bei die Tatsache, daß der Bildschirm aus 25 x 40 = 1000 Zeichen besteht, der C 16 dem Bereich 3 aber 1 KByte, also 1024 Byte zur Verfügung stellt. Die überschüssigen Bytes können Sie mit folgendem Befehl an den Arbeitsspeicher anhängen:

```
POKE 44,15:POKE 43,233:POKE4072,0:NEW
```

Die beiden ersten POKEs setzen den Zeiger für den Basic-Anfang auf das erste freie Byte. Der dritte POKE-Befehl setzt an den neuen Basic-Anfang eine notwendige Null. Durch »NEW« wird der Computer veranlaßt, weitere interne Adressen an den neuen Basic-Anfang auszurichten. Sie haben nun 22 Byte gewonnen, der Textbildschirm steht voll zur Verfügung und Sie werden im weiteren Verlauf der Arbeit nichts von der Veränderung bemerken.

2. Wesentlich mehr, nämlich bis zu 822 Byte, stehen Ihnen mit der zweiten Möglichkeit zur Verfügung. Dabei wird der Basic-Anfang in den Textbildschirm gesetzt. Das Basic-Programm, das sich dann im Bildschirmbereich befindet, äußert sich durch einen Zeichensalat in der unteren Hälfte des Bildschirms. Als besonders vorteilhaft erweist sich hier einmal mehr die Möglichkeit des C 16, Windows zu definieren. Damit kann nämlich der Bildschirmbereich, in dem sich das Basic-Programm befindet, vor dem Überschreiben geschützt werden. Es steht Ihnen also bei dieser Möglichkeit ein Fenster in der oberen Bildschirmhälfte zur Verfügung, dessen Größe Sie je nach Bedarf selbst festlegen können. In diesem Fenster können Sie Ein- und Ausgaben durchführen, Programme schreiben, listen und testen wie gewohnt. Ein Nachteil ist wohl, daß Programme, die den Textbildschirm nutzen, eventuell so geändert werden müssen, daß sie mit dem definierten Fenster auskommen.

Wollen Sie diese Möglichkeit nutzen, so gehen Sie folgendermaßen vor:

— Überlegen Sie, wieviel Speicherplatz Sie brauchen, das heißt wie groß Ihr Fenster sein darf.
— Bringen Sie den Cursor auf das letzte Zeichen der letzten nutzbaren Zeile.
— Drücken Sie nacheinander die Tasten <ESC> und <B>.

nutzbare Zeichen	H	L	N	gewonnener Speicher
5	12	201	3272	822
10	13	145	3472	622
15	14	89	3672	422
20	15	33	3872	222

— Entnehmen Sie die entsprechenden Werte aus der untenstehenden Tabelle.
— Setzen Sie die Werte für H,L und N aus der Tabelle in den folgenden Befehl ein:

```
POKE 44,H : POKE 43, L : POKE N,0 : NEW
```

3. Die dritte Möglichkeit hat zwar leider mehr Nachteile, die aber durch den Gewinn von 2 KByte Speicher leicht wettgemacht werden. Da bei Ausnutzung dieser Speichergröße nur eine einzige Textzeile im Textmodus zur Verfügung steht, muß man folgendermaßen vorgehen: Die Programme werden im Normalmodus geschrieben, dabei darf der Grafikmodus auch vorher nicht eingeschaltet worden sein, da auch »GRAPHIC 0« den Grafikspeicher nicht für Basic freimacht, sondern nur in den Textmodus umschaltet. Ein Testen von Programmen ist also nur sehr eingeschränkt, eventuell nur durch Weglassen der »GRAPHIC«-Befehle möglich.

Nun muß in das fertige Programm eine REM-Zeile so eingebaut werden, daß die Codes der Eingaben (LOAD, RUN etc.) die in der ersten Bildschirmzeile durchgeführt werden müssen, im Programm hinter eben diese REM-Anweisung zu stehen kommen. Auf diese Weise kann trotz Betrieb der ersten Bildschirmzeile bei erweitertem Basic-Speicher dieser »am Stück« gehandhabt werden. Um die richtige Zeilennummer für diese REM-Zeile zu finden, gehen Sie folgendermaßen vor:

— Beginnen Sie mit dem Wert 5
— Zu Beginn jeder Zeile zählen Sie 5 dazu.
— In den Zeilen zählt jedes Zeichen 1, dabei gelten Basic-Befehlswörter als ein Zeichen.
— Am Ende von Zeilen dürfen keine unnötigen Leerstellen stehen.
— Die Zeilennummer und die darauffolgende Leerstelle werden nicht gezählt.
— Vor der Zeile, mit der Sie den Wert 984 erreichen, fügen Sie folgende Zeile ein: <Zeilennummer> REM {80 beliebige Zeichen} Beispiel: 980 REM EEEEEEEEEEEE... und so weiter.
— Nun speichern Sie das Programm.
— Cursor auf das letzte Zeichen der ersten Zeile setzen.
— <ESC> und dann <B> drücken.
— POKE 44,8 : POKE 43,41 : POKE 2088,0 : NEW eingeben.
— Programm laden und starten.

Anmerkung: Die Möglichkeit 1 funktioniert in allen Grafikmodi, bei den Möglichkeiten 2 und 3 dürfen nur die Grafikmodi mit ungeteiltem Bildschirm, also GRAPHIC 1 und GRAPHIC 3 benutzt werden. Ich meine, daß sich mit den dargestellten Tricks trotz der erwähnten Nachteile sehr viel anfangen läßt.

(Erwin Huber/tr)

Die (un)gleichen Brüder C 16 und Plus/4

Jeder C 16- oder Plus/4-Besitzer weiß es: Die Programme für den jeweils anderen Computer laufen auch auf dem eigenen. Zumindest die meisten. Oder doch nicht? Wir haben die Computer für Sie ausführlich verglichen.

In letzter Zeit ist viel Verwirrung um die Kompatibilität der Computer C 16/116 und Plus/4 entstanden (siehe 64'er, 9/86, Seite 67). Eindeutig geklärt war bisher nur, daß C 16 und C 116 absolut hard- und softwarekompatibel sind, da sie sich lediglich durch Gehäuse, Tastatur und eine geringfügig modifizierte Platine unterscheiden.

Weitgehend ungeklärt war dagegen ihr »verwandtschaftliches Verhältnis« zum Plus/4. Überraschend:

Beim C 16/116 handelt es sich lediglich um einen »abgemagerten« Plus/4: Die Hardware des C 16/116 ist mit der des Plus/4 — bis auf den statt 64 KByte nur 16 KByte großen Schreib/-Lesespeicher (RAM), den fehlenden User-Port mit seiner RS232-Schnittstelle und die nicht eingebaute Textverarbeitung- und Dateiverwaltungs-Software — praktisch identisch.

Gemeinsam ist den drei Computern der Mikroprozessor, der Quartz, der Tastatur-Port, der Videocontroller, das programmierbare Logik-Array sowie — die ROMs. — Ja, Sie haben richtig gelesen! Das Basic- und auch das Betriebssystem (»Kernel«)-ROM ist bei allen drei Computern absolut gleich.

Anders ausgedrückt: Man könnte zum Beispiel die beiden ROM-Bausteine aus dem C 16/116 ausbauen und in den Plus/4 setzen — dieser lief exakt wie mit seinen »eigenen« ROMs. Alle Adressen, jedes Byte, jedes Bit stimmt genau überein.

Das bedeutet: Alle PEEK-, POKE- und SYS-Befehle sind auf allen drei Computern gültig, womit die diesbezügliche Frage in dem oben erwähnten Artikel beantwortet wäre.

Auch Basic- und Maschinenprogramme laufen in der Regel unverändert; auf dem C 16/116 natürlich nur, solange die Speicherkapazität nicht überschritten wird und der User-Port beziehungsweise die RS232-Schnittstelle des Plus/4 nicht angesprochen werden.

Dennoch gibt es bei einigen Maschinenprogrammen Probleme, und zwar dann, wenn sie ursprünglich für die 16-KByte-Version geschrieben wurden, nun aber auf dem auf 32 KByte oder 64 KByte erweiterten C 16/116 oder dem Plus/4 zum Laufen gebracht werden sollen. Zur Lösung dieses Problems komme ich am Ende des Artikels, der jetzt zum besseren Verständnis zunächst einmal auf die Speicherverwaltung bei C 16, C 116 und Plus/4 eingehen soll.

Ein Computer — drei Versionen

Wichtig ist zu wissen, daß das Betriebssystem, das ja bei allen drei Computertypen absolut identisch ist, beim Einschalten beziehungsweise bei einem Reset die Speichergröße des RAMs feststellt: Dazu dient die Routine »RAMTAS« ab Adresse 62290 im ROM, sie erkennt dabei nur drei verschiedene Speicherausbaustufen: 16 KByte (beim C 16/116 ohne Speichererweiterung), 32 KByte (beim C 16/116 mit 16 KByte-Erweiterung) und 64 KByte (beim C 16/116 mit 64 KByte-Erweiterung, sowie beim Plus/4).

Ein auf 64 KByte erweiterter C 16/116 unterscheidet sich vom Plus/4 also nur durch den fehlenden User-Port mit der RS232-Schnittstelle und die eingebaute Software, ansonsten sind die beiden absolut gleich. Ich spreche deshalb im folgenden nur noch von 16-, 32- beziehungsweise 64 KByte-Version.

Auf dem Speicheraufteilungsdiagramm (Bild 1) sehen Sie die drei verschiedenen Speichergrößen nebeneinander: Der Quader im Vordergrund stellt jeweils den RAM-Bereich dar. Er reicht von Adresse 0 bis 16838 (16-KByte-Version), bis 32767 (32-KByte-Version), beziehungsweise bis 65535 (64-KByte-Version).

Die Adressen 0 bis 4096 haben bei allen Versionen die gleiche Belegung (RAM für Betriebssystem und Basic-Interpreter sowie Bildschirmspeicher). Auch die Anfangsadresse des Basic-Programm-Speichers ist bei allen drei Speichergrößen immer 4097 (hexadezimal $1001), solange der hochauflösende Grafikmodus nicht eingeschaltet ist (siehe jeweils die linke Hälfte des RAM-Blocks).

Der Basic-Speicher reicht jeweils bis kurz vor RAM-Ende: Bei der 16-KByte-Version bis 16374 (=$3FF6), bei der 32-KByte-Version bis 32758 (=$7FF6) und bei der 64-KByte-Version bis 64768 (=$FD00).

Der Einfluß der Grafik

Die Situation ändert sich, wenn der Grafikspeicher benutzt wird, der sich, wie dem Bild 1 (jeweils rechte Hälfte des RAM-Blocks) zu entnehmen ist, bei allen Versionen von Adressen 6144 (=$1800) bis 16383 (=$3FFF) — also mitten im Basic-Bereich — erstreckt.

Bei der 16-KByte-Version wird das RAM-Ende nun einfach auf Adresse 6144 herabgesetzt. Ist ein eventuell vorhandenes Basic-Programm länger als die verbleibenden 2 KByte (6144-4097 = 2047 Byte), so wird die Fehlermeldung »OUT OF MEMORY ERROR« ausgegeben.

Bei der 32- und der 64-KByte-Version wird beim Einschalten der Grafik mit dem GRAPHIC-Befehl ein eventuell vorhandenes Basic-Programm an den RAM-Bereich oberhalb des Grafikspeichers ab Adresse 16385 (=$4001) verschoben.

Durch einen Fehler in der Verschiebe- und Anpaßroutine kann es passieren, daß vor dem GRAPHIC-Befehl im Programm mit DEF FN definierte Funktionen durcheinander geraten. Die erste Programmzeile sollte deshalb immer »GRAPHIC 1:GRAPHIC 0« lauten. Dies bewirkt, daß das Basic-Programm hochkopiert wird, bevor die Funktionen definiert werden.

Mit dem Befehl GRAPHIC CLR wird der Grafikspeicher wieder freigegeben und das Basic-Programm an Adresse 4097 zurückkopiert. Dies ist auch der Grund, warum man den durch Einschalten der Grafik freigewordenen und unbenutzten Bereich zwischen 4096 und 6143 nicht verwenden sollte (zum Beispiel für Maschinenroutinen). Der GRAPHIC-CLR-Befehl überschreibt diesen Bereich beim Zurückkopieren nämlich rücksichtslos.

Trotz diesem internen »Hin- und Hergeschiebe«, von dem der Basic-Programmierer nichts merkt, laufen Programme in der Regel problemlos und unverändert auf allen drei Computertypen, sofern bei den kleineren Versionen der Speicherplatz ausreicht und beim C 16/116 die RS232-Schnittstelle und der User-Port des Plus/4 nicht angesprochen werden.

Achten Sie aber darauf, daß PEEK- und POKE- Befehle, die sich auf Adressen zwischen 16384 und 64767 und SYS-Befehle, die sich auf Adressen zwischen 16384 und 32767 beziehen, gegebenenfalls auch mit RAM belegt sind. Auch PEEK- und POKE-Befehle auf Adressen zwischen 43 und 56 (Zeiger auf Basic-Programm und Variablen) dürfen den RAM-Bereich bei der 16- und 32-KByte-Version nicht überschreiten.

Alle anderen PEEK,- POKE- und SYS-Befehle gelten für alle drei Computertypen unabhängig von ihrer Speichergröße.

Was tun?

Laufen Maschinenprogramme, vor allem Spiele, mit hochauflösender Grafik, die ursprünglich für den C 16 ohne Speichererweiterung vorgesehen waren, auf der 32- beziehungsweise 64-KByte-Version oder dem Plus/4 nicht, so sollte man folgendermaßen vorgehen: Vor dem Laden des Programms »POKE 1331,246:POKE 1332,63:SYS 32768« eingeben. Der Computer meldet sich mit »12277 BYTES FREE«. Das Betriebssystem beziehungsweise der Basic-Interpreter »glaubt« nun, daß der Speicher nur 16 KByte groß sei. Dies hat zur Folge, daß beim Einschalten der Grafik das Programm nicht mehr hochkopiert wird und ein SYS-Befehl also nicht mehr an die falsche Adresse führt.

Danach lädt man das Programm mit LOAD beziehungsweise DLOAD und startet es mit RUN. Mehr können Sie softwaremäßig nicht tun, um die 16-KByte-Version zu simulieren.

Sollte der Trick nicht funktionieren, so kann es sich eigentlich nur noch um einen Fehler im Programm oder aber um einen gemeinen Programmierer-»Gag« handeln, den ich noch kurz erklären möchte: Wie Sie dem Speicherbelegungsdiagramm der 16-KByte-Version entnehmen können, wiederholt sich der RAM-Bereich von nur 16 KByte Größe im gesamten Adreßraum des Prozessors von 64 KByte viermal hintereinander (4 x 16 = 64), während er ja bei der 64-KByte-Version durchgehend mit verschiedenen Speicherzellen versehen ist. Der Speicherzelle 0 entsprechen also bei der 16-KByte-Version den Speicherzellen 16384 ($4000), 32768 (=$8000) und 49152 (=$C000). Besitzer der 16-KByte-Version können es selbst ausprobieren: POKE 16384+1339,17 (die Cursor-Farbe ändert sich).

Die »gemeinen« Programmierer ersetzten nun zum Beispiel den Befehl »JSR $1400«, der ein Unterprogramm aufruft, durch »JSR $5400«. Bei der 32- oder 64-KByte-Version dagegen spricht er ganz verschiedene Speicherzellen an, was zum »Absturz« des Programms führt.

In diesem Fall gibt es zwei Lösungsmöglichkeiten: Entweder ändert man im Programm alle Befehle, die sich dieses Gags bedienen, was sehr aufwendig ist, oder einfacher, weil hardwaremäßig: Man baut einen Schalter in den Rechner ein, der auf 16 KByte umschaltet. Doch dazu ein andermal.

(Ulrich Schäfer/tr)

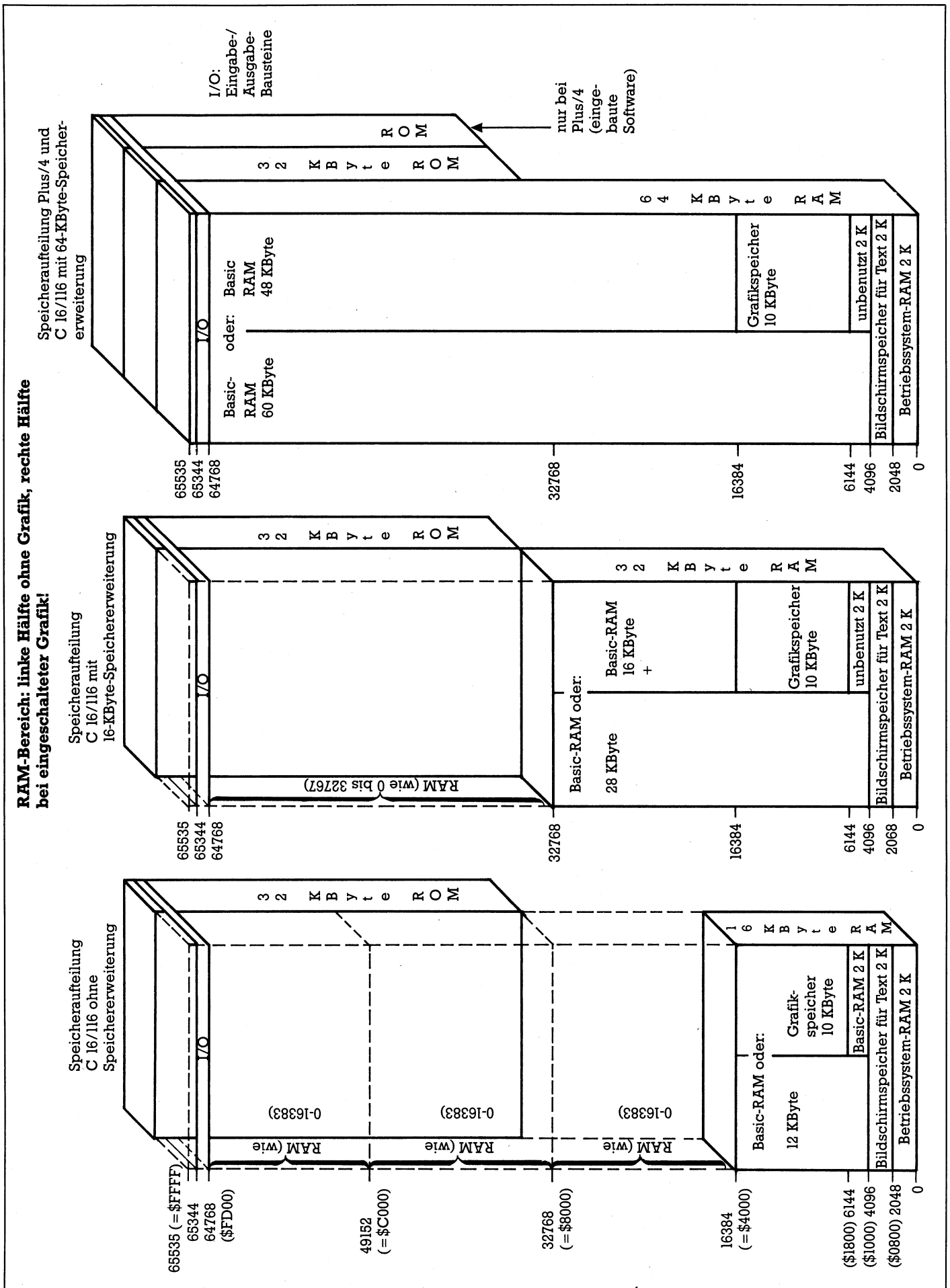


Bild 1. Die Speicheraufteilung des C 16 und Plus/4 in den verschiedenen Ausbaustufen