Etwa Mitte Juli wird COMMODORE den

BENUTZER-CLUB für den PET-2001-COMPUTER

eröffnen.

Seine Aufgaben sind u. a.:

1. Information über Hard- und Software
2. Vermittlung von Software
3. Beantwortung schriftlicher Anfragen

zu 1: Es ist vorgesehen, die Mitteilungen in einer Loseblattsammlung zu verfassen (HW = Hardware, FW = Firmware, SW = Software, usw.)

zu 2: Sofern Sie spezielle Problemlösungen direkt weitergeben möchten, teilen Sie uns dies bitte mit. Wir werden entsprechende Hinweise mit Ihrem Namen veröffentlichen.

Programme von allgemeinem Interesse wird Commodore sowohl selbst anbieten, als auch in Kommission. Die genauen Konditionen liegen noch nicht fest.

Wir bitten Sie vorläufig um ein Angebot (Provision pro verkaufter Kassette oder Pauschale).

zu 3: Anfragen jeder Art zum PET 2001 und zur Peripherie richten Sie bitte schriftlich an

COMMODORE Büromaschinen GmbH
PET-BENUTZER-CLUB
Frankfurter Straße 171 - 175

6078 Neu Isenburg

Bitte, haben Sie Verständnis, daß wir zu telefonischer Beratung nicht in der Lage sind.

commodore

Wenn Sie dem PET-Benutzer-Club ein von Ihnen entwickeltes Programm anbieten, fügen Sie bitte folgende Informationen bei:

1. Name des Programmierers + Angebot

2. kurze Beschreibung (max. 20 Zeilen) des Programms;
   hier sollen kurz, aber vollständig die Fragen beantwortet werden:
   "Was macht dieses Programm?" und
   "Was kann man damit machen?"

3. eine Kassette mit Ihrem Programm

4. eine kurze "Bedienungsanleitung" des Programms

5. ein Flußdiagramm (nicht zwingend, jedoch vorzugsweise)

6. wenn Drucker vorhanden, einen Programmausdruck

7. Angabe etwaiger Besonderheiten in der Entwicklung oder Anwendung dieses speziellen Programms

Die Punkte 1 bis einschließlich 3 müssen, die Punkte 4 - 7 können erfüllt werden.

? Wo erhalte ich nähere Unterlagen über den Mikroprozessor 6502 und die Peripheriebausteine?

A: Wir empfehlen vorläufig die Dokumentation zum KIM-Mikrocomputer, insbesondere das Programmierhandbuch (Maschinensprache) und das Hardware-Handbuch.

		ELBATEX GMBH Cäcilienstr. 24 7100 Heilbronn Tel. 07131 89001 Preis exkl.MwSt	COMMODORE GmbH Abt. MOS Frankfurter Str.171-5 6078 Neu Isenburg 06102 8003 Preis inkl. MwSt
KIM 1 Handbuch	deutsch	10,50	11,75
	englisch	-	11,75
Hardware-Handbuch	deutsch	12,50	14,00
	englisch	-	14,00
Programmierhandbuch	deutsch	14,50	16,25
	englisch	-	16,25
KIM 2,3,4 User Manual	englisch	4,70	5,25
KIM Assembler Manual	englisch	5,10	5,70
KIMath Subroutinen	englisch	8,00	8,95
KIM Text Editor	englisch	4,50	5,00
Kunststoffmappe für alle Bücher		8,00	-
Alle KIM-Bücher in Mappe		60,00	-

? Kann ich das deutsche Handbuch erhalten, ohne den PET 2001 zu bestellen?

A: Ja, es kostet DM 15,-- inkl. MwSt

commodore

PET-BENUTZER-CLUB

Ab 1. Sept. können Sie dem PET-Benutzer-Club für Deutschland, Österreich und die Schweiz beitreten. Die Mitgliedschaft ist für jeden offen und beginnt nach dem Eingang Ihres Schecks mit dem 1. des folgenden Monats. Sie gilt jeweils 1 Jahr.

Für den Mitgliedsbeitrag von DM 50,-** erhalten Sie u. a.:

* die Club-Mitteilungen (6 x pro Jahr und bei Bedarf)

* Zugang zur Programmbibliothek des Clubs. Die Programme des Benutzer-Clubs werden nur für Mitglieder erhältlich sein, im Gegensatz zur Standardsoftware von Commodore. Diese wird generell angeboten.

* 2 x jährlich senden wir Ihnen ein Verzeichnis der vom Club erhältlichen Programme, ebenfalls 2 x im Jahr die neuesten Ergänzungen.

* Sofern wir Mitteilung davon erhalten, werden wir auch ein Verzeichnis größerer Programmierungsvorhaben erstellen. Dies soll Doppelentwicklungen vermeiden helfen.

** Alle Preisangaben auf dieser Seite beinhalten 12 % MwSt.

PET-BENUTZER-CLUB

Der Preis der Programme richtet sich nach dem Umfang.

Er setzt sich zusammen aus:

Fixkosten.....DM 6,-* (Kassette, Porto, Arbeitsaufwand)

plus programmabhängige Kosten:

für Programme kleiner 10kB.....DM 10,-*

zwischen 10kB und kleiner 25kB...DM 15,-*

ab 25 kB.....DM 20,-*

Ein Programm mit 13kB würde also beispielsweise DM 6,-* plus DM 15,-* kosten.

Wenn Sie dem Benutzer-Club ein von Ihnen entwickeltes Programm zur Verfügung stellen (siehe auch Seite A Ø2), erhalten Sie dafür beliebige Programme des Clubs entsprechend dem dreifachen Umfang Ihres Programms.

Bis Ende August erhält jeder PET-Besitzer sämtliche von Commodore zum PET 2001 herausgegebenen Informationen einschließlich der bis dahin erschienenen Mitteilungen des Benutzer-Clubs. Ab 1. 9. stehen die Leistungen des Clubs nur für Mitglieder zur Verfügung.

---

* Alle Preisangaben auf dieser Seite beinhalten 12 % MwST.

commodore

PET-BENUTZER-CLUB

Anfragen jeder Art zum PET 2001 und zur Peripherie richten Sie bitte an

Commodore Büromaschinen GmbH

PET-BENUTZER-CLUB

Frankfurter Str. 171 - 175

6078 Neu Isenburg

Wir sind derzeit zu telefonischer Beratung nicht in der Lage. Aus dem Postumfang ergibt sich auch, daß nicht jede Anfrage einzeln beantwortet werden kann. Im allgemeinen werden Sie deshalb die Antworten auf Ihre Fragen in der folgenden Club-Mitteilung finden.

? Wie kann ich den Befehl PRINT AT X realisieren?

A.: Der Befehl PRINT AT X ist in manchen BASIC-Versionen enthalten. Er bewirkt den Beginn des Ausdrucks an Stelle X.

Der Bildschirm des PET ist unterteilt in 25 Zeilen (Zeile Ø bis 24) und 40 Spalten (Spalte 1 bis 39). Das sind 1000 Plätze, an denen beliebige Zeichen stehen können. Diese 1000 Zeichen sind im Bildschirmspeicher von (dezimal) Adresse 32768 bis 33767 gespeichert.

Im Speicher des PET befinden sich folgende Informationen über die Lage des Kursors:

Adresse 245 enthält die Zeile (Ø bis 24).
Adresse 226 enthält die Spalte (Ø bis 39).
Adressen 224 und 225 enthalten die der Stellung des Kursors entsprechende Adresse des Bildschirmspeichers (niederwertiges Byte in 224, höherwertiges Byte in 225).

Das untenstehende Beispiel-Programm druckt ein * in Zeile Z, Spalte S des Bildschirms. Der Übersichtlichkeit halber sind die Zeilen von 1 bis 25 und die Spalten von 1 bis 40 numeriert.

```
5 PRINT"■"
10 INPUT" ZEILE";Z
20 INPUT"SPALTE";S
25 IF Z<1 OR Z>25 OR S<1 OR S>40 THEN5
30 GOSUB1000
40 PRINT"■■■"
50 END
1000 M=32727+40*Z+S
1010 I=INT(M/256)
1020 J=M-I*256
1030 POKE224,J:POKE225,I
1040 RETURN
```

commodore

?: Wie kann ich die Rechengeschwindigkeit des PET erhöhen?

A.: 1. Entfernen Sie alle unnötigen Leerstellen und REM-Befehle aus dem Programm. Dies führt zu einer geringen Erhöhung der Rechengeschwindigkeit. Der BASIC-Interpreter spart die Zeit ein, die er sonst zum Oberprüfen und Oberspringen dieser Zeichen benötigen würde.

2. Verwenden Sie Variablen anstatt Konstanten.
Die Umwandlung einer Konstanten in die Fließkommadarstellung benötigt wesentlich mehr Zeit, als der Zugriff auf den Wert einer einfachen oder Matrix-Variablen.
Dies ist besonders wichtig bei FOR-NEXT-Schleifen oder ähnlichen Befehlsfolgen, die wiederholt ausgeführt werden.

Im folgenden Beispiel erreicht man so eine Erhöhung der Rechengeschwindigkeit fast um den Faktor 10. (Beachten Sie Zeile 40).

```
10 X=TI
30 FORG=1TO1000
40 A=A+3.14
50 NEXT
60 PRINT(TI-X)/60
```

```
10 X=TI
20 PI=3.14
30 FORG=1TO1000
40 A=A+PI
50 NEXT
60 PRINT(TI-X)/60
```

3. Variablen, die während der Programmausführung zuerst angetroffen werden, erhalten einen Speicherplatz am Beginn der Variablentabelle zugewiesen.

Die Programmzeile:

```
10 A=0:B=0:C=0
```

bewirkt, daß A als erste, B als zweite und C als letzte Variable plaziert wird. (Falls Zeile 10 die erste Programmzeile ist.)
Wenn BASIC im Lauf der weiteren Programmausführung einen Befehl findet, welcher auf die Variable Bezug nimmt, so wird es einen Suchvorgang benötigen, um A aufzufinden, zwei Suchvorgänge für B und drei für das Auffinden von C.

4. NEXT Befehle ohne die Indexvariable.
   NEXT ist etwas schneller als NEXTR, weil die Oberprüfung auf Obereinstimmung mit der zugehörigen FOR-Variablen entfällt. Man sollte den so erreichbaren Zeitgewinn abwägen gegenüber dem Verlust an Klarheit und Obersichtlichkeit des Programms.

   Beispiel:

```
10 T=TI
20 FORR=1TO10000:NEXT
30 PRINT(TI-T)/60
```

```
10 T=TI
20 FORR=1TO10000:NEXTR
30 PRINT(TI-T)/60
```

5. Ist der noch freie RAM-Speicher kleiner als etwa 300 Bytes, dann erhöht sich die Rechenzeit dramatisch. (Es wird dann für das Betriebssystem immer schwieriger, etwa einen veränderten String neu zu plazieren.)

?: Wie kann ich Speicherplatz einsparen?

A.: 1. Verwenden Sie mehrere Befehle in einer Zeile. Jede Zeilennummer bewirkt die Belegung von 5 Bytes (außer den für die Befehle benötigten Bytes), siehe FW Ø4.

2. Entfernen Sie alle unnötigen Leerstellen aus dem Programm.
   Die Zeile:

```
10 PRINT X, Y, Z
```

   benötigt 3 Bytes mehr als:

```
10 PRINTX,Y,Z
```

   (Alle Leerstellen zwischen Zeilennummer und erstem Zeichen werden vom Betriebssystem ignoriert.)

3. Entfernen Sie alle unnötigen REM-Befehle aus dem Programm.
   Jedes REM benötigt mindestens 1 Byte plus Bytes für den Text.
   Die Zeile:

```
130 REM Bemerkung
```

   benötigt (5 + 1 + 1Ø) Bytes.
   In der Zeile:

```
140 X = X + Y : REM SUMME
```

commodore

verbraucht der REM-Befehl 8 Bytes, einschließlich des trennenden Doppelpunktes.

4. Verwenden Sie Variablen als Konstanten. Wenn Sie beispielsweise die Konstante 1000 in einem Programm oft benötigen, dann fügen Sie den Befehl ein:

```
10 T=1000
```

anstatt jedesmal die Konstante auszuschreiben.

5. Der END-Befehl kann bei Commodore-BASIC entfallen.

6. Verwenden Sie Subroutinen für öfter vorkommende Befehlsfolgen (auch für Text!).

7. Verwenden Sie gleiche Variablen für Zwischenergebnisse, Schleifen, usw. Achten Sie aber darauf, nicht etwa die Variable X innerhalb einer FORX= ... TO-Schleife zu verwenden.

8. Benutzen Sie von Feldern auch das Element mit dem Index Null, also A(0), B(0), usw.

9. *Jede aktive FOR-NEXT-Schleife belegt 22 Bytes.

   *Jede aktive Subroutine (RETURN ist noch nicht ausgeführt) belegt 6 Bytes.

   *Jede offene Klammer benötigt 4 Bytes und jedes berechnete Zwischenergebnis 12 Bytes.

10. Verwenden Sie Ganzzahlvariablen, wo möglich. (Solche Variablen liegen zwischen -33767 und +33767 und haben das Zeichen % im Namen. Beispiel: X%, M%(7,2), A8%).

ACHTUNG: Für die untenstehenden Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angebenen Adressen!

---

* Service-Set für Recorder, bestehend aus 1 Entmagentisierdrossel mit plastikverkleideter Spitze und 1 Reinigungskassette (DM 38,00 plus MwSt):

  Fa. Udo Engelhardt
  Seumestraße 11
  6000 Frankfurt

  Tel.: 0611-495900

* Musikkassetten unbespielt, ohne Snap-Box mit einer Spielzeit von 8 Minuten je Seite (bei 200 Stück je DM 1,00; ab 1000 Stück je DM 0,90) und Snap-Boxes (à DM 0,13):

  Fa. Dieter Klose
  Tonträger
  Benthener Straße 8
  6000 Frankfurt 70

  Tel.: 0611-512300

* Software-Entwicklung (speziell im mathematisch-wissenschaftlichen Bereich):

  ISIDATA GmbH
  Lister Meile 23
  3000 Hannover 1

  Tel.: 0511-332100

* Software-Entwicklung (speziell Lagerhaltung, Adressenverwaltung, Rechnungswesen, mathematisch-wissenschaftliche Anwendungen, Dateien):

  Hofmann & Härtel
  Software-Beratung
  Zeil 1
  6000 Frankfurt 1

  Tel.: 0611-295838

commodore

ACHTUNG: Für die untenstehenden Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angegebenen Adressen!

---

* Software-Entwicklung (speziell Baustatik):

Dipl.-Ing. Jens Friedrich VDI
Pforzheimer Straße 248
7000 Stuttgart 31

Tel.: 0711-883411

* Software-Entwicklung (speziell ingenieur-wissenschaftliche, kommerzielle Programme, sowie Programme für Arzt- und Anwaltspraxen):

Dipl. Math. Hans Algayer
Donndorfer Straße 93
8580 Bayreuth

Tel.: 0921/39001

* Software-Entwicklung (speziell wissenschaftlicher Bereich):

S. V. I. - GmbH
Spechtweg 2
4150 Krefeld

Tel.: 02151-396932

* Interface von IEEE-488 nach RS232C (V24):

R. Bailey Associates
31 Bassett Road
London W 10
England

* TV-Modulator für PET 2001, auch für mehrere Fernsehgeräte
(ca. 298,-- incl. MWSt)

* Speicherweiterung für PET 2001
Grundgerät 8kB (ca. 1 690,-- incl. MWSt)
Zusatzplatinen je 8kB (je ca. 650,-- incl. MWSt)

Fa. Runow Büroelektronik
Eisenacher Straße 73
1000 Berlin 62
Tel.: 030-7813040

Diese Peripheriegeräte sind auch bei allen Commodore-Händlern erhältlich

* Software-Entwicklung aller Art:

Dr. Franz Trefny
Eschfeldstraße 4
4660 Gelsenkirchen-Buer
Tel.: 0209-593710

* Software-Entwicklung aller Art:

Ingenieurbüro Deininger
Neue Bahnhofstraße 14-1
7504 Weingarten/Karlsruhe
Tel.: 07244-1006
07244-1007

ACHTUNG: Für die obenerwähnten Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angebenen Adressen!

commodore

ACHTUNG: Für die untenstehenden Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angegebenen Adressen!

---

* Externe Tastatur für PET-2001

  Die CPT-Tastatur bietet alle Funktionen der Originaltastatur des PET, und zusätzlich den Komfort einer Normtastatur.

  Umschalter für Textverarbeitungsmodus. Ihr Anschluß erfordert keine Veränderungen am Gehäuser des PET. (ca. DM 600,--)

  Fa. Comp Mark
  Kiesweg 2
  6452 Hainburg/Hess.

  Tel.: 06182-5487

* Software-Entwicklungen aller Art:

  Ingenieurbüro Uhrmann
  Deller Straße 34
  Postfach 190530
  5650 Solingen 19 (Wald)

  Tel.: 02122-312918

* Software-Entwicklung (speziell Auswertung von Marktstudien, Netzplantechnik, Auftragsbearbeitung):

  Ing. (grad.) Otto Weller
  Bergstraße 7
  6334 Asslar

* Literatur über Mikrocomputer
  Im Oktober erscheint im Franzis-Verlag ein Sonderheft der "Elektronik" mit dem Titel "Hobby-Computer". Es enthält auch mehrere Artikel zum PET-2001 und zum KIM 1. Erhältlich in Buchhandlungen.

ACHTUNG: Für die unterstehenden Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angegebenen Adressen!

---

* Datenübertragungsgerät mit einer IEC- und zwei V24-Schnittstellen. Beide Obertragungspfade (IEC-V24/1 und IEC-V24/2) sind bidirektional und voneinander unabhängig. Je zwei Puffer für Ein- und Ausgabe.
  Geeignet für Anschluß aller Peripheriegeräte mit V24-Schnittstelle an den PET 2001:

  EDV Benutzer Verband
  Postfach 29
  8473 Pfreimd

  Tel.: 09606-696

* Software-Entwicklung für Computer - unterstützten Unterricht:

  Hannes Schiessl
  Augustenstraße 91/2
  8000 München 40

  Tel.: 089-533513 und 527203

* Software-Entwicklung für den PET 2001:

  Dipl.Math. A.C. Quint
  Hünefeldstraße 19
  6200 WI-Erbenheim

  Tel.: 06121-711464

* Software-Entwicklung aller Art:

  Hans-J- Koch
  Menzenweg 7
  3013 Barsinghausen 1

  Tel.: 05105-83026

commodore

ACHTUNG: Für die unterstehenden Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angegebenen Adressen!

---

* Software-Entwicklung (speziell verfahrenstechnsicher und mathematisch-wissenschaftlicher Bereich):

Dipl.-Ing. K.-D. Kaufmann
Morassisstraße 26
8000 München 5

Tel.: 089-223026

* Programmiervorhaben: Integrierte Daten- und Informationsverarbeitung für kleine und mittlere Unternehmen (Fakturierung, Buchhaltung Logistik, Lohn und Gehalt, Betriebsabrechnung, Personalwesen, Unternehmensplanung, Textverarbeitung und Dokumentation). Interessenten können sich kooperativ beteiligen.

Josef Niedermair
NM-IDV
Becker-Gundahl-Straße 3
8000 München 71

Tel.: 089-781262

* Steckverbinder für den PET 2001

1. User Port (DM 6,70)
2. IEEE-488 (DM 6,70)
3. 2. Kassette (DM 3,45)
4. Speicherbus (DM 11,10)

10-81-56-902 LL 12/12
von 29.80

Auch verschiedene konfektionierte Stecker mit Flachbandkabel sind lieferbar.
Bitte beziehen Sie sich auf Angebot Nr. 269678, Position 1 bis 4.

Fa. Deltrona
Postfach 1220
7050 Waiblingen

Tel.: 07151-51051

ACHTUNG: Für die untenstehenden Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angegebenen Adressen!

---

* Programme für den PET. Verschiedene Mathematik- und Spielprogramme für den PET sind erhältlich bei:

Harald B. Hardiek
Parkstraße 30
4174 Issum 2 - Sevelen

* Zweimonatliches Software-Journal für alle 65xx-Systeme, 6 Ausgaben DM 40,- (Inland). Herausgeber/Bezug:

65xx MICRO MAG
Dipl.-Volkswirt Roland Löhr
Hansdorfer Straße 4
2070 Ahrensburg
Tel.: 04102-55 816

* BASIC-Literatur:
Uni-Taschenbücher Stuttgart UTB Bd. 588/589: Basic 1/2. (Autoren: K. Weber und C.W. Türschmann).

VDI-Taschenbuch: BASIC - leicht gemacht (Autor: H. Rehbein).

* Textverarbeitungssoftware für den PET 2001:

Bense KG
Postfach 1127
4420 Coesfeld
Tel.: 02541-2377

commodore

ACHTUNG: für die untenstehenden Angebote dient der PET-Benutzerclub lediglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angegebenen Adressen.

---

* Petty-Interface PET zu 5-Bit-Fernschreiber, Hard- und Software (DM 280,- incl. MWSt., sfr 280,-, öS 2000)

'Bureau de Communication'
Postfach 520
8228 Freilassing

'Bureau de Communication'
Postfach 1
A-4893 Zell am Moos

* Hard- und Softwareinterfaces für den PET. (Meßgeräte, Datenerfassungssysteme, Analog-Ausgaben, Normschnittstellen, User-Port-Erweiterungen etc.):

Mytek Mikro-Elektronik GmbH
Kaiserstraße 160a
4600 Dortmund 1
Tel.: 0231-512286

* Datenlogger mit eingebautem Drucker für PET 2001. 16 Analogeingänge, 8 Steuerungsausgänge, V24-Ein-Ausgang, 2 Analogausgänge, Ereigniszähler, Warnton. Preis etwa DM 7500,-:

Adcomp Datensysteme GmbH
Horemansstraße 8
8000 München 19

* Software-Entwicklung (Buchhaltung, Fakturierung, Lohnabrechnung, Lagerhaltung, Direktwerbung und sonstige kaufmännische Arbeiten)

Datendienst
Klaus Adams
Dorfstraße 8
5202 Hennef-Röttgen
Tel.: 02248-3961

* Software (Nettolohnabrechnung, Adressenverwaltung, Lagerverwaltung, Wareneingang mit Kostenstellenkalkulation):

Wolfgang Lowinski
Hindenburgstraße 24
7800 Freiburg
Tel.: 0761-32044

ACHTUNG: für die untenstehenden Angebote dient der PET-Benutzerclub leidglich als Vermittler und übernimmt keinerlei Garantien! Wenden Sie sich bitte direkt an die angegebenen Adressen.

---

* Software-Entwicklung (speziell Baustatik und Ingenieurwissenschaftliche Programme):

  Ingenieurbüro Alois Aschl
  Schöppingstraße 11
  8000 München 60
  Tel.: 089-8112313

* Vordrucke für PET 2001, verwendbar für Programm-Listing oder für Bildschirm-Entwurf (Grafik). Je Blatt eine Bildschirmseite, mit Spalten- und Zeilennummern. Hellgrüner Druck. 100 Blatt DM 18,65 (incl. Hefter, Porto, Verpackung) gegen Voreinsendung auf das Postscheckkonto Nr. 331449-802 (München):

  Manfred Zeller
  Martinstraße 6
  8901 Batzenhofen

* Programmierbarer Zeichengenerator. 256 frei programmierbare Zeichen zum Erstellen eigener Zeichnsätze (z.B. griechische Buchstaben) oder für hochauflösende Grafik (320 * 200 Punkte), maximal 256 Zeichen. Eigenes RAM (2k) eingebaut.
  Dieser Zeichengenerator ist über die Commodore-Vertragshändler zu beziehen und wird von deren Serviceabteilung eingebaut.
  DM 990,- (Zeichengenerator), DM 160,- (Software) o.MWSt.

  Schießl & Steiner
  Augustenstraße 91/2
  8000 München 40
  Tel.: 089-522513 und 527203

* BASIC-Literatur

  BASIC für Anfänger von Julius Schärf (Oldenburg Verlag)

  BASIC Programmieren für Anfänger von V. Haase/W. Stucky (Hochschultaschenbücher Band 744).

COMMODORE BÜROMASCHINEN GMBH
Frankfurter Str. 171-175 · 6078 Neu-Isenburg

KURZINFO

DATASSETTE:

Der externe Recorder für Ihren PET 2001

- Elektrisch und mechanisch dem eingebauten Recorder gleichwertig
- Benötigt keine externe Stromversorgung (Anschluß am PET 2001 serienmäßig vorhanden)
- Ermöglicht beispielsweise das Kopieren von Datenfiles unter Programmkontrolle

Die externe Kassette wird als Gerät # 2 am IEEE-Bus angesprochen.

Die entsprechenden Befehle sind:

	SAVE " ", 2	VERIFY " ", 2	LOAD " ", 2
bzw.:	SAVE "NAME", 2	VERIFY "NAME", 2	LOAD "NAME", 2

Mit dem Befehl OPEN X, 2, Y bzw. OPEN X, 2, Y, " NAME" wird das logische File X der Datassette (= Recorder # 2) zugeordnet. Die zugehörigen Ein- und Ausgabebefehle lauten dann:

INPUT # X, . . .
GET # X, . . .
PRINT # X, . . .

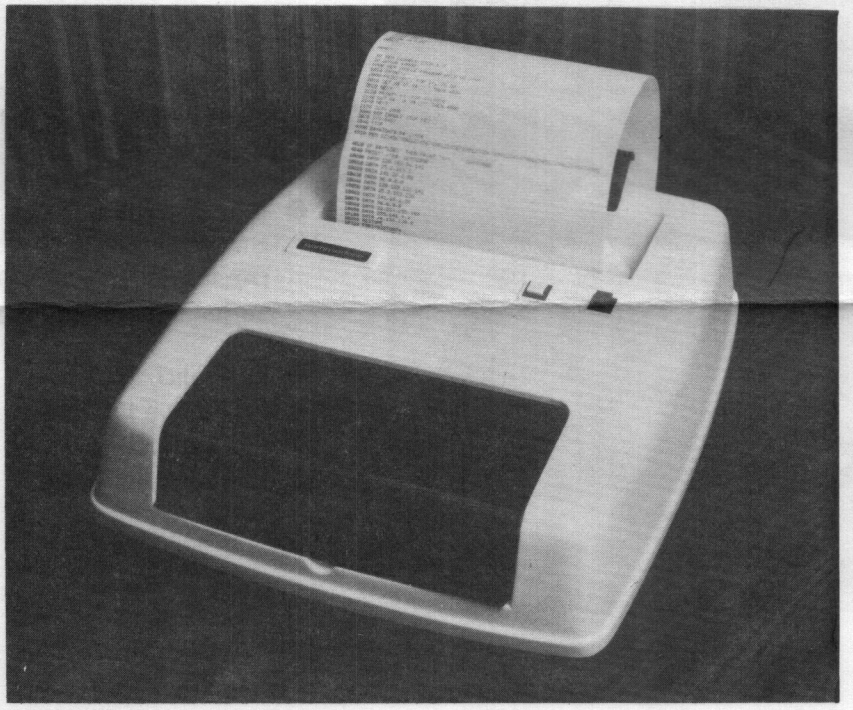
COMMODORE BÜROMASCHINEN GMBH
Frankfurter Str. 171-175 · 6078 Neu-Isenburg

KURZINFO

PET 2020:
Der neue Drucker für Ihren PET 2001

ER KANN GROSS- UND Kleinschreibung, sowie:
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑← !"#$%&'()*+,-./0123456789:;<=>?
@abcdefghijklmnopqrstuvwxyz+ ZEICHEN

DER PET PRINTER KANN SPERRSCHRIFT ODER ZEICHEN
AN BELIEBIGER STELLE DRUCKEN



- Matrixdrucker (7 x 8 bzw. 7 x 16)
- Zeichenvorrat: 160
- 80 Zeichen pro Zeile (40 bei Sperrschrift)
- Geschwindigkeit: 120 Zeichen/Sekunde (effektive minimale Druckgeschwindigkeit: 35 Zeichen/Sekunde)
- Voll formatierbare Ausgabe (numerisch und alphanumerisch)
- Vorgesehen als # 4 am IEEE-Bus (modifizierbar zu # 5 bis # 11)

- Normalpapier 8,5" breit. Rolle mit 3,25" ∅. Bei offenem Papierfach bis zu 4,5" ∅
- Papiervorschub über Rollen
- Großschreibung, Kleinschreibung **und** alle Zeichen in einer Zeile darstellbar
- Sperrschrift (doppelte Zeichenbreite)
- Negativdarstellung
- Abschaltbarer Fehlerausdruck
- Automatischer Rücklauf am Zeilenende
- Unterdrückung des Zeilenvorschubs (Überschreiben von Zeilen) möglich
- Paging (erzeugt automatisch oberen und unteren Rand beim Drucken jeder Seite)
- Datenausdruck wie empfangen (z. B. vom Bildschirm), oder formatiert möglich (links- oder rechtsbündige Darstellung, Vorzeichenunterdrückung, Vornullenunterdrückung, etc.)
- Auch Strings formatierbar
- 1 Sonderzeichen pro Zeile frei programmierbar

Bei den PET-Computern der ersten Serie sind gewisse Besonderheiten in der Programmierung zu beachten.

In den Festwertspeichern (ROM's) des PET-Computers ist ein Programm gespeichert, das folgende Aufgaben erfüllt:

1. Übersetzen von BASIC in die Maschinensprache (Interpreter)

2. Steuerung und Koordinierung von Bildschirm, Recorder, Peripherie usw. (Betriebssystem)

3. Selbsttest (Testroutinen)

Dieses Programm hat immerhin einen Speicherumfang von 13 kB. Es ist verständlich, daß bei einem Programm dieser Länge manche Fehler (engl. bugs = Käfer) erst nach längerem Betrieb auftreten. In solchen Fällen wird Sie der Benutzer-Club informieren.

In absehbarer Zeit werden wir ein ROM anbieten, das alle nötigen Verbesserungen aufweist.

1. IF F OR I = 1Ø THEN 25Ø wird zusammengezogen zu:

   IF FOR I = 1Ø THEN 25Ø und führt auf einen ?SYNTAX ERROR

   Das einzige BASIC-Wort, welches mit Zwischenraum geschrieben werden darf, ist GOTO; es ist auch in der Form GO TO zulässig.

   Der Befehl IF F OR I = 1Ø THEN 25Ø ist in Zukunft zulässig (Nach dem Einsetzen des neuen ROM's).

2. Die BYTES FREE - Nachricht nach dem Einschalten des PET-Computers ist verschieden von dem Ergebnis, das man durch Eingabe von ?FRE(Ø) unmittelbar nach dem Einschalten erhält.

   Dies ist kein Fehler, sondern kommt dadurch zustande, daß der Befehl PRINT FRE(Ø) 3 bytes Speicherplatz benötigt.

commodore

* PET 2001 SPEICHERBELEGUNG: PAGE 2,3*

(Nicht aufgefuehrte Adressen werden gebraucht, haben aber keine eindeutige Funktion)
! VORLAEUFIGE SPEZIFIKATION - AENDERUNGEN VORBEHALTEN !

512	514	200	202	24-Stunden-Uhr in 1/60 Sekunden
515	-	203	-	Gedrueckte Taste
516	-	204	-	Flag fuer SHIFT
517	518	205	206	Korrekturfaktor fuer die Uhr, L,H
519	520	207	208	Interrupt-Flags fuer Recorder-(#1, #2)-Schalter
521	-	209	-	Flag fuer RVS, SPACE, STOP, . , [, = .
523	-	20B	-	Flag fuer VERIFY und LOAD
524	-	20C	-	Statusbyte fuer I/O
525	-	20D	-	Anzahl Bytes im Tastaturpuffer
526	-	20E	-	Flag fuer RVS
527	536	20F	218	Tastaturpuffer
537	538	219	21A	IRQ-Vektor, L,H (zeigt auf: 'Tastendruck speichern')
539	540	21B	21C	BRK-Vektor, L,H
547	-	223	-	Tastatur-Input-Code, solange Taste gedrueckt
549	-	225	-	Zaehler fuer die Blinkdauer des Cursors
551	-	227	-	Flag fuer Cursor ein/aus
553	577	229	241	Tabelle der LSB's der Zeilenanfaenge (Bildschirm)
578	587	242	24B	Tabelle der Filenummern
588	597	24C	255	Tabelle der Geraetenummern
598	607	256	25F	Tabelle der Sekundaeradressen
608	-	260	-	Flag fuer Input (Bildschirm oder Tastatur)
610	-	262	-	Anzahl der offenen Files
611	-	263	-	# des Input-Geraetes (0 = Tastatur)
612	-	264	-	# des Output-Geraetes (3 = Bildschirm)
613	-	265	-	Paritaetsberechnung beim Schreiben auf Cassette
616	-	268	-	Zaehler fuer Recorder-Puffer
621	-	26D	-	Zaehler fuer Blocks auf Kassette
624	-	270	-	Synchronisation fuer Schreiben auf Kassette
625	626	271	272	Zeiger auf naechstes Zeichen Recorder-Puffer rein/raus
627	-	273	-	Synchronisation fuer Lesen von Kassette
628	-	274	-	Flag fuer Bit/Byte-Kassette-Fehler
630	631	276	277	Zeiger fuer Korrektur beim Lesen von Kassette
632	-	278	-	Flag fuer Lesen von Kassette
633	-	279	-	Dauer der 'shorts' vor dem Schreiben von Daten
634	825	27A	339	Puffer fuer Recorder #1
826	1017	33A	3F9	Puffer fuer Recorder #2
1017	1023	3FA	3FF	nicht benutzt

commodore

3. Der Befehl SAVE sollte vom Betriebssystem mit "PRESS REC & PLAY" anstatt "PRESS PLAY & REC" beantwortet werden, da die Taste REC zuerst gedrückt werden muß.

   Dies ist tragbar und wird wahrscheinlich nicht geändert.

4. Die Funktion POS wird aus Commodore-BASIC entfernt.

5. SPACE und SPACE bei gedrückter SHIFT-Taste ergeben verschiedene ASCII-Werte.

   Dies ist kein Fehler; die obenliegenden Zeichen sind generell anders indiziert als die untenliegenden.

6. Bei Output (=Ausgabe) eines Anführungszeichens (Code 34 oder 98) werden etwaige Cursor-Bewegungen wörtlich aufgefaßt.

   B E I S P I E L : 10?CHR$(34),1,2,3
   (Die Cursor-Bewegungen zu den vortabulierten Positionen werden also geschrieben, anstatt ausgeführt.)

   Wird vorläufig nicht geändert.

7. SPC(Ø) druckt 256 Leerstellen.

   Wird geändert.

8. Bei direkter Eingabe werden Zeilen mit Doppelpunkt am Anfang ignoriert.

   B E I S P I E L : :?A

   Wird geändert.

9. Bei Matrizen (Feldern) sind nur 255 Elemente pro Feld zulässig.

   B E I S P I E L : DIM B(25,9) hat zuviele Elemente

   Wird geändert.

10. CHR$ akzeptiert auch Strings als Argument.

11. Die Befehle: OPEN n,1,m
OPEN n,2,m

(n bedeutet die Nummer des Files - n ist eine ganze Zahl von 1 bis 255. m = Ø bewirkt Einlesen
m = 1 bewirkt Schreiben und
m = 2 bewirkt Schreiben mit EOT-Zeichen.)

Es hat sich herausgestellt, daß unter bestimmten Bedingungen das BOF-Zeichen (Beginning Of File = Fileanfang) trotz des OPEN-Befehls nicht ordnungsgemäß geschrieben bzw. gelesen wird.

Das korrekte Eröffnen eines Files kann erzwungen werden durch die Befehle

POKE 243,122:POKE 244,2

vor einem OPEN-Befehl für Recorder #1.

Für den Recorder #2 lauten die entsprechenden Befehle:

POKE 243,58:POKE 244,3

vor dem OPEN-Befehl für Recorder #2.

Die Anfangsadresse des jeweils aktiven Kassettenpuffers ist in 243 (niederwertiges Byte) und 244 (höherwertiges Byte) abgespeichert. Die Startadresse des Pufferspeichers für Recorder #1 ist $27A ($2 = 2 und $7A = 122). Die Startadresse des Puffers für Recorder #2 ist $33A ($3 = 3 und $3A = 58). Durch die obigen POKE-Befehle wird also eine korrekte Initialisierung sichergestellt.

Diese Befehle können auch im Programm bleiben, nachdem das in Zukunft erhältliche ROM mit dem neuen Betriebssystem eingesetzt wurde.

commodore

*SPEICHERUNG VON PROGRAMMEN (8k-Version des PET 2001)

Eine Programmzeile besteht aus Zeilennummer und Befehlen. Alle Programmzeilen werden in Reihenfolge der Zeilennummer von 1Ø24 an aufwärts folgendermaßen gespeichert (n ist die Zeilennummer, $(n+x)$ ist die auf n folgende Zeilennummer):

Das erste Byte ist Null. (A_n)

Es folgen 2 Bytes (B_n, C_n). Sie stellen die Verbindung zur nächsten Programmzeile her. Der Wert $B_n+256*C_n$ gibt die Adresse an, in der die Größe B_{n+x} steht. Sind B_n und C_n beide Null, dann ist n die letzte Zeilennummer des Programms.

Die nächsten zwei Bytes (D_n, E_n) geben die Zeilennummer an:
Zeilennummer = $D_n+256*E_n$

Es folgen die BASIC-Befehle (codiert in 1 Byte pro Befehl bzw. 1 Byte pro Zeichen).

Beispiel: Das Programm

```
1ØØØ PRINTX
2ØØØ GOTO 1ØØØ
```

wird folgendermaßen gespeichert:

Ø	8	4	232	3	153	88	Inhalt
1Ø24	1Ø25	1Ø26	1Ø27	1Ø28	1Ø29	1Ø3Ø	Speicherstelle
A	B	C	D	E	PRINT	X	Übersetzung
0	7	4	5	0	153		

Ø	18	4	208	7	137	49	48	48	48
1Ø31	1Ø32	1Ø33	1Ø34	1Ø35	1Ø36	1Ø37	1Ø38	1Ø39	1Ø4Ø
A	B	C	D	E	GOTO	1	Ø	Ø	Ø

Ø	Ø	Ø
1Ø41	1Ø42	1Ø43

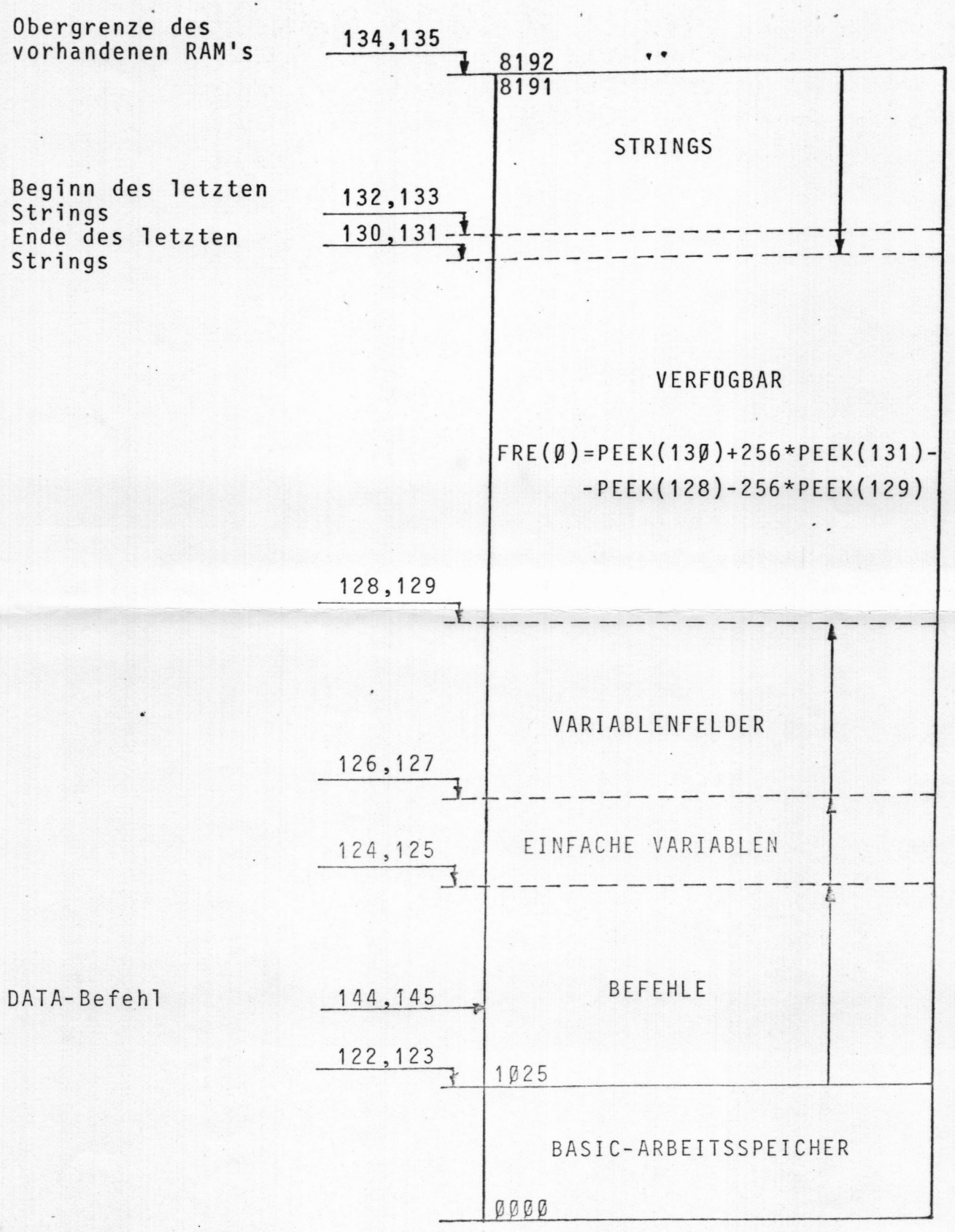
PET-2001 INTERPRETERCODE

000 ZL.ANFG	064 @	128 END	192 TAN
001 N.V.	065 A	129 FOR	193 ATN
002 N.V.	066 B	130 NEXT	194 PEEK
003 N.V.	067 C	131 DATA	195 LEN
004 N.V.	068 D	132 INPUT#	196 STR$
005 N.V.	069 E	133 INPUT	197 VAL
006 N.V.	070 F	134 DIM	198 ASC
007 N.V.	071 G	135 READ	199 CHR$
008 N.V.	072 H	136 LET	200 LEFT$
009 N.V.	073 I	137 GOTO	201 RIGHT$
010 N.V.	074 J	138 RUN	202 MID$
011 N.V.	075 K	139 IF	203 N.V.
012 N.V.	076 L	140 RESTORE	204 N.V.
013 N.V.	077 M	141 GOSUB	205 N.V.
014 N.V.	078 N	142 RETURN	206 N.V.
015 N.V.	079 O	143 REM	207 N.V.
016 N.V.	080 P	144 STOP	208 N.V.
017 N.V.	081 Q	145 ON	209 N.V.
018 N.V.	082 R	146 WAIT	210 N.V.
019 N.V.	083 S	147 LOAD	211 N.V.
020 N.V.	084 T	148 SAVE	212 N.V.
021 N.V.	085 U	149 VERIFY	213 N.V.
022 N.V.	086 V	150 DEF	214 N.V.
023 N.V.	087 W	151 POKE	215 N.V.
024 N.V.	088 X	152 PRINT#	216 N.V.
025 N.V.	089 Y	153 PRINT	217 N.V.
026 N.V.	090 Z	154 CONT	218 N.V.
027 N.V.	091 [	155 LIST	219 N.V.
028 N.V.	092 \	156 CLR	220 N.V.
029 N.V.	093]	157 CMD	221 N.V.
030 N.V.	094 ↑	158 SYS	222 N.V.
031 N.V.	095 ←	159 OPEN	223 N.V.
032 SPC	096	160 CLOSE	224 N.V.
033 !	097 !	161 GET	225 N.V.
034 "	098 "	162 NEW	226 N.V.
035 #	099 #	163 TAB(	227 N.V.
036 $	100 $	164 TO	228 N.V.
037 %	101 %	165 FN	229 N.V.
038 &	102 &	166 SPC(	230 N.V.
039 '	103 '	167 THEN	231 N.V.
040 (	104 (	168 NOT	232 N.V.
041)	105)	169 STEP	233 N.V.
042 *	106 *	170 +	234 N.V.
043 +	107 +	171 -	235 N.V.
044 ,	108 ,	172 *	236 N.V.
045 -	109 -	173 /	237 N.V.
046 .	110 .	174 ↑	238 N.V.
047 /	111 /	175 AND	239 N.V.
048 0	112 0	176 OR	240 N.V.
049 1	113 1	177 >	241 N.V.
050 2	114 2	178 =	242 N.V.
051 3	115 3	179 <	243 N.V.
052 4	116 4	180 SGN	244 N.V.
053 5	117 5	181 INT	245 N.V.
054 6	118 6	182 ABS	246 N.V.
055 7	119 7	183 USR	247 N.V.
056 8	120 8	184 FRE	248 N.V.
057 9	121 9	185 POS	249 N.V.
058 :	122 :	186 SQR	250 N.V.
059 ;	123 ;	187 RND	251 N.V.
060 <	124 <	188 LOG	252 N.V.
061 =	125 =	189 EXP	253 N.V.
062 >	126 >	190 COS	254 N.V.
063 ?	127 ?	191 SIN	255 π

* N.V. = Nicht verwendet

commodore

* RAM-ZUTEILUNG UND POINTER (8k-Version des PET 2001)



63488	H7	ROM 4K
63487 61440	H4	
61439 53392	A2	ZEICHENGENERATOR 2K
59931 57344	H3	ROM 10K
57343 55296	H6	
55295 53248	H2	
53247 51200	H5	
51199 49152	H1	
49151 33792		15K ROM-ERWEITERUNG
33791 32768	C3,C4	BILDSCHIRM-RAM 1K
32767 8192		24K RAM-ERWEITERUNG
8191 7168	I8 J8	8K RAM
7167 6144	I7 J7	
6143 5120	I6 J6	
5119 4096	I5 J5	
4095 3072	I4 J4	
3071 2048	I3 J3	
2047 1024	I2 J2	
1023 0000	I1 J1	

ADRESSENBELEGUNG DES PET-2001*

(Erste Spalte = Dezimale Adresse

Zweite Spalte = Lage des entsprechenden IC's auf der Platine)

*Dies ist eine vorläufige Spezifikation, Änderungen vorbehalten.

commodore

* PET 2001 SPEICHERBELEGUNG: PAGE 0 *

(Nicht aufgefuehrte Adressen werden gebraucht, haben aber keine
eindeutige Funktion)
! VORLAEUFIGE SPEZIFIKATION - AENDERUNGEN VORBEHALTEN !

0	-	000	-	$4C (JMP-Befehl)
1	2	001	002	Adresse fuer die USR-Funktion; L,H

Input-Output:

3	-	003	-	# des aktiven Ein- Ausgabekanals
5	-	005	-	Naechste Druckspalte fuer BASIC
10	89	00A	059	80 Bytes BASIC-Input-Puffer
90	-	05A	-	Hilfszaehler fuer BASIC
91	-	05B	-	Konstante $00 als Trennungsfeld
92	-	05C	-	Hilfszaehler fuer BASIC

Variablen:

93	-	05D	-	Flag fuer dimensionierte Variablen
94	-	05E	-	Flag fuer Variablenart (0=numerisch, 1=String)
95	-	05F	-	Flag fuer Integers
96	-	060	-	Flag fuer '' und REM
97	-	061	-	Flag fuer Indizierung
98	-	062	-	Flag fuer INPUT (=0), GET/GET# (=64) oder READ (=152)
99	-	063	-	Flag fuer das Vorzeichen von TAN
100	-	064	-	Flag fuer Output (negativ, wenn Ausgabe unterdrueckt)
101	-	065	-	zeigt auf naechsten Descriptor
102	103	066	067	Zeiger auf den zuletzt verwendeten String; L,H
104	111	068	06F	Tabelle der Doppelbyte-Descriptoren fuer Variablen
112	113	070	071	Indirekter Index #1; L,H
114	115	072	073	Indirekter Index #2; L,H
116	121	074	079	Pseudoregister fuer Operanden von Funktionen

Datenabspeicherung:

122	123	07A	07B	Zeiger auf den Beginn der BASIC-Befehle; L,H
124	125	07C	07D	Zeiger auf den Beginn der Variablentabelle; L,H
126	127	07E	07F	Zeiger auf den Beginn der Array-Tabelle; L,H
128	129	080	081	Zeiger auf das Ende der Variablentabelle; L,H
130	131	082	083	Zeiger auf Ende des letzten Strings; L,H
132	133	084	085	Zeiger auf Beginn des letzten Strings; L,H
134	135	086	087	Zeiger auf hoechste RAM-Adresse; L,H
136	-	088	-	ist 255, wenn ein direkter Befehl ausgefuehrt wird
136	137	088	089	# der gerade ausgefuehrten Zeile; L,H
138	139	08A	08B	Zeilen# fuer CONT; L,H
140	141	08C	08D	Zeiger auf den naechsten Befehl; L,H
142	143	08E	08F	# der DATA-Zeile bei ERROR; L,H
144	145	090	091	Zeiger auf naechstes DATA; L,H

Terme:

146	147	092	093	Herkunft des Input
148	149	094	095	Name der lfd. Variablen (1. und 2. Zeichen)
150	151	096	097	Zeiger auf die lfd. Variable; L,H
152	153	098	099	Zeiger auf die lfd. FOR-NEXT-Variable; L,H
154	155	09A	09B	Zeiger auf den lfd. Operator; L,H
156	-	09C	-	Maske fuer lfd. Operator ( ) = ( )
157	158	09D	09E	Zeiger fuer DEFFN; L,H
159	160	09F	0A0	Zeiger auf den String-Descriptor; L,H
161	-	0A1	-	Laenge des obigen Strings
162	-	0A2	-	Konstante
163	-	0A3	-	$2C (JMP-Befehl in Maschinensprache)
164	165	0A4	0A5	Vektor fuer Funktionen; L,H
166	171	0A6	0AB	Fliesskommaakkumulator #3
172	173	0AC	0AD	Zeiger #1 fuer Blocktransfer; L,H
174	175	0AE	0AF	Zeiger #2 fuer Blocktransfer; L,H
176	181	0B0	0B5	Fliesskommaakkumulator #1 (u.a. fuer USR)
182	-	0B6	-	Vorzeichen der Mantisse aus Akku #1
183	-	0B7	-	Zaehlt die zum Normalisieren von Akku #1 noetigen Shifts
184	189	0B8	0BD	Fliesskommaakkumulator #2
190	-	0BE	-	Overflowbyte des Arguments aus Akku #2
191	-	0BF	-	Vorzeichen der Mantisse aus Akku #2
192	193	0C0	0C1	Zeiger auf ASCII-Darstellung des Akku #2; L,H

RAM-Subroutinen:

194	199	0C2	0C7	CHRGOT-Routine; holt naechstes Zeichen des BASIC-Texts
200	-	0C8	-	CHRGOT; enthaelt lfds. Zeichen des BASIC-Texts
201	202	0C9	0CA	Zeiger in den BASIC-Text; L,H
203	223	0CB	0DF	enthaelt naechste Zufallszahl fuer RND

Betriebssystem:

224	225	0E0	0E1	Zeiger auf Cursorstellung im Bildschirm-RAM; L,H
226	-	0E2	-	Cursorposition (Spalte)
227	228	0E3	0E4	Indirekter Zeiger fuer diverse Zwecke; L,H
234	-	0EA	-	Flag fuer ''
238	-	0EE	-	Laenge des lfd. Filenamens
239	-	0EF	-	# des lfd. Files
240	-	0F0	-	lfd. Sekundaeradresse
241	-	0F1	-	lfd. Primaeradresse
242	-	0F2	-	Flag fuer einfache (=39) oder doppelte (=79) Zeile
243	244	0F3	0F4	Zeiger auf den lfd. Tape-Puffer; L,H
245	-	0F5	-	Cursorposition (Zeile)
246	-	0F6	-	Zwischenspeicher fuer Input-Output
247	248	0F7	0F8	Zeiger fuer das Operating-System (LOAD, VERIFY); L,H
249	250	0F9	0FA	Zeiger auf den lfd. Filenamen; L,H
255	-	0FF	-	Overflowbyte fuer ASCII-Darstellung des Akku-Inhalts

* PET 2001 SPEICHERBELEGUNG: PAGE 1 *

256-317 (dezimal) werden zur Fehlerkorrektur beim Lesen vom Band benutzt, sowie als Puffer fuer Zahlenumwandlungen. Der Rest von Page 1 dient als Speicher im Zusammenhang mit GOSUB und FOR-NEXT, sowie als Hardware-Stack.

002		066	B	130			195		c
003		067	C	131			196		d
004		068	D	132			197		e
005		069	E	133			198		f
006		070	F	134			199		g
007		071	G	135			200		h
008		072	H	136			201		i
009		073	I	137			202		j
010		074	J	138			203		k
011		075	K	139			204		l
012		076	L	140			205		m
013	RETURN	077	M	141	SH RETURN		206		n
014		078	N	142			207		o
015		079	O	143			208		p
016		080	P	144			209		q
017	CRSR N.U.	081	Q	145	CRSR N.O.		210		r
018	RVS	082	R	146	RVS OFF		211		s
019	HOME	083	S	147	CLR		212		t
020	DEL	084	T	148	INST		213		u
021		085	U	149			214		v
022		086	V	150			215		w
023		087	W	151			216		x
024		088	X	152			217		y
025		089	Y	153			218		z
026		090	Z	154			219		
027		091	[	155			220		
028		092	\	156			221		
029	CRSR N.R.	093	]	157	CRSR N.L.		222		
030		094	↑	158			223		
031		095	←	159			224	SH SPC	
032	SPC	096	SPC	160	SH SPC		225		
033	!	097	!	161			226		
034	"	098	"	162			227		
035	#	099	#	163			228		
036	$	100	$	164			229		
037	%	101	%	165			230		
038	&	102	&	166			231		
039	'	103	'	167			232		
040	(	104	(	168			233		
041	)	105	)	169			234		
042	*	106	*	170			235		
043	+	107	+	171			236		
044	,	108	,	172			237		
045	-	109	-	173			238		
046	.	110	.	174			239		
047	/	111	/	175			240		
048	0	112	0	176			241		
049	1	113	1	177			242		
050	2	114	2	178			243		
051	3	115	3	179			244		
052	4	116	4	180			245		
053	5	117	5	181			246		
054	6	118	6	182			247		
055	7	119	7	183			248		
056	8	120	8	184			249		
057	9	121	9	185			250		
058	:	122	:	186			251		
059	;	123	;	187			252		
060	<	124	<	188			253		
061	=	125	=	189			254		
062	>	126	>	190			255		
063	?	127	?	191					

ASCII-LISTE (Bildschirm): Sind zwei verschiedene Zeichen angegeben, so entsteht das zweite Zeichen durch POKE 59468,14 (Kleinschreibung). Der Betriebszustand nach dem Einschalten des PET entspricht POKE 59468,12 (Grafik).

commodore

? Wann sind Speichererweiterung und Floppy-Disc erhältlich?

A: Etwa gegen Jahresende; Daten und Preise werden in einer der nächsten Benutzer-Info's bekanntgegeben.

? Wann ist der Drucker lieferbar?

A: Etwa ab Mitte Oktober. Lieferungen erfolgen zunächst in der Reihenfolge der Bestellungen.
Ein Kurzprospekt liegt bei.

? Arbeitet der Drucker mit Normalpapier?

A: Ja.

? Wann sind ROM's mit geändertem Betriebssystem lieferbar?

A: Das ROM 6540-019 verhindert ein unkontrolliertes Wegbleiben des Cursors beim Schreiben und Korrigieren von Programmen. Es wird etwa 30,-- DM kosten und Anfang August erhältlich sein. Näheres in der nächsten Info. 59,43 incl MWST

Das ROM mit den Korrekturen bezüglich Syntax, Matrizen etc. (siehe Seite FW Ø) ist noch nicht lieferbar. Sobald dies der Fall ist, erhalten Sie Mitteilung.

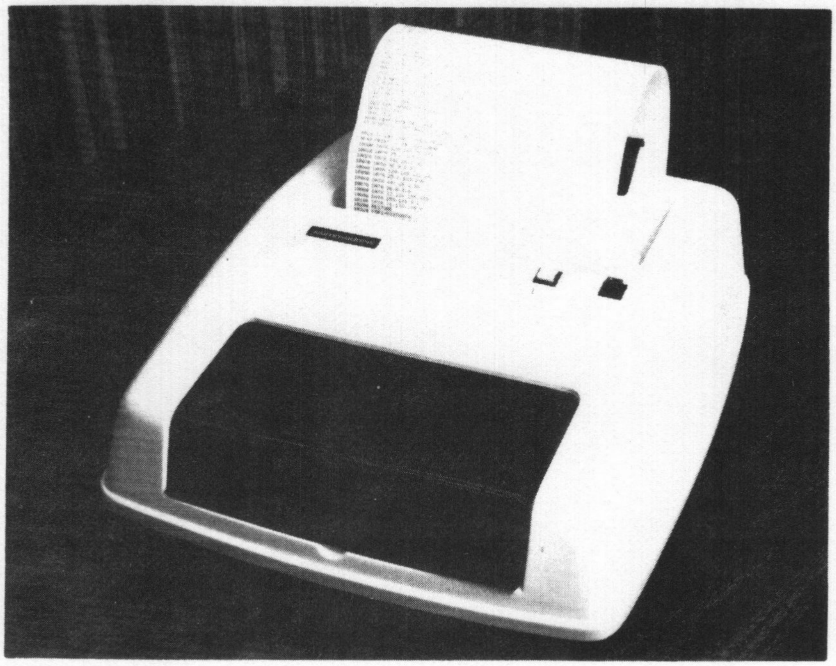
commodore GMBH
Frankfurter Str. 171-175 · 6078 Neu-Isenburg

KURZINFO

Der neue Drucker für Ihren PET:

```
er kann sowohl GROSSE und kleine Buchstaben  als auch ♠♦♣╳╱╲╲|╱
   @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑← !"#$%&'()*+,-./0123456789:;<=>?
   -♠|-‾‾-| ╮╰ ╯L╲╱┌┐●_♥| ╱XO♣|♦┼▒ |▗▜ ▌▄▔▁▁▁┘ ▒ ▙▟ |├▗┘_┼┤|| ┌▔▄▁▂ ▀▝▝▚
   -abcdefghijklmnopqrstuvwxyz┼▒ |▧▧ ▌▄▔▁┘ ▒ ▙▨ |├▗┘_┼┤|| ┌▔▄▞▗ ▀▝▝▚
```

```
DER PET PRINTER KANN S P E R R S C H R I F T ODER REVERSE FIELD ZEICHEN
AN BELIEBIGER STELLE DRUCKEN
```



- Matrixdrucker (7x8 bzw. 7x16)
- Zeichenvorrat: 160
- 80 Zeichen pro Zeile (40 bei Sperrschrift)
- Geschwindigkeit: 120 Zeichen/sec

- Normalpapier
- Großschreibung, Kleinschreibung **und** alle Zeichen in einer Zeile darstellbar
- Sperrschrift (doppelte Zeichenbreite)
- Negativdarstellung
- Abschaltbarer Fehlerausdruck
- Automatischer Rücklauf am Zeilenende
- Unterdrückung des Zeilenvorschubs (Überschreiben von Zeilen) möglich
- Paging (erzeugt automatisch oberen und unteren Rand beim Drucken jeder Seite)
- Datenausdruck wie empfangen (z. B. vom Bildschirm), oder formatiert möglich (links- oder rechtsbündige Darstellung, Vorzeichenunterdrückung, Vornullenunterdrückung, etc.)
- Auch Strings formatierbar
- Vorgesehen als # 4 am IEEE-488-Bus (modifizierbar)

commodore

KASSETTENRECORDER
ENTMAGNETISIERUNG UND REINIGUNG

Der Tonkopf des Recorders wird bei jedem Schreibzyklus leicht aufmagnetisiert. Commodore empfiehlt daher häufiges

(jeweils nach etwa 3 Stunden SAVE-Betrieb)

Entmagnetisieren des Recorders.

Zusätzlich sollte ungefähr nach jeweils 100 Betriebsstunden des Recorders oder bei Auftreten von Einlesefehlern folgende Prozedur vorgenommen werden:

*1. Legen Sie eine Reinigungskassette ein und lassen Sie diese abspielen.

*2. Schalten Sie den PET-Computer aus und entfernen Sie alle bespielten Kassetten aus seiner Nähe, damit diese nicht beim Einschalten der Löschdrossel gelöscht werden.

*3. Öffnen Sie das Kassettenfach des Recorders und drücken Sie die Wiedergabetaste PLAY.

*4. Schalten Sie die Löschdrossel ein. Die Entfernung zum Tonkopf sollte dabei mindestens einen halben Meter betragen.

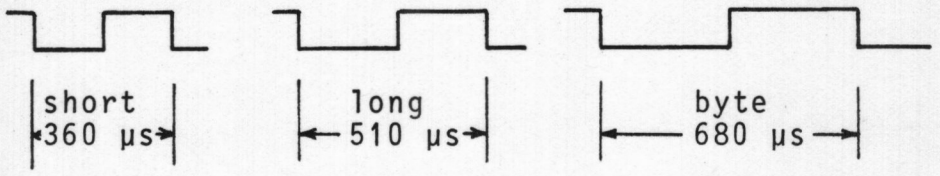
*5. Bewegen Sie die Entmagnetisierungsdrossel langsam (ca. 3 cm/sec oder langsamer) auf den Tonkopf zu und führen Sie diese langsam über alle metallischen Teile, die mit dem Tonband in Berührung kommen, einschließlich Löschkopf. Verwenden Sie keine Drossel mit blanker Oberfläche, um den Tonkopf nicht zu zerkratzen.

*6. Entfernen Sie die Drossel langsam vom Recorder. In etwa einem Meter Entfernung schalten Sie aus.

*7. Überprüfen Sie den Aufnahme-Wiedergabekopf auf Verschleiß. Ist er mehr als einige Banddicken abgeschliffen und treten immer noch Einlesefehler auf, so muß der Tonkopf ersetzt werden. Üblicherweise ist dies erst nach mehr als 3000 Betriebsstunden der Fall.

AUFZEICHNUNGSFORMAT KASSETTE

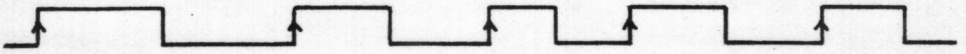
Es werden drei Frequenzen (jeweils eine Vollwelle) verwendet: "kurzer" Impuls (short), "langer" Impuls (long) und Bytezeichen (byte).



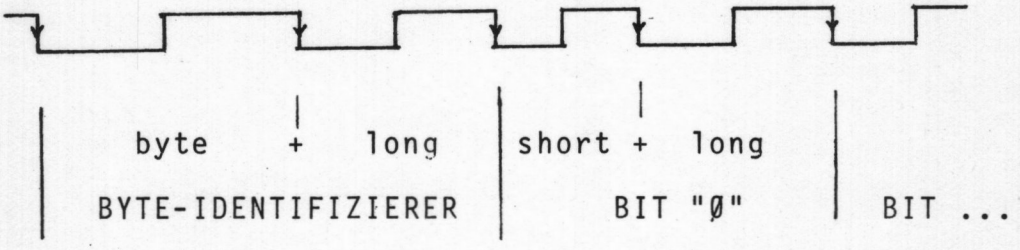
* Logisch Null wird durch die Folge short-long dargestellt.
* Logisch Eins wird durch die Folge long-short dargestellt.

Beim Schreiben auf Kassette wird die Information durch die positiv gerichteten Pegelwechsel übertragen. Beim Lesen von Kassette liegt die Information in den negativen Übergängen.

WRITE:



READ:



Daten werden mit konstantem Strom (ohne Equalizer) aufgezeichnet.

Bei der Wiedergabe folgt auf den Verstärker ein Equalizer und Impulsformer.

TIM: MONITOR FÜR DIE MASCHINENSPRACHE

TIM (Terminal Interface Monitor) wurde für die Mikroprozessoren der Serie 65xx entwickelt. Eine erweiterte und an den PET 2001-Computer angepaßte Version steht vorläufig auf Kassette zur Verfügung. Zu einem späteren Zeitpunkt wird TIM im ROM des PET 2001 fest eingespeichert sein.

TIM BEFEHLE

M	display memory;	Anzeige des Speicherinhalts.
R	display register;	Anzeige der Registerinhalte.
G	begin execution;	Ausführung eines Programmes in Maschinenspr.
X	exit to BASIC;	Rückkehr zu BASIC.
L	load;	Laden eines Programms von Kassette.
S	save;	Überspielen eines Programms auf Kassette.

TIM UNTERPROGRAMME

JSR WRT	$FFD2	type a character;	Ausgabe eines Zeichens.
JSR RDT	$FFCF	input a character;	Eingabe eines Zeichens.
JSR GET	$FFE4	get a character;	Holen eines Zeichens.
JSR CRLF	$Ø4F2	type a CR;	Schreibe CR = CHR$(13).
JSR SPACE	$Ø63A	type a space;	Schreibe Zwischenraum = CHR$(32).
JSR WROB	$Ø613	type a byte;	Schreiben eines Bytes.
JSR RDOB	$Ø65E	read a byte;	Lesen eines Bytes.
JSR HEXIT	$Ø685	ASCII to hex in A;	Umwandlung ASCII-Code nach hexadezimal.

TIM SPEICHERBENUTZUNG

$ØA bis $22 und	1Ø bis 34
$4ØØ bis $76A	1Ø24 bis 1898

commodore

TIM ZAHLENSYSTEM

Generell hexadezimale Ein- und Ausgabe.

.M XXXX, YYYY

ANZEIGE DES SPEICHERINHALTS

Anfangsadresse (XXXX) und Endadresse (YYYY) müssen vollständig (als vierziffrige Hex-Zahlen) angegeben werden.
Um den Inhalt einer Speicherzelle zu verändern, wird der Kursor an die entsprechende Stelle bewegt, dann die Korrektur vorgenommen und mit der Taste RETURN bestätigt.

Beispiel:

```
.M CØØØ, CØ1Ø
.: CØØØ1D C7 48 C6 35 CC EF C7
.: CØØ8C5 CA DF CA 7Ø CF 23 CB
.: CØ1Ø9C C8 9C C7 74 C7 1F C8
```

.R

ANZEIGE DER REGISTERINHALTE

Die Inhalte der Register der CPU (Central processing unit, 6502) werden angezeigt:

PC	Programm counter
SR	Statusregister des Prozessors
AC	Accumulator
XR	Indexregister X
YR	Indexregister Y
SP	Stack-pointer

Beispiel:

```
.R PC  SR AC XR YR SP
.:C6ED ØØ 2Ø ØØ 2Ø F5
```

Änderungen können wie bei .M mit Hilfe des Kursors und der Taste RETURN vorgenommen werden. Bei jedem Eintritt von BASIC nach TIM werden die Registerinhalte gespeichert und bei der Rückkehr von

.G XXXX

PROGRAMMAUSFÜHRUNG

Der Go-Befehl bewirkt einen Sprung zu der durch XXXX angegebenen Adresse. Wird XXXX nicht eingegeben, so dient der Inhalt des Programmzählers PC als Zieladresse-

Beispiel: .G C38B bewirkt Sprung nach C38B

.X

EXIT-RÜCKKEHR ZU BASIC

.X bewirkt einen Rücksprung nach BASIC. Die Speicherinhalte bleiben dabei unverändert, und BASIC befindet sich im selben Zustand wie vor dem Aufruf des Monitors.

.L XX, NAME

LOAD - EINLESEN DES PROGRAMMS VON KASSETTE

XX steht für die Nummer des Peripheriegeräts (Ø1 = eingebaute Kassette, Ø2 = externe Kassette), Gerätenummer und Filename müssen angegeben werden. Das OS (Operating system) reagiert wie bei BASIC. Die mit dem SAVE-Befehl definierten Speicheradressen werden geladen.

Unterprogramme in Maschinensprache können auch von BASIC geladen werden. Dabei ist aber zu beachten, daß der Variablenpointer auf das zuletzt geladene Byte plus eins zeigt. Daher dürfen BASIC-Variablen nach LOAD nicht verwendet werden.

Beispiel:

```
.L Ø1, MONITOR
PRESS PLAY ON TAPE #1
OK
SEARCHING FOR MONITOR
FOUND MONITOR
LOADING
```

commodore

.S XX, NAME, YYYY, ZZZZ

SAVE - SPEICHERN EINES PROGRAMMS

XX steht für die Nummer des Peripheriegeräts (siehe .L).
YYYY ist die hexadezimale Anfangsadresse, und ZZZZ die Endadresse plus eins.

Beispiel:

```
.S Ø1, MONITOR, Ø4ØØ, Ø76B
PRESS PLAY & RECORD ON TAPE #1
OK
WRITING MONITOR
```

Ø76B ist Endadresse plus eins, das letzte Datenbyte befindet sich also in Ø76A

BREAK UND INTERRUPTS

Der Maschinenbefehl BRK ($ØØ) bewirkt einen Software-Interrupt. Dies bedeutet, daß der Prozessor das gegenwärtige Programm unterbricht, (PC+2) und SR auf dem Stapel ablegt und dann an eine Stelle verzweigt, die durch den Vektor in $Ø21B und $Ø21C gegeben ist. TIM initialisiert diesen Vektor in der Weise, daß er auf TIM zeigt. Nach einem BRK-Befehl übernimmt TIM daher die Kontrolle, druckt B* aus (entry via breakpoint - im Gegensatz zu C*, entry via call), zeigt die Registerinhalte an und wartet auf Befehle vom Operator.

Der oben erwähnte Vektor kann auch verändert werden, etwa um auf eine spezielle Routine zu verzweigen.

Beachten Sie:

1) Nach einem BRK, der auf TIM verzweigt, zeigt der PC auf das dem BRK-Befehl folgende Byte.

2) Bei einem BRK, der nicht auf TIM verzweigt, zeigt der PC nach der Rückkehr (via RTS) auf das zweite Byte nach dem BRK-Befehl.

ANRUF VON TIM

Nach dem Laden des Programms kann TIM entweder mit dem Befehl

RUN (entspricht SYS (1Ø39)),

oder mit dem Befehl

SYS (1Ø24)

aufgerufen werden. Der erste Fall stellt einen "entry by call" dar, der zweite einen "entry via breakpoint".

BEISPIEL:

Das folgende Programm mit dem Namen CHSET schreibt 64 ASCII-Zeichen auf dem Bildschirm. Es soll im Puffer für Recorder #2 gespeichert werden:

```
                    * = $33A
                    CRLF = $4F2
                    WRT  = $FFD2

33A 20 F2 04;  CHSET JSR CRLF
33D A2 2Ø            LDX #$2Ø
33F 8A         LOOP  TXA
340 20 D2 FF         JSR WRT
343 E8               INX
344 EØ 6Ø            CPX #$6Ø
346 DØ F7            BNE LOOP
348 ØØ               BRK
349 4C 3A Ø3         JMP CHSET
```

Um dieses Programm in den PET einzugeben, werden mit dem Befehl

.M Ø33A, Ø34B

die betreffenden Speicherinhalte zur Anzeige gebracht und geändert. Schließlich erhält man die Anzeige:

```
.M   Ø33A, Ø34B
.:  Ø33A 2Ø F2 Ø4 A2 2Ø 8A 2Ø D2
.:  Ø342 FF E8 EØ 6Ø DØ F7 ØØ 4C
.:  Ø34A 3A Ø3
```

commodore

Der Befehl:

.G Ø33A

bringt das Programm jetzt zur Ausführung:

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFG
HIJKLMNOPQRSTUVWXYZ[\]↑← !
```

```
B*  PC  SR AC XR YR SP
.; Ø349 3B 5F 6Ø XX FE
```

Beachten Sie den Inhalt des Programmzählers (siehe BREAK und INTERRUPTS). Es ist jetzt möglich, ohne Angabe der Anfangsadresse das Programm mit dem Befehl:

.G

nochmals zu starten.

Um CHSET von BASIC aus aufrufen zu können, ersetzt man zunächst die BRK-Anweisung in $348 durch ein RTS (return from subroutine). $348 wird also von ØØ auf 6Ø geändert.

Dann ändert man den USR-Vektor in $1 und $2 dermaßen, daß er auf die Subroutine CHSET zeigt:

```
.M ØØØØ,ØØØ2
       Ø  1  2
 ØØØØ 4C 3A Ø3
```

Jetzt kann man zu BASIC zurückkehren mit dem Befehl:

.X

Der Aufruf von CHSET kann nun (von BASIC aus) entweder mit

A = USR (Ø)

oder mit

SYS (826)

erfolgen.

Folgende Programme (Nr. 1 bis Nr. 5) können wir in der englischen Fassung anbieten:

Nr. 1: Target Pong and Off-The-Wall

Zwei Spiele für bis zu 9 Spieler mit 9 möglichen Schwierigkeitsgraden.

Mit zwei Tasten können Sie den Ball ablenken und zugleich Hindernisse aufbauen. Bei Target Pong versuchen Sie, ein bewegliches Ziel zu treffen. Bei Off-The-Wall ist die Aufgabe, den Ball so zu bewegen, daß er weder auf die Spielfeldbegrenzung trifft, noch auf einen der auf dem Feld verteilten Kakteen.

Spitzenleistung in Programmiertechnik.

DM 24,95 incl. MWSt.

Nr. 2: Mortgage

Ein Finanzprogramm. Sie können folgende Daten eingeben: Grundschuld, jährlicher Zinssatz, Beginn der Rückzahlung, Zahlungsperiode, Zeitpunkt der Anleihe, Datum der nächsten Zahlung.

PET berechnet: Den Betrag jeder Rate (unterteilt in Schuld- u. Zinsanteil) noch offene und bereits bezahlte Schulden und Zinsen sowie die Anzahl der Raten.

Sehr übersichtliche und klare Darstellung auf dem Bildschirm des PET.

DM 37,95 incl. MWSt.

Nr. 3: Galaxy Games

Wieder mal unterwegs im Weltraum und wohlversehen mit Torpedos versuchen Sie, einen amoklaufenden Roboter oder mehrere feste Ziele abzuschießen. (Sie können unter zwei Möglichkeiten wählen, Ihr Raumschiff zu steuern, damit Sie nicht etwa in einem der Sterne verglühen.) Für bis zu 5 Spieler. Schwierigkeitsgrad wählbar von 1 bis 9

DM 24,95 incl. MWSt.

Nr. 4: Space Talk, Space Fight

Space Talk enthält die Bedienungsanleitung für das Raumschiff und das Universum.

In der anschließenden Verfolgungsjagd (Space Fight) versuchen Sie, den Schüssen des Mitspielers und anderen Hindernissen auszuweichen und ihm zuvorzukommen.

Für 2 Spieler.

DM 24,95 incl. MWSt.

Nr. 5: Diet Planner and Biorythm

Diet Planner erstellt einen auf Ihre persönlichen Bedürfnisse zugeschnittenen Kalorienplan und, falls Sie abnehmen wollen, eine Liste der zu erwartenden Fortschritte. Die Ein- und Ausgaben erfolgen wahlweise in englischen oder metrischen Maßen.

Biorythm zeichnet nach Eingabe Ihres Geburtsdatums ein Diagramm der emotinalen, intellektuellen und physischen Zyklen für beliebige 30 Tage.

DM 37,95 incl. MWSt.

commodore

Die folgenden Programme sind nur für Mitglieder des PET-Benutzerclubs erhältlich. Die Programme laufen (falls nichts anderes angegeben) auf der 8k-Version des PET und kosten je DM 16,- incl. MWSt. Lieferumfang: Kassette mit Programm.

Die Software ist von uns auf grobe Fehler überprüft, Commodore kann jedoch keinerlei Gewähr übernehmen. Bitte bestellen Sie nur mit beiliegender Bestellkarte.

PC1 : LIFE (Dr. Randow). Spiel.

Eine Gruppe von Individuen vermehrt und vermindert sich nach bestimmten mathematischen Regeln. (Graphisch dargestellt).

PC2 : LOGIK (Dr. Randow). Mathematik.

Ein Programm zur Berechnung von Wahrheitswerten in der Aussagelogik. Mit dreiseitiger Anleitung.

PC3 : PET-UHR 3 und PET-UHR 4 (K. Knäpper und Commodore).

Das Programm enthält zwei verschiedene Versionen, den PET 2001 als Digitaluhr zu verwenden.

PC4 : PRIMZ und PRIMF (Commodore). Mathematik.

Berechnet Primzahlen ab einer vorgegebenen Anfangszahl oder zerlegt eine ganze Zahl in die Primfaktoren.

PC5 : CODEBRECHER (Rolf Krämer). Spiel.

Der Rechner erzeugt eine vierstellige Zahl, die erraten werden soll, Dazu gibt PET Hinweise auf Anzahl und Position der richtigen Ziffern.

PC6 : DREI SPIELE (H.M. Kath).

Das "U-BOOT" soll in einem 10 x 10-Feld aufgespürt und vernichtet werden.
Der "TURM VON HANOI" soll scheibchenweise umgestellt werden, ohne eine größere Scheibe auf eine kleine zu legen.
In "HARTE NUSS" soll ein Muster in das komplementäre Muster umgewandelt werden.

PC7 : KALENDER (F. Scheuer).

Berechnet den Wochentag zu einem Datum, den Abstand zweier Daten und das Datum nach einer beliebigen Anzahl von Tagen. Kann außerdem als fortlaufender Kalender verwendet werden. Gültig zwischen 1.3.1900 und 28.2.2100.

Die folgenden Programme sind nur für Mitglieder des PET-Benutzerclubs erhältlich. Die Programme laufen (falls nichts anderes angegeben) auf der 8k-Version des PET. Lieferumfang: Kassette mit Programm.

PC 8 : ALLESWISSER (A. Dripke)

Prüft Ihre Allgemeinwissen in Geschichte, Naturwissenschaften, Musik, Geographie, usw. Am Ende des Programms erhalten Sie ein Diplom.
Zwei Teile, je ca. 7 kB (insgesamt 14086 Bytes) DM 21,-*

PC 9 : DIFFERENTIALGLEICHUNGEN 1. UND 2. ORDNUNG (K. Lehner)

Das Programm löst numerisch folgende Aufgaben nach der Runge-Kutta-Methode: y' = f(x,y) oder y' = f(x,y,z) und z' = g(x,y,z) oder y" = f(x,y,y') oder y" = f(x,y). Mit kurzer Anleitung. 2460 Bytes DM 16,-*

PC1Ø : CHECKERS (F. Vischer)

Eine amerikanische Version des Damenspiels. Wird auf einem kreuzförmigen Brett von einer Person gespielt. Es sollen möglichst viele Steine durch Überspringen in geschickter Reihenfolge entfernt werden. 4701 Bytes DM 16,-*

PC11 : TRANSISTOR-DC-ANALYSE (L. Miedel)

Berechnet den DC-Arbeitspunkt einer Transistorstufe mit Gegenkopplung. Verschiedene Schaltungsparameter können geändert werden, wobei die Auswirkungen auf die Schaltung gezeigt werden (I, U, P). 6326 Bytes DM 16,-*

PC12 : PET-MONITOR (Commodore)

TIM (Terminal Interface Monitor) wurde für die Mikroprozessoren der Serie 65xx entwickelt. Eine erweiterte und an den PET angepaßte Version steht auf Kassette zur Verfügung. Dieses Programm ist vor allem für Programmierung in Maschinensprache (hexadezimale Ein- und Ausgabe) nützlich.
TIM verfügt über die Befehle:

M	display memory;	Anzeige des Speicherinhalts
R	display register;	Anzeige der Registerinhalte
G	begin execution;	Ausführung eines PGMs in Maschinenspr.
X	exit to BASIC;	Rückkehr zu BASIC
L	load;	Laden eines PGMs von Kassette
S	save;	Überspielen eines PGMs auf Kassette

TIM wird mit 6 Seiten Anleitung geliefert. 784 Bytes
DM 16,-*

---

*Die Software ist von uns auf grobe Fehler überprüft, Commodore kann jedoch keinerlei Gewähr übernehmen. Bitte bestellen Sie nur mit beiliegender Bestellkarte. Alle Preisangaben auf dieser Seite beinhalten die gesetzliche MWSt.

commodore

? Wie wird DEFFN verwendet?

A: B E I S P I E L (B1):
```
 Ø REM B1
10 INPUT"X";X
20 DEFFNA(X)=2↑X
30 DEFFNB(X)=3↑X
40 DEFFNC(X)=4↑X
50 PRINT:PRINT
60 PRINT"2↑";X;"=";FNA(X)
70 PRINT"3↑";X;"=";FNB(X)
80 PRINT"4↑";X;"=";FNC(X)
90 PRINT:PRINT:PRINT:GOTO 1Ø
```

Die Variablen A, B und C in Zeilen 20, 30 und 40 sind sogenannte "dummy"-Variablen (Variablen ohne zugewiesenen Wert). Für verschiedene Funktionen im gleichen Programm müssen verschiedene Dummy-Variablen verwendet werden; zulässig sind A, B, C ..... Z. Sobald das obige Programm einmal gelaufen ist, können die drei Funktionen auch über die Tastatur aufgerufen werden;

B E I S P I E L : ?FNB(7)

ergibt 2187(= 3↑7).

? Kann man erreichen, daß der Bildschirm nicht zeilenweise, sondern auf einmal vollgeschrieben wird?

A: Verwenden Sie den Befehl POKE 594Ø9,52 (Bildschirm dunkel), schreiben Sie den Text und geben Sie dann den Befehl POKE 594Ø9,6Ø . Die Anzeige erscheint auch dann wieder, wenn der Cursor in Zeile 25 steht und ein Zeilenvorschub erfolgt.

? Wie kann ich zuverlässig Daten auf die Kassette schreiben und einlesen?

A: Im folgenden bedeutet n die Nummer des betreffenden Files. n ist eine ganze Zahl von 1 bis 255.

1. Beim Schreiben des Programms darf für PRINT#n die Abkürzung ?#n nicht verwendet werden. ?#n würde bei der Ausführung des Programms einen ?SYNTAX ERROR zur Folge haben.

2. Nach mindestens 79 Zeichen Output muß ein "carriage return" - CHR$(13) - ausgegeben werden, falls die Daten später mit INPUT#n eingelesen werden sollen. CHR$(13) kann auch öfter als nach jeweils 79 Zeichen geschrieben werden.

   Sollen die Daten mit GET#n eingelesen werden, so ist CHR$(13) nicht erforderlich.

3. GET#n liest über EOF (end of file) hinweg, wenn dies nicht verhindert wird. Nach GET#n sollte deshalb generell der Status abgefragt werden. (IF ST<>ØTHEN ...).

4. Wurde das File mit einem Namen bezeichnet, dann muß dieser Name auch im OPEN-Befehl zum Einlesen stehen. Beachten Sie Zeile 110 im folgenden Beispiel:

```
 1Ø OPEN 72,1,1"FILE1"
          :
 6Ø CLOSE 72
          :
11Ø OPEN 72,1,Ø,"FILE1"
          :
16Ø CLOSE 72
```

5. PRINT#n schreibt in derselben Weise auf das Band, wie der Befehl PRINT auf den Bildschirm schreiben würde, wenn dieser nur aus einer einzigen, entsprechend langen Zeile bestünde.

6. Werden mit PRINT#n mehrere Daten in eine "Zeile" geschrieben, so müssen sie durch ein Komma "," voneinander getrennt werden.

commodore

7. Anstelle des Befehls INPUT#n,X,Y kann man in Zweifelsfällen auch INPUT n,X$,Y$ verwenden. Anschließend wandelt man die String- wieder in Zahlvariablen um.

B E I S P I E L (B2):

```
 Ø REM B2
 1Ø OPEN 1,1,1
 2Ø FOR J=1 TO 2.9 STEP Ø.1
 3Ø PRINT#1,J;",";J*1Ø
 35 PRINT#1,CHR$(13)
 4Ø NEXTJ
 5Ø CLOSE1
 6Ø PRINT"BITTE DAS BAND ZURUECKSPULEN
    UND DANN EINE TASTE DRUECKEN"
 7Ø GETA$:IFA$= ""THEN 7Ø
 8Ø OPEN1
 9Ø FORJ=1TO2Ø
1ØØ INPUT#1,X$,Y$
1Ø5 X=VAL(X$):Y=VAL(Y$)
11Ø PRINT X,Y
12Ø NEXTJ
13Ø CLOSE1
14Ø PRINT"FERTIG!"
```

Zeile 30: Beachten Sie die Trennung der Variablen J und J*1Ø durch ",".

Zeile 35: Siehe Punkt 2.

Zeile 60: Die Daten sind jetzt auf das Band geschrieben und sollen im folgenden wieder eingelesen werden.

Zeile 100: Die Daten werden als Strings eingelesen.

Zeile 105: Die Strings werden in Zahlvariablen umgewandelt und in

Zeile 110 auf dem Bildschirm ausgegeben.

8. Der Pufferspeicher für die Kassette #1 reicht von einschließlich 634 bis 825 (dezimale Adressen) bzw. von $Ø27A bis $Ø339 (Hexadezimaladressen).

   Der Puffer für die externe Kassette #2 reicht von 826 bis 1Ø17 bzw. von $Ø33A bis $Ø3F9.

9. Beim Schreiben von Daten auf die Kassette bewegt der Motor das Band ein kurzes Stück weiter, sobald ein Record geschrieben ist. Ein Record entspricht 191 Bytes (Bufferkapazität minus eins).

   Beim Einlesen der Daten kann es geschehen, daß der Abstand zwischen den einzelnen Records zu gering ist, insbesondere bei Verwendung von leichtgängigen Kassetten. Dies führt dann zu Fehlern.

   ABHILFE: Verwenden Sie beim Schreiben der Daten eine Routine, die mindestens nach jeweils 191 ausgegebenen Zeichen den Motor für kurze Zeit einschaltet, ohne daß eine Datenaufzeichnung erfolgt.

B E I S P I E L (B3):

```
 Ø REM B3
 1Ø BU=Ø
 2Ø OPEN 2,1,1
 3Ø FORA=1 TO 999
 4Ø A$=STR$(A)
 5Ø LA=LEN(A$)
 55 BU=BU+LA
 6Ø IF BU<191 THEN 12Ø
 7Ø POKE 59411,53
 8Ø T=TI
 9Ø IF(TI-T)<5THEN9Ø
1ØØ POKE 59411,61
11Ø BU=BU-191
12Ø PRINT#2,A
13Ø NEXTA
14Ø CLOSE 2
15Ø END
```

commodore

```
5ØØ J=Ø
51Ø OPEN 2
52Ø FOR I = 1 TO 999
53Ø INPUT#2,A
54Ø IF(ST)AND64THEN58Ø
55Ø PRINTA
56Ø IF A<>B+1 THENPRINT"FEHLER":J=J+1
57Ø B=A:NEXTI
58Ø CLOSE2
59Ø PRINT:PRINTJ"FEHLER"
```

Zeile 40: Umwandlung der Zahlen (1 bis 999) in Strings.

Zeile 50: Berechnung der Anzahl Bytes, die für die Speicherung des Strings nötig sind.

Zeile 70: Einschalten des Motors von Recorder #1.

Zeile 90: Warten für 5/6Ø Sekunden.

Zeile 100: Ausschalten des Motors von Recorder#1.

Mit RUN5ØØ kann anschließend die einwandfreie Aufzeichnung der Daten überprüft werden.

Bei Verwendung des externen Recorders kann mit den Befehlen POKE 59456,239 und POKE 59456,255 der Motor des Recorders#2 ein- bzw. ausgeschaltet werden.

? Wie kann ich Programme in Maschinensprache schreiben?

A.: (Für Fragen über die Maschinensprache des Mikroprozessors MOS 65Ø2 empfehlen wir als Literatur die Programmierfiebel von MOS-Technology, siehe Seite A Ø3).

Angenommen, Sie möchten die Inhalte zweier Speicherplätze (839 und 840) addieren und das Ergebnis in einem dritten Speicherplatz (841) ablegen.
Ein Assemblerprogramm könnte folgendermaßen aussehen:

```
CLC           Clear Carry-Flag
LDA  $Ø347    Lade-Akku mit Inhalt von 839
ADC  $Ø348    Addiere zum Akku den Inhalt von 840
STA  $Ø349    Speichere das Ergebnis in 841
RTS           Return (Rückkehr aus der Subroutine)
```

Umgesetzt in Maschinensprache erhält man:

```
 18
 AD  71  3
 6D  72  3     (Hexadezimalcode; Adressen im Doppel-
 8D  73  3      byte-Hexcode)
 6Ø
```

Dieses Maschinenprogramm soll ab Adresse 826 gespeichert werden. 826 bis 1Ø17 ist der Puffer für den Recorder #2 und damit ein "sicherer" Bereich. (Sofern man den Recorder #2 nicht verwendet, wird dieser Bereich von BASIC nicht benutzt.)

Zunächst werden die Befehle nach Dezimal umgewandelt:

```
 24
173  71  3
109  72  3     (Dezimalcode; Adressen im Doppel-
141  73  3      byte-Dezimalcode)
 96
```

und dann mit einem kurzen BASIC-Programm in den Speicher geschrieben:

```
1Ø DATA24,173,71,3,1Ø9,72,3,141,73,3,96
2Ø FORM=826TO836
3Ø READA
4Ø POKEM,A
5Ø NEXT
```

Zur Verbindung des obigen Maschinenprogramms mit BASIC kann man den Befehl SYS (826) verwenden.

* SYS(A) bewirkt einen Sprung nach der Dezimaladresse A. Der
* Inhalt von A wird als Maschinenprogramm verstanden und so-
* lange ausgeführt, bis der Befehl RTS (Ø6Ø) erkannt wird.
* Nach RTS wird die Kontrolle an den auf SYS(A) folgenden
* Befehl übergeben. RTS bewirkt also die Rückkehr aus dem
* Maschinenprogramm.

Da der Befehl NEW den Pufferspeicher nicht löscht, kann man jetzt NEW eingeben und etwa folgendes Programm:

```
1Ø INPUT"1.ZAHL";Z1:POKE839,Z1
2Ø INPUT"2.ZAHL";Z2:POKE84Ø,Z2
3Ø SYS(826)
4Ø PRINTZ1"+"Z2"="PEEK(841)
```

Ein weiteres Beispiel für ein Programm in Assembler:

```
            LDX  #ØØØ      Lade Indexregister X mit Ø
SCHLEIFE    TXA            Bringe Inhalt von X in den Akkumulator
            STA  Ø815Ø,X   Speichere Inhalt des Akk. in (Ø8150+X)
            INX            Erhöhe X um 1
            BNE SCHLEIFE   Ist das Ergebnis ungleich Ø, dann
                           verzweige nach SCHLEIFE, sonst
            RTS            Return (Rückkehr aus der Subroutine)
```

Es kann mit folgendem BASIC-Programm eingegeben werden:

```
10 DATA 162,0,138,157,80,129,232,208,249,96
20 FOR M=826TO835
30 READ A
40 POKE M,A
50 NEXT
```

und wird dann mit SYS(826) gestartet.

Für einen Vergleich der Ausführungszeiten Maschinenprogramm/ BASIC-Programm ist unten noch ein BASIC-Programm angegeben. Es bewirkt dasselbe, wie obiges Maschinenprogramm.

```
100 A=0
110 POKE33104+A,A
120 A=A+1
130 IFA<256THEN110
```

Bem.: 32768 bis 33767 sind die Adressen des Bildschirmspeichers.

? Wo anders als im Puffer für Recorder #2 kann ich ein Maschinenprogramm speichern, ohne daß es von BASIC überschrieben wird?

A.: Die Dezimaladressen 134 und 135 im PET stellen einen Pointer dar, der auf das Ende des Schreib-Lese-Speichers +1 zeigt. 134 enthält die 8 niederwertigen und 135 die 8 höherwertigen Bits dieses Binärpointers.

In der 8 k-Version:

	Inhalt (binär)	Inhalt (Doppelbyte-Dez.)
Adresse 134	ØØØØØØØØ	Ø
Adresse 135	ØØ1ØØØØØ	32

Der Befehl

```
? PEEK(134)+256*PEEK(135)-1
```

gibt also die Dezimaladresse des obersten Speicherplatzes im RAM an.
Verändert man unmittelbar nach dem Einschalten oder nach einem NEW-Befehl den Inhalt von 134 und 135 beispielsweise auf 6ØØØ, so existiert für BASIC der darüberliegende Speicherplatz nicht mehr und kann für andere Zwecke verwendet werden.

Also:

```
NEW
? 6ØØØ/256
  23.4375
? 6ØØØ-23*256
  112
POKE 134,112
POKE 135,23
```

Die Obergrenze für BASIC ist damit auf 6ØØØ festgelegt.
?FRE(Ø) zeigt 4972 (=5999 minus 1024 für den Arbeitsspeicher minus 3 für den Befehl ?FRE(Ø)).
Ab 6ØØØ bis 8191 können Sie jetzt das Maschinenprogramm speichern.

ABKÜRZUNGEN FÜR BASIC-BEFEHLE

Viele BASIC-Befehle können abgekürzt werden. Bei der Programmierung ist aber folgendes zu beachten: Es ist möglich, beispielsweise 39 PRINT-Befehle in einer Zeile unterzubringen, indem man nach der Zeilennummer 39 Mal die Abkürzung ?: eingibt. Es ist aber dann nicht mehr möglich, diese Zeile ohne weiteres zu ändern (normalerweise sind nur 80 Zeichen pro Zeile zulässig).

Untenstehende Abkürzungen sind unabhängig von der Betriebsart Groß- oder Kleinschreibung. Lediglich die Darstellung auf dem Bildschirm erscheint verschieden.

BEFEHL	ABK	BEFEHL	ABK
ABS	Ab	OPEN	Op
AND	An	PEEK	Pe
ASC	As	POKE	Po
ATN	At	PRINT	?
CHR$	Ch	PRINT#	Pr
CLOSE	CLo	READ	Re
CLR	Cl	RESTORE	REs
CMD	Cm	RETURN	REt
CONT	Co	RIGHT$	Ri
DATA	Da	RND	Rn
DEF	De	RUN	Ru
DIM	Di	SAVE	Sa
END	En	SGN	Sg
EXP	Ex	SIN	Si
FOR	Fo	SPC	Sp
FRE	Fr	SQR	Sq
GET	Ge	STEP	STe
GOSUB	GOs	STOP	St
GOTO	Go	STR$	STr
INPUT#	In	SYS	Sy
LEFT$	LEf	TAB	Ta
LET	Le	TAN	TAN
LIST	Li	THEN	Th
LOAD	Lo	USR	Us
LOG	LOG	VAL	Va
MID$	Mi	VERIFY	Ve
NEXT	Ne	WAIT	Wa
NOT	No		

commodore

OVERLAYS

BASIC-Programme werden im PET von Adresse 1Ø24 an aufwärts gespeichert, siehe auch FWØ4. Die Pointer (Zeiger in die nächste BASIC-Zeile) und Zeilennummern sind in je zwei Bytes (niederwertiges - höherwertiges) codiert. Das höherwertige Byte mal 256 wird zum niederwertigen Byte addiert, um den tatsächlichen Wert zu erhalten.

Wenn ein Programm den LOAD-Befehl enthält, so impliziert dieser weder CLR noch NEW. Das neue Programm wird eingelesen, ab 1Ø24 gespeichert und abgearbeitet. (Bei einigen BASIC-Versionen ersetzt das neue PGM nur die entsprechenden Zeilennummern; beim PET ist dies nicht so.) Alle Befehle des Aufrufprogramms aber, die über dem Platzbedarf des geladenen Programms liegen, bleiben unverändert. Einzig der Zeiger im neuen Programm zum Rest des alten ist nicht vorhanden.

Nennen wir also P1 das aufrufende PGM und P2 das von P1 geladene PGM und sind folgende Bedingungen erfüllt:

* Die Befehle von P1, die erhalten bleiben sollen, stehen in Zeilen mit hohen Nummern;
* Die Befehle von P1, die erhalten bleiben sollen, werden durch P2 nicht überschrieben (d.h.: P2 ist kleiner, als der nicht mehr benötigte Teil von P1);
* Die Befehle von P1, die erhalten bleiben sollen, haben Zeilennummern, die in P2 nicht vorkommen;

Dann kann man folgendermaßen vorgehen:

* Man suche in P1 den letzten Befehl, der gelöscht werden soll. Dieser enthält einen Zeiger in die nächste Zeile (die erhalten bleiben soll).
* Man speichere diesen Zeiger und
* Nach dem Laden von P2 suche man die letzte Zeile von P2 und schreibe den obenerwähnten Zeiger in diese Zeile.

Beispiel: Folgende Programme sollen hintereinander auf Kassette gespeichert werden:

P1:

```
10 ?"HIER IST P1
20 LOAD
5000 ?"DIESE ZEILE SOLL ERHALTEN BLEIBEN"
```

P2:

```
10 ?"HIER IST P2"
20 GOTO5000
```

P1 würde zwar ausdrucken:

HIER IST P1!

und dann das nächste PGM von der Kassette laden. Damit Zeile 5000 aber erhalten bleibt, muß man unmittelbar vor dem LOAD-Befehl einfügen:

```
16 GOSUB4998
18 Z1=PEEK(Z4):Z2=PEEK(Z4+1)
```

und unmittelbar vor der Zeile 5000 (die erhalten bleiben soll):

```
4998 Z4=256*PEEK(202)+PEEK(201)+3
4999 RETURN
```

Z4 enthält die Dezimaladresse des Zeigers aus Zeile 4999, der auf die Zeile 5000 zeigt (siehe auch FW09). Z1+256*Z2 ergibt die Dezimaladresse des Beginns von Zeile 5000.
Fügen wir zur Sicherheit noch Zeile 9 hinzu, so erhalten wir:

```
9 REM DIESE ZEILE DIENT NUR ZUR VERGROESSERUNG VON P1
10 PRINT"HIER IST P1!"
16 GOSUB4998
18 Z1=PEEK(Z4):Z2=PEEK(Z4+1)
20 LOAD
4998 Z4=256*PEEK(202)+PEEK(201)+3
4999 RETURN
5000 PRINT"DIESE ZEILE SOLL ERHALTEN BLEIBEN"
```

P1 kann jetzt mit SAVE aufs Band geschrieben werden.

In P2 muß als erstes der Zeiger auf Zeile 5000 gesetzt werden (Z1 und Z2 stammen von P1):

```
1 GOSUB3998
2 POKEZ3,Z1:POKEZ3+1,Z2
```

als letzte Befehle von P2 fügen wir ein:

```
3998 Z3=256*PEEK(202)+PEEK(201)+3
3999 RETURN
```

Z3 gibt die Dezimaladresse des Zeigers aus Zeile 3999 an, der auf das Ende des PGM's zeigt. Dieser Zeiger wird in Zeile 2 so umgestellt, daß er auf die (noch immer vorhandene) Zeile 5000 zeigt. Die Variablen Z1 und Z2 wurden durch das LOAD nicht gelöscht.

Wir haben also:
(P2)

```
1 GOSUB3998
2 POKEZ3,Z1:POKEZ3+1,Z2
10 PRINT"HIER IST P2!"
20 GOTO5000
3998 Z3=256*PEEK(202)+PEEK(201)+3
3999 RETURN
```

P2 kann jetzt hinter P1 auf das Band geschrieben werden. Rückspulen, dann LOAD und RUN ergibt, wie gewünscht:

```
HIER IST P1!
HIER IST P2!
DIESE ZEILE SOLL ERHALTEN BLEIBEN
```

ABSCHNEIDEN UND RUNDEN VON DEZIMALZAHLEN

```
10 INPUT"ZAHL,NACHKOMMASTELLEN";Z,N
20 PRINT INT(Z*10↑N)/10↑N    :REM ABSCHNEIDEN
30 PRINT INT(Z*10↑N+.5)/10↑N :REM RUNDEN
40 GOTO10
```

INT(10↑N)

TASTATURPUFFER

Der PET-Computer kann (im allgemeinen auch während der Abarbeitung eines Programms) bis zu 10 über die Tastatur eingegebene Zeichen speichern.

* Diese Zeichen werden in die RAM-Speicherplätze 527 bis 536
* geschrieben. Die Anzahl der eingegebenen Zeichen steht im
* Speicherplatz 525.

Unmittelbar nach der Anzeige READY werden diese Zeichen auf den Bildschirm geschrieben.

Beispiel 1: Geben Sie ein

```
POKE 527,33 : POKE 525,1
```

Der erste Befehl schreibt den ASCII-Code für ! in den Tastaturpuffer, der zweite Befehl teilt dem PET mit, daß 1 Zeichen in diesem Puffer gültig ist.

Diese Fähigkeit des PET kann unter anderem dazu benutzt werden, einen Text auf dem Bildschirm zu schreiben und die Betätigung der RETURN-Taste zu simulieren.

Beispiel 2:

```
2 DATA71,207,50,13:REM * ASCII FUER "G┌2 [RETURN]"
10 PRINT"DIESES PROGRAMM LISTET SICH SELBST"
12 PRINT"SOOFT SIE DIE TASTE L DRUECKEN":PRINT
16 GETR$:IFR$=""THEN16
18 IFR$<>"L"THEN16
20 RESTORE
22 FORI=527TO530:READD:POKEI,D:NEXT
24 POKE525,4:REM * 4 ZEICHEN IM PUFFER
26 LIST
```

Dieses Programm listet sich selbst (Zeile 26). Dann wird (wie üblich) READY auf dem Bildschirm geschrieben. Der Computer hat damit das Programm verlassen.

Im Tastaturpuffer stehen noch die vier Zeichen:

```
Go2 (RETURN)
```

Diese Zeichen werden jetzt auf dem Bildschirm ausgegeben und bewirken dasselbe, was die manuelle Eingabe von GOTO2 (RETURN) bewirken würde.

commodore

Beispiel 3:

Oft ist es wünschenswert, einen über INPUT eingegebenen Ausdruck als Programmzeile zu übernehmen.

In diesen und ähnlichen Fällen kann man so vorgehen:

```
5 DATA 145,145,145,145,13,71,207,50,48,13
10 INPUT"FNA(X)=";A$:PRINT"20DEFFNA(X)="A$
12 FORI=527TO536:READJ:POKEI,J:NEXT:POKE525,10
14 STOP
26 PRINT:PRINT:PRINT"FUNKTION IST DEFINIERT"
```

Zeile 5 enthält den ASCII-Code für:

4 Mal Cursor nach oben, RETURN, Go2Ø, RETURN;

Zeile 1Ø verlangt die Eingabe einer beliebigen Funktionsausdrucks und schreibt diesen auf den Bildschirm;

Zeile 12 schreibt in den Tastaturpuffer;

Zeile 14 unterbricht das Programm. Es erscheint READY und der Inhalt des Tastaturpuffers. Nach Ausführen des vierfachen Cursor nach oben steht der Cursor auf der von Programmzeile 1Ø ausgegebenen Bildschirmzeile:

2ØDEFFNA(X)= ...

und übernimmt diese Zeile mit RETURN ins Programm. Der Puffer ist noch nicht leer, dies bewirkt schließlich ein GOTO2Ø.

Beachten Sie, daß eine Programmänderung erfolgte. Zeile 2Ø ist neu. Wie bei jeder Änderung des Programms werden dadurch alle Variablen initialisiert. Nötigenfalls müssen in solchen Fällen Variablen mit POKE "gerettet" und nach der Programmänderung mit PEEK wieder geholt werden.

Die obigen Beispiele zeigen nur einen Teil der Möglichkeiten dieses Verfahrens. Man denke etwa an sich selbst verändernde (lernende) Programme, an berechnete Sprünge der Form ONXGOTOX, oder an die Möglichkeit, bei INPUT auch Terme wie SIN(X) zuzulassen. Auch ist damit programmgesteuertes LOAD ohne die in SW11-13 erwähnten Einschränkungen möglich.

OVERLAYS II

Geben Sie folgendes Programm in den PET ein:

```
20 PRINT"HIER IST ZEILE 20 (OVL 1)"
30 GOTO 0
```

legen Sie eine Leerkassette in den Recorder#1, und nehmen Sie das PGM als Datenfile auf, indem Sie direkt eingeben:

```
POKE243,122:POKE244,2:OPEN1,1,1:CMD1:LIST
```

Wenn der Cursor wieder erscheint, dann schließen Sie das File und löschen den PGM-Speicher mit:

```
PRINT#1:CLOSE1:NEW
```

Geben Sie als neues Programm ein:

```
30 PRINT"HIER IST ZEILE 30 (OVL 2)"
40 LIST
```

und zeichnen Sie es auf wie das erste Programm (hinter dieses).

Geben Sie jetzt als drittes und letztes Programm ein:

```
0 OPEN1
2 POKE611,1
3 POKE527,71:POKE528,207:POKE529,53:POKE530,13:POKE525,4
4 END
5 POKE611,0
6 IFST=0THEN2
7 GOTO20
8 END
```

Spulen Sie das Band mit den beiden Overlayprogramme zurück und starten Sie (endlich) mit RUN.

Folgendes sollte geschehen:

Zeile 0 eröffnet das File#1 zum Lesen von Recorder#1

Zeile 2 verändert die Nummer des Gerätes von dem der PET Daten erwartet.

Dies bewirkt, daß automatisch Text von Recorder#1 eingelesen wird, bis ein CR-Zeichen - CHR$(19) - angetroffen wird. Das CR bewirkt Übernahme des Textes (1 Programmzeile) in den Speicher.

commodore

Zeile 3 Hier wird direkt in den Tastaturpuffer geschrieben:

GO5 (RETURN)

Dies bewirkt dasselbe, wie die manuelle Eingabe von GOTO5.

Zeile 5 Stellt den ursprünglichen Zustand bezüglich des Input-Geräts wieder her (Tastatur).

Zeile 6 Fragt auf korrekte Datenübernahme und EOF (Fileende) ab. Ist ST=Ø, dann wird die nächste Programmzeile eingelesen.

Ansonsten erfolgt ein Sprung in Zeile 2Ø (die inzwischen als Overlay eingelesen wurde), von da nach Ø, was das Einlesen des 2. Overlays bewirkt. Schließlich führt dies zum Ausdruck von:

```
HIER IST ZEILE 20 <OVL 1>
HIER IST ZEILE 30 <OVL 2>
```

und zum Listen des neuen Programms.

Mit der hier angedeuteten Technik können sowohl einzelne Zeilen (etwa DATA-Befehle) überschreiben werden, als auch ganze Subroutinen. Ebenfalls können neue Zeilen an beliebiger Stelle des ursprünglichen Programms eingefügt werden. Beachten Sie, daß eine solche Programmänderung immer CLR impliziert.

* Die einzige bisher bekannte Einschränkung ist, daß Zeile 1
* nicht verwendet werden sollte. Sie wird gelöscht, bzw. nicht
* mit übernommen. Dies wird weiter untersucht.

PEEK INS ROM

Der PEEK-Befehl erlaubt nicht den Zugang zum Inhalt der ROM's. Man kann stattdessen folgendes Maschinenprogramm verwenden:

```
*** = $1FF9

$1FF9  AD NB,HB     lade Akumulator mit Inhalt der Adresse
                    NB + 256*HB. (Die Speicherplätze $1FFA
                    und $1FFB werden von BASIC mit NB und
                    HB beschrieben.)

$1FFC  8D F8 1F     Speichere das Ergebnis in $1FF8

$1FFF  6Ø           Rückkehr von der Subroutine
```

oder dezimal:

```
8185   173,NB,HB    LDA AD

8186   141,248,31   STA 8184

8191    96          RTS
```

Das zugehörige BASIC-Programm könnte etwa folgendermaßen aussehen:

```
0 POKE134,249:POKE135,31:REM * OBERGRENZE BASIC-RAM WIRD 8184 *
1 DATA173,0,0,141,248,31,96
2 FORI=8185TO8191:READJ:POKEI,J:NEXT
10 INPUT"AB ADRESSE";AD
20 GOSUB1000
30 AD=AD+1:GOTO20
1000 REM * DIESES UNTERPROGRAMM BERECHNET VON ADRESSE AD DAS
1001 REM   HOEHERWERTIGE (HB) UND DAS NIEDERWERTIGE BYTE (NB). *
1002 REM * ENTSPRICHT INSGESAMT DEM BEFEHL ?PEEK(AD).
1003 REM * AUCH FUER ROM-ADRESSEN *
1010 HB=INT(AD/256):NB=AD-256*HB
1020 POKE8186,NB:POKE8187,HB
1030 SYS 8185:    PRINTPEEK(8184)
1040 RETURN
```

commodore

RECHENGESCHWINDIGKEIT

Oft stehen Vergleichsbefehle innerhalb von Schleifen und werden dann vom Programm sehr oft durchlaufen.

In solchen Fällen ist es sinnvoll, auf die Rechenzeit zu achten. Beispiel: Die Programmzeile:

```
IF I = 2 AND J = 10 AND K = 40 THEN ...
```

bewirkt, daß jedesmal alle drei Bedingungen überprüft werden. Wesentlich schneller wird das Programm mit folgendem Vergleich:

```
IF I = 2 THEN IF J = 10 THEN IF K = 40 THEN ...
```

Hier wird J nur dann geprüft, wenn I = 2 erfüllt ist und K nur dann, wenn I und J richtig sind.

50.5333334 SEKUNDEN

```
1 OPEN4,4:CMD4:TV=TI
2 FORI=1TO99
3 FORJ=1TO99
4 FORK=1TO99
5 IFI=2THENIFJ=8THENIFK=9THEN7
6 NEXTK,J,I
7 TN=TI
8 PRINT(TN-TV)/60"SEKUNDEN"
9 LIST
```

112.05 SEKUNDEN

```
1 OPEN4,4:CMD4:TV=TI
2 FORI=1TO99
3 FORJ=1TO99
4 FORK=1TO99
5 IFI=2ANDJ=8ANDK=9THEN7
6 NEXTK,J,I
7 TN=TI
8 PRINT(TN-TV)/60"SEKUNDEN"
9 LIST
```

```
0 REM              *** A E N D E R U N G ***
1 REM
2 REM* VERBESSERUNGSVORSCHLAG FUER RUNDEN UND ABSCHNEIDEN IN SW13.
3 REM* PGM IN SW13 LAEUFT NICHT KORREKT FUER Z.B.Z=1.0003 UND N=3
4 REM* VON A.SCHUMACHER, BAD BEVENSEN
5 REM
10 INPUT"ZAHL,NACHKOMMASTELLEN";Z,N
20 PRINTINT(Z*10↑N)/INT(10↑N)    :REM ABSCHNEIDEN
30 PRINTINT(Z*10↑N+.5)/INT(10↑N):REM RUNDEN
40 GOTO10
```

commodore

KLUBMITTEILUNGEN

A03 010978 ..... LITERATUR
A04 151078 ..... KLUBBEDINGUNGEN
A05 151078 ..... PROGRAMMANGEBOTE
A06 151078 ..... PROGRAMMANGEBOTE
A07 151078 ..... KURSOR
A08 150978 ..... RECHENGESCHWINDIGKEIT
A09 150978 ..... RECHENGESCHW. UND SPEICHERBEDARF
A10 150978 ..... SPEICHERBEDARF

ANGEBOTE1 010878
ANGEBOTE2 010878
ANGEBOTE3 150978
ANGEBOTE4 011078
ANGEBOTE5 151278
ANGEBOTE6 151278
ANGEBOTE7 151278
ANGEBOTE8 150379
ANGEBOTE9 150379

FW01 010778 ..... BASIC-BUGS
FW02 010778 ..... BASIC-BUGS
FW03 010878 ..... OPEN (KASSETTE)
FW04 011078 ..... SPEICHERUNG VON PROGRAMMEN
FW05 011078 ..... INTERPRETERCODE
FW06 011078 ..... RAM-ZUTEILUNG UND POINTER
FW07 011078 ..... ADRESSENBELEGUNG
FW08 151278 ..... PAGE 0
FW09 151278 ..... PAGE 0 UND 1
FW10 151278 ..... PAGE 2 UND 3
FW11 150379 ..... ASCII-CODE (BILDSCHIRM)

HW01 010978 ..... DIVERSES
HW02 010978 ..... KURZINFO DRUCKER
HW03 010878 ..... RECORDERSERVICE
HW04 010978 ..... KURZINFO DATASSETTE
HW05 150379 ..... AUFZEICHNUNGSFORMAT KASSETTE

P01 150978 ..... PROGRAMMANGEBOTE ALLGEMEIN
P02 150978 ..... PROGRAMMANGEBOTE ALLGEMEIN
PC01 151278 ..... PROGRAMMANGEBOTE FUER KLUBMITGLIEDER
PC02 150379 ..... PROGRAMMANGEBOTE FUER KLUBMITGLIEDER

SW01 010878 ..... DEFFN, BILDSCHIRM, FILES
SW02 010778 ..... KASSETTE INPUT-OUTPUT
SW03 010778 ..... KASSETTE INPUT-OUTPUT
SW04 010978 ..... KASSETTE INPUT-OUTPUT
SW05 010978 ..... KASSETTE INPUT-OUTPUT
SW06 010978 ..... MASCHINENSPRACHE
SW07 010978 ..... MASCHINENSPRACHE
SW08 010978 ..... MASCHINENSPRACHE
SW09 010978 ..... MASCHINENSPRACHE
SW10 010978 ..... ABKUERZUNGEN DER BASIC-BEFEHLE
SW11 151278 ..... OVERLAYS
SW12 151278 ..... OVERLAYS
SW13 151278 ..... OVERLAYS. ABSCHNEIDEN UND RUNDEN
SW14 150379 ..... TASTATURPUFFER
SW15 150379 ..... TASTATURPUFFER
SW16 150379 ..... OVERLAYS II
SW17 150379 ..... OVERLAYS II

I N H A L T

A - ALLGEMEINES

ANGEBOTE (VERMITTLUNG)

FW - FIRMWARE

HW - HARDWARE

commodore

P - LIEFERBARE PROGRAMME

PØ1 150978
PØ2 150978

PC - PROGRAMME DES CLUBS

PCØ1 151278

SW - SOFTWARE

SWØ1	Ø1Ø878	.....	DEFFN, Bildschirm, Files
SWØ2	Ø1Ø778	.....	Kassette INPUT-OUTPUT
SWØ3	Ø1Ø778	.....	Kassette INPUT-OUTPUT
SWØ4	Ø1Ø978	.....	Kassette INPUT-OUTPUT
SWØ5	Ø1Ø978	.....	Kassette INPUT-OUTPUT
SWØ6	Ø1Ø978	.....	Maschinensprache
SWØ7	Ø1Ø978	.....	Maschinensprache
SWØ8	Ø1Ø978	.....	Maschinensprache
SWØ9	Ø1Ø978	.....	Maschinensprache
SW1Ø	Ø1Ø978	.....	Abkürzung für BASIC-Befehle
SW11	151278	.....	Overlays
SW12	151278	.....	Overlays
SW13	151278	.....	Overlays. Abschneiden und runden.

MOA
MOC