Erster ausführlicher Test PC 128 (Teil 1)

Der PC 128 war bei Commodore noch im Vorbereitungsstadium für den deutschen Markt, da hat die 64'er-Redaktion sich eines der ersten Geräte über die Osterfeiertage besorgt, um den PC 128 für Sie auf Herz und Nieren zu testen. In diesem Teil beschreiben wir ausführlich die Hardware, das Basic 7.0 und die Kompatibilität zum C 64.

Nicht nur die 6510-Programmmierer kommen mit dem PC 128 auf ihre Kosten. Der PC 128 hat nämlich zusätzlich einen Z80-Prozessor eingebaut, der einen wesentlich komfortableren Befehlssatz als die 65xx-Familie besitzt, zu der auch der 8502-Prozessor im PC 128 gerechnet werden muß. Der Z80 wird im PC 128 mit 4 MHz getaktet. Dadurch, und mit dem neuen 1571-Laufwerk, das auch CP/M-Disketten lesen kann, werden endlich CP/M-Programme wie Wordstar auf einem preiswerten Computer verwendbar.

Der PC 128 unterscheidet sich vom C 64 in folgenden Punkten:

1. Geschwindigkeit

Der 8502-Prozessor des PC 128 kann mit zwei wählbaren Frequenzen getaktet werden; mit 1 MHz oder mit 2 MHz. Zur Umschaltung braucht in einem Basic-Programm nur der SLOW-(1 MHz) oder der FAST-(2 MHz-)Befehl gegeben werden.

Slow und Fast

Im SLOW-Modus werden Programme etwas langsamer als beim C 64 (auch 1 MHz) verarbeitet. Das kommt durch die komplizierte Verwaltung des 128 KByte-Speichers des PC 128 zustande. Diese Verlangsamung ist aber kein Grund, den PC 128 als lahmen Computer zu verdammen, hat man doch sehr leistungsfähige Basic-Befehle zur Verfügung. Und gerade diese können den PC 128 schneller als den C 64 machen. Denn man muß nicht mehr mit Unmengen von POKE-, PEEK- und SYS-Befehlen arbeiten, die in einem Programm viel Zeit »fressen« können. Der Prozessor braucht nun mal wesentlich mehr Taktzyklen, wenn beispielsweise der Grafik-Modus mit POKEs, anstelle mit einem einzigen Maschinenprogramm eingeschaltet wird. Denn jeder einzelne POKE-Befehl muß erst vom Basic-Interpreter erkannt und ausgewertet werden, bevor an eine Ausführung zu denken ist.

Der FAST-Modus läßt jedes Programm um das Doppelte schneller werden. Mit einem Haken allerdings: Es ist ein RGB- oder Schwarz-Weiß-Monitor erforderlich.

Während für den SLOW-Modus der Video-Interface-Chip (VIC) aus dem C 64 noch ausreicht, muß im FAST-Modus ein anderer, schnellerer Baustein verwendet werden, denn der VIC aus dem C 64 »schafft« maximal 1 MHz Taktfrequenz. Der neue Video-Controller hat die Bezeichnung 8563. Er übernimmt im FAST- und 80-Zeichen-Modus die Bilderzeugung. Im Unterschied zum VIC (er liefert ein Composite-Videosignal) stellt der 8563 leider nur ein genormtes RGB-(IBM-kompatibel) und Luminanz-(Helligkeits-)Signal bereit.

Die RGB-Norm sieht für jede Grundfarbe ein eigenes Intensitäts-(Helligkeits-)Signal vor. Dadurch wird eine wesentlich größere Bildauflösung (Punktschärfe) als bei der Composite-Norm möglich. Die Composite-Norm kennt nur zwei Signale: ein Farb- und ein Luminanzsignal.

Für den Benutzer bedeuten die zwei verwendeten Normen, daß er zwei Monitore braucht. Einen Composite-Monitor für den SLOW- und C 64-Modus (ein Fernseher reicht notfalls auch) und einen RGB- oder Schwarzweiß-Monitor (mit Luminanzeingang) für den FAST- und 80-Zeichen-Modus.

Eine einfachere Lösung bietet der neue für den PC 128 lieferbare Commodore-Monitor 1902, der mit einem Schalter von RGB auf Composite umgeschaltet werden kann.

40/80-Zeichendarstellung und Grafik

Wie bei Personal Computern schon immer üblich, kann der PC 128 auch achtzig Zeichen darstellen. Allerdings nur auf einem RGB-Monitor; unabhängig vom SLOW- oder FAST-Modus.

Ein großer Nachteil ist allerdings der fehlende Grafik-Modus bei FAST oder 80-Zeichen-Darstellung. Der dazu benötigte Video-Controller kann nämlich nur 80 Zeichen (und keine Hires-Grafik!) in einer Auflösung von 640 x 200 Punkten in 16 Farben darstellen und sonst nichts. Ein Zeichen besteht dabei aus 8 x 8 Punkten, Zeilen- und Zeichenabstand mitgerechnet. Hochauflösende Grafik, Sprites (bewegbare Objekte, die selbst definiert werden können) und zweifarbige Buchstaben sind in dieser Auflösung (620 x 200 Punkte) nicht möglich. Grafikbefehle wie DRAW, CIRCLE und BOX sind im 80-Zeichen-Modus ebenfalls nicht funktionsfähig, da eben der Video-Controller keinen eigenen Grafikprozessor wie der Video-Interface-Chip hat. Ein Grafik-Modus existiert bei 2 MHz also nicht.

Im 40-Zeichen- und SLOW-Modus übernimmt der aus dem C 64 bekannte Video-Interface-Chip (VIC) das Regiment. Für den PC 128 wurde er überarbeitet und heißt 8564. Das bedeutet, daß die grafischen Fähigkeiten des 40-Zeichen- und

In letzter Minute

haben wir den Erlkönig für den deutschen Markt zu Gesicht bekommen. Der »deutsche« PC 128 überrascht mit einem kleinen, aber sehr wichtigen Detail: er verfügt über eine deutsche Tastatur und kann auch Umlaute auf dem Bildschirm darstellen. Mehr darüber in der nächsten Ausgabe.

SLOW-Modus denen des C 64 entsprechen.

Wir haben teilweise mit zwei gleichzeitig angeschlossenen Monitoren gearbeitet, einem RGB-Farbmonitor und einem Commodore 1701. Es war schon ungewöhnlich, am 80-Zeichen-RGB-Bildschirm Grafik-Befehle zu programmieren und zu starten und auf dem 1701 (Composite-Monitor) den Grafikaufbau zu beobachten. Schaltet man nämlich im 80-Zeichen-Modus auf Grafik-Modus, wird der für die Darstellung von 80 Zeichen benötigte 8563-Video-Controller abgeschaltet und dafür der 8564-VIC mit seinem Grafikprozessor aktiviert. Alle Bildausgaben gehen dann über den Composite-Ausgang. Da der 8563-Video-Controller dann zwar nicht mehr vom Prozessor angesprochen wird, das zuletzt gesendete RGB-Bild aber nicht gelöscht wird, bleibt auf dem RGB-Monitor das 80-Zeichen-Textbild weiterhin zu sehen.

Zwei in Einem

Wie anfangs erwähnt: Der PC 128 besteht aus zwei oder eigentlich aus drei Computern. Einem C 64, einem PC 128 und einem CP/M-Computer. Für diese drei Betriebsarten stehen zwei Prozessoren zur Verfügung: ein 8502 und ein Z80. Der 8502 ist vollständig kompatibel zu dem 6510 im C 64, kann aber mit 2 MHz getaktet werden. Der Zilog Z80 bedarf eigentlich keiner Erklärung. Er wird seit vielen Jahren in vielen großen und kleinen Computern eingesetzt, sowohl im ZX-81 als auch in großen CP/M-Maschinen.

Schon beim Einschalten des PC 128 macht sich der Z80 bemerkbar. Er versucht das CP/M-Betriebssystem von Diskette zu booten (zu laden und zu starten). Ohne irdendeinen 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 PC 128-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 bestimmte I/O-Operationen der BIOS-(Betriebssystem-) Befehlssatz der CP/M nicht aus, übernimmt der 8502 diese Aufgaben. Wie weit man sich das zunutze machen kann, bleibt abzuwarten. Wir werden genaueres über den Z80 und das CP/M-Betriebssystem in der nächsten Ausgabe berichten.

Z80 und 8502 teilen sich im PC 128 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 angegeben 4 MHz getaktet wird. Das bedeutet, daß CP/M-Programme auf dem PC 128 nicht ganz so schnell laufen, wie auf einem 4 MHz-CP/M-Computer.

Schneller 8502

Der auffälligste Unterschied des 8502 zum 6510 im C 64 liegt in seiner Geschwindigkeit. Der 6510 »verträgt« nämlich nur Taktfrequenzen bis 1 MHz, der 8502 aber bis zu 2 MHz. Der 8502 kann also doppelt so schnell arbeiten. Jedes Programm, sei es Basic oder Maschinensprache, kann demnach auf dem 8502 doppelt so schnell gefahren werden wie auf dem 6510.

PC 128 — der Grafikbeschleuniger?

Diese Frage ist nur mit »Jein« zu beantworten. Einerseits schaltet, wie erwähnt, der VIC im Fast-Modus einfach ab. Der Bildschirm sieht dann genauso aus, wie beim Laden von Datasette. Andererseits werden alle Grafikbefehle im Speicher ausgeführt, so daß nach Zurückschalten auf SLOW die fertige Grafik sichtbar wird. Das gleiche gilt generell, wenn Sie nur einen Fernseher oder 1701/1702-Composite-Monitor besitzen und trotzdem mit 2 MHz Taktfrequenz arbeiten möchten. Sie müssen vor jeder Bildschirmausgabe mit dem SLOW-Befehl den 1-MHz-Modus einschalten, denn der VIC ist ja bei 2 MHz abgeschaltet. Erst nach der Umschaltung kann der VIC dann wieder Ihren Fernseher oder 1701/1702 ansteuern. Nach der Ausgabe können Sie dann mit FAST wieder den 2-MHz-Takt aktivieren. Das Programm läuft dann wieder mit doppelter Geschwindigkeit — allerdings mit abgeschalteten Composite-Bildschirm, da bei 2 MHz wieder der »simple« RGB-Video-Controller das Bild erzeugt.

Bild 4. Der PC 128 von hinten gesehen. Von rechts nach links: User-Port, RGB-Ausgang, Fernseher, Composite Video, Serieller Port, Datasetten-Anschluß, Expansion-Port für Steckmodule

Bild 7. Anschlüsse und Schalter an der rechten Seite: Netzteilanschluß, Einschaltknopf, Reset-Taster, Joystick-Ports 2 und 1

Bild 1 zeigt die Speicherorganisationen der möglichen Betriebsarten. Die Speicheraufteilung im C 64-Modus ist mit der Memory Map des C 64 vollkommen identisch: im Bereich von $A000 bis $BFFF liegt das Basic-ROM und von $E000 bis $FFFF das Kernal-ROM mit dem

Die Speicherlandschaft

40-Zeichen-Editor. $D000 bis $DFFF belegen die I/O-Bausteine (CIA und VIA) und das Character-ROM. Beide ROM-Bereiche, Basic und Betriebssystem für den C 64-Modus, sind beim PC 128 in einem 16-KByte-ROM-Baustein abgelegt, im Gegensatz zum C 64, wo Basic und Betriebssystem in je einem 8-KByte-ROM untergebracht sind.

Im PC 128-Modus wird die Aufteilung komplizierter; ist doch ein wesentlich umfangreichers Basic (Version 7.0) und ein zusätzlicher 80-Zeichen-Editor unterzubringen.

Das Basic-ROM gliedert sich in zwei Teile á 16-KByte: Basic Teil 1 und Teil 2 in der Memory Map (Bild 1 und 2). Der erste Teil liegt zwischen den Adressen $4000 und $7FFF, der zweite geht von $8000 bis $BFFF. Insgesamt sind das 32 KByte Basic-ROM. Zum Vergleich: die Basic-Version 2.0 des C 64 ist nur 8 KByte lang.

Das Betriebssystem (Kernal) des PC 128 befindet sich zwischen den Adressen $E000 und $FFFF, wie beim C 64. Das Betriebssystem enthält ein Monitorprogramm. Die Routinen für den 40/80-Zeicheneditor liegen von $C000 bis $CFFF.

Für Basic-Programme hat der PC 128 etwa 81,5 KByte mehr Platz als der C 64. Möglich wird das durch Bank-Switching zwischen zwei 64 KByte großen RAM-Bänken.

Was ist Bank-Switching?

Es ist nicht möglich, mit einem Prozessor, der 16 Adreßleitungen hat, wie alle 65xx und der 8502, mehr als 2^{16} = 65536 Speicherzellen (64 KByte) direkt zu adressieren. Will man mehr Speicher als 64 KByte »haben«, gibt es nur eine Möglichkeit, will man keinen leistungsfähigeren Prozessor verwenden: Das Bank-Switching.

Bank-Switching heißt soviel wie Speicherblock-Umschaltung.

Die 128 KByte Speicher des PC 128 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 Managment 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 Kernal oder aus einem EPROM einer Erweiterungskarte.

Die MMU ist kein toter Baustein, an dem nichts verändert werden kann. Im Gegenteil. Bei unserem Test kamen wir auf die interessantesten Ideen, die man mit diesem Baustein realisieren könnte, angefangen von einem Interrupt-gesteuerten Kopierschutz, der sich auf verschiedene Bänke verteilt, bis zu mehreren Programmen, die auf verschiedenen Bänken »sitzen«.

Im PC 128 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 nicht möglich, größere Programme auf Kosten des Variablenspeichers anzulegen. Die vollen 64 KByte pro Bank können auch nicht vollständig genutzt wer-

122365 Basic Bytes Free

den, da ein Bereich in beiden Bänken für die Zeropage, den Stack und den Bildschirmspeicher reserviert ist. Der Bereich geht bis $0400. Während ein Basic-Programm läuft, regelt die Memory Managment 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 (Bild 2).

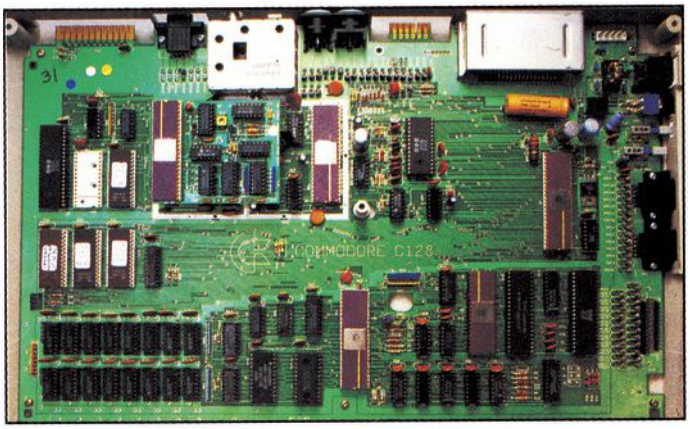


Bild 6. Ein Blick auf die Platine des PC 128. Für das Seriengerät sind allerdings noch einige Änderungen vorgesehen.

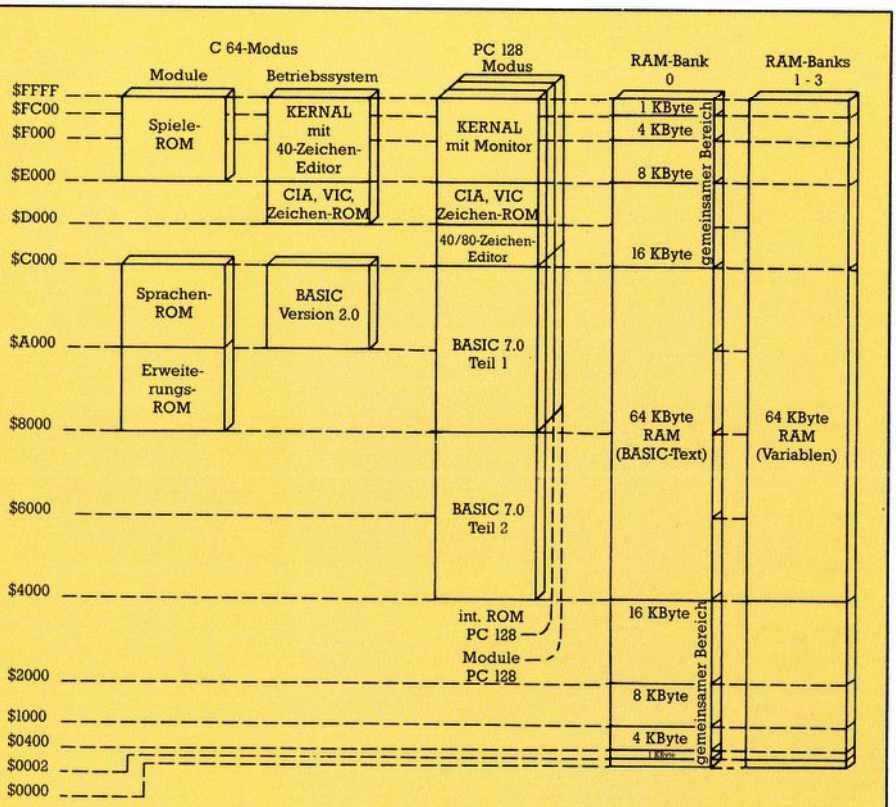


Bild 1. Die Memory Map des PC 128. Halblinks die bekannte Speicheraufteilung des C 64

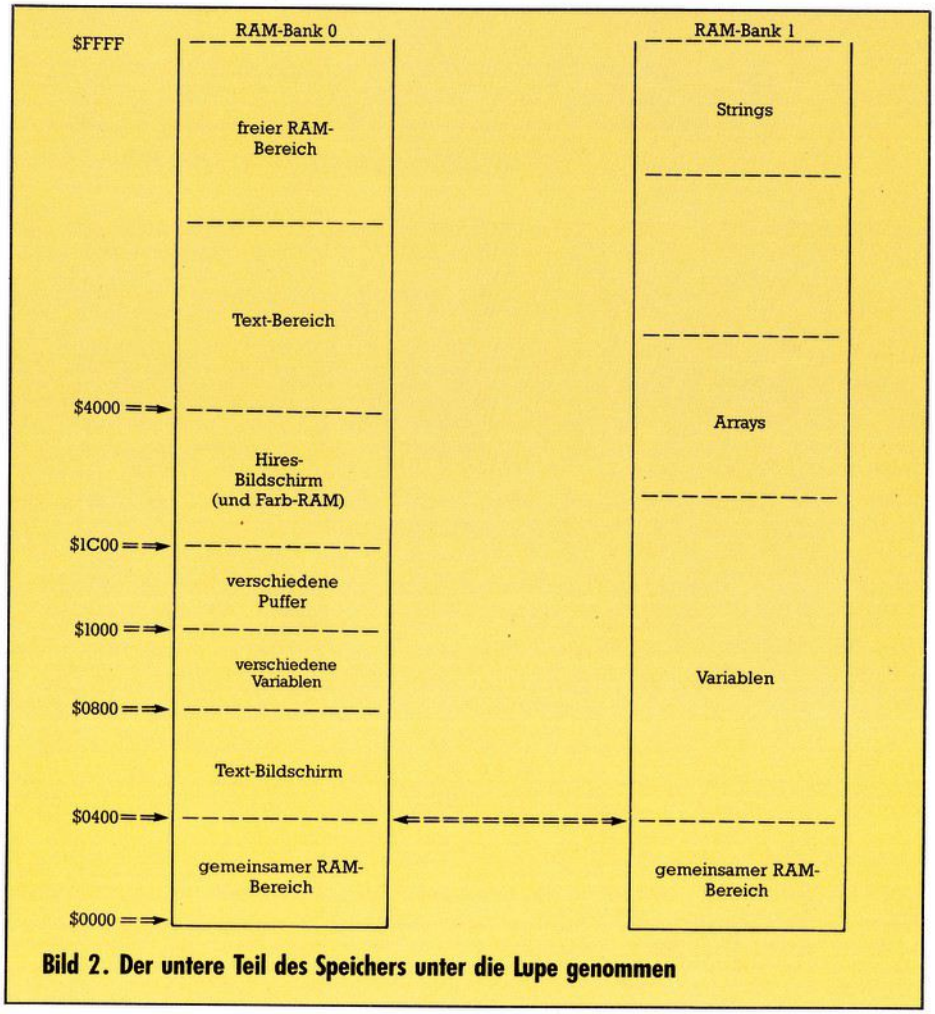


Bild 2. Der untere Teil des Speichers unter die Lupe genommen

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.

Zur Bereichswahl hat die MMU das RAM-Configuration-Register (RCR, Bild 3). Bit 0 und 1 des RCR bestimmen die Größe des gemeinsamen Speicherbereichs von Bank 0 und Bank 1: 1, 4, 8, oder 16 KByte. Sind beide Bits »0«, beträgt der gemeinsame Speicher 1 KByte, oder wenn beide »1« sind, 16 KByte.

Bit 2 und 3 des RCR bestimmen die Bereichslage. Ist Bit 2 gesetzt, liegt der gemeinsame Bereich an der Speicheruntergrenze, ist Bit 3 gesetzt liegt er an der Speicherobergrenze. Wenn beide Bits »1«, gesetzt sind, wird sowohl am Anfang und am Ende des Speichers der angegebene Bereich reserviert. Der Bereich kann also entsprechend zum jeweiligen Programm angelegt werden. Wie schon erwähnt, liegt der gemeinsame Speicher immer in Bank 0. Aber die MMU weiß durch das RCR, bei welcher Adresse die Bank gewechselt werden soll, ohne daß der Wechsel explizit im Programm angegeben werden muß.

Der Organisator — Die Memory Managment Unit

Die MMU regelt den Aufbau des Speichers. Sie bestimmt, welche RAM-Bank aktiv ist oder beim nächsten Zyklus aktiviert werden soll. Die interessantesten Register der MMU sind das schon erwähnte RAM Configuration Register (RCR, Bild 3) und das Configuration Register (CR). Das CR kontrolliert die ROM-, die RAM- und die I/O-Konfiguration des PC 128. Das Register hat die Adresse $D500 im I/O- und $FF00 im Kernal-Bereich. Das CR bei $D500 wird nur bei I/O-Zugriffen benötigt. Die MMU stellt sich dann das Register selbst ein. Findet kein I/O-Zugriff statt, ist das CR, mit den gesamten I/O-Routinen, in der Memory Map nicht vorhanden. Im Gegensatz zum CR bei $FF00, das ständig in der Memory Map präsent ist.

Bit 0 des Configuration Register (CR) regelt im PC 128-Modus den Prozessorzugriff; entweder auf den I/O-Bereich ($D000-$DFFF, High) oder auf das ROM/RAM (Low).

Bit 2 und 3 bestimmen im PC 128-Modus den Speichertyp, zwischen

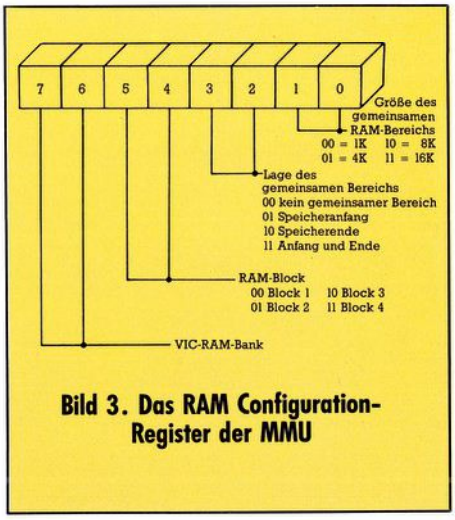


Bild 3. Das RAM Configuration-Register der MMU

den Adressen $8000 und $BFFF. Sind beide Bits »0«, wird auf Basic-High in der Memory Map, also dem zweiten Basic-ROM-Teil, zugegriffen. Ist Bit 2 »1«, wird ein internes ROM eingeblendet. In unserem Testgerät war noch ein Stecksockel auf der Platine für dieses ROM frei. Es ist jedoch geplant, dort einen deutschen Zeichensatz unterzubringen. Auch die Tastatur soll dann eine DIN-Belegung besitzen.

Interessant wird es erst richtig, wenn nur Bit 3 »1« ist, dann wird nämlich im Bereich von $8000 bis $BFFF ein Steckmodul eingeblendet. Sind beide Bits »1« sieht der PC 128 in diesem Bereich nur RAM.

Die nächsten beiden Bits, 4 und 5, haben die gleiche Funktion wie Bit 2 und 3, nur bestimmen sie den Speicheraufbau im Bereich von $C000 bis $FFFF. Zu bemerken ist, daß der Speicherbereich von $D000 bis $DFFF ein »ROM-Loch« darstellt. Man kann die Bits 4 und 5 setzen wie man will, der Computer entscheidet, ob I/O-Bereich oder das Zeichensatz-ROM in diesem Bereich eingeblendet wird.

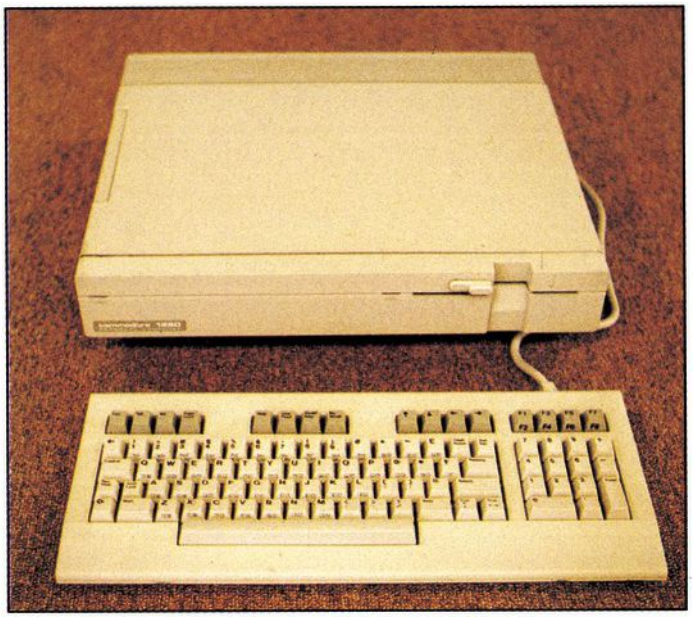
Es ist also bei einem Steckmodul zu berücksichtigen, daß der Bereich von $D000 bis $DFFF tabu ist. Es können bis zu 32 KByte ROM eingeblendet werden, doppelt soviele wie beim C 64.

Bit 6 und 7 schließlich selektieren die RAM-Bank. Für die 128-KByte-Version des PC 128 ist nur Bit 6 wichtig. Ist es »0«, ist Bank 0 ausgewählt, ist es »1«, Bank 1.

Verbindung nach draußen

Wie der C 64 hat der PC 128 einen Expansion-Port, der von beiden Betriebsmodi angesteuert wird (Bild 4).

▲ **Bild 5. Die neue Floppy-Station 1571 und der ebenfalls neue 1902-RGB/ Composite-Monitor**



▶ **Bild 8. In der Version PC 128 sind Computer und 1571-Floppy im selben Gehäuse untergebracht**

Im C 64-Modus entspricht die Pin-Belegung genau der des C 64.

Speicherzugriff erlaubt

Im Gegensatz zum C 64 erlaubt der Expansion-Port des PC 128 einen direkten Speicherzugriff (DMA, Direct Memory Access). Direkter Speicherzugriff bedeutet, daß ohne Umwege über den Prozessor in den Speicher des PC 128 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 PC 128 macht das der 8564-VIC. Er steuert den Daten- und Adreßbus so, daß Prozessor und DMA sich nicht ins Gehege kommen, was beim C 64 nicht immer sichergestellt ist. Bei einem gleichzeitigen Bus-Zugriff von Prozessor und externen Gerät erweist sich der Prozessor meist als der Schwächere, was zu ernsthaften Problemen führen kann.

Daß ein direkter Speicherzugriff ohne weiters machbar ist, eröffnet dem PC 128 gegenüber dem C 64 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-Laufwerkes. 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

Der serielle Bus und der Kassetten-Port des PC 128 sind von den Anschlüssen her identisch mit denen des C 64. Die Bedienung des seriellen Bus wurde überarbeitet, so daß der PC 128 zusammen mit dem neuen Commodore Laufwerk

Bild 9. Der PC 128 mit RGB- und Composite-Monitor. Links das Programm und rechts die dadurch erzeugte Grafik.

1571 wesentlich schneller speichern und laden kann als der C 64 (Bild 5).

In der nächsten Ausgabe berichten wir über das neue Floppy-Disk-Laufwerk 1571 und den CP/M-Modus. Ferner erfahren Sie, was die deutsche Version des C 128 mehr bietet.

GO 64 — Wie kompatibel ist der PC 128?

Kompatibilität war schon immer ein Reizwort für Commodore. Deswegen war Skepsis angesagt, ob der PC 128 wirklich kompatibel zum C 64 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-C 64 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.

Ein Blick ins Innere des PC 128 (Bild 6 und 10) zeigte auch, daß immer noch dieselben Bus-Bausteine verwendet werden. Gleichzeitig entspricht die Taktfrequenz des PC 128 im C 64-Modus der des C 64, so daß hier eigentlich auch keine Probleme erwartet wurden. 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 C 64 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.

Wir verwendeten bei unserem Test das bekannte 1541-Floppy-Laufwerk (Bild 13). Wie es sich mit dem Nachfolgemodell, der 1571, verhält, berichten wir im zweiten Teil unseres PC 128-Tests.

Getestet wurden von uns diverse Spiele wie Ghostbusters und Pit Stop II. Auch hier ein eindeutiges Ergebnis: Grafik und Musik stimmen mit dem C 64 überein. Da auch die Datasette an den PC 128 angeschlossen werden kann, standen als nächstes Kassettenprogramme auf dem Plan: Diverse Spiele, zum Teil mit Turboladern und Autostart versehen, liefen genauso problemlos wie die Diskettenprogramme.

Letzter, und unserer Ansicht nach härtester Prüfstand: Module im Expansionport. Auch hier, wie fast schon erwartet, keine Probleme, egal ob Soccer oder GBasic. Insbesondere das GBasic-Modul, das ja im Modul selbst noch eine Bank-Switching-Elektronik enthält, also zwischen zwei Speicherbausteinen hin und herschaltet, wie der PC 128 selbst, dürfte der letzte Beweis dafür sein, daß der PC 128 im C 64-Modus vollkommen software-kompatibel zum C 64 ist. Wie es mit diversen Hardwareerweiterungen, beispielsweise Turbo Access oder Speeddos aussieht, ist noch ungewiß. Sicher ist, daß hier zumindest die Platinen dieser Erweiterungen geändert werden müssen, da beim PC 128 das Kernal und das Basic in einem einzigen 16 KByte ROM und nicht, wie beim C 64, in 2 ROMs zu je 8 KByte enthalten ist.

Klares Fazit unseres Kompatibilitätstests: Wer schon einen C 64 mit Floppy 1541 hat, der kann zumindest seinen Computer beruhigt weiterverkaufen, steigt er auf den PC 128 um. Es gibt keinen Programmtyp, den wir nicht getestet haben; wir versuchten es mit Programmen, die auf übelste Art im Betriebssystem herumspringen: keine Chance. Der PC 128 ist voll kompatibel zum C 64, selbst wieder bei den Joystickanschlüssen (Bild 7).

Super-Basic 7.0

Beim Basic-Interpreter zeigt sich der PC 128 und der PC 128D (Bild 8) ohne Zweifel von einer seiner stärksten Seite: 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 1 beschrieben.

Schon eine erste, oberflächliche Betrachtung dieser Tabelle läßt eine neue Dimension der Basic-Pro-

grammierung 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 Paddles.

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 PC 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 Programms wohl für mehr als eine Minute anhalten will. Zwei weitere ungewöhnliche Befehle fallen sofort auf, nämlich SLOW und FAST. Mit diesen Befehlen kann der PC 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 C 64. Durch die komplizierte Art der Speicherverwaltung mit den verschiedenen Speicherbänken für Programme, Variablen und Betriebssystem/Basic ist das PC 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 Prinzip auch ist, die Geschwindigkeitsvorteile des FAST-Modus muß man sich mit dem bereits erwähnten Nachteil erkaufen.

Doch wenden wir uns wieder dem Basic selbst und damit erfreulicheren Dingen zu.

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 Programms 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. 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.

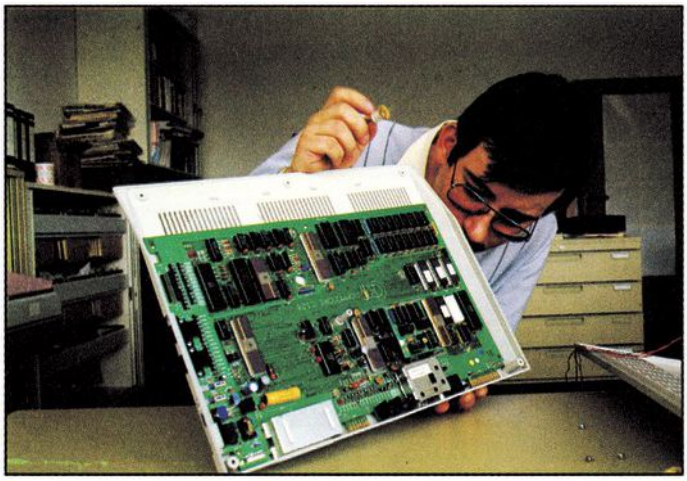


Bild 10. Hardware-Spezialist und Redakteur Harald Meyer bei der Analyse der PC 128-Platine.

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 C 64 bei jedem auftretenden Fehler unerbittlich sein Programm mit einer entsprechenden Meldung beendet, bietet der PC 128 hier einiges mehr an Flexibilität. Mit der TRAP-Anweisung können alle auftretenden Fehler während des Programmlaufes 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 gezielt 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 drücken" : 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 Programms 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 verursachte, wird dabei revers dargestellt.

Natürlich lassen sich auch von der Diskettenstation gemeldete Fehler in ähnlich 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 PC 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/Abspeichern 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.

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
```

Programmieren ohne GOTO

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

```
10 IF A$="N" 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 COMPU-
TER COMMODORE ODER
SCHNEIDER ?";C$
20 IF C$="COMMODORE" THEN
BEGIN
30 :    PRINT "PC 128 KAUFEN !"
40 : BEND : ELSE BEGIN
50 :    PRINT "VERRAETER !"
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 PC 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 C 64 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, zum 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 Anweisungen noch erhöht.

Die Grafik ist für alle da

Um hochauflösende Grafik auf dem C 64 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 PC 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 PC 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 Mehrecke 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 Text, Hochauflösung, Hochauflösung mit Textfenster, Mehrfarbengrafik, Mehrfarbengrafik mit Textfenster und schließlich Text mit 80 Zeichen pro Zeile.

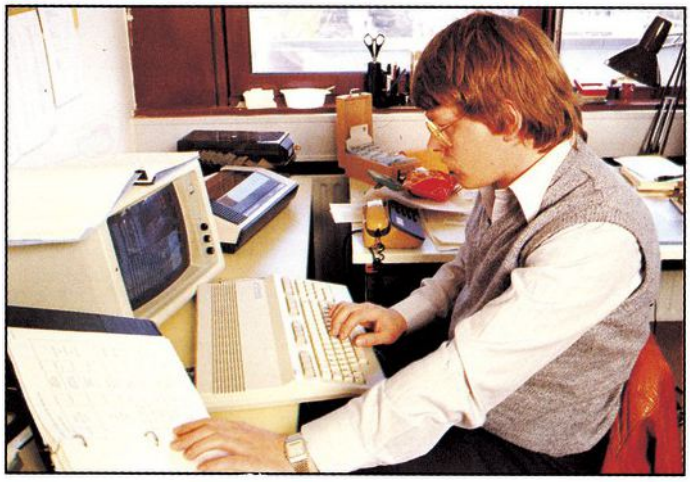


Bild 11. In unermüdlicher Kleinarbeit wühlt sich Georg Klinge durch Handbuch und Speicherorganisation.

Commodore spricht zwar von einer Auflösung von 640 x 200 Punkten, die im 80-Zeichen-Modus möglich ist, an wirklich doppelt auflösende Grafik ist dabei aber nicht zu denken: Die 640 Punkte ergeben sich als reines Rechenexempel aus 80 x 8, also 80 Zeichen mal 8 Punkte je Zeichen (Zeichenmatrix 8 x 8). Die uns beim Test vorliegende Vorab-Version des Handbuchs schweigt sich völlig über eine doppeltauflösende Grafik aus, ebenso das Hardware-Manual. Versuche ergaben, daß der GRAPHIC-Befehl tatsächlich nur mit den genannten sechs Parametern (40-Zeichen-Text, Hochauflösung, Hochauflösung mit Textfenster, Mehrfarben, Mehrfarben mit Textfenster, 80-Zeichen-Text) funktioniert, alles andere ergibt einen »Illegal Quantity Error«. Um es ganz deutlich zu sagen: Wirkliche Grafik mit einer Auflösung von 640 x 200 Punkten ist nach unseren bisherigen Erfahrungen mit dem PC 128 zumindest ohne ausgiebiges Tricksen nicht möglich. Insbesondere beziehen sich alle Grafikbefehle des 7.0-Basic ausschließlich auf die vom C 64 her bekannte 320 x 200 Punkte-Auflösung (und natürlich wahlweise auf den Mehrfarbenmodus mit 160 x 200 Punkten).

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 (Bild 9 und 12). Abhilfe schafft hier der neue 1902-Monitor von Commodore, der speziell zum PC 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 PC 128-Grafik bleibt festzuhalten, daß sie von der Auflösung her dem durch den C 64 gesetzten Standard (320 x 200 Punkte) entspricht und wie beim C 16 vorbildlich durch das Basic unterstützt wird.

Shapes, Sprites und Sprite-Editor

Wenn von Grafik die Rede ist, dürfen natürlich Shapes und Sprites nicht fehlen. Hinsichtlich dieser beweglichen Grafikobjekte ist beim PC 128 eine gelungene Synthese von C 64-Hardware und C 16-Software zu verzeichnen. Vom C 64 stammen die acht Sprites, freiprogrammierbare, 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.

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 eröffenen in Zusammenhang mit der hohen Speicherkapazität des PC 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 (0).

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-Befehle 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

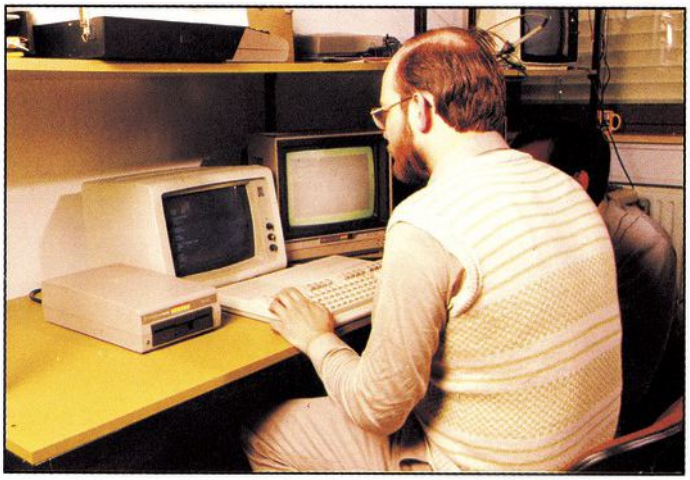


Bild 12. Das gesamte 7.0-Basic wurde von Volker Everts, dem Sprachenspezialisten, konsequent durchprobiert und ausgetestet.

Diese Befehle sind im

AUTO	Automatische Zeilennumerierung
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 die 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 (2 MHz Takt)
FETCH	Holt Daten aus beliebiger Speicherbank (RAM-Floppy)
FILTER	Setzt die Klangfilter-Parameter für den SID
GETKEY	Wartet auf Tastendruck
GO64	Schaltet in den C 64-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
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

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 C 64 kann man durch PEEKen in die Sprite-Kollisionsregister des VIC feststellen, ob ein Sprite mit einem anderen Sprite oder mit Hintergrunddaten kollidiert ist. Beim PC 128 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 leistungstarke Befehle, die ein Basic-Programm hinsichtlich der Sprite-Steuerung sehr stark entlasten. Um ein Sprite quer über den Bildschirm zu bewegen, muß man beim C 64 noch mit einer FOR... NEXT-Schleife arbeiten; um Kollisionen festzustellen, war daneben noch ein ständiges PEEKen in die Kollisionsregister des VIC nötig. Beim PC 128 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.

Basic 7.0 dazugekommen

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 Grafik-Modus
RREG	Weist Variablen die Werte der Prozessorregister zu
RSPRCOLOR	Liefert den aktuellen Code des Mehrfarbenmodus für Sprites
RSPPOS	Liefert Position und Geschwindigkeit eines Sprites
RSPRITE	Ergibt je nach Parameter alle Sprite-Attribute
RWINDOW	Liefert Parameter des eingestellten Bildschirmfensters
SCALE	Ermöglicht 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-Atribute
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 1. Die Befehle von Basic 2.0 (C 64/VC 20) sind nicht aufgeführt, aber dennoch voll im Basic 7.0 integriert

Musikalisches Basic

Ein ähnlicher Komfort ist auch bei der Programmierung des aus dem C 64 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, Soustain, 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 PC 128 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 bewandertem Einsteiger sofort eine Fülle einfach anwendbarer Klangeffekte zur Verfügung. Mit der FILTER-Anweisung können zudem alle Filtermöglichkeiten des SID zur

Klangverfremdung ausgeschöpft werden.

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.

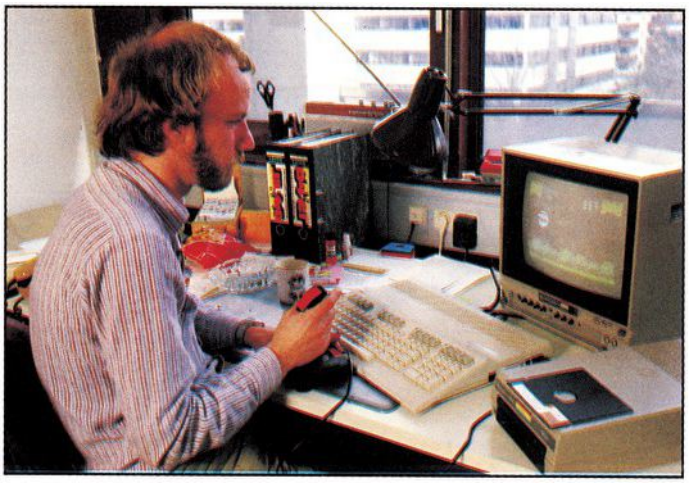


Bild 13. Christian Rogge testete die Kompatibilität des PC 128 zum C 64. Selbst undefinierte Opcodes bereiteten dem PC 128 im 64er-Modus keinerlei Probleme.

Musik per Warteschlange

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, könne 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 freigeworden sind.

Obwohl Dank des leistungsstarken Basics nur selten nötig, gibt es natürlich auch beim PC 128 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 PC 128-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 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 schließt 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.

Dieser dem C 16-»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 PC 128 natürlich nicht alle 16 Bänke belegt, einige sind für ROM-Module, andere für die RAM-Floppy reserviert.

Zurück ins Basic gelangt man mit dem X-Kommando. Und wer schließlich genug hat von komfortabler Basic-Programmierung und lieber wieder mit POKEs und DATAs arbeiten will, dem steht schließlich für alle Fälle noch der C 64-Modus offen: GO 64.

(ev/hm)