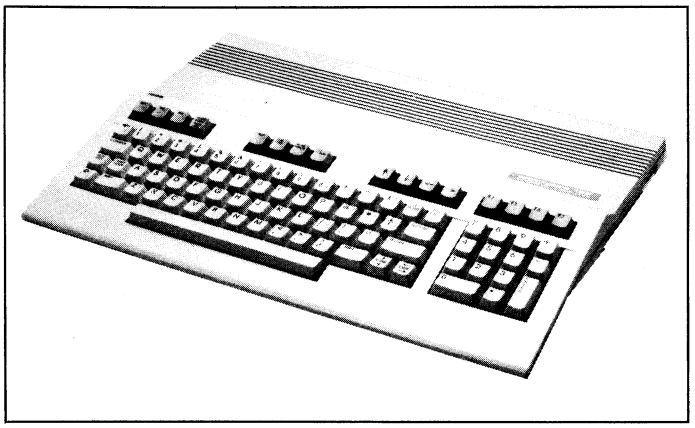
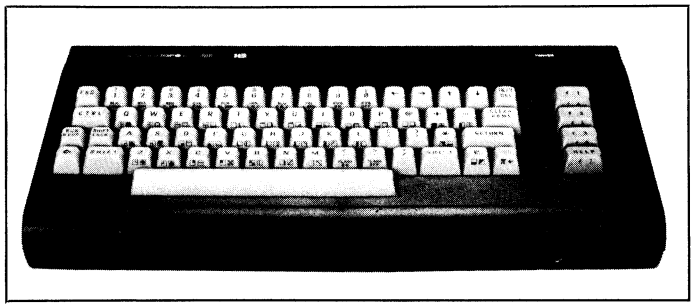
C 128-Programme auf dem C 16?

Da es für den C16 bisher noch recht wenig Software gibt, werden sich viele C16-Besitzer fragen, wie man ohne großen Aufwand Programme anderer Commodore-Computer umschreiben kann. Die Umwandlung vom Programm für den C128 auf den C16 wollen wir hier etwas näher betrachten.





Im letzten VC 20/C 16-Sonderheft (SH 3/86) wurden die Gemeinsamkeiten des C 16 und VC 20 behandelt. Dieser Vergleich ermöglichte es bestimmt vielen Lesern, VC 20-Programme an den C 16 anzupassen. Dies ist vor allem möglich, weil diese beiden Computer noch sehr viele Gemeinsamkeiten aufweisen. Eine Anpassung von C 128-Programmen auf den C 16 bedeutet allerdings etwas mehr Aufwand.

Wenn im folgenden Artikel vom C 16 gesprochen wird, ist natürlich ebenso der C 116 und Plus/4 gemeint.

Erste Schwierigkeiten treten auf, wenn man versucht, »einfache« Basic-Programme vom C 128 auf dem C 16 laufen zu lassen. Allzuoft tritt der »beliebte« SYNTAX ERROR auf. Dies liegt daran, daß der C 128 im Vergleich zum C 16 ein noch besseres Basic (Basic 7.0) besitzt. Tabelle 1 enthält den gesamten Befehlsvorrat des C 128. Die hervorgehobenen Befehle sind im Basic 3.5 des C 16 nicht enthalten. Bei einer Anpassung von Basic-Programmen müssen solche Befehle durch geeignete Routinen ersetzt oder gänzlich weggelassen werden.

Zunächst soll versucht werden, die fehlenden Befehle aufzuführen und eine eventuelle Umgehungsmöglichkeit anzubieten.

APPEND

Der Append-Befehl öffnet eine existierende, serielle Datei vom Typ SEQ, USR oder PRG als Ausgabedatei und positioniert den Schreibzeiger auf das Dateiende. Damit besteht die Möglichkeit, zusätzliche Daten in eine schon geschlossene serielle Datei zu schreiben. Der Befehl hat folgendes Format:

APPEND #log Filnr, Dateiname, D Laufw., U Geräteadr.

zum Beispiel:

```
APPEND #2,"TEXT"
```

Dasselbe kann beim C 16 mit dem normalen OPEN-Befehl erreicht werden:

```
OPEN 2,8,2,"TEXT,A"
```

BANK

Die BANK-Anweisung definiert eine von 16 möglichen 64-KByte-Speicherbänken für nachfolgende PEEK-, POKE-, SYS- oder WAIT-Anweisungen. Der BANK-Befehl erübrigt sich beim C 16, da sich seine 16-KByte-RAM-Speicher ohne weiteres über eine 16-Bit-Adreßleitung adressieren lassen.

BEGIN-BEND

Die Anweisungen BEGIN-BEND, welche einen beliebig langen Block von Basic-Anweisungen umschließen, kann man ohne weiteres mit GOTO, GOSUB oder einer IF...THEN-Anweisung umgehen.

BLOAD-BSAVE

Der Befehl BSAVE ermöglicht es, beim C 128 beliebige Speicherbereiche direkt zu speichern. Mit BLOAD kann dieser Datenblock wieder an die gleiche Stelle geladen werden. Die Befehle haben folgendes Format:

BSAVE Dateiname, D Laufwerk, U Gerät, ON B Bank, P Anfangsadresse TO P Endadresse

Sowohl die Anfangs- als auch die Endadresse werden in dezimaler Form erwartet.

BLOAD Dateiname, D Laufwerk, U Gerät, ON B Bank...

Beide Befehle können mit Hilfe einer kleinen Maschinenroutine simuliert werden. Das Programm heißt »BASICTOOL« und ist unter der Rubrik Tips & Tricks zum C 16 nochmals veröffentlicht. Mit Hilfe dieses Programms werden die Befehle BSAVE und BLOAD wie folgt simuliert:

BSAVE: SYS 1548"name",G,1,AA,EA+1

BLOAD: SYS 1536"name",G,1

Hierbei gibt G die Geräteadresse an, AA steht für die Anfangsadresse und EA für die Endadresse.

BOOT

Der Befehl BOOT erübrigt sich beim C 16, da er nicht CP/M-fähig ist und deshalb auch kein CP/M-Programm von Diskette laden und starten kann.

BUMP

Da der C 16 keine Sprites besitzt, erübrigt sich auch ein Ersatz für die BUMP-Funktion des C 128. Dieser Befehl liefert die Sprite-Nummer bei einer Sprite-Kollision.

CATALOG-DIRECTORY

Der CATALOG-Befehl listet das Directory der Diskette auf. Gleiches bewirkt auch der DIRECTORY-Befehl des C 16 und C 128. Warum man beim C 128 für ein und dieselbe Funktion zwei Befehle benötigt, ist mir ein Rätsel, zumal beide Befehle dieselbe Betriebssystemroutine verwenden.

COLLISION

Auch die COLLISION-Anweisung für die Sprites des C 128 ist auf dem C 16 nicht zu ersetzen.

CONCAT

Der CONCAT-Befehl des C 128 hängt an eine sequentielle

Datei eine andere gleichartige Datei an. Das heißt, es können auf einer Diskette mehrere Dateien des Typs SEQ zusammengefaßt werden. Dabei wird die Zieldatei, also die Datei, an die eine andere angehängt wird, verändert. Sie existiert nach dem Kommando in ihrer ursprünglichen Form nicht mehr. Die angehängte Datei bleibt dagegen weiterhin zusätzlich in der alten Form im Directory.

Zum Beispiel:

```
CONCAT "DATEN1" TO "DATEN2"
```

Hier wird die Datei DATEN1 an DATEN2 angehängt. DATEN2 wird dadurch verändert.

Beim C16 kann man den CONCAT-Befehl mit der COPY-Anweisung über den Kommandokanal der Floppy umgehen:

```
OPEN 1,8,15
PRINT#1,"CO:DATEN2=0:DATEN2,0:DATEN1
CLOSE 1
```

Auch hier wird die Datei DATEN1 an DATEN2 angehängt. DATEN2 wird dadurch verändert. Will man beim C16 eine Veränderung der Datei DATEN2 verhindern, aber trotzdem beide Dateien verknüpfen, verwendet man folgende Anweisung:

```
OPEN 1,8,15
PRINT#1,"CO:DATEN3=0:DATEN2,0:DATEN1
CLOSE 1
```

Eine Kopie der Dateien DATEN2 und DATEN1 wird zur Datei DATEN3 zusammengefaßt. Die Ursprungsdateien bleiben in ihrer alten Form unverändert.

DCLEAR

Der Befehl DCLEAR zum Schließen aller geöffneten Diskettenkanäle für ein bestimmtes Laufwerk muß im Basic 3.5 durch den Befehl CLOSE umgangen werden.

DCLOSE-DOPEN

Auch der Befehl DCLOSE zum Schließen aller Dateien auf einer Diskette muß durch CLOSE ersetzt werden.

Zur Eröffnung einer Datei auf Diskette steht beim C128 der DOPEN-Befehl zur Verfügung. So kann mit DOPEN1,...Kanal 1 zum Floppy eröffnet werden. Beim C16 wird dasselbe durch OPEN 1,8,... erreicht.

DVERIFY

Zusätzlich zu den Befehlen DSAVE und DLOAD zum Speichern und Laden von Programmen auf Diskette wurde beim C128 der DVERIFY-Befehl integriert. Während das Laden und Speichern mittels DLOAD und DSAVE beim C16 möglich ist, muß die Programmspeicherung auf Disk mit VERIFY "PROGRAMM",8 (eventuell auch noch »,1«) überprüft werden.

ENVELOPE-FILTER

Da der C16 leider keinen SID-Chip für die Tonerzeugung hat, erübrigt sich auch der ENVELOPE-Befehl zur Definition einer Hüllkurve.

Der Tongenerator des C16 ist nur über die Befehle VOL und SOUND ansteuerbar. Alle anderen beliebten Synthesizerfunktionen, die man vom C64 oder C128 kennt, sind beim C16 nicht integriert. Deshalb hat auch die FILTER-Anweisung, zum Setzen der Klangfilterparameter des SID, beim C16 keine »Daseinsberechtigung«.

FAST-SLOW

Weil man den C16 im Gegensatz zum C128 nicht auf doppelte Geschwindigkeit schalten kann, sind die Befehle FAST und SLOW, zum Wählen der Taktfrequenz zwischen 1 und 2 MHz, nicht sinnvoll.

FETCH - STASH - SWAP - GO64

Die Befehle FETCH, STASH und SWAP zur RAM-Floppy-Verwaltung sind auch beim C128 nur mit eingesteckter RAM-Floppy-Karte wirksam. Da mir nicht bekannt ist, daß von Commodore eine RAM-Floppy für den C16 angeboten wird, sind diese Befehle beim C16 unnötig.

Mit der Anweisung GO64 kann man den C128 zum C64 umschalten. Da aber der C16 nicht den Luxus von mehreren Computern in einem Gehäuse bietet, erübrigt sich auch dieser Befehl beim C16.

GRAPHIC-MOVSPR

Der GRAPHIC-Befehl zum Umschalten in die Grafik-Modi existiert bei beiden Geräten. Der C128 berücksichtigt zusätzlich die 80-Zeichen-Darstellung.

Die MOVESPR-Anweisung zur Sprite-Animation entfällt beim C16 wegen »akutem Sprite-Mangel«.

PEN-POT

Die Befehle PEN zur Abfrage des Lichtgriffels und POT zur Überprüfung des Paddles sind beim C16 nicht vorhanden und müssen bei Bedarf durch selbstgeschriebene Programme ersetzt werden.

PLAY

Die PLAY-Anweisung zum Spielen einer Tonsequenz läßt sich beim C16 sehr leicht durch ein kleines Basic-Programm ersetzen, zumal sich der simple Tongenerator mit Tonhöhe, Notenlänge und Lautstärke zufriedengibt.

POINTER

Der POINTER-Befehl gibt beim C128 die Adresse einer Variablen im Speicher an. Diese nützliche Funktion fehlt leider beim C16 und läßt sich auch nicht ohne weiteres durch ein »kleines« Basic-Programm ersetzen. Für einen geübten Programmierer dürfte aber eine entsprechende Maschinenroutine kein großes Problem darstellen.

RECORD

Zum komfortablen Positionieren des Schreib/Lese-Zeigers innerhalb einer REL-Datei, auf Diskette, steht dem C128 Benutzer der RECORD-Befehl zur Verfügung. Der RECORD-Befehl hat folgenden Syntax:

```
RECORD #LA,R,B
```

Hierbei gibt #LA die logische Adresse der Datei, R die Record-Nummer und B die Stelle im Record an. Die Anweisung RECORD #1,200 positioniert zum Beispiel den Record-Zeiger der Datei mit LA=1 auf 200.

Will man beim C16 den Schreib/Lese-Zeiger positionieren, muß man sich des Befehls POSITION im Kommandokanal der Floppy bedienen. Dieser hat folgendes Format:

P SA RL RH B

Hier gibt SA die Sekundäradresse für die Position an. Zu diesem Wert muß 96 addiert werden. Wenn Sie die Datei durch OPEN 1,8,2,..., also mit SA=2 geöffnet haben, so müssen Sie beim POSITION-Befehl für SA den WERT 98 (2+96) angeben.

RL und RH beinhalten die Record-Nummer zerlegt in Low- und High-Byte. Diese ist nötig, weil mit einem Byte nur 256 Werte dargestellt werden können, aber eine relative Datei mehr als 256 Records haben kann. Wenn R die absolute Record-Nummer ist, errechnen sich beide Parameter wie folgt:

RH = INT (R/256)
RL = R-RH*256

Der letzte Parameter stimmt mit der entsprechenden Angabe im RECORD-Befehl beim C128 überein. Er gibt auch hier die Stellung im Record an. Dieser Parameter kann in beiden Fällen nur Werte von 1 bis 254 annehmen. Alle vier Parameter sind BYTE-Parameter und müssen mit Hilfe der CHR$()-Anweisung gesendet werden.

Will man beim C16 beispielsweise eine REL-Datei mit der Kanalnummer 5 eröffnen und den Schreib/Lese-Zeiger auf Record 1000, Byte 15 stellen, so benutzt man folgende Anweisungen:

```
OPEN 1,8,15
RH = INT (1000/256)
RL = 1000-RH*256
PRINT#1,"P" + CHR$(5+96) + CHR$(RL) + CHR$(RH)
+ CHR$(15)
CLOSE 1
```

RREG

Die RREG-Anweisung des C128 weist vier Variablen die Inhalte der Prozessorregister (Akku, X-Reg., Y-Reg.) und des Statusregisters zu. Beim C16 lassen sich diese Werte durch die altbekannte PEEK-Anweisung auslesen:

A-Reg. = PEEK (2034)
X-Reg. = PEEK (2035)
Y-Reg. = PEEK (2036)
Statusreg. = PEEK (2037)

RSPRCOLOR - RSPPOS - PSPRITE

Folgende Befehle zur Sprite-Animation und Kontrolle sind, wie schon erwähnt, beim C16 nicht zu realisieren. Zur Sprite-Abfrage dienen beim C128 die Befehle RSPRCOLOR, RSPPOS und PSPRITE. Zur Sprite-Definiton und -Animation können die Basic-Befehle SPRCOLOR, SPRDEF, SPRITE und SPRSAV verwendet werden.

WINDOW - RWINDOW

Während der C128 fähig ist, mehrere Windows zu definieren, steht bei dem C16 nur ein Fenster zur Verfügung. Dieses kann zudem nur über die ESC-Taste definiert werden. Der C128 bietet dagegen komfortable Window-Befehle. Mit der Anweisung WINDOW lassen sich die Bildschirmfenster für die Textausgabe setzen.

Will man die aktuellen Parameter eines Windows abfragen, so steht dem C128-Nutzer der Befehl RWINDOW zur Verfügung. Damit lassen sich allerdings nur drei Parameter abfragen:

- letzte Zeile des Fensters
- letzte Spalte des Fensters
- Bildschirmbreite (40 oder 80 Zeichen)

Beim C16 ist der Aufwand, im Direktmodus ein Window zu definieren, keineswegs hinderlich, sondern bringt eher Vorteile, da kein Befehl »Window« am Bildschirm erscheint. Im laufenden Programm ist es jedoch sehr umständlich, ein Fenster zu generieren. Es sind sehr viele PRINT-Anweisungen notwendig.

Da die WINDOW-Funktion zweifellos in vielen C16-Programmen nützlich wäre, lohnt es sich schon, diesen Befehl im C16-Wortschatz aufzunehmen. Ein kleines Maschinenprogramm aus dem 64'er-Sonderheft 3/86, Seite 174, macht es möglich. Nach dem Initialisieren des Programms steht dem C16-Nutzer der Befehl FENSTER zur Verfügung. Mit ihm läßt sich durch Angabe von vier Parametern ein Window eröffnen. Weitere Anregungen zur Window-Technik auf dem C16 sind den Artikeln »Das Auge ißt mit« in dieser Ausgabe und »Windows - Fenster zum neuen Bedienungskomfort« im Sonderheft 7/86 zu entnehmen.

Um die Parameter des Bildschirmfensters zu erfahren bedient man sich ebenfalls der PEEK-Anweisung. In den Systemadressen lassen sich unter den Labels SCBOT, SCTOP, SCLF und SCRT die Bildschirmgrenzen abrufen:

BESCHREIBUNG	LABEL	ADRESSE
Unterer Bildrand:	SCBOT	PEEK(2021)
Oberer Bildrand:	SCTOP	PEEK(2022)
Linker Bildrand:	SCLF	PEEK(2023)
Rechter Bildrand:	SCRT	PEEK(2024)

SLEEP

Um den C16 in der Programmabarbeitung für eine gewisse Zeit anzuhalten, bedient man sich in der Regel einer FOR...NEXT-Schleife. Beim C128 kann der Computer mit SLEEP für eine bestimmte Zeit (zwischen 1 und 65535 Sekunden) »eingefroren« werden. Will man den C16 ebenfalls für eine ganz bestimmte Zeit stillegen, so kann man sich der Variablen der Systemuhr (TI und TI$) bedienen.

TEMPO

Der TEMPO-Befehl setzt das Abspieltempo für die PLAY-Anweisung beim C128. Da der C16 aber keinen PLAY-Befehl besitzt, hat auch TEMPO wenig Sinn.

WIDTH

Mit Ausnahme einer Anweisung besitzen beide Computer dieselben Grafik-Befehle. Um die Strichstärke bei allen Grafik-Befehlen zu variieren, besitzt der C128 zusätzlich den Befehl WIDTH. Mit ihm kann zwischen einfacher und doppelter Strichstärke umgeschaltet werden. Soll beim C16 doppelt-dick gezeichnet werden, muß wohl oder übel der Grafik-Befehl zweifach angewandt werden. Zwischen beiden Anweisungen sollte man natürlich die entsprechenden Parameter um jeweils ein Pixel vergrößern oder verkleinern.

XOR

Als letzter Basic-Befehl ist im Basic 7.0 zusätzlich zu den anderen logischen Operationen (and, or, not) die XOR-Anweisung hinzugekommen. Diese dient zur Exklusiv-Oder-Verknüpfung zweier Werte. Einen entsprechenden Ersatz gibt es dafür beim C16 leider nicht. Die einzige Möglichkeit wäre, die Werte in 16-Bit-Zahlen zu zerlegen und diese »von Hand« exklusiv-oder-verknüpfen. So ergibt die Anweisung: PRINT XOR (124,12) den Wert 112. Von Hand sieht die Sache so aus:

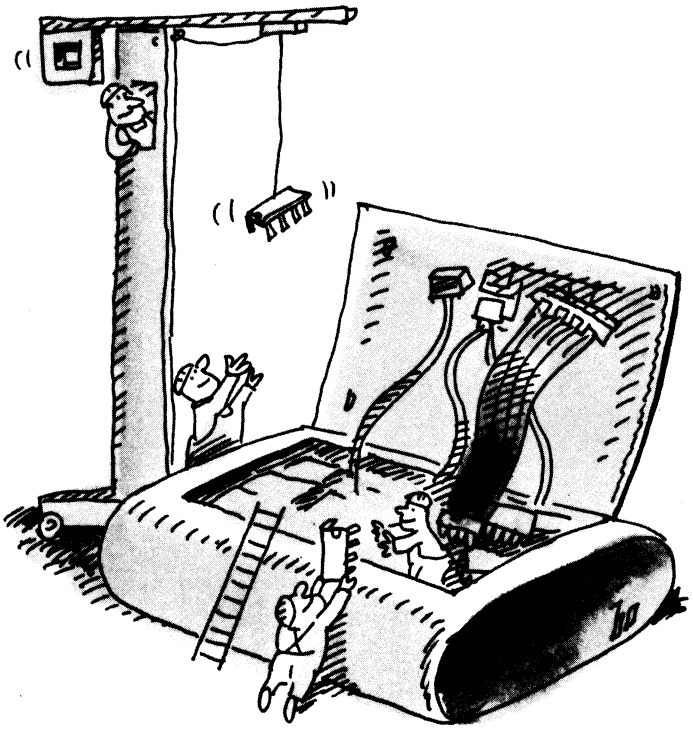
Dezimalzahl		Binärschreibweise:
	124	0000000001111100
	12	0000000000001100
XOR	112	0000000001110000

Ein entsprechendes Basic-Programm zur Umwandlung von Dezimalzahlen in Binärschreibweise und anschließende XOR-Verknüpfung wäre relativ lang und würde den Rahmen dieses Artikels sprengen.

Maschinenprogramme

Soweit zur Basic-Anpassung der C128-Programme an den C16. Bis zu diesem Punkt ist es für den C16-Anwender noch recht einfach, Programme für sein System umzusetzen. Wenn es darum geht, Maschinenprogramme umzuschreiben, wird die Sache schon komplizierter.

Zunächst werfen wir einen Blick auf die Prozessoren beider Geräte. Der C16 sowie der C128 besitzen sozusagen einen 65 xx-Prozessor. Im C16 arbeitet ein Prozessor mit der



Bezeichnung 7501. Diese Commodore-Entwicklung hat denselben Befehlsvorrat wie der 6502 des VC 20 oder 6510 des C 64. Er unterscheidet sich unter anderem dadurch, daß er das Bankswitching unterstützt. Diese Option erscheint für den C 16/116 überflüssig, da er mit seinem 16-KByte-RAM-Speicher keine Probleme haben dürfte, Betriebssystem, Basic-Interpreter und RAM-Speicher mit einer 16-Bit-Adreßleitung zu adressieren. Da aber auch der Plus/4 mit eingebauter Software und 64-KByte-RAM-Speicher denselben Prozessor besitzt, ist die Option BANKSWITCHING durchaus sinnvoll. Wenn Sie nun glauben, daß diese Funktion beim C 16 zwar vorhanden ist, aber nicht genutzt wird, so haben Sie weit gefehlt. Ein Beweis hierfür ist eine 64-KByte-Karte für den C 16. Diese stellt dem Benutzer wie beim Plus/4 fast 64-KByte-Basic-Speicher zur Verfügung. Das »Herz« des C 128, der 8502-Prozessor, ist ebenfalls eine Entwicklung von Commodore. Er wurde speziell für den C 128 entwickelt und enthält ebenfalls alle Eigenschaften der Vorgänger. Dieser Prozessor hat den Vorteil, daß er im Gegensatz zu seinen Vorgängern mit unterschiedlicher Frequenz getaktet werden kann (erinnern wir uns an die Basic-Befehle FAST und SLOW).

Ein wichtiger Teil für Maschinen-Programmierer ist die Zeropage bei beiden Computern. Dieser Bereich ist einer der wichtigsten Bereiche des Computers. Hier werden die meisten Werte von ROM-Operationen »zwischengelagert«. Außerdem liegen hier viele Betriebssystem-Vektoren. Da der Zeropage-Bereich im RAM-Bereich liegt, kann man durch Veränderung von Vektoren und Parametern fast beliebig ins »Computergeschehen« eingreifen. Aber Vorsicht: Unkontrollierte Änderungen (durch POKE) im Zeropage führen fast immer zu einem Totalabsturz!

Die Zeropage beider Computer ist ähnlich aufgebaut. Dies können sie den Tabellen 2 und 3 entnehmen. Natürlich verfügt der C 16 nicht über alle Funktionen des C 128. Im folgenden sollen die Zeropage-Adressen mit gleicher Bedeutung gegenübergestellt werden.

Die Adressen 0 und 1 haben bei beiden Computern die gleiche Funktion. Folgende Speicherstellen des C 128 (Adresse 9 bis 98) liegen beim C 16 zwei Byte tiefer (also: Adresse 7 bis 96) und haben im großen und ganzen dieselben Aufgaben. Einige Bytes werden beim C 128 mit einer Doppelfunktion belegt, die aber beim C 16 keine Rolle spielen. Auch die folgenden Speicherstellen 99 bis 105 und 106 bis 111 enthalten dasselbe, wie die C 16-Speicherstellen 97 bis 103 und 105 bis 110. Der Überlauf für den Fließkomma-Akkumulator 1 fehlt beim C 128 an der entsprechenden Stelle. Das heißt, daß im folgenden der Abstand zwischen den Speicherstellen des C 16 und C 128 auf ein Byte zusammenschrumpft: Die Zeropage-Adressen von 112 bis 126 befinden sich beim C 16 zwischen 111 und 127. Wo sich beim C 16 der Arbeitsbereich für den Sound befindet (126 bis 143), wurden beim C 128 nützliche Parameter für die Grafik eingeschoben.

Identisch sind die Adressen 144 bis 150. Der nun folgende Bereich der Zeropage (151 bis 255) enthält im großen und ganzen dieselben Parameter und Zeiger. Ihre Lage wurde aber dermaßen »durcheinandergewürfelt«, daß eine genaue Gegenüberstellung sehr umfangreich ausfallen würde. Aus den Tabellen 2 und 3 kann aber nach Bedarf die gesuchte Adresse sehr leicht entnommen werden.

Die Zeiten, in denen man die wichtigsten Systemadressen innerhalb der ersten 256 Byte unterbrachte, sind jedoch längst vorbei. Nur die alten Commodore-Geräte (PET, CBM 30xx und CBM 40xx) kamen mit der Zeropage aus. Die Nachfolger benötigten immer mehr Systemadressen für Farbe, Grafik, Sprites und so weiter. Beim C 16 wurden schließlich acht »Seiten« mit jeweils 256 Byte verwendet (Adresse 0 bis 2047) (Tabelle 4).

Der C 128 mußte bedingt durch den 40-Zeichen-Bildschirm im Bereich 1024 bis 2023 die Systemadressen bis zur Adresse 4860 ausweiten (Tabelle 5).

All diese Systemadressen zu erläutern und zu vergleichen würde dieses Sonderheft füllen. Um Ihnen einen Überblick zu ermöglichen, werden die erweiterten Zeropages in den Tabellen 4 und 5 gegenübergestellt.

Zum Schluß noch ein weiterer wichtiger Punkt für Maschinen-Programmierer. Viele Maschinenprogramme nutzen die sogenannten Kernel-Sprungadressen am Ende des Betriebssystems für vielfältige Aufgaben. Hier sind Betriebssystemroutinen adressiert, die sehr häufig benötigt werden. Die wohl bekannteste ist die »BSOUT«-Routine bei $FFD2 (Dez: 56490), die ein Zeichen am Bildschirm ausgibt. Auch beim C 128 wurden diese Sprungadressen implementiert. Das heißt, Sprünge ins Betriebssystem ab Adresse $FF81 (Dez: 56490) müssen nicht angepaßt werden.

Fazit:

Beide Computer haben mehr Gemeinsamkeiten, als zunächst angenommen. Obwohl eine Anpassung der C 128-Programme etwas aufwendiger ist als beim VC 20, ist es doch machbar. Ich hoffe, Sie haben mit diesem Artikel ein Mittel in Händen, das Ihnen hilft, diese »schwierige« Arbeit zu erleichtern. (Christian Quirin Spitzner/bj)

ABS	Liefert absoluten Wert eines Ausdrucks
AND	Logische Verknüpfung
APPEND	Öffnet eine sequentielle Datei zur Datenanfügung
ASC	Liefert ASCII Wert eines Zeichens
ATN	Liefert Arcus Tangens
AUTO	Automatische Zeilennumerierung
BACKUP	Kopiert eine komplette Diskette
BANK	Wählt Speicherbank für PEEK, POKE und SYS
BEGIN-BEND	Faßt mehrere Basic-Zeilen zu einem Block zusammen
BLOAD	Lädt beliebigen Speicherbereich von Floppy
BOOT	Lädt und startet CP/M von Diskette
BOX	Zeichnet Rechteck
BSAVE	Speichert beliebige Speicherbereiche auf Floppy
BUMP	Liefert die Spritenummer bei Sprite-Kollisionen
CATALOG	Listet Inhaltsverzeichnis der Diskette
CHAR	Fügt Text in hochauflösende Grafik ein
CHR$	Liefert Zeichenkette
CIRCLE	Zeichnet Kreise, Ellipsen und Vielecke
CLOSE	Schließt einen Ein-/Ausgabe-Kanal
CLR	Löscht alle Variablen
CMD	Adressiert ein Gerät an einer Ein-/Ausgabe-Schnittstelle
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
CONT	Setzt Basic-Programm, das durch Drücken der STOP-Taste beendet wurde, fort
COPY	Kopiert Disketten-Dateien
COS	Liefert den Cosinus
DATA	Speichert Daten im Basic-Programm für READ-Anweisung
DCLEAR	Schließt alle Kanäle zur Diskettenstation
DCLOSE	Schließt Kanal zur Diskettenstation
DEC	Liefert Dezimalwert einer Hexadezimalzahl
DEF	Definiert und benennt eine Funktion
DELETE	Löscht einen Zeilenbereich aus einem Programm
DIM	Definiert und reserviert die maximale Anzahl von Elementen einer Feld-Variablen
DIRECTORY	Listet Inhaltsverzeichnis der Diskette
DLOAD	Lädt ein Programm von Diskette
DO...LOOP	Programmschleife LOOP springt immer zu DO zurück
DOPEN	Öffnet Kanal zur Diskettenstation
DRAW	Setzt Punkte oder zeichnet Linien
DS	Enthält den Fehlerstatus des Diskettenlaufwerks

Tabelle 1. Befehlsvorrat des C 16 und C 128. Die hervorgehobenen Befehle existieren nur im Basic 7.0 des C 128.

DS$	Enthält Fehlerstatus der Floppy im Klartext
DSAVE	Speichert ein Programm auf Diskette
DVERIFY	Überprüft Programmspeicherung auf Diskette
EL	Enthält Zeilennummer bei Auftreten eines Fehlers
ELSE	Alternative bei IF-THEN, falls Bedingung nicht erfüllt
END	Beendet Programm
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
EXP	Liefert Potenz der Zahl e
FAST	Schaltet auf doppelte Geschwindigkeit
FETCH	Holt Daten aus beliebiger Speicherbank (RAM-Floppy)
FILTER	Setzt den Klangfilterparameter für den SID
FNxx	Bearbeitet Funktion xx
FOR..NEXT	Programmschleife mit einer definierten Zahl von Durchläufen
FRE	Liefert freien Speicherplatz
GET	Holt ein Zeichen von der Tastatur
GET#	Liest Zeichen aus einer Datei
GETKEY	Wartet auf Tastendruck und holt Zeichen
GO64	Schaltet in den C64-Modus
GOSUB	Verzweigt in ein Unterprogramm
GOTO	Verzweigt zu einer bestimmten Zeile
GRAPHIC	Wählt Grafik-Modus
GSHAPE	Schreibt ein SHAPE aus einem String auf den Bildschirm
HEADER	Dient zur Formatierung von Disketten
HELP	Listet nach Fehlermeldung die Fehlerzeile am Bildschirm auf
HEX$	Wandelt Dezimalzahl in Hexadezimal-Strings
IF	Stellt Bedingungen und verzweigt im Programm
INPUT	Erlaubt Dateneingabe über die Tastatur
INPUT#	Liest Daten aus einer sequentiellen oder relativen Datei
INSTR	Ergibt Position eines Teilstrings in einem anderen String
INT	Rundet auf eine ganze Zahl ab
JOY	Fragt Joystick-Position ab
KEY	Dient zur Belegung der Funktionstasten
LEFT$	Holt den linken Teil einer Zeichenkette
LEN	Ermittelt die Länge einer Zeichenkette
LET	Weist einer Variablen einen Wert zu (LET ist nicht zwingend)
LIST	Listet Programm auf
LOAD	Lädt ein Programm
LOCATE	Positioniert den Grafikcursor
LOG	Liefert den natürlichen Logarithmus
DO...LOOP	Programmschleife LOOP springt immer zu DO zurück
MID$	Holt einen Teilstring mitten aus einer anderen Zeichenkette
MONITOR	Ruft den eingebauten Maschinensprache-Monitor auf (TEDMON)
MOVESPR	Bewegt einen Sprite über den Bildschirm
NEW	Löscht Programm und alle Variablen
NOT	Logische Verknüpfung
ON..GOSUB	Verzweigt zu einem von mehreren Unterprogrammen in Abhängigkeit einer Variablen
ON...GOTO	Verzweigt zu einer von mehreren spezifizierten Zeilennummern in Abhängikeit einer Variablen
OPEN	Eröffnet einen Ein-/Ausgabe-Kanal oder eine Datei
OR	Logische Verknüpfung
PAINT	Füllt einen Bereich der hochauflösenden Grafik aus
PEEK	Liefert den Inhalt einer Speicherstelle
PEN	Fragt Lightpen ab
PLAY	Spielt die in einem String abgelegte Tonfolge
POINTER	Ergibt die Adresse einer Variablen im Speicher
POKE	Schreibt einen Wert zwischen 0 und 255 in eine Speicherstelle
POS	Liefert die Spaltenposition des Cursors
POT	Fragt Paddles ab
PRINT	Gibt Daten am Bildschirm aus
PRINT#	Gibt Daten über einen spezifizierten Ausgabekanal aus
PUDEF	Definiert Steuerzeichen für PRINT USING
RCLR	Liefert gewählten Farbcode für Text und Grafik
RDOT	Liefert Bildschirmposition des Grafik-Cursors
READ	Liest Daten aus DATA-Anweisung
RECORD	Positioniert Schreib-/Lesezeiger bei relativen Dateien
REM	Dient zur Dokumentation eines Programms
RENAME	Dient zum Umbenennen von Diskettendateien
RENUMBER	Numeriert das Basic-Programm neu
RESTORE	Setzt DATA-Zeiger auf beliebige Zeilennummer
RESUME	Rückkehr aus einer Fehlerbehandlungsroutine

Tabelle 1. Befehlsvorrat des C16 und C128. Die hervorgehobenen Befehle existieren nur im Basic 7.0 des C128 (Fortsetz.).

RETURN	Kehrt aus Unterprogramm zurück
RGR	Liefert die Nummer des eingestellten Grafikmodus
RIGHT$	Holt den rechten Teil einer Zeichenkette
RLUM	Gibt die der Farbzone zugewiesene Farbintensität an
RND	Liefert eine Zufallszahl zwischen 0 und 1
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
RUN	Startet ein Basic-Programm
RWINDOW	Liefert Parameter des eingestellten Bildschirmfensters
SAVE	Speichert ein Basic-Programm
SCALE	Maßstabswahl bei hochauflösender Grafik
SCNCLR	Löscht Text- oder Grafikbildschirm
SCRATCH	Löscht eine Diskettendatei
SGN	Liefert Vorzeichen eines Argumentes
SIN	Liefert Sinus
SLEEP	Hält die Programmausführung für eine wählbare Zeit an
SLOW	Schaltet auf 1 MHz Takt
SOUND	Erzeugt Toneffekte mit wählbarer Frequenz und Dauer
SPC	Liefert eine bestimmte Anzahl von Leerzeichen (Spaces) in der PRINT-Anweisung
SPRCOLOR	Setzt im Mehrfarben-Modus Farben für Sprites
SPRDEF	Ruft den integrierten Sprite-Editor auf
SPRITE	Setzt Sprite-Attribute
SPRSAV	Speichert ein Sprite in einem String oder umgekehrt
SQR	Liefert Quadratwurzel
SSHAPE	Speichert ein Shape in eine Stringvariable
ST	Liefert Statusbyte
STASH	Überträgt Daten in eine Speicherbank (RAM-Floppy)
STEP	Gibt Schrittweite bei FOR...NEXT-Schleife
STOP	Bricht Programm ab
STR$	Liefert Zeichenkettendarstellung einer Zahl
SWAP	Tauscht Daten zwischen zwei Speicherbänken aus
SYS	Maschinensprache-Aufruf
TAB	Tabuliert den Cursor
TAN	Liefert Tangens
TEMPO	Setzt Abspieltempo für PLAY-Anweisung
THEN	Führt Bedingung nach IF-Anweisung aus
TI	Liefert Stand der internen Systemuhr
TI$	Uhrzeit im Klartext (HHMMSS für Stunden, Minuten, Sekunden)
TO	Gibt das Endziel einer FOR...NEXT-Schleife an (FOR X=1 TO 1000...)
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...)
USING	Erlaubt formatierte Zahlenausgabe
USR	Hilfsmittel zur Übergabe und Bearbeitung einer Variablen an ein Maschinenprogramm
VAL	Liefert den numerischen Wert einer Zeichenkette, die ausschließlich aus Zahlen besteht
VERIFY	Überprüft Programmspeicherung
VOL	Setzt Lautstärke für die SOUND-Anweisung
WAIT	Wartet, bis eine Speicherstelle ein spezifiziertes Bitmuster angenommen hat
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. Befehlsvorrat des C16 und C128. Die hervorgehobenen Befehle existieren nur im Basic 7.0 des C128 (Schluß).

Dezimal	Hexadezimal	Bemerkung
0	$0000	Datenrichtungsregister des 7501
1	$0001	Ein-/Ausgabe-Port des 7501
2	$0002	Flag für Schleifen
3- 4	$0003-0004	Neue Startadresse (Renumber)
5- 6	$0005-0006	Schrittweite (Renumber)
7	$0007	Gesuchtes Zeichen
8	$0008	Flag für Anführungszeichen-Modus
9	$0009	TAB-Spaltenzähler
10	$000A	Flag: 0 = Load, 1 = Verify

Tabelle 2. Die Zeropage beim C16

Dezimal	Hexadezimal	Bemerkung
11	$000B	Zeiger für Eingabepuffer, Anzahl der Elemente
12	$000C	Flag für Standard-DIM
13	$000D	Datentyp: $FF=String, $00=Numerisch
14	$000E	Datentyp: $80=Integer, $00=Fließkomma
15	$000F	Flag für DATA/LIST
16	$0010	Flag: Element/FNx-Flag
17	$0011	Flag: $00=INPUT, $40=GET, $98=READ
18	$0012	Flag: Vorzeichen des ATN
19	$0013	Flag: Input-Prompt
20- 21	$0014-0015	Int. Adresse
22	$0016	Zeiger auf temporären Stringstapel
23- 24	$0017-0018	Letzter temporärer Stringvektor
25- 33	$0019-0021	Stapel für temporäre Strings
34- 35	$0022-0023	Bereich für Hilfszeiger 1
36- 37	$0024-0025	Bereich für Hilfszeiger 2
38	$0026	Bereich für Produkt bei Multiplikation
39	$0027	Bereich für Produkt bei Multiplikation
40	$0028	Bereich für Produkt bei Multiplikation
41	$0029	Bereich für Produkt bei Multiplikation
42	$002A	Bereich für Produkt bei Multiplikation
43- 44	$002B-002C	Zeiger auf Basic-Anfang
45- 46	$002D-002E	Zeiger auf Variablen-Anfang
47- 48	$002F-0030	Zeiger auf Beginn der Arrays
49- 50	$0031-0032	Zeiger auf Ende der Arrays (+1)
51- 52	$0033-0034	Zeiger auf Stringspeicher
53- 54	$0035-0036	Hilfszeiger für Strings
55- 56	$0037-0038	Zeiger auf Speichergrenze
57- 58	$0039-003A	Laufende Basiczeilennummer
59- 60	$003B-003C	Textpointer
61- 62	$003D-003E	Zeiger auf Basic-Statement für CONT
63- 64	$003F-0040	Nummer der aktuellen DATA-Zeile
65- 66	$0041-0042	Adresse des aktuellen DATA-Elementes
67- 68	$0043-0044	Sprungvektor für INPUT
69- 70	$0045-0046	Aktueller Variablenname
71- 72	$0047-0048	Adresse der aktuellen Variablen
73- 74	$0049-004A	Variablenzeiger für FOR...NEXT
75- 76	$004B-004C	Zwischenspeicher für Basic-Zeiger
77	$004D	Akkumulator für Vergleichssymbole
78- 79	$004E-004F	Arbeitsbereich (Zeiger etc.)
80- 81	$0050-0051	Arbeitsbereich (Zeiger etc.)
82	$0052	Arbeitsbereich (Zeiger etc.)
83	$0053	Grafik-Modus
84- 86	$0054-0056	Sprungvektor für Funktionen
87- 96	$0057-0060	Bereich für numerische Operationen
97	$0061	Fließkomma-Akkumulator 1 (FAC): Exponent
98-101	$0062-0065	Fließkomma-Akkumulator 1 (FAC): Mantisse
102	$0066	Fließkomma-Akkumulator 1 (FAC): Vorzeichen
103	$0067	Zeiger für Polynom-Auswertung
104	$0068	Fließkomma-Akkumulator 1 Überlauf
105-110	$0069-006E	Fließkomma-Akkumulator 2 Exponent etc.
111	$006F	Vorzeichenvergleich Akku 1 mit Akku 2
112	$0070	Fließkomma-Akkumulator 1 niederwertige Stelle
113-114	$0071-0072	Kassettenpuffer Länge/Zeiger
115-116	$0073-0074	Automatisches Zeileninkrement 0=AUS
117	$0075	Grafik-Flag (0=Text, 255=Grafik)
118-120	$0076-0078	Arbeitsbereich
121-123	$0079-007B	Darstellung von DS$
124-125	$007C-007D	Basic-Pseudo-Stack-Pointer
126-143	$007E-008F	Arbeitsbereich (Sound)
144	$0090	Statuswort ST
145	$0091	Flag: Stop-Taste / RVS-Taste
148	$0094	Flag: Serieller Bus — Ausgabe Zeichenpuffer
149	$0095	Zeichenspeicher — Serieller Bus
150	$0096	Zwischenspeicher
151	$0097	Anzahl der geöffneten Files
152	$0098	Eingabegerät (Normalwert: 0)
153	$0099	Ausgabegerät (Normalwert: 3)
154	$009A	Flag: $80=Direkt-, $00=Programm-Modus
157-158	$009D-009E	Zeiger auf Ende des Programms
159-160	$009F-00A0	Temporärer Datenspeicher

Tabelle 2. Die Zeropage beim C16 (Fortsetzung)

Dezimal	Hexadezimal	Bemerkung
161-162	$00A1-00A2	Temporärer Datenspeicher
163-165	$00A3-00A5	Echtzeit-Uhr
166	$00A6	Register für seriellen Bus
167	$00A7	Register für Kassettenroutine
168	$00A8	Register für seriellen Bus
169	$00A9	Temporärer Farb-Vektor
170	$00AA	Register für Kassettenroutine
171	$00AB	Anzahl der Zeichen im Filenamen
172	$00AC	Aktuelle logische File-Nummer
173	$00AD	Aktuelle Sekundär-Adresse
174	$00AE	Aktuelle Geräte-Nummer
175-176	$00AF-00B0	Zeiger auf Filenamen
177	$00B1	Von Fehlerroutine benutzt
178-179	$00B2-00B3	I/O Startadresse
180-181	$00B4-00B5	Basis-Ladeadresse
182-183	$00B6-00B7	Zeiger: Anfang Kassettenpuffer
186-187	$00BA-00BB	Zeiger auf Zeichen im Kassettenpuffer
190-191	$00BE-00BF	Register für Long-Fetch-Routine
192-193	$00C0-00C1	Register für Scrolling
194	$00C2	RVS-Flag (Bildschirm)
195	$00C3	Zeiger: Ende der Zeile (Input)
196-197	$00C4-00C5	Cursorposition (Input), Reihe/Spalte
198	$00C6	Tastaturabfrage ($40=keine Taste)
199	$00C7	Flag: Eingabe Bildschirm/Tastatur
200-201	$00C8-00C9	Zeiger auf Bildschirmzeile
202	$00CA	Cursorposition in aktueller Bildschirmzeile
203	$00CB	Flag: Anführungszeichen-Modus ($00=nein)
204	$00CC	Länge der aktuellen Bildschirmzeile
205	$00CD	Zeile, in der sich der Cursor befindet
206	$00CE	Letztes Zeichen (I/O)
207	$00CF	Anzahl der Zeichen (Insert-Modus)
208-215	$00D0-00D7	Reserviert für Software
216-232	$00D8-00E8	Reserviert für Anwendungssoftware
233	$00E9	Arbeitsbereich
234-235	$00EA-00EB	Bildschirm-Editor (Farbe)
236-238	$00EC-00EE	Arbeitsbereich (Bildschirm)
239	$00EF	Anzahl der Zeichen im Tastaturpuffer
241-242	$00F1-00F2	Register für Monitor
250	$00FA	Register für X bei Stoptastentest
251	$00FB	Aktuelle Bank-Konfiguration
254	$00FE	Arbeitsregister (Editor)

Tabelle 2. Die Zeropage beim C16 (Schluß)

Dezimal	Hexadezimal	Bemerkung
0	$0000	Datenrichtungsregister des 8502
1	$0001	Datenregister des 8502
2	$0002	BANK-Teil des PC bei Registeranzeige (PC= Programmzähler)
3	$0003	MSB des PC bei Registeranzeige
4	$0004	LSB des PC
5	$0005	Prozessorstatus SR bei Registeranzeige
6	$0006	Akkumulator A bei Registeranzeige
7	$0007	X-Register X bei Registeranzeige
8	$0008	Y-Register Y bei Registeranzeige
9	$0009	Stapelzeiger SP bei Registeranzeige
9	$0009	Suchzeichen (sucht ':' oder Zeilenende)
10	$000A	Hochkommaflag
11	$000B	Spaltenspeicher beim TAB-Befehl
12	$000C	Flag: 0=Load, 1=Verify
13	$000D	Zeiger für Eingabepuffer, Anzahl der Elemente
14	$000E	Flag für Standard-DIN
15	$000F	Datentyp: $FF=String, $00=Numerisch
16	$0010	Datentyp: $80=Integer, $00=Fließkomma
17	$0011	Flag für DATA/LIST/Garbage-Collection
18	$0012	Flag: Element/FNxx-Flag
19	$0013	Flag: $00=INPUT,$40=GET,$98=READ
20	$0014	Flag: Vorzeichen des ATN
21	$0015	Aktuelles Ein-/Ausgabegerät
22- 23	$0016-$0017	Integerwert für Zeilennummer o. 2 Byte-Adr. für GOTO, GOSUB, POKE, PEEK, SYS, WAIT
24	$0018	Zeiger auf temporären Stringstapel

Tabelle 3. Die Zeropage beim C128

Dezimal	Hexadezimal	Bemerkung
25- 26	$0019-$001A	Letzter temporärer Stringvektor
27- 35	$001B-$0023	Stapel für temporäre Strings
36- 37	$0024-$0025	Bereich für Hilfszeiger 1
38- 39	$0026-$0027	Bereich für Hilfszeiger 2
40	$0028	Bereich für Produkt bei Multiplikation
41	$0029	Bereich für Produkt bei Multiplikation
42	$002A	Bereich für Produkt bei Multiplikation
43	$002B	Bereich für Produkt bei Multiplikation
44	$002C	Bereich für Produkt bei Multiplikation
45- 46	$002D-$002E	Zeiger auf Basic-Anfang
47- 48	$002F-$0030	Zeiger auf Variablen-Anfang
49- 50	$0031-$0032	Zeiger auf Beginn der Arrays
51- 52	$0033-$0034	Zeiger auf Ende der Arrays (+1)
53- 54	$0035-$0036	Zeiger auf Stringspeicher
55- 56	$0037-$0038	Hilfszeiger für Strings
57- 58	$0039-$003A	Zeiger auf Speichergrenze
59- 60	$003B-$003C	Laufend Basic-Zeilennummer
61- 62	$003D-$003E	Zeiger auf Basic-Text für CHRGET
63- 64	$003F-$0040	Hilfszeiger für PRINT USING
65- 66	$0041-$0042	Nummer der aktuellen DATA-Zeile
67- 68	$0043-$0044	Adresse des aktuellen DATA-Elementes
69- 70	$0045-$0046	Sprungvektor für INPUT
71- 72	$0047-$0048	Aktueller Variablenname
73- 74	$0049-$004A	Adresse der aktuellen Variablen
75- 76	$004B-$004C	Variablenzeiger für FOR...NEXT
77- 78	$004D-$004E	Zwischenspeicher für Basic-Zeiger
79	$004F	Akkumulator für Vergleichssymbole: $01=größer, $02=gleich, $04=kleiner
80- 81	$0050-$0051	Zeiger auf Variable einer DEF FN-Funktion
82- 84	$0052-$0054	Zeiger auf String-Descriptor in einer Variablen-Liste
85	$0055	Flag: HELP oder LIST
86- 88	$0056-$0058	Sprungvektor für Funktionen
89- 98	$0059-$0062	Bereich für numerische Operationen
99	$0063	Fließkomma-Akkumulator 1 (FAC): Exponent
100-103	$0064-$0067	Fließkomma-Akkumulator 1 (FAC): Mantisse
104	$0068	Fließkomma-Akkumulator 1 (FAC): Vorzeichen
105	$0069	Zeiger für Polynomauswertung
106-111	$006A-$006F	Fließkomma-Akkumulator 2 (Exponat,4 Byte Mantisse, Vorzeichenbyte)
112	$0070	Vorzeichenvergleich Akku 1 und Akku 2
113	$0071	Akku 1, niederwertige Stelle, Rundung
114-115	$0072-$0073	Kassettenpuffer Zeiger/Länge
116-117	$0074-$0075	Inkrement bei AUTO-Befehl, $00=aus
118	$0076	Grafik-Flag 0=Text, 255=Grafik
119-121	$0077-$0079	Arbeitsbereich
122-124	$007A-$007C	Darstellung von DS$
125-126	$007D-$007E	Basic-Pseudo-Stack-Pointer
127	$007F	Flag für Direktmodus oder Programm
128	$0080	Zeiger bei PRINT USING auf Dezimalpunkt
129	$0081	PARSTX
130	$0082	Enthält Stapelzeiger bei TRAP-Befehl
131	$0083	Aktuelle Zeichenfarbe
132	$0084	Aktuelle Multicolor-Farbe 1
133	$0085	Aktuelle Multicolor-Farbe 2
134	$0086	Aktuelle Vordergrundfarbe
135-136	$0087-$0088	X-Faktor bei SCALE-Befehl
137-138	$0089-$008A	Y-Faktor bei SCALE-Befehl
139	$008B	Flag für Randbegrenzung bei PAINT
140-141	$008C-$008D	Zeiger für Bitmap
142-143	$008E-$008F	Zwischenspeicher für Grafikoperationen
144	$0090	Statuswort ST
145	$0091	Flag: Stop-Taste
146	$0092	Zeit-Konstante für Kassette
147	$0093	Flag: $00=LOAD, $01=VERIFY
148	$0094	Flag: Serieller Bus - Ausgabe Zeichenpuffer
149	$0095	Zwischenpeicher - Serieller Bus
150	$0096	Zwischenpeicher (Kassetten SYNC.-Nr.)
151	$0097	Temporäre Datenadresse
152	$0098	Anzahl der offenen Dateien/Datentabellen-Index
153	$0099	Eingabegerät
154	$009A	Ausgabegerät
155	$009B	Paritätsbyte von Kassette
156	$009C	Flag: Byte empfangen
157	$009D	Flag: $80=Direkt-,$00=Programm-Modus
158	$009E	Bandfehler/Zeichenpuffer/Temporärer Speicher
159	$009F	Bandfehler korrigiert/Temporärer Speicher
160-162	$00A0-$00A2	Echtzeit-Uhr
163	$00A3	Bitzähler für serielle Ausgabe
164	$00A4	Zähler auf Band
165	$00A5	Startsynchronisation bei Kassette
166	$00A6	Zeiger im Kassettenpuffer
167-171	$00A7-$00AB	Flags für Schreiben/Lesen auf Kassette und RS232
172-173	$00AC-$00AD	Zeiger: Kassettenpuffer/Scrolling/Boot-Adresse
174-175	$00AE-$00AF	Zeiger auf Programmende bei LOAD
176-177	$00B0-$00B1	Bandzeitkonstante
178-179	$00B2-$00B3	Startadresse des Bandpuffers
180	$00B4	Bitzähler für Band und RS232
181	$00B5	Nächstes zu übertragendes Bit für Band und RS232
182	$00B6	RS232-Bytepuffer
183	$00B7	Länge des Filenamens
184	$00B8	Logische Filenummer
185	$00B9	Sekundäradresse
186	$00BA	Gerätenummer
187-188	$00BB-$00BC	Zeiger auf Filenamen
189	$00BD	Ein-/Ausgabespeicher (für seriell)
190	$00BE	Blockzähler für Band
191	$00BF	Puffer für serielle Ausgabe/Booten=$00
192	$00C0	Kassettenmotor-Flag
193-194	$00C1-$00C2	Startadresse für Ein-/Ausgabe beim Booten: Track(194) & Sektor (193)
195-196	$00C3-$00C4	Endadresse für Ein-/Ausgabe-Zwischenspeicher für Kernalvektoren
197	$00C5	Kassette: Schreib-/Lese-Daten
198	$00C6	Aktuelle Bank für LOAD/SAVE und VERIFY
199	$00C7	Bank, in der sich der Filename befindet
200-201	$00C8-$00C9	Zeiger auf RS232-Eingabepuffer
202-203	$00CA-$00CB	RS232-Ausgabe-Zeiger
204-205	$00CC-$00CD	Vektor: Tastatur-Decodiertabelle
206-207	$00CE-$00CF	Hilfszeiger für Strings
208	$00D0	Anzahl der Zeichen im Tastaturpuffer
209	$00D1	Länge eines Funktionstasten-Strings
210	$00D2	Zeiger auf String einer Funktionstaste
211	$00D3	Flag für SHIFT-/CTRL- und COMMODORE-Taste
212	$00D4	Nummer der augenblicklich gedrückten Taste
213	$00D5	Nummer der zuletzt gedrückten Taste
214	$00D6	RETURN-Input-Flag
215	$00D7	Flag für Bildschirmformat ($00=40 Zeichen, $80=80 Zeichen)
216	$00D8	Flag für Grafikmodus
217	$00D9	RAM/ROM-Flag für Video-Chip
218-219	$00DA-$00DB	Hilfszeiger für Bildschirmeditor (Scrolling etc.)
220-221	$00DC-$00DD	Hilfszeiger für Bildschirmeditor (Scrolling etc.)
222	$00DE	Temporäre Speicher für Editor
223	$00DF	Temporäre Speicher für Editor
224-225	$00E0-$00E1	Zeiger der aktuellen Zeile im Bildschirm-RAM
226-227	$00E2-$00E3	Zeiger der aktuellen Zeile im Attribut-RAM
228	$00E4	Fenster: Unterer Rand
229	$00E5	Fenster: Oberer Rand
230	$00E6	Fenster: Linker Rand
231	$00E7	Fenster: Rechter Rand
232	$00E8	Startspalte bei Eingabe
233	$00E9	Startzeile bei Eingabe
234	$00EA	Schlußzeile bei Eingabe
235	$00EB	Aktuelle Cursor-Zeile
236	$00EC	Aktuelle Cursor-Spalte
237	$00ED	Maximale Anzahl der Zeilen (24)
238	$00EE	Maximale Anzahl der Spalten (39 oder 79)
239	$00EF	Nächstes auszugebendes Zeichen
240	$00F0	Vorhergehendes Zeichen (für ESC-Test)
241	$00F1	Attribut des nächsten Zeichens (Farbe)
242	$00F2	Zwischenspeicher für Farbe
243	$00F3	Reverse-Flag
244	$00F4	Flag für Anführungszeichen

Tabelle 3. Die Zeropage beim C 128 (Fortsetzung)

Dezimal	Hexadezimal	Bemerkung
245	$00F5	Flag für Insert-Modus
246	$00F6	Flag für automatisches Einfügen
247	$00F7	Sperrt COMMODORE/SHIFT und CTRL-S
248	$00F8	Verhindert Bildschirmscrollen und Zeilenverknüpfung
249	$00F9	Verhindert CTRL-G (Bell)
250-255	$00FA-$00FF	freie Speicherstellen

Tabelle 3. Die Zeropage beim C128 (Schluß)

Dezimal	Hexadezimal	Bemerkung
256- 511	$0100-01FF	Prozessorstack
512- 600	$0200-0258	Eingabepuffer
601- 604	$0259-025C	Basic-Puffer
605- 684	$025D-02AC	Basic-/DOS-Arbeitsbereich
605	$025D	DOS-Schleifenzähler
606- 621	$025E-026D	Bereich für Filenamen
622	$026E	1. Filename (Länge)
623	$026F	DOS (Laufwerk 1)
624- 625	$0270-0271	1. Filename (Adresse)
626	$0272	2. Filename (Länge)
627	$0273	DOS (Laufwerk 2)
628- 629	$0274-0275	2. Filename (Adresse)
630	$0276	DOS logische Adresse
631	$0277	DOS (Geräteadresse)
632	$0278	DOS (Sekundäradresse)
633- 634	$0279-027A	DOS (Disketten-ID)
635	$027B	ID-Flag
636	$027C	DOS (Ausgabepuffer)
637- 684	$027D-02AC	DOS (Arbeitsbereich)
685- 688	$02AD-02B0	Grafik-Cursor
689- 692	$02B1-02B4	Grafik-Cursor (Register)
693- 715	$02B5-02CB	Grafik-Register
716- 739	$02CC-02E3	Print-Using, Grafik-Arbeitsbereich
740	$02E4	High-Byte-Adresse des Charakter-ROM
747	$02EB	Trace-Flag
752	$02F0	Anzahl der Grafik-Parameter
753	$02F1	Parameter: Relativ (1), Absolut (0)
754- 755	$02F2-02F3	Fließkomma-Vektor
756- 757	$02F4-02F5	Integer-Vektor
768- 769	$0300-0301	Sprungvektor: Fehlermeldung
770- 771	$0302-0303	Sprungvektor: Basic-Warmstart
772- 773	$0304-0305	Sprungvektor: Token-Generierung
774- 775	$0306-0307	Sprungvektor: Keywort erzeugen
776- 777	$0308-0309	Sprungvektor: Hauptschleife
778- 779	$030A-030B	Sprungvektor: Eval
780- 781	$030C-030D	Sprungvektor: Token-Generierung (User)
782- 783	$030E-030F	Sprungvektor: Keywort erzeugen
784- 785	$0310-0311	Sprungvektor: User-Token bearbeiten
786- 787	$0312-0313	Sprungvektor: Interrupt (Uhr)
788- 789	$0314-0315	Sprungvektor: Interrupt
790- 791	$0316-0317	Sprungvektor: Break Interrupt
792- 793	$0318-0319	Sprungvektor: Open
794- 795	$031A-031B	Sprungvektor: Close
796- 797	$031C-031D	Sprungvektor: Kanal für Eingabe öffnen
798- 799	$031E-031F	Sprungvektor: Kanal für Ausgabe öffnen
800- 801	$0320-0321	Sprungvektor: I/O zurücksetzen
802- 803	$0322-0323	Sprungvektor: Input
804- 805	$0324-0325	Sprungvektor: Ausgabe
806- 807	$0326-0327	Sprungvektor: Abfrage der Stoptaste
808- 809	$0328-0329	Sprungvektor: Getin-Routine
810- 811	$032A-032B	Sprungvektor: Schließen aller Files
812- 813	$032C-032D	Sprungvektor: Monitor Break
814- 815	$032E-032F	Sprungvektor: Lade-Routine
816- 817	$0330-0331	Sprungvektor: Save-Routine
818-1010	$0332-03F2	Kassettenpuffer
1011-1012	$03F3-03F4	Datenzähler (Write)
1013-1014	$03F5-03F6	Datenzähler (Read)
1015-1078	$03F7-0436	RS232-Input-Puffer (64 Byte)
1079-1108	$0437-0454	System-Arbeitsbereich

Tabelle 4. Die restlichen Systemadressen beim C16

Dezimal	Hexadezimal	Bemerkung
1109-1138	$0455-0472	System-Arbeitsbereich
1139-1144	$0473-0478	Chrget-Routine
1145-1156	$0455-0484	Chrgot-Routine
1172-1244	$0494-04DC	Bankswitching-Routinen
1255-1258	$04E7-04EA	Print-Using-Parameter
1263-1270	$04EF-04F6	Trap- und Error-Flags
1280-1282	$0500-0502	USR-Sprungbefehl
1283-1287	$0503-0507	Startwert für RND
1288	$0508	Flag für Kalt- oder Warmstart
1289-1298	$0509-0512	Tabelle der logischen Filenummern
1299-1308	$0513-051C	Tabelle der Gerätenummern
1309-1318	$051D-0526	Tabelle der Sekundäradressen
1319-1328	$0527-0530	Tastaturpuffer
1329-1330	$0531-0532	Basic-Anfang
1331-1332	$0533-0534	Basic-Ende
1337	$0539	Zeiger: Kassettenpuffer
1338	$053A	Typ des Kassettenfiles
1344	$0540	Tasten-Wiederholfunktion ($80=alle, $40=keine, $00=nur DEL, CUR, SPC)
1345-1346	$0541-0542	Zähler für Wiederholfunktion
1347	$0543	Shift-Flag
1348	$0544	Letztes Shift-Zeichen
1349-1350	$0545-0546	Sprungvektor: Keylog
1351	$0547	Text-/Grafik-Flag
1352	$0548	Scroll-Flag
1355-1372	$054B-0551	Cpu-Arbeitsbereich
1362-1367	$0552-0557	Cpu-Register
1368-1372	$0558-055C	Cpu-Arbeitsbereich
1373	$055D	Zähler für Funktionstasten
1374	$055E	Zeiger: Text, Funktionstasten
1375-1510	$055F-05E6	Speicher für Funktionstasten
1516-1571	$05EC-06EB	1 Page, Bankingroutinen
1516-1519	$05EC-05EF	Adreßtabelle
1520-1521	$05F0-05F1	Long-Jump (Adresse)
1522	$05F2	Long-Jump (Akkumulator)
1523	$05F3	Long-Jump (X-Register)
1524	$05F4	Long-Jump (Status-Register)
1525-1629	$05F5-065D	RAM-Bereich für Bankswitching
1630-1771	$065E-06EB	RAM-Bereich für Benutzersoftware
1792-1967	$06EC-07AF	Basic-Pseudo-Stack
1968-1996	$07B0-07CC	Kassetten-Arbeitsbereich
1997-2000	$07CD-07D0	RS232-Arbeitsbereich
2001	$07D1	RS232: Pointer auf Anf. Eingabepuffer
2002	$07D2	RS232: Pointer auf Ende Eingabepuffer
2003	$07D3	Anzahl der Zeichen im Eingabepuffer
2021	$07E5	Bildschirm: Unterer Rand
2022	$07E6	Bildschirm: Oberer Rand
2023	$07E7	Bildschirm: Linker Rand
2024	$07E8	Bildschirm: Rechter Rand
2026	$07EA	$FF=Automatisches Einfügen
2030-2033	$07EE-07F1	Bit-Tabelle
2034	$07F2	A-Register retten bei Sys-Kommando
2035	$07F3	X-Register retten bei Sys-Kommando
2036	$07F4	Y-Register retten bei Sys-Kommando
2037	$07F5	Status-Register retten bei Sys-Kommando
2038	$07F6	Register für Tastaturabfrage
2039	$07F7	Flag: CTRL-S: $00=offen, $06=gesperrt
2040	$07F8	RAM-ROM-Umschaltung für Monitor 0=ROM, $80=RAM
2044	$07FC	Motorsteuerung

Tabelle 4. Die restlichen Systemadressen beim C16 (Schluß)

Dezimal	Hexadezimal	Bemerkung
256-271	$0100-$010F	Bereich für Filename
272	$0110	Zähler für DOS
273	$0111	Länge des 1. Filenamens
274	$0112	DOS (Laufwerk 1)
275	$0113	Länge des 2. Filenamens
276	$0114	DOS (Laufwerk 2)

Tabelle 5. Die restlichen Systemadressen beim C128

Dezimal	Hexadezimal	Bemerkung
277-278	$0115-$0116	Adresse des 2. Filenamens
279-280	$0117-$0118	Startadresse BLOAD/BSAVE
281-282	$0119-$011A	Endadresse BSAVE
283	$011B	DOS logische Adresse
284	$011C	DOS Geräteadresse
285	$011D	DOS Sekundäradresse
286	$011E	DOS Record-Länge
287	$011F	DOSBNK
288-289	$0120-$0121	DOS Disketten-ID
290	$0122	Flag: Arbeit beendet
291-310	$0123-$0136	Adressen für PRINT USING
291	$0123	Zeiger auf Beginn
292	$0124	Zeiger auf Ende
293	$0125	Flag: Dollar-Zeichen
294	$0126	Flag: Komma
295	$0127	Zähler
296	$0128	Vorzeichen Exponent
297	$0129	Zeiger auf Exponent
298	$012A	Anzahl der Zahlen vor dem Punkt
299	$012B	Justierungs-Flag
300	$012C	Anzahl der Positionen vor dem Punkt
301	$012D	Anzahl der Positionen nach dem Punkt
302	$012E	Flag: +/-
303	$012F	Flag: Exponent
304	$0130	Schalter
305	$0131	Zeichenzähler
306	$0132	Vorzeichennummer
307	$0133	Flag: */SPACE
308	$0134	Zeiger: Anfang des Feldes
309	$0135	Länge des Formatstrings
310	$0136	Zeiger: Ende des Feldes
311-511	$0137-$01FF	System-Stack
512-763	$0200-$02FB	Basic-Puffer/Monitor-Puffer
764-765	$02FC-$02FD	Sprungvektor: zusätzliche Funktionsroutinen
766-767	$02FE-$02FF	Sprungvektor: Function-Cartridge
768-769	$0300-$0301	Sprungvektor: Fehlermeldung
770-771	$0302-$0303	Sprungvektor: Basic-Warmstart
772-773	$0304-$0305	Sprungvektor: Token-Generierung
774-775	$0306-$0307	Sprungvektor: Keywort erzeugen
776-777	$0308-$0309	Sprungvektor: Befehl ausführen
778-779	$030A-$030B	Sprungvektor: Token auswerten
780-781	$030C-$030D	Sprungvektor: Escape-Umwandlung
782-783	$030E-$030F	Sprungvektor: Escape List
784-785	$0310-$0311	Sprungvektor: Escape ausführen
786-787	$0312-$0313	Sprungvektor: Interrupt (Uhr)
788-789	$0314-$0315	Sprungvektor: Interrupt (IRQ)
790-791	$0316-$0317	Sprungvektor: Interrupt (BRK)
792-793	$0318-$0319	Sprungvektor: Interrupt (NMI)
794-795	$031A-$031B	Sprungvektor: OPEN
796-797	$031C-$031D	Sprungvektor: CLOSE
798-799	$031E-$031F	Sprungvektor: Kanal für Eingabe öffnen
800-801	$0320-$0321	Sprungvektor: Kanal für Ausgabe öffnen
802-803	$0322-$0323	Sprungvektor: I/O zurücksetzen
804-805	$0324-$0325	Sprungvektor: INPUT
806-807	$0326-$0327	Sprungvektor: Ausgabe
808-809	$0328-$0329	Sprungvektor: Abfrage der Stop-Taste
810-811	$032A-$032B	Sprungvektor: GETIN-Routine
812-813	$032C-$032D	Sprungvektor: Schließen aller Files
814-815	$032E-$032F	Sprungvektor: Monitor BREAK
816-817	$0330-$0331	Sprungvektor: LOAD-Routine
818-819	$0332-$0333	Sprungvektor: SAVE-Routine
927-977	$039F-$03D1	Unterprogramme zum Laden aus beliebiger Bank
978-980	$03D2-$03D4	Konstante für Basic
981	$03D5	Bank für PEEK, POKE, SYS...
982-985	$03D6-$03D9	Temporäre Register
986	$03DA	Bank-Zeiger für Zahl/String-Umwandlung
987-990	$03DB-$03DE	Temporäre Speicher für SSHAPE
991	$03DF	BITS
992-993	$03E0-$03E1	Temporärer Speicher für SPRSAV
994	$03E2	Vordergrund-/Hintergrund-Farbe
995	$03E3	Vordergrund-/Multicolor1-Farbe
1024-2047	$0400-$07FF	Bildschirmspeicher im 40-Zeichen-Modus
2048-2559	$0800-$09FF	Basic-Run-Time-Stack

Dezimal	Hexadezimal	Bemerkung
2560-2561	$0A00-$0A01	Sprungvektor: Basic-Warmstart
2562-2563	$0A02-$0A03	Sprungvektor: Kaltstart
2563	$0A03	Flag: PAL/NTSC
2564	$0A04	Flag: RESTE/NMI
2565-2566	$0A05-$0A06	Zeiger: Anfang verfügbarer Speicher in System-Bank
2567-2568	$0A07-$0A08	Zeiger: Ende verfügbarer Speicher in System-Bank
2569-2570	$0A09-$0A0A	Sprungvektor: IRQ (Kassette)
2571	$0A0B	Zeitvergleich bei Kassettenoperationen
2572-2573	$0A0C-$0A0D	Temporäre Speicher beim Kassetten-Lesen
2574	$0A0E	Flag: Timeout
2575-2587	$0A0F-$0A1B	Arbeitsbereich RS232
2588	$0A1C	Flag: Intern/Extern
2589-2591	$0A1D-$0A1F	Zwischenspeicher Echtzeituhr
2592	$0A20	Zeiger: Tastatur-Warteschlange
2593	$0A21	Flag: CTRL-5
2594	$0A22	Flag: REPEAT $80=alle Tasten, $40=keine Tasten, $00=Cursortasten/DEL/INST
2595-2596	$0A23-$0A24	Zähler: REPEAT
2597	$0A25	Zähler: COMMODORE/SHIFT
2598	$0A26	Flag für Cursor-Modus, $40=fester Cursor
2599-2600	$0A27-$0A28	Hilfsregister für VIC
2601	$0A29	Cursorzeichen vor dem Blinken
2602	$0A2A	Cursorfarbe vor dem Blinken
2603	$0A2B	Cursormodus 80 Zeichen (VDC):$80=fest,$60=normal blinkend, $40=schnell blinkend, $20=Aus
2604-2607	$0A2C-$02AF	Basiszeiger VIC/VDC
2608	$0A30	Temporäre Zeiger auf letzte Zeile
2609-2612	$0A31-$0A34	Temporäre Register (VDC)
2613	$0A35	Cursorfarbe vor dem Blinken (VDC)
2614	$0A36	Splitscreen-Raster-Wert (VIC)
2615	$0A37	Zwischenspeicher
2616	$0A38	Hilfszähler für Echtzeituhr (PAL)
2688-2739	$0A80-$0AB4	Systemadressen für Monitor
2752	$0AC0	Aktuelle Funktionstasten ROM-Bank
2753	$0AC1	Geräteadresse
2816-3071	$0B00-$0BFF	Kassettenpuffer und Puffer für BOOT-Sektor
3072-3327	$0C00-$0CFF	RS232-Eingabepuffer
3328-3583	$0D00-$0DFF	RS232-Ausgabepuffer
3584-4095	$0E00-$0FFF	Bereich für Sprite-Definition
4096-4351	$1000-$10FF	Bereich für programmierbare Funktionstasten
4352-4400	$1100-$1130	Bereich für DOS-String
4401-4408	$1131-$1138	Koordinaten für Grafikcursor
4409-4424	$1139-$1149	Parameter für LINE
4425-4431	$1149-$114E	Variablen für Winkel-Routinen
4432-4439	$1150-$1157	Variablen für CIRCLE-Befehl
4432-4447	$1150-$115F	Variablen für BOX-Befehl
4433-4449	$1151-$1162	Variablen für SSHAPE/GSHAPE
4456-4566	$1168-$11D6	Allgemeine Grafikvariablen
4608-4609	$1200-$1201	Vorhergehende Basic-Zeilennummer
4610-4611	$1202-$1203	Zeiger auf Basic-Befehl (für CONT)
4612-4615	$1204-$1207	Symbole bei PRINT USING
4616-4621	$1208-$120D	Parameter für Fehlerbehandlung
4622-4641	$12DE-$1221	Allgemeine Basic-Variablen
4642-4725	$1222-$1275	Variablen für Musikbefehle
4726-4736	$1276-$1280	diverse Interrupt-Flags
4737-4784	$1281-$12B0	Variablen für SOUND-Befehle
4785-4860	$12B1-$12FC	Register für Paddles, WINDOW und SPRDEF
4864-6143	$1300-$17FF	Unbenutzter RAM-Bereich
7168-8191	$1C00-$1FFF	Start Basic-Text/bei Grafik: Video-Matrix
8192-16383	$2000-$3FFF	VIC-Bitmap (im Grafikmodus)
16384	$4000	Start Basic-Text (im Grafikmodus)
53248	$D000	Zeichensatz/VIC-Register
54272	$D400	SID-Register
54528	$D500	Register der MMU im I/O-Bereich
54784	$D600	80-Zeichen-Videochip
55296	$D800	VIC Color-Nibble
56320	$DC00	CIA 1
56576	$DD00	CIA 2
56832	$DE00	Expansion-Port I/O
57088	$DF00	Expansion-Port I/O (für DMA)
65280	$FF00	Register der MMU

Tabelle 5. Die restlichen Systemadressen beim C128 (Schluß)