

Fragen und Antworten zum C16 und VC20

Hier sollen Fragen zum C16 und VC20, die sehr häufig gestellt wurden, umfassend beantwortet werden.

Maschinenprogramme kopieren?

Wie kopiert man beim VC20 ein Maschinenprogramm im Steckmodulbereich (\$A000) von einer Disk auf eine andere?

1. Speichererweiterung auf \$A000 umadressieren.
2. Maschinenprogramm absolut laden
(»LOAD "Name", 8, 1«), aber nicht starten.
3. Diskette wechseln und folgende Zeile im Direktmodus hintereinander eingeben:
POKE 43,0 : POKE 44,160 : POKE 45,0 :
POKE 46,192 : SAVE "Name", 8

Vor dem weiteren Arbeiten mit dem Computer »NEW« eingeben.

VC20-Programme auf C16?

Kann man Programme für den VC20 auch auf dem C16 verwenden?

VC20/C64-Programme laufen immer dann garantiert auf dem C16, wenn Sie keine PEEK-, POKE-, SYS- und WAIT-Befehle enthalten. In vielen Fällen handelt es sich bei POKE-Befehlen beim VC20/C64 allerdings nur um einfache Dinge wie Bildschirmfarbe einstellen oder einen Ton erzeugen. Falls ein Listing nur wenige POKES enthält, lohnt sich oft ein Versuch auf dem C16, bei dem man die POKE-, PEEK- und SYS-Befehle einfach fortläßt. Farbe und Sound kann man ja schließlich selbst noch ganz nach Belieben ins Programm einbauen.

Modulbereich nutzen?

Wie kann man beim VC20 mit 24 KByte-Erweiterung den geänderten Zeichensatz in den Modulbereich (\$A000 bis \$BFFF) verlegen? Welcher Wert muß dazu in die Speicherstelle 36869 gePOKEt werden? Muß dazu auch der Bildschirmspeicher verlegt werden?

Die Bits 0 bis 3 aus Adresse 36869 (Register 5 des VIC) bestimmen die Lage des Zeichengenerators im Speicher und zwar nach einem recht komplizierten Schema, bei dem einige Bits der generierten effektiven Adresse immer auf Null sind. Leider gehören auch die Bits 13 und 14 der Zeichenspeicher-Adresse zu diesen Null-Bits. Dadurch ist der Speicherbereich ab \$A000 (=1010 0000 0000 0000 binär) nicht als Zeichengenerator-Adresse einstellbar. Auch der Bildschirmspeicher läßt sich beim besten Willen nicht in diesen Bereich verlegen.

Speicherverlust durch Grafik?

Bei Verwendung hochauflösender Grafik verringert sich die Speicherkapazität des C16 um rund 10 KByte. Woran liegt das, und wie kann man das verhindern?

Damit die Grafik am Bildschirm sichtbar sein kann, muß sie irgendwo im Computer gespeichert sein. Jedem einzelnen Punkt am Bildschirm ist dabei ein Bit im Speicher zugeordnet. Nun hat der C16 eine Grafikauflösung von 360 x 200 Punkten, das sind insgesamt 72000 Punkte, die ebensovielen Speicherbits entsprechen. 8 Bit ergeben ein Byte, die 72000 Bildpunkte belegen also genau 8000 Byte im Speicher. Außerdem wird noch zusätzlicher Speicher benötigt, um die Farbinformationen für jede Bildschirmstelle festzuhalten, so daß insgesamt ungefähr 10 KByte benötigt werden, um überhaupt die Grafik einschalten zu können. Das läßt sich unter keinen Umständen verhindern. Abhilfe schafft hier nur eine Speichererweiterung, bei der ein Verlust von 10 KByte nicht so schwer ins Gewicht fällt wie bei den mageren serienmäßigen 16 KByte Speicher.

Probleme mit Speichererweiterung?

Spiele, die für den VC20 mit 3 KByte-Erweiterung geschrieben wurden, laufen leider nicht mit einer 8- oder 16 KByte-Erweiterung. Gibt es einen Trick, um sich dennoch das ständige Umstecken der Erweiterungen zu ersparen?

Abhilfe schafft hier zum Beispiel die Verwendung einer Modulbox, in die mehrere Steckmodule gleichzeitig eingesteckt werden können. Aber es geht auch ohne Modulbox: Mit der folgenden kurzen Routine lassen sich die meisten Grundversions- oder +3 KByte-Programme auch mit einer 8 (16) KByte-Erweiterung laden und ausführen:

```
POKE 648,30 : SYS 64821
```

```
POKE 4096, 0 : POKE 44, 16 : NEW
```

Da der Bildschirmspeicher durch die kleine Routine an der gleichen Stelle wie in der Grundversion liegt, sind die meisten Grundversions-Programme direkt lauffähig.



63299 Bytes free beim VC20?

Die Befehlsfolge »POKE 56,255 : SYS 58234« soll beim VC20 einen freien Speicherplatz von über 63000 Bytes ergeben. Sind diese 63299 Bytes wirklich frei zum Programmieren und wird dadurch auch keine andere Funktion beeinflusst?

Na, da hat Sie aber wirklich einer an der Nase herumgeführt. Der POKE-Befehl setzt zwar (allerdings nur rein rechnerisch) die Speichergrenze für den Basic-Interpreter nach oben, aber sonst tut sich überhaupt nichts, denn selbstverständlich kann dort, wo einfach kein Speicher ist, nicht durch einen simplen POKE-Befehl welcher aus dem Nichts erzeugt werden. Im Gegenteil, wenn Sie längere Programme laden oder mit Variablen arbeiten, dann kann es sogar zum Absturz des Computers kommen, weil dieser verzweifelt versucht, Programm oder Variablen in nicht vorhandenen Speicherbereichen abzulegen. Das kann natürlich nicht gutgehen.

Der SYS-Befehl hinter dem POKE ist übrigens reine Augenscheinerei, denn der bewirkt nur einen Sprung in die NMI-Routine, also dasselbe, was beim Drücken der Tastenkombination RUN/STOP-RESTORE passiert.

Assembler und andere Myssterien

Was ist ein Assembler, was ein Disassembler und was versteht man unter einem Maschinensprache-Monitor?

Wie Sie vielleicht wissen, versteht der im VC 20 arbeitende 6502-Prozessor (wie auch der 7501-Prozessor im C 16) die Programmiersprache Basic nicht direkt, sondern nur eine sehr viel einfacher (für den Prozessor) aufgebaute Sprache, eben die sogenannte Maschinensprache. Sie können mit Ihrem Computer nur deshalb in Basic arbeiten, weil der VC 20/ C 16 ein Programm fest gespeichert hat, das einen »Basic-Prozessor« simuliert. Der Prozessor selbst versteht also kein Basic, aber dieses Programm, der sogenannte »Basic-Interpreter«, simuliert einen viel leistungsfähigeren Prozessor, der Basic verstehen kann. Diese Simulation kostet natürlich Zeit, viel Zeit sogar. Daher benötigt der Computer für die Abarbeitung eines Basic-Befehls im Schnitt einige tausendstel oder sogar hundertstel Sekunden, ein Befehl in Maschinensprache dagegen nur wenige millionstel Sekunden.

Natürlich können Sie Ihren VC 20 oder C 16 auch direkt in Maschinensprache programmieren – das ist ja quasi seine Muttersprache. Allerdings geht das von Basic aus nicht ganz so einfach. Sie müssen sich klarmachen, daß ja bereits die Abfrage der Tastatur oder das Blinken des Cursors durch recht komplexe Programme in Maschinensprache realisiert werden, auch wenn Sie nichts davon bemerken. Es gibt aber für die Programmierung in Maschinensprache spezielle Hilfsprogramme. In der einfachsten Form handelt es sich dabei um Maschinensprache-Monitore, mit denen man – vereinfacht gesagt – kleine Maschinenprogramme gleich in der dem Prozessor geläufigen Zahlenform eingeben kann.

Dieses Programmieren in Zahlen ist zwar für die Maschine sehr angenehm, aber weniger für den Menschen. Daher wurden zu jedem Maschinenbefehl sogenannte »Mnemonics«, also leicht zu merkende Abkürzungen für die Wirkungsweise der Befehle, geschaffen. Um diese Mnemonics in das der Maschine geläufige Zahlenformat zu übersetzen, gibt es wiederum andere Programme – die Assembler.

Ein Disassembler schließlich tut das Gegenteil vom Assembler: Mit ihm können fertige Maschinenprogramme wieder in die Assembler-Mnemonics zurückübersetzt werden.

Kein Speicherplatz mehr?

Mein Datenverwaltungs-Programm für den C16 ist durch eine Reihe von nachträglichen Verbesserungen so umfangreich geworden, daß kein Platz mehr für die Daten im Speicher ist. Was kann ich tun?

Als erstes: Ruhe bewahren. Als zweites bieten sich zwei Methoden an, eine einfache und eine schwierige. Zunächst die Schwierige: Es ist eine altbekannte Programmierer-Erfahrung, daß Programme insbesondere durch nachträgliche Änderungen sehr umfangreich werden. Sie könnten also versuchen, ein neues Konzept für Ihr Programm zu entwerfen, das alle Änderungen berücksichtigt, aber von Grund auf neu programmiert wird. Häufig läßt sich dadurch einiges an Speicherplatz gegenüber »wildem« Änderungen einsparen. Aber auch die einfache Methode soll angesprochen werden: Wenn der Speicher nicht mehr reicht, reicht häufig mehr Speicher. Besorgen Sie sich also doch ganz einfach eine Speichererweiterung für Ihren C 16. Mit der in diesem Sonderheft vorgestellten 64 KByte-Erweiterung von Kingsoft meldet sich der C 16 nach dem Einschalten zum Beispiel mit »60671 Bytes free«. Damit dürften dann sogar noch weitere kleine Erweiterungen an Ihrem Programm zu realisieren sein...

VC 20 zum C64 aufrüsten?

Kann man den VC 20 zum C64 aufrüsten?

Der VC 20 kann nicht in dem Sinne zum C 64 aufrüstet werden, daß anschließend darauf die C 64-Software laufen würde (es sei denn, man baut eine komplette C 64-Platine in das VC 20-Gehäuse ein). Speichermäßig ist ein Ausbau natürlich möglich, die C 64-Programme laufen dann allerdings trotzdem nicht richtig, denn die gesamte Hardware des VC 20 weist große Unterschiede zum C 64 auf.

Wer C 64-Programme auf dem VC 20 laufen lassen will, sollte sich den Artikel »C 64-Programme für C 16 und VC 20« ansehen, den wir in diesem Sonderheft veröffentlicht haben.

Programmunterbrechung bei Druckerausgabe?

Bei der Druckerausgabe erscheint oft ein »DEVICE NOT PRESENT ERROR«, obwohl der Drucker angeschlossen und eingeschaltet ist. Was kann man dagegen tun?

Dieser Fehler tritt vergleichsweise oft bei Druckern auf, die direkt am seriellen Bus angeschlossen sind. Die Ursache hierfür liegt in der Konzeption des seriellen Busses begründet. Abhilfe: Statt mit »PRINT #« mit CMD arbeiten, dann tritt der Fehler nur noch äußerst selten oder überhaupt nicht mehr auf.

Sprites beim C16?

Ist es möglich, beim C16 Sprites im Textmodus softwaremäßig zu simulieren?

Der C 16 besitzt die Shape- beziehungsweise Sprite-Fähigkeit nur im Grafik-Modus, weil beim TED-Chip (der Chip, der beim C 16 für die Video-Ausgabe zuständig ist) Sprites nicht hardwaremäßig vorgesehen sind. Falls man aus Speicherplatzgründen auf den Grafik-Modus verzichten will oder muß, dann kann man sich aber recht gut wie beim VC 20 mit einem selbstdefinierten Zeichensatz behelfen, der nur etwa zwei KByte vom knappen Speicher beansprucht und dennoch schon beachtliche Grafikmöglichkeiten, gerade für den Spielbereich, bietet. Mehr Informationen zu diesem Thema bietet Ihnen der Beitrag »Schnelle Spielegrafik für den C 16« in diesem Sonderheft.

Wie kompatibel sind 6502 und 7501?

Der VC 20 hat als Prozessor den altbekannten 6502 eingebaut, der C16 dagegen den 7501. Gibt es Unterschiede im Befehlssatz beider Prozessoren?

Der 7501 ist vom Befehlssatz her identisch mit dem 6502, so daß es zumindest von daher keine Probleme etwa beim Umschreiben von Maschinenprogrammen des VC 20 für den C 16 gibt. Allerdings ist es denkbar, daß sich einige der sogenannten »undefinierten Opcodes« beider Prozessoren voneinander unterscheiden. Bei diesen undefinierten Opcodes handelt es sich um vom Hersteller nicht vorgesehene und nicht dokumentierte Maschinenbefehle, von deren Benutzung abgeraten werden muß.

DFÜ mit C16?

Ist es möglich, mit dem C16 oder Plus 4 Datenfernübertragung zu betreiben, und was benötigt man dazu?

Zur Datenfernübertragung, egal mit welchem Computer, brauchen Sie grundsätzlich eine serielle Schnittstelle (RS232C) mit entsprechender Treibersoftware und einen Akustikkoppler. Die Wahl des Akustikkopplers ist unabhängig vom verwendeten Computermodell, da der Koppler immer über eine zwischengeschaltete serielle Schnittstelle betrieben wird. Über serielle Schnittstellen zum C16 informiert Sie Ihr Commodore-Händler, der Plus 4 hat bereits eine solche eingebaut. Der eigentlich kritische Punkt ist die notwendige Betriebssoftware für den Akustikkoppler. Sie brauchen ein Terminalprogramm, um Daten über Akustikkoppler mit anderen Computern austauschen zu können.

Leider tut sich in dieser Hinsicht bei den Herstellern von Terminalprogrammen nichts in Richtung C16 oder Plus 4. Der einzig gangbare Ausweg scheint das Selberschreiben zu sein – gegebenenfalls nach Lektüre eines oder mehrerer Bücher über DFÜ.

C16 statt C64?

Ich besitze seit geraumer Zeit einen C64, bin aber einerseits mit dem sehr mageren Basic unzufrieden, andererseits reicht auch der Basic-Speicher des C64 für meine selbstprogrammierten Anwendungen so langsam nicht mehr aus. Ich überlege nun schon seit einiger Zeit, ob sich der Umstieg auf den C16 lohnt, eventuell mit 64 KByte Speichererweiterung.

Der C16 ist durch sein wirklich umfangreiches und komfortables Basic 3.5 und aufgrund seines sehr niedrigen Preises eigentlich ein idealer Computer für Leute, die selbst in Basic programmieren wollen, was man vom C64 nicht sagen kann. Dessen Stärke ist ganz klar das Spielen, aber wegen seines Minimal-Basics eignet er sich kaum für den etwas ernsthafteren Hobby-Programmierer. Das soll natürlich nicht heißen, daß man mit dem C64 keine vernünftigen Programme selbst entwickeln kann, es heißt nur, daß es schwieriger ist, länger dauert und ohne Maschinensprachekenntnisse in vielen Fällen (Grafik) sogar fast unmöglich ist, leistungsfähige Basic-Programme auf dem C64 zu entwerfen.

Der große Schwachpunkt beim C16 ist eindeutig der mit 16 KByte völlig unterdimensionierte Speicher. Doch das läßt sich mittels Speichererweiterung leicht beheben. Ein einfaches Rechenexempel: Der Computer kostet um die 150 Mark, eine Speichererweiterung auf 64 KByte kostet knapp 200 Mark, und schon hat man einen leistungsfähigen Computer mit über »60000 Bytes free« zum Programmieren. Das ist um einiges billiger, als ein C64 mit schlechtem Basic und 38 KByte frei für Basic-Programme. Allerdings soll nicht verschwiegen werden, daß die C16-Lösung wirklich nur dem eingefleischten Selbst-Programmierer empfohlen werden kann. Mit der Wahl eines C16 verzichtet man nämlich auf das größte Softwareangebot, das es auf der Welt überhaupt für einen einzelnen Computer gibt, nämlich auf das breite Spektrum der C64-Software, das von brandheißen Spielen bis hin zu Textverarbeitung und Programmiersprachen reicht.

Ob sich der Umstieg vom C64 zum C16 oder umgekehrt lohnt, muß jeder für sich selbst entscheiden. Wer einmal mit dem komfortablen 3.5 Basic programmiert hat, wird sich nur schwer an die Programmierung des C64 gewöhnen, wer andererseits einmal beispielsweise »Summer Games« oder »Elite« auf dem C64 gespielt hat, wird vermutlich in dieser Hinsicht von allen anderen 8-Bit-Computern enttäuscht sein.

Probleme beim »FensterIn«

In meinem Dateiprogramm für den C16 ist es nicht möglich, die Position des gerade definierten Windows in einer Variablen WINDOW zu speichern. Ich erhalte immer nur einen »SYNTAX ERROR«. Bei Verwendung eines anderen Variablennamens, zum Beispiel FENSTER, gibt es keinen Fehler. Woran liegt das?

Sie verwenden in Ihrem Programm lange Variablennamen, was zwar einerseits die Übersichtlichkeit und Lesbarkeit Ihres Programms erhöht, andererseits aber auch gefährlich sein kann, wie Ihr Beispiel zeigt. Der Variablenname WINDOW enthält nämlich das Basic-Schlüsselwort DO. Während der Abarbeitung einer Programmzeile, die den Variablennamen WINDOW enthält, gerät der C16 daher arg ins Schleudern. Während er versucht, den kompletten Namen der Variablen zu erkennen und ihn sich zu merken, stößt er unvermittelt nach den Buchstaben W, I und N auf den Befehl »DO«, mit dem er an dieser Stelle gar nichts anfangen kann und sich nicht anders als mit einem »SYNTAX ERROR« zu helfen weiß.

Wenn Sie also in Ihren Programmen lange Variablennamen verwenden wollen, dann sollten Sie sehr genau prüfen, ob in diesen Namen etwa ein reserviertes Basic-Wort enthalten ist. Sie können das sehr einfach testen, indem Sie einfach PRINT und den Variablennamen eingeben. Antwortet der C16 mit »OK«, dann ist alles ok.

Aber auch die »normalen« Variablennamen aus zwei Buchstaben sind nicht ohne Tücke. Folgende »kurze« Variablennamen sind ebenfalls nicht erlaubt: ST, TI, T\$, DS, DS\$, ER, EL, GO, TO, DO, IF, OR.

Fehler durch Dimensionierung?

Wenn ich bei meinem C16 im Direktmodus eingebe »DIM A(2000,4)«, dann erhalte ich sofort die Fehlermeldung »OUT OF MEMORY«. Was mache ich falsch?

Die Fehlermeldung besagt ganz einfach, daß Ihr C16 nicht genug Speicher hat, um den Befehl auszuführen. Wenn es Ihnen ein Trost ist, auch der C64 gibt bei diesem Befehl die gleiche Fehlermeldung aus. Mit »DIM A(2000,4)« veranlassen Sie den Computer, ein Zahlenfeld namens »A« mit 2001 Zeilen (0 bis 2000) zu je 5 Spalten (0 bis 4) anzulegen. Das gesamte Feld enthält dann also $2001 \times 5 = 10005$ Elemente. Das allein sieht noch gar nicht so schlimm aus, aber überlegen wir weiter. Der C16 benötigt 5 Byte Speicherplatz für jeden Zahlenwert, den er sich merken soll. Das Feld A enthält 10005 Elemente, benötigt also 50025 Bytes Speicher! Da der C16 nach dem Einschalten nur 12277 Byte frei hat (der C64 genau 38911), ist einfach nicht genug Speicher vorhanden, um den DIM-Befehl ausführen zu können.

Sie können bei Feldern einiges an Speicherplatz sparen, wenn Sie Integer-Zahlen verwenden, denn Integer-Werte benötigen zur Speicherung nur zwei Byte pro Zahl. Allerdings können Integer-Variablen nur ganzzahlige Werte im Bereich zwischen -32768 und 32768 annehmen. Die Anweisung »DIM A%(2000,4)« benötigt beispielsweise »nur« 20010 Byte Speicher statt 50025 bei Verwendung normaler Fließkommazahlen (vereinfacht: Zahlen mit Werten hinter dem Komma). Da aber auch das für den C16 noch zuviel ist, hilft nur noch eines: Sie müssen versuchen, mit weniger Feldelementen auszukommen. Die Anweisung »DIM A%(1000,5)« packt der C16 zum Beispiel lässig (anschließend sind noch über 2000 Byte frei für ein Programm, das mit diesem Feld arbeitet).