teile davon. Während jedoch RENUMBER beim bekannten Simons-Basic für den C 64 weder GOTO- noch GOSUB-Adressen ändert (und mithin eher ein Problem als ein Hilfsmittel darstellt), korrigiert das 7.0-Basic automatisch alle Zeilennummern hinter GOTO, GOSUB, THEN, ELSE, RESTORE und RESUME und sogar bei Abfragen von Fehlerzeilen mittels der Spezialvariablen EL in einer Fehlerbehandlungsroutine. Wobei wir gleich bei einem weiteren interessanten Aspekt des 7.0-Basic wären.

Fehlerbehandlung ohne Programmabbruch

Während der C64 bei jedem auftretenden Fehler unerbittlich sein Programm mit einer entsprechenden Meldung beendet, bietet der C128 hier einiges mehr an Flexibilität. Mit der TRAP-Anweisung können alle auftretenden Fehler während des Programmablaufs abgefangen werden. Zum Beispiel wird nach der Anweisung »TRAP 500« beim Auftreten eines Fehlers das Programm nicht unterbrochen, sondern es wird in eine Fehlerbehandlungsroutine (hier ab Zeile 500) verzweigt. Alle wichtigen Daten über den Fehler werden in Systemvariablen gespeichert und können von der (vom Programmierer zu schreibenden) Basic-Routine ab Zeile 500 ausgewertet werden: EL enthält die Zeilennummer, in der der Fehler auftrat, ER enthält die Fehlernummer und ERR$ liefert die Fehlermeldung im Klartext. Die Fehlerbehandlungsroutine kann diese Variablen auswerten, um gezielte Maßnahmen zu ergreifen. Anschließend sollte das Programm natürlich weiter fortgesetzt werden können. Dazu dient die RESUME-Anweisung, die eine Fehlerbehandlung abschließt (vergleichbar mit RETURN bei Unterprogrammen). RESUME kann auf drei verschiedene Arten verwendet werden. RESUME ohne weitere Parameter kehrt zu der Anweisung zurück, die den Fehler verursacht hat und setzt das Programm dort ganz normal fort. In diesem Falle muß natürlich in der Fehlerbehandlungsroutine die Fehlerursache behoben worden sein, sonst tritt der Fehler sofort wieder auf. Ein gutes Beispiel ist der Test, ob der Drucker eingeschaltet ist:

```
10 TRAP 90 : OPEN 1,4
20 PRINT#1,"DRUCKER OK"
30 END
90 IF ER=5 AND EL=10 THEN PRINT
"BITTE DRUCKER EINSCHALTEN UND
TASTE DRUECKEN" : GETKEY A$
95 RESUME
```

Dieses kleine Demo-Programm gibt den Text »Drucker OK« auf einem angeschlossenen Drucker aus. Falls der Drucker nicht eingeschaltet sein sollte, würde der OPEN-Befehl in Zeile 10 normalerweise zur Fehlermeldung »Device not present« führen. Diese Meldung wird aber durch den TRAP-Befehl im Falle eines Falles abgefangen und statt dessen zur Zeile 90 verzweigt, wo nach Überprüfung auf Fehlernummer und -zeile der Benutzer höflich aufgefordert wird, doch bitteschön den Drucker einzuschalten. Der Befehl GETKEY wartet anschließend auf einen Tastendruck, worauf das Programm durch den RESUME-Befehl wieder zum OPEN-Kommando zurückkehrt.

Soll das zum Fehler führende Kommando nicht nochmals ausgeführt werden, dann muß die Fehlerbehandlungsroutine mit RESUME NEXT abgeschlossen werden, wodurch mit dem nächsten Befehl nach der Fehlerursache weitergemacht wird. In besonderen Fällen kann es nach einem Fehler nützlich sein, ganz woanders im Programm fortzufahren. In einem solchen Falle kann hinter RESUME eine Zeilennummer angegeben werden, an der das Programm fortgesetzt werden soll.

Mit diesen Möglichkeiten zur Fehlerbehandlung im Programm selbst steht dem Programmierer ein leistungsfähiges Werkzeug zur Verfügung. Und sollte in der Entwicklungsphase eines Programmes doch einmal ein Fehler auftreten, dann genügt ein Druck auf die HELP-Taste, um die fehlerhafte Zeile aufzulisten. Der Teil der Zeile, der den Fehler

verursacht hat, wird dabei revers dargestellt.

Natürlich lassen sich auch von der Diskettenstation gemeldete Fehler in ähnlicher eleganter Weise abfragen. Statt umständlich die Zeile 1 OPEN 1,8,15: INPUT#1,A,B$,C,D: PRINT A,B$, C,D : CLOSE 1: END einzugeben (und dabei womöglich sein Programm zu überschreiben) tippt man beim 7.0-Basic einfach »?DS$« und erhält die gleiche Meldung. Die Systemvariable DS$ enthält nämlich den Fehlerstatus der Diskettenstation als Klartext, die Systemvariable DS den entsprechenden Fehlercode.

Überhaupt stehen beim C 128 alle Diskettenbefehle als Basic-Kommandos zur Verfügung. SCRATCH beispielsweise löscht ein File von der Diskette, DIRECTORY oder CATALOG listen das Inhaltsverzeichnis ohne Programmverlust, mit DLOAD, DSAVE und DVERIFY spart man sich das lästige »,8«. BLOAD und BSAVE dienen zum Laden/Speichern beliebiger Speicherinhalte (Maschinenprogramme, Grafik etc.). Neu sind auch eine Reihe von Befehlen zur komfortablen Verwaltung sequentieller und relativer Dateien. Mit RECORD kann beispielsweise direkt auf einen Datensatz einer relativen Datei zugegriffen werden, APPEND ermöglicht das Anfügen weiterer Datensätze bei sequentiellen Dateien.

Programmieren ohne GOTO

Zwei weitere ungewöhnliche Befehle fallen sofort auf, nämlich SLOW und FAST. Mit diesen Befehlen kann der C 128 zwischen 1 MHz Taktfrequenz (SLOW) und 2 MHz umgeschaltet werden. Nach dem Einschalten läuft der Computer mit einem Takt von 1 MHz, also mit ähnlicher Geschwindigkeit wie der C64. Durch die komplizierte Art der Speicherverwaltung mit den verschiedenen Speicherbänken für Programme, Variablen und Betriebssystem/Basic ist das C 128-Basic prinzipiell geringfügig langsamer als das C 64-Basic. Dies wird jedoch, wie schon erwähnt, einerseits durch den wesentlich leistungsfähigeren Befehlssatz mehr als aufgewogen, zum anderen kann durch den FAST-Befehl die Abarbeitungsgeschwindigkeit exakt verdoppelt werden.

So schön das im Pinzip auch ist, die Geschwindigkeitsvorteile des FAST-Modus muß man sich mit dem bereits erwähnten Nachteil erkaufen.

Das 7.0 Basic bietet eine ganze Reihe spezieller Schleifen- und Strukturbefehle zur GOTO-freien, strukturierten Programmierung. Da wäre zunächst einmal die Erweiterung der IF...THEN-Abfrage um die ELSE-Klausel. Bisher mußte man beispielsweise alternative Entscheidungen wie folgt programmieren:

```
10 IF A$="N" THEN PRINT "NEIN" :
GOTO 30
20 PRINT "JA"
30 REM HIER GEHT'S WEITER.
```

Im 7.0 Basic reicht dazu eine Zeile, und die ist noch um einiges leichter verständlich:

```
10 IF A$="NEIN" THEN PRINT
"NEIN":ELSE PRINT "JA"
```

Wenn A$ gleich »N« ist, dann wird »nein« gedruckt, sonst »ja«.

Leider ist die ELSE-Anweisung in dieser Form auf eine Zeile beschränkt. Abhilfe schafft hier die Klammerung mit BEGIN... BEND.

Alle zwischen BEGIN und BEND stehenden Basic-Zeilen stellen einen Block dar, der vom Basic-Interpreter genauso wie eine einzelne Zeile behandelt wird. Deshalb wird BEGIN...BEND besonders vorteilhaft bei IF-Abfragen benutzt:

```
10 INPUT "HEISST DEIN COMPUTER
COMMODORE ODER SCHNEIDER?";C$
20 IF C$="COMMODORE" THEN BEGIN
30:     PRINT "C 128 KAUFEN!"
40: BEND: ELSE BEGIN
50:     PRINT"VERRÄTER!"
60 BEND
```

Man beachte, daß sich die IF-Anweisung insgesamt von Zeile 20 bis Zeile 60 erstreckt. In diesem Beispiel erhält man den Ratschlag, sich einen C 128 zu kaufen, falls der Computer »Commodore« heißt. Hat man jedoch »Schneider» (oder etwas anderes) als Namen angegeben, wird man sofort als »Verräter« tituliert.

Natürlich können derartige IF...THEN...ELSE-Abfragen mit BEGIN...BEND auch geschachtelt werden, das heißt, man kann sowohl in den THEN- als auch in den ELSE-Teil weitere IF-Abfragen einbauen.

Somit lassen sich auch größere Programmblöcke ohne GOTO programmieren. Der Verzicht auf GOTO erhöht nicht nur die Übersichtlichkeit, sondern auch die Geschwindigkeit beim Programmlauf. Bei jedem GOTO-Befehl muß der Basic-Interpreter nämlich erstens die Zeilennummer, die im Programm ja als Dezimalzahl steht, in das interne binäre Format umrechnen und zweitens dann auch noch die angegebene Zeile suchen. Ein weiterer Vorteil: In den Programmbefehlen selbst kommen keine weiteren Zeilennummern mehr vor, das Beispielprogramm kann unverändert in allen möglichen Zeilenbereichen laufen.

Aber nicht nur Verzweigungen lassen sich derart elegant programmieren, besonders bei Schleifen, also bei Wiederholungen von bestimmten Programmteilen, spielt das 7.0 Basic seine Stärken erst richtig aus. Es ist ja vom C64 her bekannt, daß eine FOR...NEXT-Schleife um einiges schneller ist als die gleiche Schleife mittels IF und GOTO programmiert. Nachteilig bei der FOR...Next Schleife ist, daß die Anzahl der Schleifendurchläufe schon bei Eintritt in die Schleife bekannt sein muß. Dieser Nachteil wird durch die neue, schnelle DO...LOOP-Schleifenstruktur behoben. Wie FOR...NEXT umklammert auch DO...LOOP einen beliebig großen Programmteil. Die Wirkung des DO-Befehls besteht einfach darin, daß der Basic-Interpreter sich den Anfang der Schleife »merkt«. Bei Erreichen des zugehörigen LOOP wird dann sehr schnell, ohne Suchzeiten, zu DO zurückgesprungen. Es ergibt sich also eine »unendliche Schleife« zwischen DO und LOOP. Um diese Schleife dennoch verlassen zu können, ist der EXIT-Befehl vorgesehen. Die Wirkung von EXIT besteht einfach darin, die Programmausführung hinter LOOP ganz normal fortzusetzen. Normalerweise wird EXIT daher von einer Bedingung abhängig gemacht. Beispiel:

```
10 X=1
20 DO
30 : X=X*2 : PRINT X
40 : IF X > 1500 THEN EXIT
50 LOOP
```

Der Wert X wird hier solange verdoppelt und ausgedruckt, bis der Wert 1500 überschritten wird. Neben dieser unbedingten DO...LOOP-Schleife sind noch zwei von Bedingungen abhängige Formen vorgesehen. DO WHILE... LOOP wird so lange ausgeführt, wie eine nach WHILE stehende Bedingung wahr ist:

```
10 DO WHILE A$="":GET A$:LOOP
```

Solange keine Taste gedrückt wird, ist A$ immer leer, die WHILE-Bedingung also erfüllt. Die Schleife wird daher erst verlassen, wenn eine Taste gedrückt wird.

Die DO UNTIL-Schleife wird dagegen nicht ausgeführt, solange die Bedingung wahr ist, sondern im Gegenteil so lange, bis die hinter UNTIL angegebene Bedingung wahr wird. Natürlich können auch bei DO WHILE oder DO UNTIL zusätzliche EXITs in die Schleife eingebaut werden, was die Leistungsfähigkeit dieser Anweisung noch erhöht.

Die Grafik ist für alle da

Um hochauflösende Grafik auf dem C64 zu realisieren, gibt es außer dem Kauf diverser Basic-Erweiterungen (oder dem Abtippen von 64'er-Listings) im wesentlichen nur die Alternative, selbst zum Maschinensprache-Profi zu werden – ungefähr so, als wenn man Radio- und Fernsehmechaniker werden müßte, um an seinem Farbfernseher die Farbe einstellen zu können. Ein sicherlich unhaltbarer Zustand, dessen Änderung Commodore allerdings bereits mit dem 3.5-Basic des C 16 in Angriff genommen hatte. Die hochauflösende Grafik des C 128 ist genauso wie die des C 64/C 16 aufgebaut. Insgesamt 64000 Einzelpunkte können getrennt angesprochen werden, was einer Auflösung von 320 x 200 Punkten entspricht. Daneben ist ein Mehrfarbenmodus mit einer Auflösung von 160 x 200 Punkten vorgesehen, bei dem jeder Einzelpunkt eine von vier Farben haben kann.

Der große Unterschied zum C 64 liegt darin, daß die C 128-Grafik voll vom Basic unterstützt wird. Befehle wie DRAW, BOX oder CIRCLE ermöglichen schnelles und unkompliziertes Zeichnen geometrischer Figuren von Linien über Drei-, Vier- und Mehrecken bis hin zu Kreisen und Ellipsen. Alle Figuren können beliebig vergrößert, verkleinert und sogar gedreht oder ausschnittsweise dargestellt werden. PAINT füllt geschlossene Flächen aus, SCALE dient zur Skalierung der Zeichenfläche und SCNCLR löscht den Grafikbildschirm.

Alle Grafikbefehle arbeiten sowohl im Hochauflösungs- wie auch im Mehrfarben-Modus. Mit dem Befehl GRAPHIC wird der gewünschte Grafik-Modus eingestellt. Zur Wahl stehen Text mit 40 Zeichen pro Zeile, Hochauflösung, Hochauflösung mit Textfenster, Mehrfarbengrafik mit Textfenster und schließlich Text mit 80 Zeichen pro Zeile. Leider beziehen sich alle Grafikbefehle des 7.0 Basic ausschließlich auf die vom C64 her bekannte 320 x 200 Punkte-Auflösung (und natürlich wahlweise auf den Mehrfarbenmodus mit 160 x 200 Punkten).

Doppelte Auflösung (640 x 200 Punkte) ist zwar im 80-Zeichen-Modus möglich, wird aber weder vom Basic noch vom Betriebssystem unterstützt. Dies ist um so ärgerlicher, als sich diese Fähigkeit mit wenig Aufwand hätte realisieren lassen. Das 64'er-Magazin brachte eine kleine Basic-Erweiterung zu diesem Thema in der Ausgabe 12/85 und zeigte Commodore damit, wie's geht. Wegen der großen Nachfrage finden Sie diese interessante Grafik-Erweiterung auch in diesem Sonderheft.

Ein weiterer Wermutstropfen: Die ganze schöne Grafik, Sprites und 40-Zeichen-Text sind ausschließlich über einen Composite-Monitor verfügbar, auf einem RGB-Monitor tut sich überhaupt nichts. Andersherum ist die 80-Zeichen-Textdarstellung nur über RGB (oder natürlich einen monochromen Monitor) möglich.

Der verblüffte Anwender stellt spätestens jetzt fest, daß er einfach einen Monitor zu wenig hat. Damit dürfte Commodore sich die Urheberrechte am ersten Zwei-Monitor-Heimcomputer der Welt gesichert haben. Wohlgemerkt, man hat nicht die Wahl zwischen Composite und RGB, sondern braucht unbedingt einen Composite-Monitor für 40-Zeichen, Grafik und Sprites und ebenso unbedingt entweder einen RGB- oder einen SW-Monitor für 80 Zeichen.

Abhilfe schafft hier der neue 1901-Monitor von Commodore (Bild 7), der speziell zum C 128 entwickelt wurde und sowohl über einen Composite- als auch über einen RGB-Eingang verfügt. Zwischen beiden Betriebsarten des Monitors wird mit einem kleinen Schalter an der Frontseite hin- und hergeschaltet – eine softwaremäßige Umschaltung ist nicht vorgesehen. Man kann daher nur wünschen, daß der Umschalter stabil genug gebaut ist – er wird oft betätigt werden müssen.

Als Fazit zur C 128-Grafik bleibt festzuhalten, daß sie von der Auflösung her dem durch den C64 gesetzten Standard (320 x 200 Punkte) entspricht. Neu ist allerdings die vorbildliche Unterstützung durch das 7.0 Basic.

Wenn von Grafik die Rede ist, dürfen natürlich Shapes und Sprites nicht fehlen. Hinsichtlich dieser beweglichen Grafikobjekte ist beim C 128 eine gelungene Synthese von C 64-Hardware und C 16 Software zu verzeichnen. Vom C 64 stammen die acht Sprites, frei programmierbare, bewegliche Grafikobjekte, die von der Hardware (VIC-Chip) erzeugt und in den Bildschirm eingeblendet werden. Sprites können sowohl im 40-Zeichen-Textmodus als auch in den verschiedenen Grafik-Modi erzeugt werden, nicht allerdings im 80-Zeichen-Modus, denn der VIC, der sie erzeugt, ist nicht RGB-fähig.

Shapes und Sprites

Aus dem 3.5 Basic des C 16 wurde das Konzept der softwaremäßig erzeugten Shapes übernommen. Shapes sind rechteckige Ausschnitte aus der hochauflösenden oder der Mehrfarben-Grafik, die in Stringvariablen abgespeichert werden und daraus auch wieder auf den Bildschirm gebracht werden können. Da es sich um reine Grafikelemente handelt, können sie weder im 40- noch im 80-Zeichen-, sondern nur im Grafik-Modus dargestellt werden. Mit »SSHAPE X$,100,100,150,120« wird beispielsweise der Inhalt des Rechtecks mit linker oberer Ecke (100,100) und rechter unterer Ecke (150,120) aus der hochauflösenden Grafik in der Stringvariablen A$ abgelegt. Mit »GSHAPE A$,X,Y« wird die in A$ enthaltene Grafik-Information an der Grafikposition X,Y wieder auf den Bildschirm gebracht. Neben den Sprites sind die Shapes also eine zweite leistungsfähige Möglichkeit zur Darstellung grafischer Objekte und eröffnen in Zusammenhang mit der hohen Speicherkapazität des C 128 völlig neue Möglichkeiten für Spiele in hochauflösender Grafik.

Integrierter Sprite-Editor

Das Basic 7.0 enthält sogar einen integrierten Sprite-Editor, mit dem man direkt am Bildschirm das Punktmuster des gewünschten Sprites entwerfen kann. Mit dem SPRITE-Befehl werden für jedes Sprite folgende Attribute gesetzt: Aktivität, Farbe, Priorität, Dehnung in X-und Y-Richtung und Modus (hochauflösend oder Mehrfarben).

Mit «SPRITE 4,1,6,1,1,0,0« wird zum

Beispiel das Sprite Nr.4 aktiviert (1). Es wird in der Farbe Grün (6) angezeigt, hat Priorität über bereits angezeigte Bildschirmdaten (1), ist in X-Richtung gedehnt (1), in Y-Richtung nicht gedehnt (0) und wird im Hochauflösungs-Modus angezeigt (1).

Um umgekehrt die Attribute eines bereits definierten Sprites zu bestimmen, kann die RSPRITE-Funktion verwendet werden.

Mit dem SPRSAV-Kommando können die Daten eines Sprites in einer Stringvariablen abgelegt werden oder umgekehrt aus einem String ausgelesen werden.

Die Steuerung der Sprites erfolgt über den MOVESPR-Befehl mit dem ein Sprite an eine bestimmte Bildschirmposition gesetzt werden kann. Die Positionsangabe kann entweder in absoluten Koordinaten oder auch relativ zur bisherigen Position erfolgen. Doch damit noch nicht genug. Gibt man zusätzlich noch eine Geschwindigkeit als Zahlenwert zwischen 1 und 15 an, so gleitet das Sprite automatisch an die angegebene neue Position. Durch Setzen von Plus- oder Minuszeichen vor die Koordinatenangaben werden aus den absoluten Koordinaten relative Koordinaten. Ohne Geschwindigkeitsangabe erscheint das Sprite sofort an der neuen Position. »MOVESPR 7,--30,+40« versetzt Sprite 7 augenblicklich um 30 Punkte nach links und um 40 Punkte nach oben. Beim C64 kann man durch PEEKen in die Sprite-Kollisionsregister des VIC feststellen, ob ein Sprite mit einem anderen Sprite oder mit Hintergrunddaten kollidiert ist. Beim C128 bedient man sich für den gleichen Zweck um einiges eleganter der COLLISION-Anweisung. Damit kann eine automatische Programmunterbrechung bei Eintritt entweder einer Sprite/Sprite oder einer Sprite/Hintergrund-Kollision programmiert werden. »COLLISION 1,500« hat beispielsweise folgende Bedeutung: Falls im weiteren Verlauf des Programms eine Sprite-Sprite-Kollision (Kennziffer 1) auftritt, dann wird das laufende Basic-Programm unterbrochen, und es wird ein Unterprogramm ab Zeile 500 ausgeführt. Nach dem RETURN wird das Programm an der Unterbrechungsstelle fortgesetzt.

COLLISION und MOVESPR sind leistungsstarke Befehle, die ein Basic-Programm hinsichtlich der Sprite-Steuerung sehr stark entlasten. Um ein Sprite quer über den Bildschirm zu bewegen, muß man beim C64 noch mit einer FOR...NEXT-Schleife arbeiten; um Kollisionen festzustellen, war daneben noch ein ständiges PEEKen in die Kollisionsregister des VIC nötig.

Beim C128 reichen zwei Basic-Befehle, die zudem noch interruptgesteuert arbeiten, so daß das Basic-Programm während der Bewegung der Sprites weiterlaufen kann, bei einer eventuell auftretenden Sprite-Kollision dagegen automatisch unterbrochen wird, um schnell darauf reagieren zu können.

COLLISION dient nicht nur zur Sprite-Sprite und Sprite-Hintergrund-Kollisionsabfrage, sondern nebenbei auch noch zur Kontrolle eines Lichtgriffels, neuhochdeutsch auch als Lightpen bezeichnet. Um festzustellen, welches Sprite eine Kollision ausgelöst hat, verwendet man die BUMP-Funktion.

Zusammen mit vielen weiteren Befehlen zur Sprite- und Shape-Kontrolle sowie für hochauflösende Grafik ergeben sich natürlich ungeahnte Möglichkeiten für den Basic-Programmierer. Aber das ist nicht nur bei der Grafikprogrammierung der Fall.

Musikalisches Basic

Ein ähnlicher Komfort ist auch bei der Programmierung des aus dem C64 übernommenen Synthesizer-Bausteins, des SID, zu finden. Alle Musik-Parameter müssen nicht mehr aus DATA-Wüsten in den SID hineingePOKEt werden, sondern können elegant und leichtverständlich per Basic-Befehl gesetzt werden.

VOL regelt zum Beispiel die Lautstärke, mit dem SOUND-Kommando wird einer der Tongeneratoren gestartet. Dabei kann über entsprechende Parameter nicht nur die Frequenz, sondern auch die Dauer des Tones sowie das An- und Abschwellen festgelegt werden.

Mit ENVELOPE wird jeweils eine von zehn möglichen Tonhüllkurven für Musikinstrumente definiert. Attack, Decay, Sustain, Release werden damit ebenso festgelegt wie Wellenform und Impulsbreite. Jede der zehn möglichen Hüllkurven bleibt gespeichert, bis sie durch einen weiteren ENVELOPE-Befehl zur gleichen Hüllkurvennummer überschrieben wird.

Nach dem Einschalten des C128 sind bereits alle zehn Hüllkurven mit der Klangstruktur verschiedener Musikinstrumente vordefiniert: Klavier, Akkordeon, Zirkusorgel, Trommel, Flöte, Gitarre, Cembalo, Orgel, Trompete und Xylophon. Damit steht auch dem musikalisch wenig bewanderten Einsteiger sofort eine Fülle einfach anwendbarer Klangeffekte zur Verfügung. Mit der FILTER-Anweisung können zudem alle Filtermöglichkeiten des SID zur Klangverfremdung ausgeschöpft werden.

Musik per Warteschlange

Der Play-Befehl ermöglicht das automatische Abspielen von in Strings gespeicherten Musiknoten. In dem als Parameter angegebenen String können Informationen über Hüllkurve, Oktave, Lautstärke, Tonkanal und Filter enthalten sein, in der Hauptsache aber natürlich die zu spielenden Noten. Die Noten werden einfach durch Angabe des Notennamens (A,B,C,D,E,F,G) ausgewählt, wobei das B der in Deutschland üblichen Notenbezeichnung H entspricht. Natürlich können die einzelnen Noten um Halbtöne erhöht oder erniedrigt werden, es sind ganze, halbe, viertel, achtel und sechzehntel Noten, jeweils auch punktiert, möglich.

Auch der PLAY-Befehl wird interruptgesteuert ausgeführt, das heißt die zu spielenden Noten gelangen in eine Ton-Warteschlange, was nichts anderes bedeutet, daß sie in einem reservierten Speicherbereich abgelegt werden. Während des Interrupts stellt das Betriebssystem fest, ob Tondaten in der Warteschlange stehen. Wenn ja, wird der erste Wert aus der Schlange geholt und, vereinfacht gesprochen, an den Synthesizer-Chip (SID) zum Abspielen übergeben. Alle anderen in der Warteschlange stehenden Tondaten rücken jetzt einen Platz vor. Bei jedem weiteren Interrupt wird überprüft, ob die vorgesehene Tondauer bereits erreicht ist. Wenn dies schließlich der Fall ist, wird wieder nach wartenden Tondaten Ausschau gehalten, und das ganze Spiel setzt sich fort.

Wie gesagt, läuft dies alles jeweils während des System-Interrupts ab. Das Basic selbst »merkt« davon nichts. Es stellt nur fest, ob noch Platz in der Tonwarteschlange ist oder nicht. Falls noch Plätze frei sind, können weitere Tondaten angefügt werden und das Programm fährt anschließend normal fort, während die Musik automatisch abgespielt wird. Nur dann,

wenn zuviele Töne zum Abspielen anstehen, muß das Programm tatsächlich anhalten und warten, bis wieder Plätze in der Warteschlange frei geworden sind.

Obwohl Dank des leistungsstarken Basics nur selten nötig, gibt es natürlich auch beim C128 den Zugriff auf die Maschinenebene. Allerdings ist es hier nicht einfach mit POKE, PEEK und SYS getan. Vielmehr ergibt sich aus dem Konzept der verschiedenen Speicherbereiche, die mittels Bank-Switching umgeschaltet werden, das Problem, in welche Speicherbank der POKE-, PEEK- oder SYS-Befehl gehen soll. Das C128-Basic löst dieses Problem ebenso einfach wie elegant: Mit dem BANK-Befehl kann die gewünschte Speicherbank ausgewählt werden. Damit erfolgt natürlich nicht wirklich eine Bankumschaltung (während ein Basic-Programm läuft, muß natürlich immer der Basic-Interpreter eingeschaltet sein), sondern das Basic merkt sich nur, in welcher Speicherbank beispielsweise ein POKE-Wert abgelegt werden muß, oder aus welcher Bank die Daten für PEEK stammen müssen.

Die Verbindung zur Maschinensprache

So kann man nach Belieben entweder in den Programm- oder in den Variablenspeicher POKEn und PEEKen. Betriebssystem- und Basic 7.0-Routinen können nach »BANK 15« einfach mit SYS aufgerufen werden.

Daneben gibt es noch die Möglichkeit, von Basic aus ganze Speicherbereiche zwischen Bank 1 (Basic-Arbeitsspeicher) und anderen Speicherbänken hin- und herzuladen. Hierzu dienen die Befehle FETCH; STASH und SWAP. »FETCH 2000, 50000,4,35000« holt beispielsweise 2000 Byte ab Adresse 35000 aus Speicherbank 4 und legt diese ab Adresse 50000 im Basic-Arbeitsspeicher (immer Bank 1) ab. STASH ist die Umkehrfunktion zu FETCH: Es wird eine Anzahl Bytes aus dem Arbeitsspeicher in eine andere Speicherbank gebracht. SWAP schließlich tauscht die angegebenen Speicherbereiche in beiden Bänken gegeneinander aus.

Diese drei Befehle sind hauptsächlich für den Einsatz im Zusammenhang mit Speichererweiterungen (RAM-Floppy) gedacht.

Es können damit Datenmengen verwaltet werden, die ein mehrfaches von 64 KByte im Speicher belegen, und das mit Geschwindigkeiten, wie sie mit einer Floppy niemals zu realisieren sind.

Maschinensprache-Monitor eingebaut

Wem trotz allem die Möglichkeiten des 7.0 Basic noch nicht reichen, der kann mit dem MONITOR-Kommando das Basic verlassen und landet im fest im ROM eingebauten Maschinensprachemonitor. (Tabelle 4 zeigt den kompletten Monitor-Befehlssatz).

Befehl	Bedeutung
A(assemble)	Assembliert eine Zeile im 6502-Befehlscode.
C(compare)	Vergleicht zwei Speicherbereiche byteweise und zeigt Unterschiede an
D(disassemble)	Disassembliert einen Speicherbereich mit Maschinensprache-Code.
F(fill)	Füllt einen Speicherbereich mit einem angegebenen Byte.
G(go)	Startet ein Maschinensprache-Programm bei einer angegebenen Adresse.
H(hunt)	Durchsucht einen Speicherbereich nach einer angegebenen Bytefolge.
L(load)	Lädt eine Programmdatei von Kassette oder Diskette.
M(memory)	Zeigt den Inhalt eines angegebenen Speicherbereiches hexadezimal an.
R(register)	Zeigt den Inhalt der Prozessorregister an.
S(save)	Speichert den angegebenen Speicherbereich auf Kassette oder Diskette.
T(transfer)	Verschiebt den Inhalt eines angegebenen Speicherbereiches.
V(verify)	Vergleicht einen Speicherbereich mit einer Programmdatei auf Kassette oder Diskette.
X(exit)	Beendet den Monitor und kehrt in den Basic-Direktmodus zurück.
>	Modifiziert ein bis acht Speicherzellen ab der angegebenen Adresse.
.	Identisch mit dem A-Befehl
.	Modifiziert die Prozessorregister.
@	Zeigt den Floppy-Disk-Status an oder überträgt einen Floppy-Disk-Befehl.

Tabelle 4. Der Befehlssatz des eingebauten Maschinensprache-Monitors.

Dieser dem C16-»Tedmon« nachempfundene Monitor enthält neben den üblichen Funktionen zum Listen und Beschreiben des Speichers und einem Disassembler auch einen kleinen Assembler, mit dem Maschinenprogramme sehr komfortabel eingegeben werden können. Statt der sonst üblichen vierstelligen hexadezimalen Adresseneingabe verlangt dieser Monitor allerdings fünf Stellen:

Die erste Stelle gibt an, welche von 16 möglichen Speicherbänken ausgewählt werden soll. Allerdings sind in der Grundversion des C128 natürlich nicht alle 16 Bänke belegt, einige sind für ROM-Module, andere für die RAM-Floppy reserviert.

Der gesamte RAM-Bereich des C128 umfaßt 128 KByte, aufgeteilt in zwei Banks zu je 64 KByte. RAM-Bank 1 enthält Zeropage, Stackbereich und Bildschirmspeicher; 60 KByte stehen nur für das Basic-Programm zur Verfügung. Der Speicherbereich für die Variablen befindet sich in RAM-Bank 2 und umfaßt 62 KByte. Doch damit noch nicht genug. In Stufen zu je 128 KByte kann der RAM-Bereich auf bis zu 512 KByte erweitert werden, wobei der zusätzliche Speicher als RAM-Disk angesprochen wird.

Der erreichbare RAM-Speicher in Zusammenhang mit dem wahlweise ladbaren CP/M 3.0-Betriebssystem beträgt ebenfalls 128 KByte. Auch hier läßt sich der Speicher mit der externen RAM-Disk-Option auf insgesamt 512 KByte erweitern.

Das Betriebssystem des C128 belegt 16 KByte ROM für sich allein, das sehr umfangreiche Basic 7.0 ist in 32 KByte ROM untergebracht. Zu diesen 48 KByte ROM kommen noch weitere 16 KByte, die das Basic und das Betriebssystem des C64 enthalten und die nur in der C64-Betriebsart aktiv sind.

C64-Steckmodule passen auch in den Modul-Port des C128. Spezielle C128-Steckmodule mit einer Kapazität bis zu 32 KByte können natürlich ebenfalls eingesteckt werden. Stellt das Betriebssystem nach dem Einschalten fest, daß ein C64 Modul eingesteckt ist, dann wird automatisch zur entspechenden Betriebsart übergegangen und das Modul-Programm gestartet. Normalerweise gelangt man vom C128

aber über den »GO 64«-Befehl in den C64-Modus.

Zur Verwaltung der diversen Speicherbänke bedient sich der C128 des sogenannten »Bank-Switching«. Bank Switching heißt soviel wie Speicherblock-Umschaltung. Die 128 KByte Speicher des C128 werden dazu in zwei Teile mit je 64 KByte gespalten. Mit einem Trick wird dafür gesorgt, daß der Prozessor abwechselnd die eine oder die andere 64-KByte-Bank »sieht«. Der Trick heißt Memory Management Unit (MMU). Wie der Name schon sagt, managt diese Schaltung die Speicherkonfiguration. Die MMU bestimmt, auf welche RAM-Bank der Prozessor »sehen« darf, also wo Schreib-/Lesezugriffe im Speicher erfolgen sollen. Aber nicht nur das. Die MMU gibt auch die ROM-Konfiguration an, sie sagt also dem Prozessor, aus welchem ROM er seine Befehle zu holen hat. Entweder aus dem Kernel oder aus einem EPROM einer Erweiterungskarte.

Im C128 wird der Basic-Speicher so verwaltet, daß die RAM-Bank 0 (64 KByte) für Basic-Programme und Bank 1 (64 KByte) für Basic-Variable reserviert ist. Für den Basic-Programmierer bedeutet das, daß er je etwa 60 KByte Speicher für das Programm und die Variablen zur Verfügung hat. Es ist nicht möglich, größere Programme auf Kosten des Variablenspeichers anzulegen. Die vollen 64 KByte pro Bank können auch nicht vollständig genutzt werden, da ein Bereich in beiden Bänken für die Zeropage, den Stack und den Bildschirmspeicher reserviert ist.

122365 Basic Bytes Free

Der Bereich geht bis $0400. Während ein Basic-Programm läuft, regelt die Memory Management Unit, auf welche Bank zugegriffen werden soll. Die Informationen darüber, ob gerade eine Variable oder Befehle zu verarbeiten sind, erhält die MMU vom Basic-Interpreter.

Aber nochmal zurück zu dem Bereich von $0000 bis $0400 der für das System reserviert ist. Die Besonderheit daran ist, daß in diesem Bereich nur Bank 0 existiert. Bank 1 kann dort nicht angesprochen werden. Auf diese Weise ist sichergestellt, daß der Prozessor immer auf dieselbe Bank zugreift und nicht deshalb abstürzt, weil in Bank 1 vielleicht ein anderer Stack steht als in Bank 0. Zusätzlich kann man sich selbst Bereiche von 1 bis 16 KByte in beiden Bänken reservieren, die am Speicheranfang oder Speicherende liegen können.

Man kann diesen Bereich beispielsweise in eigenen Maschinenroutinen als Stack oder Speicher verwenden, wenn zwischen den Bänken umgeschaltet werden muß und die gleichen Daten zur Verfügung stehen sollen.

Speicherzugriff erlaubt

Im Gegensatz zum C64 erlaubt der Expansion-Port des C128 einen direkten Speicherzugriff (DMA, Direct Memory Access). Direkter Speicherzugriff bedeutet, daß ohne Umwege über den Prozessor in den Speicher des C128 geschrieben oder der Speicher ausgelesen werden kann. Das wichtigste, um einen DMA realisieren zu können, ist, daß der Prozessor während des Zugriffs abgeschaltet bleibt. Beim C128 macht das der 8564-VIC. Er steuert den Daten- und Adreßbus so, daß der Prozessor und DMA sich nicht ins Gehege kommen, was beim C64 nicht immer sichergestellt ist. Bei einem gleichzeitigen Bus-Zugriff von Prozessor und externen Geräten erweist sich der Prozessor meist als der Schwächere, was zu ernsthaften Problemen führen kann.

Daß ein direkter Speicherzugriff ohne weiteres machbar ist, eröffnet dem C128 gegenüber dem C64 zusätzliche Einsatzgebiete in der Meßwerterfassung. Ein Meßgerät kann dadurch beispielsweise Meßwerte so schnell direkt in den Speicher schreiben, daß eine Echtzeiterfassung eines Meßvorganges möglich ist. Eine andere Möglichkeit wäre der Anschluß eines Festplatten-Laufwerks. Die Daten könnten dann viel schneller in den RAM-Bereich geladen oder aus dem Arbeitsspeicher geholt werden, als wenn der Prozessor vorher jedes Bit ein paar mal »umdreht«.

Der serielle Bus und der Kassetten-Port des C128 sind von den Anschlüssen her identisch mit denen des C64. Die Bedienung des seriellen Bus wurde überarbeitet, so daß der C128 zusammen mit den neuen Commodore Laufwerken 1570/71 wesentlich schneller speichern und laden kann als der C64.

Kompatibilität war schon immer ein Reizwort für Commodore. Deswegen war Skepsis angesagt, ob der C128 wirklich kompatibel zum C64 ist.

Also haben wir eine Zahl von Programmen ausprobiert, die direkt oder indirekt über einen Kopierschutz im Betriebssystem herumpfuschen oder sonstige Gemeinheiten anstellen, die jeden Nicht-C64 sofort zum Aussteigen bewegen würden. Erster Testkandidat war Hypra-Load. Einige Probeläufe zeigten, daß sich hier in Verbindung mit der 1541 überhaupt keine Probleme ergeben. Damit dürfte gesichert sein, daß alle Programme mit geänderten Busroutinen einwandfrei funktionieren.

Nächstes Testobjekt war ein Kopierprogramm, das intensiven Gebrauch von illegalen Opcodes macht. Mit Opcodes bezeichnet man den Befehlssatz des Prozessors. Illegale Opcodes sind Befehle, die der Hersteller des Prozessors eigentlich gar nicht vorgesehen hat. In Wirklichkeit bewirken aber manche von ihnen auch beim C64 schon etwas. Und einige Programme nutzen sie. Es hätte also sein können, daß der 8502-Prozessor einige dieser beim 6502 an sich undefinierten Opcodes benutzt. Doch traten hier keine Probleme auf. Auch alle anderen kopiergeschützten (und nicht kopiergeschützten) Diskettenprogramme konnten wir ohne Schwierigkeiten laden und benutzen.

GO 64 – wie kompatibel ist der C128?

Getestet wurden von uns diverse Spiele wie Ghostbusters und Pit Stop II. Auch hier ein eindeutiges Ergebnis: Grafik und Musik stimmen mit dem C64 überein. Von über 100 speziell für diesen Test ausgewählten Disketten-Programmen gab es nur bei einem Schwierigkeiten. Dies ist »Rescue on Fractalus, bei dem die Grafik vom C128 nicht richtig aufgebaut wird.

Auch die Datasetten-Besitzer dürften im allgemeinen keine Probleme mit ihren Programmen haben. Bei allen getesteten Kassetten-Programmen gab es nur in einem einzigen Fall Schwierigkeiten. Es handelt sich hierbei um das Spiel »Rolands Rat Race«.

Das Ansteuern von Drucker-Interfaces verlief ebenfalls problemlos. Akkustikkoppler beziehungsweise Modems, die über den User-Port angeschlossen werden, konnten einwandfrei betrieben werden. Keine Schwierigkeiten gab es auch bei dem Betrieb der Btx-Module von Astech und Commodore.

Alle für den C64 vorhandenen Peripherie-Geräte, die über ein Interface angeschlossen werden, dürfen also auch weiterhin nutzbar bleiben.

Bei Modulen mit Modul-Start, die in den Expansion-Port eingeschoben werden, wird beim Einschalten des C128 direkt in den C64-Modus gesprungen. Andere Erweiterungen oder Geräte, die über den Expansion-Port angeschlossen werden (zum Beispiel der Ascom-Akkustikkoppler), beeinflussen den C128-Modus nicht. Nach dem Einschalten gelangt man also in den C128-Modus.

Nach unseren Tests können wir dem C128 tatsächlich die Kompatibilität zum C64 bestätigen. Doch der C128 kann im C64-Modus noch etwas mehr.

Im C128-Modus kann man zwischen ASCII- und DIN-Tastatur über eine Umschalttaste oder über POKE-Befehle vom Programm aus wählen. Diese Umschaltung steht dem Benutzer auch im C64-Modus zur Verfügung. Die DIN-Taste funktioniert auch im C64-Modus. Allerdings wird im C64-Modus nur der Zeichensatz, nicht aber die Tastaturcodes geändert. Dadurch kommt eine etwas merkwürdige Tastenbelegung zustande. Im Programm wird diese Umschaltung (ASCII nach DIN) folgendermaßen vorgenommen:

```
POKE 0, PEEK(0) OR 64: POKE1,0
```

Im C128-Modus kann der Benutzer mit dem FAST-Befehl vom 1-MHz Takt auf den 2-MHz-Takt umschalten. Hierdurch kann eine erhebliche Steigerung der Verarbeitungsgeschwindigkeit erreicht werden. Dies ist auch im C64-Modus möglich. Falls Sie bereits einen C128 besitzen, geben Sie einmal folgendes Programm ein:

```
10 POKE 53265, PEEK (53265)
and 239
20 POKE 53296, PEEK (53296) OR 1
30 FOR I=1 TO 1000:PRINT"A":NEXT
40 POKE 53296, PEEK (53296) AND
254
50 POKE 53265, PEEK (53265) OR 16
```

Im Vergleich zwischen dem C64 und dem C128 im C64-Modus mit 2 MHz wird die Ausgabe von 1000 mal »A« von 35 Sekunden auf 20 Sekunden verkürzt. Hierbei wird allerdings auch der Bildschirm abgeschaltet. Diese Routine kann für alle zeitaufwendigen Programmschritte genutzt werden, bei denen keine Bildschirmausgabe erfolgt.

Der dritte Unterschied zum C64 taucht bei Benutzung der C 157 1-Floppy-Station auf. Im C128-Mode wird die Diskette doppelseitig genutzt. Durch den Befehl

```
OPEN 15,8,15,"UO > M1":CLOSE 15
```

ist dies auch im C64-Modus möglich. Formatieren Sie jetzt eine Diskette und lassen sich das Directory auflisten, werden Sie mit der Meldung »1328 Blocks free« überrascht. Sie haben also den Speicherplatz auf der Diskette verdoppelt.

Sind die Disketten schon einseitig beschrieben (zum Beispiel vom C64) und man möchte die Rückseite nutzen, muß folgender Befehl eingegeben werden:

```
OPEN 15,8,15,"UO > H1"
```

Das Laufwerk gibt dann zwar ein starkes Rattern von sich, nutzt die Diskette aber von der Rückseite. Ist die Diskette neu formatiert, so unterbleibt das Rattern.

Der CP/M-Profi

Der C128 enthält zwei Zentraleinheiten, nämlich einmal einen 8502-Prozessor (kompatibel zum 6510 und 6502), der sowohl im Normal- als auch im C64-Modus aktiv ist, und einen Z80A-Prozessor (4 MHz), der unter CP/M die Kontrolle über den Computer übernimmt.

Schon beim Einschalten des C128 macht sich der Z80 bemerkbar. Er versucht, das CP/M-Betriebssystem von Diskette zu booten (zu laden und zu starten). Ohne irgendeinen Befehl beginnt dabei das angeschlossene Diskettenlaufwerk zu laufen und versucht das Programm zu laden. Wird kein entsprechendes Programm gefunden, aktiviert der Z80 den 8502-Prozessor und der C128-Modus wird eingeschaltet.

Geteilte Datenschienen

Beide Prozessoren, Z80 und 8502, sind in der Lage, miteinander zu kommunizieren, was auch im Betriebssystem vorgesehen ist. Reicht nämlich für bestimmt I/O-Operationen der BIOS-(Betriebssystem-)Befehlssatz des CP/M nicht aus, übernimmt der 8502 diese Aufgaben.

Z80 und 8502 teilen sich im C128 die Adreß- und Datenleitungen. Da der Z80 schneller arbeitet als die übrigen Bausteine, paßt eine Interface-Schaltung die Geschwindigkeit des Z80 an das System an, was natürlich die Arbeitsgeschwindigkeit des Z80 verringert. Diese Interface-Schaltung sorgt dafür, daß der Z80 bei Buszugriffen nur mit 2 MHz anstelle der angegebenen 4 MHz getaktet wird. Das bedeutet, daß CP/M-Programme auf dem C128 nicht ganz so schnell laufen, wie auf einem 4-MHz-CP/M-Computer.

Bleiben wir noch einen Moment beim CP/M. Es handelt sich dabei um eine neuere Version dieses 8-Bit-Betriebssystems für Z80-Prozessoren. CP/M 3.0 plus ist in der Lage, auch über 64 KByte hinausgehende Speicherbereiche einigermaßen sinnvoll zu verwalten.

Der C128 wird einfach dadurch in den CP/M-Modus versetzt, daß man die CP/M-Boot-Diskette ins Laufwerk einlegt und den Computer einschaltet (oder einen Reset auslöst). Das Betriebssystem erkennt, daß es sich beim ersten Sektor der Diskette um einen Autostart-Sektor handelt (die ersten drei Byte eines Autostart-Sektors sind »CBM«), lädt und startet das in diesem Sektor vorhandene Maschinenprogramm automatisch. Dieses Programm stellt die erforderliche Speicherkonfiguration ein und lädt seinerseits Teile des CP/M-Systems nach. Dann geht die Kontrolle an den Z80A-Prozessor über, der das System schließlich vollständig lädt.

Das Diskettenformat unter CP/M ist kompatibel zu den gängigsten anderen Formaten, es können also beispielsweise CP/M-Disketten, die mit einem Kaypro oder Osborne CP/M-Computer geschrieben wurden ohne Probleme gelesen werden. Das Textverarbeitungsprogramm Wordstar als Testprogramm war beispielsweise auch in der Kaypro-Version nach kurzer Installation problemlos lauffähig. Damit ist für den C128-Anwender die Tür zu professioneller Software weit aufgestoßen.

Wer bisher nur mit dem C64 gearbeitet hat, dem stehen bei der Umstellung auf eine CP/M-fähige Maschine natürlich einige Umstellungen bevor.

Was ist CP/M?

Die Bezeichnung CP/M steht für »Control Program for Microcomputers«, also einfach für ein – inzwischen weit verbreitetes – Betriebssystem für Microcomputer. CP/M wurde bereits 1974 von Gary Kildall

bei Intel entwickelt und diente ursprünglich als Dateiverwaltungs-System für einen von Intel vertriebenen Compiler für die Programmiersprache PL/M. Seit 1975 wird CP/M als allgemeines Betriebssystem kommerziell angeboten und ständig weiterentwickelt. Das Erfolgsrezept von CP/M heißt »Standard«.

Da jeder Hersteller eines Mikrocomputers in der Regel sein eigenes Betriebssystem einbaut, ergeben sich sehr große Probleme bei der Übertragung von Programmen und Daten zwischen verschiedenen Computern. Wer schon einmal versucht hat, ein VC 20-Programm an den C 64 anzupassen (oder umgekehrt) der weiß, wovon hier die Rede ist.

Das CP/M-Betriebssystem tritt nun als Vermittler zwischen Anwenderprogramm und Hardware auf. Beispielsweise fragt ein CP/M-Programm niemals die Tastatur direkt ab, denn die kann ja infolge unterschiedlicher Hardware von Computer zu Computer anders konzipiert und angesteuert sein. Statt dessen ruft das Anwenderprogramm eine Routine im CP/M-System auf, die die gewünschte Funktion ausführt. Natürlich muß das CP/M-System für jeden Computer an die spezielle Hardware angepaßt werden, aber, und das ist das Entscheidende, die Anwenderprogramme laufen unverändert, da sie nicht direkt auf die Hardware zugreifen.

Da alle CP/M-Programme generell im gleichen Speicherbereich liegen, treten kaum Probleme bei der Übertragung von einer CP/M-Maschine zur anderen auf.

Grundvoraussetzung für die Übertragbarkeit von Software ist allerdings die Verwendung des gleichen Mikroprozessors in allen CP/M-Computern. Die Wahl fiel bei Intel nicht schwer: Der 8080 aus dem eigenen Hause wurde zum CP/M-Prozessor gekürt. Der Fairneß halber muß man allerdings zugeben, daß der 8080 damals (Mitte der siebziger Jahre) tatsächlich einer der leistungsfähigsten Mikroprozessoren überhaupt war.

Heutzutage wird allerdings fast ausschließlich der Z80 verwendet, der voll aufwärtskompatibel zum 8080 ist, aber über einen stark erweiterten Befehlsvorrat und mehr interne Register verfügt. Wer sich ernsthaft für CP/M interessiert, der sollte sich daher mit diesem Prozessor vertraut machen.

Im Gegensatz zur 6502-Prozessor-Familie (zu der auch der 6510 im C 64 und der 8502 im C 128 gehört) handelt es sich beim Z80 ebenso wie beim Vorgänger 8080 um eine sogenannte »registerorientierte CPU«. Das bedeutet, daß der Z80 über eine größere Anzahl interner Register verfügt, die sowohl als Daten- wie auch als Adreßregister benutzt werden können.

Der Z80-Prozessor

Insgesamt stehen dem Benutzer sechs allgemeine 16-Bit-Register (BC, DE, HL, BC', DE', HL') zur Verfügung, die wahlweise auch als zwölf 8-Bit-Register angesprochen werden können. Daneben gibt es zwei Akkus und ebenfalls zwei Flag-Register, zwei 16-Bit-Indexregister (IX und IY), einen 16-Bit-Stackpointer (SP) und natürlich einen Programmzähler (PC). Zwei weitere Register, das Interrupt-Vektor-Register I und das Refresh-Kontroll-Register R haben spezielle Funktionen.

Die Register AF, BC, DE und Hl bilden die sogenannten Primärregister, die vom 8080 übernommen wurden und die in erster Linie direkt angesprochen werden können. Mit speziellen Befehlen kann zwischen diesen Primärregistern und den »Sekundärregistern« AF', BC', DE' und HL' umgeschaltet werden. Die Indexregister IX und IY haben ähnliche Aufgaben wie die 6502-Register X und Y, sind jedoch 16 Bit breit.

Der Z80-Befehlssatz ist sehr umfangreich und umfaßt unter anderem auch 16-Bit-Arithmetik, Blockverschiebebefehle, automatische Such- und Ein-/Ausgabebefehle, bedingte Unterprogrammaufrufe sowie Einzelbitbefehle. Insgesamt kennt die Z80-CPU über 700 Opcodes. Um diese vielen Maschinenbefehle verschlüsseln zu können (mit einem Befehlsbyte können nur 256 Befehle codiert werden), gibt es einige spezielle »Umschaltopcodes«, die einfach bewirken, daß der Z80 intern auf eine andere Befehlstabelle umschaltet und das nächste Befehlsbyte in einer anderen Form interpretiert.

Nach diesem kurzen Ausflug zum Z80 jetzt jedoch wieder zum CP/M-Betriebssystem. Beim Studium von Handbüchern und CP/M-Literatur stößt man immer wieder auf vier Abkürzungen: BIOS, BDOS, CCP und TPA. Hinter diesen Bezeichnungen verbergen sich die wichtigsten Bestandteile eines CP/M-Systems.

BIOS steht für »Basic Input Output System«, hat aber absolut nichts mit der Programmiersprache Basic zu tun. Die Bezeichnung deutet lediglich an, daß es sich hierbei um eine Sammlung grundlegender Routinen zur Ein-/Ausgabe von Daten handelt.

Das CP/M-BIOS entspricht von der Funktion her ziemlich genau den Kernel-Routinen bei Commodore-Computern. Der gesamte Kontakt eines Anwenderprogramms mit der Hardware-Umgebung läuft über diese Routinen. Alle BIOS-Routinen werden indirekt über das BDOS, das Basic Disk Operating System, durchgeführt. Der Aufruf aller Systemroutinen geschieht durch einen Unterprogrammsprung zur Adresse 5, dem sogenannten BDOS-CALL. Die verschiedenen geforderten Funktionen werden dadurch selektiert, daß im C-Register des Z80 ein Funk-

Code	Funktion
0	System-Kaltstart
1	Zeichen von Tastatur holen
2	Zeichen auf Bildschirm ausgeben
3	Zeichen von externem Gerät holen
4	Zeichen an externes Gerät senden
5	Zeichen an Drucker senden
6	Direkte Ein-/Ausgabe über Konsole
7	I/O-Zuordnung holen
8	I/O-Zuordnung festlegen
9	String auf Bildschirm ausgeben
10	String von Tastatur einlesen
11	Feststellen, ob Taste gedrückt
12	Liefert Versionsnummer des CP/M-Systems
13	Reset des Diskettensystems
14	Laufwerk auswählen
15	Datei öffnen
16	Datei schließen
17	Datei auf Diskette suchen
18	Zweite Datei suchen
19	Datei löschen
20	Nächsten Block lesen
21	Nächsten Block speichern
22	Neue Datei erzeugen
23	Dateinamen ändern
24	Angesprochene Laufwerke feststellen
25	Aktuelles Laufwerk feststellen
26	DMA-Pufferadresse festlegen
27	Bit-Map-Adresse holen
28	Schreibschutz setzen
29	Schreibschutzinformation holen
30	Datei-Attribute setzen
31	File-Parameter-Adresse holen
32	USER-Nummer setzen
33	Daten von Diskette lesen
34	Daten auf Diskette schreiben
35	Dateigröße feststellen
36	Datensatzadresse holen

Tabelle 5. Die CP/M-Funktionsaufrufe

tionscode übergeben wird. Die Routine ab Adresse 5 verzweigt dann abhängig vom Inhalt des C-Registers zu den verschiedenen BIOS-Funktionen (Tabelle 5).

Diese beiden Teile des CP/M-Systems, BIOS und BDOS, werden beim »Booten«, also beim Laden des Systems von der Diskette, einmal in den Speicher geholt und bleiben dann ständig resident vorhanden.

Für den Kontakt zwischen CP/M und Anwender sorgt der »Console Command Processor« (CCP). Es handelt sich dabei um einen einfachen Kommando-Interpreter, der über die Tastatur eingegebene CP/M-Kommandos erkennt und ausführt. Mit CP/M-Kommandos hat es dabei eine besondere, grundlegende Bewandtnis. Hierzu ist es wichtig zu wissen, daß Dateinamen unter CP/M aus zwei Teilen bestehen, der eigentlichen Dateibezeichnung (acht Zeilen Länge) und der Namenserweiterung (»Extension«, drei Zeichen Länge). Diese Erweiterung, die durch einen Punkt vom eigentlichen Namen abgetrennt wird, gibt den Typ der Datei an.

Kommandostruktur

Besonders wichtig ist der Dateityp »COM«. Wenn ein Dateiname die Erweiterung COM hat, dann stellt sie ein CP/M-Kommando dar. Das funktioniert folgendermaßen: Wenn der Benutzer ein Kommando eingibt, dann sucht der CCP auf der Diskette nach einer COM-Datei, die den Namen des Kommandos trägt. Diese Datei wird dann geladen und gestartet, mit anderen Worten, die Datei ist das Kommando.

Will man beispielsweise das Directory der Diskette sehen, dann gibt man den Befehl DIR ein. Der CCP sucht, lädt und startet daraufhin das Programm »DIR.COM«, das wiederum das Inhaltsverzeichnis der Diskette ausgibt. Derartige Kommandos heißen »transiente Kommandos«, weil sie eben nicht fest eingebaut sind, sondern nur bei Bedarf geladen werden. Der für transiente Kommandos reservierte Speicherbereich heißt »Transient Program Area« (TPA), womit auch die vierte der oben angesprochenen Abkürzungen geklärt wäre. Jedes CP/M-System verfügt auf der Systemdiskette über eine Standard-Bibliothek von wichtigen transienten Kommandos zum Ändern von CP/M-Parametern, zum Kopieren, Löschen, Listen und Umbenennen von Dateien und ähnliches, was man zum sinnvollen Arbeiten mit einer Diskettenstation benötigt.

Natürlich ist es möglich, diesen Befehlsvorrat zu erweitern, indem man entsprechende Assemblerroutinen schreibt oder auch einen Compiler einsetzt. Einzige Bedingung für transiente Kommandos ist die Lauffähigkeit des Maschinenprogramms in der TPA. Generell sind alle CP/M-Anwenderprogramme transiente Kommandos. Das Textverarbeitungsprogramm Wordstar heißt unter CP/M zum Beispiel »WS.COM« und wird einfach mit dem Kommando »WS« aufgerufen.

Mit dem CP/M-Betriebssystem hat der C128-Benutzer Zugriff auf das größte Softwareangebot der Welt. Immerhin werden bereits seit zehn Jahren CP/M-Programme entwickelt und immer weiter verbessert. Textverarbeitung, Datenbanken, Tabellenkalkulation stehen in großer Auswahl ebenso zur Verfügung wie spezielle Branchenlösungen für den kommerziellen Einsatz. Statistik-Pakete, mathematische und naturwissenschaftliche Software ist in Mengen erhältlich. Wer selber programmieren will, der findet unter CP/M ein Angebot an Programmiersprachen, das von Fortran und Cobol über Pascal bis zu Ada und Lisp reicht. Unter CP/M gibt es so ziemlich alles, was auch auf Großrechnern läuft.

Allerdings soll nicht verschwiegen werden, daß trotz Standard und Kompatibilität gewisse Probleme auftreten können. So muß CP/M-Software generell erst einmal an einem speziellen Computer angepaßt werden. Das erledigt ein zu jeder Software gehöriges sogenanntes Installations-Programm, das zum Beispiel die Tastaturbelegung, Bildschirmansteuerung etc. abfragt, um beispielsweise ein Textprogramm optimal auf den jeweiligen Computer einstellen zu können. Dieser kleine Arbeitsaufwand ist aber natürlich gar nichts im Vergleich beispielsweise zur Aufgabe, etwa den Textomat oder Vizawrite an den VC20 anzupassen.

Als Fazit darf festgehalten werden, daß dem C128-Besitzer durch das CP/M-Betriebssystem ein riesiges Software-Potential zur Verfügung steht. Inzwischen gibt es auch bereits die erste »schlüsselfertige« CP/M-Software für den C128: Wordstar, Turbo Pascal, dBase II und Multiplan gibt es fix und fertig an den C128 angepaßt. Alle vier Programme sind zu Preisen unter 200 Mark erhältlich – geradezu sensationell, wenn man sich die Preise sonstiger CP/M-Software ansieht. Es dürfte von entscheidender Bedeutung sein, ob auch andere Hersteller von CP/M-Software willens sind, die Preise für ihre Produkte, die sich teilweise noch im vierstelligen Bereich befinden, drastisch zu senken.

Die Chancen dafür stehen nicht schlecht, denn nachdem CP/M im professionellen Bereich inzwischen durch MS-DOS, ein 16-Bit-Betriebssystem, abgelöst worden ist, sollten die CP/M-Hersteller eigentlich schon an neuen Märkten interessiert sein, und diese neuen Märkte könnten durchaus im oberen Heim-Bereich liegen. Wenn die Software jedoch teurer als der Computer ist, dürfte wohl kein Geschäft zu machen sein.

Fazit

Der C 128 ist ein Gerät einer neuen Leistungsklasse im 8-Bit-Bereich zwischen Homecomputer und Personal Computer. Ganz sicher ist er, bei einem Preis von immerhin knapp 1000 Mark, zumindest auf mittlere Sicht keine Ablösung für den C64, der für die Hälfte zu haben ist.

Der C128 ist aber die ideale Maschine sowohl für den »Aufsteiger«, der vom C64 kommt als auch für den Einsteiger, der einen Computer nicht (nur) zum Spielen sucht, sondern ernsthaft damit arbeiten will.

Dank des CP/M-Betriebssystems ist aus dem Stand jede Menge professioneller Software aus allen Bereichen verfügbar und natürlich auch jede Menge Programmiersprachen. Unter CP/M läuft ja fast alles, was es auch auf Großrechenanlagen gibt, von Cobol und Fortran bis hin zu ausgefeilten Datenbanksystemen.

Und schließlich ist da auch noch das Riesenangebot an C64-Software, die zwar auch so schon läuft, die aber durch vielfach nur kleine Änderungen auch für den C128-Modus verfügbar gemacht werden kann. Die meisten Textverarbeitungs-Programme für den C64 sind mittlerweile auch für den C128-Modus mit 80-Zeichen am Bildschirm erhältlich oder werden es bald sein.

Fest steht, daß der Markt der Heim- und Personal Computer mit dem C128 um einen Computer bereichert worden ist, der das Zeug zu einem neuen Stern am 8-Bit-Computerhimmel hat. (ev)