Mit dem C 16 stellt Commodore nicht nur einen neuen Computer vor, sondern ein ganzes System (Bild 1, 6 und 7). Computer, Datasette und Joysticks präsentieren sich im schwarz-grauen »Profi-Look«, Drucker und Floppy-Laufwerk des VC 20/C 64-Systems passen auch an den C 16. Gleichzeitig mit der Markteinführung des neuen Gerätes ist auch erste Software — zunächst auf Kassette — erhältlich. Commodore trägt damit der Tatsache Rechnung, daß angesichts eines umkämpften Marktes ein neuer Computer ohne Peripherie und Softwareangebot kaum noch erfolgreich einführbar ist.

Um es vorweg zu sagen: C 16 und 116 sind hard- und softwaremäßig völlig identisch. Der einzige Unterschied besteht bei Gehäuse und Tastatur. Während der C 16 hier der durch VC 20 und C 64 vorgegebenen Linie folgt, ist der 116 mit seiner Radiergummi-Tastatur und dem Miniatur-Gehäuse wohl eher als direkter Angriff auf den Sinclair Spectrum zu werten.

Doch sehen wir uns zunächst den C 16 etwas genauer an. Ein erster Blick auf die Tastatur (Bild 2) offenbart schon einige Unterschiede zum VC 20/C 64. Aus den beiden Cursortasten des Vorgängers wurden beim C 16 deren vier, die aber leider etwas ungünstig rechts oben in einer Reihe in den Tastaturblock integriert wurden. Um für diese Anordnung Platz zu schaffen, mußten einige andere Tasten verlegt werden. So findet der an die VC 20-Tastatur gewöhnte Programmierer die häufig benötigten Tasten »+«, »-«, »*«, »↑« und »=« nicht mehr an ihrem gewohnten Platz, was zu Anfang recht lästig ist. Insbesondere die Anordnung der »=«-Taste ganz rechts unten ist sehr unglücklich gewählt.

Die unterste Funktionstaste ist jetzt mit »HELP« beschriftet und hat eine spezielle Bedeutung bei der Fehlersuche. Drückt man nämlich diese Taste, nachdem der Computer eine Fehlermeldung angezeigt hat, dann wird die fehlerhafte Zeile sofort aufgelistet und der Abschnitt, in dem der Fehler auftrat, blinkt mit der Cursorfrequenz. Eine feine Sache bei der Programmentwicklung, allerdings wäre es schön, wenn man das mit der Zeit doch etwas nervende Blinken auf irgendeine einfache Art abstellen könnte.

Eine Restore-Taste gibt es nicht mehr, die Linkspfeiltaste des VC 20 / C 64 ist jetzt mit »ESC« (mehr darüber später) beschriftet und im Vordergrund fallen zwei neu beschriftete Tasten, »FLASH ON« und »FLASH OFF«, ins Auge. Zusammen mit der CTRL-Taste wird dadurch ein Blink-Modus ein- beziehungsweise ausgeschaltet (analog zu RVS ON/OFF).

Die Funktionstasten sind mit Basic-Befehlen belegt. Sonst entspricht sowohl die Tastatur als auch der Zeichensatz dem »Commodore-Standard«, man wird als »Umsteiger« wenig Schwierigkeiten haben.

Bild 2. Die Tastatur ist bewährt und gut. Deutlich erkennbar ist die gegenüber dem VC 20 geänderte Belegung einiger Tasten.

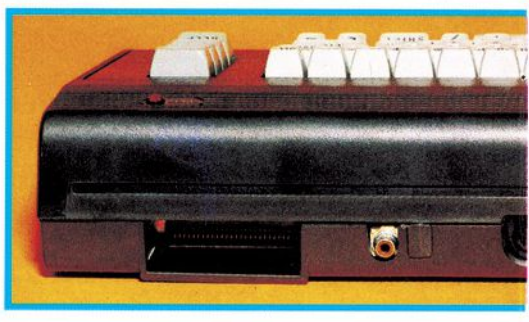


Bild 4. Viele Anschlußmöglichkeiten an der Rückseite (von li
buchse, Monitorausgang, serieller Port und Datasetten-Anschl

swechsel

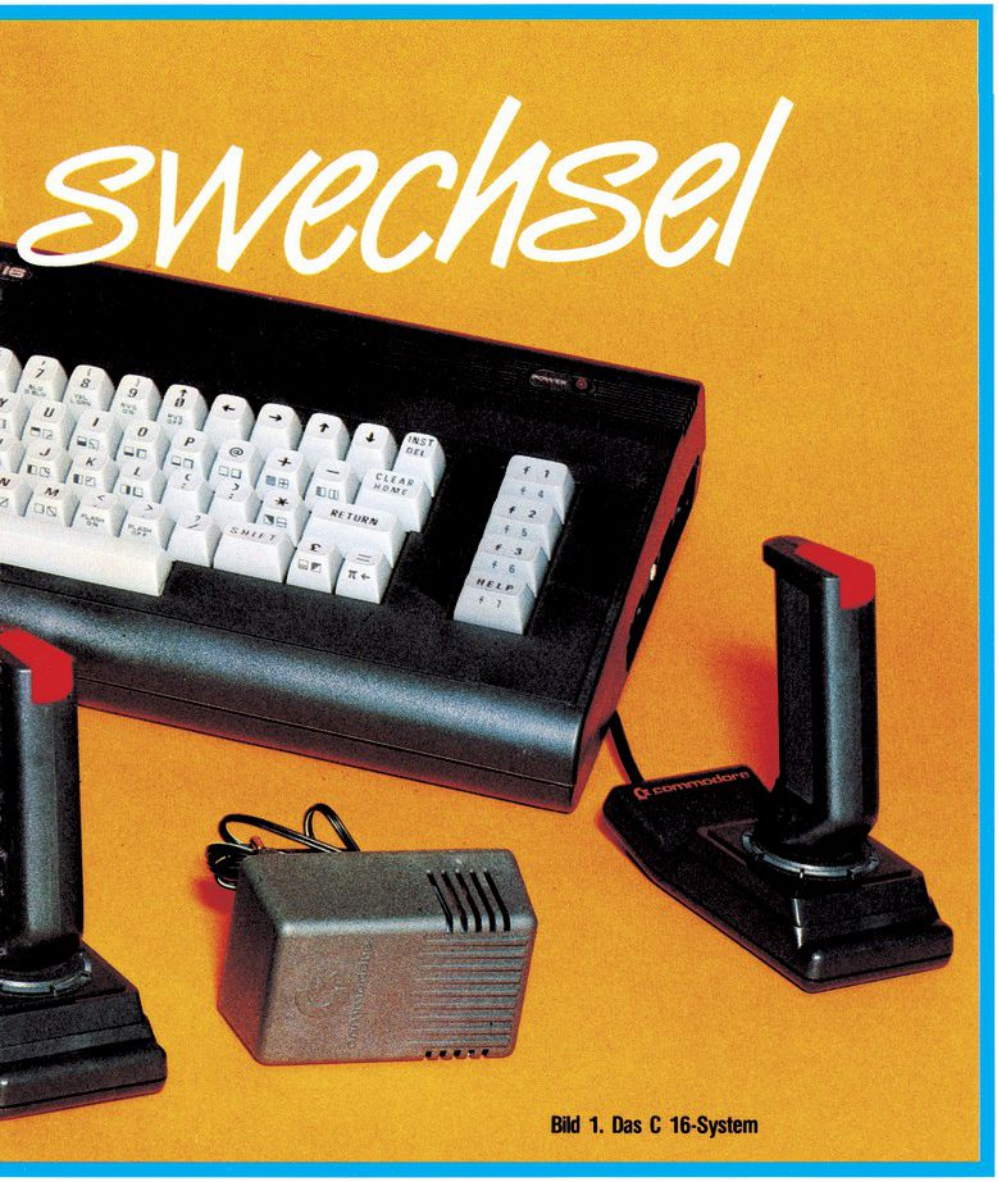


Bild 1. Das C 16-System

Ein erster Blick an die rechte Seite des C 16, wo man ganz richtig den Einschaltknopf vermutet, und sofort

Kompatibel um keinen Preis?

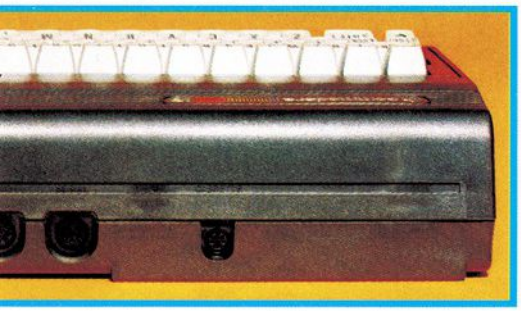
fallen zwei Dinge ins Auge (Bild 3). Als erstes, und zwar sehr positiv, ein kleiner Reset-Schalter — unverständlich, daß es Computer gibt, die keinen haben. Links daneben zwei winzige Buchsen, darüber steht etwas geschrieben. Man liest es, reibt sich die Augen, schaut nochmals hin — tatsächlich, es ist wahr: Die Mikro-Buchsen sind mit JOY 1 und JOY 2 beschriftet. Endlich einmal ein Computer, an den garantiert kein Joystick außer einem ganz speziellen Commodore-Stick mehr paßt. Diese neuen Joystickanschlüsse sollen über eine verbesserte Abschirmung verfügen, aber es bleibt die Frage, ob man den gleichen Effekt nicht auch mit Standard-Buchsen hätte erreichen können.

Was macht nun jemand, der zum Beispiel vom VC 20 auf den C 16 umsteigt, mit seinen vorhandenen Joysticks ? Nun, vermutlich das gleiche wie mit seiner Datasette, denn auch der Datasettenanschluß wurde geändert. Da hilft alles nichts, entweder wird im Eigenbau ein entsprechender Zwischenstecker hergestellt, oder die alte Datasette wandert zusammen mit dem Lieblingsjoystick in eine Verkaufsanzeige. Eine dritte Möglichkeit: Auf ein Angebot aus dem Zubehörhandel warten. Ob so etwas aber gerade ein besonders gutes Argument ist, auf den C 16 umzusteigen, das mag dahingestellt bleiben.

Problemlos erweiterbar?

Ein ängstlicher Blick auf die Rückseite des Computers (Bild 4) zeigt, daß wenigstens der serielle Bus nicht mit Spezialbuchsen versehen wurde. Floppy und Drucker sind also weiterhin problemlos anzuschließen. Ein Video-Modulator ist wie beim C 64 fest eingebaut. Auffällig ist das Fehlen eines User-Ports, bisher Kennzeichen aller Commodore-Computer.

Der Expansion-Port dient zum Aufnehmen von Steckmodulen mit fertiger Software, sowie zum Anschluß einer (noch nicht erhältlichen) Speichererweiterung. Zu diesem Thema wäre zu bemerken, daß das Betriebssystem mit zwei Speicherbanks arbeitet. Zwischen den 32 KByte ROM von Betriebssystem und Basic und den (noch) 16 KByte RAM wird mittels Bank-Switching hin- und hergeschaltet. Dadurch kann man mit PEEK nicht ins ROM »hineinschauen«, sondern bewegt sich nur auf der RAM-Ebene. Nach Einbau einer 64-KByte-RAM-Erweiterung sollten daher tatsächlich fast 60 KByte für Basic-Programme zur Verfügung stehen.



nks nach rechts): Expansion-Port, Antennen-
uß.

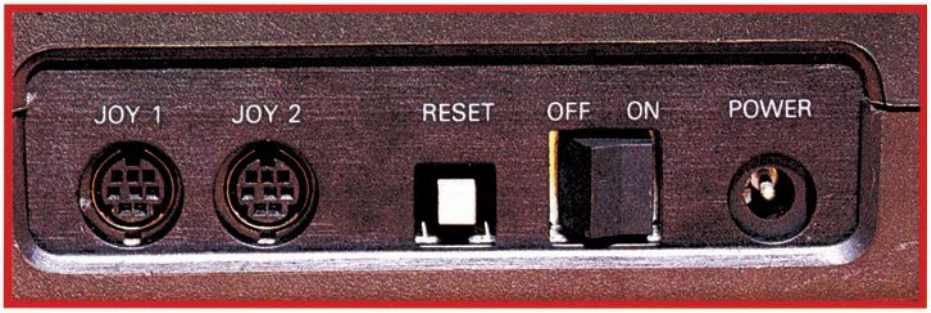


Bild 3. Der C 16 von der Seite. Deutlich zu erkennen der weiße Reset-Schalter und die Mikro-Joystickbuchsen. Auch der Netzteil-Anschluß wurde geändert.

Eine solche RAM-Erweiterung hätte übrigens bequem noch im Gehäuse des C 16 Platz. Ein Blick dort hinein auf die Platine (Bild 5) offenbart ein sehr aufgeräumtes Innenleben. Die großen integrierten Bausteine, insbesondere die neue CPU 7501 (kompatibel mit 6502/6510) und der ebenfalls neuentwickelte TED 7360, stellen Eigenentwicklungen von Commodore dar und werden nicht frei gehandelt. Informationen über diese Bausteine gibt es daher — ganz nach Art des Hauses — praktisch keine.

Auffällig ist das Fehlen weiterer Peripheriebausteine wie VIA oder CIA. Die Funktionen dieser Bausteine wurden in den TED integriert, der sich auch um die Video-Darstellung und die Tonerzeugung kümmert und so den Prozessor entlastet.

Betriebssystem und Basic sind in je einem 128 KBit ROM untergebracht, die 16 KByte RAM befinden sich in zwei TMS 4416-Chips. Zur besseren Wärmeableitung und Abschirmung ist über der Platine eine gelochte Metallplatte angebracht (im Bild 5 entfernt), von der ein Ausleger direkt mit dem scheinbar besonders kühlungsbedürftigen TED-Baustein Kontakt hat.

Besonders bemerkenswert: Bei unserem Testgerät waren alle hochintegrierten IC gesockelt. Es bleibt abzuwarten, ob diese gute Technik auch bei größeren Stückzahlen beibehalten wird. Und größere Stückzahlen werden von diesem Computer mit Sicherheit verkauft werden, dafür sorgt schon das bemerkenswert gute und umfangreiche Basic 3.5, das den C 16 zum idealen Computer für alle diejenigen macht, die wenig mit Maschinensprache im Sinn haben, aber trotzdem gute Programme schreiben wollen.

Bild 6. Die Datasette präsentiert sich auch im neuen Design

Gute Programme für C 64 und VC 20 zeichnen sich im wesentlichen dadurch aus, daß man sie nicht mehr lesen kann. Nach LIST flimmert dort, zumindest bei Programmen, die mit Grafik und Tonuntermalung arbeiten, ein unergründliches Gemisch von POKE, PEEK, SYS und sehr viel DATA über den Bildschirm, ab und zu auch einmal ein »normaler« Basic-Befehl (in der Regel ein »GOTO«). Wenn man sich einmal vor Augen hält, daß POKE, PEEK und SYS die Verbindung zwischen Basic und Maschinensprache herstellen, dann kann man ruhigen Gewissens sagen, daß die bisherigen Commodore-Heimcomputer im wesentlichen in Maschinensprache programmiert werden mußten — unverständlich für den Einsteiger, schwer erträglich aber auch für den Profi, der kostbare Programmierzeit damit vergeudete, sich seine eigene Basic-Erweiterung zu basteln, um überhaupt erst vernünftig arbeiten zu können.

Programmieren ohne POKEs

Mit dem Basic 3.5 beschreitet Commodore jetzt ganz offensichtlich einen anderen Weg. Von der simplen Farbwahl über die Tonerzeugung bis hin zur hochauflösenden Grafik läßt sich alles mit entsprechend leistungsfähigen Basic-Befehlen programmieren. Daneben wird natürlich auch die strukturierte Programmierung unterstützt. Sprachkonstruktionen wie IF...THEN ...ELSE, DO WHILE oder DO UNTIL machen die dem Programm zugrunde liegende Idee im Listing sichtbar und vermeiden umständliche (und langsame) GOTO-Sprünge.

Natürlich ist Basic 3.5 vollständig aufwärtskompatibel zum altertümlichen V.2.0-Minimal-Basic des VC 20/C 64. Insgesamt ist das Basic 3.5 so leistungsfähig, daß alleine eine genaue Beschreibung aller Befehle und Funktionen leicht ein ganzes Sonderheft füllen würde (Tabelle 1). Beschränken wir uns daher auf die Betrachtung einiger wichtiger Aspekte.

Die Grafik

Der Bildschirm des C 16 hat eine Aufteilung von 25 Zeilen zu je 40 Zeichen. Jedes Zeichen wird in einer 8x8-Matrix dargestellt. Damit gibt es insgesamt also 320 Punktpositionen (40x8) pro Zeile, und 200 Punktpositionen (25x8) in vertikaler Richtung.

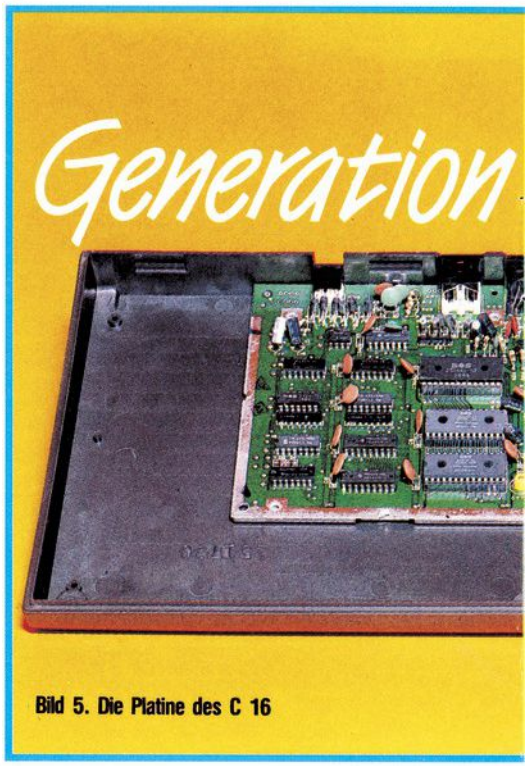


Bild 5. Die Platine des C 16

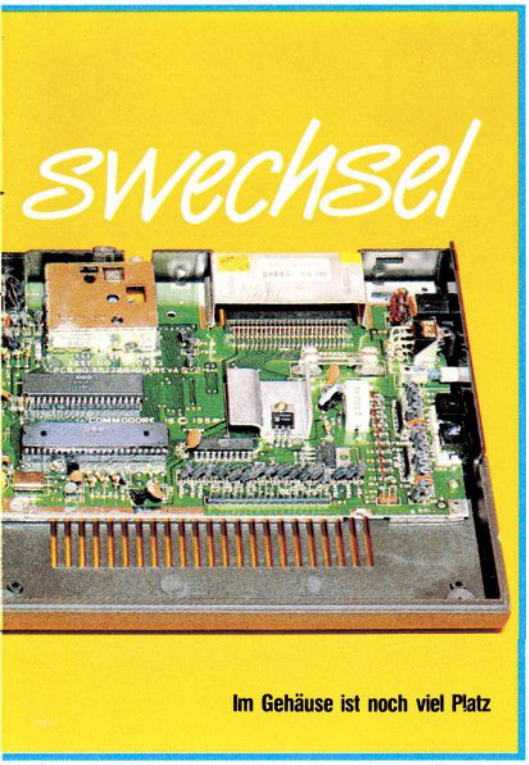
Genausoviele Punkte, nämlich 320 mal 200, können im hochauflösenden Grafik-Modus einzeln angesprochen werden. Mit insgesamt 64000 Einzelpunkten erreicht der C 16 damit die Grafik-Auflösung des C 64, was in etwa einer Verdopplung gegenüber dem VC 20 gleichkommt.

Daneben ist noch ein Mehrfarbenmodus mit halbierter Auflösung vorgesehen, bei dem jeder der dann 32000 Einzelpunkte eine von vier möglichen Farben annehmen kann.

Bis hier herrscht noch eine völlige Übereinstimmung zum Grafik-Konzept des C 64. Der große Unterschied ist nun, daß die C 16-Grafik vom Basic voll unterstützt wird.

Um beispielsweise mit dem C 64 (ohne Erweiterung!) einen Kreis oder eine Ellipse in hochauflösender Grafik zu zeichnen, und zusätzlich noch einen Text einzublenden, benötigt man neben fundierten Kenntnissen über die interne Organisation seines Computers und einigen Jahren Programmiererfahrung mindestens einige Dutzend Basic-Zeilen und jede Menge Geduld — denn Basic-Grafik ist langsam, wenn die primitivsten Dinge mühselig simuliert werden müssen. So werden ganze Kurse und Bücher damit gefüllt, auf dem C 64 das zu erreichen, wofür man beim C 16 drei Basic-Befehle und — als absoluter Neuling — maximal einige Minuten blättern im Handbuch braucht:

```
10 GRAPHIC 1,1 : CIRCLE, 160,100,
60,30 : CHAR,18,12, »HALLO«
```



Im Gehäuse ist noch viel Platz

Mit GRAPHIC 1,1 wird in den hochauflösenden Grafikmodus geschaltet. Der erste Parameter gibt den gewählten Modus an (0 für Text, 1 für Hochauflösung, 2 für Hochauflösung mit Textfenster, 3 für Mehrfarben-Modus, 4 für Mehrfarben-Modus mit Textfenster). Als Textfenster sind dabei die untersten fünf Bildschirmzeilen vorgesehen. Dieses Textfenster ist dann mit normalen PRINT-Befehlen ansprechbar. Will man Text direkt in die hochauflösende Grafik hineinmischen, dann bedient man sich des CHAR-Befehls. Der erste Parameter bestimmt im Mehrfarbenmodus die Textfarbe (und wird daher in unserem Beispiel für hochauflösende Grafik durch ein einzelnes Komma ersetzt). Die beiden folgenden Parameter kennzeichnen die Cursorposition, an welcher der Text ausgegeben werden soll. Als letztes muß noch ein Textstring angegeben werden, der dann ab der spezifizierten Position in die Grafik eingeblendet wird. Ein fünfter Parameter ist optional, nämlich die Angabe, ob der Text normal oder revers erscheinen soll.

Bleibt nur noch der CIRCLE-Befehl, der mit bis zu neun Parametern sehr komplex sein kann. Der erste Parameter gibt wieder die Farbzone an und ist nur im Mehrfarbenmodus zu verwenden. Dann folgen die Mittelpunktkoordinaten und der Radius in X- und Y-Richtung. Die letzten beiden Werte sind bei einem Kreis natürlich gleich groß, und daher reicht es, nur den ersten davon anzugeben. In unserem kleinen Beispiel haben wir jedoch eine Ellipse gezeichnet, und zwar mit Mittelpunkt in (160,100) und den Halbachsenabmessungen 60 beziehungsweise 30 Punkte.

Aber der CIRCLE-Befehl ist noch weitaus leistungsfähiger. Weitere Parameter regeln das Zeichnen nur einzelner Segmente sowie eine Drehung der ganzen Figur um einen beliebigen Winkel. Neben dem Zeichnen von Kreisen und Ellipsen kann der CIRCLE-Befehl auch für beliebige Vielecke verwendet werden.

Daneben steht noch eine Anzahl weiterer leistungsfähiger Grafik-Befehle zur Verfügung. DRAW zeichnet Einzelpunkte oder Linien, BOX zaubert blitzschnell alle möglichen Rechtecke auf den Bildschirm. LOCATE dient zum Positionieren des Grafikcursors, mit PAINT werden geschlossene Flächen ausgefüllt. SCALE schließlich dient zur Skalierung der Zeichenfläche und SCNCLR löscht den Bildschirm unabhängig vom eingestellten Grafikmodus.

Shapes statt Sprites

Im Gegensatz zum C 64 sind beim C 16 keine Sprites vorgesehen. Dafür gibt es jedoch leistungsstarke Basic-Befehle, um Bildausschnitte aus der hochauflösenden Grafik — sogenannte »Shapes« — in Stringvariable abzuspeichern oder wieder auf den Bildschirm zu bringen. Zum Beispiel wird mit »SSHAPE A$,100, 100,120,130« der Inhalt des Rechtecks mit linker oberer Ecke (100,100) und rechter unterer Ecke (120,130) auf dem Grafikbildschirm in der Stringvariablen A$ abgelegt. Mit »GSHAPE A$, 60,70« wird die Grafikinformation aus A$ wieder als Rechteck mit linker oberer Ecke (60,70) abgelegt. Über einen zusätzlichen (optionalen) Parameter kann der Wiedergabemodus bestimmt werden: Shapes können genauso wie aufgenommen auch wieder eingeblendet werden (und überschreiben dabei den Hintergrund), sie können revers dargestellt und schließlich wahlweise auch ODER, UND oder EXKLUSIV-ODER mit dem Hintergrund verknüpft werden.

Mit diesen Shapes eröffnen sich natürlich speziell bei der Spieleprogrammierung ungeahnte Möglichkeiten. Basic-Spiele in hochauflösender Grafik sind keine Utopie mehr. Unter diesem Gesichtspunkt verzichtet man gerne auf ein paar kümmerliche Sprites, die sich über einem Blockgrafik-Hintergrund bewegen. Dies um so leichter, als sich Sprites mit dem Shape-Konzept leicht simulieren lassen: Man arbeitet einfach mit zwei Stringvariablen. Die eine enthält die Spielfigur, die andere den zugehörigen Hintergrund (der ja durch die Figur verdeckt ist). Durch wechselseitiges Laden und Speichern von Hintergrund und Spielfigur lassen sich sehr einfach die entsprechenden Bewegungseffekte erzielen — und das sowohl in hochauflösender Grafik als auch in Mehrfarbengrafik.

Farbe und Sound

Der C 16 hat eine Grundpalette von 16 Farben zur Verfügung, von denen jede (bis auf Schwarz) noch in acht verschiedenen Intensitätsstufen dargestellt werden kann. Das ergibt insgesamt eine beachtliche Auswahl von 121 Farbtönen. Wer bietet mehr für 398 Mark?

Mit dem COLOR-Kommando können dabei die Farben für Bildschirmrahmen, Hintergrund, Mehrfarbenmodus und auch die Zeichenfarbe gewählt werden.

Zur Tonerzeugung stehen zwei unabhängige Tongeneratoren zur Verfügung, von denen einer auch für Geräuscheffekte eingesetzt werden kann. POKE-Befehle sind auch hier nicht nötig. Mit dem SOUND-Befehl werden sowohl der gewünschte Tongenerator angewählt als auch Tonhöhe (Notenwert) und Klangdauer angegeben, mit VOL wird die Lautstärke eingestellt.

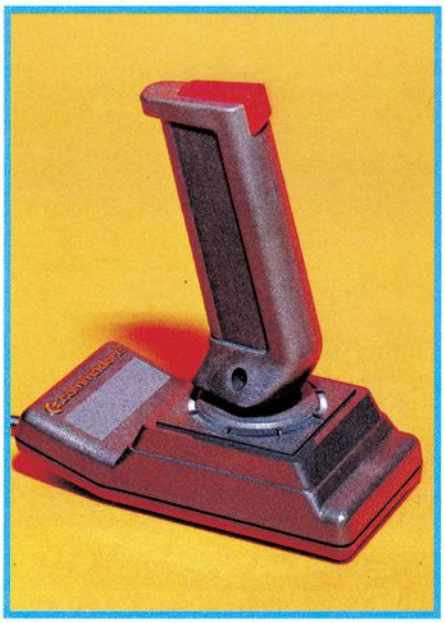


Bild 7. Nur dieser Commodore-Joystick kann derzeit angeschlossen werden

Das Besondere dabei ist, daß bis zu zwei SOUND-Befehle (einer pro Kanal) parallel zum Basic-Programm ausgeführt werden. Der Programmablauf wird durch einen SOUND-Befehl also nicht etwa aufgehalten, bis der Ton zu Ende gespielt wurde; das Programm läuft normal weiter, während der Ton entsprechend der gewählten Tondauer erklingt. Man kann sich wohl vorstellen, wie diese Fähigkeit ein Programm beschleunigt, wenn man sich vor Augen hält, daß Computer der Vorgängergeneration (VC 20/C 64) die Tondauer über leere FOR...NEXT-Schleifen bestimmten.

Komfortable Programmierung

Bei der Entwicklung von eigenen Programmen und der in der Regel notwendigen Fehlersuche kommen die eingebauten Programmierhilfen des 3.5 Basic erst richtig zur Geltung. Eine Reihe von Befehlen und speziellen Funktionen stehen zur Verfügung, die bisher bei Commodore-Computern unter der Abkürzung OGNV (oft gebraucht, nie vorhanden) liefen.

Eine automatische Zeilennumerierung mittels AUTO ist ebenso selbstverständlich wie ein RENUMBER-Befehl zum Neunumerieren des Programms (wobei wahlweise auch nur Programmteile numeriert werden können und der Zeilenabstand sowie die Startzeile natürlich frei wählbar sind).

Diskettenkommandos, die man früher umständlich mit »OPEN 1,8,15...« an die Floppy senden mußte, sind jetzt als Basic-Kommandos integriert. SCRATCH löscht beispielsweise ein File auf der Diskette. Das Laden und Speichern von Diskettenprogrammen geschieht jetzt mit DLOAD beziehungsweise DSAVE. DIRECTORY holt das Inhaltsverzeichnis der Diskette auf den Bildschirm, selbstverständlich ohne Programmverlust.

Für das häufig benötigte Warten auf einen Tastendruck gibt es den Spezialbefehl GETKEY A$, wodurch man sich das lästige »IF A$="" "THEN...« spart.

Hinter RESTORE kann eine Zeilennummer angegeben werden, was das einfache Hantieren mit mehreren unabhängigen DATA-Blöcken erlaubt.

Formatierte Ausgabe ist mit PRINT USING möglich. Die dabei verwendeten Zeichen lassen sich mit dem PUDEF-Kommando umdefinieren. Beispielsweise kann man für die formatierte Ausgabe den (amerikanischen) Dezimalpunkt durch das in Europa übliche Komma ersetzen. Mit ZONE kann die Weite der TAB-Bereiche geändert werden.

Die häufig gebrauchte INSTR-Funktion (hat als Ergebnis die Position eines Teilstrings in einem anderen String) ist ebenso vorhanden wie die Funktion JOY zur einfachen Joystickabfrage.

Bemerkenswert ist auch, daß die MID$-Funktion jetzt auch auf der linken Seite einer Wertzuweisung stehen kann. Sei beispielsweise A$="HALLO". Nach Ausführung des Befehls »MID$(A$,2,1) = "E"« ist A$ dann gleich der Zeichenfolge »HELLO«.

Für die Umrechnung zwischen Dezimal und Hexadezimal sind die beiden Funktionen DEC und HEX$ vorhanden.

Strukturierte Programmierung

Basic 3.5-Programme sind in der Regel um einiges übersichtlicher (und dabei schneller) als VC 20/C 64-Programme. Der Grund ist einleuchtend: Durch zusätzliche Schleifenbefehle werden GOTO-Anweisungen eingespart, und damit entfällt auch die Suchzeit, um die Zeilennummer zu finden.

Daneben wurde auch die IF-Anweisung um die ELSE-Klausel erweitert, was die Programmierung in vielen Fällen vereinfacht.

Der Kern der neuen Schleifenstruktur besteht aus den Anweisungen DO und LOOP. Ähnlich wie FOR...NEXT umklammert DO...LOOP einen Programmteil. Die Wirkung ist die folgende: Bei Erreichen eines DO merkt sich der Basic-Interpreter die Adresse dieses DO-Befehls als Schleifenanfang. Wird dann im weiteren Verlauf das zugehörige LOOP gefunden, erfolgt sofort ein Rücksprung zur Position des DO-Befehls. Das ergibt eine »unendliche« Schleife, was allerdings in den meisten Fällen nicht erwünscht ist. Daher ist die EXIT-Anweisung vorgesehen, die ein Verlassen der Schleife und eine Fortsetzung des normalen Programmablaufs hinter dem LOOP-Befehl ermöglicht. In der Regel wird man das EXIT von einer bestimmten Bedingung abhängig machen. Beispiel:

```
10 DO
20 GET A$ : IF A$=»X« THEN EXIT
30 LOOP
```

Dieses Programm wartet, bis die Taste X gedrückt wird.

Die (unbedingte) DO...LOOP-Schleife kann unter Verwendung von UNTIL oder WHILE in eine bedingte Schleife abgewandelt werden. DO WHILE ⟨Bedingung⟩ ... LOOP wird ausgeführt, solange die ⟨Bedingung⟩ erfüllt ist. Durch UNTIL wird praktisch der umgekehrte Fall erzeugt. DO UNTIL ⟨Bedingung⟩ ... LOOP wird solange durchlaufen, bis die Bedingung erfüllt ist. Natürlich können auch bei bedingten Schleifen zusätzliche EXITs eingebaut werden. Das ermöglicht sehr effiziente Programme, insbesondere, wenn mehrere Bedingungen beachtet werden müssen.

Gemäß der Parole »wer viel programmiert macht viele Fehler« ist jedes Basic nur so gut wie seine Hilfen zur Fehlersuche und Fehlerbehandlung. Und hier hat der C 16 einiges zu bieten.

Die HELP-Funktion wurde bereits anfangs erwähnt und ermöglicht die schnelle Lokalisierung eines Fehlers innerhalb einer Programmzeile.

Für den nicht seltenen Fall, daß keine Fehlermeldung erfolgt, das Programm jedoch unsinnige Sachen macht (also irgendwo noch ein logischer Fehler steckt) kann man mit TRON eine Trace-Funktion einschalten. Dabei wird die Zeilennummer der gerade abgearbeiteten Zeile angezeigt, wodurch man so manchem Fehler leichter auf die Spur kommen kann. TROFF schalten den Trace wieder ab.

Debugging leicht gemacht

Für die Fehlerbehandlung innerhalb des Programms ist der TRAP-Befehl vorgesehen. Zum Beispiel wird mit der Anweisung »TRAP 1000« beim Auftreten eines Fehlers das Programm nicht mit entsprechender Meldung abgebrochen, sondern es wird in eine Fehlerbehandlungsroutine gesprungen (hier ab

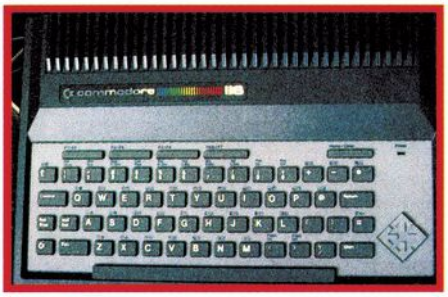


Bild 8. Der Gummitasten-Zwilling C 116

Generationswechsel

TEST C16

Zeile 1000). Die Nummer der Zeile, in der der Fehler auftrat, wird dabei in der Systemvariablen EL gespeichert. Die Variable ER enthält die Fehlernummer, und ERR$ die Fehlermeldung im Klartext. Mit diesen Informationen kann man in der Fehlerbehandlungsroutine entsprechende Maßnahmen ergreifen und schließlich mit RESUME den normalen Programmablauf wieder aufnehmen lassen. Übrigens wird auch das Drücken der Stop-Taste mit TRAP abgefangen.

Window-Technik

Bei soviel Licht fällt gelegentlich auch ein Schatten. Die für den C 16/116 angekündigte moderne Window-Technik, also das Arbeiten mit verschiedenen Bildschirmfenstern, ist leider nicht in einer vollends überzeugenden Form implementiert.

Es kann überhaupt nur ein einziges Window erzeugt werden, und das nicht etwa per Basic-Befehl (wie man es an sich erwarten würde), sondern über eine ESC-Funktion. Damit kommen wir gleich zur Bedeutung der ESC-Taste auf der Tastatur.

Es gibt nämlich 26 ESC-Funktionen, die durch Drücken von ESC, gefolgt von einer Buchstabentaste, aufgerufen werden (Tabelle 2).

Um das eine mögliche Fenster zu erzeugen, muß man den Cursor in die linke obere Ecke des vorgesehenen Windows bringen, dann ESC T drücken, anschließend in die rechte untere Ecke fahren und ESC B betätigen. Dadurch ist das Fenster definiert. Alle Ein- oder Ausgaben spielen sich jetzt ausschließlich hier ab.

Durch zweimaliges Drücken der Home-Taste wird das Window wieder gelöscht.

So fortgeschritten das Konzept auch gegenüber dem VC 20/C 64 ist, es bleibt einiges zu wünschen übrig. Die Methode der Window-Definition ist viel zu umständlich und zu langsam, zumal ein Befehl zur direkten Cursorpositionierung nicht vorhanden ist. Das fällt um so schwerer ins Gewicht, als immer nur ein einziges Window definiert werden kann, was aber in der Regel nicht sehr sinnvoll ist. Wenn man den Bildschirm aber in verschiedene Bereiche aufteilen will, dann wirkt sich das ständige umständliche Definieren der Fenster doch zum einen auf die Programmlänge, zum anderen auf die Abarbeitungsgeschwindigkeit negativ aus.

Dennoch ist das Windowing ein Schritt in die richtige Richtung, hin zum benutzerfreundlichen Computer. Für eine übersichtliche Bildschirmaufteilung besteht jedenfalls in fast jeder Programmiersituation ein Bedarf. Wo man sich früher damit behalf, den gesamten Bildschirm bei jeder Veränderung neu aufzubauen, ist es jetzt möglich, nur den wirklich zu ändernden Bereich anzusprechen, ohne dabei die übrigen Informationen zu beeinflussen.

Maschinensprache-Monitor eingebaut

Für Maschinensprachefreunde — und solche, die es werden wollen — hält der C 16 noch einen ganz besonderen Leckerbissen parat. Er verfügt nämlich über einen fest im ROM vorhandenen Maschinensprache-Monitor, genannt TEDMON.

TEDMON ist genau genommen sogar mehr als nur ein Monitorprogramm für Maschinensprache. Er enthält nämlich einen Disassembler und auch einen kleinen Assembler. Maschinenprogramme können mit TEDMON sehr komfortabel entwickelt und anschließend als schnelle Unterroutinen von Basic aus aufgerufen werden. Tabelle 3 zeigt den TEDMON-Befehlssatz.

Fazit

Neben dem C 16 bietet Commodore noch den 116 an, also praktisch den gleichen Computer, nur im anderen Gehäuse (Bild 8). Der Preisunterschied zwischen den Geräten (real 398 Mark für den C 16, 348 Mark für den 116, der empfohlene Verkaufspreis liegt jeweils 50 Mark höher) ist so gering, daß wohl kaum jemand für 50 Mark die Nachteile der Gummitastatur beim 116 in Kauf nehmen wird (es sei denn, ein hartnäckiger Spectrum-Freund, aber für den ist das dann eh der falsche Computer).

abs	log
and	loop
asc	mid$
atn	monitor
auto	new
backup	next
box	not
char	on
chr$	open
circle	or
close	paint
clr	peek
cmd	poke
collect	pos
color	print
cont	print#
copy	printusing
cos	pudef
data	rclr
dec	rdot
def	read
delete	rem
dim	rename
directory	renumber
dload	restore
do	resume
draw	return
ds	rgr
ds$	right$
dsave	rlum
el	rnd
end	run
er	save
err$	scale
exp	scnclr
fn	scratch
for	sgn
fre	sin
get	sound
getkey	spc(
get#	sqr
gosub	sshape
goto	st
graphic	stop
gshape	str$
header	sys
hex$	tab(
if	tan
input	ti
input#	ti$
instr	trap
int	troff
joy	tron
key	until
left$	usr
len	val
let	verify
list	vol
load	wait
locate	while

Tabelle 1. der Leistungsfähige Befehlssatz des C 16/116. Die farbig unterlegten Befehle sind beim VC 20/C 64 nicht vorhanden.

⟨ESC⟩ & Taste	Funktion	⟨ESC⟩ & Taste	Funktion
A	Automatisch einfügen	O	(Off) Hebt Einfüge-, Anführungszeichen-, Reverse- und Blink-Modus wieder auf
B	(Set Bottom) Fixiert an der gegenwärtigen CURSOR-Position die rechte, untere Fensterecke	P	Löscht Bildschirmzeile vom Anfang bis zur CURSOR-Position
C	(Clear auto insert) Hebt automatisch Einfügen auf	Q	Löscht Bildschirmzeile ab der CURSOR-Position bis zum Ende
D	(Delete) Löscht eine Zeile an der CURSOR-Position	R	Verkleinert das Bildschirmformat und löscht den Bildschirm
I	(Insert) Fügt eine Zeile an der CURSOR-Position ein	T	(Set Top) Fixiert an der gegenwärtigen CURSOR-Position die linke, obere Ecke des Fensters
J	CURSOR wird an den Anfang der CURSOR-Positionszeile gesetzt	V	SCROLLEN des Bildschirminhalts nach oben
K	CURSOR wird an das Ende der CURSOR-Positionszeile gesetzt	W	SCROLLEN des Bildschirminhalts nach unten
L	Schaltet SCROLLING-Modus ein	X	(Exit ESC) Befreit Sie aus dem ESCAPE-Modus nach versehentlicher Betätigung der ⟨ESC⟩-Taste
M	Schaltet SCROLLING-Modus aus		
N	Schaltet zur normalen Bildschirmgröße zurück und löscht den Bildschirm		

Tabelle 2. Die ESC-Funktionen

A	Assemble	Wandelt ein Mnemonic (Klartext-Befehl) in den entsprechenden Maschinencode des 6502 beziehungsweise 7501
C	Compare	Vergleicht zwei Speicherbereiche und zeigt die Unterschiede
D	Disassemble	Wandelt Maschinencode in Mnemonics (Klartext)
F	Fill	Füllt einen Speicherbereich mit wählbarem Wert
G	Go	Startet Maschinenprogramm an angegebener Adresse
H	Hunt	Durchsucht einen Speicherbereich nach einem bestimmten Wert und zeigt alle Speicherplätze an, die diesen Wert enthalten
L	Load	Lädt ein Programm von Kassette oder Diskette
M	Memory	Zeigt alle Inhalte eines wählbaren Speicherbereichs in Hexdarstellung an
R	Register	Ausgabe der aktuellen Registerinhalte
S	Save	Speichert ein Programm auf Kassette oder Diskette
T	Transfer	Blockkopierbefehl, kopiert einen bestimmten Speicherbereich in einen anderen
V	Verify	Vergleicht ein Programm im Arbeitsspeicher mit einem auf Kassette oder Diskette
X	Exit	Zurück zu Basic
.	(Punkt)	Entspricht dem A (Assemble)
>	(größer als)	Ändert bis zu 8 Byte ab bestimmter Speicherstelle (nach M-Befehl)
;	(Semikolon)	Ändert die 7501-Registerinhalte (nach R-Befehl)

Tabelle 3. Die TEDMON-Befehle

Der mit 16 KByte zu kleine Anwenderspeicher (nach Einschalten der hochauflösenden Grafik bleiben noch exakt 2045 Byte zum Programmieren) dürfte bereits in naher Zukunft mit entsprechenden RAM-Modulen erweiterbar sein. Man sollte aber nicht vergessen, daß man bei der Leistungsstärke des C 16 Basic in 2 KByte etwa das gleiche an Grafik-Programm unterbringen kann wie beim C 64 in 8 KByte.

Der C 16 jedenfalls ist insbesondere vom Basic her in der Tat mindestens eine ganze Generation weiter als der VC 20 und der C 64. Durch sehr komfortable Programmierhilfen und ein umfangreiches, praxisnahes Basic ist er nicht nur der ideale Einsteiger-Computer; auch Profis fahren in der Regel lieber einen Rolls Royce, als daß sie wirklich jede Strecke zu Fuß gehen.

(ev)