C16/P4-SPECIAL

# P/4 C16 116

Nr. 1/89 DM 14,80/ÖS 124/SFR 14,80

## SPECIAL

## Die beste Software

## Die besten Tips & Tricks

## Die besten Programme

## Machen Sie mehr aus Ihrem C16/P4

# SOFT-WARE JAHR-BUCH 1989

Sonderheft Nr. 1/89 — DM 14,80/ÖS 124/SFR 14,80 CA-Special

COMPUTER-TEST-JAHRBUCH

# COMPUTER TEST JAHR BUCH

Rund 150 Seiten Kaufberatung

Computer von A bis Z

Was sie können
Was sie kosten
Was sie leisten

Jetzt an Ihrem Kiosk

Der Computer-Einkaufsführer

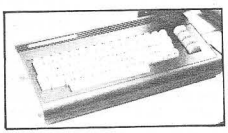
Für Sie getestet:
Computer
Programme
Zubehör

## GRUNDLAGEN

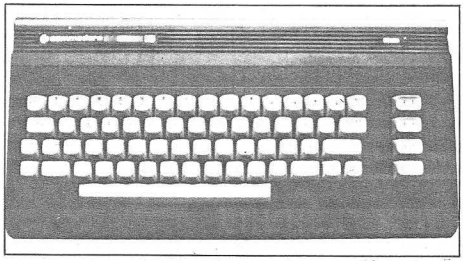


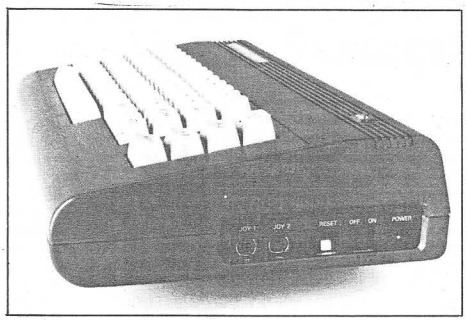
## TIPS & TRICKS













## SERVICE





## LISTINGS

MASCHINENSPRACHE

# Black Beauty: In den „Kleinen" steckt mehr, als Commodore eingebaut hat

**Mehr als eine bloße Übersicht über Befehlssatz, Adressierungsarten und Flagbeeinflussungen bietet diese Einführung. Im Vordergrund stehen Übung und Praxis. Dafür sorgt das Programm CPU-TRAINER, mit dem CPU-Befehle direkt eingegeben und auf ihre Wirkung beobachtet werden können.**

Viele Probleme lassen sich in BASIC nur unzureichend lösen. Oft ist BASIC zu langsam oder total ungeeignet. Der direkte Weg, der Griff zur Maschinensprache, ist vielen noch versperrt, da die nötigen Informationen fehlen. Sollte dem einen oder anderen der Befehlssatz der CPU-7501 oder 6502 bereits bekannt sein, so hilft dieses herzlich wenig, wenn er das Betriebssystem des Rechners nicht kennt.
Ohne die Kenntnis wichtiger I/O-Adressen und -Routinen läßt sich nicht einmal ein einziges Zeichen von der Tastatur einlesen oder auf den Bildschirm bringen. Wir besprechen daher auch die wichtigsten Kernel-Routinen, das Port-Register und einige interessante Register des TED-Chips, der für Bild und Ton zuständig ist.
Sie lernen, wie ein Ein- oder Ausgabekanal definiert werden kann, wie Sie Zeichen auf den Drucker senden, wie Sie etwas auf Kassette oder Diskette ausgeben können, und wie Sie sie auch wieder von dort einlesen. Wenn die CPU auch keine Multiplikation kennt, mit einem bestimmten Algorithmus läßt sie sich verwirklichen. Mit dem TEDMON, dem eingebauten Maschinenmonitor, können wir kleine Programme schreiben. Vom CPU-TRAINER aus können wir sie aufrufen und deren Ergebnisse kontrollieren.
Wie Sie sehen, ist der Rahmen unserer Einführung sehr weit gespannt. Wenn Sie diese gewissenhaft durcharbeiten, sind Sie bereits in der Lage, selbst kleine Maschinenprogramme zu verfassen. In den folgenden C16-P4-SPECIAL-Heften werden wir Ihnen, da Sie jetzt die Grundlagen dafür bekommen, erstaunliche Tips und Tricks zeigen, mit denen Sie das Betriebssystem des Rechners nach Ihrem Gutdünken manipulieren können. Ein Druckerpuffer, eine interruptgesteuerte Uhr oder ein nichtblinkender Cursor sind dann nicht mehr schwer zu programmieren.

## I. DER CPU-TRAINER

**Dieses Programm wird uns durch unseren ganzen Einführungskurs begleiten. Es ermöglicht die Direkteingabe von CPU-Befehlen.**

Das Programm besitzt einen Maschinensprache- und einen BASIC-Teil. In *Zeile 100* wird das BASIC-Ende auf die Adresse $3000 herabgesetzt. Ab dieser Adresse können wir uns im Hauptspeicher unseres Rechners austoben. Der in den *DATA-Zeilen 130 bis 320* abgelegte Maschinenteil wird durch die *Zeilen 110 und 120* im Bereich von $065E bis $06D2 abgelegt. Er enthält am Anfang die Routine zur Übernahme der Registerinhalte in die CPU, zur Ausführung des CPU-Befehles und zum Zurückschreiben der CPU-Register in die dafür vorgesehenen Speicherstellen.
Die restlichen Routinen sorgen für die Sonderbehandlung der in *Zeile 1260* vermerkten Befehle. In *Zeile 330* werden die für die Registerinhalte reservierten Speicherstellen mit den Inhalten der CPU-Register oder mit dem Wert 127 initialisiert.
Ab *Zeile 370* beginnt der eigentliche BASIC-Teil. Es werden Variablen, die für die Bildschirmausgabe wichtig sind, die entsprechenden Inhalte zugewiesen. Interessant sind hier die *Zeile 480 und 490*, durch die die Inhalte der Variablen A$(0) bis A$(6) in *Zeile 380 bis 440* umgewandelt werden.
Die auf jedem Drucker darstellbaren Zeichen 0 und a bis j werden durch die in den *DATA-Zeilen 450 und 460* vermerkten Codes ersetzt, die nachher auf dem Bildschirm erscheinen sollen.
Der Bildschirmaufbau – *Zeile 570 bis 630* – begnügt sich mit wenig Programmzeilen, da geschickt auf die Bildschirmvariablen und Unterprogramme zurückgegriffen wird.
Die Eingabe in den *Zeilen 680 bis 760* wurde nicht durch einen INPUT-Befehl realisiert.
Auffallend ist, daß auch kein GET- oder PRINT-Befehl im Programm vorkommt. Anstelle der Befehle, die auf ein definiertes Ein- oder Ausgabegerät zugreifen, oder dieses umdefinieren, verwendeten wir neben dem CHAR-Befehl die SYS-Aufrufe SYS60381 und SYS56393. SYS 60381 ersetzt den GET-Befehl. Das Zeichen wird unabhängig vom aktuellen Eingabegerät immer von der Tastatur eingelesen. Das abgefragte Zeichen wird von der Systemroutine im Akku abgelegt.
Nach dem SYS-Aufruf befindet es sich daher in Speicherstelle 2034, aus der wir es mit einem PEEK hervorholen können. Statt PRINT verwendeten wir SYS56393. Es kann nur ein einziges Zeichen ausgegeben werden. Dessen ASCII-Wert ist vorher mit einem POKE in die Speicherstelle 2034 zu bringen.
Der SYS-Aufruf lädt den Akku mit diesem Wert, die Systemroutine gibt das Zeichen unabhängig vom aktuellen Eingabegerät immer auf den Bildschirm aus. So wurde dafür Sorge getragen, daß auch bei umdefinierten Ein- und Ausgabegeräten unsere Bildschirmausgaben immer auf den Bildschirm gelangen und auch bei umdefiniertem Eingabegerät die Abfrage unserer Eingabe funktioniert.
Nach einer Eingabe definieren wir uns in den *Zeilen 830 und 840* ein leeres Fenster. Bildschirmausgaben, die wir durch ein Maschinenprogramm veranlassen, sollen die Darstellung der CPU-Register nicht beeinflussen. Beim Sprung in den Monitor mit BRK soll uns der ganze Bildschirm zur Verfügung stehen *(Zeile 820)*.

Nach der Befehlsabarbeitung sorgte *Zeile 680* wieder für den neuen Bildschirmaufbau. In *Zeile 850* wird durch den GOSUB-Befehl die eigentliche Abarbeitung veranlaßt. In *Zeile 1180* findet eine Verzweigung statt, wenn es sich nicht um einen Ein-Byte-Befehl handelt. Die Unterroutine in *Zeile 1050* untersucht, ob ein existierender Befehl vorliegt, und übergibt in diesem Falle einen weiter zu verarbeitenden Wert.
Dieses ist im Normalfall der Operationscode für die CPU oder bei der Sonderbehandlung in *Zeile 1240 bis 1260* die Adresse der anzuspringenden Routine. In *Zeile 1200* wird der Operationscode an die richtige Stelle in das Maschinenroutine eingefügt, und nachher die Routine mit SYS1630 aufgerufen. Da für den Operationscode drei Byte reserviert sind, werden die verbleibenden zwei Stellen mit dem NOP-Code gefüllt, der die CPU zu keiner Operation veranlaßt.
In *Zeile 1380* bei den Zwei-Byte-Befehlen folgt dem Operationscode ein Datenparameter, daher ist hier nur noch ein NOP-Code erforderlich. In den *Zeilen 1550 bis 1570* bei den Drei-Byte-Codes folgen dem Operationscode zwei Adreß-Byte.

## II. DIE ERSTEN SCHRITTE

**Die Datenübergabe vom Speicher zu den Registern, von den Registern zum Speicher, dem Datentransfer zwischen den Registern, und das Retten von Registerinhalten auf den Stapel bekommen wir mit einigen wenigen Befehlen in den Griff.**
Fünf der sechs CPU-Register sind auf dem Bildschirm dargestellt. Die Inhalte können wir in binärer, hexadezimaler und dezimaler Form betrachten. Ein Fragezeichen signalisiert, daß das Programm offensichtlich auf eine Eingabe wartet. Wenn wir jetzt CPU-Befehle in mnemonischer Form eingeben, können wir beobachten, wie die CPU-Register auf unsere Befehle reagieren. Machen wir uns also mit der Materie vertraut und starten unsere ersten Versuche.

### Laden und Speichern

Wir geben ein:

```
LDA #$01
LDY #$02
LDY #$03
```

Wie wir wohl vermutet haben, zeigen unsere ersten drei Register die Werte eins, zwei und drei. Die Adressierungsart, die durch #$ gekennzeichnet wird, und die den darauffolgenden Wert in das Register lädt, heißt *unmittelbare Adressierung.*
Es existieren auch Befehle, die für sich allein stehen können, ohne daß noch irgendwelche Daten oder sonstige Parameter zu folgen brauchen. Diese Adressierungsart heißt *implizite Adressierung.* Ein sehr interessanter Befehl ist der Break, der mit BRK abgekürzt wird. Geben Sie doch einmal ein:

```
BRK
```

Das Bild auf dem Bildschirm hat sich auf einmal total verändert. Wir befinden uns jetzt nicht mehr in unserem BASIC-Programm, sondern im Maschinen-Monitor TEDMON, der in unseren Rechner von Haus aus eingebaut ist. Der BRK-Befehl löst einen Software-Interrupt aus, der beim C16/116/Plus4 und auch beim C128 zum Aufruf des TED-

### SPRUNG IN DEN MONITOR

MON führt. Daß nichts mehr von unserem früheren Bildschirminhalt zu sehen ist, dafür kann der TEDMON nichts. Das CPU-Programm war so frei, für einen sauberen Bildschirm zu sorgen. Auch der Maschinen-Monitor zeigt uns die Registerinhalte, wenn auch nur in hexadezimaler Form allein.
Ein Unterschied fällt sofort ins Auge. Gleich an erster Stelle findet sich der Programmzähler, der mit PC für *Programm Counter* abgekürzt ist. Dieses Register ist nicht nur ein Byte, sondern zwei Byte breit. Schließlich muß es alle Adressen von $0000 bis $FFFF erreichen können. Zur Zeit zeigt es auf eine Stelle im Maschinenteil unseres Programmes, wo es anschließend weitergeht, wenn wir mit einem Go dafür sorgen. Wir geben ein:

```
G
```

Nach dem Druck der RETURN-Taste zeigt der Bildschirm wieder das vertraute Bild mit den CPU-Registern. Sehr kurz war der Blick in den TEDMON. Vielleicht ist Ihnen bereits noch eine zweite kleine Abweichung aufgefallen. Prägen Sie sich doch einmal den Inhalt des Stapelzeigers ein und gehen dann wieder mit BRK in den Monitor.
Der Stackpointer SP weist einen ganz anderen Wert auf. Das CPU-Programm simuliert einen Stapelzeiger mit anderem Wert, damit beim Herumprobieren ein Programmabsturz nicht so leicht verursacht werden kann.
Ein sehr wichtiger Bestandteil des Rechners neben der CPU ist der Speicher. Mit dem Monitor können Sie sehr leicht Speicherbereiche betrachten. Wir geben ein:

```
M3000
```

Von der Adresse $3000 ausgehend, wird ein Speicherabschnitt in hexadezimaler Form dargestellt. Rechts finden wir invers die ASCII-Darstellung davon. Da ein Rechner auch Texte speichern soll, ist es interessant, zu sehen, wo ein Text abgelegt ist.
Das BASIC-Ende wurde vom CPU-Programm auf $3000 gelegt. Daher steht uns ab dieser Adresse der Hauptspeicher des Rechners fast uneingeschränkt zur Verfügung. Wir sorgen zunächst einmal für einen sauberen Speicher mit

```
F300,3FFF,0
```

Der Bereich von $3000 bis $3FFF ist nun vollständig mit Nullen gefüllt. Wir sehen uns nun wieder den ersten Abschnitt an $3000 mit

```
M3000
```

Wenn nun nach den folgenden Manipulationen andere Werte hier drinstehen sollten, kann keiner mehr sagen, dies wäre schon vorher der Fall gewesen. Wir können nun mit *G* getrost den Monitor wieder verlassen.
Mit LDA, LDX und LDY nur unmittelbar Werte in die Register zu schieben, ist uns zu wenig, denn die Daten, um die es geht, befinden sich in der Regel irgendwo im Hauptspeicher des Rechners. Daten können nur durch die CPU verarbeitet werden. Daher muß es auch Befehle geben, die die Daten vom Speicher in CPU und wieder von der CPU in den Rechner befördern.
Beginnen wir mit dem Weg von der CPU in den Speicher. Uns stehen die Speicher- oder auch die Store-Befehle STA, STX und STY zur Verfügung. Diesen Speicherbefehlen muß eine Adreßangabe folgen, die angibt, wo im Hauptspeicher die Registerinhalte gespeichert werden sollen. Wir geben ein:

```
LDA #$54
STA $3000
LDA #$45
STA $3001
LDA #$53
STA $3002
LDA #$54
STA $3003
```

Diese Adressierungsart, bei der dem Operationscode die Speicheradresse

folgt, heißt *absolute Adressierung*. Gekennzeichnet wird sie durch das $-Zeichen, gefolgt von einer vierstelligen HEX-Zahl. Unsere CPU kennt auch noch weitere Adressierungsarten, bei denen nur ein einziges Byte in Form von zwei Hexziffern anzugeben ist, oder solche, bei denen das X- oder Y-Register mit zur Adreßbildung verwandt wird. Um das Programm nicht allzusehr aufzublasen, wurde darin auf weitere Adressierungsarten jedoch verzichtet.
Um den Erfolg unseres Speichertests zu begutachten, sollten Sie sich im Monitor nochmals den Bereich ab $3000 ansehen. Wie dieses vor sich geht, dürfte Ihnen inzwischen bereits bekannt sein. Wir finden die erwarteten Werte, die in ASCII-Darstellung das Wort TEST ergeben.
Nicht nur die Speicherbefehle, sondern auch die Ladebefehle lassen sich absolut adressieren. Spaßeshalber wechseln wir einmal das Register:

```
LDX $3000
STX $3008
LDX $3001
STX $3009
LDX $3002
STX $300A
LDX $3003
STX $300B
```

Wir haben damit die von $3000 bis $3003 abgelegten Werte nach $3008 bis $300B übertragen, wie uns die Überprüfung mit dem Monitor bestätigt.

### Transferieren zwischen Registern

Datenübertragung von CPU-Register nach Speicher und umgekehrt kennen wir jetzt schon. Irgend etwas scheint aber noch zu fehlen. Wenn wir einen Wert verändern, so kommt es oft vor, daß wir gerne den ursprünglichen Zustand wiederherstellen würden. Im Akku könnte zum Beispiel ein Wert stehen, den wir für irgendwelche Zwecke benötigen. Vorher sollen aber noch ein paar Rechenoperationen erfolgen, die leider den Akkuinhalt verändern.
Eine Lösungsmöglichkeit ist mit unserem bisherigen Wissen bereits denkbar. So könnten wir den Akkuinhalt einfach an irgendeiner Adresse abspeichern und unseren Wert, sobald er wieder gebraucht wird, erneut einladen. Doch warum sollen Daten, die wir bereits in der CPU bereit haben, erst wieder in das RAM geschrieben und dann wieder hergeholt werden?
Speicherzugriffe kosten mehr Zeit als reine CPU-Operationen, und außerdem mehr Platz, da immer noch eine Speicheradresse anzugeben ist. Es sollte doch auch Möglichkeiten geben, Daten direkt von einem Register in das andere zu transferieren. Diese Transferbefehle gibt es selbstverständlich. Um Daten vom Akku in die Intexregister X und Y zu schaufeln, gibt es die Befehle:

```
TAX
TAY
```

Den umgekehrten Weg ermöglichen:

```
TXA
TYA
```

Scheuen Sie sich nicht, dies ruhig auszuprobieren. Solange nur ein oder zwei Werte gerettet zu werden brauchen, und die anderen Register nicht für andere Zwecke benötigt werden, ist dieses Verfahren ein brauchbarer Weg.
Oft werden aber Unterroutinen angesprungen, die alle Registerinhalte verändern. In diesen Unterroutinen kann wiederum der Bedarf bestehen, neue Registerinhalte zu retten. Als Ausweg erscheint in diesem Falle nur das Abspeichern in dafür vorgesehenen Speicherplätzen gangbar zu sein. Jede Unterroutine bräuchte also wieder ihre besonderen Speicherstellen zum Retten der Register.
Nicht leicht ist es wohl, da noch den Überblick zu behalten. Für rekursive Programmierung, bei der ein Unterprogramm sich selbst mehrmals aufruft, bringt auch dieses keine Lösung. Die vorgesehenen Speicherstellen würden bei jedem rekursiven Aufruf neu überschrieben werden, wodurch die früheren Werte verlorengehen. Das Stapelregister sorgt hier für Abhilfe.

### Der Prozessorstapel

Laden Sie einmal einen beliebigen Wert in den Akku. Um diesen zu retten, geben wir jetzt den Befehl ein:

```
PHA
```

Wenn Sie dabei auf das Stapelregister geachtet haben, haben Sie sicherlich eine Veränderung wahrgenommen. Wenn nicht, so beobachten wir es bei einem erneuten Versuch. Laden Sie bitte den Akku mit einem neuen Wert und retten Sie diesen wieder mit PHA.
Der Wert des Stapelzeigers wurde jedesmal um den Betrag eins verringert. Wenn wir unsere geretteten Werte vom Stapel wieder abheben, so beobachten wir umgekehrt ein Erhöhen des Stapelzeigers. Vorher laden wir noch schnell den Akku mit einem anderen Wert, damit wir das Erscheinen der alten Werte sehen. Wir geben ein:

```
PLA
```

Es erscheint der zuletzt gerettete Wert. Nach einem erneuten PLA haben wir auch den zuerst geretteten Wert wieder. Der Stapelzeiger müßte nun wieder, wie am Anfang, den Wert 7F aufweisen. Um zu untersuchen, was geschehen ist, geben wir ein:

```
LDX $017F
LDY $017E
```

Wir sehen, vielleicht auch zu unserer Verblüffung, die ehemals geretteten Werte nun in den Indexregistern stehen.
Der Stapel funktioniert folgendermaßen: Für den Stapel ist ein RAM-Bereich von $0100 bis $01FF reserviert. Der Inhalt des Stapelzeigers gibt das Low-Byte der Adresse an, in die der Registerinhalt bei einem PUSH-Befehl geschrieben wird.
Der Stapelzeiger wird zusätzlich um den Wert eins erniedrigt, so daß er nun auf die nächstniedrige Adresse weist. So können verschiedene Daten hintereinander gerettet werden, ohne daß sie sich gegenseitig ins Gehege kommen.
Beim PULL-Befehl wird der Stapelzeiger wieder um Eins erhöht, und der unter der Adresse mit High-Byte 01 und dem Stapelzeigerinhalt als Low-Byte in das entsprechende Register geladen. Nicht nur der Akku kann „gestapelt" werden, sondern auch das Statusregister. Die Befehle hierfür sind:

```
PHP
PLP
```

Wir können somit alle Registerinhalte auf den Stapel retten:

```
PHP
PHA
TXA
PHA
TYA
PHA
```

Abheben müssen wir den Stapel in umgekehrter Reihenfolge:

```
PLA
TAY
PLA
TAX
PLA
PLP
```

Die Stapelbefehle erlauben uns gar, mit dem Statusregisterinhalt genauso, wie mit sonstigen Daten umzugehen. Wir bekommen die Werte in den Griff mit:

```
PHP
PLA
```

Das Zurückübertragen macht auch keine besonderen Schwierigkeiten:

```
PHA
PLP
```

Sogar das Stapelregister bekommen wir mit zwei Transferbefehlen völlig in den Griff:

```
TSX
TXS
```

**Übersicht der besprochenen Befehle**

| | | | |
|---|---|---|---|
| Laden | LDA, | LDX, | LDY |
| Speichern | STA, | STX, | STY |
| Transferieren | TAX | TAY, | TSX |
| | TXA, | TYA, | TXS |
| Stapeln | PHA, | PHP | |
| | PLA, | PLP | |

## III. SCHALTEN MIT DER CPU

**Einzelne Bit lassen sich mit den Befehlen ORA, AND und EOR setzen. Damit können wir verschiedene Umschaltungen in den TED-Registern vornehmen. Den Bildschirm auf 24 Zeilen oder auf 38 Spalten umzuschalten, ist relativ einfach zu handhaben.**

Mit dem Computer lassen sich nicht nur Daten verwalten. Besondere Bausteine, die wir über bestimmte Adressen ansprechen können, erlauben uns, die verschiedensten Umschaltungen vorzunehmen. Oft ist es nur ein einziges Bit, das geändert werden soll. Die Änderung der anderen Bit könnte zu unerwünschten Ergebnissen führen.
Interessant sind vor allen Dingen die Multifunktionsregister des TED-Chips und der serielle Port unseres Rechners. Bevor wir uns mit diesen Bausteinen ausgiebig befassen, sind zuerst die Befehle zur Bit-Manipulation zu besprechen. Wir geben ein:

```
LDA #$00
LDA #$01
LDA #$02
LDA #$04
LDA #$08
LDA #$10
LDA #$20
LDA #$40
LDA #$80
```

Ein bestimmtes Bit läßt sich mit dem Ladebefehl leicht setzen. Der Nachteil ist, daß die bereits vorhandenen Bit leider auch verändert werden. Zum Verändern einzelner Bit eignet sich der Befehl EOR. Probieren Sie doch einmal:

```
LDA #$99
EOR #$01
EOR #$02
EOR #$04
EOR #$08
EOR #$10
EOR #$20
EOR #$40
EOR #$80
```

Wir haben ein Bit nach dem anderen umgedreht. Es lassen sich natürlich auch alle umdrehen mit:

```
EOR #$FF
```

Für uns könnte der EOR-Befehl bereits ausreichend sein, da wir ja sehen, ob ein bestimmtes Bit bereits den gewünschten Wert hat, oder nicht. Wenn nein, wenden wir den EOR-Befehl an, wenn ja, dann eben nicht. Die CPU nachsehen zu lassen, ob ein Bit gesetzt ist, oder nicht und gegebenenfalls mit dem EOR-Befehl zu reagieren, wäre wohl etwas aufwendig. Daher existieren die Befehle ORA und AND, mit denen unabhängig vom Bit-Zustand diesem in jedem Falle der Wert eins oder null zugewiesen werden kann:

```
LDA #$99
ORA #$01
ORA #$02
ORA #$04
ORA #$08
ORA #$10
ORA #$20
ORA #$40
ORA #$80
```

Ebenso wie beim EOR-Befehl lassen sich auch mehrere Bit setzen:

```
LDA #$81
ORA #$18
```

Die zwei mittleren Bit sind durch den ORA-Befehl hinzugekommen. Um zu wissen, welche Werte wir für den AND-Befehl brauchen, sollten wir die Werte, mit denen wir vorher geordert hatten, invertieren, und uns die Resultate merken:

```
LDA #$01
EOR #$FF
LDA #$02
EOR #$FF
LDA #$04
EOR #$FF
LDA #$08
EOR #$FF
LDA #$10
EOR #$FF
LDA #$20
EOR #$FF
LDA #$40
EOR #$FF
LDA #$80
EOR #$FF
```

Wir können nun den Akku wieder mit #$99 oder auch einem anderen Wert laden, und das Nullsetzen der Bit beobachten.

```
LDA #$99
AND #$FE
AND #$FD
AND #$FB
AND #$F7
AND #$EF
AND #$DF
AND #$BF
AND #$7F
```

Es lassen sich wieder mehrere Bit auf einmal setzen. Wir laden hierzu das X-Register.

```
LDX #$7E
```

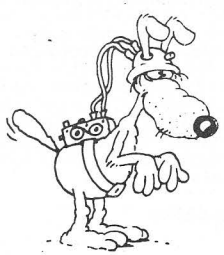
Wir können sehen, daß die äußeren beiden Bit den Wert Null haben. Beim Undieren mit diesem Wert würden folglich die äußeren Bit auf Null gesetzt.

```
LDA #$99
AND #$7E
```

Probieren Sie ruhig noch ein wenig mit den Befehlen EOR, ORA und AND herum, denn Übung macht ja bekanntlich den Meister. Sie können auch die absolute Adressierung verwenden, bei welcher der Inhalt einer Speicherzelle mit dem Akkuinhalt verknüpft wird.

### Die Register des TED-Chips

Die Register des TED-Chips werden angesprochen über die Adressen $FF00 bis $FF3F. Hier ist eine kleine Zusammenstellung:



| | |
|---|---|
| $FF00 | Low-Byte Timer A |
| $FF01 | High-Byte Timer A |
| $FF02 | Low-Byte Timer B |
| $FF03 | High-Byte Timer B |
| $FF04 | Low-Byte Timer C |
| $FF05 | High-Byte Timer C |
| $FF06 | Multi-Funktions-Register I |
| $FF07 | Multi-Funktions-Register II |
| $FF08 | Tastaturregister |
| $FF09 | Interrupt Request Register |
| $FF0A | Interrupt Mask Register |
| $FF0B | Rasterinterrupt |
| $FF0C | Cursorpositon High-Byte |
| $FF0D | Cursorpositon Low-Byte |
| $FF0E | Tonhöhe Tongenerator 1 |
| $FF0F | Tonhöhe Tongenerator 2 |
| $FF10 | Tonhöhe Tongenerator 2 Bit 8/9) |
| $FF11 | Multi-Funktions-register III |
| $FF12 | Multi-Funktions-register IV |
| $FF13 | Zeichensatz-Adresse |
| $FF14 | Basisadresse Farb-/ Video-RAM |
| $FF15 | Hintergrundfarbe |
| $FF16 | Zeichenfarbe |
| $FF17 | Multicolor-Farbe 1 |
| $FF18 | Multicolor-Farbe 2 |
| $FF19 | Rahmenfarbe |
| $FF1A | Bit-Mapping 1 |
| $FF1B | Bit-Mapping 2 |
| $FF1C | Aktuelle Rasterzeile Bit 8 |
| $FF1D | Aktuelle Rasterzeile Bit 0–7 |
| $FF1E | Rasterspalte |
| $FF1F | Vertikal-Adresse |
| $FF3E | Umschaltung auf ROM |
| $FF3F | Umschaltung auf RAM |

Der TED-Chip besitzt eine ziemliche Anzahl von Registern. Viele davon sind erst bei fortgeschrittenen Systemkenntnissen interessant. Wir picken uns einige davon heraus, an denen wir sehr schön einige herausstechende Effekte beobachten können. Gut eignen sich hier die vier Multi-Funktions-Register, die Zeichensatzadresse, Hintergrundfarbe, Zeichenfarbe und Rahmenfarbe. Zur Sicherung wird der Inhalt von $FF06 zusätzlich in das X-Register geladen.

```
LDA $FF06
TAX
```

Wir verändern die Bit null bis zwei, die angeben, um wieviel Pixel der Bildschirm vertikal verschoben werden soll.

```
ORA #$07
STA $FF06
AND #$F8
STA $FF06
STX $FF06
```

Beim Verschieben des Bildschirmes treten am oberen und unteren Rand unerwünschte Effekte auf. Bit drei, das den Bildschirm auf 24 Zeilen umschaltet, schafft vielleicht Abhilfe.

```
TXA
AND #$F7
STA $FF06
```

Nicht wie erwartet, verschwindet die letzte Zeile, sondern sowohl ein Teil der ersten und der letzten. Dieses müßte sich jedoch durch Bildschirmverschieben wohl wieder ausgleichen lassen.

```
ORA #$07
STA $FF06
AND #$F8
STA $FF06
```

Bei 24 Zeilen ist es möglich, den Bildschirm pixelweise ohne störende Nebeneffekte zu verschieben, bis schließlich die Breite einer ganzen Zeile erreicht wird. So können Smooth-Scrolling-Effekte programmiert werden. Wir stellen wieder den Anfangszustand her:

```
STX $FF06
TXA
```

Erschrecken Sie bitte beim nächsten Bit nicht, wenn plötzlich auf dem Bildschirm nichts mehr zu sehen sein sollte. Die Tastatur nimmt weiter Ihre Eingaben an.

```
EOR #$10
STA $FF06
STX $FF06
TXA
```

Sollen Programme etwas schneller laufen, oder Routinen völlig gleichmäßig ausgeführt werden, so ist es zweckmäßig, den Bildschirm auszuschalten. Die CPU wird so nicht mehr durch den TED-Chip gestoppt, damit dieser in der Zwischenzeit den Bildschirm aufbauen kann. Dies ist besonders für Signale wichtig, die nicht durch eine extra Clock-Leitung getaktet werden, sondern bei denen es einzig und allein auf die Impulsdauer ankommt. Bei der Datenübertragung zur oder von Datasette haben Sie diesen Effekt sicherlich schon beobachtet.
Einen Bildschirm, der nicht sehr viel herzeigt, bekommen Sie auch mit

```
EOR #$20
STA $FF06
STX $FF06
TXA
```

**Multi-Funktions-Register I ($FF06)**

| | |
|---|---|
| Bit 0–2: | Vertikale Verschiebung des Bildschirms |
| Bit 3: | Umschaltung auf 24 Zeilen |
| Bit 4: | Bildschirm aus/ein |
| Bit 5: | Grafikmodus aus/ein |
| Bit 6: | Grafikzeichen werden als reverse Buchstaben wiedergegeben |
| Bit 7: | Funktion unbekannt |

**Multi-Funktions-Register II ($FF07)**

| | |
|---|---|
| Bit 0–2: | Horizontale Bildschirmverschiebung |
| Bit 3: | Umschaltung auf 38 Spalten. Schaltet unerwünschte Effekte beim Smooth-Scrolling aus. |
| Bit 4: | Multicolor enable |
| Bit 5: | Funktion unbekannt |
| Bit 6: | PAL/NTSC-Norm |
| Bit 7: | Invert off. Anstelle inverser Zeichen können Zeichen aus dem zweiten Zeichensatz mit dargestellt werden. Damit lassen sich gleichzeitig der Groß/ Grafik- und der Klein/Groß-Zeichensatz darstellen. |

**Multi-Funktions-Register III ($FF11)**

| | |
|---|---|
| Bit 0–3: | Lautstärke (0–8) |
| Bit 4: | Tongenerator 1 ein/aus |
| Bit 5: | Tongenerator 2 ein/aus |
| Bit 6: | Tongenerator 2 Rauschen |
| Bit 7: | Tongenerator ein/aus |

| Multi-Funktions-Register IV ($FF12) | |
|---|---|
| Bit 0/1: | Tonhöhe Tongenerator 1, Bit (8/9) |
| Bit 2: | Zeichensatz in RAM/ROM |
| Bit 3–7: | Basisadresse der hochauflösenden Grafik |

Von Grafik ist leider keine Spur zu sehen, auch wenn vorher einmal der Grafikbildschirm aktiviert war, zeigt sich noch nichts von einer Grafik. Diese Bit-Umstellung hat nur eine Wirkung, wenn Sie zu ganz bestimmten Zeiten mit einer Vielzahl anderer Parameter zusammen erfolgt. Die Interruptroutine, die auch für die Bildschirmausgabe verantwortlich ist, macht beim Bearbeiten der Grafik von diesem Bit Gebrauch.
Fraglich ist, ob das nächste Bit uns sehr viel Nutzen bringen kann.

```
EOR #$40
STA $FF06
STX $FF06
TXA
```

Für irgendwelche schleierhaften Testzwecke soll angeblich Bit sieben dienen.

```
EOR #$80
STA $FF06
STX $FF06
TXA
```

Mit den Bit-Manipulationen dürften Sie nun schon ein wenig vertraut sein. So wird es wohl reichen, wenn wir ein paar weitere TED-Register ohne zusätzliche Experimentieranweisungen vorstellen.

### Zeichensatzadresse ($FF13)

Die Bit zwei bis sieben enthalten die Adreß-Bit 10 bis 15 der Zeichengeneratoradresse. Wir starten einen Versuch.

```
LDX $FF13
```

Nun schalten wir mit Commodore-Taste und Shift den Zeichensatz um.

```
LDY $FF13
```

Der Wert der Zeichensatzadresse hat sich offensichtlich verändert. Wir schalten wieder um. Durch das Speichern der Werte in den Indexregistern ist auch ein Umschalten zwischen Groß- und Kleinschrift möglich.

```
STY $FF13
STX $FF13
```

### Zeichenfarbe ($FF16)

Manipulationen dieses Registers bleiben ohne Wirkung. Die Zeichenfarbe bestimmt schließlich der im Farb-RAM abgelegte Code. Welcher Code abgelegt wird, bestimmt die Speicherstelle $053B. Wir laden diesen Wert.

```
LDA $053B
TAX
```

Bit null bis drei, die vier rechten Bit zeigen eine Null als Kennung für die Farbe schwarz. Bit vier bis sechs geben die Helligkeit an. Schwarz ist schwarz, deshalb ist die Helligkeit uninteressant. Die Frage ist, ob Bit sieben auch noch eine Funktion besitzt.

```
ORA #$80
STA $053B
STX $053B
TXA
```

Bit acht ist das sogenannte Blinkflag. Die neu ausgegebenen Registerwerte und unsere Eingabe blinkten fröhlich vor sich hin. Wenn Sie gerne Farb- und Helligkeitsänderungen vornehmen wollen, so dürfen Sie dieses gerne tun.

### Rahmenfarbe ($FF19)

Mit diesem Register kann die Rahmenfarbe geändert werden. Dieses geht genauso wie bei der Zeichenfarbe. Bit sieben ist ohne Funktion. ein Blinken von Rahmen oder Hintergrund gibt es nicht.

### Hintergrundfarbe ($FF15)

Es gilt das bereits in Verbindung mit der Rahmenfarbe Gesagte.

## IV. SCHALTEN MIT DEM CPU-PORT ($0001)

**Der C16/116/Plus4 besitzt als Prozessor keine CPU 6502, sondern die CPU 7501. Diese CPU zeichnet sich durch einen speziellen Port aus, den der Rechner zum Ansteuern von Kassettenport und seriellen Port benützt.**

Wir sehen uns die Bit des Port-Registers an:

```
LDA $0001
TAX
```

Bit null bis drei, die vier rechten Bit, dienen zum Senden, Bit vier bis sieben, die vier linken Bit, zum Empfangen. Die Sendeleitungen laufen über einen Inverter. Wir sehen, daß Spannung auf den zu Bit null bis zwei dazugehörigen Leitungen liegt. Die zu Bit drei gehörige Leitung liegt auf Masse. Wir nehmen einige Manipulationen vor.

```
ORA #$01
STA $0001
LDA $0001
```

Der Zustand von Bit null hat sich geändert, allerdings ist mit Bit sieben das Gleiche passiert. Wenn wir Bit eins abändern, so können wir denselben Effekt auch bei Bit sechs beobachten. Bit sieben scheint irgendwie mit Bit null und Bit sechs mit Bit eins gekoppelt zu sein. Dieses ist auch der Fall. Mit Bit null schalten wir die DATA-Leitung, mit Bit eins die CLOCK-Leitung und mit Bit zwei die ATN-Leitung des seriellen Ports. Wer einen Spannungsmesser besitzt, kann dieses nachmessen.
Mit Bit sieben messen wir die Spannung auf der DATA-Leitung und mit Bit sechs die Spannung an der CLOCK-Leitung. Die ATN-Leitung können wir leider nicht abfragen. Hat Bit eins den Wert null, so liegt Spannung auf der DATA-Leitung, was Bit sechs mit einer Eins anzeigt. Im umgekehrten Fall zeigt Bit sechs keine Spannung an.
Wenn uns ein zweiter C16/116/Plus4 zur Verfügung steht, und wir ein serielles Kabel besitzen, können wir gar Signale von einem Rechner auf den anderen übertragen. Wir müssen hierzu im empfangenden Rechner das Sende-Bit auf Null setzen, so daß auf der entsprechenden Leitung fünf Volt anliegen. Wenn der Senderechner die Leitung des Sendebit auf eins setzt, fällt das Potential ab und der Empfangsrechner registriert dieses durch das Empfangs-Bit.
Serielle Übertragungen von Rechner zu Rechner sind so ohne Weiteres machbar. Bit eins dient nicht nur als Clock-Signal für den seriellen Port, sondern auch zur Ansteuerung der Write-Leitung des Kassettenports, was mit einem Voltmeter überprüft werden kann. Der Kassettenport stellt uns allerdings nur drei Volt zur Verfügung. Legen wir eine Spannung an die Read-Leitung des Kassettenports an, so sehen wir, daß Bit vier den Zustand dieser Leitung anzeigt.
Ob die Playtaste gedrückt wurde, oder nicht, darüber gibt uns das

Port-Register keinen Aufschluß, jedoch das Bit zwei in Speicherstelle $FD10. Ist dieses Null, so ist die Play-Taste gedrückt. Der Plus4 erlaubt uns über dieses Bit, den Kassettenmotor zu steuern. Da hier ein I/O-Baustein vorliegt, der auch für den Userport gedacht ist, können wir dort einfach etwas hineinschreiben und so eine gedrückte Rekordertaste simulieren.

```
LDA $FD10
EOR #$04
STA $FD10
LDA $0001
```

Der Plus4 zeigt im Port-Register, daß die Motor-Leitung, die durch Bit drei angesteuert wird, eingeschaltet ist. Das Spannungsmeßgerät sollte über sechs Volt anzeigen. Der C16/116 zeigt leider keinerlei Reaktion. Wir stellen beim Plus4 das Play-Flag wieder auf den ursprünglichen Wert

```
LDA $FD10
ORA #$04
STA $FD10
LDA $0001
```

Wir versuchen, die Motorleitung auf Spannung zu setzen.

```
EOR #$08
STA $0001
LDA $0001
```

Leider hatten wir keinen Erfolg. Wir schließen die Datasette an und drücken auf die Playtaste.

```
LDA $0001
```

Wie wir sehen, ist die Motorleitung nun angeschaltet. Wir können bei gedrückter Play-Taste nun nach Belieben schalten und walten.

```
EOR #$08
STA $0001
LDA $0001
```

Der Motor ist wieder aus.

```
EOR #$08
STA $0001
LDA $0001
```

Nun ist er wieder an. Für Schaltungen mit dem Kassettenport kann durch Verbinden der Play-Leitung mit Masse dafür gesorgt werden, daß wir auch die Motor-Leitung zum Schalten benutzen können.
Eine andere Möglichkeit gewinnen wir durch einen kleinen Speichereingriff:

```
LDY #$45
STY $0312
```

Damit ist die Betriebssystemroutine, die für die Kassettenmotoransteuerung bei gedrückter Play-Taste sorgt, ausgeschaltet. Auf einen Druck der Play-Taste allein hin rührt sich noch lange nichts. Der Motor wird nur noch durch Bit drei des Portregisters gesteuert. Auch wenn gar nichts angeschlossen ist, können wir dieses Bit jetzt nach Belieben verändern und die Spannung messen. Wir bringen nach unseren Versuchen unser Betriebssystem wieder in Ordnung.

```
LDY #$42
STY $0312
```

## V. DIE LOGIK DER CPU

**CPU-Operationen beeinflussen bestimmte Flags des Statusregisters. Acht bedingte Sprungbefehle reagieren auf die Zustände von vier Statusflags.**

Viele sprechen von der Intelligenz des Computers. Alles, was eine CPU kann, ist jedoch nur auf ganz gewisse Schalterstellungen hin zu verzweigen. Damit diese Verzweigungen richtig stattfinden, ist die Intelligenz des Programmierers gefordert. Die Verzweigungsbefehle sind:

```
BEQ
BNE
BPL
BMI
BCS
BCC
BVS
BVC
```

Diese Befehle können Sie mit unserem CPU-Programm nicht eingeben. Sie sind nicht für sich, sondern nur in einem Programm brauchbar. Wir können uns jedoch mit den Schaltern befassen. Ein Register wurde bisher noch kaum besprochen, das Statusregister. Es ist sozusagen das Auge des Computers. Nur auf das, was das Statusregister anzeigt, kann die CPU mit Verzweigungen reagieren. Set- und Clear-Befehle erlauben uns die gezielte Einflußnahme auf einzelne Statusflags.

```
SEC
CLC
```

Das Carry-Flag ist das einzige Flag, das wir mit Set und Clear sowohl an- und ausschalten können und auf welches außerdem noch Verzweigungsbefehle reagieren, die Befehle BCS und BCC. Daher wird dieses Flag oft gesetzt, um den erfolgreichen Ablauf einer Operation zu signalisieren.
Zurückgekehrt aus einem Unterprogramm, das zum Beispiel Daten auf ein externes Gerät ausgeben sollte, läßt sich mit dem Branch-Befehl auf das Carry-Flag reagieren, für den Fall, daß die Ausgabe nicht geklappt hätte, da vielleicht der Drucker gar nicht angeschaltet war.

```
SEI
CLI
```

Mit diesen Befehlen können wir das Interrupt-Flag setzen und löschen. Wie wir wissen, kommt es zu Zwekken der Bildschirmausgabe und der Tastaturabfrage immer wieder zu Unterbrechungen des laufenden Programms. Dies kann aus irgendwelchen Gründen unerwünscht sein. Ist gar die RAM-Bank eingeschaltet und der Prozessor bearbeitet hierin ein Programm, so würde der Interrupt einen Absturz bewirken, wird doch durch diesen wieder die ROM-Bank aktiviert. Anstelle des Rücksprungs in die unterbrochene Routine, landet der Programmzähler dann irgendwo im Betriebssystem oder im BASIC-Interpreter, so daß dann nur mehr der Himmel weiß, was anschließend stattfindet.

```
SED
CLD
```

Damit eingegebene Zahlen nicht mühsam erst in Hex-Zahlen umgerechnet werden müssen, wenn wir Rechenoperationen vornehmen wollen, können wir dem Rechner durch Setzen des Dezimal-Flags mitteilen, daß er seine Additionen und Subtraktionen nun im sogenannten BCD-System vornehmen soll.

```
CLV
```

Bei Rechenoperationen mit vorzeichenbehafteten Zahlen ist das Überlauf-Flag von Interesse. Es sollte daher auch wieder zurückgesetzt werden können.

| Zusammenfassung der Set- und Clear-Befehle |
|---|
| SEC |
| CLC |
| SEI |
| CLI |
| SED |
| CLI |
| CLV |

### Das Zero-Flag

Wissen Sie, warum viele Programmierer das Ende eines Datensatzes mit einer Null kennzeichnen?
Laden Sie den Akku oder irgend ein anderes Register mit einer von Null verschiedenen Zahl, dann mit der Zahl null. Beim Beobachten des Statusregisters bemerken wir eine Veränderung von Bit eins, des zweiten Bit von rechts. Ist der geladene Wert Null, so zeigt das Zero-Flag eine Eins, im anderen Falle eine Null. Mit den Branchbefehlen BEQ und BNE kann daraufhin entsprechend verzweigt werden. Branch Equal ist gleichbedeutend mit Branch Zero und heißt soviel wie: „Verzweige, wenn Zero-Flag gleich eins".
Branch Not Equal verzweigt im umgekehrten Falle.

### Das Vorzeichenflag

Datensätze in Standard-ASCII, wobei nur Werte bis #$7F vorkommen, lassen sich sogar noch kürzer abspeichern. Ein extra Byte für die Endemarkierung kann entfallen. Es braucht nur Bit sieben des letzten Zeichens auf Eins gesetzt werden. Das Vorzeichenflag zeigt dieses an.

```
LDA #$01
LDA #$81
```

Bit sieben, das ganz links wiedergegeben wird, ist das Vorzeichenflag. Es heißt so, da beim Rechnen mit negativen Zahlen definiert wird, daß Zahlen von #$00 bis #$7F positiv seien und Zahlen von #$80 bis #$FF negativ. Bit sieben des Statusregisters gibt also Bit sieben eines geladenen oder auf Grund einer sonstigen Operation erhaltenen Wert eines Registers oder Speicherinhaltes wieder.
Bit sieben läßt sich ganz einfach mit dem Vorzeichenflag abfragen. Der Prozessor sollte aber auch die restlichen Bit aus einem Register oder aus einer Speicherzelle abfragen können. Einen Befehl, mit dem wir dieses leicht bewerkstelligen können, kennen wir bereits.

```
LDA #$01
AND $0001
LDA #$02
AND $0001
LDA #$04
AND $0001
LDA #$08
AND $0001
LDA #$10
AND $0001
LDA #$20
```

```
AND $0001
LDA #$40
AND $0001
LDA #$80
AND $0001
```

Wir setzen im Akku alle Bit auf Null, mit Ausnahme desjenigen, das wir abfragen wollen. Durch den AND-Befehl werden alle Bit, die uns nicht interessieren, zu Null. An der Stelle, die wir mit einer Eins maskiert hatten, gelangt das gewünschte Bit aus der Speicherstelle in den Akku. Wenn dieses Null war, so ist der gesamte Akku Null und das Zero-Flag ist gesetzt. Im anderen Falle wissen wir, daß uns eine Eins vorliegt.
Der *AND*-Befehl ändert auch den Akku-Inhalt. Manchmal wäre es wünschenswert, wenn dieser uns erhalten bliebe. Außerdem kann mit dem AND-Befehl nur ein einziges Bit über das Zero-Flag abgefragt werden. Ein paar Möglichkeiten mehr bietet der Bit-Befehl. Um Bit sieben abzufragen, braucht nicht extra der Wert eines Registers durch Hereinladen zerstört werden. Zusätzlich zu Bit sieben im Vorzeichenflag wird Bit sechs im Überlauf-Flag erfaßt. Die restlichen Bit können auf dieselbe Art, wie wir es bereits vom AND-Befehl her kennen, abgefragt werden.
Die Besonderheit des *BIT*-Befehls ist, daß zwar auch eine Undierung stattfindet, jedoch das Ergebnis nur im Setzen des Zero-Flags sichtbar wird, und so der Akkuinhalt unverändert bleibt.

```
LDA #$01
BIT $0001
LDA #$02
BIT $0001
LDA #$04
BIT $0001
LDA #$08
BIT $0001
LDA #$10
BIT $0001
LDA #$20
BIT $0001
```

Wir haben die Abfragen jetzt nur bis Bit fünf ausgeführt, da das Statusregister uns sowieso immer die Werte von Bit sechs und sieben wiedergab. Da das Überlauf-Flag auf Eins steht, gibt uns dieses die Gelegenheit, den Befehl *CLV* auszuprobieren. Es funktioniert wie gewünscht.
Wir können den Wert #$00 abfragen, wir können abfragen, ob ein Wert größer als #$7F ist, sogar vor einzelnen Bit brauchen wir nicht zu kapitulieren. Dennoch fehlt noch irgend etwas. Was tun wir denn, wenn Datensätze nicht mit einer Null abgeschlossen sind und auch nicht mit einem Wert größer als #$7F, sondern schlicht mit einem Carriage-Return oder einem sonstigen Wert?

Wir müßten demnach vergleichen können, ob wir einen bestimmten Wert vor uns haben, oder nicht. Dafür gibt es den *Compare*-Befehl, der zwei Werte miteinander vergleicht. Vergleichen können wir die Registerinhalte unmittelbar mit einem bestimmten Wert, oder aber mit dem Inhalt einer beliebigen Speicherzelle. Die Compare-Befehle sind:

```
CMP
CPX
CPY
```

CMP vergleicht mit dem Akkuinhalt, die beiden anderen Befehle sind wohl nicht schwer zu erraten.

```
LDA #$04
CMP #$03
```

Das Zero-Flag zeigt eine Null.

```
LDA #$04
CMP #$04
```

Das Zero-Flag zeigt eine Eins zum Zeichen, daß die beiden verglichenen Werte übereinstimmen. Einen Vergleich können wir uns so vorstellen, daß vom Wert im Register der verglichene Wert subtrahiert wird, ohne daß der Wert im Register verändert wird. Wie beim Bit-Befehl erscheint das Resultat nur im Statusregister.
Die Namensgebung der Befehle BEQ und BNE ist auf diesen Vergleich zurückzuführen. Stimmen die beiden Werte überein, so ist die Differnz der Werte Null, das Zero-Flag somit gesetzt und BEQ verzweigt entsprechend. *Branch Equal* heißt somit: „Verzweige, wenn Gleichheit besteht."
Von BASIC her kennen wir:

```
IF A=B
IF A<>B
IF A<B
IF A>B
```

Die ersten beiden Fälle sind in Maschinensprache bereits abgehandelt. Die letzten beiden fehlen uns noch. Wir beobachten bei den folgenden Vergleichen einmal das Carry-Flag, ob hier ein Unterschied wahrnehmbar ist.

```
LDA #$04
CMP #$03
CMP #$05
CMP #$04
```

Ist der Registerinhalt größer oder gleich dem zu vergleichenden Wert, so wird das Carry gesetzt, im anderen Falle gelöscht.

Mit den Branch-Befehlen *BCS*, Branch Carry SET, und *BCC*, Branch Carry Clear, können wir nach Wunsch verzweigen. Beim *CMP*-Befehl wird auch noch das Vorzeichenflag beeinflußt. Eine große Bedeutung kommt diesem Umstand wohl nicht zu. Wir können aber leicht noch einmal nachprüfen, ob Bit sieben in einem Regisfer gesetzt ist, wenn zwischendurch auch schon andere Ladeoperationen erfolgt sein sollten.

```
LDA #$90
LDX #$01
CMP #$00
```

Wir besitzen jetzt ein fundiertes Wissen über die Programmlogik unseres Rechners. Ob wir unser Gerät Rechner nennen dürfen, werden uns die Rechenbefehle zeigen.

## VI. DIE RECHENOPERATIONEN DER CPU

**Addition, Subtraktion, Rotations-, und Schiebe-, Ikrementier- und Dekrementierbefehle sind Operationen, mit denen der Prozessor aufwartet. Multiplikationen und höhere Rechenoperationen lassen sich mit diesen Befehlen realisieren.**

Ihr Taschenrechner wartet mit Grundrechenarten, Wurzeln, Potenzen und vielen mathematischen Funktionen auf. BASIC besitzt ebenso eine Menge von Rechenoperationen. Der Rechner muß also eine Menge können. Mancher wird wohl sehr verblüfft sein, wie wenig die CPU tatsächlich nur kann. Sie wartet nicht mit vielstelligen Dezimalzahlen und Exponenten, sondern nur mit Byte-weiser Addition und Subtraktion auf.
Multiplikation und Division sind für sie bereits Fremdwörter. Solche Wörter wie Sinus oder Cosinus getrauen wir uns gar nicht mehr in den Mund zu nehmen. Wenn auch der arithmetische Befehlssatz nicht sehr umfangreich ist, so läßt sich doch mit diesem alles programmieren, was das Herz begehrt.

### Addition und Subtraktion

Wir haben die Wahl zwischen binärem oder BCD-System. Vor dem Addieren ist das Carry-Flag auf Null zu setzen, vor das Subtrahieren auf Eins.

```
CLC
LDA #$04
ADC #$03
```

Wie gewünscht, ist das Ergebnis im Akku sichtbar. Wir nehmen nun Zahlen, die geringfügig größer sind.

```
CLC
LDA #$08
ADC #$07
```

Die dezimale Darstellung zeigt den Wert 15, wie es sein soll. Die hexadezimale Darstellung wartet mit #$0F auf, was zwar noch vorstellbar ist, jedoch für uns Zehn-Finger-Menschen schon etwas unanschaulich wirkt. Doch es kommt noch schlimmer.

```
CLC
LDA #$28
ADC #$49
```

Weder die deziamel Darstellung noch die Hex-Darstellung zeigen an, daß 28 und 49 die Summe 77 ergeben müßten. Wir und die CPU benützen eben zwei unterschiedliche Zahlensysteme. Wenn wir die Zahlen #$28 und #$49 in die Index-Register laden, sehen wir, daß #$28 eben nicht auch dezimal 28 bedeutet. Die Summe der Werte in den Indexregistern ergibt jedoch genau den Wert, den wir im Akku vorfinden. Wir brauchen nur die dezimal dargestellten Werte miteinander vergleichen. Keine Sorge, wir brauchen uns deshalb nicht umgewöhnen, von nun an in Hex-Zahlen zu rechnen.

### HEX, DEZIMAL UND BCD

Die Schwierigkeit ist nur, daß der Rechner und wir andere Zahlensysteme verwenden. Der Programmanwender merkt davon nichts.
Der Programmierer hat dafür Sorge zu tragen, daß die dezimal eingegebenen Zahlen in Hex-Zahlen umgewandelt werden. Die CPU kann nun auf ihre Weise mit diesen Zahlen rechnen. Das Ergebnis muß wieder in das Dezimalsystem zurückübersetzt werden. Wenn der Rechner allerdings nicht einmal multiplizieren kann, so ist dieses wohl gerade kein leichtes Unterfangen. Glücklicherweise kommt uns das Statusregister mit dem Dezimalflag einen Schritt entgegen, so daß wir die Rechenleistungen gut verfolgen können.

```
SED
CLC
LDA #$08
ADC #$07
LDA #$28
ADC #$49
```

Bei der Subtraktion setzen wir das Carry-Flag.

```
SEC
LDA #$77
SBC #$49
```

Das Ergebnis entspricht auch unseren Erwartungen. Wenn der Rechner nur mehr Stellen zulassen würde! Mit dem Carry-Flag hat es noch eine besondere Bewandtnis.

```
CLC
LDA #$60
ADC #$50
```

Wir beobachten, daß das Carry-Flag nun gesetzt ist. Da die Summe aus 50 und 60 größer als 99 ist, ist ein Übertrag entstanden, der bei weiteren Additionen dazuaddiert wird.

```
TAX
LDA #$01
ADC #$02
```

Eins und zwei sind plötzlich vier, da der Übertrag aus der vorangegangenen Rechnung Berücksichtigung fand.
Betrachten wir die beiden Rechnungen im Zusammenhang, so haben wir eigentlich die Summe aus 160 und 250 gebildet. Akku und X-Register enthalten 0410. Wenn wir mehrmals hintereinander die einzelnen Stellen einer vielstelligen Zahl unter Berücksichtigung der Überträge addieren, so sind uns von der Stellenzahl her keine Grenzen gesetzt. Wir brauchen nur einen bestimmten Speicherbereich unserer Rechnenoperation vorsehen. Die jeweilige Summe aus den beiden Summanden schreiben wir wieder in den Speicher.

```
BRK
M3000
F3000,3FFF,0
M3000
```

Nach der Adreßangabe >3000 finden wir die Hexzahlen 00 00 00. Diese ändern wir in eine beliebige sechsstellige BCD-Zahl. Mit der Zeile weiter unten wollen wir genauso verfahren.

```
G
CLC
LDA $3002
ADC $300A
STA $3012
LDA $3001
ADC $3009
STA $3011
LDA $3000
ADC $3008
STA $3010
BRK
M3000
```

Wir können nun das Ergebnis betrachten. Wenn nicht vorne bei den Hunderttausendern noch ein Überlauf geschehen ist, sollte unser Ergebnis stimmen.
Für mehrstellige Subtraktionen dient ebenso wieder das Carry-Flag. Normalerweise müßte es in diesem Falle Borrow-Flag heißen. Für unsere CPU gilt, daß ein gesetztes Carry ein gelöschtes Borrow bedeutet und umgekehrt. Andere CPUs können auch eine andere Regelung besitzen. Wir versuchen uns nun an der mehrstelligen Subtraktion. Zahlen brauchen wir keine hierzu noch eingeben. Wir löschen lediglich eine Zeile im Monitor.

```
F3000,3007,0
M3000
```

Der erste Summand aus unserer Addition ist verschwunden. Von der früheren Summe ziehen wir den zweiten Summanden ab und sollten wieder die erste Zeile, wie gehabt, bekommen.

```
G
SEC
LDA$3012
SBC $300A
STA $3002
LDA$3011
SBC $3009
STA $3001
LDA$3010
SBC $3008
STA $3000
BRK
M3000
1;
```

Für unsere Zwecke reicht das Rechnen mit positiven Zahlen vollständig. Die CPU kennt schließlich nur positive Adressen von $0000 bis $FFFF. Wenn einmal etwas errechnet werden muß, wie die Länge eines Speicherbereiches aus der Anfangsadresse und der Endadresse, so ist maximal eine zweistellige Rechnung vonnöten. Für diejenigen, die sich einmal vornehmen sollten, eine Tabellenkalkulation oder ein Buchrechnungsprogramm in Maschinensprache zu programmieren, sind auch das Vorzeichenflag und das Überlauf-Flag von Bedeutung. Wir definieren uns nun, daß Zahlen von #$00 bis #$7F positiv sein sollen und Zahlen von #$80 bis #$FF negativ, und schenken den Status-Bit sechs und sieben nun unsere Aufmerksamkeit. Zuerst verlassen wir aber erst einmal den Monitor und setzen das Dezimalflag zurück.

```
G
CLD
```

Wir können jetzt mit dem Rechnen beginnen.

```
CLC
LDA #$40
ADC #$50
```

Das Ergebnis #$90 ist als negative Zahl definiert, was das Vorzeichen-Flag auch angibt. Nur kann die Summe aus zwei positiven Zahlen wohl kaum negativ sein. Daher ist das Überlauf-Flag gesetzt. Es signalisiert, daß das Vorzeichen-Flag den Sachverhalt nicht richtig wiedergibt.

```
CLC
LDA #$40
ADC #$20
```

Beide Flags sind null. Die Zahl ist positiv. Das Vorzeichen-Flag zeigt es richtig an.

```
CLC
LDA #$40
ADC #$FF
```

Die Hexzahl #$FF entspricht einer negativen Eins. Das Ergebnis ist daher positiv und richtig.

```
SEC
LDA #$10
SBC #$20
```

Das Ergebnis ist negativ, das Vorzeichen-Flag zeigt dieses an.

```
SEC
LDA #$90
SBC #$20
```

Von einer negativen Zahl wird eine positive abgezogen. Das Ergebnis muß daher negativ sein. Daß das Vorzeichen-Flag den Sachverhalt nicht richtig wiedergibt, wird vom Überlauf-Flag signalisiert.

```
SEC
LDA #$05
SBC #$90
```

Von einer positiven Zahl wird eine negative subtrahiert. Das Ergebnis muß daher positiv sein. Das Vorzeichen-Flag gibt es richtig wieder.

```
SEC
LDA #$30
SBC #$90
```

Das Ergebnis muß aus den vorher genannten Gründen wieder positiv sein. Das Überlauf-Flag widerspricht dem Vorzeichen-Flag.

```
CLC
LDA #$90
ADC #$90
```

Zu einer negativen Zahl wird eine negative Zahl addiert. Das Ergebnis muß negativ sein. Das Überlauf-Flag sagt aus, daß das Vorzeichen-Flag nicht recht hat.
Wir sprachen davon, daß die CPU nicht multiplizieren könne. Ganz richtig ist dieses jedoch nicht. Wir, die im Dezimalsystem rechnen, benötigen das Einmaleins bis zum Neunereinmaleins. Das Multiplizieren mit zehn geschieht durch einfaches Linksverschieben. Im Binärsystem, das nur die Zahlen null oder eins kennt, brauchen wir nur das Einereinmaleins und das Verschieben, das einer Multiplikation mit zwei gleichkommt. Mit eins multiplizieren ist keine Kunst, da dieses sich von selbst erübrigt. Für das Verschieben stehen uns diverse Befehle zur Verfügung.

```
LDA #$01
ASL
```

Der Akkumulatorinhalt wird durch den *ASL-Befehl,* Arithmetic Shift Left, um eins nach links geschoben. Wiederholen Sie den ASL-Befehl, bis die Eins aus dem Akku verschwindet. Ist dieses der Fall, erscheint ein gesetztes Bit im Carry-Flag. Nach einem weiteren Befehl ist auch dieses verschwunden. Sie brauchen den ASL-Befehl nicht jedesmal neu einzugeben. Unser Programm hat eine Wiederholautomatik. Wenn Sie nur die Return-Taste drücken, wird der zuletzt eingegebene Befehl immer wieder ausgeführt.

```
LDA #$01
ROL
```

Fas das gleiche wie der ASL-Befehl scheint der *ROL-Befehl,* ROtate Left, zu bewirken, daß dieses jedoch nicht so ist, sehen wir, wenn die Eins aus dem Akku verschwunden und im Carry-Flag gelandet ist. Beim nächsten ROL-Befehl geht sie von dort wieder in den Akku über.

```
LDA #$80
LSR
```

*Logical Shift Right* bewirkt dasselbe wie ASL, nur in umgekehrter Richtung.

```
LDA #$02
ROR
ROR
ROR
```

*Rotate Right* funktioniert genau wie ROL in umgekehrter Richtung. Diese Befehle greifen nicht nur auf den Akku zu, sondern es lassen sich auch beliebige Speicherstellen dadurch verändern. Auf ASL und LSR könnte zur Not auch verzichtet werden, da durch Löschen des Carry-Flags dafür gesorgt werden kann, daß die Rotate-Befehle stets eine Null in den Akku oder die Speicherstelle rotieren.
Das Hereinnehmen des Carry-Flags durch die Rotate-Befehle ist äußerst nützlich. Denken wir hierzu nur einmal an eine Grafik-Hardcopy. Durch acht mal acht Bit wird das Bit-Muster eines Zeichens bestimmt. Der Bildschirm hat es zu acht Reihen mit jeweils einem Byte, der Drucker jedoch braucht es in Spaltenform. Nachdem die Matrix aus dem Bildschirmbereich herauskopiert wurde – schließlich wollen wir den Bildschirminhalt nicht zerstören –, können wir mit ROL-Befehlen für den Drucker zurechtschneidern. Aus jedem der acht Bildschirm-Byte rotieren wir je ein Bit heraus und rotieren es mit einem zweiten ROL-Befehl in den Akku. Nachdem der Akkuinhalt ausgegeben ist, holen wir uns nach demselben Verfahren die zweite Spalte heraus, bis alle acht Spalten ausgegeben sind.
Zum Multiplizieren laden wir jeweils eine Zahl, deren Produkt nicht größer als 255 oder #$FF betragen soll, in die Indexregister. Das Produkt soll nach der Multiplikation im Akku stehen, die Faktoren noch in den Indexregistern.

```
LDX #$06
LDY #$09
```

Es gilt nun ein geeignetes Verfahren, die Multiplikation zu finden. Wir laden hierzu den Akku mit Null und legen die beiden Werte in zwei Speicherstellen ab, so daß wir sowohl schieben als auch addieren können.

```
LDA #$00
STX $3000
STY $3001
```

Eine Zahl können wir dadurch mit sechs multiplizieren, daß wir sie in den Akku laden. Dadurch haben wir sie einmal erfaßt. Indem wir sie nach links verschieben, haben wir sie bereits mit zwei multipliziert. Wenn wir sie noch einmal dazuaddieren, ist sie schon mit drei malgenommen. Ein abschließendes Linksverschieben bringt das gewünschte Ergebnis.
Für beliebige Zahlen, die bis zu acht Bit lang sein dürfen, benötigen wir noch ein paar weitere Befehle. Acht Bit sind zu untersuchen. Daher muß unser Rechner bis acht zählen können. Zum Zählen dienen Befehle, mit denen vorwärts und rückwärts gezählt werden kann.

```
LDY #$08
DEY
```

Bei jedem DEY wird das Y-Register um Eins erniedrigt, bis beim Erreichen von Null das Zero-Flag diesen Zustand anzeigt. Beim weiteren Decrementieren spricht das Vorzeichen-Flag an.

```
INY
```

Mit INY kann das Y-Register hochgezählt werden. Die restlichen Befehle sollen nun nicht gleich ausprobiert werden, da wir den Inhalt des X-Registers weder decrementieren noch incrementieren wollen, da wir diesen, bevor wir nicht multipliziert haben, gerne unverändert hätten.
Neben den Indexregistern können auch Speicherstellen angesprochen werden. Es folgt eine Auflistung der Increment- und Decrement-Befehle

```
INX
DEX
INY
DEY
INC
DEC
```

Wir laden also das Y-Register mit dem Wert #$08 und unser Algorithmus kann beginnen. Da hierbei auch Verzweigungen stattfinden, sei er in einzelnen Schritten verfaßt.

1. ASL
2. ASL $3000
3. Wenn Carry = 0, dann Sprung nach 6.
4. CLC
5. ADC $3001
6. DEY
7. Wenn Zero = 0, dann Sprung nach 1.
8. LDY $3001

Etwas mühsam ist es, solch einen Algorithmus mit der Hand durchzuprobieren. Zum Programmieren fehlt uns fast nichts mehr, denn unser Befehlssatz ist bereits nahezu vollständig. Uns fehlt nur noch der Befehl, um ein Unterprogramm anzuspringen, und der Befehl für den Rücksprung. Der Befehl für den Rücksprung heißt *RTS*. Wir können ihn in unserem CPU-Programm nicht aufrufen. Genauso, wie die Branch-Befehle verwenden wir ihn erst in einem richtigen Programm.
Der *JSR*-Befehl, der ein Unterprogramm anspringt, ist auch schon in unserem CPU-Programm verfügbar. Wir dürfen ihn allerdings erst verwenden, wenn wir ein anzuspringendes Maschinenprogramm vor uns haben. Besonders beachtet werden muß bei diesen Befehlen, daß der JSR, Jump SubRoutine, die Rücksprungadresse auf dem Stapel ablegt. Nur so kann *RTS*, ReTurn from Subroutine, wieder an die richtige Stelle im Hauptprogramm zurückkehren. Dies bedeutet, daß ein auf den Stapel gepushter Wert nicht nach einem Sprung in eine Unterroutine so ohne weiteres abgehoben werden kann. Anstatt dem geretteten Akkuinhalt würde ein Pull-Befehl ein Byte der Rücksprungadresse abheben.
Der RTS hätte als Rücksprungadresse eine Kombination von beretteten Akkuinhalt als High-Byte und High-Byte-Rücksprungadresse als Low-Byte vor sich und würde ganz schön ins Nirwana stürzen. Außer den zwei soeben genannten Befehlen gibt es noch einen unbedingten Sprung namens *JMP*, nur für Sprünge in einem Bereich von $7F Adressen vorwärts und $80 Adressen rückwärts eingesetzt werden kann, sondern den Programmzähler auf jede beliebige Speicheradresse einstellen kann. Stapelbeeinflussungen finden hierbei nicht statt.
Um vollständig zu sein, wollen wir auch einen ganz unscheinbaren Befehl nicht unerwähnt lassen. Sie können ihn ruhig gleich ausprobieren. Er heißt *NOP*, was No OPeration bedeutet. Dieser Befehl bewirkt nichts. Er wird gerne als Platzhalter verwendet, so wie wir es auch im Maschinenteil unseres CPU-Programmes taten. An eine bestimmte Stelle des Programms setzen wir je nach auszuführendem Befehl entweder ein zwei oder drei Byte lange Befehle ein. Damit die CPU bei ein oder zwei Byte langen Befehlen nicht über den zurückgebliebenen Rest der drei Byte langen Befehle stolpert, füllten wir die verbleibende ein oder zwei Adressen mit NOP-Codes auf. Wir schreiben jetzt unser erstes Maschinenprogramm. Da wir den Algorithmus für die Multiplikation bereits notiert hatten, fällt es uns bestimmt nicht schwer, diesen mit Hilfe des Monitors einzugeben.

```
BRK                 BCC $3010
A3002 LDA #$00      CLC
STX $3000           ADC $3001
STY $3001           DEY
LDY #$08            BNE $300C
ASL                 LDY $3001
ASL $3000           RTS
```

Der Sprungbefehl bei Adresse $3010 stimmt noch nicht ganz. Wir haben, da wir die Sprungadresse beim Niederschreiben noch nicht gekannt haben, einen provisorischen Wert eingesetzt. Da jetzt die Adresse des DEY-Befehles bekannt ist, ersetzen wir BCC $3010 durch BCC $3016. Den Monitorbefehl zum Betrachten eines Speicherbereichs im Hexformat kennen wir bereits. Zum Betrachten eines Maschinenprogramms wäre es sehr schön, wenn dieses sich auch in Assembler-Mnemonics anschauen ließe. Auch dafür gibt es einen Monitorbefehl. Er heißt *Disassemble* und wird mit dem Buchstaben **D**, gefolgt von einer Adresse eingegeben. Sie dürfen ruhig einmal den Bildschirm löschen. Danach sehen wir uns das Maschinenprogramm wieder an.

```
D3002
```

Ganz wird das Multiplikationsprogramm durch diesen Monitorbefehl nicht angezeigt. Um den Rest zu disassemblieren, ist nur noch ein D ohne nachfolgende Adresse einzugeben.

```
D
```

Disassembliert sollte das Programm sich nun darstellen, wie es nachfolgend zu sehen ist.

```
. 3002  a9 00 30   lda  #$00
. 3004  8e 00 30   stx  $3000
. 3007  8c 01 30   sty  $3001
. 300a  a0 08      ldy  #$08
. 300c  0a         asl
. 300d  0e 00 30   asl  $3000
. 3010  90 04      bcc  $3016
. 3012  18         clc
. 3013  6d 01 30   adc  $3001
. 3016  88         dey
. 3017  d0 f3      bne  $300c
. 3019  ac 01 30   ldy  $3001
. 301c  60         rts
```

Rechts neben dem Punkt erscheint die jeweilige Speicheradresse. Die nachfolgenden ein, zwei oder drei Byte sind der Operationscode und ein oder zwei Parameter. Darauf folgen die disassemblierten Mnemonics. Interessant ist, daß bei der absoluten Adressierung dem Operationscode erst das Low-Byte der Speicheradresse und dann erst das High-Byte folgt. Den Operationscodes für die bedingten Sprünge folgt nicht eine absolute Sprungadresse, sondern nur ein einziges Byte, das angibt, um wieviel Adressen der Programmzähler verstellt werden soll. Im Verzweigungsfalle landet der Programmzähler beim BCC in Adresse $3010 nicht bei $3012 sondern, vier Adressen weiter, bei $3016. Zahlen ab $80 sind negativ, wobei $ff der Dezimalzahl –1 entspricht und $80 der Dezimalzahl –128.
Der Sprungbefehl in Adresse S3017 landet daher bei Adresse $300 c. Relative Sprünge bringen einen entscheidenden Vorteil. Die Programme sind verschiebbar. Wir kopieren unser Maschinenprogramm.

```
T3002,301C,3100
D3100
```

Die disassemblierten Sprungadressen sind andere, als wir ursprünglich erfaßt hatten. Der Disassembler zeigt jetzt automatisch die richtigen neuen Sprungadressen an, da ja nur die Sprungweite im Maschinenprogramm steht.

```
G
```

Wir stoßen auf eine Eigenheit des Maschinenmonitors. Sobald wir Assemblierungen vorgenommen haben, landen wir beim Fortsetzen mit *Go* nicht, wie wir es gewohnt sind, im CPU-Programm, sondern im Eingabemodus des BASIC-Interpreters. Unser CPU-Programm ist nicht verlorengegangen, es läßt sich wieder starten.

```
RUN
```

Jetzt haben wir die Gelegenheit, den JSR-Befehl und unser Maschinenprogramm auszuprobieren.

```
LDX #$05
LDY #$07
JSR  $3002
```

Das Multiplizieren klappt ja direkt hervorragend. Mit dem Befehlssatz des Prozessors sind wir am Ende angelangt. Zu besprechen sind lediglich noch die verschiedenen Adressierungsarten.

## VII. ADRESSIERUNGSARTEN DER CPU

**Neben impliziter, unmittelbarer und absoluter Adressierung bietet die CPU unseres Rechners eine Vielzahl weiterer Adressierungen. Dreizehn sind es insgesamt. Mit indirekter und indizierter Adressierung können wir ganze Speicherbereiche durchwühlen. Eine Befehlstabelle gibt uns den Überblick.**

In der Beschreibung der Adressierungsarten sind auch Befehlslänge und Befehlsdauer angegeben. Die Befehlsdauer miß sich in Taktzyklen. Im Normalfall wird unsere CPU mit der Frequenz von einem Megahertz getaktet. Das sind eine Million Takte in der Sekunde. Ein Takt dauert so eine Millionstel Sekunde, anders ausgedrückt, eine Mikrosekunde. Ein Befehl, der in zwei Taktzyklen abgearbeitet wird, braucht somit zwei Mikrosekunden. 500000 solcher Befehle könnten also in der Sekunde durchgeführt werden, eine ganz erklecklliche Anzahl. Kein Wunder also, daß in Maschinensprache geschriebene Programme als ungeheuer schnell gelten.
Manche Adressierungsarten arbeiten schneller als andere. Manche nehmen weniger Programmspeicher in Anspruch, weil die Befehslänge eventuell kürzer ist. Andere Adressierungsarten fixieren uns nicht auf bestimmte Adressen, sondern erlauben uns, Routinen zu schreiben, die auf jeden beliebigen Speicherbereich zugreifen können. In der Befehlstabelle sehen wir, welche Adressierungen für welche Operationen zur Verfügung stehen, und können so das für unseren Zweck geeignetste auswählen.

### A. Unmittelbare Adressierung

**Beispiel:** LDY #$05
**Befehlslänge:** Zwei Byte
**Taktzyklen:** Zwei

Dem Operationscode folgt unmittelbar das Daten-Byte, das mit dem Akkumulatorinhalt verknüpft wird.

### B. Absolute Adressierung

**Beispiel:** STA $3000
**Befehlslänge:** Drei Byte
**Taktzyklen:** Drei bis sechs

Dem Operationscode folgt eine Zwei-Byte-Adresse. Hierbei wird erst das Low- und dann das High-Byte im Speicher abgelegt. Der JMP-Befehl benötigt drei Taktzyklen, der JSR sechs. Befehle, die eine Änderung einer Speicherzelle bewirken, benötigen sechs Taktzyklen. Die Ausnahme bildet der Store-Befehl mit vier Taktzyklen. Lade- und sonstige Befehle, die den Inhalt einer Speicherzelle mit einem CPU-Register verknüpfen, dauern ebenfalls vier Taktzyklen.

### C. Zero-Page-Adressierung

**Beispiel:** LDA $01
**Befehlslänge:** Zwei Byte
**Taktzyklen:** Drei bis fünf

Um Adressen von $00 bis $FF anzusprechen, kann auf ein High-Adress-

Byte verzichtet werden. Dem Operationscode folgt nur ein einziges Adress-Byte. Kürzer ist nicht nur die Länge des Befehls, sondern auch die Abarbeitungszeit. Ein Taktzyklus wird gespart.

### D. Akku-Adressierung

**Beispiel:** ASL
**Befehlslänge:** Ein Byte
**Taktzyklen:** Zwei

Da keine Werte übergeben werden und keine Speicherzelle zu adressieren ist, wird nur der Operationscode benötigt.

### E. Implizite Adressierung

**Beispiel:** SEC
**Befehlslänge:** Ein Byte
**Taktzyklen:** Zwei bis sechs

Die Adressierung ist bereits durch den Operationscode bestimmt. Daten oder Adreßparameter erübrigen sich daher. Befehle, die nicht auf den Speicher zugreifen, dauern zwei Taktzyklen. Push-Befehle vier, die Rücksprünge RTS und RTI benötigen gar sechs Zyklen. Mit RTI wird von einem Interrupt wieder zurückgesprungen. Da ein Interrupt nicht nur die Rücksprungadresse, sondern auch das Statusflag auf den Stapel rettet, wird dieses durch RTI wieder restauriert.

### F. Indiziert-indirekte Adressierung

**Beispiel:** LDA ($D0,X)
**Befehlslänge:** Zwei Byte
**Taktzyklen:** Sechs

Dem Operationscode folgt ein Adreß-Byte. Indirekt adressiert heißt, daß nicht die durch den Adreßparameter bestimmte Zelle adressiert wird, sondern, daß in der spezifizierten und der nachfolgenden Speicherstelle sich erst die Adresse der Zelle findet, auf die die Operation zugreift. Indiziert heißt, daß zur Adreßbildung zusätzlich ein Index-Register einbezogen wird. Bei der indiziert-indirekten Adressierung der CPU 6502 ist dieses das X-Register.
Der Inhalt des X-Registers wird in unserem Beispiel zur Adresse $D0 addiert. Wir erhalten so die Adresse der Speicherstelle. In dieser und der nachfolgenden finden wir endlich die Adresse unserer Speicherstelle, die es anzusprechen gilt. Bei der Summierung der X-Register und der Zero-Page-Adresse bleibt ein Übertrag unberücksichtigt, so daß für indirekte Adressierungen auch wirklich nur Speicherstellen von $00 bis $FF in Frage kommen.

Die meisten Zero-Page-Adressen sind bereits für Operationen des Betriebssystems vergeben. Zur freien Verfügung steht dem C16/116/Plus4-User der Bereich von $D0 bis $E8, in dem er nach Belieben schalten und walten kann. Um die Sache noch etwas zu verdeutlichen, nehmen wir einmal an, das X-Register hätte den Wert #$02, in der Speicherstelle $D2 stünde der Wert #$00 und in $D3 der Wert #$30. Die adressierte Speicherstelle wäre in diesem Falle $3000.

### G. Indirekt-indizierte Adressierung

**Beispiel:** LDA $(D0),Y
**Befehlslänge:** Zwei Byte
**Taktzyklen:** Fünf bis sechs

Der *Store*-Befehl benötigt sechs Zyklen, die anderen Befehle nur fünf. Diese Adressierungsart wird öfter benötigt als die indiziert-indirekte. Zur Adresse in den zwei Speicherstellen in der Zero-Page wird der Inhalt des Y-Registers addiert. Die Summe davon ist die Adresse unserer anzusprechenden Speicherzelle. Hätte das Y-Register den Wert #$81, die Speicherstelle $D0 den Wert #$80 und $D1 den Wert #$30, so würde durch die Operation die Speicherstelle $3101 angesprochen werden. Überträge bei der Summierung der Adress-Low-Byte werden berücksichtigt. Bei allen indizierten Adressierungsarten ist, wenn solche Page-überschreitungen auftreten, ein Taktzyklus hinzuzuaddieren.

### H. Zero-Page-indizierte Adressierung mit Indexregister X

**Beispiel:** LDA $D0,X
**Befehlslänge:** Zwei Byte
**Taktzyklen:** Vier bis sechs

Die Abarbeitungszeit der Befehle entspricht derjenigen der absoluten Adressierung. Zur Adresse wird der Inhalt des X-Registers addiert. Es können nur Speicherstellen von $00 bis $FF angesprochen werden. Ein Übertrag bei der Addition der Adressen wird unterschlagen.

### I. Direkt indizierte Adressierung mit Indexregister X

**Beispiel:** LDA $3000,X
**Befehlslänge:** Drei Byte
**Taktzyklen:** Vier bis sieben

Der *STA*-Befehl benötigt fünf Taktzyklen, sonstige Befehle, die den Inhalt einer Speicherstelle verändern, gar sieben. Alle übrigen Operationen geben sich mit vier Zyklen zufrieden.

Wie bei der direkten Adressierung besteht die Basisadresse aus zwei Byte.
Anders als bei der Zero-Page-indizierten Adressierung wird bei der Bildung der Operandenadresse ein aus der Summierung der Low-Byte resultierender Übertrag berücksichtigt. Hätte X den Wert #$81, so würde mit LDA $3080,X die Speicherstelle $3101 angesprochen.

### J. Direkt indizierte Adressierung mit Indexregister Y

**Beispiel:** LDA $3000,Y
**Befehlslänge:** Drei Byte
**Taktzyklen:** Vier bis fünf

Zur Basisadresse wird das Y-Register addiert. Der STA-Befehl braucht fünf Taktzyklen, die anderen Befehle nur vier.

### K. Relative Adressierung

**Beispiel:** BCC $3016
**Befehlslänge:** Zwei Byte
**Taktzyklen:** Zwei bis drei

Dem Operanden folgt ein Byte, das die Sprungweite signalisiert. Zahlen von #$80 bis #$FF gelten als negativ. Sprünge können so nur in einem Bereich von Null bis 127 Adressen nach vorne und −1 bis 128 Adressen nach rückwärts erfolgen. Ist die Verzweigungsbedingung nicht erfüllt, so benötigt der Befehl zwei Taktzyklen, im Verzweigungsfalle drei. Bei Page-Überschreitung ist ein weiterer Taktzyklus hinzuzuaddieren.

### L. Indirekte Adressierung

**Beispiel:** JMP ($0324)
**Befehlslänge:** Drei Byte
**Taktzyklen:** Fünf

Eine indirekte Adressierung, die nicht auf Zero-Page-Adressen beschränkt ist, kennt nur der *JMP*-Befehl. In unserem Falle erfolgt der Sprung nach der Adresse, die in den Speicherstellen $0324 und $0325 vermerkt ist. Einige wichtige Betriebssystemroutinen unserer Commodore-Rechner werden indirekt über sogenannten Sprungvektoren aufgerufen, die im RAM lokalisiert sind. Durch das Ersetzen der Adresse in den Speicherstellen $0324 und $0325, ist es ein Leichtes, eine neue Ausgaberoutine zu schreiben, die zum Beispiel für einen Epson-Drucker den Commodore-ASCII in Standard-ASCII umwandelt. Die Adresse eines direkten Sprunges im ROM hätten wir nicht manipulieren können, da aus dem ROM ja bekanntlich nur gelesen werden kann.

| | A | B | C | C | E | F | G | H | I | J | K | L | M | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC | * | * | * | | | * | * | * | * | * | * | | | N | V | | | Z | C |
| AND | * | * | * | | | * | * | * | * | * | * | | | N | | | | Z | |
| ASL | | * | * | * | * | | | | * | * | | | | N | | | | Z | C |
| BCC | | | | | | | | | | | * | | | | | | | | |
| BCS | | | | | | | | | | | * | | | | | | | | |
| BEQ | | | | | | | | | | | * | | | | | | | | |
| BIT | | * | * | * | | | | | | | | | | N | V | | | Z | |
| BMI | | | | | | | | | | | * | | | | | | | | |
| BNE | | | | | | | | | | | * | | | | | | | | |
| BPL | | | | | | | | | | | * | | | | | | | | |
| BRK | | | | | * | | | | | | | | | | | | | | |
| BVC | | | | | | | | | | | * | | | | | | | | |
| BVS | | | | | | | | | | | * | | | | | | | | |
| CLC | | | | | * | | | | | | | | | | | | | | C |
| CLD | | | | | * | | | | | | | | | | | D | | | |
| CLI | | | | | * | | | | | | | | | | | | I | | |
| CLV | | | | | * | | | | | | | | | | V | | | | |
| CMP | * | * | * | | | * | * | * | * | * | * | | | N | | | | Z | C |
| CPX | * | * | * | | | | | | | | * | | | N | | | | Z | C |
| CPY | * | * | * | | | | | | | | * | | | N | | | | Z | C |
| DEC | | * | * | | | | | | | * | | | | N | | | | Z | |
| DEX | | | | | * | | | | | | | | | N | | | | Z | |
| DEY | | | | | * | | | | | | | | | N | | | | Z | |
| EOR | * | * | * | | | * | * | * | * | * | * | | | N | | | | Z | |
| INC | | * | * | | | | | | * | * | | | | N | | | | Z | |
| INX | | | | | * | | | | | | | | | N | | | | Z | |
| INY | | | | | * | | | | | | | | | N | | | | Z | |
| JMP | | * | | | | | | | | | | * | | | | | | | |
| JSR | | * | | | | | | | | | | | | | | | | | |
| LDA | * | * | * | | | * | * | * | * | * | * | | | N | | | | Z | |
| LDX | * | * | * | | | | | | | | * | | * | N | | | | Z | |
| LDY | * | * | * | | | | | | * | * | | | | N | | | | Z | |
| LSR | | * | * | * | | | | | * | * | | | | N | | | | Z | C |
| NOP | | | | | | * | | | | | | | | | | | | | |
| ORA | * | * | * | | | * | * | * | * | * | * | | | N | | | | Z | |
| PHA | | | | | * | | | | | | | | | | | | | | |
| PHP | | | | | * | | | | | | | | | | | | | | |
| PLA | | | | | * | | | | | | | | | N | | | | Z | |
| PLP | | | | | * | | | | | | | | | N | V | D | I | Z | C |
| ROL | | * | * | * | * | | | | * | * | | | | N | | | | Z | C |
| ROR | | * | * | * | * | | | | * | * | | | | N | | | | Z | C |
| RTI | | | | | | * | | | | | | | | N | V | D | I | Z | C |
| RTS | | | | | | * | | | | | | | | | | | | | |
| SBC | * | * | * | | | * | * | * | * | * | * | | | N | V | | | Z | C |
| SEC | | | | | | * | | | | | | | | | | | | | C |
| SED | | | | | | * | | | | | | | | | | D | | | |
| SEI | | | | | | * | | | | | | | | | | | I | | |
| STA | | * | * | * | | | * | * | * | * | * | | | | | | | | |
| STX | | * | * | | | | | | | | | | * | | | | | | |
| STY | | * | * | | | | | * | | | | | | | | | | | |
| TAX | | | | | * | | | | | | | | | N | | | | Z | |
| TAY | | | | | * | | | | | | | | | N | | | | Z | |
| TSX | | | | | * | | | | | | | | | N | | | | Z | |
| TXA | | | | | * | | | | | | | | | N | | | | Z | |
| TXS | | | | | * | | | | | | | | | | | | | | |
| TYA | | | | | * | | | | | | | | | N | | | | Z | |

### M. Zero-Page-indizierte Adressierung mit Indexregister Y

Beispiel: LDX $D0,Y
Befehlslänge: Zwei Byte
Taktzyklen: Vier

Indexregister ist Y. Ansonsten gilt das bereits über diese Adressierungsart mit Indexregister X Gesagte.
Die *Übersichtstabelle* gibt Auskunft, welche Adressierungsarten für welche Operationen bereitstehen. Sollte der Assembler einmal einen Befehl nicht annehmen wollen, wie LDA $D0,Y, so belehrt uns die Tabelle sogleich, daß es eine Zero-Page-indizierte Adressierung mit Y-Register nur für LDX und STX gibt. Ein paar Spalten weiter links sagt sie uns, daß die direkt indizierte Adressierung jedoch möglich ist, und wir mit LDA $00D0,Y zum Ziel kommen.

### Beeinflussung der Statusflags

Zur Erstellung der Programmlogik ist es notwendig, zu wissen, welcher Befehl welches Statusflag beeinflußt. In der rechten Spalte unserer Übersichtstabelle findet sich diese Information. Auf die Darstellung von Bit vier und fünf des Statusregisters haben wir verzichtet, da Bit fünf keine Funktion hat und Bit vier, das Break-Bit, nur bei einem einzigen Befehl gesetzt wird, dem BRK-Befehl.
Dieses Flag wird im Betriebssystem unseres Rechners nur ein einziges Mal beim Interrupt abgefragt. Ist dieses Bit dort gesetzt, wird über den Sprungvektor $032C der Maschinenmonitor aufgerufen. Wollen wir den Sprung in den Monitor durch eine andere Routine ersetzen, so genügt eine Umstellung des Break-Vektors. Um das Break-Bit brauchen wir uns also nicht zu kümmern, wenn wir nicht selbst ein vollkommen neues Betriebssystem schreiben wollen.
Wir ersehen aus der Tabelle, daß die Statusflags bei Sprüngen, ob bedingten, absoluten oder Unterroutinen-Aufrufen, nicht beeinflußt werden. Ebenfalls keine Änderungen treten beim Schreiben von Registerinhalten in Speicherstellen auf, ob dieses nun mit Store-Befehlen, wie STA, STX, STY oder mit den Stapelbefehlen PHA und PHP geschieht.
Für den Registertransfer TSX, BRK, dem Sprung in den Break-Interrupt, und NOP, das gar nichts macht, gilt dasselbe.
Die spezifischen Set- und Clear-Befehle für die einzelnen Statusflags ändern nur das jeweilige Bit, auf das sie sich beziehen. PLP und RTI stellen das ehemals gerettete Statusflag wieder her. Die sonstigen Befehle beeinflussen das Vorzeichen-Bit N und das Zero-Flag Z. Bei Shift-, Rotate-Befehlen, Addition, Subtraktion und Vergleichen wird zusätzlich das Carry-Flag beeinflußt. Addition, Subtraktion und der Bit-Befehl wirken sich auch auf das Überlauf-Flag V aus.

## VIII. EIN- UND AUSGABEROUTINEN

**Über eine Standard-Sprungtabelle, das Kernel, kann jedes Ein- und Ausgabegerät angesprochen werden. Das Kernel sorgt dafür, daß Maschinenprogramme, die sonst keine systemspezifischen Adressen verwenden, auf jedem Acht-Bit-Commodore-Rechner lauffähig sind.**

Was nutzt ein Rechner, der eine CPU besitzt, über ROM und RAM verfügt, jedoch keine Möglichkeiten bietet, Daten ein- oder auszugeben? Ein Rechner, der nicht die Möglichkeit bietet, mit seiner Umgebung zu kommunizieren, ist zu überhaupt nichts nütze. Wir benötigen daher noch diverse Ein- und Ausgabeoperationen. Für die Kommunikation mit der Außenwelt besitzt der Rechner verschiedene I/O-Bausteine, die über bestimmte Adressen angesprochen werden können.
Da für das Ansprechen solcher Bausteine ziemliche Kenntnisse über dieselben und über den Aufbau des bestimmten Rechners erforderlich

wären, haben die Systemprogrammierer bereits die erforderlichen Routinen programmiert.
Ein solches Betriebssystem muß vorhanden sein, sonst könnten wir gar kein Programm in den Rechner bringen. Nicht einmal ein einziger Tastendruck würde vom Rechner verarbeitet werden können:
Der Teil des Betriebssystem, der für Ein- und Ausgaben verantwortlich ist, heißt *BIOS*, was Basic Input Output System bedeutet. Das Wort Basic hat nichts mit der Programmiersprache BASIC zu tun, sondern sagt aus, daß es hier um einfache Ein- und Ausgaberoutinen geht, die die Grundlage für die Kommunikation mit der Peripherie bilden, und auf die alle komplexeren Ein- und Ausgaberoutinen wie beispielsweise der PRINT-Befehl oder der INPUT-Befehl zurückgreifen.
Um auch bei unterschiedlich organisierten Rechnern dafür zu sorgen, daß diese in gleicher Weise zu handhaben sind, wird der Aufruf dieser Funktionen standardisiert. Beim VC20, C64, C16/116/Plus4 und dem C128 gibt es einen Speicherbereich im ROM, der an genau denselben Stellen eine Sprungliste, das Kernel, für diese wichtigen Systemaufrufe enthält. Um ein Zeichen aus dem Tastaturpuffer zu lesen, brauchen wir deshalb nicht zu wissen, wo dieser in unserem Rechner ist, und wie er verwaltet wird. Es existiert schließlich schon die Routine, die dies für uns erledigt. Wo diese Routine in unserem Rechner genau liegt, braucht uns ebenfalls nicht zu interessieren. Alles, was wir wissen müssen, ist, daß bei Ansprung der Adresse $FFE4 der Rechner in die gewünschte Routine verzweigt.

### Eingaberoutinen

Zwei fundamentale Routinen besitzt unser Rechner, mit denen wir Daten von einem externen Gerät lesen können:

### GETIN ($FFE4)

Mit dieser Routine können wir Daten von einem Eingabegerät einlesen. Die BASIC-Befehle GET und GET# bauen darauf auf. Wurde zuvor kein anderes Gerät als Eingabegerät definiert, so wird ein Zeichen aus dem Tastaturpuffer oder dem Funktionstastenpuffer an den Akku den Wert #$00 zugewiesen.

### JSR $FFE4

Im Akku sollte sich nun der Wert #$00 befinden. Wir geben den Befehl jetzt erneut ein. Da eine Wiederholung vorliegt, brauchen wir hierzu nur die Return-Taste zu drücken. Gleich nach dem Druck dieser Taste, sollten wir blitzschnell noch eine andere Taste drücken. Bis das in BASIC verfaßte CPU-Programm in das Maschinenprogramm verzweigt, sollte der ASCII-Wert der gedrückten Taste bereits im Tastenpuffer abgelegt sein. Der Akkumulator zeigt im Anschluß diesen Wert. GETIN wartet nicht auf einen Tastendruck, wie dieses der BASIC-Befehl GETKEY tut. Da je nach erhaltenem Wert auch das Statusflag beeinflußt wird, können wir das Warten auf einen Tastendruck leicht selbst programmieren.

```
BRK
A301D JSR $FFE4
BEQ $301D
RTS
G
RUN
JSR $301D
```

### BASIN ($FFCF)

Auch BASIN liest sowohl von der Tastatur, wie auch von sonstigen Eingabegeräten. Ist nicht die Tastatur das Eingabegerät, so besteht zwischen GETIN und BASIN kein Unterschied. Beim Lesen von der Tastatur ist der Unterschied nicht zu übersehen.

### JSR $FFCF

Deutlich wird hier, daß BASIN die Grundlage des INPUT-Befehls bildet. Es erscheint ein blinkender Cursor und wir können soviel Zeichen eingeben, wie wir gerne möchten. Erst beim Druck der Return-Taste wird die BASIN-Routine beendet und wir bekommen das an erster Stelle eingegebene Zeichen in den Akku. Wie bekommen wir aber nun die restlichen Zeichen? Da das CPU-Programm hierauf keine Antwort zu geben vermag, so sei Ihnen dieses jetzt verraten. Bei jedem folgenden Aufruf von BASIN wird das nächste Zeichen in den Akku geladen, bis schließlich das Ende der Eingabe erreicht ist, was der Akku uns mit #$0D für das Returnzeichen anzeigt. Eingegebene Leerstellen am Ende werden ignoriert. Das CPU-Programm stört diese Eigenheit von BASIN. Deshalb verfassen wir ein kleines Maschinenprogramm, mit dem wir Eingaben nach Wahl im Hauptspeicher unseres Rechners positionieren können. Wie dieses geht, sollten Sie nun schon wissen. Hier ist das disassemblierte Monitorlisting.

```
. 3023  84 d0     sty  $d0
. 3025  86 d1     stx  $d1
. 3027  a0 00     ldy  #$00
. 3029  20 cf ff  jsr  $ffcf
. 302c  91 d0     sta  ($d0),y
. 302e  e6 d0     inc  $d0
. 3030  d0 02     bne  $3034
. 3032  e6 d1     inc  $d1
. 3034  c9 0d     cmp  #$0d
. 3036  d0 f1     bne  $3029
. 3038  a4 d0     ldy  $d0
. 303a  a6 d1     ldx  $d1
. 303c  60        rts
```

Um das Programm verschiebbar zu gestalten, benutzten wir die indirekte Adressierung. Um einen größeren Bereich als 256 Byte ansprechen zu können, und nicht am Ende extra den Wert des Y-Registers zum Inhalt der Speicherstelle $D0 addieren zu müssen, verzichteten wir auf eine Indizierung, indem wir das Y-Register kurzerhand auf Null setzten. Schnell genug ist das Programm wohl schon auch so.
Vor dem ersten Aufruf dieses Programmes ist dem X-Register das High-Byte der Speicheradresse und dem Y-Register das Low-Byte mitzuteilen. Die Eingabe wird beim Aufruf ab dieser Adresse gespeichert. Sowohl die Speicherzellen $D0 und $D1, als auch die Indexregister weisen hinterher auf die den abgespeicherten Daten folgende Speicherstelle. Durch mehrmaligen Aufruf dieses Maschinenprogrammes können so kontinuierlich weitere Abspeicherungen vorgenommen werden. Sollten bei zwischendurch erfolgenden Operationen die Indexregisterinhalte verlorengehen, so braucht die Eingabe das nächste Mal anstatt mit JSR $3023 nur mit JSR $3027 angesprungen zu werden. Wir probieren in unserem CPU-Programm ein paar Eingaben.

```
LDX #$31
LDY #$00
JSR $3023
```

Durch Druck der Returntaste können wir weitere Eingaben vornehmen.

```
BRK
M3100
G
```

Im Monitor können wir uns vom Funktionieren der Routine überzeugen.

### Die Ausgaberoutine BSOUT ($FFD2)

BSOUT bildet die Grundlage für alle Ausgaben. Der im Akku befindli-

che Wert wird an das Ausgabegerät gesandt.
Sofern nichts anderes definiert wurde, ist dieses der Bildschirm.

```
LDA #$41
JSR $FFD2
```

Auf dem Bildschirm sollte der Buchstabe A erscheinen. Bei einem längeren auszugebenden Text ist jedesmal ein Zeichen in den Akku zu laden und es dann mit BSOUT auszugeben, ist die schnellste, aber natürlich nicht die richtige Methode. Genauso, wie bei der Eingabe, verfassen wir daher eine Routine, die Daten ausgibt, welche in irgendeinem Speicherbereich liegen.

```
. 303d  84 d2     sty  $d2
. 303f  86 d3     stx  $d3
. 3041  a0 00     ldy  #$00
. 3043  b1 d2     lda  ($d2),y
. 3045  20 d2 ff  jsr  $ffd2
. 3048  e6 d2     inc  $d2
. 304a  d0 02     bne  $304e
. 304c  e6 d3     inc  $d3
. 304e  c9 0d     cmp  #$0d
. 3050  d0 f1     bne  $3043
. 3052  a6 d3     ldx  $d3
. 3054  a4 d2     ldy  $d2
. 3056  60        rts
```

Oft erfolgen Ein- und Ausgaben abwechselnd. Damit keine gegenseitige Verstellung der Zeiger auftritt. Haben wir die Adressen $d2 und $d3 als Ausgabezeiger verwendet. Soll beim erweiterten C16 oder Plus4 auch der gebankte RAM-Speicher ab Adresse $8000 angesprochen werden, so wäre die Routine etwas zu verändern. Dieses Thema werden wir aber erst später behandeln. Nun wollen wir die vorher eingegebenen Daten auf den Bildschirm ausgeben.

```
LDX #$31
LDY #$00
JSR $303C
JSR $303C
JSR $303C
```

### Ansteuerung externer Geräte

OPEN, CLOSE, und CMD gibt es auch für Maschinensprache. Zur Parameterübergabe an den OPEN-Befehl bedarf es aber erst einiger vorbereitetender Routinen.

### SETLFS ($FFBA)

Mit SETLFS werden die logische Dateinummer, die Geräte- und die Sekundäradresse erfaßt. Wir laden die Dateinummer in den Akku, die Geräteadresse in das X-Register und die Sekundäradresse in das Y-Register.

Beispiel: OPEN1,4,7

```
LDA #$01
LDX #$04
LDY #$07
JSR $FFBA
```

Soll keine Sekundäradresse angegeben werden, so ist Y mit #$FF zu laden.

### SETNAM ($FFBD)

Um bei Kassetten- oder Diskettenoperation auch einen Dateinamen angeben zu können, existiert die SETNAM-Routine. Hierbei wird dem Akku die Länge des Dateinamens, dem X-Register das Low-Byte der Adresse, wo der Dateiname sich befindet, und dem Y-Register das Adreß-High-Byte mitgeteilt. Beim Drucker ist kein Dateiname erforderlich. In diesem Falle geben wir die Länge des Dateinamens mit Null an. Um die Adresse brauchen wir uns dann nicht mehr zu kümmern.

```
LDA #$00
JSR $FFBD
```

### OPEN ($FFC0)

Nachdem die Parameter durch SETLFS und SETNAM festgelegt sind, können diese durch OPEN in der File-Tabelle des Rechners vermerkt werden.

```
JSR $FFC0
```

Bei Floppy oder Datasette werden diese Geräte zusätzlich angesteuert, um entweder den entsprechenden File zu suchen oder anzulegen.

### CHKOUT ($FFC9)

Dieser Befehl entspricht dem CMD-Befehl, der uns bereits vom BASIC her bekannt ist. Nachdem ein entsprechender File mit OPEN eröffnet wurde, können wir das Ausgabegerät umdefinieren. Hierzu ist das X-Register mit der Kanalnummer zu laden und CHKOUT aufzurufen.

```
LDX #$01
JSR $FFC9
```

Daten, die jetzt mit BSOUT ausgegeben werden, erscheinen nun nicht mehr auf dem Bildschirm. Spätestens beim ersten Zeichen nach einem Return mit Code #$0D sollte der Drucker mit dem Druck beginnen.

```
LDA #$41          LDA #$42
JSR $FFD2         JSR $FFD2
LDA #$0D          LDA #$0D
JSR $FFD2         JSR $FFD2
```

Der Buchstabe A wird jetzt in jedem Fall auf dem Drucker ausgegeben. Wenn der Buchstabe B noch nicht erscheint, so tut er dies bestimmt, wenn der Druckerkanal wieder geschlossen wird.

### CLRCHN ($FFCC)

Um wieder die Tastatur als Eingabegerät und den Bildschirm als Ausgabegerät zu definieren, ist keine Parameterübergabe, sondern nur der Aufruf von CLRCHN nötig.

```
JSR $FFCC
LDA #$41
JSR $FFD2
```

Das Zeichen A erscheint wieder auf dem Bildschirm. Mit CHKOUT kann jederzeit wieder auf den Drucker umgeschaltet werden.

### CLOSE ($FFC3)

Wurde ein Kanal mit CLRCHN geschlossen, so können wir, sofern wir ihn nicht mehr benötigen sollten, ganz aus unserer File-Tabelle beseitigen. Hierzu ist der Akkumulator mit der Kanalnummer zu laden.

```
LDA #$01
JSR $FFC3
```

Mit CHKOUT auf den Drucker umschalten zu wollen, ist jetzt zwecklos. Denn der Rechner kennt die Geräteparameter nicht mehr. Erst nach erneutem Anlegen mit SETLFS, SETNAM und OPEN bekommen wir unseren Drucker wieder in den Griff. Der CLOSE-Befehl ist besonders wichtig bei Kassetten- und Diskoperationen, damit Floppy- und Datasettendatei ordentlich geschlossen werden.

### CLALL ($FFE7)

Ziemlich radikal ist diese Routine. Wie CLRCHN schließt sie den jeweiligen Gerätekanal. Zusätzlich beseitigt sie aber alle Einträge aus der Filetabelle. Wenn ein Gerät noch ordentlich mit CLOSE abgeschlossen werden sollte, gibt es Probleme, da die Geräteparameter und der Dateiname nicht mehr greifbar sind.

### CHKIN ($FFC6)

Wie CHKOUT einen Ausgabekanal öffnet, so tut dieses CHKIN mit einem Eingabekanal. Mit GETIN oder BASIN können die Daten dann vom entsprechenden Gerät eingelesen werden. Das Öffnen eines Eingabekanals hat keinen Einfluß auf den Ausgabekanal und umgekehrt. Daten können so von ei-

nem externen Gerät eingelesen und mit BSOUT gleich wieder auf das Ausgabegerät ausgegeben werden. Die Parameterübergabe erfolgt bei CHKIN ebenfalls über das X-Register.

### READST ($FFB7)

Von BASIC her ist uns die Statusvariable bereits bekannt. Wenn READST aufgerufen wird, bekommen wir die Statusvariable in den Akku. Daraus können wir ersehen, ob das Ende einer Datei erreicht ist, oder sonstige Fehler vorliegen.

### Ein- und Ausgabe-Experimente

Wenn unsere beiden Maschinenprogramme für Ein- und Ausgabe noch im Speicher vorliegen,so können wir ein wenig experimentieren.

### Datei für Drucker anlegen

```
LDA #$ 04
TAX
LDY #$07
JSR $FFBA
LDA #$00
JSR $FFBD
JSR $FFC0
```

### Daten erfassen

```
LDA #$31
LDY #$00
JSR $3023
Eingabe nach Wahl
JSR $3023
Eingabe nach Wahl
JSR $3023
Eingabe nach Wahl
```

### Daten auf Drucker ausgeben

```
LDX #$04
JSR $FFC9
LDX #$31
LDY #$00
JSR $303D
JSR $303D
JSR $303D
JSR $FFCC
```

### Dateiname erfassen

```
LDX #$38
LDY #$00
JSR $3023
TESTDATEI,S,W
```

### Diskettendatei eröffnen

```
LDA #$08
TAX
TAY
JSR $FFBA
LDA #$0D
LDX #$00
LDY #$38
JSR $FFBD
JSR $FFC0
```

### Auf Diskette speichern

```
LDX #$ 08
JSR FFC9
LDX #$31
LDY #$00
JSR $303D
JSR $303D
JSR $303D
JSR $FFCC
LDA #$08
JSR $FFC3
```

Da die Daten nun gespeichert sind, können wir in den Monitor gehen und sie löschen.

```
BRK
F3100,3FFF,0
M3100
G
```

Wir führen diese Operation durch, damit niemand sagen kann, wir würden jetzt die Daten aus dem Hauptspeicher lesen.

### Dateiname eingeben

```
LDX #$38
LDY #$00
JSR $3023
TESTDATE,S,R
```

### Diskettendatei eröffnen

```
LDA #$08
TAX
TAY
JSR $FFBA
LDA #$0D
LDX #$00
LDY #$38
JSR $FFBD
JSR $FFC0
```

### Daten einlesen und ausgeben

```
LDX #$08
JSR $FFC6
LDX #$31
LDY #$00
STX $00D3
STY $00D2
JSR $3023
JSR $3041
JSR $3023
JSR $3041
JSR $3023
JSR $3041
JSR $FFCC
LDA #$08
JSR $FFC3
JSR $FFE7
```

Ein Blick in den Monitor belehrt uns zusätzlich, daß unsere Daten nun auch wieder im Hauptspeicher vorhanden sind.

```
BRK
M3100
G
```

Damit sind wir schon fast am Ende unserer Einführung angelangt. Das Speichern und Laden von Maschinenprogrammen und das Wandeln in DATA-Zeilen sollen noch kurz abgehandelt werden.

### Laden und Speichern von Maschinenprogrammen

Das Speichern geschieht am besten vom Monitor aus. Wir speichern den Bereich ab, in dem das Maschinenprogramm steht. Zuerst ist der Buchstabe S für Save einzugeben. Diesem folgt zwischen Anführungszeichen der Filename. Durch Komma getrennt geben wir die Gerätenummer ein. Nach einem weiteren Komma folgt die Anfangsadresse und zuallerletzt, ebenfalls durch Komma getrennt, die Endadresse. Die Endadresse muß um eine Speicherstelle höher angegeben werden. Der folgende Befehl würde so zum Beispiel den Bereich von $3000 bis $30FF auf Diskette sichern.

```
S"TESTFILE",8,3000,3100
```

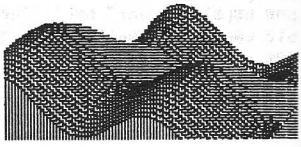
Beim Laden erübrigen sich die Adreßangaben.

```
L"TESTFILE",8
```

Bei der Kassette ist anstelle einer Acht die Eins als Gerätenummer zu verwenden.

### Wandeln in DATA-Zeilen

Für kurze Maschinenroutinen, die in ein BASIC-Programm eingebunden werden sollen, empfiehlt sich die Umwandlung in DATA-Zeilen. Die DATA werden vom BASIC-Programm wieder in die ursprünglichen Speicherstellen gePOKEd. Ein kurzes Programm *Datawandler* besorgt die Umwandlung. Nach dem Start ist die Zeilennummer einzugeben, ab der die DATA abgelegt werden sollen. Danach will das Programm Anfang und Ende des umzuwandelnden Speicherbereiches in hexadezimaler Form wissen. Die erzeugten Zeilen werden nur auf dem Bildschirm ausgegeben. Erst wenn Sie mit dem Cursor auf die zu sehenden Zeilen fahren und sie mit Return aufnehmen, stehen diese im Programm zur Verfügung. Mit dem DELETE-Befehl brauchen Sie nur noch die Programmzeilen des Datawandlers zu löschen. *AM* □



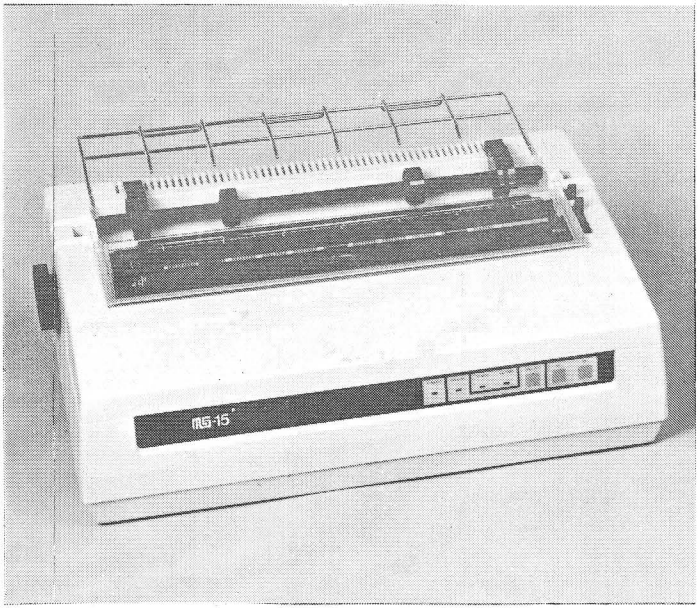
```
10 rem cpu-trainer=============c16 <de>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by alfons mittelmeyer   <cg>
50 rem                             <pd>
60 rem                             <ah>
70 rem basic v3.5                  <nl>
80 rem c16/116/plus4               <ki>
90 rem ========================== <jg>
100 poke55,0:poke56,48:clr         <bm>
110 fori=1630to1746                <jp>
120 reada:pokei,a:next             <ag>
130 data 173,214,006,072,173,211   <fn>
140 data 006,174,212,006,172,213   <li>
150 data 006,040,076,050,003,141   <hk>
160 data 211,006,142,212,006,140   <db>
170 data 213,006,008,104,141,214   <if>
180 data 006,088,216,096,173,211   <om>
190 data 006,174,215,006,157,000   <jp>
200 data 001,206,215,006,096,238   <el>
210 data 215,006,174,215,006,173   <ef>
220 data 214,006,072,040,189,000   <fd>
230 data 001,141,211,006,008,104   <no>
240 data 141,214,006,088,216,096   <oi>
250 data 173,214,006,024,144,215   <dm>
260 data 238,215,006,174,215,006   <kd>
270 data 189,000,001,141,214,006   <nn>
280 data 096,173,214,006,072,040   <jd>
290 data 173,215,006,141,212,006   <pb>
300 data 008,104,141,214,006,088   <ni>
310 data 024,096,173,212,006,141   <ik>
320 data 215,006,096               <no>
330 sys1647:poke1751,127           <nj>
340 rem --------------------------- <ip>
350 rem  variablen                 <oo>
360 rem --------------------------- <gg>
370 b$=chr$(32):b4$=b$+b$+b$+b$    <cg>
380 a$(0)="aiiiiiiiib"             <cd>
390 a$(1)="j00000000j"             <ff>
400 a$(2)="diiiiieiif"             <kh>
410 a$(3)="000000j00j"             <dl>
420 a$(4)="00000agiif"             <eh>
430 a$(5)="00000j000j"             <en>
440 a$(6)="00000diiic"             <bc>
450 data 032,176,174,189,173       <hf>
460 data 178,179,177,171,192,221   <fl>
470 g$="":fori=0to10:reada:g$=g$+c
hr$(a):next                        <il>
480 forj=0to6:fori=1to10           <oe>
490 mid$(a$(j),i,1)=mid$(g$,((asc(
mid$(a$(j),i,1))and15))+1,1):next:
nextj                              <pp>
500 b$(0)="binaer":b$(1)="hex"     <cp>
510 b$(2)="dezimal"                <gd>
520 c$="akku"+b4$+"x-register y-re
gister"                            <le>
530 d$="nv--dizc"+b4$+"stapel"     <fh>
540 rem --------------------------- <eg>
550 rem  bildschirmaufbau          <mp>
560 rem --------------------------- <eh>
570 char,0,0,"":scnclr             <jo>
580 gosub910:y=3:gosub900          <ll>
590 char,10,1,c$:x=7:y=2:gosub890  <kk>
600 x=18:gosub890:x=29:gosub890    <mn>
610 y=12:gosub900                  <ea>
620 char,8,10,d$:x=7:y=11:gosub890 <of>
630 x=18:gosub890:char,0,18,"":gos
ub910                              <op>
640 x$="":gosub950                 <ml>
650 rem --------------------------- <fi>
660 rem  eingabe                   <mc>
670 rem --------------------------- <mk>
680 ifx$="brk"then570              <ig>
690 char,0,19,chr$(27)+"q? "+chr$(
164)                               <ji>
700 y$=" ":x=1                     <id>
710 sys60381:a=peek(2034):if(a<32a
nda<>13anda<>20)ora>95then710      <le>
720 ifa=13then800                  <fj>
730 ifa<>20theny$=y$+chr$(a):x=x+1
:goto760                           <hk>
740 ifx=1then700                   <an>
750 x=x-1:y$=left$(y$,x)           <ja>
760 poke2034,157:sys56393:poke2034
,a:sys56393:poke203,0:poke2034,164
:sys56393:goto710                  <ga>
770 rem --------------------------- <ob>
780 rem  eingabe bearbeiten        <mg>
790 rem --------------------------- <io>
800 ify$<>" "thenx$=right$(y$,x-1) <kh>
810 poke2034,13:sys56393           <oi>
820 ifx$="brk"thenscnclr           <hp>
830 poke2034,27:sys56393:poke2034,
84:sys56393                        <pg>
840 poke2034,147:sys56393          <jc>
850 gosub1180:goto680              <gj>
860 rem --------------------------- <ie>
870 rem  routinensammlung          <pe>
880 rem --------------------------- <ad>
890 fori=0to6:char,x,y+i,a$(i):nex
t:return                           <pc>
900 fori=0to2:char,0,y+i+i,b$(i):n
ext:return                         <lb>
910 poke2034,61:fori=1to40:sys5639
3:next:return                      <ch>
920 data 1747,7,2,1748,18,2        <dl>
930 data 1749,29,2,1750,7,11       <hk>
940 data 1751,18,11                <gh>
950 ifx$="brk"thenreturn           <fj>
960 restore920:forz=1to5:reada     <bd>
970 readx:ready:a=peek(a)          <nf>
980 char,x+1,y+1,"":s=a:n=128      <mb>
990 fori=1to8:a$="0"               <ma>
1000 ifs>=nthens=s-n:a$="1"        <nd>
1010 char,x+i,y+1,a$:n=n/2:next    <bl>
1020 char,x+7,y+3,right$(hex$(a),2
```

```
)                                          <ep>
1030 char,x+6,y+5,right$(" "+str$(
a),3)                                      <dl>
1040 nextz:return                          <mb>
1050 x=-1:fori=1tom:reada$:reada:i
fa$=y$theni=50                             <em>
1060 next:return                           <en>
1070 char,0,19,"fehler":poke239,0:
wait239,1:return                           <ja>
1080 rem ------------------------          <fh>
1090 rem  ein-byte-befehle                 <bm>
1100 rem ------------------------          <om>
1110 data asl,10,brk,0,clc,24              <ab>
1120 data cld,216,cli,88,clv,184           <kg>
1130 data dex,202,dey,136,inx,232          <lf>
1140 data iny,200,lsr,74,nop,234           <nn>
1150 data rol,42,ror,106,sec,56            <if>
1160 data sed,248,sei,120,tax,170          <pg>
1170 data tay,168,txa,138,tya,152          <cg>
1180 iflen(x$)>4then1340                   <pg>
1190 y$=x$:restore1110:m=21:gosub1
050                                        <gc>
1200 ifi=51thenpoke1644,a:poke1645
,234:poke1646,234:sys1630:goto950          <mh>
1210 rem ------------------------          <pj>
1220 rem  sonderbehandlung                 <mg>
1230 rem ------------------------          <ib>
1240 m=6:gosub1050:ifi<>51then1070         <dh>
1250 sysa:goto950                          <ei>
1260 data pha,1664,pla,1677,php,17
02,plp,1708,tsx,1721,txs,1740              <kb>
1270 rem ------------------------          <le>
1280 rem  zwei-byte-befehle                <lk>
1290 rem ------------------------          <ii>
1300 data adc,105,and,41,cmp,201           <go>
1310 data cpx,224,cpy,192,eor,73           <lp>
1320 data lda,169,ldx,162,ldy,160          <kc>
1330 data ora,9,sbc,233                    <gf>
1340 ifleft$(right$(x$,4),2)<>"#$"
then1510                                   <on>
1350 restore1300:y$=left$(x$,3)            <ig>
1360 m=11:gosub1050:ifi<>51then107
0                                          <mc>
1370 x=dec(right$(x$,2))                   <jg>
1380 poke1644,a:poke1646,234               <bk>
1390 poke1645,x:sys1630:goto950            <ah>
1400 rem ------------------------          <ea>
1410 rem  drei-byte-befehle                <gp>
1420 rem ------------------------          <jc>
1430 data adc,109,and,45,asl,14            <eh>
1440 data bit,44,cmp,205,dec,206           <nc>
1450 data eor,77,inc,238,jsr,32            <fh>
1460 data lda,173,ldx,174,ldy,172          <he>
1470 data lsr,78,ora,13,rol,46             <in>
1480 data ror,110,sbc,237,sta,141          <hg>
1490 data stx,142,sty,140,cpx,224          <ll>
1500 data cpy,192                          <em>
1510 ifleft$(right$(x$,5),1)<>"$"t
hen1070                                    <ba>
1520 restore1430:y$=left$(x$,3)            <if>
1530 m=22:gosub1050:ifi<>51then107
0                                          <cg>
1540 y$=right$(x$,4)                       <oh>
1550 poke1646,dec(left$(y$,2))             <fn>
1560 poke1645,dec(right$(y$,2))            <in>
1570 poke1644,a                            <gn>
1580 sys1630:goto950                       <gi>
1590 rem ========================          <mi>
1600 rem 12277 bytes memory                <ph>
1610 rem 04414 bytes program               <ch>
1620 rem 00105 bytes variables             <cg>
1630 rem 00080 bytes arrays                <fb>
1640 rem 00451 bytes strings               <ih>
1650 rem 03087 bytes free (0)              <mh>
1660 rem 04096 bytes reserviert            <nn>
1670 rem ========================          <jd>
```

## DATAWANDLER

```
1 rem datawandler==========c16             <pa>
2 rem (p) commodore welt team              <lm>
3 rem ========================             <cd>
4 rem (c) by alfons mittelmeyer            <ce>
5 rem                                      <jj>
6 rem                                      <jl>
7 rem basic v3.5                           <ni>
8 rem c16/116/plus4                        <jk>
9 rem ========================             <nd>
10 input"zeilennummer";zn                  <cg>
11 input "anfang,ende";x$,y$               <pf>
12 x=dec(x$):y=dec(y$)                     <mf>
13 x$=str$(x):y$=str$(y)                   <hc>
14 x$=right$(x$,len(x$)-1)                 <ca>
15 y$=right$(y$,len(y$)-1)                 <ng>
16 gosub29:print" fori="x$"to"y$:z
n=zn+10                                    <jc>
17 gosub29:print" reada:pokei,a:ne
xt"                                        <ac>
18 zn=zn+10                                <bf>
19 l=int((y-x+1)/6)+1:k=(y-x+1)-(l
-1)*6                                      <dm>
20 ifk=0thenl=l-1:k=6                      <hh>
21 forj=1tol:n=6:ifj=lthenn=k              <ip>
22 gosub29:print" data ";                  <lh>
23 for i=0 to n-1:a=peek(x+i)              <kj>
24 a$=str$(a):a$=right$(a$,len(a$)
-1)                                        <ef>
25 a$=right$("00"+a$,3)                    <cc>
26 print a$",";:next:printchr$(20)         <li>
27 x=x+6:zn=zn+10:next                     <ec>
28 end                                     <li>
29 printright$(str$(zn),len(str$(z
n))-1);                                    <mi>
30 return                                  <mk>
31 rem ========================           <kh>
32 rem  p r o g r a m m e n d e            <mb>
33 rem ========================           <md>
```

# Mit zehn Mark sind Sie dabei: Drucken mit jedem Modell




**Man sollte es eigentlich kaum glauben. Auf dem Markt werden Interfaces für 200 bis 300 DM angeboten, doch mit ein bißchen Material, das nicht einmal 10 DM kostet, und einem kleinen Programm können wir uns bereits ein Centronics-Interface für unseren C16/116/Plus4 basteln.**

Ursprünglich wollten wir eigentlich die Centronics-Schnittstelle am Kassettenport realisieren. Wir dachten, daß der Kassettenport mit dem seriellen Port nichts zu tun hätte, weil im Betriebssystem des C16 eine Routine „Kassettenport einschalten" existiert.
Bei näherem Hinsehen jedoch merkten wir, daß hier gar keine Umschaltung auf den Kassettenport vorlag, sondern lediglich der Bildschirm ausgeschaltet und ein Timer gesetzt wurde. Durch Aufruf dieser Routine fand also keine elektrische Trennung von seriellem Port und Kassettenport statt. Bit Nummer eins in Adresse eins steuert also sowohl den Clock-Ausgang des seriellen Ports und die Schreib-Leitung des Kassettenports. Damit würde die serielle Ausgabe von Daten auf die Floppy 1541 also über unser geplantes Interface auch den Centronics-Drukker ansprechen, was wir nur allzugerne vermieden hätten. Da dieses sich leider nun sowieso nicht vermeiden läßt, wählten wir nun den seriellen Port für unser Interface aus, da wir hier nicht nur eine, sondern gleich drei Schreibleitungen zur Verfügung haben, was die Schaltung und vor allem die Programmlogik wesentlich vereinfacht.
Der Artikel über das Schieberegister zeigt uns bereits, wie wir mit nur drei Ausgabeleitungen acht Datenleitungen und eine Strobeleitung an den Drucker realisieren können. Für die Busy-Leitung hätten wir nun zwei Lösungen, nämlich doch noch den Kassettenport zu benützen oder aber eine unserer drei Leitungen am seriellen Port, mit denen wir nicht nur Daten ausgeben, sondern auch Daten lesen können, noch zusätzlich zur Abfrage des Busy-Signals zu verwenden.

## ABFRAGEN DER BUSY-LEITUNG

Daß wir zur Erzeugung von acht parallelen Datenbits ein Schieberegister mit seriellem Eingang und parallelem Ausgang benötigen, dürfte wohl nicht all zu schwer zu erraten sein. Was noch ein Problem darstellt, ist, wie wir wohl das Busy-Signal auf eine Datenleitung bekommen, ohne daß durch dieses unsere auszugebenden Daten beeinflußt werden können. Der Umstand, daß das Busy-Signal bei der Datenausgabe immer auf Low gesetzt ist, und erst nach dem Zurücksetzen des Strobe-Signals den Wert High anzunehmen pflegt, kommt uns hierbei sehr gelegen. Mit einer Diode, die den Strom nur in einer Richtung durchläßt, könnten wir auf billige Weise den Einfluß des Strobe-Signals auf unsere Daten ausschalten. Die Diode allein nutzt uns aber noch nicht allzu viel. Sehen wir uns den Aufbau unseres seriellen Ports etwas näher an, so kommen wir sehr schnell darauf, daß noch ein weiterer elektronisches Bauteil vonnöten ist. Die Datenleitungen des seriellen Ports liegen über einen Pull-Up-Widerstand auf High, sofern durch entsprechen-

des Poken in Adresse eins die Ausgänge hochohmig sind. Durch Setzen der Bits null bis zwei auf den Wert eins wird dieser Zustand erreicht. Die Bits fünf bis sieben, mit Hilfe derer wir die anliegende Spannung lesen können, geben uns darüber Aufschluß. Es ist hierbei zu bemerken, daß Bit sieben mit Bit null korrespondiert, Bit sechs mit Bit eins und Bit fünf mit Bit zwei, wobei Bit null und sieben für die Daten-Leitung, Bit eins und sechs für die Clock-Leitung und Bit zwei und fünf für die ATN-Leitung zuständig sind. Die Low-Setzung einer Ausgabeleitung durch Setzen einer Eins in den entsprechenden

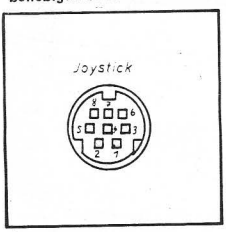
## EXTERNE 5-VOLT-SPANNUNGS-VERSORGUNG

Bit oder das Anlegen von Low durch eine externe Verbindung sorgt dafür, daß die Spannung, die an den einem Ende des ein Kilo-Ohm großen Widerstandes noch fünf Volt betragen hatte, am anderen Ende auf null Volt heruntergezogen werden kann. Zum Lesen eines Signales ist somit der Ausgang hochohmig zu machen, und abzufragen, ob ein externes Signal eine Änderung des High-Zustandes bewirke oder nicht. Da aber das Busy-Signal, durch eine Diode gesperrt, nur die Werte hochohmig oder High annehmen kann, und somit immer der Wert eins gelesen würde, ist mit den erwähnten Bausteinen das Lesen des Busy-Signales leider noch nicht möglich. Wenn das Busy-Signal sich allerdings genau umgekehrt verhielte, nämlich die Werte Low oder hochohmig annehmen könnte, so bräuchten wir nur die Diode umzupolen, mit Sperrichtung zum Drukker und das Problem wäre gelöst. Würde der Drukker nach Zurücksetzen des Strobe-Signales das Busy auf Low setzen, so würde der Strom nun vom

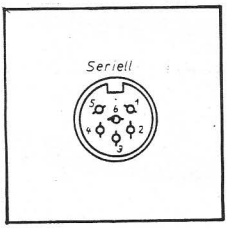
**Anschlußtabelle**

| Port | 74LS64 | Diode | Inverter | Drucker |
|---|---|---|---|---|
| 4 | 1 | | | |
| 4 | 2 | | | |
| | 3 | | | 2 |
| | 4 | | | 3 |
| | 5 | | | 4 |
| | 6 | | | 5 |
| 2 | 7 | | 7 | 16 |
| 5 | 8 | | | |
| | 9 | | | (18) |
| | 10 | | | 6 |
| | 11 | | | 7 |
| | 12 | | | 8 |
| | 13 | | | 9 |
| | 14 | | 14 | (18) |
| 4 | | >I | | |
| | | >I | 2 | |
| | | | 1 | 1 |
| 3 | | | | 11 |

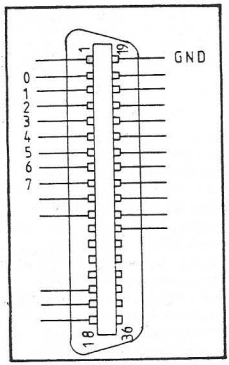
**Pinbelegung des seriellen Ports des Rechners: Über den 74LS164 TTL-Baustein, Diode, Inverter zu einem beliebigen Centronics-Drucker.**



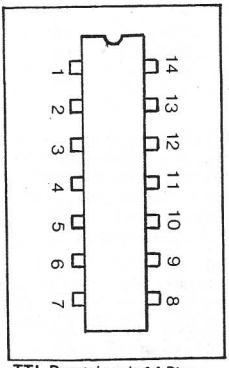

**Joystick Port PIN 5**




**Serieller Port PIN 2/3/4/5**




**Centronicsbelegung**




**TTL-Baustein mit 14 Pins**

**Was Sie hierzu brauchen:**
**Elektronische Bauteile für ca. 2 DM, Stecker und Platine für ca. 5 DM, etwas Kabel und Geduld beim Löten**

Rechner zum Drucker fließen, somit die Spannung an unserem Eingang abfallen, und wir könnten diesen Zustand durch Abfragen des entsprechenden Bits wahrnehmen. Ginge das Busy-Signal im Anschluß auf High zurück, so bemerkten wir auch dieses wieder durch unsere Abfrage. Da von High zu High sowieso kein Strom fließt, und der Stromfluß von High nach Low durch die Diode unterbunden ist, hätte im Anschluß das Strobe-Signal keinen Einfluß mehr auf an das Schieberegister angelegte Spannungen. Was wir also nur zu tun hätten, wäre das Umdrehen oder Invertieren der Busy-Leitung. Hierzu aber gibt es auch ein elektronisches Bauteil, nämlich den Inverter, als letztes, uns noch fehlendes, Glied zur Realisierung unserer Schaltung.

## ELEKTRONISCHE BAUTEILE

Die elektronischen Bauteile bekommen wir im Elektronik-Fachgeschäft. Falls an Ihrem Wohnort keines vorhanden sein sollte, so erfahren Sie sicherlich in einer Elektronik-Fachzeitschrift die Adresse eines Elektronik-Versandgeschäftes. Was Sie beim Kauf oder bei der Bestellung der Bausteine wissen müssen, ist deren Bezeichnung. Als serielles Schieberegister mit serieller Eingabe und paralleler Ausgabe kommt in Betracht der TTL-Baustein 74LS164. Eine schnelle niederohmige Diode mit geringem Spannungsabfall ist die Germaniumdiode AA112. Zum Invertieren des Busy-Signales können wir einen Inverter, zum Beispiel den TTL-Baustein 74LS04 oder auch ein NAND-Gatter zum Beispiel den TTL-Baustein 74LS03 verwenden. Inverter wie auch NAND-Gatter war in unserem Elektronik-Geschäft für

0,50 DM zu bekommen, die Diode für 0,49 DM und das Schieberegister für 1,10 DM. Für ca. 2,10 DM haben wir also unsere Elektronik beieinander. Eine gelochte Platine ohne Leiterbahnen ist für 0,60 DM zu haben, der Centronics-Stecker für 2,95 DM und ein sechspoliger Diodenstecker wird wohl auch nur 1,25 bis 1,95 DM kosten.
Wenn wir nun nur noch unsere Kabel richtig verlöten, so ist unser Interface schon fertig.
Die Zahlen in der Tabelle geben die Nummern der Anschlußpins wieder. Wo die einzelnen Pins sitzen, können wir den nachfolgenden Bildern entnehmen.
Hierbei ist zu beachten, daß wir die Anschlußbuchse am Computer so wiedergegeben haben, wie sie sich dem Auge von außen darbietet, den Centronics-Stecker dagegen ist von hinten gesehen, wo wir unsere Leitungen anzulöten haben.
Die Dateneingänge des Schieberegisters liegen auf Pin eins und zwei. Da diese Anschlüsse intern an einem AND-Gatter anliegen, ist die vom Rechner kommende Leitung mit beiden Pins zugleich zu verbinden. Wir haben die Clock-Leitung an Pin vier des seriellen Ports als Datenleitung auserwählt, die Data-Leitung desselben Ports an Pin fünf dagegen als Clock-Leitung für das Schieberegister, da andersherum Probleme auftraten. Den Clock-Anschluß des Schieberegisters ist auf Pin acht zu finden. Als Strobe-Signal für den Drucker, das am Centronics-Stecker an Pin elf anzulegen ist, wählten wir das ATN-Signal des seriellen Ports an Pin drei. Die parallelen Ausgänge des Schieberegisters, Pins drei bis sechs und Pins 10 bis 13 sind an die Dateneingänge des Centronics-Steckers Pins zwei bis neun zu legen. Das Busy-Signal des Druckers auf Pin eins wird zum Eingang des Inverters, ebenfalls Pin eins geführt. Bei Verwendung des NAND-Gatters ist das Busy-Signal auf beide Eingänge des Gatters, Pin eins und zwei zu legen. Der Ausgang, Pin zwei beim Inverter bzw. Pin drei beim NAND-Gatter, kommt an das sperrende Ende der Diode, welches durch einen schwarzen Ring kenntlich gemacht ist. Das andere Ende der Diode verbinden wir mit einem unserer Schieberegistereingänge oder mit der Clock-Leitung des seriellen Ports, was auf dasselbe herauskommt. Da nun alle Signalleitungen verbunden sind, ist nur noch die Stromversorgung des Interface zu gewährleisten. Die Masse des Druckers, Pin 16, und die Masse des Rechners, Pin zwei, sind auf ein gemeinsames Potential zu bringen und mit den Pins sieben unserer zwei TTL-Bausteine zu verbinden. Die Versorgung mit 5 Volt ist etwas problematisch, denn nicht jeder Drucker hat die gewünschte Spannung auch wirklich an Pin 18 anliegen. Wer Lust und Laune hat, durch ein wenig Löten im Drucker, diesem abzuhelfen, kann dieses gerne tun. Anderenfalls gibt es auch noch eine zweite, leicht zu realisierende Möglichkeit. Da die Benützung zweier Joystickports sowieso nur in speziellen Spielprogrammen vorkommt, liegen zumindest einer davon, wenn nicht alle zwei völlig brach. Wir können also ohne weiteres dort unsere Spannung abgreifen. Wir benötigen dazu nur ein isoliertes Kupferkabel von der nötigen Stärke, daß wir das abisolierte Ende gut in Pin fünf eines unserer Joystickports stecken können. Wenn die Spannung also nicht vom Drucker zu bekommen ist, dann bitte Pin 14 der TTL-Bausteine nicht mit Pin 18 des Centronics-Steckers verbinden sondern mit Pin fünf einer Joystick-Buchse.

## CENTRONICS C16

```
10 rem centronics==============c16 <hl>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by alfons mittelmeyer   <cg>
50 rem                             <pd>
60 rem                             <ah>
70 rem c16/116/plus4               <ja>
80 rem + centronicsinterface       <gc>
90 rem ========================== <jg>
100 fori=1015to1067                <df>
110 reada:pokei,a:next             <ne>
120 fori=1630to1729                <gl>
130 reada:pokei,a:next             <gp>
140 fori=818to858                  <kp>
150 reada:pokei,a:next:sys818:new  <ki>
160 data 201,013,208,007,072,169   <el>
170 data 010,032,002,004,104,072   <le>
180 data 133,245,165,001,162,008   <lk>
190 data 009,003,006,245,144,002   <gn>
200 data 041,253,133,001,041,254   <aa>
210 data 133,001,202,208,239,009   <kp>
220 data 004,133,001,041,249,133   <ge>
230 data 001,036,001,080,252,024   <bk>
240 data 174,061,001,104,096       <im>
250 data 072,165,153,201,003,208   <ih>
260 data 089,165,173,201,116,208   <ok>
270 data 083,104,142,061,001,201   <cf>
280 data 017,208,002,024,096,170   <jh>
290 data 041,127,201,065,048,012   <kk>
300 data 201,091,016,008,224,000   <ni>
310 data 048,005,073,032,208,001   <fa>
320 data 138,174,061,001,076,247   <gg>
330 data 003,166,174,224,004,208   <pf>
340 data 008,162,003,134,174,162   <nd>
350 data 116,134,173,076,178,006   <ha>
360 data 166,173,224,116,208,004   <dm>
370 data 162,000,134,173,032,012   <ha>
380 data 239,162,000,134,173,096   <oi>
390 data 032,083,239,076,173,006   <ne>
400 data 032,093,238,076,173,006   <mi>
410 data 104,076,075,236           <ik>
420 data 169,160,141,032,003,169   <kc>
430 data 006,141,033,003,169,143   <gj>
440 data 141,024,003,169,006,141   <lb>
450 data 025,003,169,184,141,026   <bp>
460 data 003,169,006,141,027,003   <fe>
470 data 169,094,141,036,003,169   <ml>
480 data 006,141,037,003,096       <gf>
490 rem ========================== <pm>
500 rem  p r o g r a m m e n d e   <cp>
510 rem ========================== <af>
```

### C16-CENTRONICS DRUCKER-INTERFACE

### HAT IHR DRUCKER 5 VOLT AUF PIN 18?

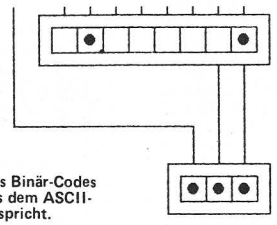
# Aus Zwei machAcht

**Mit einem Schieberegister werden aus zwei Datenleitungen acht. Ein Demoprogramm zeigt nicht nur die Funktionsweise einer solchen Datenübertragung auf, sondern ermöglicht zu Textzwecken gar die Ausgabe von Daten.**

Wer eine Datasette oder eine Floppy 1551 benutzt, dürfte mit diesem Interface gut bedient sein. Für eine Floppy 1541 allerdings, die ebenfalls am seriellen Port betrieben wird, ist diese Lösung kaum vertretbar, denn Floppy und Drucker würden sich gegenseitig ins Gehege kommen. Mit ein paar Bausteinen mehr könnte auch hier Abhilfe geschaffen werden. Aber das soll nicht der jetzige Stoff sein.

## SOFTWARE-TREIBER

Ausgaben über den seriellen Port laufen normalerweise völlig anders ab, als sie unser Interface benötigt. Es bleibt uns nichts anderes übrig, als das auszugebende Zeichen durch Verbiegen des Ausgabevektors BSOUT und eine geeignete Routine abzufangen und selbst die Ausgabe in geeigneter Weise zu veranlassen. Unser Programm führt eine ASCII-Umwandlung durch, damit Groß- und Kleinbuchstaben richtig ausgegeben werden, macht einen zusätzlichen Linefeed, damit auch diejenigen, die einen automatischen Linefeed am Drucker nicht einstellen können, keine Probleme damit haben und unterdrückt den ASCII-Code 17, womit Commodore-Drucker auf Groß-/Kleinschreibung umgestellt werden, andere Drucker aber zu unerwünschten Dingen, wie Einschalten irgendeines Grafik-Modus, veranlaßt würden. Wenn Sie unser Programm CENTRONICS-C16 gestartet haben, kann der Drucker auf Commodore-übliche Weise angesprochen werden. Commodore-spezifische Steuer- und Grafikzeichen allerdings sind von Ihnen durch die entsprechenden Drucker-spezifischen Zeichen zu ersetzen. Wir wünschen Ihnen viel Spaß mit Ihrem selbstgebauten Interface. *a.m.* □



**Darstellung des Binär-Codes 01000001, das dem ASCII-Zeichen A entspricht.**

Zehn Leitungen sind normalerweise nötig, um eine Centronics-Schnittstelle zu realisieren: nämlich acht Datenleitungen, die Strobe-Leitung, die dem Drucker das Bereitstehen von Daten signalisiert, und die Busy-Leitung, die uns die Bereitschaft zum Empfang weiterer Daten meldet. Sehen wir uns den seriellen Port unseres Rechners an, so entdekken wir nur drei Leitungen, die wir zum Senden und Empfangen benützen können.

## AUS ZWEI MACH ACHT

Das sind wohl entschieden zu wenig, könnte man meinen. Doch glücklicherweise gibt es elektronische Bauteile, die dieses Manko beseitigen können. Was wir benötigen, ist ein Baustein, den wir seriell Bit für Bit mit Daten beschicken können und der diese schön parallel nebeneinander für den Drucker bereithält. Also lediglich ein Acht-Bit-Schieberegister mit seriellem Eingang und parallelem Ausgang, das es für eine Mark in jedem gut sortierten Elektronik-Fachgeschäft zu kaufen geben sollte.

Zur Ansteuerung eines solchen Schieberegisters sind nur zwei Leitungen erforderlich. Zum ersten eine Datenleitung, mit der wir entweder ein High- oder Low-Signal, je nach dem Zustand des zu übertragenden Bit, am Dateneingang des Registers anlegen können. Zum zweiten eine Taktleitung, durch die wir mit einem kurzen Signal, dem sogenannten Clocksignal, die Datenaufnahme auslösen. Nachdem sich nach acht Übertragungen die acht Bit des zu übertragenen Byte im Schieberegister befinden, ziehen wir kurz die dritte Leitung, die Strobe-Leitung, auf Low, und der Drucker hat seine Daten. Wie wir die Busy-Leitung abfragen, soll uns im Moment noch nicht interessieren. Eventuell können wir hierzu entweder die Daten- oder die Clock-Leitung hernehmen.

## CENTRONICS-DEMOPROGRAMM

Da eine Demonstration oft mehr zu zeigen vermag, als viele Worte auszusagen versuchen, haben wir für Sie ein Demoprogramm verfaßt, welches wir jetzt starten wollen.
Unten am Bildschirm ist der Zustand unserer Leitungen durch einen gesetzten oder nichtgesetzten schwarzen Punkt symbolisiert. Die Strobe-Leitung links liegt normalerweise auf High. Die Clock- und Daten-Leitung liegen noch auf Low. Mit den beiden letztgenannten Leitungen steuern wir das Schieberegister, das noch gänzlich leer vor uns liegt.
Indem wir die Tasten [0] oder [1] drücken, legen wir unsere Daten an das Schieberegister an. Mit dem Druck der Leertaste lösen wir das Clock-Signal aus, das unser Datenbit in das Schieberegister bringt. Die nachfolgende Tabelle gibt uns Aufschluß über die Bit-Codes einiger ASCII-Zeichen.

| | 0010 | 0011 | 0100 | 0101 |
|---|---|---|---|---|
| 0000: | : | 0 : | @ : | P : |
| 0001: | ! : | 1 : | A : | Q : |
| 0010: | " : | 2 : | B : | R : |
| 0011: | # : | 3 : | C : | S : |
| 0100: | $ : | 4 : | D : | T : |
| 0101: | % : | 5 : | E : | U : |
| 0110: | & : | 6 : | F : | V : |
| 0111: | ' : | 7 : | G : | W : |
| 1000: | ( : | 8 : | H : | X : |
| 1001: | ) : | 9 : | I : | Y : |
| 1010: | * : | : : | J : | Z : |
| 1011: | + : | ; : | K : | [ : |
| 1100: | , : | < : | L : | £ : |
| 1101: | – : | = : | M : | J : |
| 1110: | . : | > : | N : | ^ : |
| 1111: | / : | ? : | O : | |

Über jeder Spalte befinden sich die vier ersten zu übertragenden Bit, am Anfang jeder Zeile die vier restlichen Bit. Nachdem das einem ASCII-Zeichen entsprechende Bitmuster im Schieberegister steht, löst ein Druck auf die Return-Taste das Strobe-Signal aus. Dadurch wird das Zeichen auf den Drukker, in unserem Demoprogramm hingegen in die erste Bildschirmzeile, ausgegeben.
Wenn Sie CW-EXTRA auf den Bildschirm zu bringen vermögen, so dürften Sie das Funktionsprinzip einer Centronics-Schnitt-

stelle am C16/116 schon voll erfaßt haben.

## AUTOMATIK-MODUS

Eine Möglichkeit, die unser Demoprogramm noch bietet, haben wir Ihnen bis jetzt noch verschwiegen. Mit der ESC-Taste können Sie in den Automatik-Modus um- und von dort auch wieder zurückschalten. Im Automatik-Modus können wir hintereinander bis zu zehn Tasten drücken, denn soviel faßt der Tastaturpuffer. Nach der Eingabe setzt der uns bereits vertraute Ablauf programmgesteuert ein, und bald darauf ist der Buchstabe auf dem Bildschirm sichtbar.

Als wertvolle Hilfe beim Bau des Interface erwies sich die im Centronics-Demoprogramm vorgesehene Möglichkeit, die Daten auch wirklich am seriellen Port auszugeben. Hierzu sind nur die entsprechenden REMs, die die Ausgabe normalerweise verhindern, zu beseitigen.
Falls wir im Besitz eines Centronics-Schnittstellen-Testers sind, können wir an dessen leuchtenden Lämpchen ablesen, ob das Interface auch wirklich das tut, was auf dem Bildschirm zu sehen ist. Vollführt es das Erwartete, so steht nichts mehr im Wege, Daten an den Drucker zu senden.

*a.m.* □

## CENTRONICS-DEMO

```
10 rem centronics-demo ========c16 <ip>
20 rem (p) commodore welt team    <ho>
30 rem ========================== <ng>
40 rem by alfons mittelmeyer      <nm>
50 rem                            <pd>
60 rem                            <ah>
70 rem basic v3.5                 <nl>
80 rem c16/116/plus4              <ki>
90 rem ========================== <jg>
100 scnclr:gosub630               <jf>
110 char,7,4,"B "+lo$+"C"+u$+"C"+u
$+"C"+u$+"C"+u$+"C"+u$+"C"+u$+"C"+
u$+"C"+u$+"C"+ro$                 <ka>
120 char,7,5,"B B"+lo$+"C"+o$+"C"+
o$+"C"+o$+"C"+o$+"C"+o$+"C"+o$+"C"
+o$+"C"+ro$+"B"                   <kn>
130 char,7,6,"B BB B B B B B B B B
B"                                <co>
140 char,7,7,"B B"+lu$+"C"+u$+"C"+
u$+"C"+u$+"C"+u$+"C"+u$+"C"+u$+"C"
+u$+"C"+ru$+"B"                   <oh>
150 char,7,8,"B "+lu$+"CCCCCCCCCCC
CC"+o$+"C"+o$+"C"+ru$             <ci>
160 for i=9to12:char,7,i,"B":char,
23,i,"B B":next                   <oh>
170 char,7,13,lu$+"CCCCCCCCCCCCC"+
ro$+" B B"                        <bm>
180 for i=14to15:char,21,i,"B B B"
:next                             <an>
190 char,19,16,lo$+"C"+u$+"C"+u$+"
C"+u$+"C"+ro$                     <lm>
200 char,19,17,"B"+lo$+"C"+o$+"C"+
o$+"C"+ro$+"B"                    <ok>
210 char,19,18,"BBQBQB BB"        <ke>
220 char,19,19,"B"+lu$+"C"+u$+"C"+
u$+"C"+ru$+"B"                    <hd>
230 char,19,20,lu$+"CCCCCCC"+ru$  <gp>
240 a$=" ":fori=1to7:a$=a$+1$+" ":
next                              <hf>
250 cl=0:da=0:sr=-1               <oh>
260 getkeyx$                      <bf>
270 ifx$="0"thenda=0              <oo>
280 ifx$="1"thenda=-1             <ec>
290 ifx$=" "thencl=-1             <bm>
300 ifx$=chr$(13)thensr=0         <dg>
310 ifx$=chr$(27)thengoto580      <ii>
320 gosub330:goto260              <me>
330 ifsrthen430                   <hc>
340 char,21,18," ":fori=1to500:nex
t:char,21,18,"Q"                  <fd>
350 b=0:fori=1to15step2:b=b+b     <dk>
360 ifmid$(a$,i,1)="Q"thenb=b+1   <cf>
370 next:sr=-1:fork=1to500:next   <on>
380 char,x,y,chr$(b):x=x+1        <nm>
390 rem ------------------------- <fj>
400 rem  strobe                   <pb>
410 rem ------------------------- <oa>
420 rem  poke1,peek(1)or4:poke1,pe
ek(1)and251                       <aj>
430 ifdathenda$="Q":elseda$=" "   <ei>
440 rem ------------------------- <hf>
450 rem  data                     <go>
460 rem ------------------------- <cj>
470 rem  ifdathenpoke1,peek(1)and2
53:elsepoke1,peek(1)or2           <nl>
480 ifnotclthen550                <no>
490 char,23,18," ":fori=1to500:nex
t:char,23,18,"Q":cl=0             <mk>
500 a$=right$(a$,13)+1$+da$       <gl>
510 rem ------------------------- <mb>
520 rem  clock                    <bf>
530 rem ------------------------- <ah>
540 rem poke1,peek(1)or1:poke1,pee
k(1)and254                        <jp>
550 char,25,18,da$                <kp>
560 char,11,6,a$                  <io>
570 return                        <ag>
580 getkeyz$:z=asc(z$):ifz=27then2
60                                <pg>
590 forj=1to8:z=z+z:fork=1to500:ne
xt                                <je>
600 da=0:ifz>=256thenda=-1:z=z-256 <ab>
610 gosub330:cl=-1:fork=1to500:nex
t:gosub330:next:fork=1to500:next  <eb>
620 sr=o:gosub330:goto580         <hp>
630 l$=chr$(29):lo$=chr$(176)     <eo>
640 ro$=chr$(174):lu$=chr$(173)   <pb>
650 ru$=chr$(189):o$=chr$(178)    <mb>
660 u$=chr$(177):return           <ok>
670 rem ========================= <fh>
680 rem  p r o g r a m m e n d e  <ji>
690 rem ========================= <cn>
```

# In Basic programmieren? Kein Problem!

**Immer wieder erreichen uns Leser-Zuschriften, die ein großes Interesse an Anwendungsprogrammen bekunden. Viele dieser Leser geben auch ohne große Umschweife zu, daß ihre BASIC-Kenntnisse noch nicht sehr umfangreich sind, vor allem, wenn sie den Computer erst vor kurzem gekauft haben. Wir haben uns nun gedacht, künftig diesem Teil unserer Leser ein wenig den Einstieg in meistverbreiteste Programmiersprache der Welt zu erleichtern.**

Anwendungsprogramme können sowohl professioneller als auch eigen-gestrickter Natur sein. Professionelle Programme können Sie kaufen und sofort einsetzen. Sie können aber auch versuchen, selbst welche zu schreiben. Da aber hakt es doch bei manchem, das Wissen reicht noch nicht ganz aus. Programmieren lernen Sie aber nicht nur durch Lesen von Fachbüchern und Zeitschriften, sondern vor allem durch Übung.

## WAS SOLL DENN PROGRAMMIERT WERDEN?

Dabei tritt anscheinend bei dem einem oder anderen ein Problem auf, nämlich die Frage: Was soll ich denn programmieren? Die Antwort darauf ist eigentlich recht einfach: Ein Programm, das Ihr Anwendungsproblem löst. Um Ihnen aufzuzeigen, wie Sie ein Problem lösen können, werden wir eines darstellen und Ihnen einen Lösungsweg bieten und selbstverständlich ein Programm hierfür bieten und besprechen.
Sicher haben auch Sie schon einmal darüber nachgedacht, wie denn chiffrierte Nachrichten decodiert werden können. Im Zeitalter der Computer sind für solche Arbeiten zwar auch noch Spezialisten erforderlich, aber ein Großteil der – oft sehr mühseligen – Arbeit kann durch Computer in sehr kurzer Zeit erledigt werden.
Bestimmt haben Sie auch schon von verschiedenen Methoden der Verschlüsselung gehört. Möglichkeiten gibt es viele. Eine vielleicht bereits bekannte Art ist das Verschlüsselungsquadrat.
Das bedeutet, der zu codierende Text wird beispielsweise senkrecht in ein Rasterfeld mit gleicher Spalten- und Zeilenanzahl eingegeben und waagrecht ausgelesen.
Selbst wenn Sie aber bei einem Text wissen, daß er nach dieser Methode verschlüsselt wurde, ist es „zu Fuß" trotzdem noch schwer, die Decodierung durchzuführen. Da muß der Computer ran.

## VERSUCHEN SIE ES ZUERST OHNE PROGRAMM!

In Zeile 130 des Listings sehen Sie eine „verschlüsselte" Mitteilung.
Versuchen Sie doch einmal, ohne Computerunterstützung herauszufinden, wie die Nachricht lautet. Wenn Sie es so schnell schaffen wie Ihr Commodore-Rechner mit Hilfe des abzutippenden Programmes, dann sind Sie absolut Spitze und könnten Ihren Computer eigentlich wieder verkaufen.

## VOM PROBLEM ZUR LÖSUNG

Nun wissen Sie also, worum es geht und haben die Aufgabe, ein Programm zu schreiben, das diesen verschlüsselten Text als Klartext ausgibt. Bevor Sie nun Ihren Computer einschalten und versuchen, wie ein Weltmeister zu programmieren, sollten Sie vielleicht erst einmal überlegen, wie ein Lösungsweg aussehen könnte.
Bei einem Quadrat haben wir es mit einer Fläche zu tun, deren vier Seitenlängen gleich sind. Denken Sie zum Beispiel an ein Schachbrett.
Die Buchstaben des Klartextes wurden so auf die Felder geschrieben, daß der Text – links oben beginnend – nach rechts läuft, so wie eben normalerweise geschrieben und gelesen wird. Ausgegeben wird der Text aber derart, daß von oben nach unten gelesen werden muß, um ihn zu verstehen.
Obwohl Sie den codierten Text kennen, wissen Sie nicht, welche Seitenlänge dieses Verschlüsselungsquadrat hat – oder doch? Wenn, wie in unserem Beispiel, eine sehr einfache Codierung erfolgt, kann Ihnen die Länge des Textes diese Information liefern. Bei einer quadratischen Anordnung muß der Text im Quadrat komplett enthalten sein. In diesem simplen Beispiel muß die Quadratzahl der Seitenlänge also gleich oder größer der Zeichenketten(= String)-Länge sein. Das heißt, die nächstmögliche, ganzzahlige Wurzel aus der Länge der verschlüsseiten Nachricht ergibt die Seitenlänge des Quadrates.

## ES GEHT NICHT OHNE MATHEMATIK

Kurz zur Erinnerung und Auffrischung: Die Wurzel einer Zahl ist der Wert, der mit sich selbst multipliziert eine Quadratzahl ergibt, zum Beispiel Wurzel aus 49 = 7, weil 7 mal 7 eben wieder 49 ergibt. Wenn Sie nun beispielsweise einen Text mit dreißig Zeichen Länge haben, dann paßt dieser in ein minimales Feld mit einer Seitenlänge von 6, denn eine mit nur 5 ergäbe als maximale Textlänge nur fünfundzwanzig Zeichen (5 * 5 = 25).

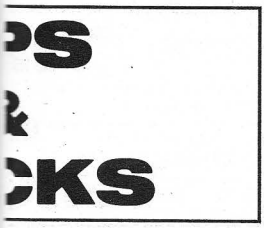
TI
&
TRI

Damit haben wir auch schon den Lösungsweg dieses Problems gefunden. Wir brauchen nun nur – beim ersten Buchstaben beginnend – den codierten Text vertikal (senkrecht) in ein gezeichnetes Quadrat einzutragen und horizontal (waagrecht) zu lesen. Aber wir wollen diese Aufgabe ja nicht selbst erledigen, sondern ein Programm entwerfen, das uns diese Arbeit abnimmt.

## DIE PROGRAMM-STRUKTUR

Zunächst einmal wird bei Programmstart der Bildschirm gelöscht (PRINT

CHR$(147)). Damit wir aber auch sehen, welchen Zweck unser kleines Programm erfüllen soll, geben wir eine entsprechende Mitteilung durch den PRINT-Befehl aus.
PRINT ist die einfachste Anweisung an den Computer, etwas auszugeben, im Normalfall auf dem Bildschirm. Wenn wir Text ausgegeben haben wollen, dann kann dies so geschehen, daß nach dem PRINT-Befehl der gewünschte Text in Anführungszeichen eingeschlossen steht, dann handelt es sich um einen Textstring. Hierfür ist in unserem Programm Zeile 100 zuständig.
Zeile 110 bewirkt die Ausgabe des Textes: „Waagrecht schreiben" und in der folgenden Zeile „Senkrecht auslesen".
Als nächstes wollen wir dem Computer mitteilen, welcher Text decodiert werden soll. Dies kann beispielsweise über eine



Zuweisung oder auch durch den INPUT-Befehl erfolgen (INPUT = Gib etwas ein!). Der INPUT-Befehl in seiner einfachsten Form benötigt aber einen Zusatz.
Der Computer muß sich ja unsere Eingabe vorübergehend merken können, deshalb gehört zum INPUT-Befehl noch die Zuweisung eines Variablen-Namens, unter dem Ihre Eingabe zwischengespeichert wird. Es können sowohl String- als auch reine Zahlen-Variablen sein.
Allerdings kennt Ihr Computer dabei nur ein Entweder–Oder: Haben Sie String-Variablen vorgesehen, so wird auch jede eingegebene Zahl als reine Zeichenkette (wie etwa Buchstaben) behandelt und interpretiert. Rechnen können Sie dann mit diesen Zahlen nicht, sondern sie nur wie einen Text auf dem Bildschirm ausgeben. So ein Variablenname wird immer mit einem davorgesetzten Semikolon (Strichpunkt) an die INPUT-Anweisung angehängt, in unserem Beispiel eben CO$.
Reine numerische Variablennamen werden ebenso angefügt, haben aber den Vorteil, daß die Eingabe nicht als Zeichenkette, sondern als im weiteren Programmverlauf als berechenbare Gleitkommazahl behandelt wird.
Allerdings dürfen Sie hier auch nur echte Zahlen eingeben, also keine anderen Zeichen wie Buchstaben, Interpunktions- oder Grafikzeichen, auch nicht mit einer Zahl gemischt (beispielsweise A1), das wird Ihr Computer immer grimmig mit einer Fehlermeldung quittieren.
Dann ist es für ihn nämlich eine Zeichenkette, ein String, und Sie haben ihn ja angewiesen, eine rein numerische Variable zu speichern, etwa INPUT"ZAHL:"; Z (also ohne $-Dollarzeichen!).
Verzeihen Sie uns die kleine Abschweifung zum Wesen des INPUT-Befehls, fahren wir in unserem Beispielprogramm fort.
Die Aufforderung zur Eingabe erfolgt – seitens des Computers – durch Ausgabe eines Fragezeichens (das mit ein paar Tricks auch unterdrückt werden kann) und des Cursorsymbols.
Bei Programmen, deren Erstellung schon längere Zeit zurückliegt, wissen wir nicht immer, was als Eingabe zu folgen hat. Eine kurze Mitteilung innerhalb des INPUT-Befehls ist daher unbedingt zu empfehlen.
Betrachten Sie dazu einmal Zeile 120. Allerdings haben wir zunächst diese Zeile durch das davorgesetzte REM außer Gefecht gesetzt. Alles, was nach so einer Anweisung in einer Programmzeile folgt, wird vom Computer ignoriert und nicht ausgeführt, sondern nur als Bemerkung betrachtet, auf Englisch REMARK, daher die Abkürzung REM.
Um Ihnen (und uns) die Arbeit zu erleichtern, haben wir in der folgenden Zeile 130 der Variablen CO$ die codierte Zeichenfolge direkt zugewiesen.
Falls nämlich noch Programmierfehler enthalten sein sollten, dann braucht nach deren Korrektur diese sehr fremdländisch klingende Zeichenfolge nicht jedesmal mühsam von Hand eingegeben zu werden.
Für spätere Abläufe, also wenn das Programm fertig ist und fehlerfrei läuft, können Sie vor die Anweisung CO$ = ... ein REM setzen und dafür das in Zeile 120 (vor der INPUT-Anweisung) entfernen.

## DURCH DIE ZEICHENKETTE WEISS DER COMPUTER ALLES NÖTIGE

Alles, was wir dem Computer an Informationen geben konnten, haben wir getan. Die Zeichenfolge ist ihm nun bekannt, die Länge dieses Strings kann er sich gefälligst selbst ausrechnen.
Die Anweisung dazu lautet LEN(String). LEN ist leicht zu merken (was übrigens für die meisten BASIC-Befehle gilt), wenn Sie ein bißchen Englisch können. Es wurde aus dem Wort length (=Länge) hergeleitet.
Die Befehlsfolge von LE = LEN(CO$) läßt sich am besten so übersetzen: Die Länge der Zeichenkette CO$ in der numerischen Variablen LE merken.
Nachdem der Computer die Länge berechnet hat, soll er die Wurzel ziehen, damit er die Seitenlänge des Quadrates weiß.
Die Wurzel (in diesem Fall die Quadratwurzel) wird durch den Befehl SQR(LE) gezogen. Da es aber keine halben Felder gibt, ist die Lösung nur dann richtig, wenn es sich bei der Wurzel um eine echte INTeger-Zahl handelt. Deshalb wird in Zeile 150 gleich festgelegt, daß für die Variable WU der Integer-Wert berechnet wird.
Wenn die Mitteilung aber nicht so lang ist, daß sie ein Quadrat ganz ausfüllt, dann sind noch die Spaces, die Leerstellen im Text. Dadurch würde sich über die Integerberechnung ein Fehler einschleichen.
Wir fangen dies in Zeile 160 ab, indem immer dann, wenn das Quadrat der Seitenlänge kleiner ist als die Anzahl der Zeichen, ein Leerzeichen angefügt und die Berechnung neu durchgeführt wird. Der Code fürs Leerzeichen ist 'CHR$(32)'.

## DIE WURZEL DARF NUR GANZZAHLIG SEIN

Zur Verdeutlichung des eben Gesagten: Die Länge der Mitteilung beträgt 63 Zeichen. Berechnen wir daraus die Quadratwurzel, dann ergibt sich ein „krummer" Wert von 7,937.
Dieser Wert kann in unserem Falle nicht die Seitenlänge des Quadrates sein, denn wir arbeiten ja nicht mit Bruchteilen von Buchstaben oder Zeichen. Deshalb die künstliche Verlängerung der Zeichenkette, eine neuerliche Berechnung der Wurzel und dann wieder ein Vergleich. Dies erfolgt so lange, bis der Wert für WU der Zahl 8 entspricht.
Nun ist die Seitenlänge bekannt. Was aber noch zu tun ist: Die einzelnen Buchstaben in korrekter Weise zusammenzufügen und auszugeben.
Zu diesem Zweck haben wir zwei Schleifen pro-

grammiert, die uns den jeweils aktuellen Buchstaben, im Programm EM$ genannt, herausfiltern. Dies passiert in den Zeilen 160 bis 210.
Dabei erledigen wir in diesen Zeilen gleich zweierlei: Einmal möchten wir den Klartext so ausgeben, wie er im ursprünglichen Quadrat stand, und zum anderen wollen wir ihn gleich in einen neuen String schreiben, der anschließend ausgegeben werden soll.
Wir wissen es und der Computer berechnet es aus der Länge; unser Beispieltext bestand aus einem 8 * 8-Quadrat. Deshalb brauchen wir nur, beim ersten Zeichen beginnend, jeweils immer das neunte (acht plus eins) Zeichen zu lesen, bis wir am Ende angelangt sind; dann wird der gleiche Vorgang wiederholt, nur diesmal beim zweiten Zeichen beginnend und so fort. Aber dafür haben wir ja das Programm entwickelt.

## DAS CODIERUNGS-PROGRAMM WIRD ENTSCHLÜSSELT

Die Lösung ist recht einfach: Eine Schleife (Zeile 160) zählt immer von 1 bis WU und die andere von 0 bis LE, mit der Schrittweite von WU. Die Schrittweite wird durch den Befehl STEP und den anschließenden numerischen Ausdruck definiert (STEP WU).
Die Zeile 180 isoliert nun den jeweilig korrekten Buchstaben aus der Gesamtzeichenkette. Denken wir das einmal theoretisch durch.
Beim ersten Durchlauf der Zeilen 160 bis 210 ist der Wert in der Variablen J = 1, der Wert in K = 0. Das heißt, in der Zeile 180 wird der Stringvariablen EM$ der erste Buchstabe (= 'C') zugewiesen. Dies können wir leicht feststellen, wenn wir die Variablennamen in Zeile 180 durch ihre Werte ersetzen. Daraus ergibt sich folgender Befehl:

EM$ = MID$( CO$,
1 + 0, 1)

MID$, in dieser Form angewandt, ordnet EM$ die Zeichenkette aus CO$ zu, die an der Position J + K (=1) beginnt und ein Zeichen lang ist ( ,1). Durch Zeile 190 wird die Zeichenkette für KL$ zusammengebaut. Beim ersten Durchlauf enthält KL$ deshalb das Zeichen C.

```
Codierter Text

10 rem codierter text                    <hp>
20 rem by lothar miedel                  <jl>
30 rem (c) commodore welt                <mi>
100 scnclr: print"loesung fuer ver
schluesselungsquadrat":print             <nl>
110 print"waagrecht schreiben,":pr
int"senkrecht auslesen!":print           <hd>
120 rem input"bitte text eingeben"
;co$                                     <em>
130 co$="cedsf n o-icucu mwehe1t m
e rr2z olzi 8e dtefd-r o-iteb  r t
 ne!"                                    <nf>
140 le=len(co$):wu=int(sqr(le))          <bf>
150 if(wu^2)<le thenco$=co$+chr$(3
2):goto140                               <ii>
160 forj=1towu                           <da>
170 fork=0tolestepwu                     <ob>
180 em$=mid$(co$,j+k,1)                  <kn>
190 kl$=kl$+em$                          <dg>
200 printem$"  ";                        <da>
210 nextk:print:nextj                    <mp>
220 print:print"klartext: ";kl$          <dk>
230 rem                                  <fm>
240 forj=1towu:fork=0tolestepwu          <fd>
250 em$=mid$(kl$,j+k,1):c1$=c1$+em
$:nextk,j                                <gj>
260 print:print"codierter text: "c
1$                                       <ih>
```

Durch Zeile 200 wird das jeweils aktuelle Zeichen und anschließend ein Leerzeichen ausgegeben. Das Semikolon am Ende dieser Zeile weist den Computer an, nun keinen Zeilenvorschub auszuführen, sondern in der gleichen Zeile zu bleiben.
Der Befehl NEXT K in Zeile 210 bewirkt, daß der Schleifenwert für K erhöht und dort weitergemacht wird, wo die eigentliche FOR-NEXT-Schleife begann.
Es wird deshalb in Zeile 170(!) fortgefahren. Diese bewirkt nun eine Erhöhung des Wertes für K um den Wert WU. Das heißt, K erhält eine Erhöhung um acht und hat nun den Wert 8. Wenn Sie die Zeile 180 nun wieder auflösen, dann steht dort:

EM $ = MID$ (CO$,
8 + 1, 1)

Das zweite Zeichen steht also nun an neunter Stelle und entspricht dem Buchstaben 0. Beim nächsten NEXT K wird wieder um acht erhöht und so weiter.
Wenn Sie sich die Mühe machen wollen, können Sie nun – Schritt für Schritt – den Programmablauf nachvollziehen.
Der Befehl PRINT in Zeile 210 bewirkt, daß bei der Ausgabe, nach jeweils acht Zeichen, ein Zeilenvorschub durchgeführt wird. Nach diesem wird dann der Wert für J erhöht. Da in der Schleife für J keine Schrittweite angegeben ist, wird jeweils um 1 erhöht.
Nach der ersten PRINT-Ausführung in Zeile 210 wird bei Zeile 160 weitergemacht. Der Wert für J ist dann 2. Gleichzeitig wird durch Zeile 170 der Wert für K wieder auf 0 gesetzt. Dadurch ergibt sich für Zeile 180 nun folgende Wertstellung:

EM $ = MID$ (Co$,
0 + 2, 1)

Das heißt, nun wird beim zweiten Buchstaben begonnen und die innere Schleife (wieder neu) abgearbeitet. Dadurch stehen die aktuellen Zeichen nun an den Positionen: 2, 10, 18, 26 usw. Diese Vorgänge wiederholen sich so lange, bis der komplette Text entschlüsselt, also sowohl in der 8 * 8-Matrix ausgegeben als auch in KL$ (=Klartext) zusammengebaut ist. In Zeile 220 wird dann der decodierte String ausgegeben.

## UND ALLES AUCH NOCH UMGEKEHRT

Die Zeile 240 bewirkt das Gegenteil, nämlich die Rück-Verschlüsselung des Klartextes (KL$) in die ursprüngliche Form. Deshalb kann dieses Programm auch zur Verschlüsselung eingesetzt werden. Es ändert sich dann lediglich immer der Sinn der Ausgaben. Aus Klartext wird codierter Text und umgekehrt.
Ein kurzer Hinweis bezüglich des Programmes sei uns noch gestattet: Bei der Ausgabe der Zeichenkette in der quadratischen Bildschirmdarstellung fehlt selbstverständlich eine Zeile (es sind nur sieben), denn der ursprüngliche Klartext war ja nur 50 Zeichen lang, der Rest waren Leerzeichen. *LM/hb*□

Megatool

# 31 neue Befehle

**Mit einer richtigen Syntax ausgestattet, bietet Mega-Tool eine Reihe nützlicher Funktionen an. So gibt es einen OLD-Befehl, einen Plot-Befehl und eine Merge-Routine, sogar Windows und Sprites sind spielend zu handhaben. Mega-Tool ist für jeden, dessen Rechner 64 KByte aufweist, eine wahre Bereicherung.**

MEGA-TOOL ist eine BASIC-Erweiterung für den Plus4 und den C16 mit 64 KByte. Es erweitert das ohnehin umfangreiche BASIC V3.5 um weitere 31 Befehle, die die Programmierung dieses Computers sehr erleichtern.

OLD
Holt ein versehentlich gelöschtes Programm wieder zurück. OLD hilft manchmal auch, ein verstümmeltes Listing zurückzuholen.

LLIST g,s[,zeilen]
g. . . . . Geräteadr.
s. . . . . Sekundäradr.
zeilen . . wie bei LIST
Gibt Listing am Drucker aus.

MERGE "name",g
Verkettet zwei BASIC-Programme, wobei die Zeilennummern des nachgeladenen Programms höher sein müssen als die des ersten.

DMERGE "name"
Wie MERGE,nur auf Diskette.

BLOCK "name",g
Lädt einen Speicherbereich, ähnlich wie LOAD"name",g,1, allerdings mit dem Vorteil, daß der Programmzähler nicht zurückgesetzt wird, das Programm also nicht wieder von vorne beginnt. BLOCK kann zum Beispiel zum Nachladen eines Zeichensatzes verwendet werden.

DBLOCK "name"
Wie BLOCK, nur auf Diskette.

SBLOCK "name",g,1,
anfang,ende+1
anfang. . . Anfangsadr.
ende. . . . Endadr.
Speichert einen bestimmten Speicherbereich auf Kassette oder Diskette ab. Kann zum Beispiel zum Abspeichern von Grafikseiten verwendet werden: SBLOCK "graphik",8,1,3072, 16384

DVERIFY "name"
Verifiziert auf Diskette.

ID
Initialisiert eine Diskette.

FAST
Schaltet den Bildschirm aus, wodurch der Computer um zirka 30 Prozent schneller arbeitet, da der TED-Chip nicht mehr auf das RAM zugreifen muß.

SLOW
Schaltet den Bildschirm wieder ein.

MULTI 0/1
Steuert das Multicolor-Bit des TED-Chips: MULTI 1 schaltet den Multicolor-Modus ein, MULTI 0 schaltet ihn wieder aus.

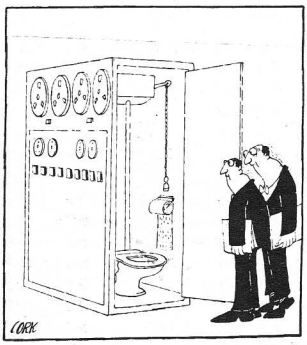
REVERS 0/1
Steuert das Revers-Flag des TED-Chips. Ist dieses Bit gesetzt, so erzeugt der TED keine reversen Zeichen und man kann auf den gesamten Zeichenvorrat des Computers zugreifen (statt revers Kleinschrift).

BIT adr,bit-#,0/1
adr. . . . . Speicheradr.
bit-# . . . Stellung des Bit (0–7)
Ermöglicht ein bitweises Poken.
Zum Beispiel entspricht BIT65298,2,0 dem Befehl POKE65298,PEEK (65298)AND255–4

TRANS 0/1
TRANS 1 kopiert den ROM-Zeichensatz ins RAM und zwar ab der Speicheradresse $E800, wo er verändert werden kann. Alle TED-Register werden auf diesen Zeichensatz umgestellt. Um zu verhindern, daß er von BASIC überschrieben wird, wird die Speichergrenze um ein KByte nach unten gesetzt. Dadurch werden alle Variablen gelöscht.

SPRITE nummer,x,y,col
nummer . . Spritenummer (0–15)
x,y. . . . . Koordinaten des Sprites
col. . . . . Farbe des Sprites (Bildschirmcode)
Der SPRITE-Befehl ist einer der mächtigsten Befehle dieser Erweiterung. Er dient zur Verwaltung

von 16 verschiedenen Pseudosprites, allerdings nur in Koppelung mit einem veränderten Zeichensatz.
Die Struktur der Sprites ist folgendermaßen:
Jedes Sprite besteht aus 2 mal 2 Zeichen.
Sprite Nummer 0 besteht aus den Zeichen 64, 65, 66 und 67, Sprite Nummer 1 aus 68, 69, 70 und 71 und so weiter, bis zu Sprite Nummer 15 aus den Zeichen 124, 125, 126 und 127.
Mit dem Zeichen 64 ist nicht das ASCII-Zeichen Nummer 64 (Klammeraffe) gemeint, sondern der Bildschirmcode Nummer 64, also das Sternchen mit SHIFT. Die vollständige Tabelle steht auf den Seiten 213 und 214 im Handbuch. Da fast alle Zeichensatz-Programme diesen Bildschirmcode verwenden (etwa der Graphic-Designer von Kingsoft), dürften sich damit kaum Probleme ergeben.
Das erste Zeichen des Sprites befindet sich links oben, das zweite rechts oben, das dritte links unten und das vierte rechts unten.
Wem ein Sprite aus vier Zeichen zu klein ist, muß mehrere Sprites zu einem zusammenfassen.
Zur Anwendung des Befehls: X ist die Spalte, Y die Zeile, in welcher das Sprite erscheint.
COL ist die Farbe, wobei Werte über 127 ein Blinken hervorrufen.
COL kann man sich mit folgender Formel errechnen: Farbe + Luminanz * 16.
Mit dem SPRITE-Befehl können Sprites neu gesetzt und bewegt werden. Allerdings sollte man es vermeiden, Sprites übereinander zu legen, da dies bei falscher Anwendung (Bewegen des darunterliegenden Sprites) unerwünschte Folgen haben könnte.
Der Status eines Sprites wird mit der USR-Funktion abgefragt: Ist der Wert größer als 128, so ist das Sprite am Bildschirm. Ist das sechste Bit gesetzt, so kollidierte es mit dem Untergrund: Es traf auf irgend ein anderes Zeichen als das Leerzeichen; andere Sprites gelten nicht als Hindergrund. Sollte das Sprite mit einem anderen kollidiert sein, so liefert USR(Spritenummer) AND 15 dessen Nummer.



DEL nummer
Löscht ein Sprite vom Bildschirm.

PLOT x,y
Setzt den Cursor an die Koordinaten x,y.

WINDOW x1,y1,x2,y2
x1,y1 .. linke obere Ecke
x2,y2 .. rechte untere Ecke
Definiert Fenster.

SCREEN x TO y
x .. Quellbildschirm
y .. Zielbildschirm
Der SCREEN-Befehl dient zum Verwalten der beiden RAM-Bildschirme.
Nummern:
0 . . .sichtbarer Bildschirm
1 . . .RAM-Bildschirm ab $EC00
2 . . .RAM-Bildschirm ab $F400
Mit SCREEN kann man den Bildschirm x in den Bildschirm y kopieren.
Als Anwendung wäre das Verwalten von Fenstern, wie beim Amiga, denkbar: Sichtbaren Bildschirm ins RAM kopieren, mit CHAR das Fenster bilden, zum Schließen des Fensters den Bildschirm wieder zurückkopieren. Ebenfalls möglich ist das Einblenden von Informationen wie Anleitungen oder Hi-Score-Tabellen.
Beim SCREEN-Befehl werden alle Bildschirmparameter wie Cursorposition, Revers-Flag, Hintergrund- und Rahmenfarbe mitkopiert.

HARDCOPY g,s
g,s . . . wie bei LLIST
Druckt den momentanen Bildschirm mit allen Grafikzeichen, sofern sie am Drucker vorhanden sind, aus.

INIT
Initialisiert Bildschirm, TED und Sprungvektoren, ohne ein Programm zu unterbrechen.

SWAP a$,b$
a$,b$ . . . beliebige Stringvariablen
Vertauscht zwei Stringvariablen ohne die Erzeugung von „Stringmüll".

SLEEP dauer
dauer . . . Zeit in Hundertstel Sekunden
SLEEP ersetzt die FOR-NEXT-Warteschleifen. Der Befehl wird durch den Raster-Interrupt gesteuert und kann durch die RUN/STOP-Taste unterbrochen werden.
Ein Tip: Mit SLEEP 1 vor dem SPRITE-Befehl wird, vor allem bei sehr schnellen Spielen, Bildschirmflimmern unmöglich gemacht, da nach einem SLEEP-Befehl immer ein neuer Bildschirmaufbau stattfindet.

```
MEGA-TOOL=======================c16
(p) COMMODORE WELT TEAM
==================================
(c) by Bernhard Oemer

C16/116/Plus4
==================================
Das Programm ist mit dem Maschinen-
sprachmonitor TEDMON unter Zuhilfe-
nahme des Pruefsummenprogrammes
CHECKMON einzugeben und auf Diskett-
te mit s"mega-tool",8,1001,17e8 ab-
zuspeichern. Auf Kassette bitte mit
s"manager",1,1001,17e8 abspeichern.
Geladen und gestartet wird wie bei
einem normalen Basic-Progamm.
==================================
>1000 00 0e 10 c3 07 9e 20 28 :<da>
>1008 34 31 31 32 29 00 00 00 :<9f>
>1010 a9 18 85 2c a9 00 8d 00 :<3a>
>1018 18 a9 ec 85 38 a2 05 bd :<d4>
>1020 96 10 9d 0c 03 ca 10 f7 :<c2>
>1028 20 81 ff 20 8a ff a9 7b :<5e>
>1030 8d 2a 03 a9 16 8d 2b 03 :<e2>
>1038 20 7b 16 a9 f0 8d 15 ff :<a8>
>1040 8d 19 ff 20 4f ff 0d 05 :<e7>
>1048 93 20 43 4f 4d 4d 4f 44 :<4b>
>1050 4f 52 45 20 42 41 53 49 :<7c>
>1058 43 20 56 33 2e 35 20 2b :<35>
>1060 20 4d 45 47 41 2d 54 4f :<08>
>1068 4f 4c 20 56 31 38 2e 30 :<6f>
>1070 0d 0d 20 20 33 30 20 4e :<61>
>1078 45 57 20 43 4f 4d 4d 41 :<10>
>1080 4e 44 53 20 41 4e 44 20 :<24>
>1088 00 20 e5 80 20 4f ff 0d :<2b>
>1090 0d 9e 00 4c 0a 80 9c 10 :<7b>
>1098 b0 10 d2 10 85 da a0 e1 :<c5>
>10a0 a9 10 20 07 8a 90 04 48 :<82>
>10a8 4c d6 89 a5 da 4c 6c 89 :<ea>
>10b0 aa a0 e1 84 22 a0 10 84 :<d6>
>10b8 23 a0 00 ca 10 10 b1 22 :<94>
>10c0 85 03 e6 22 d0 02 e6 23 :<0d>
>10c8 a5 03 10 f2 30 ed 18 4c :<e9>
>10d0 88 8b 29 7f 0a a8 b9 87 :<97>
>10d8 11 48 b9 86 11 48 4c 73 :<23>
>10e0 04 4f 4c c4 42 4c 4f 43 :<a2>
>10e8 cb 44 42 4c 4f 43 cb 4d :<f5>
>10f0 45 52 47 c5 44 4d 45 52 :<06>
>10f8 47 c5 53 50 52 49 54 c5 :<bf>
>1100 44 45 cc 46 41 53 d4 53 :<58>
>1108 4c 4f d7 4d 55 4c 54 c9 :<09>
>1110 52 45 56 45 52 d3 54 52 :<46>
>1118 41 4e d3 53 4c 45 45 d0 :<81>
>1120 50 4c 4f d4 57 49 4e 44 :<15>
>1128 4f d7 53 57 41 d0 42 49 :<26>
>1130 d4 52 45 53 45 d4 48 41 :<aa>
>1138 52 44 43 4f 50 d9 53 42 :<f1>
>1140 4c 4f 43 cb 4c 4c 49 53 :<e9>
>1148 d4 44 56 45 52 49 46 d9 :<a9>
>1150 53 43 52 45 45 ce 49 4e :<82>
>1158 49 d4 49 c4 57 41 49 54 :<ef>
>1160 4b 45 d9 42 55 46 46 45 :<1e>
>1168 d2 52 45 50 45 41 d4 52 :<28>
>1170 45 47 49 53 54 45 d2 44 :<7f>
>1178 49 53 cb 55 53 45 d2 00 :<48>
>1180 00 00 00 00 00 00 c3 11 :<4b>
>1188 ce 11 d4 11 e4 11 ea 11 :<4d>
>1190 16 12 a9 12 f5 12 fe 12 :<bd>
>1198 15 13 27 13 32 13 87 13 :<9f>
>11a0 cf 13 e7 13 18 14 43 14 :<4a>
>11a8 77 14 80 14 f7 14 0c 15 :<1f>
>11b0 37 15 52 15 e1 15 0c 16 :<44>
>11b8 27 16 58 16 86 16 0d 17 :<e1>
>11c0 26 17 ba 17 a9 01 a8 91 :<ef>
>11c8 2b 20 18 88 4c 4b 88 20 :<c2>
>11d0 4f 17 4c dc 11 20 56 17 :<39>
>11d8 a9 01 85 ad a2 ff a0 ff :<8a>
>11e0 a9 00 4c d5 ff 20 4f 17 :<a6>
>11e8 4c ee 11 20 56 17 a5 2b :<57>
>11f0 85 da a5 2c 85 db a5 2d :<44>
>11f8 a4 2e 38 e9 02 b0 01 88 :<e5>
>1200 aa a9 00 20 d5 ff a5 da :<d2>
>1208 85 2d a5 db 85 2c 20 c4 :<36>
>1210 11 20 18 88 4c dc 8b a9 :<9f>
>1218 08 8d 01 05 a9 13 8d 02 :<e0>
>1220 05 a9 00 85 dc 85 dd 20 :<64>
>1228 aa 12 20 d8 9d e0 27 b0 :<b7>
>1230 4d 86 db 20 d8 9d e0 18 :<47>
>1238 b0 44 86 d5 20 6a 17 20 :<7c>
>1240 d8 9d 86 d4 a5 da 0a 0a :<77>
>1248 aa a9 00 85 dc 85 dd a4 :<c2>
>1250 db 20 81 12 c8 20 81 12 :<0e>
>1258 98 18 69 27 a8 20 81 12 :<5d>
>1260 c8 20 81 12 a5 dc 09 80 :<45>
>1268 a6 dd f0 02 09 40 a6 da :<21>
>1270 9d 8b 17 a5 db 9d 9b 17 :<b8>
>1278 a5 d5 9d ab 17 60 4c 1c :<af>
>1280 99 b1 d0 c9 40 90 0a 29 :<b8>
>1288 3c 4a 4a 85 dc b1 d0 d0 :<34>
>1290 06 c9 20 f0 02 e6 dd 9d :<6d>
>1298 80 fc b1 d2 9d c0 fc a5 :<23>
>12a0 d4 91 d2 8a 09 40 91 d0 :<64>
>12a8 e8 60 20 84 9d e0 10 b0 :<48>
>12b0 cd 86 da bd 8b 17 30 01 :<a2>
>12b8 60 a9 00 9d 8b 17 bc 9b :<72>
>12c0 17 bd ab 17 aa 20 6a 17 :<50>
>12c8 a5 da 0a 0a aa 20 e3 12 :<d5>
>12d0 e8 c8 20 e3 12 98 18 69 :<93>
>12d8 27 a8 e8 20 e3 12 e8 c8 :<29>
>12e0 4c e3 12 78 8d 3f ff bd :<c0>
>12e8 80 fc 91 d0 bd c0 fc 91 :<3b>
>12f0 d2 8d 3e ff 58 60 ad 06 :<7d>
>12f8 ff 29 ef 8d 06 ff 60 ad :<ab>
>1300 06 ff 09 10 8d 06 ff 60 :<45>
>1308 20 87 9d e0 10 b0 1e bd :<b6>
>1310 8b 17 a8 4c 81 9a a9 10 :<7a>
```

```
>1318 85 da a9 07 85 14 a9 ff :<07>
>1320 85 15 20 84 9d 4c 57 14 :<d2>
>1328 a9 80 4c 18 13 4c 1c 99 :<eb>
>1330 4c 81 86 20 84 9d e0 01 :<39>
>1338 f0 07 e0 00 f0 37 4c 2d :<90>
>1340 13 a9 e8 85 38 20 9d 8a :<af>
>1348 a0 00 b9 00 d0 99 00 e8 :<2a>
>1350 b9 00 d1 99 00 e9 b9 00 :<e0>
>1358 d2 99 00 ea b9 00 d3 99 :<aa>
>1360 00 eb c8 d0 e5 a9 fb 2d :<20>
>1368 12 ff 8d 12 ff a9 e8 8d :<1c>
>1370 13 ff 4c dc 8b a9 ec 85 :<20>
>1378 38 20 9d 8a a9 04 0d 12 :<99>
>1380 ff 8d 12 ff a9 d0 d0 e7 :<9a>
>1388 20 e1 9d a5 14 d0 05 a5 :<f0>
>1390 15 d0 01 60 a5 15 30 95 :<6b>
>1398 a9 00 85 da 78 a9 be 8d :<54>
>13a0 14 03 a9 13 8d 15 03 58 :<95>
>13a8 a5 da d0 05 20 e1 ff d0 :<ea>
>13b0 f7 78 a9 0e 8d 14 03 a9 :<d1>
>13b8 ce 8d 15 03 58 60 c6 14 :<d5>
>13c0 d0 08 c6 15 10 04 a9 ff :<9b>
>13c8 85 da 4c 0e ce 4c 1c 99 :<a5>
>13d0 20 70 de 20 84 9d e0 28 :<34>
>13d8 b0 f3 86 ca 20 d8 9d e0 :<db>
>13e0 19 b0 ea 86 cd 4c a8 d8 :<95>
>13e8 20 84 9d e0 27 b0 de 8e :<44>
>13f0 e7 07 20 d8 9d e0 18 b0 :<d4>
>13f8 d4 8e e6 07 20 d8 9d 8a :<6c>
>1400 cd e7 07 90 c8 8d e8 07 :<e3>
>1408 20 d8 9d 8a cd e6 07 90 :<4a>
>1410 bc 8d e5 07 a9 93 4c 49 :<8b>
>1418 dc 20 2c 93 20 1a 93 a5 :<18>
>1420 64 85 d0 a5 65 85 d1 20 :<54>
>1428 91 94 20 2c 93 20 1a 93 :<fb>
>1430 a0 00 b1 d0 85 da b1 64 :<bb>
>1438 91 d0 a5 da 91 64 c8 c0 :<ee>
>1440 03 d0 ef 60 20 d2 9d e0 :<89>
>1448 08 b0 82 8a 49 07 aa bd :<d4>
>1450 89 c2 85 da 20 d8 9d a0 :<de>
>1458 00 b1 14 e0 01 f0 07 e0 :<35>
>1460 00 f0 08 4c 7e 12 05 da :<8f>
>1468 91 14 60 48 a5 da 49 ff :<c5>
>1470 85 da 68 25 da 91 14 60 :<7a>
>1478 20 2b cd f0 01 60 4c f9 :<d8>
>1480 ff 08 a9 00 85 ad a9 04 :<ff>
>1488 85 ae 28 f0 16 20 84 9d :<3c>
>1490 86 ae 20 73 04 f0 0c c5 :<d6>
>1498 2c f0 03 4c a1 94 20 84 :<17>
>14a0 9d 86 ad a9 00 85 ac 85 :<d7>
>14a8 ab 20 85 a7 a2 00 20 97 :<09>
>14b0 a7 a9 0d 20 8b a7 a2 00 :<9a>
>14b8 20 6a 17 a0 00 b1 d0 29 :<70>
>14c0 7f c9 20 b0 05 69 40 4c :<1f>
>14c8 d9 14 c9 40 90 0b c9 60 :<8c>
>14d0 b0 04 69 20 d0 03 18 69 :<38>
>14d8 40 20 8b a7 c8 c0 28 90 :<ef>
>14e0 dc a9 0d 20 8b a7 e8 e0 :<f0>
>14e8 19 90 cd a9 0d 20 8b a7 :<59>
>14f0 20 cc ff a9 00 4c c3 ff :<a8>
>14f8 20 6b a8 20 de 9d 84 d0 :<60>
>1500 85 d1 20 de 9d a6 14 a8 :<8f>
>1508 a9 d0 4c d8 ff 20 84 9d :<05>
>1510 86 ae 20 d8 9d 86 ad a9 :<a0>
>1518 00 85 ac 85 ab 20 85 a7 :<90>
>1520 a2 00 20 97 a7 a9 0d 20 :<27>
>1528 8b a7 20 79 04 f0 03 20 :<91>
>1530 91 94 20 ff 8a 4c eb 14 :<7e>
>1538 a9 e6 20 21 cb 20 b5 cc :<c8>
>1540 a2 00 8e 78 02 e8 85 0a :<b9>
>1548 a0 05 20 3f ca 4c fa a7 :<2a>
>1550 4c 1c 99 20 84 9d e0 03 :<f6>
>1558 b0 f6 86 da a9 a4 20 93 :<9a>
>1560 94 20 84 9d 8a c9 03 b0 :<f3>
>1568 e7 c5 da f0 e3 85 db a2 :<a3>
>1570 0b b5 c2 9d e8 0f ca 10 :<39>
>1578 f8 a2 03 bd ee 07 9d f4 :<95>
>1580 0f ca 10 f7 ad 15 ff 8d :<a6>
>1588 f8 0f ad 19 ff 8d f9 0f :<1f>
>1590 a9 00 85 d0 85 d2 a6 da :<5c>
>1598 bd 88 17 85 d1 a6 db bd :<79>
>15a0 88 17 85 d3 78 8d 3f ff :<41>
>15a8 a2 07 a0 00 b1 d0 91 d2 :<eb>
>15b0 c8 d0 f9 e6 d1 e6 d3 ca :<54>
>15b8 10 f0 8d 3e ff 58 ad f9 :<7d>
>15c0 0f 8d 19 ff ad f8 0f 8d :<7f>
>15c8 15 ff a2 03 bd f4 0f 9d :<35>
>15d0 ee 07 ca 10 f7 a2 0b bd :<1f>
>15d8 e8 0f 95 c2 ca 10 f8 4c :<f6>
>15e0 a8 d8 20 81 ff 20 84 ff :<4f>
>15e8 20 8a ff 20 cc ff 20 4f :<e2>
>15f0 ff 1b 4c 9e 93 00 a2 00 :<e3>
>15f8 8e 15 ff 8e 19 ff 86 de :<70>
>1600 a9 7b 8d 2a 03 a9 16 8d :<e5>
>1608 2b 03 4c 38 c7 a9 00 a2 :<1c>
>1610 08 a0 0f 20 ba ff a9 01 :<f0>
>1618 a2 27 a0 16 20 bd ff 20 :<cb>
>1620 85 a7 a9 00 4c c3 ff 49 :<4e>
>1628 20 7b 17 a5 61 f0 49 a0 :<f2>
>1630 00 78 8d 3f ff b1 62 8d :<3a>
>1638 3e ff 58 85 da 84 ef a5 :<e8>
>1640 91 c9 7f d0 01 60 a5 ef :<12>
>1648 f0 f5 78 a9 00 85 ef ad :<ce>
>1650 27 05 58 c5 da d0 e8 58 :<2b>
>1658 60 20 7b 17 a4 61 c0 0b :<05>
>1660 b0 16 84 ef 88 30 10 78 :<7a>
>1668 8d 3f ff b1 62 99 27 05 :<f7>
>1670 88 10 f8 8d 3e ff 58 60 :<64>
>1678 4c 1c 99 a2 ff 86 d6 86 :<97>
>1680 d7 e8 86 de 4c 08 ef 08 :<48>
>1688 20 86 9a 28 f0 3a c9 fd :<f0>
>1690 f0 6b c9 fc f0 6d 24 de :<a5>
>1698 10 0b a4 df f0 18 88 f0 :<ce>
>16a0 1c 84 df d0 23 20 e1 9d :<10>
>16a8 a5 14 85 df a5 15 85 e0 :<7f>
>16b0 a9 80 85 de d0 12 c6 df :<42>
```

```
>16b8 c6 e0 4c c8 16 84 df a4 :<58>
>16c0 e0 d0 05 84 de 4c e1 9d :<97>
>16c8 a5 d6 c5 39 d0 0f a5 d7 :<8a>
>16d0 c5 3a d0 09 a5 d8 85 3b :<cf>
>16d8 a5 d9 85 3c 60 a5 39 85 :<31>
>16e0 d6 85 14 a5 3a 85 d7 85 :<58>
>16e8 15 20 3d 8a a5 5f e9 01 :<77>
>16f0 85 3b 85 d8 a5 60 e9 00 :<5c>
>16f8 85 3c 85 d9 60 20 4c b6 :<55>
>1700 d0 c6 60 20 4c b6 f0 c0 :<c8>
>1708 60 a2 17 4c 86 86 20 84 :<ec>
>1710 9d 8e f2 07 20 d8 9d 8e :<5c>
>1718 f3 07 20 d8 9d 8e f4 07 :<a3>
>1720 20 d8 9d 8e f5 07 60 20 :<52>
>1728 7b 17 a4 61 f0 1e c9 28 :<b3>
>1730 b0 d7 a9 00 a2 08 a0 0f :<02>
>1738 20 ba ff a5 61 a6 62 a4 :<f3>
>1740 63 20 bd ff 20 85 a7 a9 :<3a>
>1748 00 4c c3 ff 4c 1c 99 a9 :<84>
>1750 00 85 0a 4c 6b a8 a9 e6 :<04>
>1758 20 21 cb 20 b5 cc a9 00 :<85>
>1760 8d 78 02 85 0a a0 05 4c :<81>
>1768 3f ca bd 02 d8 85 d0 85 :<05>
>1770 d2 bd 1b d8 85 d1 29 0b :<70>
>1778 85 d3 60 a5 16 48 20 2c :<62>
>1780 93 20 1a 93 68 85 16 60 :<c3>
>1788 08 ec f4 00 00 00 00 00 :<db>
>1790 00 00 00 00 00 00 00 00 :<38>
>1798 00 00 00 00 00 00 00 00 :<48>
>17a0 00 00 00 00 00 00 00 00 :<58>
>17a8 00 00 00 00 00 00 00 00 :<68>
>17b0 00 00 00 00 00 00 00 00 :<78>
>17b8 00 00 00 20 4f ff 0d 28 :<a5>
>17c0 43 29 20 42 59 20 42 45 :<3c>
>17c8 52 4e 48 41 52 44 20 4f :<fb>
>17d0 45 4d 45 52 20 28 31 39 :<2b>
>17d8 38 37 20 41 55 53 54 52 :<af>
>17e0 49 41 29 0d 00 4c 03 80 :<99>
```

P R O G R A M M E N D E

### WAITKEY a$
Wartet, bis die Taste a$ gedrückt wird und entleert vorher und nachher den Tastaturpuffer.

### BUFFER a$
Belegt den Tastaturpuffer mit dem String a$, wobei a$ nicht länger als 10 Zeichen sein darf.
Der BUFFER-Befehl kann vor allem zur Selbstmodifikation von Programmen verwendet werden.

### REGISTER a,x,y,s
Definiert die Prozessor-Register vor einem SYS-Aufruf
a = Akkumulator
x = X-Register
y = Y-Register
s = Statusregister.

### DISK a$
Sendet einen Befehl an die Diskettenstation
a$ = Befehlsstring.

### REPEAT x / UNTIL cond / WHILE cond
Dieser Befehl dient zur Programmierung von kurzen Schleifen. Er steht am Ende einer Zeile.
Bei REPEAT x wird diese Zeile x-mal wiederholt, bei REPEAT UNTIL cond wird die Zeile wiederholt, bis die Bedingung cond erfüllt ist. Bei REPEAT WHILE cond wird die Zeile wiederholt, solange die Bedingung erfüllt ist.

### RESET
Löst ein RESET aus. Im Direktmodus muß man vorher die Sicherheitsabfrage ARE YOU SURE? mit Y beantworten.

### USER
Ruft die USER-Routine ($17BB) auf. Dort kann zum Beispiel der Aufruf eines eigenen MS-Programmes stehen. Nach dem Start der Erweiterung befindet sich an dieser Stelle die Copyright-Meldung.

### Speicherplan von MEGA-TOOL:

$1001 – $100F
BASIC-Kopf
$1010 – $1095
Initialisierung
$1096 – $10E0
Behandlung des Tokens
$10E1 – $1180
Befehlswörter (Tokens)
$1186 – $11C3
Befehlsadressen
$11C4 – $1782
Routinen der Befehle
$1783 – $17BA
Arbeitsspeicher
$17BB – $17E7
USER-Routine

### Abkürzungen der Befehle:

| | |
|---|---|
| old | oL |
| block | bL |
| dblock | dB |
| merge | mE |
| dmerge | dM |
| sprite | sP |
| del | dE |
| fast | fA |
| slow | sL |
| multi | mU |
| revers | rE |
| trans | tR |
| sleep | sL |
| plot | pL |
| window | wI |
| swap | sW |
| bit | bI |
| reset | reS |
| hardcopy | hA |
| sblock | sB |
| llist | lL |
| dverify | dV |
| screen | sC |
| init | iN |
| id | id |
| waitkey | wA |
| buffer | bU |
| repeat | reP |
| register | reG |
| disk | dI |
| user | usE |

Die Abkürzungen der herkömmlichen BASIC-Befehle können sich ändern: Die Abkürzung „scR, für „scratch", bedeutet mit MEGA-TOOL „screen".

### FUNKTIONSWEISE DES PROGRAMMS:

Die Erweiterung liegt ab $1000 im Speicher. Nach dem Starten mit RUN wird der BASIC-Anfang auf $1800, das BASIC-Ende auf $EC00 gesetzt, um ein Überschreiben der Erweiterung, des Sprite-Speichers und der RAM-Bildschirme zu verhindern.
Außerdem werden die Vektoren zur Handhabung des USER-Tokens ($030C bis $0311) auf eigene Routinen gelegt. Dann werden der Bildschirm und der Speicher gelöscht, und MEGA-TOOL meldet sich mit der Einschaltmeldung.
Durch die Verwendung des USER-Tokens werden auch die neuen Befehle in Tokens umgewandelt und sind daher nur zwei Byte lang.
Als Arbeitsspeicher verwendet MEGA-TOOL den Bereich $EC00 bis $FCFF sowie die Zeropage-Adressen $D0 bis $D5, allerdings nur während der Ausführung des Befehls. Der übrige Systemspeicher bleibt unberührt, kann also für weitere Maschinensprache-Programme verwendet werden.
Die hochauflösende Grafik steht uneingeschränkt zur Verfügung. □

# Grafix 3D in Bewegung

**Mit Grafix können auf dem Plus4 und dem auf 64 KByte erweiterten C16/116 bewegte 3-D-Grafiken erzeugt werden. Diese Grafiken sind nicht nur zum Ansehen gedacht, sondern können auch als interessante Vorspänne oder Zwischentitel in eigene Programme eingebunden werden.**

Aller guten Dinge sind drei, und so besteht Grafix aus drei Teilprogrammen. Das erste, GRAFIX.START, schaltet den Grafikmodus ein und lädt die zwei restlichen Teile nach. GRAFIX.MC ist der Maschinenspracheteil, der für die schnellen Vektoroperationen zuständig ist. Das BASIC-Programm GRAFIX enthält die zur Dateneingabe und Datenkorrektur erforderlichen Routinen. Grafiken können editiert, gespeichert, geladen und überarbeitet werden.

## DIE ARBEIT MIT DEM PROGRAMM

Es lassen sich maximal 127 Punkte und 127 Linien definieren. Den einzelnen Koordinaten, Tiefe, Höhe und Breite, können ganzzahlige Werte zwischen −511 und +511 zugewiesen werden. Wenn Sie bei der Definierung der Linien, die an einem Punkt gezogen werden sollen, zum nächsten Punkt übergehen wollen, drücken Sie bei der erneuten Abfrage einfach RETURN.

Beim Löschen eines Punktes werden sämtliche Linien, die von oder zu diesem Punkt gezogen werden, ebenfalls gelöscht. Außerdem verschieben sich die Punktnummern der nachfolgenden Punkte um jeweils Eins nach unten. Die Daten der Linien erfahren hierbei automatisch eine entsprechende Änderung, so daß dadurch keine Fehler entstehen.

## BEDIENUNG DER GRAFIK

Während der 3-D-Berechnungen stehen Ihnen folgende Funktionen auf Tastendruck zur Verfügung:

R = Zurück ins Menü.
F = Bild einfrieren.
S = Programm starten.
C = Programm nach Unterbrechung fortsetzen.
L = Der Bildschirm wird von nun an ständig gelöscht. Die Berechnungen werden dadurch schneller. Nachteilig macht sich aber ein Flackern bemerkbar (L-Modus).
T = Die Bildschirmdarstellung erfolgt jetzt ohne Flakkern, die Berechnungen dauern aber länger (T-Modus).

**Drehbewegungen:**

Cursor rechts = Rechtsdrehung
Cursor links = Linksdrehung
Cursor oben = Drehung nach oben
Cursor unten = Drehung nach unten

**Drehbewegungen mit doppelter Schrittweite:**

> = Rechtsdrehung + = Drehung nach oben
< = Linksdrehung – = Drehung nach unten

Mit den Tasten 1 bis 8 werden beide Winkel gleichzeitig beeinflußt.

**Weitere Bewegungen:**

O = Drehwinkel auf Null
G = Objekt näher heranholen
K = Objekt weiter wegrücken
N = Drehen und heranholen
M = Drehen und wegrücken

Neben Laden, Speichern, Ändern, Löschen und Hinzufügen von Daten bietet das Menü auch die Möglichkeit, Bewegungsabläufe vorzuprogrammieren. In diesem Menü-Unterpunkt ist die Taste mit der entsprechenden Bewegungsfunktion zu drücken und danach anzugeben, wie viele solcher Bewegungen, maximal 255, das Objekt ausführen soll.

**Eingabebeispiel:**

**Anzahl der Punkte? 5**

| Punkt | Tiefe | Höhe | Breite |
|---|---|---|---|
| 1 | 0 | 60 | 0 |
| 2 | 40 | 0 | 40 |
| 3 | −40 | 0 | 40 |
| 4 | −40 | 0 | −40 |
| 5 | 40 | 0 | −40 |

**Linien von**

| | |
|---|---|
| 1 zu 2 | 2 zu 3 |
| 1 zu 3 | 2 zu 5 |
| 1 zu 4 | 3 zu 4 |
| 1 zu 5 | 4 zu 5 |

## BESONDERE HINWEISE

Die neue Draw-Routine arbeitet teilweise ungenau, wodurch es zu Veränderungen im Farbspeicher kom-

men kann. Dieser Fehler tritt allerdings nur im L-Modus auf.
Wenn ein Objekt sehr nahe an den Betrachter herangerückt wurde, wird, um Fehlberechnungen vorzubeugen, die Abbildung des Punktes auf 160,100 gesetzt. Das Programm ist mit einem vollständigen Clipp-Teil versehen, so daß auch sehr große Objekte dargestellt werden können.

## EINGABE UND ABSPEICHERN DES GRAFIX-PROGRAMMES

Da die Datasette nur der Reihe nach zu lesen vermag, müssen Datasetten-Benutzer sich beim Abspeichern der Programmteile an eine gewisse Reihenfolge halten:
1. GRAFIX.START
2. GRAFIX.MC
3. GRAFIX
4. GRAFIX.CBM

GRAFIX.START und GRAFIX sind reine BASIC-Programme, so daß hierbei keine Besonderheiten zu beachten sind.
GRAFIX.MC und GRAFIX.CBM sind gemäß den im Programmvorspann vermerkten Anweisungen mit Hilfe des Maschinenmonitors TEDMON abzusaven. Der Maschinenteil liegt bei $5750 bis $6E00, die Daten der Objekte bei $1000 bis $1800 im Speicher.
GRAFIX.CBM ist ein Datenfile, das das Commodore-Emblem in 3-D enthält. Sie können nach dem Start von GRAFIX.START diese Grafik durch entsprechende Wahl einladen. GRAFIX verwendet im T-Modus noch den Bereich $8000 bis $9FFF als Pseudo-Grafikspeicher.

```
10 rem grafix.start=============p4 <mo>
20 rem (p) commodore welt team     <ho>
30 rem ============================ <ng>
40 rem (c)  by frank koster        <ge>
50 rem                             <pd>
60 rem                             <ah>
70 rem basic v3.5 und m-code       <jd>
80 rem plus4 (c16/116 +64 kb)      <fd>
90 rem ============================ <jg>
100 graphic1,1:graphic0            <oa>
110 x$=right$(str$(peek(174)),1)   <fd>
120 a$="grafix":r$=chr$(13)        <em>
130 d$=chr$(34)                    <op>
140 b$=d$+a$+".mc"+d$+","+x$+",1"+
r$                                 <oj>
150 c$=d$+a$+d$+","+x$+r$          <nn>
160 key1,"10"+b$+"10"+c$+"rU"+r$   <nf>
170 poke2035,0:sys56364:end        <df>
180 rem ============================ <cn>
190 rem dieses programm laedt      <cc>
200 rem "GRAFIX.MC" und "GRAFIX"   <hj>
210 rem nach und startet das       <af>
220 rem letztere.                  <pm>
230 rem ============================ <jc>
240 rem  p r o g r a m m e n d e   <nf>
250 rem ============================ <jf>
```

```
GRAFIX.MC=========================p4
(p) COMMODORE WELT TEAM
==================================
(c) by Frank Koster

Plus4 (C16/116 + 64 KB)
==================================
Das Programm ist mit dem Maschinen-
sprachmonitor TEDMON unter Zuhilfe-
nahme des Pruefsummenprogrammes
CHECKMON einzugeben und auf Disket-
te mit s"grafix.mc",8,5750,6e00 ab-
zuspeichern, auf Kassette mit
s"grafix.mc",1,5750,6e00.
==================================
>5750 a5 d4 c5 da 90 49 d0 2f :<53>
>5758 a5 d3 c5 d9 90 41 d0 27 :<e4>
>5760 20 da 58 38 a5 d0 e5 d6 :<be>
>5768 b0 10 49 ff 69 02 aa ea :<4a>
>5770 ea ca d0 01 60 20 ea 59 :<11>
>5778 d0 f7 aa e8 ea ea ca d0 :<45>
>5780 01 60 20 4a 59 d0 f7 a5 :<84>
>5788 d0 a4 d6 85 d6 84 d0 a5 :<31>
>5790 d3 a4 d9 85 d9 84 d3 a5 :<6f>
>5798 d4 a4 da 85 da 84 d4 20 :<6a>
>57a0 da 58 38 a5 d9 e5 d3 85 :<ce>
>57a8 d3 85 dc a5 da e5 d4 85 :<33>
>57b0 d4 85 dd 38 a5 d0 e5 d6 :<35>
>57b8 90 1a d0 21 a0 00 e6 dc :<46>
>57c0 d0 02 e6 dd e6 dd c6 dc :<c4>
>57c8 d0 05 c6 dd d0 01 60 20 :<d8>
>57d0 bc 59 d0 f2 49 ff a8 c8 :<78>
>57d8 84 d0 4c 5c 58 85 d0 e6 :<b5>
>57e0 d0 e6 dc e6 d3 d0 02 e6 :<ed>
>57e8 d4 e6 dd 20 60 5a a0 00 :<65>
>57f0 a5 d6 85 e0 a5 d7 85 e1 :<62>
>57f8 aa a5 d8 85 e2 8a 29 0f :<e8>
>5800 85 e1 8a 46 e2 6a 46 e2 :<c1>
>5808 6a 46 e2 6a 46 e2 6a f0 :<6d>
>5810 09 aa ca f0 05 20 4a 59 :<90>
>5818 d0 f8 c6 dc d0 05 c6 dd :<3c>
>5820 d0 01 60 18 a5 e0 65 d6 :<33>
>5828 85 e0 a5 e1 65 d7 85 e1 :<8a>
>5830 aa a5 e2 65 d8 85 e2 8a :<2a>
>5838 29 0f 85 e1 8a 46 e2 6a :<38>
>5840 46 e2 6a 46 e2 6a 46 e2 :<84>
>5848 6a d0 05 20 bc 59 d0 ca :<c7>
>5850 aa 20 7f 59 ca f0 c3 20 :<c3>
>5858 4a 59 d0 f8 e6 d0 e6 dc :<47>
>5860 e6 d3 d0 02 e6 d4 e6 dd :<d5>
>5868 20 60 5a a0 00 a5 d6 85 :<5a>
>5870 e0 a5 d7 85 e1 aa a5 d8 :<43>
>5878 85 e2 8a 29 0f 85 e1 8a :<9b>
>5880 46 e2 6a 46 e2 6a 46 e2 :<04>
>5888 6a 46 e2 6a f0 09 aa ca :<ec>
>5890 f0 05 20 ea 59 d0 f8 c6 :<b3>
>5898 dc d0 05 c6 dd d0 01 60 :<85>
>58a0 18 a5 e0 65 d6 85 e0 a5 :<7e>
```

```
>58a8 e1 65 d7 85 e1 aa a5 e2 :<52>
>58b0 65 d8 85 e2 8a 29 0f 85 :<07>
>58b8 e1 8a 46 e2 6a 46 e2 6a :<e3>
>58c0 46 e2 6a 46 e2 6a d0 05 :<a1>
>58c8 20 bc 59 d0 ca aa 20 1f :<ae>
>58d0 5a ca f0 c3 20 ea 59 d0 :<3a>
>58d8 f8 ea a5 d0 48 aa 29 07 :<65>
>58e0 85 05 8a 29 f8 0a 26 d1 :<ce>
>58e8 0a 26 d1 0a 26 d1 85 26 :<c3>
>58f0 a6 d1 86 27 0a 26 d1 0a :<6a>
>58f8 26 d1 18 65 26 65 05 85 :<2f>
>5900 26 90 02 e6 27 a5 d1 65 :<85>
>5908 27 69 80 85 27 a5 d3 a8 :<20>
>5910 29 07 aa 98 29 f8 18 65 :<e1>
>5918 26 85 26 a5 27 65 d4 85 :<47>
>5920 27 a9 00 e8 38 6a ca d0 :<9e>
>5928 fc 85 28 a9 08 e5 05 85 :<b7>
>5930 06 a0 00 78 8d 3f ff b1 :<8d>
>5938 26 05 28 91 26 8d 3e ff :<21>
>5940 58 ea e6 05 e6 06 68 85 :<40>
>5948 d0 60 c6 05 f0 11 c6 26 :<6f>
>5950 78 8d 3f ff b1 26 05 28 :<34>
>5958 91 26 8d 3e ff 58 60 a9 :<d2>
>5960 08 85 05 38 a5 26 e9 39 :<ef>
>5968 85 26 b0 02 c6 27 c6 27 :<1f>
>5970 78 8d 3f ff b1 26 05 28 :<f4>
>5978 91 26 8d 3e ff 58 60 c6 :<fc>
>5980 05 f0 15 c6 26 46 28 f0 :<ce>
>5988 46 78 8d 3f ff b1 26 05 :<df>
>5990 28 91 26 8d 3e ff 58 60 :<1b>
>5998 a9 08 85 05 38 a5 26 e9 :<e8>
>59a0 39 85 26 b0 02 c6 27 c6 :<c2>
>59a8 27 46 28 f0 22 78 8d 3f :<b2>
>59b0 ff b1 26 05 28 91 26 8d :<60>
>59b8 3e ff 58 60 46 28 f0 0f :<4c>
>59c0 78 8d 3f ff b1 26 05 28 :<8a>
>59c8 91 26 8d 3e ff 58 60 66 :<92>
>59d0 28 a5 26 69 08 85 26 90 :<ca>
>59d8 02 e6 27 78 8d 3f ff b1 :<81>
>59e0 26 05 28 91 26 8d 3e ff :<92>
>59e8 58 60 c6 06 f0 11 e6 26 :<33>
>59f0 78 8d 3f ff b1 26 05 28 :<f3>
>59f8 91 26 8d 3e ff 58 60 a9 :<91>
>5a00 08 85 06 18 a5 26 69 39 :<0a>
>5a08 85 26 90 02 e6 27 e6 27 :<da>
>5a10 78 8d 3f ff b1 26 05 28 :<b3>
>5a18 91 26 8d 3e ff 58 60 c6 :<c2>
>5a20 06 f0 15 e6 26 46 28 f0 :<11>
>5a28 a6 78 8d 3f ff b1 26 05 :<7f>
>5a30 28 91 26 8d 3e ff 58 60 :<ca>
>5a38 a9 08 85 06 18 a5 26 69 :<fa>
>5a40 39 85 26 90 02 e6 27 e6 :<cb>
>5a48 27 46 28 f0 82 78 8d 3f :<68>
>5a50 ff b1 26 05 28 91 26 8d :<9e>
>5a58 3e ff 58 60 ea ea ea ea :<6e>
>5a60 a0 00 84 86 84 87 84 88 :<7e>
>5a68 84 d6 84 d7 84 d8 a5 d0 :<b7>
>5a70 84 d0 a2 00 86 d2 0a 26 :<0c>
>5a78 d2 0a 26 d2 0a 26 d2 0a :<3a>
>5a80 26 d2 85 d1 e6 87 a5 d3 :<e5>
>5a88 84 d3 a6 d4 86 d5 d0 09 :<57>
>5a90 06 87 26 88 0a 26 d5 f0 :<f7>
>5a98 f7 0a 26 d5 06 87 26 88 :<d4>
>5aa0 0a 26 d5 06 87 26 88 0a :<99>
>5aa8 26 d5 06 87 26 88 85 d4 :<6e>
>5ab0 38 a5 d1 e5 d4 a8 a5 d2 :<f5>
>5ab8 e5 d5 90 0c 84 d1 85 d2 :<59>
>5ac0 a5 88 85 d8 a5 87 85 d7 :<6a>
>5ac8 46 d5 66 d4 66 d3 46 88 :<fb>
>5ad0 66 87 66 86 90 03 60 3d :<d8>
>5ad8 47 38 a5 d0 e5 d3 aa a5 :<c1>
>5ae0 d1 e5 d4 a8 a5 d2 e5 d5 :<ac>
>5ae8 90 de 86 d0 84 d1 85 d2 :<a8>
>5af0 18 a5 d6 65 86 85 d6 a5 :<01>
>5af8 d7 65 87 85 d7 a5 d8 65 :<d5>
>5b00 88 85 d8 4c c8 5a ea ea :<e7>
>5b08 a9 80 85 9a a9 01 a6 d0 :<34>
>5b10 a0 00 ad 5e 06 20 ba ff :<0f>
>5b18 ad 5e 06 a2 5f a0 06 20 :<95>
>5b20 bd ff a2 00 a0 10 86 d8 :<6a>
>5b28 84 d9 a9 d8 a0 18 20 d8 :<23>
>5b30 ff a9 00 85 9a 60 00 00 :<bd>
>5b38 00 00 00 00 00 00 00 00 :<cb>
>5b40 00 00 00 00 78 8d 3f ff :<54>
>5b48 b1 26 05 28 91 26 8d 3e :<1e>
>5b50 ff 58 00 00 a2 00 a5 88 :<b2>
>5b58 f0 22 a5 87 c9 0f 90 0f :<22>
>5b60 a9 68 e5 87 a8 a9 80 85 :<dd>
>5b68 8b b9 f7 03 85 89 60 69 :<41>
>5b70 4b a8 a9 80 85 8b b9 f7 :<e7>
>5b78 03 85 89 60 a5 87 c9 b4 :<a7>
>5b80 90 0d e9 b4 a8 a9 80 85 :<49>
>5b88 8b b9 f7 03 85 89 60 c9 :<54>
>5b90 5b 90 0d a9 b4 e5 87 a8 :<05>
>5b98 86 8b b9 f7 03 85 89 60 :<d7>
>5ba0 a8 86 8b b9 f7 03 85 89 :<fb>
>5ba8 60 a2 00 a5 88 f0 20 a5 :<96>
>5bb0 87 c9 0f 90 0b e9 0e a8 :<bb>
>5bb8 b9 f7 03 85 89 86 8b 60 :<c8>
>5bc0 a9 0f e5 87 a8 a9 80 85 :<6d>
>5bc8 8b b9 f7 03 85 89 60 a5 :<62>
>5bd0 87 c9 b5 90 0f 49 ff 69 :<7c>
>5bd8 0e a8 a9 80 85 8b b9 f7 :<9d>
>5be0 03 85 89 60 c9 5a 90 0d :<9b>
>5be8 e9 5a a8 a9 80 85 8b b9 :<97>
>5bf0 f7 03 85 89 60 a9 5b e5 :<cf>
>5bf8 87 a8 86 8b b9 f7 03 85 :<04>
>5c00 89 60 06 8c 26 8d 26 10 :<b8>
>5c08 06 8c 26 8d 26 10 a6 8c :<1a>
>5c10 a5 8d 85 8c a5 10 85 8d :<a9>
>5c18 a9 00 85 10 8a 10 06 e6 :<2b>
>5c20 8c d0 02 e6 8d 60 a5 c6 :<70>
>5c28 c9 40 b0 03 8d cf 07 ad :<a7>
>5c30 cf 07 c9 30 d0 28 ad d2 :<58>
>5c38 07 d0 0e ad d3 07 d0 17 :<a6>
>5c40 a9 66 8d d3 07 ee d2 07 :<43>
```

```
>5c48 60 ad d3 07 d0 09 ce d2 :<a4>
>5c50 07 a9 fe 8d d3 07 60 ce :<c8>
>5c58 d3 07 ce d3 07 60 c9 33 :<e3>
>5c60 d0 21 ad d2 07 f0 10 ad :<b8>
>5c68 d3 07 c9 66 90 09 a9 00 :<a7>
>5c70 8d d2 07 8d d3 07 60 ee :<d2>
>5c78 d3 07 ee d3 07 d0 03 ee :<bd>
>5c80 d2 07 60 c9 2b d0 28 ad :<12>
>5c88 d0 07 d0 0e ad d1 07 d0 :<9d>
>5c90 17 a9 66 8d d1 07 ee d0 :<41>
>5c98 07 60 ad d1 07 d0 09 ce :<a6>
>5ca0 d0 07 a9 fe 8d d1 07 60 :<e3>
>5ca8 ce d1 07 ce d1 07 60 c9 :<1a>
>5cb0 28 d0 21 ad d0 07 f0 10 :<61>
>5cb8 ad d1 07 c9 66 90 09 a9 :<8d>
>5cc0 00 8d d0 07 8d d1 07 60 :<c2>
>5cc8 ee d1 07 ee d1 07 d0 03 :<6c>
>5cd0 ee d0 07 60 4c 68 6d ea :<84>
>5cd8 ea ea 60 a5 88 45 8e 85 :<f9>
>5ce0 8f a9 00 85 88 85 26 85 :<9f>
>5ce8 27 85 28 a6 8d a4 8c 98 :<af>
>5cf0 29 01 f0 0c a5 26 65 86 :<b3>
>5cf8 85 26 a5 27 65 87 85 27 :<e6>
>5d00 06 86 26 87 98 29 02 f0 :<52>
>5d08 0c a5 26 65 86 85 26 a5 :<2b>
>5d10 27 65 87 85 27 06 86 26 :<18>
>5d18 87 98 29 04 f0 0c a5 26 :<c5>
>5d20 65 86 85 26 a5 27 65 87 :<9d>
>5d28 85 27 06 86 26 87 98 29 :<d7>
>5d30 08 f0 0c a5 26 65 86 85 :<e5>
>5d38 26 a5 27 65 87 85 27 06 :<ca>
>5d40 86 26 87 98 29 10 f0 0c :<01>
>5d48 a5 26 65 86 85 26 a5 27 :<26>
>5d50 65 87 85 27 06 86 26 87 :<bd>
>5d58 98 29 20 f0 0c a5 26 65 :<68>
>5d60 86 85 26 a5 27 65 87 85 :<e7>
>5d68 27 06 86 26 87 26 88 98 :<f6>
>5d70 29 40 f0 12 a5 26 65 86 :<62>
>5d78 85 26 a5 27 65 87 85 27 :<e6>
>5d80 a5 28 65 88 85 28 06 86 :<c5>
>5d88 26 87 26 88 98 29 80 f0 :<ec>
>5d90 12 a5 26 65 86 85 26 a5 :<62>
>5d98 27 65 87 85 27 a5 28 65 :<9a>
>5da0 88 85 28 06 86 26 87 26 :<d2>
>5da8 88 8a 29 01 f0 12 a5 26 :<b4>
>5db0 65 86 85 26 a5 27 65 87 :<2d>
>5db8 85 27 a5 28 65 88 85 28 :<16>
>5dc0 06 86 26 87 26 88 8a 29 :<0f>
>5dc8 02 f0 0c a5 27 65 87 85 :<21>
>5dd0 27 a5 28 65 88 85 28 06 :<ee>
>5dd8 26 26 27 26 28 a6 26 a5 :<f1>
>5de0 27 85 26 a5 28 85 27 a5 :<bb>
>5de8 8f 85 28 8a 10 06 e6 26 :<41>
>5df0 d0 02 e6 27 60 00 b1 26 :<58>
>5df8 05 28 91 26 60 ea ea ea :<f9>
>5e00 ea ea ea ea 00 a5 8d d0 :<13>
>5e08 0d a5 8c c9 70 b0 07 a9 :<66>
>5e10 00 85 86 85 87 60 a0 00 :<36>
>5e18 84 8a 84 8b e6 8a a5 8d :<49>
>5e20 06 8a 26 8b 06 8c 2a 10 :<11>
>5e28 f7 c9 81 b0 11 a8 a5 8c :<30>
>5e30 d0 0b a5 8a 0a 85 86 a5 :<ac>
>5e38 8b 2a 85 87 60 98 85 8d :<93>
>5e40 49 ff 85 89 a5 8c 49 ff :<1d>
>5e48 a8 c8 84 88 d0 02 e6 89 :<75>
>5e50 a5 8a 85 86 a5 8b 85 87 :<65>
>5e58 46 8d 66 8c 46 8b 66 8a :<0d>
>5e60 90 03 4c 87 5e 38 a5 88 :<7b>
>5e68 e5 8c a8 a5 89 e5 8d 90 :<90>
>5e70 e7 85 89 84 88 18 a5 86 :<16>
>5e78 65 8a 85 86 a5 87 65 8b :<ec>
>5e80 85 87 4c 58 5e ea ea a4 :<4e>
>5e88 89 c4 8d 90 0e d0 06 a4 :<7e>
>5e90 88 c4 8c 90 06 e6 86 d0 :<a0>
>5e98 02 e6 87 60 00 00 00 00 :<72>
>5ea0 00 00 00 00 00 00 00 00 :<9f>
>5ea8 00 00 00 00 46 27 66 26 :<ad>
>5eb0 8d 66 8c a5 28 45 8e d0 :<e1>
>5eb8 12 18 a5 26 65 8c 85 8c :<c4>
>5ec0 a5 27 65 8d 85 8d a5 28 :<e0>
>5ec8 85 8e 60 a5 8e 85 8f a5 :<08>
>5ed0 28 30 28 85 8e 38 a5 26 :<1c>
>5ed8 e5 8c 85 8c a5 27 e5 8d :<ba>
>5ee0 85 8d b0 16 49 ff 85 8d :<8d>
>5ee8 a5 8c 49 ff 69 01 d0 02 :<b4>
>5ef0 e6 8d 85 8c a5 8e 49 80 :<0d>
>5ef8 85 8e 60 a5 8f 85 8e 38 :<b7>
>5f00 a5 8c e5 26 85 8c a5 8d :<19>
>5f08 e5 27 85 8d b0 14 49 ff :<38>
>5f10 85 8d a5 8c 49 ff 69 01 :<1b>
>5f18 d0 02 e6 8d 85 8c a5 28 :<76>
>5f20 85 8e 60 00 00 00 00 00 :<bd>
>5f28 00 00 00 00 00 00 00 00 :<af>
>5f30 00 00 46 27 66 26 46 8d :<01>
>5f38 66 8c a5 28 45 8e f0 03 :<b8>
>5f40 4c b9 5e ea ea a5 28 4c :<48>
>5f48 d3 5e a9 00 85 8c 85 8d :<45>
>5f50 a5 88 45 8b 85 8e a6 89 :<2d>
>5f58 18 8a 29 01 f0 0c a5 8c :<06>
>5f60 65 86 85 8c a5 8d 65 87 :<04>
>5f68 85 8d 06 86 26 87 8a 29 :<a6>
>5f70 02 f0 0c a5 8c 65 86 85 :<c7>
>5f78 8c a5 8d 65 87 85 8d 06 :<eb>
>5f80 86 26 87 8a 29 04 f0 0c :<b0>
>5f88 a5 8c 65 86 85 8c a5 8d :<2f>
>5f90 65 87 85 8d 06 86 26 87 :<e7>
>5f98 8a 29 08 f0 0c a5 8c 65 :<57>
>5fa0 86 85 8c a5 8d 65 87 85 :<74>
>5fa8 8d 06 86 26 87 8a 29 10 :<70>
>5fb0 f0 0c a5 8c 65 86 85 8c :<04>
>5fb8 a5 8d 65 87 85 8d 06 86 :<84>
>5fc0 26 87 8a 29 20 f0 0c a5 :<d8>
>5fc8 8c 65 86 85 8c a5 8d 65 :<4a>
>5fd0 87 85 8d 06 86 26 87 8a :<3c>
>5fd8 29 40 f0 0c a5 8c 65 86 :<52>
>5fe0 85 8c a5 8d 65 87 85 8d :<e4>
```

```
>5fe8 06 86 26 87 8a 29 80 f0 :<a5>
>5ff0 0c a5 8c 65 86 85 8c a5 :<57>
>5ff8 8d 65 87 85 8d 60 ea ad :<8b>
>6000 00 10 85 42 20 26 5c ad :<98>
>6008 d3 07 85 87 ad d2 07 85 :<ac>
>6010 88 20 54 5b a5 89 85 03 :<db>
>6018 a5 8b 85 04 20 a9 5b a5 :<31>
>6020 89 85 05 a5 8b 85 06 ad :<05>
>6028 d0 07 85 88 ad d1 07 85 :<4e>
>6030 87 20 54 5b a5 89 85 e0 :<28>
>6038 a5 8b 85 e1 20 a9 5b a5 :<da>
>6040 89 85 e2 a5 8b 85 e3 a0 :<09>
>6048 00 84 2a a4 2a b9 01 10 :<c1>
>6050 85 d0 85 86 b9 02 10 aa :<8c>
>6058 29 7f 85 d1 85 87 8a 29 :<08>
>6060 80 85 d2 85 88 b9 01 11 :<89>
>6068 85 d3 b9 02 11 aa 29 7f :<92>
>6070 85 d4 8a 29 80 85 d5 b9 :<06>
>6078 01 12 85 d6 b9 02 12 aa :<5f>
>6080 29 7f 85 d7 8a 29 80 85 :<3a>
>6088 d8 84 2a a5 05 85 89 a5 :<d5>
>6090 06 85 8b 20 4a 5f a5 8c :<ed>
>6098 85 26 a5 8d 85 27 a5 8e :<07>
>60a0 85 28 a5 d6 85 86 a5 d7 :<96>
>60a8 85 87 a5 d8 85 88 a5 03 :<7e>
>60b0 85 89 a5 04 85 8b 20 4a :<e5>
>60b8 5f 20 ab 5e 20 02 5c a5 :<b6>
>60c0 8c 85 86 85 d9 a5 8d 85 :<64>
>60c8 87 85 da a5 8e 85 88 85 :<ae>
>60d0 db a5 e2 85 89 a5 e3 85 :<c1>
>60d8 8b 20 4a 5f 20 08 5c 18 :<1e>
>60e0 a9 00 85 28 69 01 85 27 :<8c>
>60e8 a9 35 85 26 20 b3 5e a5 :<ce>
>60f0 8c 85 26 a5 8d 85 27 a5 :<07>
>60f8 8e 85 28 a5 d3 85 86 a5 :<41>
>6100 d4 85 87 a5 d5 85 88 a5 :<95>
>6108 e0 85 89 a5 e1 85 8b 20 :<da>
>6110 4a 5f 20 08 5c 20 b3 5e :<ce>
>6118 20 05 5e a5 86 85 dc a5 :<0c>
>6120 87 85 dd a5 8e 85 de a5 :<a0>
>6128 d6 85 86 a5 d7 85 87 a5 :<22>
>6130 d8 85 88 a5 05 85 89 a5 :<81>
>6138 06 85 8b 20 4a 5f a5 8c :<7d>
>6140 85 26 a5 8d 85 27 a5 8e :<d7>
>6148 85 28 a5 d0 85 86 a5 d1 :<91>
>6150 85 87 a5 d2 85 88 a5 03 :<fa>
>6158 85 89 a5 04 85 8b 20 4a :<22>
>6160 5f 20 32 5f 20 02 5c a5 :<68>
>6168 dc 85 86 a5 dd 85 87 a5 :<08>
>6170 de 85 88 20 db 5c a9 00 :<ed>
>6178 85 8d 85 8e a9 a0 85 8c :<13>
>6180 20 b3 5e a4 2a a5 8c 99 :<ec>
>6188 00 08 a5 8d 05 8e 99 01 :<2a>
>6190 08 84 2a a5 d3 85 86 a5 :<93>
>6198 d4 85 87 a5 d5 85 88 a5 :<63>
>61a0 e2 85 89 a5 e3 85 8b 20 :<4a>
>61a8 4a 5f a5 8c 85 26 a5 8d :<bb>
>61b0 85 27 a5 8e 85 28 a5 d9 :<77>
>61b8 85 86 a5 da 85 87 a5 db :<d6>
>61c0 85 88 a5 e0 85 89 a5 e1 :<3f>
>61c8 85 8b 20 4a 5f 20 32 5f :<15>
>61d0 20 02 5c a5 dc 85 86 a5 :<7f>
>61d8 dd 85 87 a5 de 85 88 20 :<bf>
>61e0 db 5c a9 00 85 8d 85 8e :<be>
>61e8 a9 64 85 8c 20 b3 5e a4 :<d8>
>61f0 2a a5 8c 99 00 09 a5 8d :<83>
>61f8 05 8e 99 01 09 c8 c8 84 :<3c>
>6200 2a c6 42 f0 03 4c 4b 60 :<dc>
>6208 ad 0a 59 10 4c 20 5d 6b :<46>
>6210 86 de a6 de bc 02 13 d0 :<72>
>6218 03 4c d0 6b 88 98 0a a8 :<25>
>6220 b9 00 08 85 d3 b9 01 08 :<50>
>6228 85 d4 b9 00 09 85 d0 b9 :<57>
>6230 01 09 85 d1 bc 01 13 e8 :<05>
>6238 e8 86 de 88 98 0a a8 b9 :<7e>
>6240 00 08 85 d9 b9 01 08 85 :<bf>
>6248 da b9 00 09 85 d6 b9 01 :<58>
>6250 09 85 d7 20 b0 66 4c 12 :<36>
>6258 62 20 ff 6c 86 de a6 de :<c8>
>6260 bc 02 13 d0 05 84 10 4c :<b7>
>6268 ff 5f 88 98 0a a8 b9 00 :<ae>
>6270 08 85 d3 b9 01 08 85 d4 :<84>
>6278 b9 00 09 85 d0 b9 01 09 :<34>
>6280 85 d1 bc 01 13 e8 e8 86 :<8c>
>6288 de 88 98 0a a8 b9 00 08 :<a8>
>6290 85 d9 b9 01 08 85 da b9 :<09>
>6298 00 09 85 d6 b9 01 09 85 :<17>
>62a0 d7 20 b0 66 4c 5e 62 00 :<b5>
>62a8 00 00 00 00 00 00 00 00 :<b3>
>62b0 00 00 00 00 00 00 00 00 :<c3>
>62b8 00 00 00 c9 11 d0 03 68 :<99>
>62c0 68 60 c9 15 d0 03 4c 26 :<1b>
>62c8 5c c9 23 d0 0f a9 00 8d :<30>
>62d0 d0 07 8d d1 07 8d d2 07 :<d2>
>62d8 8d d3 07 60 c9 38 d0 07 :<77>
>62e0 20 36 5c 20 87 5c 60 c9 :<70>
>62e8 3b d0 07 20 62 5c 20 b3 :<1d>
>62f0 5c 60 c9 08 d0 07 20 36 :<c6>
>62f8 5c 20 b3 5c 60 c9 0b d0 :<67>
>6300 07 20 62 5c 20 87 5c 60 :<cc>
>6308 c9 10 d0 0a 20 36 5c 20 :<3b>
>6310 36 5c 20 87 5c 60 c9 13 :<ca>
>6318 d0 0a 20 62 5c 20 62 5c :<8b>
>6320 20 b3 5c 60 c9 18 d0 0a :<22>
>6328 20 87 5c 20 87 5c 20 62 :<80>
>6330 5c 60 c9 1b d0 0a 20 b3 :<7e>
>6338 5c 20 b3 5c 20 36 5c 60 :<8d>
>6340 c9 2f d0 07 20 36 5c 20 :<32>
>6348 36 5c 60 c9 2c d0 07 20 :<c6>
>6350 62 5c 20 62 5c 60 c9 2e :<c2>
>6358 d0 07 20 87 5c 20 87 5c :<7f>
>6360 60 c9 36 d0 07 20 b3 5c :<db>
>6368 20 b3 5c 60 c9 25 d0 1d :<18>
>6370 18 ad e9 60 69 30 8d e9 :<67>
>6378 60 90 11 ac e5 60 c8 c0 :<b0>
>6380 50 90 06 a9 f8 8d e9 60 :<4d>
```

```
>6388 88 8c e5 60 60 c9 1a d0 :<49>
>6390 29 ac e5 60 c0 02 b0 13 :<85>
>6398 38 ad e9 60 e9 30 8d e9 :<3d>
>63a0 60 c9 35 b0 05 a9 35 8d :<5f>
>63a8 e9 60 60 38 ad e9 60 e9 :<f2>
>63b0 30 8d e9 60 b0 f4 ce e5 :<15>
>63b8 60 60 c9 24 d0 0a 20 70 :<37>
>63c0 63 20 1a 63 20 1a 63 60 :<d9>
>63c8 c9 27 d0 0a 20 91 63 20 :<08>
>63d0 1a 63 20 1a 63 60 c9 14 :<a2>
>63d8 d0 0f ad 00 06 f0 01 60 :<83>
>63e0 a9 0d 8d cf 07 60 ea ea :<73>
>63e8 ea c9 0d d0 f8 ad 00 06 :<7a>
>63f0 f0 14 a9 00 8d 03 06 8d :<f2>
>63f8 00 06 ad 59 15 8d 01 06 :<06>
>6400 ad 21 16 8d 02 06 ad 01 :<a4>
>6408 06 20 32 5c ce 02 06 f0 :<fc>
>6410 01 60 ee 03 06 ac 03 06 :<cf>
>6418 ea ea b9 59 15 10 09 a9 :<62>
>6420 01 8d 00 06 8d cf 07 60 :<93>
>6428 8d 01 06 b9 21 16 8d 02 :<1a>
>6430 06 60 a5 d0 a0 ff c8 d9 :<e4>
>6438 50 64 f0 06 be 50 64 10 :<e6>
>6440 f5 60 b9 70 64 85 d0 60 :<91>
>6448 00 00 00 00 00 00 00 00 :<f4>
>6450 38 3b 08 0b 10 13 18 1b :<21>
>6458 23 36 2e 2c 2f 33 30 2b :<b0>
>6460 28 24 27 1a 25 15 11 0d :<c1>
>6468 1c 14 16 2a ff 00 ff ff :<51>
>6470 31 32 33 34 35 36 37 38 :<4c>
>6478 30 2b 2d 3e 3c 80 81 82 :<54>
>6480 83 4d 4e 47 4b 46 52 53 :<1d>
>6488 42 43 54 4c 00 00 00 00 :<ed>
>6490 00 00 00 00 00 00 00 00 :<85>
>6498 00 00 00 00 00 00 00 00 :<95>
>64a0 00 00 00 00 00 00 00 00 :<a5>
>64a8 00 00 00 00 00 00 00 00 :<b5>
>64b0 a9 00 85 8c 85 8d 85 8e :<03>
>64b8 a9 01 85 e6 8a 25 e6 f0 :<70>
>64c0 13 18 a5 8c 65 26 85 8c :<1e>
>64c8 a5 8d 65 27 85 8d a5 8e :<76>
>64d0 65 28 85 8e 06 26 26 27 :<79>
>64d8 26 28 06 e6 90 de 26 e6 :<03>
>64e0 a9 04 85 e7 98 25 e6 f0 :<ad>
>64e8 13 18 a5 8c 65 26 85 8c :<c9>
>64f0 a5 8d 65 27 85 8d a5 8e :<e6>
>64f8 65 28 85 8e 06 26 26 27 :<cf>
>6500 26 28 06 e6 c6 e7 d0 dc :<44>
>6508 a5 8d 46 8e 6a 46 8e 6a :<42>
>6510 46 8e 6a 46 8e 6a 90 06 :<11>
>6518 69 00 d0 02 e6 8e 85 8c :<05>
>6520 a5 8e 85 8d a5 12 85 8e :<e9>
>6528 60 a0 00 84 e3 84 e4 84 :<5a>
>6530 e5 84 e0 84 e1 84 e2 a5 :<a8>
>6538 8d 85 8e a5 8c 85 8d 84 :<39>
>6540 8c 06 8d 26 8e 06 8d 26 :<b1>
>6548 8e 06 8d 26 8e 06 8d 26 :<82>
>6550 8e a5 41 85 42 a5 40 85 :<88>
>6558 41 84 40 06 41 26 42 06 :<ec>
>6560 41 26 42 06 41 26 42 a9 :<d5>
>6568 08 85 e4 a5 42 06 e4 26 :<ed>
>6570 e5 06 41 2a 10 f7 85 42 :<a6>
>6578 38 a5 8d e5 41 aa a5 8e :<8b>
>6580 e5 42 90 0c 85 8e 86 8d :<34>
>6588 a5 e4 85 e1 a5 e5 85 e2 :<05>
>6590 46 42 66 41 66 40 46 e5 :<a3>
>6598 66 e4 66 e3 b0 2d 38 a5 :<fa>
>65a0 8c e5 40 aa a5 8d e5 41 :<30>
>65a8 a8 a5 8e e5 42 90 e1 85 :<16>
>65b0 8e 84 8d 86 8c 18 a5 e0 :<86>
>65b8 65 e3 85 e0 a5 e1 65 e4 :<ec>
>65c0 85 e1 a5 e2 65 e5 85 e2 :<14>
>65c8 4c 90 65 a4 8e c4 42 90 :<86>
>65d0 1a d0 0e a4 8d c4 41 90 :<fe>
>65d8 12 d0 06 a4 8c c4 40 90 :<96>
>65e0 0a e6 e0 d0 06 e6 e1 d0 :<37>
>65e8 02 e6 e2 60 a4 d4 a6 d3 :<1e>
>65f0 a5 03 85 26 a5 04 85 27 :<f4>
>65f8 a5 05 85 28 20 b0 64 a5 :<9b>
>6600 d0 85 26 a5 d1 85 27 a5 :<9e>
>6608 d2 85 28 4c b3 5e a5 d0 :<64>
>6610 85 8c a5 d1 85 8d a9 00 :<94>
>6618 85 8e a5 03 85 40 a5 04 :<5e>
>6620 85 41 a5 05 85 42 20 9a :<42>
>6628 6a a5 e2 f0 0d 85 8e a5 :<e8>
>6630 e1 85 8d a5 e0 85 8c ea :<70>
>6638 ea 60 85 28 a5 e1 85 27 :<ee>
>6640 a5 e0 85 26 a5 d5 85 8e :<5f>
>6648 a5 d4 85 8d a5 d3 85 8c :<4d>
>6650 4c b3 5e a9 00 85 28 a9 :<03>
>6658 01 85 27 a9 3f 85 26 a5 :<e4>
>6660 d5 85 8e a5 d4 85 8d a5 :<2b>
>6668 d3 85 8c 20 3a 5f a6 8c :<82>
>6670 a4 8d 4c f0 65 a9 00 85 :<6f>
>6678 28 85 27 a9 c7 85 26 a5 :<52>
>6680 d2 85 8e a5 d1 85 8d a5 :<b8>
>6688 d0 85 8c 20 3a 5f 4c 16 :<90>
>6690 66 a5 e0 85 03 a5 e1 85 :<ba>
>6698 04 a5 e2 85 05 60 a5 10 :<1e>
>66a0 a4 11 85 11 84 10 60 00 :<4a>
>66a8 00 00 00 00 00 00 00 ea :<ac>
>66b0 a0 00 84 10 84 11 a5 d1 :<e6>
>66b8 10 08 a5 10 09 04 85 10 :<45>
>66c0 d0 10 f0 08 a5 10 09 02 :<67>
>66c8 85 10 d0 06 a5 d0 c9 c8 :<66>
>66d0 b0 f2 a5 d4 10 08 a5 10 :<43>
>66d8 09 08 85 10 d0 14 f0 12 :<4b>
>66e0 c9 02 90 08 a5 10 09 01 :<81>
>66e8 85 10 d0 06 a5 d3 c9 40 :<52>
>66f0 b0 f2 a5 d7 10 08 a5 11 :<18>
>66f8 09 04 85 11 d0 10 f0 08 :<01>
>6700 a5 11 09 02 85 11 d0 06 :<b4>
>6708 a5 d6 c9 c8 b0 f2 a5 da :<7a>
>6710 10 08 a5 11 09 08 85 11 :<3e>
>6718 d0 14 f0 12 c9 02 90 08 :<d5>
>6720 a5 11 09 01 85 11 d0 06 :<ec>
```

```
>6728 a5 d9 c9 40 b0 f2 a5 10 :<a1>
>6730 f0 05 25 11 f0 09 60 a5 :<69>
>6738 11 d0 04 4c 50 57 ea a5 :<a7>
>6740 d1 aa 29 7f 85 d1 8a 29 :<06>
>6748 80 85 d2 a5 d4 aa 29 7f :<cb>
>6750 85 d4 85 8d 8a 29 80 85 :<50>
>6758 d5 85 8e a5 d7 aa 29 7f :<21>
>6760 85 d7 8a 29 80 85 d8 a5 :<2e>
>6768 da aa 29 7f 85 da 85 27 :<83>
>6770 8a 29 80 85 db 85 28 a5 :<08>
>6778 d3 85 8c a5 d9 85 26 20 :<54>
>6780 3a 5f a5 8d d0 47 a5 8c :<1c>
>6788 d0 43 a5 10 aa 29 04 f0 :<9a>
>6790 0a a9 00 85 d0 85 d1 85 :<dc>
>6798 d2 f0 0f 8a 29 02 f0 0a :<a6>
>67a0 a9 00 85 d1 85 d2 a9 c7 :<52>
>67a8 85 d0 a5 11 aa 29 04 f0 :<b1>
>67b0 0a a9 00 85 d6 85 d7 85 :<5e>
>67b8 d8 f0 0f 8a 29 02 f0 0a :<ed>
>67c0 a9 00 85 d7 85 d8 a9 c7 :<7f>
>67c8 85 d6 4c 60 57 a5 8e 10 :<ab>
>67d0 33 a5 d5 a4 db 85 db 84 :<60>
>67d8 d5 a5 d4 a4 da 85 da 84 :<9d>
>67e0 d4 a5 d3 a4 d9 85 d9 84 :<78>
>67e8 d3 a5 d2 a4 d8 85 d8 84 :<0f>
>67f0 d2 a5 d1 a4 d7 85 d7 84 :<8b>
>67f8 d1 a5 d0 a4 d6 85 d6 84 :<04>
>6800 d0 20 9e 66 a6 8c a4 8d :<1d>
>6808 e8 d0 01 c8 86 40 84 41 :<d5>
>6810 a9 00 85 42 a5 d2 85 8e :<82>
>6818 a5 d1 85 8d a5 d0 85 8c :<25>
>6820 a5 d8 85 28 a5 d7 85 27 :<2a>
>6828 a5 d6 85 26 20 3a 5f a5 :<c4>
>6830 8d d0 44 a5 8c d0 40 a5 :<4e>
>6838 10 aa 29 08 f0 06 a9 00 :<0a>
>6840 85 d4 85 d3 a5 11 29 01 :<fd>
>6848 f0 08 a9 01 85 da a9 3f :<90>
>6850 85 d9 20 da 58 38 a5 d9 :<10>
>6858 e5 d3 aa a5 da e5 d4 85 :<68>
>6860 d4 a0 00 e6 d4 e8 d0 02 :<9b>
>6868 e6 d4 ca d0 05 c6 d4 d0 :<55>
>6870 01 60 20 bc 59 d0 f3 e6 :<a5>
>6878 8c d0 02 e6 8d a5 8e 85 :<0e>
>6880 12 a9 00 85 8e 20 29 65 :<52>
>6888 20 91 66 a5 12 10 03 4c :<e2>
>6890 86 69 a5 10 29 08 f0 59 :<31>
>6898 20 ec 65 a5 8e 10 2f 20 :<0a>
>68a0 0e 66 a5 8d f0 0b c9 02 :<0f>
>68a8 90 01 60 a5 8c c9 40 b0 :<cc>
>68b0 f9 a9 00 8d 00 0d 8d 01 :<fe>
>68b8 0d 8d 02 0d a5 8c 8d 03 :<21>
>68c0 0d a5 8d 8d 04 0d a5 8e :<ae>
>68c8 8d 05 0d 4c 11 69 a5 8d :<0d>
>68d0 d0 d8 a5 8c c9 c8 b0 d2 :<56>
>68d8 a9 00 8d 03 0d 8d 04 0d :<16>
>68e0 8d 05 0d 8d 01 0d 8d 02 :<2e>
>68e8 0d a5 8c 8d 00 0d 4c 11 :<5f>
>68f0 69 a5 10 29 04 d0 a8 a5 :<13>
>68f8 d0 8d 00 0d a9 00 8d 01 :<13>
>6900 0d 8d 02 0d 8d 05 0d a5 :<00>
>6908 d3 8d 03 0d a5 d4 8d 04 :<c3>
>6910 0d a5 11 29 01 f0 48 20 :<9d>
>6918 53 66 a5 8d d0 21 a5 8c :<48>
>6920 c9 c8 b0 1b 8d 06 0d a9 :<0b>
>6928 00 8d 07 0d 8d 08 0d 8d :<ba>
>6930 0b 0d a9 01 8d 0a 0d a9 :<4b>
>6938 3f 8d 09 0d 4c 55 6a 20 :<8a>
>6940 75 66 a9 00 8d 07 0d 8d :<13>
>6948 08 0d 8d 0b 0d a9 c7 8d :<c8>
>6950 06 0d a5 8c 8d 09 0d a5 :<12>
>6958 8d 8d 0a 0d 4c 55 6a a5 :<ee>
>6960 11 29 02 d0 da a5 d6 8d :<a1>
>6968 06 0d a5 d7 8d 07 0d a5 :<e9>
>6970 d8 8d 08 0d a5 d9 8d 09 :<10>
>6978 0d a5 da 8d 0a 0d a5 db :<9e>
>6980 8d 0b 0d 4c 55 6a a5 10 :<4f>
>6988 29 08 f0 58 20 ec 65 a5 :<59>
>6990 8e 10 01 60 a5 8d d0 1d :<be>
>6998 a5 8c c9 c8 b0 17 8d 00 :<e9>
>69a0 0d a9 00 8d 01 0d 8d 02 :<4b>
>69a8 0d 8d 03 0d 8d 04 0d 8d :<3a>
>69b0 05 0d 4c 04 6a 20 75 66 :<26>
>69b8 a5 8d f0 0b c9 02 90 01 :<24>
>69c0 60 a5 8c c9 40 b0 f9 a5 :<93>
>69c8 8c 8d 03 0d a5 8d 8d 04 :<47>
>69d0 0d a9 c7 8d 00 0d a9 00 :<f3>
>69d8 8d 01 0d 8d 02 0d 8d 05 :<d1>
>69e0 0d 4c 04 6a a5 10 29 02 :<ca>
>69e8 d0 cb a5 d0 8d 00 0d a5 :<17>
>69f0 d3 8d 03 0d a5 d4 8d 04 :<d5>
>69f8 0d a5 d1 8d 01 0d 8d 02 :<58>
>6a00 0d 8d 05 0d a5 11 29 01 :<46>
>6a08 f0 42 20 53 66 a5 8e 30 :<2e>
>6a10 1d a9 00 8d 07 0d 8d 08 :<d8>
>6a18 0d 8d 0b 0d a9 01 8d 0a :<3a>
>6a20 0d a9 3f 8d 09 0d a5 8c :<0b>
>6a28 8d 06 0d 4c 55 6a 20 0e :<9c>
>6a30 66 a9 00 8d 06 0d 8d 07 :<d1>
>6a38 0d 8d 08 0d 8d 0b 0d a5 :<59>
>6a40 8d 8d 0a 0d a5 8c 8d 09 :<95>
>6a48 0d 4c 55 6a a5 11 29 04 :<bd>
>6a50 d0 dc 4c 65 69 ad 00 0d :<9c>
>6a58 85 d0 ad 01 0d 85 d1 ad :<86>
>6a60 02 0d 85 d2 ad 03 0d 85 :<28>
>6a68 d3 ad 04 0d 85 d4 ad 05 :<5f>
>6a70 0d 85 d5 ad 06 0d 85 d6 :<2f>
>6a78 ad 07 0d 85 d7 ad 08 0d :<19>
>6a80 85 d8 ad 09 0d 85 d9 ad :<c9>
>6a88 0a 0d 85 da ad 0b 0d 85 :<01>
>6a90 db a9 00 85 12 4c 50 57 :<06>
>6a98 ea ea a0 00 84 e3 84 e4 :<7f>
>6aa0 84 e5 84 e0 84 e1 84 e2 :<70>
>6aa8 a5 8d 85 8e a5 8c 85 8d :<8b>
>6ab0 84 8c 06 8d 26 8e 06 8d :<c1>
>6ab8 26 8e 06 8d 26 8e 06 8d :<97>
>6ac0 26 8e a9 01 85 e3 a5 41 :<17>
```

```
>6ac8 06 e3 26 e4 26 e5 06 40 :<9f>
>6ad0 2a 26 42 10 f3 85 41 38 :<33>
>6ad8 a5 8d e5 41 aa a5 8e e5 :<bc>
>6ae0 42 90 10 85 8e 86 8d a5 :<a8>
>6ae8 e3 85 e0 a5 e4 85 e1 a5 :<45>
>6af0 e5 85 e2 46 42 66 41 66 :<1f>
>6af8 40 46 e5 66 e4 66 e3 b0 :<80>
>6b00 2d 38 a5 8c e5 40 aa a5 :<1a>
>6b08 8d e5 41 a8 a5 8e e5 42 :<f2>
>6b10 90 e1 85 8e 84 8d 86 8c :<21>
>6b18 18 a5 e0 65 e3 85 e0 a5 :<4d>
>6b20 e1 65 e4 85 e1 a5 e2 65 :<e0>
>6b28 e5 85 e2 4c f3 6a a4 8e :<02>
>6b30 c4 42 90 1a d0 0e a4 8d :<51>
>6b38 c4 41 90 12 d0 06 a4 8c :<61>
>6b40 c4 40 90 0a e6 e0 d0 06 :<32>
>6b48 e6 e1 d0 02 e6 e2 60 90 :<6d>
>6b50 0a e6 e0 d0 06 e6 e1 d0 :<a4>
>6b58 02 e6 e2 60 ea a9 00 aa :<7c>
>6b60 78 8d 3f ff 9d 00 80 9d :<27>
>6b68 00 81 9d 00 82 9d 00 83 :<30>
>6b70 9d 00 84 9d 00 85 9d 00 :<d7>
>6b78 86 9d 00 87 9d 00 88 9d :<48>
>6b80 00 89 9d 00 8a 9d 00 8b :<73>
>6b88 9d 00 8c 9d 00 8d 9d 00 :<05>
>6b90 8e 9d 00 8f 9d 00 90 9d :<8c>
>6b98 00 91 9d 00 92 9d 00 93 :<c6>
>6ba0 9d 00 94 9d 00 95 9d 00 :<36>
>6ba8 96 9d 00 97 9d 00 98 9d :<97>
>6bb0 00 99 9d 00 9a 9d 00 9b :<ea>
>6bb8 9d 00 9c 9d 00 9d 9d 00 :<ac>
>6bc0 9e 9d 00 9f 8d 3e ff 58 :<ad>
>6bc8 e8 d0 95 60 ea ea ea ea :<25>
>6bd0 a0 00 78 8d 3f ff b9 00 :<30>
>6bd8 80 99 00 20 b9 00 81 99 :<46>
>6be0 00 21 b9 00 82 99 00 22 :<a1>
>6be8 b9 00 83 99 00 23 b9 00 :<56>
>6bf0 84 99 00 24 8d 3e ff 58 :<9f>
>6bf8 c8 d0 d7 ea ea ea ea ea :<bb>
>6c00 78 8d 3f ff b9 00 85 99 :<ed>
>6c08 00 25 b9 00 86 99 00 26 :<b7>
>6c10 b9 00 87 99 00 27 b9 00 :<7b>
>6c18 88 99 00 28 b9 00 89 99 :<0b>
>6c20 00 29 8d 3e ff 58 c8 d0 :<f7>
>6c28 d7 ea ea ea ea 78 8d 3f :<ef>
>6c30 ff b9 00 8a 99 00 2a b9 :<0a>
>6c38 00 8b 99 00 2b b9 00 8c :<08>
>6c40 99 00 2c b9 00 8d 99 00 :<b4>
>6c48 2d b9 00 8e 99 00 2e 8d :<6b>
>6c50 3e ff 58 c8 d0 d7 ea ea :<6f>
>6c58 ea ea 78 8d 3f ff b9 00 :<1e>
>6c60 8f 99 00 2f b9 00 90 99 :<a2>
>6c68 00 30 b9 00 91 99 00 31 :<36>
>6c70 b9 00 92 99 00 32 b9 00 :<b0>
>6c78 93 99 00 33 8d 3e ff 58 :<b6>
>6c80 c8 d0 d7 ea ea ea ea 78 :<a3>
>6c88 8d 3f ff b9 00 94 99 00 :<01>
>6c90 34 b9 00 95 99 00 35 b9 :<93>
>6c98 00 96 99 00 36 b9 00 97 :<73>
>6ca0 99 00 37 b9 00 98 99 00 :<02>
>6ca8 38 b9 00 99 99 00 39 8d :<5a>
>6cb0 3e ff 58 c8 d0 d1 ea ea :<6c>
>6cb8 ea ea 78 8d 3f ff b9 00 :<e5>
>6cc0 9a 99 00 3a b9 00 9b 99 :<1b>
>6cc8 00 3b b9 00 9c 99 00 3c :<c5>
>6cd0 b9 00 9d 99 00 3d b9 00 :<fb>
>6cd8 9e 99 00 3e b9 00 9f 99 :<ee>
>6ce0 00 3f 8d 3e ff 58 c8 d0 :<a4>
>6ce8 d1 a9 00 85 10 4c ff 5f :<7d>
>6cf0 00 00 00 00 00 00 00 00 :<4d>
>6cf8 00 00 00 00 00 00 ea a2 :<a8>
>6d00 00 a9 00 9d 00 20 9d 01 :<59>
>6d08 20 9d 02 20 9d 03 20 9d :<70>
>6d10 04 20 9d 05 20 9d 06 20 :<97>
>6d18 9d 07 20 8a 69 08 aa d0 :<3f>
>6d20 e0 ae 05 6d e8 e0 3f f0 :<b2>
>6d28 1b 8e 05 6d 8e 08 6d 8e :<5f>
>6d30 0b 6d 8e 0e 6d 8e 11 6d :<6f>
>6d38 8e 14 6d 8e 17 6d 8e 1a :<45>
>6d40 6d aa f0 bd a2 20 8e 05 :<5f>
>6d48 6d 8e 08 6d 8e 0b 6d 8e :<7e>
>6d50 0e 6d 8e 11 6d 8e 14 6d :<08>
>6d58 8e 17 6d 8e 1a 6d aa a0 :<e1>
>6d60 41 99 ff 3e 88 d0 fa 60 :<b8>
>6d68 c9 16 d0 31 a9 80 8d 0a :<fe>
>6d70 59 a0 0e b9 43 5b 99 32 :<c4>
>6d78 59 99 4f 59 99 6f 59 99 :<e1>
>6d80 88 59 99 ac 59 99 bf 59 :<01>
>6d88 99 da 59 99 ef 59 99 0f :<7a>
>6d90 5a 99 28 5a 99 4c 5a 88 :<8b>
>6d98 d0 d9 60 ea ea c9 2a d0 :<1b>
>6da0 37 a9 20 8d 0a 59 a0 0e :<c0>
>6da8 b9 f5 5d 99 32 59 99 4f :<bc>
>6db0 59 99 6f 59 99 88 59 99 :<0c>
>6db8 ac 59 99 bf 59 99 da 59 :<05>
>6dc0 99 ef 59 99 0f 5a 99 28 :<c6>
>6dc8 5a 99 4c 5a 88 d0 d9 a9 :<99>
>6dd0 ea 8d 39 59 60 ea ea ea :<a1>
>6dd8 4c bb 62 00 00 00 a9 80 :<d5>
>6de0 85 9a a9 01 a6 d0 a0 01 :<5d>
>6de8 20 ba ff ad 5e 06 a2 5f :<f3>
>6df0 a0 06 20 bd ff a9 00 20 :<75>
>6df8 d5 ff a9 00 85 9a 60 00 :<67>
```

==========

PROGRAMMENDE

==========

```
GRAFIX ==========================P4
(p) COMMODORE WELT TEAM
================================
(c) by Frank Koster

PLUS4 (C16/116 + 64 KB)
================================
Programmkopf und Programmende
sind aus Platzgruenden wegzulas-
sen.
================================
100 color0,1:color1,2:color4,1:gos
ub1390                          <dd>
110 graphic1,1:graphic0,1:print"da
ten laden j/n":getkeyt$:ift$="j"th
en740                           <dn>
120 gosub1160                   <lk>
130 gosub970:input"anzahl der punk
te,die sie definieren wollen";n:po
ke4096,n                        <fg>
140 fort=0ton-1:graphic0,1:print"k
oordinaten von punkt ";t+1:input"t
iefe";l                         <kl>
150 input"hoehe";h:input"breite";b
:x=-l:a=q:gosub750:x=-h:a=e:gosub7
50                              <lf>
160 x=b:a=i:gosub750:next:h=0   <cf>
170 fort=1ton:graphic0,1:a$=""  <mi>
180 print"zu welchen anderen punkt
en sollen von punkt";t;"linien gez
ogen werden"                    <hf>
190 printt;"->":poke202,peek(202)+
4:poke205,peek(205)-1:input" ";a$:
ifa$=""then210                  <aj>
200 pokep+h,t:pokec+h,val(a$):h=h+
2:a$="":goto190                 <fj>
210 next:pokep+h,0:pokec+h,0:gosub
990                             <ga>
220 graphic0,1:print"sind die date
n korrekt oder sollen aende"    <pe>
230 print"-rungen vorgenommen werd
en j/n":getkeyt$:ift$="n"then530  <mb>
240 gosub1160                   <mj>
250 print"bei den punkten oder bei
 den linien p/l":getkeyt$:ift$="l"
then400                         <mb>
260 graphic0,1:print"koordinaten v
eraendern/punkt loeschen/neuer pun
kt v/l/n"                       <mg>
270 getkeyt$:ift$="l"then380    <kh>
280 ift$="n"then360             <kh>
290 graphic0,1:input"welcher punkt
";t:t=t-1:a=q:gosub770:l=-x      <lj>
300 a=e:gosub770:h=-x:a=i:gosub770
:b=x                            <kb>
310 print"alte werte(ti,ho,br)";l,
h,b                             <eg>
320 input"neue werte(ti,ho,br)";l,
h,b:x=-l:a=q:gosub750            <df>
330 x=-h:a=e:gosub750:x=b:a=i:gosu
b750                            <hm>
340 print"weitere punkte korrigier
en j/n"                         <nl>
350 getkeyt$:ift$="j"then260:else2
20                              <if>
360 t=peek(4096)+1:print"der neue
punkt hat die nummer ";t:poke4096,
t:t=t-1                         <mo>
370 input"tiefe";l:input"hoehe";h:
input"breite";b:x=-l:a=q:gosub750:
goto330                         <bb>
380 graphic0,1:input"welcher punkt
 soll geloescht werden";t:n=peek(4
096)                            <jb>
390 gosub790:h=0:gosub830:h=0:gosu
b1120:goto340                   <on>
400 graphic0,1:print"linien loesch
en/neue linie/a.oder endpunkt vera
endern l/n/v"                   <nb>
410 getkeyt$:ift$="l"then470:elsei
ft$="v"then490                  <ip>
420 graphic0,1:print"welche linie
soll hinzugefuegt werden"       <ih>
430 input"anfangspunkt";x:input"en
dpunkt";y:h=0                   <mb>
440 ifpeek(p+h)<>0then h=h+2:goto4
40                              <lf>
450 pokep+h,x:pokec+h,y:pokep+2+h,
0:pokec+2+h,0                   <nd>
460 print"weitere linien korrigier
en j/n":getkeyt$:ift$="j"then400:e
lse220                          <bd>
470 input"anfangspunkt der zu loes
chenden linie";x:input"endpunkt";y
:h=0                            <ei>
480 gosub870:goto460            <hf>
490 graphic0,1:print"welche linie
soll veraendert werden:"        <lo>
500 input"alter anfangspunkt";x:in
put"alter endpunkt";y           <dm>
510 input"neuer anfangspunkt";l:in
put"neuer endpunkt";b           <nn>
520 h=0:gosub940:goto460        <pf>
530 ifpeek(1016)=2andpeek(1020)=11
then550                         <eb>
540 fort=0to90:poke1015+t,int(sin(
t*π/180)*128+.5):next           <pa>
550 color0,1:color1,2:graphic1,1:p
oke1999,1:poke1536,1:sys24575   <lk>
560 graphic0,1:fort=1319to1328:pok
et,00:next:gosub1160:gosub1390:gra
phic0,1                         <ml>
570 print"verzeichnis der daten j/
n":getkeyt$:ift$="j"thengosub990  <fh>
580 gosub1160:print"daten aendern
j/n":getkeyt$:ift$="j"then250    <fb>
```

```
590 gosub1160:print"daten abspeich
ern j/n":getkeyt$:ift$="j"then670  <ei>
600 gosub1160:print"zurueck zur gr
afik j/n":getkeyt$:ift$="j"then530 <pa>
610 gosub1160:print"neue daten lad
en j/n":getkeyt$:ift$="j"then740   <dl>
620 print" bewegungsablaeufe def.
j/n":getkeyt$:ift$="j"thengosub117
0:goto640                          <gd>
630 gosub1160:print"  daten des be
weg.ablaufs j/n":getkeyt$:ift$="j"
thengosub1230                      <lg>
640 gosub1160:print"  beweg.ablauf
 aendern":getkeyt$:ift$="j"then 13
10                                 <mi>
650 gosub1160:print"programm neu s
tarten j/n":getkeyt$:ift$="j"then
run                                <bb>
660 gosub1160:print"3-d programm b
eenden j/n":getkeyt$:ift$="j"then
new:else 570                       <cg>
670 graphic0,1:print"kassette/disk
ette k/d":gosub1160:getkeyt$       <of>
680 ift$="d"thenpoke208,8:elsepoke
208,1:print"nach spulen taste drue
cken":getkeyt$                     <ia>
690 gosub1160:sys23304:getkeyt$:gr
aphic0,1:goto540                   <aj>
700 graphic0,1:print"kassette/disk
ette k/d":fort=0to35:next:getkeyt$ <ed>
710 ift$="d"thenpoke208,8:goto730:
elsepoke208,1                      <po>
720 print" sie koennen jetzt spule
n--danach taste  druecken":fort=0t
o30:next:getkeyt$                  <mj>
730 fort=0to50:next:sys28126:graph
ic0,1:goto540                      <df>
740 gosub970:goto700               <np>
750 ifx<0thenx=-x+32768            <je>
760 hb=int(x/256):lb=x-hb*256:poke
a+2*t,lb:pokea+1+2*t,hb:return     <mn>
770 lb=peek(a+2*t):hb=peek(a+1+2*t
):x=hb*256+lb:ifx>32767thenx=-(x-3
2768)                              <km>
780 return                         <km>
790 forl=tton:pokeq-2+2*l,peek(q+2
*l):pokeq-1+2*l,peek(q+2*l+1)      <ff>
800 pokee-2+2*l,peek(e+2*l):pokee-
1+2*l,peek(e+2*l+1)                <em>
810 pokei-2+2*l,peek(i+2*l):pokei-
1+2*l,peek(i+2*l+1):next:poke4096,
n-1                                <ko>
820 return                         <pm>
830 p1=peek(p+h):p2=peek(c+h):ifp1
=t or p2=tthenb=h+2:goto850        <kn>
840 ifp1=0thenreturn:elseh=h+2:got
o830                               <fk>
850 p1=peek(p+b):p2=peek(c+b):poke
p-2+b,p1:pokec-2+b,p2              <dg>
860 ifp1=0thenh=h+0:goto830:elseb=
b+2:goto850                        <dc>
870 p1=peek(p+h):p2=peek(c+h):ifp1
=0then return                      <pn>
880 ifp1<>xthen920                 <pb>
890 ifp2<>ythenh=h+2:goto870       <ip>
900 p1=peek(p+2+h):p2=peek(c+2+h):
pokep+h,p1:pokec+h,p2:ifp1=0then r
eturn                              <km>
910 h=h+2:goto900                  <lc>
920 ifp1<>ythenh=h+2:goto870       <hi>
930 ifp2<>xthenh=h+2:goto870:else9
00                                 <ii>
940 p1=peek(p+h):p2=peek(c+h):ifp1
=0thenreturn                       <cl>
950 if(p1=x and p2=y) or (p2=x and
 p1=y)then pokep+h,1:pokec+h,b:ret
urn                                <dl>
960 h=h+2:goto940                  <hd>
970 graphic0,1:input"name des obje
kts";a$:h=len(a$):fort=1toh         <bc>
980 poke1630+t,asc(mid$(a$,t,1)):n
ext:poke1630,h:return              <fm>
990 print"daten der punkte oder de
r linien":getkeyt$:ift$="l"then w=
0:goto1070                         <ep>
1000 w=1                           <en>
1010 graphic0,1:print"  daten alle
r punkte":n=peek(4096):t=0         <lh>
1020 print:print"  punkt    tiefe
  hoehe   breite"                  <jg>
1030 fort=0ton-1:y=2*t:a=q:gosub77
0:l=-x                             <fg>
1040 a=e:gosub770:h=-x:a=i:gosub77
0:b=x:print                        <mf>
1050 printt+1,l,h,b:poke205,peek(2
05)-1:poke201,peek(201)-1          <ii>
1060 getkeyt$:next:gosub1160:graph
ic0,1:ifw=0thenreturn              <hh>
1070 graphic0,1:print" daten aller
  linien":h=0                      <en>
1080 printchr$(17)"       vonCCCCCC
CCCCC>zu"                          <ha>
1090 p1=peek(p+h):p2=peek(c+h):h=h
+2:ifp1=0then1110                  <po>
1100 poke200,6:poke202,6:printp1;"
 ",p2:getkeyt$:goto1090            <af>
1110 ifw=1thenreturn:else1010      <al>
1120 p1=peek(p+h):p2=peek(c+h):ifp
1=0thenreturn                      <ei>
1130 ifp1>tthenpokep+h,p1-1        <lf>
1140 ifp2>tthenpokec+h,p2-1        <ij>
1150 h=h+2:goto1120                <la>
1160 fort=0to30:next:return        <pm>
1170 graphic0,1:h=0:print" bewegun
gen def."                          <fp>
1180 print" ";h+1;".schritt   welc
```

```
he taste?(t.druecken)":gosub1160   <ao>
1190 lb=peek(198):iflb=64then1190:
else iflb=1thenpoke5465+h,255:goto
1230                               <gg>
1200 ifpeek(198)=lbthen 1200:else
fort=0to9:poke1319+t,0:next        <fe>
1210 iflb=21orlb=17thenprint"  unz
ulaessig!!!":goto1180 .            <fm>
1220 input" fuer wiev. berechn.";h
b:poke5465+h,lb:poke5665+h,hb:h=h+
1:goto1180                         <lp>
1230 graphic0,1:print" daten der b
eweg.ablaeufe":h=0                 <co>
1240 poke208,peek(5465+h):ifpeek(2
08)=255then return                 <hj>
1250 getkeyt$:sys25650:lb=peek(208
)                                  <fp>
1260 iflb=128 thenprinth+1;"    cu
rsor rechts ",peek(5665+h):h=h+1:g
oto1240                            <nd>
1270 iflb=129 thenprinth+1;"    cu
rsor links  ",peek(5665+h):h=h+1:g
oto1240                            <pj>
1280 iflb=131 thenprinth+1;"    cu
rsor unten  ",peek(5665+h):h=h+1:g
oto1240                            <op>
1290 iflb=130 thenprinth+1;"    cu
rsor oben   ",peek(5665+h):h=h+1:g
oto1240                            <lc>
1300 printh+1;"    ";chr$(lb),peek
(5665+h):h=h+1:goto1240            <bj>
1310 graphic0,1:print" beweg.ablae
ufe aendern":h=0:gosub1230         <hk>
1320 print" der wievielte progr.sc
hritt soll":print"  geaendert werd
en"                                <eb>
1330 inputh:print" neuer schritt":
fort=1319to1328:poket,0:next:a=pee
k(5464+h)                          <cg>
1340 gosub1160:lb=peek(198):iflb=6
4then1340                          <ii>
1350 gosub1160:ifpeek(198)=lbthen1
350:elsefort=1319to1328:poket,0:ne
xt                                 <ce>
1360 input" anzahl der berechnunge
n";hb:poke5464+h,lb:poke5664+h,hb  <gi>
1370 ifa=255thenpoke5465+h,255     <dh>
1380 print" noch mehr aendern j/n"
:getkeyt$:ift$="j"then 1310:else 5
70                                 <cd>
1390 q=4097:e=4353:i=4609:p=4865:c
=4866:return                       <mn>
==================================
P R O G R A M M E N D E
==================================
```

```
GRAFIX.CBM=======================p4
(p) COMMODORE WELT TEAM
==================================
(c) by Frank Koster

Plus4 (C16/116 + 64 KB)
==================================
Das Programm ist mit dem Maschinen-
sprachmonitor TEDMON unter Zuhilfe-
nahme des Pruefsummenprogrammes
CHECKMON einzugeben und auf Disket-
te mit s"grafix.cbm",8,1000,1800
abzuspeichern, auf Kassette mit
s"grafix.cbm",1,1000,1800.
==================================
>1000 30 24 80 24 80 00 00 28 :<c7>
>1008 00 70 00 70 00 28 00 00 :<81>
>1010 00 24 80 24 80 00 00 22 :<b0>
>1018 00 40 00 40 00 22 00 00 :<07>
>1020 00 24 80 24 80 00 00 28 :<d7>
>1028 00 70 00 70 00 28 00 00 :<c1>
>1030 00 24 80 24 80 00 00 22 :<70>
>1038 00 40 00 40 00 22 00 00 :<47>
>1040 00 24 80 84 80 5c 80 24 :<68>
>1048 80 24 80 84 80 5c 80 24 :<f8>
>1050 80 24 80 5c 80 84 80 24 :<24>
>1058 80 24 80 5c 80 84 80 24 :<14>
>1060 80 ff 00 ff 00 ff 00 ff :<a6>
>1068 00 ff 00 ff 00 ff 00 ff :<36>
>1070 00 ff 00 ff 00 ff 00 ff :<46>
>1078 00 ff 00 ff 00 ff 00 ff :<d5>
>1080 00 ff 00 ff 00 ff 00 ff :<66>
>1088 00 ff 00 ff 00 ff 00 ff :<76>
>1090 00 ff 00 ff 00 ff 00 ff :<86>
>1098 00 ff 00 ff 00 ff 00 ff :<96>
>10a0 00 ff 00 ff 00 ff 00 ff :<a6>
>10a8 00 ff 00 ff 00 ff 00 ff :<b6>
>10b0 00 ff 00 ff 00 ff 00 ff :<c6>
>10b8 00 ff 00 ff 00 ff 00 ff :<d6>
>10c0 00 ff 00 ff 00 ff 00 ff :<e6>
>10c8 00 ff 00 ff 00 ff 00 ff :<f6>
>10d0 00 ff 00 ff 00 ff 00 ff :<07>
>10d8 00 ff 00 ff 00 ff 00 ff :<17>
>10e0 00 ff 00 ff 00 ff 00 ff :<27>
>10e8 00 ff 00 ff 00 ff 00 ff :<37>
>10f0 00 ff 00 ff 00 ff 00 ff :<47>
>10f8 00 ff 00 ff 00 ff 00 01 :<57>
>1100 01 42 80 72 80 78 80 78 :<87>
>1108 80 32 80 32 00 78 00 78 :<c2>
>1110 00 72 00 42 00 48 00 48 :<4d>
>1118 00 2c 00 2c 80 48 80 48 :<ff>
>1120 80 42 80 72 80 78 80 78 :<c8>
>1128 80 32 80 32 00 78 00 78 :<82>
>1130 00 72 00 42 00 48 00 48 :<0d>
>1138 00 2c 00 2c 80 48 80 48 :<3e>
>1140 80 38 80 38 80 08 80 08 :<6a>
>1148 80 38 80 38 80 08 80 08 :<3a>
>1150 80 08 00 08 00 38 00 38 :<a5>
```

```
>1158 00 08 00 08 00 38 00 38 :<15>
>1160 00 ff 00 ff 00 ff 00 ff :<27>
>1168 00 ff 00 ff 00 ff 00 ff :<37>
>1170 00 ff 00 ff 00 ff 00 ff :<47>
>1178 00 ff 00 ff 00 ff 00 ff :<56>
>1180 00 ff 00 ff 00 ff 00 ff :<67>
>1188 00 ff 00 ff 00 ff 00 ff :<77>
>1190 00 ff 00 ff 00 ff 00 ff :<87>
>1198 00 ff 00 ff 00 ff 00 ff :<97>
>11a0 00 ff 00 ff 00 ff 00 ff :<a7>
>11a8 00 ff 00 ff 00 ff 00 ff :<b7>
>11b0 00 ff 00 ff 00 ff 00 ff :<c7>
>11b8 00 ff 00 ff 00 ff 00 ff :<d7>
>11c0 00 ff 00 ff 00 ff 00 ff :<e7>
>11c8 00 ff 00 ff 00 ff 00 ff :<f7>
>11d0 00 ff 00 ff 00 ff 00 ff :<08>
>11d8 00 ff 00 ff 00 ff 00 ff :<18>
>11e0 00 ff 00 ff 00 ff 00 ff :<28>
>11e8 00 ff 00 ff 00 ff 00 ff :<38>
>11f0 00 ff 00 ff 00 ff 00 ff :<48>
>11f8 00 ff 00 ff 00 ff 00 01 :<58>
>1200 00 18 00 18 00 18 00 18 :<f1>
>1208 00 18 00 18 00 18 00 18 :<20>
>1210 00 18 00 18 00 18 00 18 :<10>
>1218 00 18 00 18 00 18 00 18 :<42>
>1220 00 18 80 18 80 18 80 18 :<58>
>1228 80 18 80 18 80 18 80 18 :<c8>
>1230 80 18 80 18 80 18 80 18 :<b8>
>1238 80 18 80 18 80 18 80 18 :<28>
>1240 80 18 00 18 00 18 00 18 :<f2>
>1248 00 18 80 18 80 18 80 18 :<88>
>1250 80 18 00 18 00 18 00 18 :<12>
>1258 00 18 80 18 80 18 80 18 :<e8>
>1260 80 ff 00 ff 00 ff 00 ff :<a8>
>1268 00 ff 00 ff 00 ff 00 ff :<38>
>1270 00 ff 00 ff 00 ff 00 ff :<48>
>1278 00 ff 00 ff 00 ff 00 ff :<57>
>1280 00 ff 00 ff 00 ff 00 ff :<68>
>1288 00 ff 00 ff 00 ff 00 ff :<78>
>1290 00 ff 00 ff 00 ff 00 ff :<88>
>1298 00 ff 00 ff 00 ff 00 ff :<98>
>12a0 00 ff 00 ff 00 ff 00 ff :<a8>
>12a8 00 ff 00 ff 00 ff 00 ff :<b8>
>12b0 00 ff 00 ff 00 ff 00 ff :<c8>
>12b8 00 ff 00 ff 00 ff 00 ff :<d8>
>12c0 00 ff 00 ff 00 ff 00 ff :<e8>
>12c8 00 ff 00 ff 00 ff 00 ff :<f8>
>12d0 00 ff 00 ff 00 ff 00 ff :<09>
>12d8 00 ff 00 ff 00 ff 00 ff :<19>
>12e0 00 ff 00 ff 00 ff 00 ff :<29>
>12e8 00 ff 00 ff 00 ff 00 ff :<39>
>12f0 00 ff 00 ff 00 ff 00 ff :<49>
>12f8 00 ff 00 ff 00 ff 00 01 :<59>
>1300 01 01 02 01 10 01 11 02 :<8c>
>1308 03 02 12 03 04 03 13 04 :<f1>
>1310 05 04 14 05 06 05 15 06 :<01>
>1318 07 06 16 07 08 07 17 08 :<12>
>1320 09 08 18 09 0a 09 19 0a :<12>

>1328 0b 0a 1a 0b 0c 0b 1b 0c :<32>
>1330 0d 0c 1c 0d 0e 0d 1d 0e :<01>
>1338 0f 0e 1e 0f 10 0f 1f 10 :<52>
>1340 20 12 13 13 14 14 15 15 :<4c>
>1348 16 16 17 17 18 18 19 19 :<4e>
>1350 1a 1a 1b 1b 1c 1c 1d 1d :<5e>
>1358 1e 1e 1f 1f 20 11 12 11 :<c4>
>1360 20 21 22 21 24 21 25 22 :<88>
>1368 23 22 26 23 24 23 27 24 :<5d>
>1370 28 25 26 25 28 26 27 27 :<4a>
>1378 28 29 2a 29 2c 29 2d 2a :<b7>
>1380 2b 2a 2e 2b 2c 2b 2f 2c :<ab>
>1388 30 2d 2e 2e 2f 2f 30 2d :<0b>
>1390 30 00 00 ff 00 ff 00 ff :<bb>
>1398 00 ff 00 ff 00 ff 00 ff :<99>
>13a0 00 ff 00 ff 00 ff 00 ff :<a9>
>13a8 00 ff 00 ff 00 ff 00 ff :<b9>
>13b0 00 ff 00 ff 00 ff 00 ff :<c9>
>13b8 00 ff 00 ff 00 ff 00 ff :<d9>
>13c0 00 ff 00 ff 00 ff 00 ff :<e9>
>13c8 00 ff 00 ff 00 ff 00 ff :<f9>
>13d0 00 ff 00 ff 00 ff 00 ff :<0a>
>13d8 00 ff 00 ff 00 ff 00 ff :<1a>
>13e0 00 ff 00 ff 00 ff 00 ff :<2a>
>13e8 00 ff 00 ff 00 ff 00 ff :<3a>
>13f0 00 ff 00 ff 00 ff 00 ff :<4a>
>13f8 00 ff 00 ff 00 ff 00 01 :<5a>
>1400 01 e7 00 ff 00 ff 00 ff :<3a>
>1408 00 ff 00 ff 00 ff 00 ff :<79>
>1410 00 ff 00 ff 00 ff 00 ff :<89>
>1418 00 ff 00 ff 00 ff 00 ff :<99>
>1420 00 ff 00 ff 00 ff 00 ff :<a9>
>1428 00 ff 00 ff 00 ff 00 ff :<b9>
>1430 00 ff 00 ff 00 ff 00 ff :<c9>
>1438 00 ff 00 ff 00 ff 00 ff :<d9>
>1440 00 ff 00 ff 00 ff 00 ff :<e9>
>1448 00 ff 00 ff 00 ff 00 ff :<f9>
>1450 00 ff 00 ff 00 ff 00 ff :<0a>
>1458 00 ff 00 ff 00 ff 00 ff :<1a>
>1460 00 ff 00 ff 00 ff 00 ff :<2a>
>1468 00 ff 00 ff 00 ff 00 ff :<3a>
>1470 00 ff 00 ff 00 ff 00 ff :<4a>
>1478 00 ff 00 ff 00 ff 00 ff :<59>
>1480 00 ff 00 ff 00 ff 00 ff :<6a>
>1488 00 ff 00 ff 00 ff 00 ff :<7a>
>1490 00 ff 00 ff 00 ff 00 ff :<8a>
>1498 00 ff 00 ff 00 ff 00 ff :<9a>
>14a0 00 ff 00 ff 00 ff 00 ff :<aa>
>14a8 00 ff 00 ff 00 ff 00 ff :<ba>
>14b0 00 ff 00 ff 00 ff 00 ff :<ca>
>14b8 00 ff 00 ff 00 ff 00 ff :<da>
>14c0 00 ff 00 ff 00 ff 00 ff :<ea>
>14c8 00 ff 00 ff 00 ff 00 ff :<fa>
>14d0 00 ff 00 ff 00 ff 00 ff :<0b>
>14d8 00 ff 00 ff 00 ff 00 ff :<1b>
>14e0 00 ff 00 ff 00 ff 00 ff :<2b>
>14e8 00 ff 00 ff 00 ff 00 ff :<3b>
>14f0 00 ff 00 ff 00 ff 00 ff :<4b>
```

```
>14f8 00 ff 00 ff 00 ff 00 01 :<5b>
>1500 01 e7 00 ff 00 ff 00 ff :<3b>
>1508 00 ff 00 ff 00 ff 00 ff :<7a>
>1510 00 ff 00 ff 00 ff 00 ff :<8a>
>1518 00 ff 00 ff 00 ff 00 ff :<9a>
>1520 00 ff 00 ff 00 ff 00 ff :<aa>
>1528 00 ff 00 ff 00 ff 00 ff :<ba>
>1530 00 ff 00 ff 00 ff 00 ff :<ca>
>1538 00 ff 00 ff 00 ff 00 ff :<da>
>1540 00 ff 00 ff 00 ff 00 ff :<ea>
>1548 00 ff 00 ff 00 ff 00 ff :<fa>
>1550 00 ff 00 ff 00 ff 00 ff :<0b>
>1558 00 23 16 28 33 ff 00 ff :<36>
>1560 00 ff 00 ff 00 ff 00 ff :<2b>
>1568 00 ff 00 ff 00 ff 00 ff :<3b>
>1570 00 ff 00 ff 00 ff 00 ff :<4b>
>1578 00 ff 00 ff 00 ff 00 ff :<5a>
>1580 00 ff 00 ff 00 ff 00 ff :<6b>
>1588 00 ff 00 ff 00 ff 00 ff :<7b>
>1590 00 ff 00 ff 00 ff 00 ff :<8b>
>1598 00 ff 00 ff 00 ff 00 ff :<9b>
>15a0 00 ff 00 ff 00 ff 00 ff :<ab>
>15a8 00 ff 00 ff 00 ff 00 ff :<bb>
>15b0 00 ff 00 ff 00 ff 00 ff :<cb>
>15b8 00 ff 00 ff 00 ff 00 ff :<db>
>15c0 00 ff 00 ff 00 ff 00 ff :<eb>
>15c8 00 ff 00 ff 00 ff 00 ff :<fb>
>15d0 00 ff 00 ff 00 ff 00 ff :<0c>
>15d8 00 ff 00 ff 00 ff 00 ff :<1c>
>15e0 00 ff 00 ff 00 ff 00 ff :<2c>
>15e8 00 ff 00 ff 00 ff 00 ff :<3c>
>15f0 00 ff 00 ff 00 ff 00 ff :<4c>
>15f8 00 ff 00 ff 00 ff 00 01 :<5c>
>1600 01 e7 00 ff 00 ff 00 ff :<3c>
>1608 00 ff 00 ff 00 ff 00 ff :<7b>
>1610 00 ff 00 ff 00 ff 00 ff :<8b>
>1618 00 ff 00 ff 00 ff 00 ff :<9b>
>1620 00 01 01 08 c8 ff 00 ff :<78>
>1628 00 ff 00 ff 00 ff 00 ff :<bb>
>1630 00 ff 00 ff 00 ff 00 ff :<cb>
>1638 00 ff 00 ff 00 ff 00 ff :<db>
>1640 00 ff 00 ff 00 ff 00 ff :<eb>
>1648 00 ff 00 ff 00 ff 00 ff :<fb>
>1650 00 ff 00 ff 00 ff 00 ff :<0c>
>1658 00 ff 00 ff 00 ff 00 ff :<1c>
>1660 00 ff 00 ff 00 ff 00 ff :<2c>
>1668 00 ff 00 ff 00 ff 00 ff :<3c>
>1670 00 ff 00 ff 00 ff 00 ff :<4c>
>1678 00 ff 00 ff 00 ff 00 ff :<5b>
>1680 00 ff 00 ff 00 ff 00 ff :<6c>
>1688 00 ff 00 ff 00 ff 00 ff :<7c>
>1690 00 ff 00 ff 00 ff 00 ff :<8c>
>1698 00 ff 00 ff 00 ff 00 ff :<9c>
>16a0 00 ff 00 ff 00 ff 00 ff :<ac>
>16a8 00 ff 00 ff 00 ff 00 ff :<bc>
>16b0 00 ff 00 ff 00 ff 00 ff :<cc>
>16b8 00 ff 00 ff 00 ff 00 ff :<dc>
>16c0 00 ff 00 ff 00 ff 00 ff :<ec>
>16c8 00 ff 00 ff 00 ff 00 ff :<fc>
>16d0 00 ff 00 ff 00 ff 00 ff :<0d>
>16d8 00 ff 00 ff 00 ff 00 ff :<1d>
>16e0 00 ff 00 ff 00 ff 00 ff :<2d>
>16e8 00 ff 00 ff 00 ff 00 ff :<3d>
>16f0 00 ff 00 ff 00 ff 00 ff :<4d>
>16f8 00 ff 00 ff 00 ff 00 01 :<5d>
>1700 01 e7 00 ff 00 ff 00 ff :<3d>
>1708 00 ff 00 ff 00 ff 00 ff :<7c>
>1710 00 ff 00 ff 00 ff 00 ff :<8c>
>1718 00 ff 00 ff 00 ff 00 ff :<9c>
>1720 00 ff 00 ff 00 ff 00 ff :<ac>
>1728 00 ff 00 ff 00 ff 00 ff :<bc>
>1730 00 ff 00 ff 00 ff 00 ff :<cc>
>1738 00 ff 00 ff 00 ff 00 ff :<dc>
>1740 00 ff 00 ff 00 ff 00 ff :<ec>
>1748 00 ff 00 ff 00 ff 00 ff :<fc>
>1750 00 ff 00 ff 00 ff 00 ff :<0d>
>1758 00 ff 00 ff 00 ff 00 ff :<1d>
>1760 00 ff 00 ff 00 ff 00 ff :<2d>
>1768 00 ff 00 ff 00 ff 00 ff :<3d>
>1770 00 ff 00 ff 00 ff 00 ff :<4d>
>1778 00 ff 00 ff 00 ff 00 ff :<5c>
>1780 00 ff 00 ff 00 ff 00 ff :<6d>
>1788 00 ff 00 ff 00 ff 00 ff :<7d>
>1790 00 ff 00 ff 00 ff 00 ff :<8d>
>1798 00 ff 00 ff 00 ff 00 ff :<9d>
>17a0 00 ff 00 ff 00 ff 00 ff :<ad>
>17a8 00 ff 00 ff 00 ff 00 ff :<bd>
>17b0 00 ff 00 ff 00 ff 00 ff :<cd>
>17b8 00 ff 00 ff 00 ff 00 ff :<dd>
>17c0 00 ff 00 ff 00 ff 00 ff :<ed>
>17c8 00 ff 00 ff 00 ff 00 ff :<ff>
>17d0 00 ff 00 ff 00 ff 00 ff :<0e>
>17d8 00 ff 00 ff 00 ff 00 ff :<1e>
>17e0 00 ff 00 ff 00 ff 00 ff :<2e>
>17e8 00 ff 00 ff 00 ff 00 ff :<3e>
>17f0 00 ff 00 ff 00 ff 00 ff :<4e>
>17f8 00 ff 00 ff 00 ff 00 01 :<5e>
==========
P R O G R A M M E N D E
==========
```

*Fortsetzung von Seite 37*

## GESCHWINDIGKEITS-STEIGERUNG

Die Rechengeschwindigkeit im L-Modus kann erhöht werden, wenn der zu löschende Bereich des Grafikbildschirm verkleinert wird. Dies kann geschehen durch:

```
POKE DEC("6D45"),HA
POKE DEC("6D26"),HE
```

Mit HA sei hierbei das Highbyte der Adresse, ab der gelöscht werden soll, bezeichnet. HE sei das Highbyte der Adresse, bis zu der gelöscht werden soll. Es ist zu beachten, daß HA größer oder gleich 20 zu sein hat und HE kleiner gleich 63. □

# HANDELSREISE

Schauplatz ist England im 18. Jahrhundert. Sie besitzen ein Handelschiff, jedoch auch einen Berg von Schulden. Wenn Sie eine glückliche Hand bei ihren Transaktionen haben, erwartet Sie das große Geld, ansonsten der Ruin.

Ihr Schiff kann in fünf verschiedene Häfen einlaufen, wo Sie Ihre Ware ein- und verkaufen können. Im Heimatort "Swansee" haben Sie Ihr Bankkonto mit mehreren tausend Pfund Schulden aufgrund des Schiffkaufes. Um Ware zu kaufen, sollten Sie noch einige tausend Pfund aufnehmen. Doch vergessen Sie nicht, je höher die Schulden, desto mehr wachsen die Zinsen. Bei 25.000 Pfund wird Ihr Schiff samt Ladung gepfändet. Wenn Sie genügend Gewinn gemacht haben, gilt es, schnellstens die Schulden zu tilgen. Sie leben vom Handel. Dieses bedeutet, Ware billig einzukaufen und woanders teuer verkaufen. Jedoch ist dieses auch mit Risiko verbunden. Auf dem Meer lauern Piraten und Stürme, welche Ihnen Ihre Ladung abspenstig machen wollen.

Schiffsreparaturen kosten Ihr Bargeld. Glücksspiele in Kneipen bringen Ihnen Gewinn oder Verlust. Jedoch gibt es auch günstige Gelegenheiten: Gönner zahlen für Sie einen Teil Ihrer Schulden. Handel mit fremden Schiffen bringt Ihnen ganz schön etwas ein, doch Vorsicht, es könnten auch Piraten sein. Sicherheit vor ihnen haben Sie durch Black Jacks Kanone, doch dürfen Sie diesem ein ordentliches Sümmchen dafür berappen. Haben Sie es geschafft, sich ein Vermögen von 800.000 Pfund anzusammeln, dürfen Sie sich zur Ruhe setzen.

Variablenbedeutung:

| | | |
|---|---|---|
| RU | = | Anzahl der Runden |
| G | = | Guthaben |
| SC | = | Schulden |
| LP | = | Ladeplatz |
| A | = | Allgemeines |
| W | = | Waffen |
| K | = | Kohle |
| R | = | Rum |
| E | = | Edelsteine |
| KE | = | Kanone (nicht-)vorhanden |
| P (1 - 5) | = | Preisfestsetzung |
| P | = | Preisberechnung |
| XG | = | x-mal geliehen . |
| LI | = | Limit |
| SE $ | = | Städte |
| S $ | = | Standort |
| X | = | Anzahl der Verletzten |
| TR | = | Verlust durch Trickbetrüger |
| GV | = | Gewinn/Verlust bei Glücksspiel |
| GZ | = | Gewinnzahl |
| PK | = | Preis der Kanone |
| VD | = | Verdienst durch Geschäft mit Kapitän des fremden Schiffes |
| GR | = | Schuldnachlaß durch Gönner |
| MR | = | Kosten der Mastreparatur |
| SR | = | Kosten der Segelreparatur |
| RR | = | Kosten der Rumpfreparatur |
| GR | = | Gesamtkosten der Reparatur |

```
10 rem handelsfahrt =====plus4 <lg>
20 rem (p) 7/87 commodore welt <pk>
30 rem ======================== <ci>
40 rem (c) 1986 by             <lo>
50 rem      claus-d. heineck   <mp>
60 rem                         <ml>
70 rem basic v3.5              <cd>
80 rem plus4 (c16/116 + 64kb)  <dl>
90 rem ======================== <ge>
100 gosub 5750                 <mc>
110 fori=1to22:qq$=qq$+c4$:nex
t:qq$=qq$+chr$(27)+"q":ql$=""  <gp>
120 fori=1to7:ql$=ql$+c1$:next <gm>
130 printchr$(14):scnclr:color
0,15,3:color4,15,3:color1,8    <pi>
140 printtab(128)"DIE"s2$"GROS
SE"s2$"HANDELSFAHRT            <cb>
150 print".......";:xx$=zn$:zz
=25:gosub5720:print            <ol>
160 printtab(135)"Written by"  <ck>
170 printtab(52)"Claus-D. Hein
eck"                           <ei>
180 printtab(240)" "rn$fl$"Dei
n Name"fo$;rf$                 <kc>
190 print                      <fh>
200 inputne$                   <ae>
210 forx=1to1000:next          <kp>
220 gosub5540                  <dh>
230 ru=0                       <gp>
240 g=10*int(rnd(1)*90)        <eo>
250 ifg<350org>750then240      <ji>
260 sc=100*int(rnd(1)*100)     <ak>
270 ifsc<4000orsc>9000then260  <kg>
280 a=0:w=0:k=0:r=0:e=0:ru=0:l
p=50:ke=0:xg=0:se$="1"         <bj>
290 gosub5450                  <ho>
300 gosub5340                  <ig>
310 printhe$                   <ib>
320 printtab(240):printtab(240
)                              <hb>
330 printtab(40)fl$"Druecke"fo
$".............Preise:"       <gg>
340 print"..............."     <lg>
350 print"1 fuer Kaufen.......
Allgemeines £"                 <bn>
360 print"2 fuer Verk.........
Waffen......£"                 <kh>
370 print"3 fuer Segeln.......
Kohle.......£"                 <mg>
380 print"4 fuer Bank-........
Rum.........£"                 <gh>
390 print".....besuch.........
Edelsteine..£"                 <lm>
400 printtab(80)chr$(27)"q"    <he>
410 printhe$                   <of>
420 printtab(131);s$           <fb>
430 printtab(51)"......."ql$;g <ed>
440 printtab(51)"......"left$(
ql$,6);sc                      <oj>
450 printtab(51)"...."left$(ql
$,4);lp                        <jl>
460 printhe$                   <bh>
470 printtab(232)"...."c1$c1$c
1$;a                           <bd>
480 printtab(32)"...."c1$c1$c1
$;w                            <ab>
490 printtab(32)"...."c1$c1$c1
$;k                            <pp>
500 printtab(32)"...."c1$c1$c1
$;r                            <ba>
510 printtab(32)"...."c1$c1$c1
$;e                            <an>
520 printtab(233);p(1)         <be>
530 printtab(33);p(2)          <on>
540 printtab(33);p(3)          <pi>
550 printtab(33);p(4)          <ad>
560 printtab(33);p(5)          <ao>
570 printhe$                   <ig>
580 printtab(240):printtab(240
)                              <hg>
590 printtab(40)fl$"Druecke"fo
$"....."                       <hj>
600 printtab(40)"1 fuer Kaufen
 "                             <ae>
610 print"2 fuer Verk..."      <oh>
620 print"3 fuer Segeln "      <oo>
630 ifse$>"1"thengoto 5510     <oa>
640 print"4 fuer Bank-.."      <ok>
650 print".......besuch "      <gf>
660 rem      printtab(80)".... <ia>
670 getd$                      <kj>
680 ifse$>"1"andd$>"3"then670  <jk>
690 ifd$="1"andlp>0then740     <dm>
700 ifd$="2"andlp<50then1200   <kg>
710 ifd$="3"then1660           <fi>
720 ifd$="4"then1920           <gc>
730 goto670                    <ac>
740 printhe$                   <da>
750 printtab(240):printtab(240
)                              <ca>
760 printtab(40)"Was willst Du
 "                             <gk>
770 printtab(40)"kaufen ?....
."                             <bp>
780 print"............."       <ec>
790 print"(a/w/k/r/e).."       <nj>
800 print"............."       <fg>
810 print"............."       <ga>
820 getk$                      <eh>
830 ifk$<>"a"andk$<>"w"andk$<>
"k"andk$<>"r"andk$<>"e"then820 <lg>
840 ifk$="a"thenp=p(1)         <ph>
850 ifk$="w"thenp=p(2)         <bi>
860 ifk$="k"thenp=p(3)         <bh>
870 ifk$="r"thenp=p(4)         <cj>
```

```
880 ifk$="e"thenp=p(5)                    <ch>
890 printhe$                              <mh>
900 printtab(240):printtab(240
)                                         <lh>
910 printtab(40)"Wieviel wills
t"                                        <bc>
920 printtab(40)"Du kaufen ?..
.."                                       <hc>
930 printtab(40)".............
."                                        <dg>
940 printtab(160)".............
.."                                       <hd>
950 printhe$:printqq$;:inputka            <cc>
960 ifka>50then890                        <di>
970 ifka>lpthengosub1130                  <ck>
980 if(ka*p)>gthen1070                    <kc>
990 g=g-(ka*p)                            <kf>
1000 lp=lp-ka                             <am>
1010 ifk$="a"thena=a+ka                   <dh>
1020 ifk$="w"thenw=w+ka                   <id>
1030 ifk$="k"thenk=k+ka                   <gk>
1040 ifk$="r"thenr=r+ka                   <ij>
1050 ifk$="e"thene=e+ka                   <gm>
1060 goto310                              <ef>
1070 printhe$                             <hm>
1080 printtab(240):printtab(24
0)                                        <gm>
1090 printtab(40)"Du hast nich
t "                                       <bc>
1100 printtab(40)"genug Geld !
.."                                       <om>
1110 forx=1to1500:next                    <dm>
1120 goto890                              <io>
1130 printhe$                             <li>
1140 printtab(240):printtab(24
0)                                        <ki>
1150 printtab(40)"Du hast kein
en"                                       <jc>
1160 printtab(40)"Platz fuer";
ka;"!"                                    <dn>
1170 printtab(160)                        <jc>
1180 forx=0to1500:next                    <ib>
1190 goto890                              <ne>
1200 printhe$                             <po>
1210 printtab(240):printtab(24
0)                                        <oo>
1220 printtab(40)"Was willst D
u "                                       <di>
1230 printtab(40)"verkaufen ?.
."                                        <cb>
1240 print"............."                 <ap>
1250 print"(a/w/k/r/e).."                 <kg>
1260 print"............."                 <cd>
1270 print"............."                 <cn>
1280 getv$                                <ca>
1290 ifv$<>"a"andv$<>"w"andv$<
>"k"andv$<>"r"andv$<>"e"then12
80                                        <om>
1300 ifv$="a"thenp=p(1)                   <na>
1310 ifv$="w"thenp=p(2)                   <pb>
1320 ifv$="k"thenp=p(3)                   <pa>
1330 ifv$="r"thenp=p(4)                   <ac>
1340 ifv$="e"thenp=p(5)                   <aa>
1350 printhe$                             <jf>
1360 printtab(240):printtab(24
0)                                        <if>
1370 printtab(40)"Wieviel will
st"                                       <oa>
1380 printtab(40)"Du verkaufen
 ?"                                       <hf>
1390 printtab(40)".............
.."                                       <ae>
1400 printtab(160)".............
.."                                       <bd>
1410 printhe$:printqq$;:inputv
n                                         <ai>
1420 ifvn>50then1350                      <eg>
1430 ifv$="a"andvn>athen1570              <ja>
1440 ifv$="w"andvn>wthen1570              <mg>
1450 ifv$="k"andvn>kthen1570              <li>
1460 ifv$="r"andvn>rthen1570              <na>
1470 ifv$="e"andvn>ethen1570              <ma>
1480 letg=g+(p*vn)                        <da>
1490 letlp=lp+vn                          <jh>
1500 ifv$="a"thena=a-vn                   <eh>
1510 ifv$="w"thenw=w-vn                   <jd>
1520 ifv$="k"thenk=k-vn                   <hj>
1530 ifv$="r"thenr=r-vn                   <ji>
1540 ifv$="e"thene=e-vn                   <hm>
1550 ifg>=800000then5130                  <ob>
1560 goto310                              <dl>
1570 printhe$                             <hc>
1580 printtab(240):printtab(24
0)                                        <gc>
1590 printtab(40)"Du hast kein
e "                                       <po>
1600 printtab(40)"....an Bord
! "                                       <gc>
1610 printhe$                             <jk>
1620 printtab(240):printtab(24
0)                                        <ik>
1630 printtab(120);vn                     <dl>
1640 forx=0to1500:next                    <ep>
1650 goto1350                             <mk>
1660 letru=ru+1                           <of>
1670 letxg=0                              <jf>
1680 printhe$                             <oa>
1690 printtab(240):printtab(24
0)                                        <na>
1700 printtab(40)"Staedte:.."             <nh>
1710 printtab(40)"1.Swansea...
."                                        <gk>
1720 print"2.Liverpool "                  <dh>
1730 print"3.London....."                 <dg>
1740 print"4.Newcastle "                  <dh>
1750 print"5.Southampton"                 <po>
```

```
1760 print                                <hp>
1770 getse$                               <en>
1780 ifse$<"1"orse$>"5"then177
0                                         <hf>
1790 ifs$=" Swansea"andse$="1"
then1770                                  <jg>
1800 gosub5450                            <gk>
1810 ifru=5orru=10orru=15then4
870                                       <if>
1820 ifrnd(1)>.22andrnd(1)<.3t
hen 3520                                  <jp>
1830 ifrnd(1)<.15then 4250                <ko>
1840 ifrnd(1)>.55andrnd(1)<.65
then 4480                                 <ph>
1850 ifrnd(1)>.45andrnd(1)<.55
then 2530                                 <pj>
1860 ifrnd(1)>.35andrnd(1)<.45
then 3290                                 <af>
1870 ifrnd(1)>.80 and sc>0 the
n 4710:else1810                           <ng>
1880 gosub5540                            <lk>
1890 letsc=int(1.12*sc)                   <ed>
1900 ifsc>=25000then5650                  <gf>
1910 goto300                              <jj>
1920 printhe$                             <nb>
1930 printtab(240):printtab(24
0)                                        <mb>
1940 printtab(40)"Willst Du Ge
ld "                                      <lm>
1950 printtab(40)fl$"l"fo$"eih
en oder "fl$"z"fo$"u-"                    <bh>
1960 print"..............."                <ba>
1970 print"rueckzahlen ? "                <bb>
1980 print"..............."                <ce>
1990 print"..............."                <co>
2000 printtab(80)"........."              <le>
2010 getlz$                               <em>
2020 iflz$<>"l"andlz$<>"z"then
2010                                      <bj>
2030 iflz$="l"andxg>0then2160             <cl>
2040 iflz$="l"then2260                    <ah>
2050 printhe$                             <fe>
2060 printtab(240):printtab(24
0)                                        <ee>
2070 printtab(40)"Wieviel will
st "                                      <jp>
2080 printtab(40)"Du zurueckza
h-..."                                    <bb>
2090 printtab(40)"len ?.....
..."                                      <cj>
2100 printtab(160)".........
..."                                      <aa>
2110 printhe$:printqq$;:inputz
u                                         <nc>
2120 ifzu>g orzu>scthen2050               <jg>
2130 letsc=sc-zu                          <bk>
2140 letg=g-zu                            <ig>
2150 goto310                              <il>
2160 printhe$                             <mc>
2170 printtab(240):printtab(24
0)                                        <lc>
2180 printtab(40)rn$"Du kannst
 pro "rf$".."                             <jc>
2190 printrn$"Fahrt nur ein-"r
f$".."                                    <ig>
2200 printrn$"mal leihen !.."r
f$".."                                    <kj>
2210 print"..............."                <dj>
2220 print"..............."                <ed>
2230 printtab(120)                        <lg>
2240 forx=0to2000:next                    <kf>
2250 goto310                              <op>
2260 printhe$                             <cg>
2270 printtab(240):printtab(24
0)                                        <bg>
2280 printtab(40)"Wieviel will
st "                                      <hb>
2290 printtab(40)"Du leihen ?.
...."                                     <pk>
2300 printtab(40)".........
."                                        <gh>
2310 printtab(120)                        <ah>
2320 letxg=xg+1                           <gm>
2330 printhe$:printqq$;:inputl
e                                         <jb>
2340 letli=100*int(rnd(1)*50)+
1000                                      <pi>
2350 ifli<1000then2340                    <dl>
2360 ifle>lithen2400                      <ba>
2370 letsc=sc+le                          <om>
2380 letg=g+le                            <fi>
2390 goto310                              <hm>
2400 printhe$                             <ld>
2410 printtab(240)                        <go>
2420 printtab(240)                        <hi>
2430 printtab(40)"Das Limit is
t "                                       <nl>
2440 printtab(40)"£.....!....
.."                                       <ed>
2450 printhe$                             <of>
2460 printtab(240)                        <ka>
2470 printtab(240)                        <kk>
2480 printtab(121);li                     <ic>
2490 letsc=sc+li                          <gi>
2500 letg=g+li                            <ne>
2510 forx=0to1700:next                    <lk>
2520 goto310                              <po>
2530 scnclr:color4,3,4:color0,
3,4:color1,8                              <kn>
2540 printtab(240)                        <pa>
2550 printtab(129)"*** P"s2$"I
"s2$"R"s2$"A"s2$"T"s2$"E"s2$"N
 ***"                                     <gb>
2560 getkeya$                             <aj>
2570 ifke=1then2850                       <ii>
2580 ifrnd(1)>.5then3210                  <gh>
```

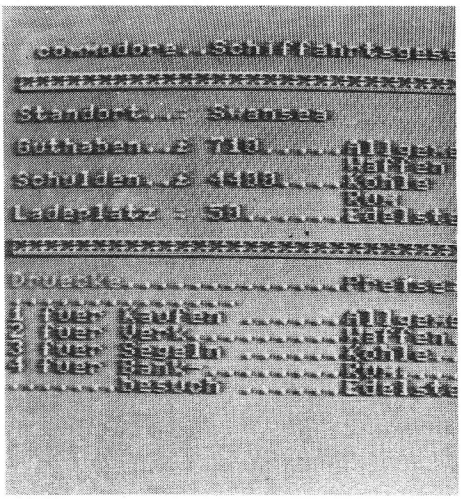
```
2590 iflp=50then2730                    <nj>
2600 scnclr                             <bk>
2610 printtab(240)                      <dh>
2620 printtab(130)"Piraten hab
en uns"                                 <gk>
2630 printtab(50)"ueberfallen
und die"                                <dm>
2640 printtab(50)"ganze Ladung
 an sich"                               <mo>
2650 printtab(50)"genommen."            <pg>
2660 getkeya$                           <gn>
2670 letlp=50                           <ld>
2680 leta=0                             <cn>
2690 letw=0                             <en>
2700 letk=0                             <el>
2710 letr=0                             <fm>
2720 lete=0                             <fj>
2730 scnclr                             <jm>
2740 letx=int(rnd(1)*10)+3              <hn>
2750 printtab(250)"Waehrend de
r Schlacht"                             <hl>
2760 printtab(50)"sind";x;"Dei
ner Mann-"                              <co>
2770 printtab(50)"schaft verwu
ndet wor-"                              <im>
2780 printtab(50)"den.Die Heil
ung durch"                              <en>
2790 printtab(50)"einen Doktor
 kostet"                                <jk>
2800 printtab(50)"Dich je Verl
etzten £75."                            <li>
2810 getkeya$                           <ad>
2820 letg=g-75*x                        <pc>
2830 ifg<0then3070                      <dp>
2840 goto1880                           <hn>
2850 scnclr                             <bf>
2860 printtab(240)                      <dc>
2870 printtab(130)"Deine Manns
chaft be-"                              <hk>
2880 printtab(50)"nutzt die Ka
none vom"                               <po>
2890 printtab(50)"schwarzen Ja
ck."                                    <np>
2900 getkeya$                           <fo>
2910 scnclr                             <fb>
2920 ifrnd(1)>.3then3000                <lh>
2930 printtab(240)                      <hi>
2940 printtab(130)"Die Kanone
ist leider"                             <ph>
2950 printtab(50)"kaputt !"             <jj>
2960 getkeya$                           <jk>
2970 scnclr                             <in>
2980 letke=0                            <kj>
2990 goto2610                           <al>
3000 printtab(240)                      <lo>
3010 printtab(90)"Die Kanone h
at das"                                 <cc>
3020 printtab(50)"Piratenschif
 f ver-"                                <ef>
3030 printtab(50)"senkt !"              <jf>
3040 printtab(50)"(Gut gemacht
,Jungs!)"                               <ah>
3050 getkeya$                           <pe>
3060 goto1880                           <fj>
3070 scnclr:color4,1:color0,1:
color1,1                                <aa>
3080 printtab(130)"Da Du nicht
 genuegend"                             <hh>
3090 printtab(50)"Geld hattest
,den Dok-"                              <hd>
3100 printtab(50)"tor zu bezah
len,hat"                                <cm>
3110 printtab(50)"Deine Mannsc
haft ge-"                               <eb>
3120 printtab(50)"meutert und
das Schiff"                             <gc>
3130 printtab(50)"verkauft,um
die Rech-"                              <cl>
3140 printtab(50)"nung bezahle
n zu koen-"                             <oi>
3150 printtab(50)"nen.........
........."                              <if>
3160 printtab(92)"Nimm's leich
t!"                                     <hn>
3170 getkeya$                           <gn>
3180 scnclr                             <ga>
3190 printtab(255)"E"s2$"N"s2$
"D"s2$"E"                               <kk>
3200 goto5260                           <oc>
3210 scnclr                             <ho>
3220 printtab(240)                      <jl>
3230 printtab(130)"Die Piraten
 haben"                                 <ck>
3240 printtab(50)"nicht angegr
iffen,"                                 <ia>
3250 printtab(50)"und du kanns
t wei-"                                 <cn>
3260 printtab(50)"fahren !"             <kl>
3270 getkeya$                           <nb>
3280 goto1880                           <dg>
3290 scnclr:color4,6,3:color0,
6,3:color1,8                            <km>
3300 printtab(240)                      <ol>
3310 printtab(211)"*** S"s2$"T
"s2$"U"s2$"R"s2$"M"s2$"***"             <op>
3320 getkeya$                           <ad>
3330 ifrnd(1)>.5orlp=50then345
0                                       <mb>
3340 scnclr                             <ab>
3350 printtab(240)                      <bo>
3360 printtab(90)"Ein Sturm is
t aufge-"                               <dl>
3370 printtab(50)"kommen,und w
ir mues-"                               <hi>
3380 printtab(50)"sen ueber di
e Haelfte"                              <pd>
```

```
3390 printtab(50)"der Ladung u
eber Bord"                    <ij>
3400 printtab(50)"werfen ! (sc
hade drum)"                   <hm>
3410 lp=lp+(a-int(a/2))+(w-int
(w/2))+(k-int(k/2))+(r-int(r/2
))+(e-int(e/2))               <pe>
3420 getkeya$                 <gi>
3430 a=int(a/2):w=int(w/2):k=i
nt(k/2):r=int(r/2):e=int(e/2) <ec>
3440 goto1880                 <nh>
3450 scnclr                   <gp>
3460 printtab(240)            <im>
3470 printtab(90)"Der Sturm ha
t sofort"                     <nn>
3480 printtab(50)"aufgehoert ,
 und du"                      <ej>
3490 printtab(50)"kannst weite
rfahren."                     <do>
3500 getkeya$                 <li>
3510 goto1880                 <bn>
3520 scnclr:color0,10,3:color4
,10,3:color1,1                <oc>
3530 printtab(240)            <nc>
3540 printtab(125)"*** L A"s2$
"N D"s2$"A"s2$"U"s2$"S"s2$"F"s
2$"L"s2$"U"s2$"G"s2$"***"     <cj>
3550 getkeya$                 <ok>
3560 ifrnd(1)>.5then3840      <eh>
3570 ifr>0then3750            <dd>
3580 tr=int(rnd(1)*3100)      <be>
3590 iftr<500then3580         <ae>
3600 scnclr                   <ag>
3610 printtab(250)"Waehrend ei
nes Land-"                    <hg>
3620 printtab(50)"ausflugs hab
en Dir"                       <gh>
3630 printtab(50)"Trickbetrueg
er £";tr                      <cj>
3640 printtab(50)"abgenommen !
"                             <eo>
3650 getkeya$                 <ep>
3660 g=g-tr                   <ol>
3670 ifg<0then3690            <jc>
3680 goto1880                 <mi>
3690 scnclr:color4,1:color0,1:
color1,2                      <ha>
3700 printtab(250)"Du bist jet
zt pleite"                    <gb>
3710 printtab(50)"und musst di
e Reise"                      <oh>
3720 printtab(50)"beenden !!!" <pi>
3730 getkeya$                 <jp>
3740 goto3160                 <pn>
3750 scnclr:printtab(250)"Waeh
rend Du einen"                <gm>
3760 printtab(50)"Landausflug
gemacht"                      <hl>
3770 printtab(50)"hast , haben
 Deine"                       <hf>
3780 printtab(50)"Matrosen den
 ganzen"                      <gd>
3790 printtab(50)"Rum ausgetru
nken !"                       <bl>
3800 getkeya$                 <of>
3810 lp=lp+r                  <ng>
3820 r=0                      <co>
3830 goto1880                 <fo>
3840 scnclr                   <ph>
3850 printtab(250)"Waehrend ei
nes Land-"                    <gh>
3860 printtab(50)"ausflugs kom
mst Du"                       <ip>
3870 printtab(50)"an einer Kne
ipe vor-"                     <jn>
3880 printtab(50)"bei und gehs
t hinein."                    <ga>
3890 getkeya$                 <ea>
3900 scnclr                   <dd>
3910 printtab(250)"Du wirst zu
 einem"                       <ge>
3920 printtab(50)"Gluecksspiel
 aufge-"                      <an>
3930 printtab(50)"fordert."   <ll>
3940 printtab(50)"Machst Du mi
t ?"                          <ad>
3950 getmm$                   <nm>
3960 ifmm$<>"j"andmm$<>"n"then
3950                          <jn>
3970 ifmm$="n"then1880        <jg>
3980 gv=int(rnd(1)*10100)     <ml>
3990 ifgv<1000then3980        <lm>
4000 gz=int(rnd(1)+1.5)       <id>
4010 scnclr                   <kb>
4020 printtab(170)"Waehle zwis
chen den"                     <hm>
4030 printtab(50)"Zahlen '1' u
nd '2' !"                     <ce>
4040 printtab(50)"Ist es die r
ichtige"                      <ao>
4050 printtab(50)"Zahl , gewin
nst Du ;"                     <in>
4060 printtab(50)"ist es die f
alsche ,"                     <hf>
4070 printtab(50)"verlierst Du
."                            <ak>
4080 getwn                    <ef>
4090 ifwn<>1andwn<>2then4080  <cl>
4100 ifwn<>gzthen4170         <lm>
4110 scnclr                   <ag>
4120 printtab(250)"Du hast ric
htig gera-"                   <ah>
4130 printtab(50)"ten und somi
t £";gv                       <mo>
4140 printtab(50)"gewonnen !" <mm>
4150 getkeya$                 <ef>
```

```
4160 g=g+gv:goto1880                    <gl>
4170 scnclr                             <ec>
4180 printtab(250)"Du hast fal
sch gera-"                              <ok>
4190 printtab(50)"ten und somi
t £";gv                                 <ak>
4200 printtab(50)"verloren !"           <be>
4210 getkeya$                           <ib>
4220 g=g-gv                             <be>
4230 ifg<0thengoto3690                  <en>
4240 goto1880                           <pk>
4250 pk=int(rnd(1)*10100)               <ni>
4260 ifpk<1000then4250                  <ma>
4270 scnclr:color4,1:color0,1:
color1,2                                <lg>
4280 printtab(240)                      <md>
4290 printtab(127)"*** DER"s2$
"SCHWARZE"s2$"JACK"s2$"***"             <ld>
4300 getkeya$                           <nl>
4310 scnclr                             <mo>
4320 printtab(250)"'Schwarzer
Jack' bie-"                             <en>
4330 printtab(50)"tet Dir eine
 Kanone"                                <lb>
4340 printtab(50)"zum Schutz v
or Pira-"                               <dp>
4350 printtab(50)"ten an."              <ma>
4360 printtab(50)"Sie kostet £
";pk;"."                                <eb>
4370 printtab(50)"Willst Du ei
ne ?"                                   <aj>
4380 getkk$                             <ii>
4390 ifkk$<>"j"andkk$<>"n"then
4380                                    <ed>
4400 ifkk$="j"andg<pkthen4450           <ho>
4410 ifkk$="n"then1880                  <em>
4420 ke=1                               <mi>
4430 g=g-pk                             <of>
4440 goto1880                           <md>
4450 printtab(90)"Du hast zuwe
nig Geld !"                             <le>
4460 getkeya$                           <hm>
4470 goto1880                           <ob>
4480 scnclr:color4,2:color0,2:
color1,1                                <ik>
4490 printtab(240):printtab(12
8)"***"s2$"SCHIFF"s2$"IN"s2$"S
ICHT"s2$"***"                           <aj>
4500 getkeya$                           <ke>
4510 scnclr                             <jh>
4520 printtab(250)"Willst Du n
aeher an"                               <ld>
4530 printtab(50)"das gesichte
te Schiff"                              <mb>
4540 printtab(50)"heranfahren
?"                                      <dm>
4550 gethf$                             <ck>
4560 ifhf$<>"j"andhf$<>"n"then
                                        <nm>
4570 ifhf$="n"then1880                  <oe>
4580 ifrnd(1)>.5then2530                <ec>
4590 vd=int(rnd(1)*10100)               <cm>
4600 ifvd<1000then4590                  <bl>
4610 scnclr                             <pm>
4620 printtab(250)"Es ist ein
englisches"                             <ch>
4630 printtab(50)"Handelsschif
f."                                     <jg>
4640 printtab(50)"Du machst mi
t dem Ka-"                              <bc>
4650 printtab(50)"pitaen ein G
eschaeft"                               <pb>
4660 printtab(50)"und verdiens
t dabei"                                <ee>
4670 printtab(50)"£";vd                 <ip>
4680 getkeya$                           <fj>
4690 g=g+vd                             <oi>
4700 goto1880                           <mi>
4710 scnclr:color4,14:color0,1
4:color1,15,3                           <gl>
4720 printtab(240):printtab(12
9)"*** EIN"s2$"GOENNER ***"             <mp>
4730 getkeya$                           <il>
4740 gr=100*int(rnd(1)*101)             <jp>
4750 ifgr<1000then4740                  <ko>
4760 ifgr>scthengr=sc                   <jn>
4770 scnclr                             <jm>
4780 printtab(250)"Ein unbekan
nter Goen-"                             <gn>
4790 printtab(50)"ner hat £";g
r;"von"                                 <dk>
4800 printtab(50)"Deinen Schul
den abbe-"                              <ca>
4810 printtab(50)"zahlt !"              <io>
4820 getkeya$                           <of>
4830 ifsc<grthen4850                    <nf>
4840 sc=sc-gr:goto1880                  <la>
4850 sc=0                               <hm>
4860 goto1880                           <gi>
4870 letmr=int(rnd(1)*5100)             <kh>
4880 ifmr<500ormr>2000then4870          <nb>
4890 letsr=int(rnd(1)*5100)             <mb>
4900 ifsr<500orsr>2000then4890          <pd>
4910 letrr=int(rnd(1)*5100)             <ne>
4920 ifrr<200orrr>2000then4910          <pl>
4930 gr=mr+sr+rr                        <nm>
4940 scnclr                             <eh>
4950 printtab(122)"Dein Schiff
 muss repariert werden !"               <fg>
4960 printtab(122)"Kosten :"            <dp>
4970 printtab(82)"Mast....£";m
r                                       <dd>
4980 printtab(82)"Segel...£";s
r                                       <fa>
4990 printtab(82)"Rumpf...£";r
r                                       <hd>
```

```
5000 printtab(49)"--------"          <bj>
5010 printtab(82)"Gesamt..£";g
r                                    <if>
5020 getkeya$                        <ko>
5030 g=g-gr                          <dn>
5040 ifg<0then5060                   <ok>
5050 goto1880                        <ch>
5060 scnclr:color4,1:color0,1:
color1,2                             <mp>
5070 printtab(250)"Da Du zuwen
ig Geld"                             <ae>
5080 printtab(50)"hast , die R
echnung"                             <ab>
5090 printtab(50)"zu bezahlen
, musst"                             <md>
5100 printtab(50)"Du die Reise
 been-"                              <dh>
5110 printtab(50)"den !!!"           <fb>
5120 goto3160                        <gh>
5130 scnclr:color0,8:color4,8:
color1,3,3                           <ie>
5140 printtab(125)"$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$"                  <hn>
5150 printtab(5)"$.............
.............$"                      <dc>
5160 printtab(5)"$....H"s2$"E"
s2$"R"s2$"Z"s2$"L"s2$"I"s2$"C"
s2$"H"s2$"E N....$"                  <cg>
5170 printtab(5)"$.............
.............$"                      <eg>
5180 printtab(5)"$..G"s2$"L"s2
$"U"s2$"E"s2$"C"s2$"K"s2$"W";        <aj>
5190 printstab(5)"$"s2$"U"s2$"N"s2$"S"s2
$"C"s2$"H"s2$s2$"$"                  <ep>
5200 printtab(5)"$.............
.............$"                      <ge>
5210 printtab(5)"$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$"                   <ga>
5220 printtab(204)"Du hast ueb
er £ 800000 verdient"                <fi>
5230 printtab(44)"und kannst D
ich jetzt zur Ruhe"                  <ch>
5240 printtab(44)"setzen !!!"        <nj>
5250 printtab(120)"Genauer Spi
elstand : £";g                       <oi>
5260 getkeya$                        <jp>
5270 scnclr                          <jc>
5280 printtab(240):printtab(13
1)"Noch ein Spiel ?"                 <oh>
5290 getns$                          <ce>
5300 ifns$<>"j"andns$<>"n"then
5290                                 <ol>
5310 ifns$="j"thenrun                <jd>
5320 printtab(250)"T"s2$"S"s2$
"C"s2$"H"s2$"U"s2$"E"s2$"S"s2$
"S..!"                               <ig>
5330 end                             <gg>
5340 letp(1)=int(rnd(1)*16)          <fc>
5350 ifp(1)<1then5340                <ko>
5360 letp(2)=10*int(rnd(1)*19)       <hh>
5370 ifp(2)<50then5360               <pj>
5380 letp(3)=10*int(rnd(1)*100
)                                    <le>
5390 ifp(3)<300then5380              <dp>
5400 letp(4)=100*int(rnd(1)*31
)                                    <mm>
5410 ifp(4)<500then5400              <ep>
5420 letp(5)=100*int(rnd(1)*10
1)                                   <ap>
5430 ifp(5)<1000then5420             <jc>
5440 return                          <od>
5450 ifse$="1"then lets$=" Swa
nsea"                                <pl>
5460 ifse$="2"then lets$=" Liv
erpool"                              <la>
5470 ifse$="3"then lets$=" Lon
don"                                 <mj>
5480 ifse$="4"then lets$=" New
castle"                              <la>
5490 ifse$="5"then lets$=" Sou
thampton"                            <hh>
5500 return                          <bp>
5510 print"............."            <pm>
5520 print"............."            <ag>
5530 goto670                         <nf>
5540 scnclr:color4,15,3:color0
,15,3:color1,8                       <np>
5550 print" ";ne$;"..Schiffahr
tsgesellschaft"                      <jh>
5560 printtab(40)rn$"*********
*****************************
*"rf$                                <ok>
5570 printtab(40)"Standort..:"       <gj>
5580 printtab(40)"Guthaben..£.
.......Allgemeines"                  <li>
5590 printtab(20)"Waffen"
     "                               <ep>
5600 print"Schulden..£.......
.Kohle"                              <pb>
5610 printtab(20)"Rum"               <ho>
5620 print"Ladeplatz :.......
.Edelsteine"                         <eb>
5630 printtab(40)rn$"*********
*****************************
*"rf$                                <da>
5640 return                          <km>
5650 scnclr:color4,1:color0,1:
color1,2                             <ca>
5660 printtab(250)"Weil Du ueb
er £ 25000"                          <ac>
5670 printtab(50)"Schulden has
t , pfaen-                           <pb>
5680 printtab(50)"den Deine Gl
aeubiger"                            <gh>
5690 printtab(50)"das Schiff ,
 und deine"                          <ob>
```

```
5700 printtab(50)"Reise ist zu
ende !!!"                      <ho>
5710 goto3160                  <lh>
5720 fori=1tozz:printxx$;:next
:return                        <nj>
5730 rem nachspann ==========  <ea>
5740 rem farbcodes/steuercodes <ap>
5750 c4$=chr$(017):rn$=chr$(01
8)                             <ok>
5760 he$=chr$(019):fl$=chr$(13
0)                             <pj>
5770 fo$=chr$(132):rf$=chr$(14
6)                             <be>
5780 c1$=chr$(157)             <kj>
5790 rem zeichensatz/graphik   <ma>
5800 s2$=chr$(160):zn$=chr$(18
4)                             <df>
5810 return                    <fg>
5820 rem ====================  <gc>
5830 rem 60671 bytes memory    <mp>
5840 rem 12881 bytes program   <bi>
5850 rem 00329 bytes variables <jn>
5860 rem 00062 bytes arrays    <nk>
5870 rem 00421 bytes strings   <dl>
5880 rem 46978 bytes fre(0)    <je>
5890 rem ====================  <kj>
5900 rem ersetzen sie bitte    <hl>
5910 rem in den´print-anwei-   <if>
5920 rem sungen die punkte     <dp>
5930 rem durch blanks.         <cp>
5940 rem ====================  <nl>
```

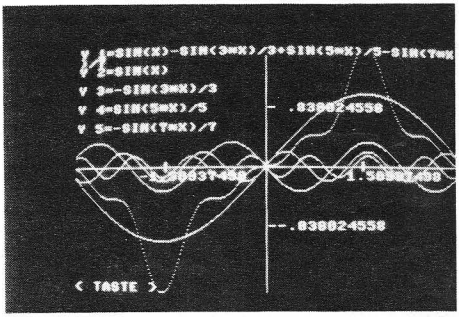


# FUNKTIONS-PLOTTER

Kurven grafisch darstellen, Kurven plotten. Bis zu 9 Kurven können auf dem Bildschirm dargestellt, und, falls ein solcher vorhanden, auch mit Hilfe des Commodore-Plotters C-1520 zu Papier gebracht werden.

Nach dem Einladen starten Sie das Programm mit RUN. Nun erscheint das Menü. Es werden Ihnen mehrere Vorschläge für den Programmablauf gemacht. Normalerweise geben Sie nun eine 1, um eine oder mehrere Funktionen einzugeben. Jetzt werden Sie nach der Anzahl der Funktionen gefragt. Geben Sie z.B. eine 3 ein.

An dieser Stelle muß darauf hingewiesen werden, daß nach der ersten Funktion der maximale Y-Wert berechnet wird. Falls in einer der folgenden Funktionen (2 bis 9) ein größerer Y-Wert berechnet wird, erscheint dieser nicht mehr auf dem Bildschirm.




Geben Sie also z.B. "sin(x)" ein und drücken RETURN. Nun wird die Funktion Y2 abgefragt. Geben Sie z.B. "cos(x)" und für Y3 = "sin)x·x)" ein. Danach wird der maximale X-Wert abgefragt. Geben Sie 3.14 ein, und ihr Computer wird alle Funktionen im Bereich zwischen X= -3.14 und X= +3.14 berechnen und auf dem Bildschirm darstellen.

Nach dem Zeichnen der einzelnen Funktionen betätigen Sie eine Taste und gelangen so in das Menü zurück. Sie können sich jetzt entscheiden, ob Sie neue Funktionen eingeben möchten oder nur den Bereich der bereits eingegebenen verändern wollen. Außerdem können Sie dargestellte Funktionen abspeichern oder neue Funktionen einladen, z.B. FOURIERENTWICKLUNG. Weiterhin ist es möglich, die dargestellten Funktionen auf den Commodore-Plotter C-1520 auszuplotten (Nur bei FUNKTIONSPLOTTER 2).

**Zur Darstellung der Funktionen:**

Bei der Darstellung auf dem Bildschirm setzen sich die einzelnen Funktionen aus 320 einzeln berechneten Punkten zusammen. Bei der Darstellung auf dem C-1520 besteht jede einzelne Funktion aus 480 Punkten.

```
10 rem funktionsplotter==plus4 <hg>
20 rem (p) 7/87 commodore welt <pk>
30 rem ======================== <ci>
40 rem (c) 3/87 p. rudzynski  <gn>
50 rem                        <mb>
60 rem                        <ml>
70 rem basic v3.5             <cd>
80 rem plus4 (c16/116 + 64kb) <dl>
90 rem ======================== <ge>
100 gosub 2630                <lm>
110 :                         <ki>
120 rem ********************** <kd>
130 rem *** menueerstellung ** <gc>
140 rem ********************** <lh>
150 :                         <na>
160 dim b$(9)                 <bg>
170 color 0,1,1:color 1,2,5:co
lor 4,1,1                     <jl>
180 scnclr:graphic 0:print chr
$(14)                         <de>
190 char 0,9,1,"FUNKTIONSPLOTT
ER"                           <ol>
200 char 0,0,4,"Es koennen meh
rere Funktionen gleich=......z
eitig dargestellt werden !"   <jn>
210 char 0,6,08,"Funktionen ei
ngeben =1"                    <he>
220 char 0,6,10,"Neuer Bereich
.......=2"                    <gn>
230 char 0,6,12,"Abspeichern..
.......=3"                    <aj>
240 char 0,6,14,"Einladen.....
.......=4"                    <lm>
250 char 0,6,16,"Ausdrucken (C
-1520) =5"                    <fp>
260 char 0,6,18,"Ende.........
.......=6"                    <gl>
270 char 0,2,22,"Erstellt von
P.Rudzynski am 1.3.1987"      <nh>
280 char 0,10,24,"Copyright (c
)"                            <if>
290 getkey a$:a=val(a$):if a<1
 or a>6 then goto 290         <fh>
300 on a goto 370,650,1460,161
0,1760,2430                   <md>
310 :                         <hb>
320 rem ********************** <gm>
330 rem *** funktionseingabe * <co>
340 rem ********************** <ia>
350 :                         <jj>
360 :                         <kd>
370 scnclr                    <fl>
380 input"Anzahl der Funktione
n =";a$                       <ln>
390 b=val(a$):if b<1 or b>9 th
en goto 370                   <ih>
400 for c=1 to b              <bo>
410 scnclr:c$="Funktion Y"+str
$(c)+"="                      <ao>
420 printc$:printhe$qr$;:input
b$(c)                         <oe>
430 next                      <db>
440 scnclr:printchr$(144);    <ga>
450 e$=""                     <cc>
460 for c=1 to b              <fk>
470 f=20*c+840                <gg>
480 print f;"y";"=";b$(c),"   <nh>
490 printf+12;"b$(";c;")=";chr
$(34);b$(c);chr$(34),":return" <hk>
500 e$=e$+chr$(13)+chr$(13)   <mf>
510 next                      <ib>
520 print" 580 b=";b          <ia>
530 print:print"goto 580"     <jk>
540 key 1,chr$(19)+chr$(13)+ch
r$(13)+chr$(13)+e$            <hl>
550 poke2035,0:sys56364       <mf>
560 color 1,1,1               <ae>
570 end                       <lm>
580 b= 2                      <gm>
590 gosub 2630                <ki>
600 :                         <je>
610 rem ********************** <ip>
620 rem * berechnen von y-max* <dk>
630 rem ********************** <kd>
640 :                         <lm>
650 color 1,2,5               <gd>
660 scnclr                    <ho>
670 print"Geben Sie jetzt den
maximalen zu berech=.........
nenden X-Wert ein."           <kn>
680 print:print:input"Maximale
r X-Wert =";a$                <nn>
690 d=val(a$):if d<=0 then got
o 660                         <ip>
700 gosub 2470                <bi>
710 g=320                     <fg>
720 gosub 730:goto 1100       <ho>
730 dx=2*d/g                  <ea>
740 yy=0                      <hk>
750 for x=-d to d step dx     <jh>
760 gosub 860                 <cf>
770 if abs(y)>yy then yy=y    <lh>
780 next                      <jb>
790 return                    <kh>
800 :                         <fn>
810 :                         <gh>
820 rem ********************** <gc>
830 rem ablegen der funktionen <jk>
840 rem ********************** <hg>
850 :                         <ip>
860 :                         <jj>
870 trap 2600                 <ai>
880 :                         <kn>
890 trap 2600                 <bm>
900 :                         <mb>
910 trap 2600                 <da>
```

```
920 :                                  <nf>
930 trap 2600                          <ee>
940 :                                  <oj>
950 trap 2600                          <fi>
960 :                                  <pn>
970 trap 2600                          <gm>
980 :                                  <bb>
990 trap 2600                          <ia>
1000 :                                 <cf>
1010 trap 2600                         <je>
1020 :                                 <dj>
1030 trap 2600                         <kj>
1040 :                                 <eo>
1050 :                                 <fi>
1060 rem ********************          <cj>
1070 rem * koordinatensystem *         <di>
1080 rem ********************          <dn>
1090 :                                 <ia>
1100 scnclr:graphic 1,1                <no>
1110 char,20,6,"-"                     <mh>
1120 char,20,18,"-"                    <ae>
1130 char,9,12,"Y"                     <il>
1140 char,29,12,"Y"                    <mh>
1150 draw,0,100 to 319,100             <ho>
1160 draw,159,0 to 159,200             <jj>
1170 char,21,6,str$(yy/2)              <dj>
1180 char,21,18,str$(-yy/2)            <cb>
1190 char, 7,13,str$(-x/2)             <ka>
1200 char,27,13,str$(x/2)              <db>
1210 :                                 <pi>
1220 :                                 <ac>
1230 rem ********************          <nd>
1240 rem * zeichnen der kurven        <jp>
1250 rem ********************          <oh>
1260 :                                 <ck>
1270 for c=1 to b                      <ih>
1280 for x=-d to d step dx             <km>
1290 on c gosub 860,880,900,92
0,940,960,980,1000,1020                <kd>
1300 i=100-100*y/yy                    <ef>
1310 draw,319*x/2/d+159,i              <gd>
1320 next                              <kp>
1330 c$="y"+str$(c)+"="+b$(c)          <lo>
1340 char ,0,c*2-2,c$                  <dk>
1350 graphic 1,0                       <lg>
1360 next                              <nh>
1370 char ,0,24,"< taste >"            <ji>
1380 getkey a$                         <gi>
1390 goto 170                          <je>
1400 :                                 <lh>
1410 :                                 <mb>
1420 rem ********************          <jc>
1430 rem * graphik speichern *         <cf>
1440 rem ********************          <kg>
1450 :                                 <oj>
1460 gosub 2520                        <ap>
1470 if k=1 then open 1,1,1,k$
:goto 1490                             <nj>
1480 open 1,8,2,k$+",s,w"              <ok>
1490 print#1,b                         <ao>
1500 for c=1 to b                      <go>
1510 print#1,b$(c)                     <nk>
1520 next                              <hh>
1530 close 1                           <na>
1540 goto 170                          <cl>
1550 :                                 <eo>
1560 :                                 <fi>
1570 rem ********************          <cj>
1580 rem * grafik laden       *        <cn>
1590 rem ********************          <dn>
1600 :                                 <ia>
1610 gosub 2520                        <kg>
1620 if k=1 then open 1,1,0,k$
:goto 1640                             <gm>
1630 open 1,8,2,k$+",s,r"              <hm>
1640 input#1,b                         <jb>
1650 for c=1 to b                      <af>
1660 input#1,b$(c)                     <fn>
1670 next                              <ao>
1680 close 1                           <gh>
1690 goto 440                          <mb>
1700 :                                 <oe>
1710 :                                 <oo>
1720 rem ********************          <lp>
1730 rem ausplotten auf c-1520         <hk>
1740 rem ********************          <nd>
1750 :                                 <bg>
1760 g=480                             <hl>
1770 open3,6,3:print#3,1:close
 3                                     <pc>
1780 open 1,6,1:open 2,6,2             <fe>
1790 open 3,6                          <di>
1800 for c=1 to b                      <jm>
1810 print#2,c                         <fc>
1820 c$="y"+str$(c)+"="+b$(c)          <kk>
1830 print#3,c$                        <il>
1840 next                              <lj>
1850 print#2,0                         <gh>
1860 pa=0:pb=192:pp=48                 <na>
1870 for c=0 to 1                      <na>
1880 for i=pa to pb step 2*pp          <mj>
1890 print#1,"d",i,-400                <mg>
1900 print#1,"m",i+pp,-400             <cd>
1910 print#1,"d",i+pp,0                <bf>
1920 if i=pb then goto 1940            <jn>
1930 print#1,"m",i+2*pp,0              <ba>
1940 next                              <bn>
1950 pa=239:pb=431                     <jm>
1960 if c=1 then goto 1980             <gc>
1970 print#1,"m",239,0                 <gf>
1980 next                              <ef>
1990 pa=0:pb=100:pp=50                 <ea>
2000 for c=0 to 1                      <fc>
2010 for i=pa to pb step 2*pp          <el>
2020 print#1,"d",0,-i                  <oe>
2030 print#1,"m",0,-(i+pp)             <jc>
```

```
2040 print#1,"d",479,-(i+pp)    <ah>
2050 print#1,"m",479,-(i+2*pp)  <pj>
2060 if i<pb then goto 2080     <ch>
2070 print#1,"d",0,-(i+2*pp)    <ja>
2080 next                       <kk>
2090 pa=201:pb=301              <bk>
2100 if c=1 then goto 2120      <oc>
2110 print#1,"m",479,-201       <ag>
2120 next                       <nc>
2130 print#1,"m",120,-230       <al>
2140 print#3,-2*d/5             <ak>
2150 print#1,"m",313,20         <ef>
2160 print#3,2*d/5              <hd>
2170 x=2*d/5:gosub 2470         <ce>
2180 gosub 730                  <ld>
2190 scnclr                     <ho>
2200 print#1,"m",250,142        <km>
2210 print#3,yy/2               <dc>
2220 print#1,"m",250,-179       <hf>
2230 print#3,-yy/2              <pb>
2240 for c=1 to b               <ff>
2250 print#2,c                  <al>
2260 for x=-d to d-dx step dx   <mk>
2270 on c gosub 860,880,900,92
0,940,960,980,1000,1020         <hk>
2280 print#1,"m";479*x/2/d+240
,200*y/yy+126                   <mo>
2290 print#1,"d";479*x/2/d+240
,200*y/yy+128                   <nb>
2300 next                       <ig>
2310 print#1,"m",0,148          <lk>
2320 next                       <jl>
2330 print#1,"m",0,-200         <go>
2340 close 3                    <aa>
2350 close 2:close 1            <be>
2360 goto 170                   <gc>
2370 :                          <if>
2380 :                          <ip>
2390 rem ********************  <ga>
2400 rem *** und schluss  **** <ck>
2410 rem ********************  <he>
2420 :                          <lh>
2430 scnclr                     <gp>
2440 print:print:print:print:p
rint:print"...............End
e"                              <aj>
2450 end                        <bl>
2460 :                          <np>
2470 scnclr:graphic 0           <np>
2480 a$=chr$(130)+"......Funkt
ion wird berechnet"             <dc>
2490 char 0,2,10,a$             <ep>
2500 return                     <fl>
2510 :                          <bb>
2520 scnclr:graphic 0           <bb>
2530 print:print"Kassette (1)
oder Diskette (2)  ?"           <mk>
2540 getkey a$:k=val(a$):if k<
1 or k>2 then goto 2520         <hb>
2550 print:print                <gl>
2560 if k=2 then directory      <fj>
2570 print:print:input"Name de
r Graphik";k$                   <cp>
2580 return                     <km>
2590 :                          <gc>
2600 resume next                <ik>
2610 rem nachspann ==========   <ae>
2620 rem farbcodes/steuercodes <nd>
2630 rn$=chr$(018):he$=chr$(01
9)                              <mg>
2640 c3$=chr$(029)              <fj>
2650 rem *** zeichenfolgen *** <la>
2660 for q=1 to 16              <co>
2670 qr$=qr$+c3$                <pm>
2680 next q                     <ff>
2690 return                     <bk>
2700 rem ====================== <cg>
2710 rem 60671 bytes memory     <jd>
2720 rem 12288 bytes grafik     <hj>
2730 rem 05267 bytes program    <og>
2740 rem 00098 bytes variables  <go>
2750 rem 00040 bytes arrays     <ke>
2760 rem 00541 bytes strings    <am>
2770 rem 42437 bytes fre(0)     <fe>
2780 rem ====================== <hg>
2790 rem bitte blanks statt     <ec>
2800 rem punkte in folgenden    <ij>
2810 rem zeilen eingeben:       <nb>
2820 rem 200-260,670,2440,2480 <ko>
2825 rem alle zeilen eintippen <  >
2830 rem ====================== <kj>
```

---

# GUT SORTIERT IST HALB GEFUNDEN

**Daten in den Computer einzutippen, sie wieder auf dem Bildschirm oder Drucker auszugeben, ist für die meisten ein Kinderspiel. Schwieriger wird's allerdings, wenn's darum geht, gezielte Ordnung in eine Datei zu bringen. COMMODORE WELT stellt Ihnen die gebräuchlichsten Sortierroutinen vor.**

Irgendein Schlaumeier hat einmal das geflügelte Wort von sich gegeben: „Die hälfte seines Lebens sucht der Mensch vergebens...". So banal der Spruch auch klingt, ein Körnchen Wahrheit steckt doch drin.

Jede Kartei, jedes Verzeichnis richtet sich nach gewissen Suchkriterien aus. Der gewünschte Name, die gesuchte Adresse soll ja ohne großen Zeitverlust möglichst nervenschonend gefunden werden. Bei der Dateiablage im Computer ist der Idealfall, die geforderten Daten auf Knopfdruck abrufen zu können.

Es gibt verschiedene Möglichkeiten, zum Ziel zu kommen. Da wir ziemlich sicher sind, daß der größte Teil unserer Leser in BASIC programmiert, wollen wir Ihnen hier ein paar Sortierunterprogramme vorstellen. Die Bezeichnung „Unterprogramme" sagt es eigentlich schon: Sie sind dazu gedacht, irgendwo in einer Ihrer eigenen Software-Entwicklungen eingebaut zu werden.

## SORTIEREN FÜR LEICHTERES SUCHEN

Suchen und Sortieren arbeiten Hand in Hand. Es ist für den Computer ungleich einfacher, aus einer bereits sortierten Datenmenge das Richtige herauszufinden, wobei es keine Rolle spielt, ob es sich um numerische Daten (Zahlen) oder Texte (Strings) handelt. Bei einer vorsortierten Datei können zwei Suchmethoden programmiert werden:

a) Durchsuchen sämtlicher Daten (Wörter, Nummern, Texte),
b) „Halbierungs"-Methode, auch binärer Suchmodus genannt.

Keiner großen Erklärung bedarf der erste Punkt. Wenn Sie den Namen Meier suchen, so durchforstet Ihr Computer alle zur Verfügung stehenden Daten, von Anfang bis Ende, bis er ihn gefunden hat. Ihr Pech, wenn die Datenmenge recht groß ist, dann dauert es eben seine Zeit.

Ein ganzes Stück schneller geht es mit der „Halbierungsmethode" vonstatten. Hier ist aber Voraussetzung, daß die Daten bereits sortiert sind. Sucht der Computer nach einem bestimmten Wort oder einer Zahl, so müssen Sie ihm ein „Erkennungsmerkmal" angeben, das der Begriff besitzt. Am einfachsten sind für die meisten Programmierer die ASCII-Werte der Zeichen auf der Commodore-Tastatur.

Der Buchstabe "A" beispielsweise hat den ASC-Code 65, „B" liegt um eins höher, also bei 66. Am besten prüfen Sie's nach mit PRINT ASC("A"), oder welches Zeichen Sie auch immer vorziehen. Zurück zu unserem Meier: Bei einem sortierten Datenfeld stellt der Computer fest, daß dieser Name mit dem von Ihnen vorgegebenen Merkmal (ASC-Wert) erst in dem Teil des Datenfeldes beginnen kann, dessen ASC-Werte gleich oder größer 77 sind (das ist nämlich der ASC-Wert von „M"). Das Gesamtfeld wird also halbiert, das verbleibende ebenso. So geht es weiter, bis nur noch zwei zu durchsuchende Elemente übrig sind, wovon dann eines todsicher das gewünschte ist. So weit scheint der Sinn des Sortierens einleuchtend zu sein. Nur: wie stellt man's an?

Es gibt viele Arten von Sortiermethoden, wovon einige allerdings recht unterschiedliche Sortierzeiten für große Datenmengen benötigen: Einfüge-, Tausch-, Einzelbyte- und Selektierungsmethode. Diese Klassiker wurden bereits zu Anfang der BASIC-Programmierung entwickelt, haben aber bis heute von ihrer Effizienz nichts verloren.

## DIE INSERT-METHODE: AM HÄUFIGSTEN ANGEWANDT

Bei der Insert-Methode macht's der Computer ganz unkompliziert: Er beginnt bei der ersten Angabe und vergleicht sie mit der nächsten. Sie ist kleiner als die erste, so wird sie vor dieser eingefügt (daher der Name). Das dritte Element wird nun mit dem zweiten verglichen und ebenso behandelt, bis alle Datensätze und -felder „gecheckt" sind.

Die Insert-Methode ist die wohl simpelste, aber nicht unbedingt langsamste Sortierroutine.

## UNMITTELBARE AUSWAHL: SELECTION

Ein weiterer Sortieralgorithmus, der neben dem Einfügeprinzip zusätzlich das des Tauschens verfolgt, ist die Auswahl-Methode. Auch hier werden zwei For-Next-Schleifen verwendet. Die erste beginnt beim höchsten Element des Datenfeldes und arbeitet sich vor bis zum nächsten. Die zweite Schleife stellt die einzelnen Datenfelder, von vorne beginnend, dagegen.

Der Grundgedanke dabei ist, daß das momentan größte Element an die letzte Stelle des gesamten Arrays gebracht wird. Die Größe des Datensatzes wird jetzt um Eins reduziert, dann erneut nach dem größten Teileelement gesucht. Das wird wiederum hinten angefügt, ohne das vorher an die letzte Stelle gebrachte Element zu beeinflussen, da dieses ja von der Gesamtmenge abgetrennt wurde. Ist nach den verschiedenen Durchläufen nur noch ein Teilelement übriggeblieben, so ist dieses – logischerweise – das erste.

## VORSORTIEREN BRINGT'S NICHT IMMER

Allerdings sind mit der Selektier-Methode die besten zeitlichen Ereignisse nur dann zu erzielen, wenn keine noch so grobe Vorsortierung vorliegt. Dann kann

es nämlich sein, daß sich die höchsten Elemente bereits ziemlich am Ende des Arrays befinden, die Selektier-Routine aber unverdrossen von vorne zu suchen beginnt. Bei einem zufällig entstandenen Datensatz werden die besten Ergebnisse erzielt.

## HAT NICHTS MIT KAUGUMMI ZU TUN: BUBBLESORT

Bubble Sort ist wohl eine der bekanntesten und am häufigsten benutzten Sortierroutinen. Sie arbeitet mit dem Prinzip des Tauschens. Das erste Element des gesamten Arrays wird mit dem folgenden verglichen. Ist es größer, so wird ausgetauscht. Auf diese Weise gelangt das größte Element des Datensatzes automatisch ans Ende, was ja auch der Sinn der Übung war.

„Bubble" bedeutet soviel wie „Luftblase" und beschreibt leicht humorig die Art, sich – bildlich gesprochen – das jeweils größte Datenelement durch die ganze Sortierroutine hindurch bis ans Ende des Arrays bewegt. Der Unterschied zu den bereits erwähnten Einfügemethoden besteht darin, daß Bubble Sort beim größten Datenelement zu sortieren beginnt, nicht beim kleinsten. Der Zeitaufwand dürfte in etwa derselbe sein.

## DIE ROUTINE NACH SHELL

Der nächste Sortier-Algorithmus, nach dessen Erfinder benannt, hat seit nahezu dreißig Jahren unverändert seine Gültigkeit. Obwohl Shell ebenfalls das Prinzip des Einfügens verfolgt, ist der Aufbau der Routine doch ein bißchen komplizierter. Es werden nämlich nicht nur unmittelbar aneinander grenzende Daten miteinander verglichen, sondern vor allem entfernt liegende, (etwa der erste mit dem sechsten). Dieser Modus ist im Programm vorzugeben.

So wird eine Art Vorsortierung erreicht. Die entsprechenden Daten werden auf neue Variablen übertragen und ebenfalls wieder sortiert, diesmal aber mit reduzierten Abständen, beispielsweise das erste mit dem vierten. Unser Sortiersieb wird auf diese Weise immer enger, so daß zum Schluß nur noch der Einser-Schritt übrig bleibt, was gleichbedeutend mit einer exakten Sortierung ist.

## HILFSROUTINE: EINFÜGEN

Genau genommen besteht dieser Sortiermodus aus zwei Sortierroutinen, denn die Grobsortierung übernimmt das bereits bekannte Insert. Da jetzt dem Hauptprogramm bereits vorbearbeitete Daten vorgelegt werden können, geht die weitere Abarbeitung sehr rasch. Shell Sort bewährt sich ideal bei recht großen Datensätzen (Adressen, Text) mit vielen Einzelelementen.

## TREFFENDER NAME: QUICK SORT

Der nächste Sortier-Oldie, den wir uns ansehen, heißt Quick Sort. Er wurde etwa zur selben Zeit wie Shell Sort entwickelt; sein Erfinder heißt Roare, ebenfalls ein Amerikaner. Quick Sort, die wohl gebräuchlichste Sortierroutine, vermutlich weil sie bei sehr großen Datenmengen einwandfrei die schnellste ist.

Bei kleineren Feldern haben Insert- und Selektiermethode die Nase vorn, einfach deswegen, weil sie viel simpler aufgebaut sind und der Computer nicht so viele Programmschritte abarbeiten muß. Bei großen Arrays dagegen macht so ein einfacher Algorithmus stur seine Programmschritte, auch wenn die Sortierung so gut wie abgeschlossen ist. Eine „intelligente" Routine wie Quick-Sort erkennt das aber beizeiten und ist dadurch, obwohl softwaretechnisch weit komplizierter angelegt, viel schneller fertig.

Aus dem gesamten Array wird irgendein Element herausgezogen und abgesondert, weil es ab sofort als Vergleichsmuster für alle weiteren Datenelemente fungiert. Nach dem Motto: „Kleiner als" oder „größer als" werden jetzt die übrigen Daten links oder rechts von unserem Muster plaziert. Die erste Vorsortierung ist damit erreicht.

Nun wird jedes so gewonnene Array (das mit den kleineren und das mit den größeren Werten) genauso behandelt wie das gesamte Datenfeld vorher: wieder ein Vergleichsmuster herausgezogen und wieder ein „Kleiner" und „Größer"-Datensatz geschaffen. Auf diese Weise wird die Einteilung immer ‚feiner', bis das fertig sortierte Array vor uns steht. Ideal für diesen Sortieralgorithmus erweist sich natürlich die Auswahl des Vergleichselements aus der Mitte der Teildatensätze.

## AUF ANWENDUNG UND MENGE KOMMT ES AN

Welchen der vorgestellten Algorithmen Sie künftig für Ihre selbstfabrizierten Dateiverwaltungen verwenden möchten, überlassen wir Ihnen und dem dafür vorgesehenen Anwendungszweck. Beachten Sie bitte dabei immer die Menge der eingegebenen und zu sortierenden Daten sowie den Grad der bereits bei der Eingabe vorliegenden Grobsortierung.

Quick Sort beispielsweise ist bei zufällig zusammengewürfelten Werten am effektivsten. Bei einer nahezu perfekt alphabetisch geordneten Datei kann es aber ohne weiteres sein, daß Bubble- oder Insert Sort um einiges schneller sind. Wenn Sie das beachten, dann sind wir sicher, daß Sie das Optimale aus diesen Sortierroutinen herausholen können.

Alle hier besprochenen Programme sind als Listing abgedruckt, als Daten haben wir zwanzig Begriffe aus dem Umfeld des Computers gewählt. Den Sortierablauf jeder Routine können Sie am Bildschirm in groben Zügen verfolgen, das fertig sortierte Array wird als „Erfolgsmeldung" rechts am Bildschirm ausgegeben. Wollen Sie die eine oder andere Routine in eigenen Programmen verwenden, so lassen Sie bitte die Zeilen weg, die Sie nicht benötigen (Bildschirmausgabe, DATA-Werte und dergleichen). Am besten bauen Sie so ein Sortprogramm als Unterroutine ein, die mit GOSUB angesprungen werden kann.

# DAS SONDERANGEBOT: PRIVATE KLEINANZEIGEN KOSTENLOS!

Das bietet Ihnen COMMODORE-WELT: KLEINANZEIGEN SIND KOSTENLOSE FÜR PRIVATANBIETER! Suchen Sie etwas, haben Sie etwas zu verkaufen, zu tauschen, wollen Sie einen Club gründen? Coupon ausfüllen, auf Postkarte kleben oder in Briefumschlag stecken und abschicken. So einfach geht das. Wollen Sie das Heft nicht zerschneiden, können Sie den Coupon auch fotokopieren. Oder einfach den Anzeigentext uns so schicken, auf Postkarte oder im Brief. Aber bitte mit Druckbuchstaben oder in Schreibmaschinenschrift!

Und: Einschließlich Ihrer Adresse und/oder Telefonnummer sollten acht Zeilen à 28 Anschläge nicht überschritten werden.

ACHTUNG: WICHTIGER HINWEIS!

Wir veröffentlichen nur Kleinanzeigen privater Inserenten, keine gewerblichen Anzeigen. Die kosten pro Millimeter DM 5.00 plus Mehrwertsteuer!

Wir versenden für Privat-Inserenten keine Beleg-Exemplare!

Chiffre-Anzeigen sind nicht gestattet! Wir behalten uns vor, Anzeigen, die gegen rechtliche, sittliche oder sonstige Gebote verstoßen, abzulehnen!

Anzeigenabdruck in der Reihenfolge ihres Eingangs, kein Rechtsanspruch auf den Abdruck in der nächsten Ausgabe!

Die Insertion ist nicht vom Kauf des Heftes abhängig! Wir behalten uns vor, Anzeigen, die nicht zum Themenkreis des Heftes – Computer – gehören, nicht abzudrucken oder sie nur insoweit zu berücksichtigen, wie es der Umfang des kostenlosen Anzeigenteils zuläßt.

# PROGRAMMSERVICE
## Achtung! Preis-Senkung!!!

Hiermit bestelle ich in Kenntnis Ihrer Verkaufsbedingungen die Listings dieses Heftes auf

☐ Kassetten zu 40,- ☐ Disketten zu 40,- (16er)

Name _____

Vorname _____

Straße/Hausnr. _____

PLZ/Ort _____

Ich bezahle:

☐ per beiliegendem Verrechnungsscheck / Bargeld

☐ bargeldlos per Bankeinzug von meinem Konto (nur möglich in der Bundesrepublik!)

bei (Bank) und Ort _____

Kontonummer _____

Bankleitzahl _____

(steht auf jedem Kontoauszug)

Verkaufsbedingungen: Lieferung nur gegen Vorkasse oder Bankabbuchung. Keine Nachnahme. Umtausch bei Nichtfunktionieren.

Unterschrift _____ VI

Bitte ausschneiden und einsenden an

COMMODORE WELT
KASSETTENSERVICE VI
POSTFACH 1161
D-8044 UNTERSCHLEISSHEIM

Da es sich um reine BASIC-Routinen ohne PEEKS und POKES handelt, sind sie für jeden Commodore-Homecomputer geeignet, vom VC20 und C16 bis hin zum 128PC. *B.U.* ■

## Insert-Sort – 1 –

```
10 rem sortierroutine                    <ab>
20 rem insert sort                       <pk>
30 fori=1to26:cr$=cr$+chr$(29)
:next                                    <dn>
40 ci$=left$(cr$,12):rn$=chr$(
18)                                      <jn>
50 m=20:dims$(m)                         <eo>
60 scnclr:printrn$"unsortiert:
"                                        <ak>
70 fori=1tom:reads$(i):prints$
(i):next:gosub 200                       <mi>
80 fori=2tom                             <kl>
90 ifs$(i)>=s$(i-1)then160:rem
 alles so lassen                         <kb>
100 i$=s$(i):printci$;i;i$:gos
ub210                                    <mg>
110 forj=i-1to1step-1                    <db>
120 s$(j+1)=s$(j):printci$;j;s
$(j+1):gosub210                          <co>
130 ifi$<=s$(j-1)then150:rem e
infuegen                                 <pe>
140 s$(j)=i$:printci$;j;s$(j):
gosub210:goto160                         <ao>
150 nextj                                <kj>
160 nexti                                <na>
170 printchr$(19)cr$rn$"sortie
rt:"                                     <la>
180 fori=i=1tom-9step10                  <fd>
190 forj=1to1+9:printcr$s$(j):
nextj,i:end                              <ch>
200 printchr$(19)ci$rn$"sortin
g:":return                               <ii>
210 fort=1to500:next:return              <ik>
220 datapascal,diskette,floppy
,sprites,drucker,chip,ram,rom,
videochip,basic                          <fb>
230 dataassembler,maske,amiga,
datei,text,grafik,pixel,softwa
re,tastatur,monitor                      <ob>
```

## Select-Sort – 1 –

```
10 rem sortierroutine                    <ab>
20 rem selection-sort                    <fg>
30 fori=1to26:cr$=cr$+chr$(29)
:next                                    <dn>
40 ci$=left$(cr$,12):rn$=chr$(
18)                                      <jn>
50 m=20:dims$(m)                         <eo>
60 scnclr:printrn$"unsortiert:
"                                        <ak>
70 fori=1tom:reads$(i):prints$
(i):next:gosub160                        <na>
80 fori=mto2step-1:m$=""                 <no>
90 forj=1toi                             <cn>
100 ifs$(j)>m$thenm$=s$(j):k=j           <cg>
110 nextj                                <ag>
120 s1$=s$(i):s$(i)=s$(k):s$(k
)=s1$:printci$;k;s$(k)                   <oo>
130 nexti                                <fi>
140 printchr$(19)cr$rn$"sortie
rt:"                                     <jl>
150 forj=1tom:printcr$s$(j):ne
xtj:end                                  <ml>
160 printchr$(19)ci$rn$"sortin
g:":return                               <ap>
170 fort=1to500:next:return              <ab>
180 datapascal,diskette,floppy
,sprites,drucker,chip,ram,rom,
videochip,basic                          <ie>
190 dataassembler,maske,amiga,
datei,text,grafik,pixel,softwa
re,tastatur,monitor                      <ok>
```

## Bubble-Sort – 1 –

```
10 rem sortierroutine                    <ab>
20 rem bubble sort                       <be>
30 fori=1to26:cr$=cr$+chr$(29)
:next                                    <dn>
40 ci$=left$(cr$,12):rn$=chr$(
18)                                      <jn>
50 m=20:dims$(m)                         <eo>
60 scnclr:printrn$"unsortiert:
"                                        <ak>
70 fori=1tom:reads$(i):prints$
(i):next:gosub160                        <na>
80 fori=m-1to1step-1:rem beim
vorletzten anfangen                      <mj>
90 forj=1toi:rem gegenprobe vo
n vorne                                  <jc>
100 ifs$(j)<=s$(j+1)then120              <bl>
110 s1$=s$(j):s$(j)=s$(j+1):s$
(j+1)=s1$:printci$;j;s$(j+1):g
osub170                                  <aj>
120 nextj                                <co>
130 nexti                                <fi>
140 printchr$(19)cr$rn$"sortie
rt:"                                     <jl>
150 forj=1tom:printcr$s$(j):ne
xtj:end                                  <ml>
```

```
160 printchr$(19)ci$rn$"sortin
g:":return                    <ap>
170 fort=1to500:next:return   <ab>
180 datapascal,diskette,floppy
,sprites,drucker,chip,ram,rom,
videochip,basic               <ie>
190 dataassembler,maske,amiga,
datei,text,grafik,pixel,softwa
re,tastatur,monitor           <ok>
```

## Shell-Sort — 1 —

```
10 rem sortierroutine         <ab>
20 rem shell-sort             <gg>
30 fori=1to26:cr$=cr$+chr$(29)
:next                         <dn>
40 ci$=left$(cr$,12):rn$=chr$(
18)                           <jn>
50 m=20:dims$(m),s1$(m):sw=int
(m/2)                         <mp>
60 scnclr:printrn$"unsortiert:
"                             <ak>
70 fori=1tom:reads$(i):prints$
(i):next:gosub320             <na>
80 fori=1tosw                 <lc>
90 forj=1toint(m/sw)          <mc>
100 s1$(j)=s$((j-1)*sw+i)     <kf>
110 nextj                     <ag>
120 s1=j-1:gosub220:rem insert
-sort benutzen                <nd>
130 forj=1toint(m/sw)         <jd>
140 s$((j-1)*sw+i)=s1$(j):prin
tci$;j;s$((j-1)*sw+i):gosub330 <ol>
150 nextj                     <kj>
160 nexti                     <na>
170 sw=int(sw/2)              <cd>
180 ifswthengoto80            <pe>
190 printchr$(19)cr$rn$"sortie
rt:"                          <ca>
200 forj=1tom:printcr$s$(j):ne
xtj:end                       <pn>
210 rem untersortierroutine   <pb>
220 fori1=2tos1               <ki>
230 ifs1$(i1)>=s1$(i1-1)then30
0                             <bg>
240 i1$=s1$(i1)               <bg>
250 forj1=i1-1tos1step-1      <hg>
260 s1$(j1+1)=s1$(j1)         <em>
270 ifi1$<=s1$(j1-1)then290   <nf>
280 s1$(j1)=i1$:goto300       <nl>
290 nextj1                    <oi>
300 nexti1                    <df>
310 return                    <pm>
320 printchr$(19)ci$rn$"sortin
g:":return                    <oo>
330 fort=1to500:next:return   <od>
340 datapascal,diskette,floppy
,sprites,drucker,chip,ram,rom,
videochip,basic               <co>
350 dataassembler,maske,amiga,
datei,text,grafik,pixel,softwa
re,tastatur,monitor           <fi>
```

## Quick-Sort — 1 —

```
10 rem sortierroutine         <ab>
20 rem quick-sort             <be>
30 fori=1to26:cr$=cr$+chr$(29)
:next                         <dn>
40 ci$=left$(cr$,12):rn$=chr$(
18)                           <jn>
50 m=20:mg=100:dims$(m),le(mg)
,ri(mg)                       <gk>
60 x=0:le(1)=1:ri(1)=m        <jg>
70 scnclr:printrn$"unsortiert:
"                             <gk>
80 fori=1tom:reads$(i):prints$
(i):next:printchr$(19)ci$rn$"s
orting:"                      <jc>
90 gosub120                   <gd>
100 printchr$(19)cr$rn$"sortie
rt:"                          <mk>
110 forj=1tom:printcr$s$(j):ne
xtj:end                       <aa>
120 x=x+1:ifle(x)>=ri(x)then24
0                             <ag>
130 y=le(x):w=ri(x)           <ci>
140 s1$=s$(int((y+w)/2))      <no>
150 ify>wthen220              <mm>
160 ifs$(y)<s1$theny=y+1:goto1
60                            <ic>
170 ifs$(w)>s1$thenw=w-1:goto1
70                            <nc>
180 ify>wthen220              <ko>
190 s2$=s$(y):s$(y)=s$(w):s$(w
)=s2$:printci$;w;s$(w):gosub25
0                             <dd>
200 y=y+1:w=w-1               <hd>
210 goto150                   <na>
220 ri(x+1)=w:le(x+1)=le(x):go
sub120                        <fe>
230 le(x+1)=y:ri(x+1)=ri(x):go
sub120                        <ch>
240 x=x-1:return              <ck>
250 fort=1to500:next:return   <be>
260 datapascal,diskette,floppy
,sprites,drucker,chip,ram,rom,
videochip,basic               <ce>
270 dataassembler,maske,amiga,
datei,text,grafik,pixel,softwa
re,tastatur,monitor           <fm>
```

# Universelle Datei

**Dateien können auf die verschiedenste Weise verwaltet werden. Die „relative" Dateiablage ist weniger verbreitet, da sie schwieriger zu programmieren ist. Mit diesem Listing geht's aber problemlos.**

Dieses Programm ermöglichst relative Dateiverwaltung aller möglichen Variationen bis 254 Byte pro Datensatz und verwaltet bis 20 unterschiedliche Dateien. Es ist möglich, pro Datensatz bis zehn Datenfelder einzurichten. Bei 40 Zeichen werden allerdings nur sechs Datenfelder zur Verfügung gestellt. Bei kürzeren Datenfeldern können je nach Recordlänge bis zu zehn Datenfelder definiert werden.
Nach dem Programmstart erscheint das Menü, dann kann mit dem Cursor der einzelne Menüpunkt angewählt werden. Wenn keine Datei vorliegt, antwortet das Programm mit entsprechenden Hinweisen. Nur der Menüpunkt „Datei einrichten" läßt sich anwählen, um eine Datei zu erzeugen!
Nach der Datums- und Dateinamen-Eingabe wird nach der Feldbezeichnung und Feldlänge gefragt. Als Feldbezeichnung sollten etwa bei Adressen, Name, Straße, Postleitzahl, Ort, Telefon, genannt werden. Der Strich bei Feldbezeichnung markiert lediglich die mögliche Eingabelänge. Die Feldlänge kann 25 Zeichen nicht überschreiten. Bei erreichter Recordlänge, Escape oder dem Erreichen des zehnten Datenfeldes wird die Funktion beendet und die Parameter werden automatisch festgestellt.
Danach erscheint die Eingabe für die Datensätze. Der Strich markiert nur die Datenfeldlänge. Es ist wichtig, daß im ersten Datenfeld die Bezeichnung steht, die für die Identifizierung maßgebend ist, also bei Adressen der Personenname oder bei Bibliotheken der Buchtitel. Sie darf sich auf keinen Fall in einem anderen Datensatz wiederholen, da dieser Datensatz sonst nicht angenommen wird. Nummern sind nicht zu empfehlen, da hier keine Kontrolle für doppelte Eingaben, die automatisch ausgeführt wird, möglich ist.

Ebenso sind Bezeichnungen, die in mehreren Datensätzen gleich sein müssen, also bei Adressen der Ort oder bei Bibliothek der Autor, im ersten Datenfeld zu vermeiden. Die Recordnummer wird vom Computer intern ermittelt und verwaltet und tritt nicht in Erscheinung. Wenn die Eingabefunktion beendet werden soll, ist die Taste Escape zu drücken. Dies gilt für alle Funktionen.

Dann erscheint das Anfangsmenü wieder. Wenn jetzt die Menüpunkte „Programm beenden", „Datei löschen" und „Datei Wahl" angewählt werden, schließen die Datenkanäle automatisch und die Parameterdatei wird als „Par.Dateiname" abgelegt. Die Dateinamen der unterschiedlichen Dateien werden unter der Datei „Rel.Files-Liste" zusammengefaßt und gespeichert. Vor dem Zugriff auf eine Datei muß „Datei Wahl" gewählt werden, denn nur hier werden die Kanäle geöffnet. Der entsprechende Dateiname wird mit dem Cursor ausgesucht und mit Return werden die entsprechenden Parameter geladen und die Kanäle geöffnet.

Nun können die Menüpunkte „Datenausgabe", „Dateneingabe" und „Daten ändern" angewählt werden. Andernfalls erscheint eine entsprechende Meldung. Da das Abbrechen des Programmes durch die Stoptaste nicht möglich ist, muß der Menüpunkt „Programm beenden" benutzt werden, um alle Kanäle zu schließen und Paramterdateien zurückzuschreiben.
Die Funktion der Suchroutine ist folgende: Nach Eingabe der Anfangsbuchstaben des Suchbegriffs werden alle Datensätze, die mit den gleichen Zeichen beginnen, hinter die Forderung Suchbegriff ausgegeben. Dann wartet das Programm drei Sekunden. Wenn in dieser Zeit nicht Return gedrückt wird, um den I.D.-Code zu übernehmen, wird an dieser Stelle der nächste Datensatz vorgeschlagen.

*Dieter Kukkel* □

```
10 rem relative datei =========c16 <eg>
20 rem (p) commodore welt team <ho>
30 rem ========================== <ng>
40 rem (c) by dieter kukkel c128 <lk>
50 rem (v) a. mittelmeyer c16 <la>
60 rem <ah>
70 rem basic v3.5 <nl>
80 rem c16/116/plus4 + floppy <dn>
90 rem ========================== <jg>
100 gosub110:sys818:goto160 <lg>
110 restore:fori=818to830 <hj>
120 reada:pokei,a:next:return <bk>
130 data 120,169,102,141,038,003 <bd>
140 data 169,204,141,039,003,088 <fj>
150 data 096 <oo>
160 gosub1690:scnclr <cc>
170 dim n$(8),t(8),x(8),y(8),dn$(25),df$(20),lr(20),dx$(20),id$(500) <ig>
180 sp$=chr$(32):pu$=chr$(163):ad=0:lf=0:rp=1:s=1:fl=0:f=0 <ol>
190 forj=0to38:s8$=s8$+sp$:nextj:restore210 <ng>
200 fori=1to7:readn$(i),t(i),x(i),y(i):nexti <kk>
210 data " datei-auswahl ",1,0,3," daten-ausgabe ",2,0,4 <gg>
220 data " daten-eingabe ",3,0,5," daten-aendern ",4,0,6 <cl>
230 data " datei einrichten ",5,20,3," datei loeschen ",6,20,4 <ie>
240 data " programm beenden ",0,20,5 <kl>
250 : <cp>
260 gosub850:char1,1,0,rf$+"relative index-datei"+b$+b3$+"c16 "+ro$ <mk>
270 char1,1,1,rf$+"(c) commodore welt/dieter kukkel"+ro$ <jh>
280 char1,0,8,rf$+s8$+ro$:char,1,8,rf$+"cursor-wahl :"+ro$ <dg>
290 fori=1to7:char1,x(i),y(i),n$(i):nexti <ic>
300 char,x(s),y(s),rf$+n$(s)+ro$:char,21,8,rf$+"funktion"+str$(t(s))+ro$ <hn>
310 getkey z$:char,x(s),y(s),n$(s) <ak>
320 ifz$=chr$(145)then s=s-1:ifs<1thens=s+7 <nd>
330 ifz$=chr$(17) then s=s+1:ifs>7thens=s-7 <go>
340 ifz$=chr$(29) then s=s+4:ifs>7thens=s-7 <nl>
350 ifz$=chr$(157)then s=s-4:ifs<1thens=s+7 <hc>
360 ifz$<>chr$(13)then300 <dn>
370 ifs>1ands<5then ifdn$=""then gosub970:goto260 <em>
380 ifs=3ors=4then gosub1060 <fl>
390 on t(s) gosub 420,590,720,590,1120,1590:ift(s)<>0then260 <bd>
400 gosub1470:scnclr:gosub110:poke820,101:poke825,242:sys818:end <cd>
410 : <hb>
420 iffl=0thengosub1360:iffl=0thenreturn <fb>
430 s=1:x=1:y=11:fori=1tofl:ifx>30thenx=1:y=y+1 <ap>
440 char1,x,y,dn$(i):x=x+20:nexti:ifdn$<>""then gosub1470 <go>
450 gosub1040:iff=0then char1,1,24,rf$+"welche datei wird gewuenscht ?"+ro$ <ak>
460 iff=1then char1,1,24,rf$+"welche datei soll geloescht werden ?"+ro$ <kl>
470 z=s-1:y=int(z/2)+11:x=((z/2)-int(z/2))/.5*20 <hn>
480 char,13,23,rf$+b3$+ro$:char,1,23,rf$+"datei-wahl"+str$(s)+ro$ <gd>
490 char,x,y,rf$+sp$+dn$(s)+sp$+ro$ <mb>
500 getkey z$:char,x,y,sp$+dn$(s)+sp$ <oh>
510 ifz$=chr$(145)then s=s-2:ifs<1thens=s+20 <kj>
520 ifz$=chr$(17) then s=s+2:ifs>20thens=s-20 <dd>
530 ifz$=chr$(29) then s=s+1:ifs>20thens=s-20 <cn>
540 ifz$=chr$(157)then s=s-1:ifs<1thens=s+20 <na>
550 ifz$<>chr$(13)then470 <no>
560 ifs=<flthen dn$=dn$(s):s=1:else s=1:goto470 <mb>
570 iff=1then return:else gosub1040:goto1410 <pl>
580 : <mg>
590 gosub870:char1,0,23,rf$+">>> achtung <<<"+b2$+"record nr."+ro$ <la>
600 char1,0,24,rf$+"mit return uebernehmen"+ro$:rn=0 <ok>
610 char1,0,2,"suchbegriff:":inputnn$ <gc>
620 fori=1toad:char1,27,23,rf$+str$(i)+ro$:ifleft$(id$(i),len(nn$))<>nn$then640 <ee>
630 char1,14,2,id$(i):gosub1680:gett$:ift$=chr$(13)then660 <ni>
640 char1,14,2,left$(s8$,25):nexti:gosub1040 <og>
650 char1,5,24,rf$+"datensatz ist nicht vorhanden !"+ro$:gosub1680:gosub1040:goto830 <gm>
660 rn=i:gosub1040:gosub900:ifrl<89then680 <pm>
670 fori=1todf:forj=1tolr(i):get#lf,eg$:dx$(i)=dx$(i)+eg$:nextj,i:go
```

```
to690                                                    <in>
680 input#1f,rc$:lr=1:fori=1todf:d
x$(i)=mid$(rc$,lr,lr(i)):lr=lr+lr(
i):nexti                                                 <mk>
690 fori=1todf:char1,0,i*2+2,df$(i
):char1,14,i*2+2,dx$(i):nexti                            <nd>
700 gosub950:ifs=4then fori=1todf:
goto730:else830                                          <co>
710 :                                                    <ml>
720 gosub870:ad=ad+1:rn=ad:fori=1t
odf                                                      <gl>
730 char1,0,i*2+2,df$(i):forj=1tol
r(i):char1,13+j,i*2+3,pu$:nextj                          <lj>
740 char1,12,i*2+2,"":inputdx$(i):
iflen(dx$(i))>lr(i)ordx$(i)=""then
730                                                      <be>
750 ifi>1then790:else ifs=4then id
$(rn)=""                                                 <kh>
760 f=0:forj=1toad:ifid$(j)<>dx$(1
)then780                                                 <im>
770 char1,1,23,rf$+"i.d.-code exis
tiert"+ro$:f=1:gosub1680                                 <ad>
780 nextj:gosub1040:iff=0then id$(
rn)=dx$(1):else730                                       <kp>
790 lr$="":forj=1tolr(i):lr$=lr$+s
p$:nextj                                                 <oi>
800 dx$(i)=left$(dx$(i)+left$(lr$,
abs(lr(i)-len(dx$(i)))),lr(i))                           <bk>
810 rc$=rc$+dx$(i):nexti:gosub900:
print#1f,rc$:gosub900:gosub950                           <io>
820 gosub1040:char1,1,24,rf$+"date
nsatz ist gesichert"+ro$:gosub1680                       <im>
830 gosub1020:ift$=chr$(27)thenret
urn:else ifs=3then720:else590                            <ji>
840 :                                                    <na>
850 scnclr:char1,0,0,rf$+s8$+ro$:c
har1,0,1,rf$+s8$+ro$:goto1040                            <no>
860 :                                                    <pi>
870 gosub850:char1,0,0,rf$+n$(s)+s
p$+dn$+ro$                                               <jn>
880 char1,1,1,rf$+"aktualisiert am
 : "+j$+ro$:return                                       <ai>
890 :                                                    <de>
900 ah=int(rn/256):al=rn-256*ah:pr
int#15,"p"+chr$(96+1f)+chr$(al)+ch
r$(ah)+chr$(rp)                                          <kc>
910 char1,1,23,rf$+ds$+ro$:ifds<20
ords=50ords=62then return                                <jd>
920 ifds=52then gosub1470:else gos
ub1520                                                   <fd>
930 run                                                  <nf>
940 :                                                    <jj>
950 fori=1todf:dx$(i)="":nexti:rc$
="":return                                               <af>
960 :                                                    <mb>
970 char1,1,23,rf$+"datei-auswahl
oder "+ro$                                               <ae>
980 char1,1,24,rf$+"datei-einricht
en waehlen"+ro$:gosub1680:return                         <ah>
990 :                                                    <pn>
1000 char1,0,2,rf$+left$(s8$,18):r
eturn                                                    <ec>
1010 :                                                   <cf>
1020 char1,1,23,rf$+"funktion wied
erholen"+ro$                                             <fn>
1030 char1,1,24,rf$+"ja=(space)"+b
2$+"/"+b2$+"nein=(escape)"+ro$:get
keyt$                                                    <cc>
1040 char1,0,23,rf$+s8$+ro$:char1,
0,24,rf$+s8$+ro$:return                                  <gm>
1050 :                                                   <hg>
1060 char1,0,10,rf$+" aktuelles da
tum "+ro$:ifj$=""thenj$="01.09.198
7"                                                       <od>
1070 char1,20,10,j$:char1,18,10,""
:inputp$:char1,0,10,s8$                                  <fe>
1080 iflen(p$)<>10orval(left$(p$,2
))>31then1060                                            <ch>
1090 ifval(mid$(p$,4,2))>12orval(r
ight$(p$,4))<1980then1060                                <mm>
1100 j$=p$:return                                        <gp>
1110 :                                                   <oo>
1120 ifdn$=""then gosub1360:else g
osub1470                                                 <fj>
1130 iffl=20thenchar1,1,23,rf$+"ma
x. dateien-anzahl ist erreicht"+ro
$:gosub1680:return                                       <ia>
1140 ifad>0thenfori=1toad:id$(i)="
":nexti:ad=0                                             <lj>
1150 gosub850:char1,1,0,rf$+"relat
ive index-datei einrichten"+ro$:go
sub1060                                                  <lj>
1160 dn$="":char1,0,10,rf$+" datei
-name "+ro$:fori=16to27                                  <dg>
1170 char1,i,11,pu$:nexti:char1,14
,10,"":inputdn$:char1,0,10,s8$                           <cg>
1180 char1,0,11,s8$:iflen(dn$)>12o
rdn$=""then1160                                          <hh>
1190 fori=1tofl+1:ifdn$(i)=dn$then
char1,1,23,rf$+"datei-name existie
rt"+ro$:goto1160                                         <gh>
1200 ifdn$(i)=""then dn$(i)=dn$:fl
=i                                                       <af>
1210 nexti:gosub1040:char1,0,2,s8$
:char1,1,1,rf$+"parameter-eingabe"
+b3$+dn$+ro$                                             <kn>
1220 rl=0:b=253:i=0:do:i=i+1:lr(i)
=0:df$(i)=""                                             <ap>
1230 char1,0,i*2+2,rf$+" feld nr."
+str$(i)+sp$+ro$                                         <me>
1240 gosub1000:char1,0,2,rf$+" fel
d-bezeichnung "+ro$                                      <hd>
1250 forj=15to26:char1,j,i*2+3,pu$
:nextj:char1,13,i*2+2,"":inputdf$(
i)                                                       <ei>
1260 iflen(df$(i))>12ordf$(i)=""th
```

```
enforj=13to35:char1,j,i*2+2,sp$:ne
xtj:goto1240                          <fg>
1270 gosub1000:char1,0,2,rf$+" fel
d-laenge "+ro$:char1,27,i*2+2,sp$:
inputlr(i)                            <oe>
1280 iflr(i)<1orlr(i)>25orlr(i)>bt
hen1270:elserl=rl+lr(i):b=b-lr(i)     <dg>
1290 gosub1000:char1,0,2,rf$+" rec
ord-laenge "+ro$:char1,36,2,b3$       <bh>
1300 char1,29,2,rf$+str$(rl)+sp$+r
o$:char1,35,2,str$(b):ifi=10orb=0t
henexit                               <kh>
1310 char1,1,23,rf$+"weitere daten
felder einrichten"+ro$                <ik>
1320 gosub1030:ift$=chr$(27)then e
xit                                   <hp>
1330 loop:df=i:rl=rl+1:gosub1020:i
ft$=chr$(27)thengosub1440:s=3:goto
720                                   <nb>
1340 dn$(fl)="":fl=fl-1:goto1150      <hb>
1350 :                                <na>
1360 open2,8,3,"0:rel.files-liste,
s,r":gosub910:ifds=62thenclose2:go
to1380                                <kj>
1370 input#2,fl:fori=1tofl:input#2
,dn$(i):nexti:close2:return           <fm>
1380 char1,1,23,rf$+"dateien nicht
 existent"+ro$                        <pj>
1390 char1,1,24,rf$+"erst datei ei
nrichten"+ro$:gosub1680:return        <mf>
1400 :                                <de>
1410 open2,8,3,"0:par."+dn$+",s,r"
:gosub910                             <gb>
1420 input#2,ad,df,rl,j$:fori=1tod
f:input#2,df$(i),lr(i):nexti          <fl>
1430 fori=1toad:input#2,id$(i):nex
ti:close2                             <bd>
1440 open15,8,15:openlf,8,lf,dn$+"
,l,"+chr$(rl):gosub910                <hm>
1450 char1,1,24,rf$+"dateiarbeit k
ann begonnen werden."+ro$:gosub168
0:return                              <fp>
1460 :                                <kn>
1470 ifdn$=""thenreturn:else gosub
1540                                  <ed>
1480 print#15,"s:par."+dn$            <ad>
1490 open2,8,3,"0:par."+dn$+",s,w"
:gosub910                             <ng>
1500 print#2,ad","df","rl","j$:for
i=1todf:print#2,df$(i)","lr(i):nex
ti                                    <am>
1510 fori=1toad:print#2,id$(i):nex
ti                                    <ng>
1520 close2:cloself:close15:dn$=""
:return                               <hb>
1530 :                                <dj>
1540 char1,1,24,rf$+"daten-kanaele
 werden geschlossen "+ro$             <hl>
1550 print#15,"s:rel.files-liste"     <jc>
1560 open2,8,3,"0:rel.files-liste,
s,w":gosub910                         <np>
1570 print#2,fl:fori=1tofl:print#2
,dn$(i):nexti:close2:return           <eb>
1580 :                                <jo>
1590 f=1:gosub420:gosub1040:iffl=0
thenreturn                            <dd>
1600 char1,1,23,rf$+"loesch-vorgan
g abbrechen ?"+ro$:gosub1030          <dk>
1610 f=0:ift$<>chr$(27)then dn$=""
:return                               <aj>
1620 char1,1,24,rf$+"datei "+dn$+"
 wird geloescht"+ro$                  <ma>
1630 fori=1tofl:ifdn$=dn$(i)thendn
$(i)=""                               <eh>
1640 ifdn$(i)=""thendn$(i)=dn$(i+1
):dn$(i+1)=""                         <pm>
1650 nexti:fl=fl-1:open15,8,15:pri
nt#15,"s:par."+dn$:print#15,"s:"+d
n$                                    <hi>
1660 print#15,"s:rel.files-liste":
close15:iffl>0then gosub1560          <np>
1670 dn$="":return                    <dn>
1680 forxx=1to2000:next:return        <hd>
1690 b$=chr$(32):b2$=b$+b$            <pc>
1700 b3$=b2$+b$:b$=b3$+b3$+b2$+b2$    <dj>
1710 rf$=chr$(18):ro$=chr$(146):re
turn                                  <eb>
1720 rem ======================       <ff>
1730 rem 12277 bytes memory           <ai>
1740 rem 06348 bytes program          <pn>
1750 rem 00217 bytes variables        <hf>
1760 rem 02037 bytes arrays           <an>
1770 rem 00642 bytes strings          <cp>
1780 rem 03033 bytes free (0)         <mf>
1790 rem (bei 3 datensaetzen)         <mi>
1800 rem ======================       <dl>
```

HARDCOPIES
Grundlagen

GRAFIKBILDSCHIRM
Organisation und Hardcopies

# Für jeden Computer und jeden Drucker

**Hardcopy zu klein, zu groß, zu wenig dicht, falsches Verhältnis von Breite und Höhe oder für den speziellen Drucker nicht verfügbar? In zwei Folgen bringen wir Grundlagen, Möglichkeiten und Listings.**

Mit den Grafikbefehlen des BASIC V3.5 kann jeder Bildpunkt auf bequeme Weise angesprochen werden. Schwierig aber wird es, wenn es gilt, eine Hardcopy des Grafikbildschirmes auszudrucken. Im Bedienerhandbuch zum Rechner ist darüber nichts zu finden, zumindest nicht im C16-Handbuch. Dieses bricht mitten im Anhang einfach ab.
Dem C116- oder Plus/4-Handbuch können wir auf Seite 228 entnehmen, daß der Bildschirmspeicher für die Grafik den Speicherplatz von $2000 bis $3FFF belegt. Um damit allerdings etwas anfangen zu können, bedarf es entweder weiterer Informationen oder einiger Experimente.
Wie wird die Grafik, die immerhin die Information über 64.000 Bildpunkte enthalten muß, in einem Adreßraum von etwa 8.000 Bytes gespeichert? Dies kann nur dadurch geschehen, daß ein Byte die Information über acht Bildpunkte enthält. Wie wir wissen, besteht ein Byte aus acht Bits. Jeder Bildpunkt kann entweder gesetzt oder nicht gesetzt sein. Ist er gesetzt, so ist auf dem Bildschirm die Vordergrundfarbe zu sehen, ansonsten die Hintergrundfarbe.
Gesetzte und nicht gesetzte Bildpunkte lassen sich in Form von gesetzten und nicht gesetzten Bits speichern. Wie dies im Einzelnen geschieht, wollen wir jetzt untersuchen. Dazu schalten wir die hochauflösende Grafik mit Text ein durch:

GRAPHIC 2,1

Außerdem ist die Anfangsadresse $2000 des Grafikbildschirmes in die dezimale Darstellung umzurechnen.

?DEC("2000")

Die dezimale Adresse lautet 8192. Ein kleiner Versuch könnte vielleicht etwas zeigen.

POKE 8192,255

Es erscheint ein kleiner waagrechter Strich in der linken oberen Ecke des Bildschirms. Die Zahl 255 wird in einem Byte durch acht gesetzte Bits realisiert. Acht nebeneinanderliegende schwarze Pünktchen, die im Computerjargon Pixel heißen, sind zu sehen. Ein gesetztes Bit bedeutet also ein gesetztes Pixel. Noch ist nicht klar, welches Bit welchem Pixel entspricht.

```
10 rem listing 1======(hardcopies) <nf>
20 rem by alfons mittelmeyer       <if>
30 rem ============================ <ng>
40 graphic1,1:clr                  <ap>
50 ax=dec("2000")                  <fk>
60 fori=axtoax+7999                <cf>
70 pokei,255                       <mc>
80 char,0,24,hex$(i)+str$(i)       <aa>
90 getkeyx$                        <gd>
100 next                           <ek>
110 graphicclr                     <bb>
120 rem ============================ <gn>
130 rem  p r o g r a m m e n d e   <jf>
140 rem ============================ <ge>
```

```
10 rem listing 2======(hardcopies) <fc>
20 rem demoprogramm                <ao>
30 rem dieses programm liest ein   <ha>
40 rem buchstabenmuster, konver-   <np>
50 rem tiert es, und gibt es dabei <dk>
60 rem auf den bildschirm aus.     <dg>
70 rem ---------------------------- <an>
80 rem  matrix aus zeichensatzrom  <ae>
90 rem ---------------------------- <cf>
100 ax=53256:rem buchstabe a       <gd>
110 poke1177,62:rem peek aus rom   <fe>
120 fori=0to7                      <hp>
130 a(i)=peek(ax+i)                <ik>
140 next                           <jl>
150 poke1177,63:rem peek aus ram   <hi>
160 gosub200:goto160               <ed>
170 rem ---------------------------- <ef>
180 rem  matrix-konvertierung      <kc>
190 rem ---------------------------- <ca>
200 forj=0to7                      <ld>
210 a=0                            <co>
220 fori=7to0step-1                <ci>
230 a(i)=2*a(i)                    <dk>
240 a=a+a                          <en>
250 ifa(i)>255thena=a+1:a(i)=a(i)-
256:print"Q";:elseprint".";       <bk>
260 nexti:print                    <jg>
270 b(j)=a                         <cb>
280 nextj                          <ld>
290 fori=0to7                      <fk>
300 a(i)=b(i)                      <ha>
310 nexti:return                   <kl>
320 rem ============================ <il>
330 rem  p r o g r a m m e n d e   <al>
340 rem ============================ <oc>
```

POKE 8192,1
POKE 8192,128

Das linke Pixel wird durch das höchstwertige Bit vertreten, das rechte Pixel durch das niederstwertige Bit. Wir können gespannt sein, ob die nächste Speicheradresse die rechts oder unten anschließenden Pixel wiedergibt.

POKE 8192,128
POKE 8193,64
POKE 8194,32

```
listing 3============(hardcopies)
hardcopy-maschinenteil
assemblerlisting
================================
. 0138  20 4e 01 jsr $014e
. 013b  a0 08    ldy #$08
. 013d  a2 08    ldx #$08
. 013f  16 d2    asl $d2,x
. 0141  2a       rol
. 0142  ca       dex
. 0143  d0 fa    bne $013f
. 0145  ea       nop
. 0146  ea       nop
. 0147  20 d2 ff jsr $ffd2
. 014a  88       dey
. 014b  d0 f0    bne $013d
. 014d  60       rts
. 014e  a0 00    ldy #$00
. 0150  85 d2    sta $d2
. 0152  a2 08    ldx #$08
. 0154  b1 d0    lda ($d0),y
. 0156  95 d2    sta $d2,x
. 0158  05 d2    ora $d2
. 015a  85 d2    sta $d2
. 015c  e6 d0    inc $d0
. 015e  d0 02    bne $0162
. 0160  e6 d1    inc $d1
. 0162  ca       dex
. 0163  d0 ef    bne $0154
. 0165  60       rts
================================
```

```
10 rem listing 4======(hardcopies) <gk>
20 rem by alfons mittelmeyer       <if>
30 rem ============================ <ng>
40 rem maschinensprachteil          <gn>
50 rem fuer hires-                  <ol>
60 rem hardcopyprogramme            <kg>
70 rem ============================ <po>
80 for i=312to357                   <ig>
90 read a:poke i,a:next             <dk>
100 data 032,078,001,160,008,162    <mm>
110 data 008,022,210,042,202,208    <gd>
120 data 250,234,234,032,210,255    <ab>
130 data 136,208,240,096,160,000    <ej>
140 data 133,210,162,008,177,208    <dg>
150 data 149,210,005,210,133,210    <im>
160 data 230,208,208,002,230,209    <jg>
170 data 202,208,239,096            <dg>
180 rem ============================ <cn>
190 rem basicteil ist anzufuegen    <oi>
200 rem ============================ <ne>
```

POKE 8195,16
POKE 8196,8
POKE 8197,4
POKE 8198,2
POKE 8199,1

Die Reihenfolge vom höchstwertigen Bit zum niederstwertigen entspricht der Reihenfolge der Pixel von links nach rechts. Mit aufsteigenden Speicheradressen dagegen scheint es am Bildschirm abwärts zu gehen. Jedoch werden wir gleich eines Besseren belehrt.

POKE 8200,255

Wir haben damit die oberste Reihe der nächsten Spalte angesprochen. Da es wohl etwas mühsam ist, sich den ganzen Grafikbereich mit POKEs per Hand zu erschließen, soll ein kurzes BASIC-Listing namens HIRES uns die Arbeit abnehmen. Sie finden es als Listing Nr. 1 im Anhang zu diesem Artikel.

## ACHT BYTES PRO ZEICHEN – 40 ZEICHEN PRO ZEILE

HIRES beschreibt 8.000 Adressen ab $2000 (8192 dezimal) mit dem Wert 255. Nach dem Programmstart ist in der linken oberen Bildschirmecke wieder der kurze Strich zu sehen, den wir schon von unseren POKE-Versuchen her kennen. Links unten am Bildschirm wird die jeweils beschriebene Speicheradresse in hexadezimaler und dezimaler Form dargestellt. Sobald irgendeine Taste gedrückt wird, ist die nächste Speicheradresse an der Reihe. Etwas Geduld ist vonnöten, soll die Organisation des Bildschirmaufbaus abgelesen werden. Nach achtmaligem Tastendruck befindet sich die kurze Linie in der zweiten Spalte. Wie weiteres Drücken zeigt, geht der Aufbau zeilenweise vonstatten. Acht Linien, die jeweils acht Pixel breit sind, füllen von oben nach unten den Raum, den im Textmodus ein einzelnes Zeichen auf dem Bildschirm benötigen würde. Solche „Zeichen" werden zeilenweise aufgebaut, mit jeweils 40 „Zeichen" pro Zeile. Der Bildschirm hat, wie im Textmodus, auch jetzt 25 Zeilen, 40 mal acht Pixel ergeben 320 Bildpunkte in der Breite, 25 mal acht Pixel sind 200 Bildpunkte in der Länge. Wir wissen jetzt, wie die Bildpunkte der hochauflösenden Grafik untergebracht sind und können uns Gedanken über das Ausdrucken machen.

## VORÜBERLEGUNGEN ZUM DRUCKEN

Statt eines einzelnen Byte für ein Zeichen haben wir im Grafimodus gleich acht Bytes vor uns. Im Textmodus hatten wir einen Code, der dem Drucker sagte, welches Zeichenmuster er aus seiner Bibliothek auswählen solle. Jetzt haben wir das Zeichenmuster selbst vorliegen.
Leider ist es anders organisiert, als es der Drucker verarbeiten kann. Auf dem Bildschirm bestimmt ein Byte acht nebeneinander liegende Pixel. Beim Drucker jedoch liegen die Nadeln nicht neben-, sondern untereinander. Würden wir die Zeichenmuster so an den Drucker senden, wie sie beim Rechner angeordnet sind, so erschienen sie auf dem Papier alle um 90 Grad gedreht. Vor dem Ausdruck ist folglich eine Konvertierung der Zeichenmatrix vorzunehmen. Listing Nr. 2 zeigt, wie das in BASIC realisiert werden könnte. Dieses Listing ist nur als Beispiel und nicht zum Abtippen gedacht.

## MATRIX-KONVERTIERUNG

AX enthalte die Anfangsadresse des zu lesenden Zeichenmusters. Mit einer Schleife werden die Inhalte dieser und der folgenden sieben Adressen in die Variablen A(0) bis A(7) eingelesen. In einer weiteren Schleife mit J=0 bis J=7 soll das Zeichenmuster spaltenweise an den Drucker gesendet werden. Das an den Druk-

ter zu sendende Byte soll in Form der Variablen A bereitgestellt werden. Hierzu ist aus den Variablen A(0) bis A(7) jeweils das höchstwertige Bit zu entnehmen. Wir setzen die Variable A auf den Wert Null und sprechen durch eine weitere Schleife mit dem Index I=0 bis I=7 die indizierten Variablen A(0) bis A(7) an. Das höchstwertige Bit ist in die Variable A zu schieben. Hierzu schieben wir die Bits in den indizierten Variablen um eine Stelle nach links mit A(I)=2*A (I). Damit liegen die folgenden Bits für den nächsten Durchgang bereit. Das frühere Bit sechs ist somit jetzt zu Bit sieben geworden und kann im nächsten Durchgang mit erhöhtem J herausgeschoben werden.
Das erhaltene Bit ist jetzt in die Variable A hineinzuschieben. Da BASIC keine Schiebebefehle kennt, verschieben wir den Inhalt von A um eine Stelle nach links mit A=A+A. Hierzu ist jetzt das aus A(I) herausgeschobene Bit zu addieren. Wir realisieren dies durch die IF-Abfrage:

```
IFA(I)>255THENA=A
+1:A(I)=A(I)-256
```

Der Übertrag von 256, der sich bei gesetztem höchstwertigen Bit ergab, mußte natürlich wieder abgezogen werden, so daß für die weitere Bearbeitung nur mehr die restlichen sieben Bits zur Verfügung stehen. Nach der Abarbeitung der inneren Schleife steht in A das an den Drucker zu sendende Byte bereit. Das höchstwertige Bit aus A(0) ist zum höchstwertigen Bit in A geworden, das höchstwertige Bit aus A (7) zum niederstwertigen Bit in A.

Viele Drucker sprechen mit dem höchstwertigen Bit die oberste Nadel an, manche aber mit dem untersten Bit. Für den letzteren Fall müßte die innere Schleife umgekehrt ablaufen, oder die Bits müßten in umgekehrter Richtung in die Variable A geschoben werden. Das erstere wäre zu verwirklichen durch die Schleife:

```
FOR I=7 TO 0 STEP-1
```

Das Schieben nach rechts geschähe durch:

```
A=A/2
IFA(I)>256THENA=A
+128:A(I)=A(I)-256
```

Wenn wir davon ausgehen, daß vorher im Hauptprogramm der Druckerkanal durch den CMD-Befehl geöffnet worden ist, können wir mit PRINT CHR$(A) unser Byte auf den Drucker ausgeben. Nach acht Durchläufen sind alle acht Bytes ausgegeben und es kann der Rücksprung in das aufrufende Hauptprogramm mit RETURN erfolgen.

## PROGRAMMIEREN IN MASCHINENSPRACHE

64 Schiebeoperationen braucht es, bevor ein ganzes Zeichen an den Drukker ausgegeben ist. In BASIC nimmt das eine Menge Zeit in Anspruch. Quälende Langsamkeit bei der Hardcopy wäre die Folge. Es ist daher zu überlegen, ob Listing Nr. 2 nicht besser durch ein Maschinenprogramm zu ersetzen wäre. Mit den Kenntnissen aus dem Einführungskurs im C16/P4-SPECIAL Nr. 3 ausgestattet, können wir uns getrost daranwagen. Damit das Maschinenprogramm nicht mit eventuellen Centronics-Routinen kollidiert, die der eine oder andere benötigt, um über Centronics-Kabel an nicht COMMODORE-kompatible Drukker zu gehen, verlegen wir die Maschinenroutine in den Stack-Bereich des Prozessors.

Ein Zeiger, mit dem die Adressen des Grafikbereichs indirekt ansprechbar sind, wird benötigt.

```
210 rem listing 5=====(hardcopies) <oe>
220 rem by alfons mittelmeyer      <bi>
230 rem ========================== <jc>
240 rem hires-hardcopy fuer        <fe>
250 rem grafikfaehige commodore-   <ah>
260 rem drucker.                   <ci>
270 rem nur in verbindung mit      <ia>
280 rem listing 4                  <ap>
290 rem ========================== <bk>
300 poke208,0:poke209,32           <hb>
310 poke325,9:poke326,128:rem ora
#$80                               <ie>
320 poke321,106:rem ror            <dn>
330 open4,4:cmd4                   <ig>
340 printchr$(8);                  <ie>
350 forj=1to25                     <of>
360 print                          <gl>
370 fori=1to40:sys312:next:nextj   <bo>
380 print#4,chr$(15):close4        <gl>
390 rem ========================== <ag>
400 rem  p r o g r a m m e n d e   <hd>
410 rem ========================== <gk>
```

```
210 rem listing 6=====(hardcopies) <cc>
220 rem by alfons mittelmeyer      <bi>
230 rem ========================== <jc>
240 rem fuer epson-kompatible      <jm>
250 rem drucker                    <ag>
260 rem                            <jj>
270 rem nur in verbindung mit      <ia>
280 rem listing 4                  <ap>
290 rem ========================== <bk>
300 poke208,0:poke209,32           <hb>
310 e$=chr$(27):x$=chr$(13)+chr$(1
0)                                 <ln>
320 open4,4:cmd4                   <hh>
330 printe$"a"chr$(8);             <pm>
340 forj=1to25                     <om>
350 printx$e$"k"chr$(64)chr$(1);   <kb>
360 fori=1to40:sys312:next:nextj   <dc>
370 print#4,e$"2"x$;:close4        <mi>
380 rem ========================== <da>
390 rem  p r o g r a m m e n d e   <ko>
400 rem ========================== <ic>
```

Dafür kommen nur die Adressen der Zero-Page in Frage. Die C16/116/P4-Rechner stellen die Adressen $D0 bis $E8 dem Benutzer zur freien Verfügung. $D0 und SD1 verwenden wir als Adreßzeiger.
Unter Umständen könnte es nützlich sein, ein Flag bereitzustellen, das dem BASIC-Programm meldet, ob ein Zeichen nicht gedruckt zu werden braucht, weil alle acht Bytes vielleicht nur den Wert Null aufweisen. Für dieses Flag wählen wir die Adresse $D2. Die Adressen $D3 bis SDA wollen wir für das zu lesende Zeichenmuster benützen. Das Programmieren kann beginnen.

```
LDY #$00
STY $D2
```

Das Flag wird auf den

```
210 rem listing 7=====(hardcopies) <ca>
220 rem by alfons mittelmeyer      <bi>
230 rem ========================== <jc>
240 rem fuer ibm-kompatible        <dk>
250 rem drucker                    <ag>
260 rem                            <jj>
270 rem nur in verbindung mit      <ia>
280 rem listing 4                  <ap>
290 rem ========================== <bk>
300 poke208,0:poke209,32           <hb>
310 e$=chr$(27):x$=chr$(13)+chr$(1
0)                                 <ln>
320 open4,4:cmd4:printe$":";       <lo>
330 printe$"a"chr$(8)e$"2"         <ja>
340 forj=1to25                     <om>
350 printx$e$"k"chr$(64)chr$(1);   <kb>
360 fori=1to40:sys312:next:nextj   <dc>
370 print#4,e$"a"chr$(12);         <lp>
380 print#4,e$"2"x$:close4         <me>
390 rem ========================== <ej>
400 rem  p r o g r a m m e n d e   <hd>
410 rem ========================== <bb>
```

```
210 rem listing 8=====(hardcopies) <cg>
220 rem by alfons mittelmeyer      <bi>
230 rem ========================== <jc>
240 rem universal-hardcopy         <be>
250 rem fuer alle drucker mit      <ci>
260 rem definierbarem geschaefts-  <ao>
270 rem zeichen. nur in verbindung <kf>
280 rem mit listing 4              <id>
290 rem ========================== <bk>
300 poke208,0:poke209,32           <hb>
310 x$=chr$(13):e$=chr$(8):f$=chr$
(15)                               <pe>
320 open5,4,5:open4,4              <md>
330 forj=1to25                     <nl>
340 print#4,e$x$f$;                <od>
350 fori=1to40:sys334:ifpeek(210)=
0then390                           <bh>
360 print#4,chr$(141);:poke2035,5:
sys65481                           <kd>
370 sys315:print#5:i$=str$(i-1)    <bm>
380 print#4,chr$(16)right$("0"+rig
ht$(i$,len(i$)-1),2)chr$(254);     <ln>
390 next:nextj                     <mc>
400 close4:close5                  <dl>
410 rem ========================== <bb>
420 rem  p r o g r a m m e n d e   <dj>
430 rem ========================== <im>
```

Wert Null gesetzt. Gleichzeitig soll das Y-Register zur indirekten Adressierung dienen und dabei den Wert Null aufweisen, da dabei keine Indizierung vonnöten ist.

**LDX#$08**

Das X-Register ist sowohl Zähler für unsere Leseschleife als auch Index zum Ablegen der gelesenen Bytes in die dafür reservierten Adressen $D3 bis $DA.

**LDA($D0),Y**

**STA $D2,X**

Das durch unseren Zeiger in $D0 und $D1 adressierte Byte aus dem Grafikbereich wird in den Akku geladen und in eine durch das Register X indizierte Adresse von $DA bis $D3 geschrieben. Warum wir nicht mit $D3, sondern $DA beginnen, wird noch erklärt.

**ORA$D2**
**STA $D2**

Alle gelesenen Bytes werden geodert und das Resultat in $D2 abgelegt. Enthält $D2 am Ende immer noch den Wert Null, so bedeutet dies, daß jedes gelesene Byte ein Null-Byte war.

**INC $D0**
**BNE $0162**
**INC $D1**

Der Adreßzeiger wird um Eins erhöht. Wenn beim Erhöhen das Low-Byte den Wert Null annimmt, muß auch das High-Byte erhöht werden. Im anderen Fall wird der Befehl zum Erhöhen des High-Bytes übersprungen.

**DEX**
**BNE $0154**

Der Zähler und Index X wird um Eins verringert. Ist X anschließend noch nicht Null, erfolgt der Sprung zum Laden des nächsten Bytes mit LDA ($D0),Y. Wenn durch DEX das X-Register Null wird, so wird das Zero-Flag gesetzt, auf das der BNE-Befehl reagiert. Hätten wir X dagegen in umgekehrter Richtung von eins nach acht laufen lassen, wäre das nicht der Fall gewesen. Nach INX hätte erst ein zusätzlicher Befehl CPX #$09 eingeschoben werden müssen, den wir uns so erspart haben.

Statt X am Anfang mit LDX #$08 zu initialisieren, hätten wir genausogut LDX #$07 verwenden können. Das Speichern wäre dann eben mit STA $D3,X vonstatten gegangen und die Schleifenendabfrage nach DEX mit BPL.

**RTS**

## GESONDERTE LESEROUTINE

Die Routine zum Lesen der Bytes aus dem Grafikbereich wurde als Unterprogramm angelegt, damit sie gegebenenfalls von BASIC aus gesondert angesprungen werden kann, um zu sehen, ob etwas auf den Drucker ausgegeben werden müßte. Wir kommen jetzt zur gesamten Routine, mit Matrixkonvertierung und Druckerausgabe.

**JSR $014E**

Am Anfang steht der Aufruf der Unterroutine zum Lesen der acht Bytes aus dem Grafikbereich.

**LDY#$08**
**LDX#$08**

An Stelle der Schleifenvariablen J und K nehmen wir in Maschinensprache die Register X und Y.

**ASL $D2,X**
**ROL**

Mit dem ASL-Befehl wird das höchstwertige Bit aus der durch X indizierten Zeile herausgeschoben. Genausogut hätte das auch ein ROL-Befehl getan. Der folgende ROL-Befehl schiebt dieses Bit von rechts nach links in den Akku. Bei Druckern, die ihr Byte andersherum brauchen, muß lediglich ROL gegen ROR ausgetauscht werden.

**DEX**
**BNE $013F**

Solange X noch nicht Null ist, wird auf den Befehl ASL $D2,X zurückgesprungen. Erst wenn

alle acht höchstwertigen Bits erfaßt sind, wird die Schleife verlassen.

```
NOP
NOP
```

Diese zwei NOP-Befehle bewirken nichts. Sie reservieren nur Platz. Commodore-Drucker brauchen die Bits in umgekehrter Reihenfolge, was ja leicht zu ändern ist. Außerdem drucken sie Grafik nicht mit acht, sondern nur mit sieben Nadeln. Bit sieben, das höchstwertige Bit, ist hierbei auf Eins zu setzen. Damit die Operation ORA #$80 in dieses Programm eingebaut werden kann, wurde der hierzu benötigte Platz mit Hilfe der beiden NOP-Befehle geschaffen.

```
JSR FFD2
```

Nachdem das Byte im Akku bereitsteht, kann es an den Drucker ausgegeben werden. Die Betriebssystemroutine BSOUT, die durch JSR $FFD2 angesprochen wird, gibt den Akkuinhalt auf das gewählte Ausgabegerät aus.

```
DEY
BNE $013D
RTS
```

Acht Bytes insgesamt sind auszugeben. Die Kontrolle darüber obliegt dem Y-Register. Ist dieses nach dem Erniedrigen noch nicht Null, erfolgt der Sprung wieder auf LDX #$08 zum Ausgeben des nächsten Bytes. Ansonsten hat das Maschinenprogramm ein vollständiges Zeichen an den Drukker gesandt und damit seinen Dienst beendet. Der Rücksprung in das aufrufende Programm, das in unserem Fall ein BASIC-Programm sein wird, erfolgt.

Listing Nr. 3 gibt das soeben besprochene Maschinenprogramm wieder. Darin lassen sich auch die Sprungadressen verfolgen. Damit nicht der Maschinenteil und der noch zu erstellende BASIC-Teil getrennt geladen werden müssen, empfiehlt sich die Wandlung in DATA-Zeilen mit dem DATA-WANDLER. Wir haben dies bereits in Form von Listing Nr. 4 für Sie erledigt.

## HARDCOPYROUTINE

Da ein Großteil der Anwender wohl einen COMMODORE-Drucker oder weitgehendst Kompatiblen ihr eigen nennen werden, soll als erstes die Hardcopy mit einem grafikfähigen Commodore-Drucker besprochen werden. Hierfür ist das Maschinenprogramm in einigen wenigen Punkten zu modifizieren, was mit POKE-Befehlen bewerkstelligt werden kann.

```
POKE 321,106
```

Damit wird der ROL-Befehl an Speicherstelle $0141 durch den Code für den ROR-Befehl überschrieben. Der Commodore-Drucker steuert im Grafikbetrieb nur sieben Nadeln an. Mit unserer einfachen Hardcopy werden wir wohl oder übe auf die achte Linie jedes Zeichens verzichten müssen. Zu ihrem Druck bedarf es eines aufwendigereren Programmes, das bei der nächsten zu drukkenden Zeile das oder die übrig gebliebenen Bytes der vorhergehenden Bildschirmzeile mit übernimmt. Dafür gibt es bereits HIRES-HARDCOPY aus dem C16/P4-SPECIAL Nr. 1. Wir dürfen beim Commodore-Drucker nicht vergessen, Bit sieben des an den Drucker auszugebenden Codes zu setzen.

```
POKE 325,9:POKE 326,
128
```

Der Operationscode für ORA #$ entspricht dem Zahlenwert 9. Mit diesem Wert ist der NOP-Befehl an Adresse $0145 zu

```
10 rem listing 9======(hardcopies) <nb>
20 rem by alfons mittelmeyer       <if>
30 rem ============================ <ng>
40 rem fuer druckdichte von        <eo>
50 rem 240 dots/zoll.              <he>
60 rem maschinensprachteil         <pj>
70 rem ============================ <po>
80 fori=312to366                   <ih>
90 reada:pokei,a:next              <dk>
100 data 032,078,001,160,008,162   <mm>
110 data 008,022,210,042,202,208   <gd>
120 data 250,234,234,032,102,001   <kd>
130 data 136,208,240,096,160,000   <ej>
140 data 133,210,162,008,177,208   <dg>
150 data 149,210,005,210,133,210   <im>
160 data 230,208,208,002,230,209   <jg>
170 data 202,208,239,096,032,210   <im>
180 data 255,032,210,255,076,210   <ae>
190 data 255                       <gi>
200 rem ============================ <ne>
210 rem  basic-teil ist anzufuegen <ac>
220 rem ============================ <hm>
```

```
10 rem centronic==========plus4 <bb>
20 rem (p) commodore welt team   <ho>
30 rem ========================== <ng>
40 rem (c) by alfons mittelmeyer <cg>
50 rem                           <pd>
60 rem besonders geeignet fuer   <bd>
70 rem grafikprogramme, da keine <bd>
80 rem codewandlung stattfindet  <ni>
90 rem ========================== <jg>
100 fori=1015 to 1068            <dg>
110 reada:pokei,a:next           <ne>
120 data 72,165,153,201,3,208,43 <pp>
130 data 165,173,201,116,208,37  <fh>
140 data 104,76,19,4,72,32,247,3 <in>
150 data 104,96,10,32,19,4,104,141 <pb>
160 data 16,253,169,0,141,2,253  <gh>
170 data 169,8,141,2,253,169,32,44 <pp>
180 data 1,253,240,251,24,96,104 <ne>
190 data 76,75,236,166,174,224,4 <in>
200 data 208,8,162,3,134,174,162 <on>
210 data 116,134,173,76,166,6,166 <nk>
220 data 173,224,116,208,4,162,0 <fc>
230 data 134,173,32,12,239,162,0 <gl>
240 data 134,173,96,32,83,239,76 <oj>
250 data 161,6,32,93,238,76,161,6 <pk>
260 fori=1667 to 1713            <mc>
270 reada:pokei,a:next           <jj>
280 poke792,131:poke793,6        <gh>
290 poke800,148:poke801,6        <cn>
300 poke804,008:poke805,4        <da>
310 poke794,172:poke795,6        <lo>
320 rem =========e=n=d=e========== <cf>
```

```
10 rem centronics==============c16 <hl>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by alfons mittelmeyer   <cg>
50 rem nur in verbindung mit dem   <hj>
60 rem centronics-interface aus    <oa>
70 rem c16-p4-spezial nr.1.        <pf>
80 rem besonders geeignet fuer     <bk>
90 rem grafik-programme            <ak>
100 rem ========================== <id>
110 fori=1026to1067                <pf>
120 reada:pokei,a:next             <ag>
130 data 072,133,245,165,001,162   <dd>
140 data 008,009,003,006,245,144   <kp>
150 data 002,041,253,133,001,041   <hi>
160 data 254,133,001,202,208,239   <np>
170 data 009,004,133,001,041,249   <ml>
180 data 133,001,036,001,080,252   <np>
190 data 024,174,001,004,104,096   <el>
200 fori=1659to1729                <bh>
210 reada:pokei,a:next             <ap>
220 data 072,165,153,201,003,208   <gh>
230 data 060,165,173,201,116,208   <bd>
240 data 054,104,142,001,004,076   <al>
250 data 002,004,166,174,224,004   <oo>
260 data 208,008,162,003,134,174   <ll>
270 data 162,116,134,173,076,178   <ig>
280 data 006,166,173,224,116,208   <of>
290 data 004,162,000,134,173,032   <gl>
300 data 012,239,162,000,134,173   <lj>
310 data 096,032,083,239,076,173   <af>
320 data 006,032,093,238,076,173   <nl>
330 data 006,104,076,075,236       <go>
340 fori=818to858                  <ai>
350 reada:pokei,a:next             <di>
360 data 169,160,141,032,003,169   <ni>
370 data 006,141,033,003,169,143   <dd>
380 data 141,024,003,169,006,141   <ef>
390 data 025,003,169,184,141,026   <ld>
400 data 003,169,006,141,027,003   <ag>
410 data 169,123,141,036,003,169   <aa>
420 data 006,141,037,003,096       <jn>
430 sys818:new                     <lc>
440 rem ========================== <kf>
450 rem  p r o g r a m m e n d e   <pc>
460 rem ========================== <jk>
```

überschreiben. Das Byte im Akku ist mit dem Wert $80 zu odern, damit Bit sieben gesetzt wird. Erreicht wird dies durch POKEn des Wertes 128, der die dezimale Entsprechung zu $80 ist, in die nachfolgende Speicherzelle $0146 (dezimal 326). Das Maschinenprogramm ist jetzt an den Commodore-Drukker angepaßt.

## INITIALISIERUNG DES ZEIGERS

Das Maschinenprogramm verwendet den Inhalt der Adressen $D0 und $D1 als Zeiger zum Lesen des jeweils nächsten Zeichens. Vor dem Start der Hardcopy ist der Zeiger auf den Anfang des Grafikbereiches $2000 einzustellen.

**POKE 208,0:POKE 209, 32**

Dezimal ergibt $D0 den Wert 208. Der in die folgende Speicherstelle zu POKEnde Wert $20 ergibt dezimal 32.

## ÖFFNEN DES DRUCKERKANALS

Der Drucker muß mit dem OPEN-Befehl als Datei angemeldet sein. Damit er als Ausgabegerät auch für die BSOUT-Routine des Maschinenprogramms bereitsteht, ist der Druckerkanal mit dem CMD-Befehl zu öffnen.

**OPEN 4,4:CMD4, CHR$(8);**

Folgt auf den CMD-Befehl ein Komma, so wird der Rest danach genau wie bei einem PRINT-Befehl ausgegeben. Mit CHR$(8) stellen wir den Drucker auf Grafik um. Codes, die größer oder gleich 128 sind, Bytes mit gesetztem siebten Bit also, steuern jetzt direkt die obersten sieben Nadeln an.
Gleichzeitig erfolgt der Zeilenvorschub in kürzeren Abständen, so daß die nächste Druckzeile ohne Zwischenraum an die vorhergehende anschließt.

## AUSGABE AN DEN DRUCKER

Die Ausgabe gestaltet sich recht einfach. Es sind 25 Zeilen zu jeweils 40 Zeichen an den Drucker zu senden.

```
FORJ=1TO25:PRINT
FORI=1TO40:SYS312:
NEXT:NEXTJ
```

Vor jeder Zeile bewirken wir mit PRINT einen Zeilenvorschub. Mit SYS 312 wird das Maschinenprogramm an Adresse $0138 (dezimal 312) aufgerufen, das genau ein Zeichen ausgibt.

**PRINT#4,CHR$(15) :CLOSE4**

Mit PRINT#4,CHR$(15) schalten wir den Drucker wieder auf Text um, geben einen zusätzlichen Wagenrücklauf und Zeilenvorschub aus, da kein Strichpunkt folgt, und schließen den Druckerkanal. Die CLOSE-Anweisung beseitigt den Eintrag des logischen Files.
Listing Nr. 5 enthält den BASIC-Teil der Hardcopy für grafikfähige Commodore-Drucker. Er ist an den in DATA-Zeilen gewandelten Maschinenteil anzubinden.

IBM- und Epson-Drucker können mit acht Nadeln drucken. Es braucht daher keine Linie unterschlagen zu werden. Da für den Acht-Nadel-Drucker jedes Bit des zu sendenden Bytes zur Nadelansteuerung dient, ergibt sich die Frage, wie wieder in den Normalmodus umgeschaltet werden kann. Durch einen Steuercode wohl kaum, denn jeder Code wird als Bitmuster verstanden.
So wird hier ein etwas anderer Weg eingeschlagen. Den Steuercodes zum Umschalten auf Grafik folgen zwei Bytes, die angeben, wieviel Daten-Bytes darauf im Grafikmodus ausgegeben werden sollen. Nach Ausgabe dieser Anzahl kann der Drucker wieder ASCII-Codes und Steuercodes interpretieren. Mehr Flexibilität erlauben IBM und Epson, da der Zeilenabstand frei wählbar ist.

```
POKE 208,0:POKE 209,
32
E$=CHR$(27):X$=
CHR$(13)+CHR$(10)
```

Der Adreßzeiger muß wieder auf den Grafikanfang gesetzt werden. Das Programm wird kürzer und übersichtlicher,

wenn mehrfach auftretende Steuercodes als Stringvariablen definiert werden. Ist der automatische Zeilenvorschub bei Wagenrücklauf eingestellt, so darf X$ nicht zusätzlich den Steuercode für Linefeed enthalten. X$=CHR$(13) darf es in diesem Fall nur heißen.

```
OPEN4,4:CMD4,E$"A"
CHR$(8);
```

Mit Escape und A wird der Zeilenabstand gewählt. Das folgende Byte n gibt diesen in n/72 Zoll wieder. Für n können Werte von Null bis 85 gewählt werden. Da der Abstand der Nadeln voneinander 1/72 Zoll beträgt und mit acht Nadeln gedruckt wird, wählten wir den dafür richtigen Zeilenabstand 8/72 Zoll.

```
PRINTE$"2";
```

Dieser Steuercode ist nur für den IBM-Drucker erforderlich. Der vorher angewählte Zeilenabstand wird erst durch diesen Befehl aktiviert, beim Epson-Drucker hingegen sofort.
ESC und "2" würden den Zeilenabstand beim Epson-Drucker auf 1/6 Zoll stellen, was wir hier nicht brauchen können.

```
FORJ=1TO25
PRINTX$E$"K"CHR$
(64)CHR$(1);
```

Mit X$ wird der Zeilenvorschub realisiert. Danach ist dem Drucker zu melden, daß er in Grafik mit einer bestimmten Dichte drucken soll. Vier verschiedene Dichten stehen für den IBM-Drucker zur Auwahl. Der Epson-Drucker kennt darüber hinaus noch fünf weitere. Die Druckpunkte, die der Drucker zu Papier bringt, heißen nicht Pixel wie beim Bildschirm, sondern Dots. Nachstehend finden Sie die Anwahl der vier Dichten, die sowohl IBM- als auch Epson-Drucker kennen.

```
ESC K =  60 Dots/Zoll
ESC L = 120 Dots/Zoll
ESC Y = 120 Dots/Zoll
ESC Z = 240 Dots/Zoll
```

Mit ESC Y erfolgt der Ausdruck in doppelter Geschwindigkeit als mit ESC K. Der Nachteil hierbei ist, daß direkt nebeneinander liegende Druckpunkte unterschlagen werden. Von zwei solchen Punkten, die schwarz zu Papier kommen sollten, bleibt einer also weiß.
Die Anzahl der horizontal zu druckenden Punkte pro Zeile wird im Anschluß an die Dichteauswahl mit Hilfe zweier Bytes angegeben. Low-Byte 64 und High-Byte eins sind 64+256=320, was 40 Zeichen mit je acht Pixeln entspricht.

```
FORI=1TO40:SYS312
:NEXTI:NEXTJ
PRINTE$"A"CHR$(12);
```

Das Ende des Programms vollzieht sich analog zur Hardcopy mit dem Commodore-Drucker. Mit ESC A und der Größenangabe stellen wir beim IBM-Drucker 12/72 = 1/6 Zoll Zeilenabstand ein, was der Standard für Textdruck ist. Beim Epson-Drucker kann entweder dieser Befehl oder der folgende entfallen.

```
PRINTE$"2";
```

Dieser Befehl aktiviert den eingestellten Zeilenabstand beim IBM-Drucker. Beim Epson-Drucker stellt er immer den Standardabstand von 1/6 Zoll ein.

```
PRINT#4,X$;:CLOSE4
```

Wir schließen nach dem Senden von Wagenrücklauf und Zeilenvorschub den Druckerkanal. Mit CLOSE wird der Eintrag in die File-Tabelle beseitigt.
Zusammengefaßt finden Sie die Programme für IBM- und Epson-Drucker als Listing Nr. 6 und Nr. 7 im Anhang.

Eine Hardcopy für nicht grafikfähige Drucker, das gibt es doch nicht, wird manch einer sagen. Wir haben uns ein Verfahren überlegt, das sowohl auf nicht grafikfähigen Com-

## UNIVERSAL-HARDCOPY AUCH FÜR NICHT GRAFIKFÄHIGE DRUCKER

modore-Druckern funktioniert als auch mit den vielen nur teilweise Commodore-kompatiblen Druckern zusammenarbeitet, die vielleicht den Textmodus von Commodore noch richtig emulieren, deren Grafik aber anders anzusteuern ist. Normalerweise wird es Epson- oder IBM-Grafik sein, kann aber auch andersartig angesteuert werden.

```
POKE 208,0:POKE209,
32
OPEN5,4,5:OPEN4,4:
PRINT#4,CHR$(8);
```

Die letzte Zeile bringt an den Tag, was wir vorhaben. Mit Sekundäradresse fünf läßt sich ein Zeichen, das sogenannte Geschäftszeichen, vom Benutzer frei definieren. Mit dem Code CHR$ (254) kann dieses Geschäftszeichen dann unter anderer Sekundäradresse zu Papier gebracht werden. Wenn ein nicht grafikfähiger Drukker allerdings mit CHR$ (8) nicht einmal einen kürzeren Zeilenabstand einstellen kann, sieht die Hardcopy ein wenig gestückelt aus, da weiß gebliebene Linien den Ausdruck zerteilen.
Ein Problem gilt es noch zu lösen. Die Definition des Geschäftszeichens darf nur am Zeilenanfang erfolgen und gilt dann für die ganze Zeile. Im Druckerhandbuch steht, daß verschiedene Geschäftszeichen nur durch mehrfaches Überdrucken derselben Zeile möglich seien.

```
FORJ=1TO25:PRINT#4
FORI=1TO40:SYS334
IFPEEK(210)=0THEN...
```

Mit SYS334 werden die Daten nur gelesen, und es wird erfaßt, ob überhaupt etwas zu drucken wäre. Ist dies nicht der Fall, was mit dem PEEK-Befehl festzustellen ist, so überspringen wir die eigentliche Druckerroutine. Nötig wäre dieses Vorgehen nicht, doch ersparen wir dadurch dem Drucker Arbeit. Grafiken, die mehrere Leerzeilen und viele leere Zeichen enthalten, kommen so schneller auf das Papier.

```
PRINT#4,CHR$(141);:
CMD5,:SYS315
```

Mit CHR$(141) erfolgt ein Wagenrücklauf ohne Zeilenvorschub. Nachdem der Druckkopf sich am Zeilenanfang befindet, kann auf Sekundäradresse fünf umgeschaltet und mit SYS315 das Zeichenmuster gesendet werden. Das Komma und der Strichpunkt bei der CMD-Anweisung sind wichtig, damit der Drukker nicht als erstes Bitmuster den Return-Code erhält. Das letzte Bitmuster würde den Drucker außerdem in Verwirrung bringen.

```
PRINT#5:A$=RIGHT$
("0"+STR$(I-1),2)
PRINT#4,CHR$(16)A$
CHR$(254);
NEXT:NEXTJ
```

Mit PRINT#5 wird die Zeichendefinition beendet. Die Tabulatorfunktion CHR$(16) verwenden wir, um den Druckkopf auf die richtige Stelle in der Zeile zu fahren.

Die Position muß im Anschluß durch eine zweistellige ASCII-Zahl angegeben werden. Die Berechnung erfolgt durch die STR$-Funktion. Ist das Ergebnis nur einstellig, so setzen wir die Ziffer Null davor. Die RIGHT$-Funktion sorgt in jedem Fall für die richtige Län-

ge. Nach der Positionsangabe A$ brauchen wir nur noch den Code CHR$ (254) zu senden, um das definierte Zeichen zu Papier zu bringen.
Bei der Definition des Zeichens fährt der Druckkopf nicht bis zum Anschlag zurück, da der Drucker so intelligent ist, zu warten, ob nicht etwa wieder eine gegenläufige Bewegung erfolgen soll. Dennoch ruckelt der Druckkopf bei den meisten Druckern ganz schön hin und her. Aus diesem Grunde hatten wir auch für eine teilweise Entlastung gesorgt.

```
PRINT#4,CHR$(15):
CLOSE4:CLOSE5
```

Das Beenden der Routine erfolgt wie gewohnt. Das Programm finden Sie als Listing Nr. 8 im Anhang.

## EXOTENDRUCKER MCS801

Obwohl der Farbdrucker MCS801 auch ein Commodore-Drucker ist, ist er, was Grafik angeht, keineswegs kompatibel.

## GESCHÄFTSZEICHEN-MODUS

Das Universalprogramm für nicht grafikfähige Drucker sollte doch eigentlich auch für den MCS801 Gültigkeit haben, da hier der Geschäftszeichenmodus genutzt wird. Doch ist dies eine Täuschung. Die Zeichen werden umgedreht wiedergegeben. Was zu tun ist, wissen wir aber bereits. Mit POKE 321, 106 sorgen wir für die richtige Wiedergabe.

## GRAFIKMODUS

Beim Drucken im Grafikmodus ist zu beachten, daß dieser ähnlich wie bei Epson oder IBM angesteuert wird. Jedoch ist die Befehlssequenz eine andere. Die Anzahl der zu druckenden Punktreihen pro Zeile ist nicht im Byte-Format anzugeben, sondern als dreistellige ASCII-Zahl. Damit würde die Sequenz am Anfang einer Druckzeile lauten:

```
PRINTX$E$"K320";
```

Epson- und IBM-Drucker erfordern dagegen:

```
PRINTX$E$"K"CHR$
(64)CHR$(1);
```

Einfacher und einprägsamer ist für den Anwender sicherlich die Ansteuerung des MCS-801. Auf einen Blick ist zu sehen, daß horizontal 320 Punkte gedruckt werden sollen. Zum Nachschlagen, wie der richtige Zeilenabstand von 8/72 Zoll eingestellt wird, verwenden Sie bitte Ihr Druckerhandbuch.

## WARUM DER KREIS ZUR ELLIPSE WIRD

Auf das richtige Verhältnis von Höhe und Breite kommt es an. Wenn es nicht stimmt, ist nicht der Computer daran schuld, sondern der Drukker. Genauer gesagt, die Software, die den Drukker ansteuert. Meist läßt sich für die richtige Wiedergabe etwas tun.
Wählen wir die normale Grafikdichte von 60 Dots/Zoll, so wird der Ausdruck zu breit. Der vertikale Nadelabstand beträgt nämlich 1/72 Zoll, was einer Dichte von 72 Dots/Zoll entspricht. Beim Commodore-Drukker im Grafikmodus gilt somit das Verhältnis:

```
Breite/Höhe=1.2
```

In der in diesem Artikel vorgestellten Hardcopy, bei der wir eine Linie pro Zeile wegließen, ist es gar noch schlechter:

```
8/60*72/7=1.37
```

Auch für den Commodore-Drucker gibt es die Möglichkeit, Breite und Höhe im Verhältnis eins zu eins darzustellen.
Wenn Sie die Hardcopy für den Geschäftszeichenmodus ausprobiert haben, so sollten Sie es bereits beobachtet haben. Im normalen Schriftmodus gibt der Drucker zehn Zeichen pro Zoll aus. Ist nicht NLQ gewählt, wird das Zeichen im Normalfall mit einer Dichte von 60 Dots/Zoll gedruckt, so daß in der Breite nur sechs Punkte dafür zur Verfügung stehen. Bei einigen Druckern wie dem MCS-801 kann im Schriftmodus auch eine größere Dichte vorliegen, weswegen vielleicht bei ihnen die Grafik-Hardcopy bereits richtig kommen könnte.
Dieselbe Dichte bekommen wir beim normalen Drucker mit dem definierbaren Geschäftszeichen. Dieses Zeichen ist ebenfalls ein zehntel Zoll breit, besteht jedoch aus acht horizontalen Punkten. Die Druckdichte beträgt somit 80 Dots pro Zoll. Bei 7/72 Zoll Zeilenabstand wird die unterste Linie einer Zeile nochmals mit der obersten der nächsten Zeile überdruckt. Unser Zeichen ist also 8/80 Zoll breit und 7/72 Zoll hoch.

```
8/80*72/7=1.03
```

Der Dehnungsfaktor von 1.03 fällt überhaupt nicht ins Auge. Damit ist für Commodore-Drukker alles klar.

## EPSON-DRUCKER BESTENS AUSGESTATTET

Keinerlei Klimmzüge braucht es beim Epson-Drucker. Speziell für Hardcopies und Bildschirmplots hat dieser auch eine Druckdichte von 72 Dots/Zoll, wodurch vertikaler und horizontaler Dot-Abstand genau gleich sind. Die Befehlssequenz, um weitere Druckdichten als es der IBM-Drucker erlaubt, einzustellen, lautet:

```
ESC * m n1 n2
```

Der Wert von m legt dabei die Druckdichte fest.

| m | Dichte |
|---|---|
| 0 | 60 Dots/Zoll |
| 1 | 120 Dots/Zoll |
| 2 | 120 Dots/Zoll |
| 3 | 240 Dots/Zoll |
| 4 | 80 Dots/Zoll |
| 5 | 72 Dots/Zoll |
| 6 | 90 Dots/Zoll |

Die ersten vier Dichten von m=0 bis m=3 entsprechen den bereits bekannten, die auch durch ESC K, ESC L, ESC Y und ESC Z erzeugt werden können. Die restlichen Dichten sind für Grafik-Hardcopies mit verschiedenen Rechnern vorgesehen. Einige Beispiele:

```
Commodore
          72 Dots/Zoll
Epson-QX-10
          80 Dots/Zoll
DEC-
Computer90 Dots/Zoll
```

Auf m als Dichteparameter muß auch hier wieder die Anzahl der zu druckenden Datenbytes folgen. Wir ändern im Epson-Listing daher:

```
PRINTX$E$"*"chr$(5)
chr$(64)chr$(1);
```

## TRICK FÜR IBM-DRUCKER

Über die Druckdichte 72 Dots/Zoll verfügt der IBM-Drucker nicht. Wenn wir 80 Dots/Zoll erreichen könnten, könnten wir dasselbe Verfahren anwenden wie bereits bei den Commodore-Drukkern. Ein Trick hilft hier weiter: Wir stellen die Druckdichte auf 240 Dots/Zoll und geben jedes Datenbyte dreimal hintereinander aus. Den Zeilenabstand stellen wir außerdem auf 7/72 Zoll.

```
OPEN4,4:CMD4,ES"A"
CHRS(7);
PRINTXSES"Z"CHRS
(192)CHRS(3);
```

Welche Zeilen im IBM-Listing zu ersetzen sind,

dürfte unschwer zu erkennen sein. Die dreimalige Ausgabe jedes Datenbyte wird am besten im Maschinenprogramm vorgenommen. Der Ausgabebefehl BSOUT, JSR $FFD2 wird einfach durch einen Sprung in eine Unterroutine ersetzt, in welcher steht:

```
JSR $FFD2
JSR $FFD2
JMP $FFD2
```

Diese neue Maschinenroutine finden Sie als Listing Nr. 9 im Anhang.

### KLEINER, GRÖSSER, DICHTER

Die Punkte, die bei der vorangegangenen Hardcopy dicht beieinanderliegen, lassen die Zwischenräume fast verschwinden. Wenn der vertikale Nadelabstand ebenfalls verringert würde, so würden schwarze Flächen ihren Punktchencharakter verlieren und beinahe als gefüllte Flächen wirken. Eine solche Hardcopy sieht wesentlich besser aus.
Durch mehrfaches Drukken von Datenbytes und Zeilen lassen sich vergrößerte Hardcopies erzielen. Größere Dichte und Zeilenabstände von 1/216 Zoll machen Verkleinerungen möglich. Mehr darüber gibt es im nächsten SPECIAL, im zweiten und letzten Teil dieser Folge.
Probleme können bei Centronics-Druckern mit Commodore-Interface auftreten. Ein gutes Interface läßt die Umschaltung in den Epson- oder IBM-Modus zu. Ein schlechtes arbeitet bei Grafik vielleicht fehlerhaft oder erlaubt nur den Commodore-Modus. Wer alle Möglichkeiten seines Druckers ausschöpfen möchte, sei nochmals auf das Centronics-Kabel für den Plus/4 und an das Centronics-Interface für den C16 hingewiesen, worüber wir in der COMMODORE WELT Nr. 12/88 und im C16-P4-SPECIAL Nr. 1 berichteten. Das Centronics-Kabel gibt es mittlerweile bei Rex-Datentechnik, die Bezugsadresse für das Interface können Sie bei uns erfragen. Da die Treibersoftware für den Textmodus gedacht und mit ASCII-Wandlung, automatischem Zeilenvorschub bei Return oder Unterdrückung des ASCII-Codes 17 ausgestattet war, finden Sie im Anhang die Centronics-Software für den Grafikmodus.

Auf einen Fehler in der Beschreibung des Interface für den C16 müssen wir die Selbstbauer hinweisen. Wir verwechselten die Leitungen Strobe und Busy. Tauschen Sie daher am Druckerstecker die Leitung an Pin eins gegen die an Pin 11 aus. Pin drei des seriellen Ports muß an Pin eins des Druckerports und Pin elf des Druckerports an den Eingang des Inverters gelegt werden. Am günstigsten ist, zuerst das Programm mit der Grafik aufzurufen und es an der gewünschten Stelle zu unterbrechen. Dann ist die Centronics-Software zu laden und zu starten, danach erst die Hardcopyroutine. Vorher drucken ist ja nicht möglich.
Weiter geht es im nächsten Heft mit kleineren, größeren und dichteren Hardcopies. *a.m.* □



DIAGRAMM

# Optik für Ihre Zahlen

**Übersichtlich und anschaulich präsentieren sich sonst nichtssagende Zahlen, wenn sie in Form von Balken-, Kreis- und X-Y-Diagrammen für das Auge aufbereitet werden. Keine besonderen Anstrengungen sind für die grafische Ausarbeitung vonnöten, sondern nur das vorliegende Programm.**

Das Programm Diagramm 16 bietet die Möglichkeit, Zahlenangaben (zum Beispiel Statistiken) in anschaulichen und übersichtlichen Grafiken darzustellen.
Prinzipiell stehen drei verschiedene Diagramme zur Auswahl, deren Verwendung durch Art und Umfang der darzustellenden Werte bestimmt wird.
Nach dem Start des Programmes wird zunächst eine Maschinenroutine zum Laden und Speichern von Hires-Bildern eingelesen und im Kassettenpuffer abgelegt.
Danach gelangt man in das Hauptmenü mit den Punkten:

1. Demos
2. Balkendiagramm
3. Kreisdiagramm
4. x-y-Diagramm
5. Laden/Speichern
6. Programmende

### 1. Demos

Man gelangt in ein Untermenü, aus dem heraus zu jedem der drei möglichen Diagramme eine Demo abgerufen werden kann. Hierdurch gewinnt man am schnellsten einen Überblick über die Einsatzmöglichkeiten der Grafiken. Durch Druck einer Taste wird anschließend wieder das Hauptmenü angesprungen.

### 2. Balkendiagramm

Nach einer kurzen Einleitung werden folgende Eingaben verlangt:

- Überschrift der Grafik (maximal 40 Zeichen lang);
- Maßeinheit der Zahlenwerte;
- Anzahl der darzustellenden Balken (1 bis 6);
- Zahlenwerte (1 bis 6) und deren Bezeichnung.

Anschließend wird die Grafik aufgebaut. Die Balken werden räumlich umrissen und farblich unterschieden. Überschrift, Maßeinheit, Zahlenwerte und Bezeichnung werden durch CHAR-Befehle plaziert. Durch Druck einer Taste gelangt man wieder in das Hauptmenü.

### 3. Kreisdiagramm

Das Kreisdiagramm stellt Prozentzahlen als Segmente eines aufgesprengten Vollkreises dar.
Auf eine kurze Einleitung folgt die Eingabe der unter 2. beschriebenen Punkte, wobei die Angabe der Maßeinheit entfällt. Es können bis zu sechs Prozentzah-

len und deren Bezeichnungen eingegeben werden. Die Summe der Prozentangaben darf 100 Prozent nicht überschreiten, sonst muß noch einmal eingegeben werden. Angaben kleiner als 5 Prozent sollten vermieden werden, da sonst Fehler in der Grafik auftreten können. Nach Abschluß der Eingaben wird das Bild erstellt. Die Segmente werden umrissen und ausgefüllt. Die Prozentangaben erscheinen neben dem jeweiligen Kreisausschnitt, die Bezeichnungen in den unteren Bildschirmzeilen. Auch hier wird nach dem Drükken einer Taste wieder in das Hauptmenü gesprungen.

### 4. x-y-Diagramm

Mit diesem Diagramm kann der Verlauf eines bestimmten Zahlenwertes über der Zeit (ein Jahr) aufgezeigt werden. Die x-Achse ist die Zeitachse. Sie umfaßt ein Jahr, unterteilt in die zwölf Monate. Die y-Achse kann mit einer beliebigen Maßeinheit belegt werden. Die einzelnen Zahlenwerte dürfen nicht mehr als fünf Stellen umfassen.
Nach einer kurzen Einleitung erfolgen die bereits beschriebenen Eingaben, wobei den einzelnen Werten hier keine Bezeichnung mehr zugeordnet wird.
Zunächst werden die Achsen gezeichnet und beschriftet. Die y-Achse ist immer in sechs Abschnitte unterteilt. Anhand des maximalen eingegebenen Zahlenwertes wird die Achsenbeschriftung berechnet, wobei geringfügige Rundungsfehler auftreten. Werte mit mehr als drei Stellen werden durch Division „verkürzt". Der entsprechende Faktor für die Maßeinheit wird mit angezeigt.
Anschließend wird die Kurve als Verbindungslinie von Wert zu Wert gezeichnet. Die Kurve wird zur Verdeutlichung in dreifacher Breite erstellt. Ein Tastendruck führt zurück in das Hauptmenü.

### 5. Laden/Speichern

Es wird das folgende Untermenü angesprungen:
1. Directory
2. Speichern
3. Laden
4. Hauptmenü

Punkt 1. zeigt lediglich den Inhalt der eingelegten Diskette, um die Suche nach bestimmten Grafiken zu erleichtern.
Mit Punkt 2. kann das Diagramm im Grafikspeicher auf Diskette abgespeichert werden. Dazu wird zunächst die vorhandene Grafik kurz eingeblendet. Es erfolgt die Abfrage, ob gespeichert werden soll. J führt zur Abfrage des Filenamens, alle anderen Tasten bewirken den Rücksprung in das Untermenü. Der Filename darf bis zu 15 Zeichen lang sein. Durch einen SYS-Befehl wird die Maschinenroutine zum Speichern aufgerufen. Es wird der Bereich von 1800 bis 3FFF (hex) gespeichert, damit auch die Helligkeits- und Farbinformationen erhalten bleiben. Nach Beendigung des Speichervorganges springt das Programm zurück in das Untermenü.
Mit Punkt 3. können auf diese Weise abgespeicherte Grafiken wieder eingeladen werden. Zuvor wird das Directory zur Auswahl des Filenamens angezeigt. Während des Ladevorganges ist der Bildschirm abgeschaltet. Wer den Aufbau der Grafik verfolgen will, muß die beiden POKE-Befehle in Zeile 2360 entfernen. Die eingelesene Grafik kann nun betrachtet werden. Der Rücksprung zum Untermenü erfolgt durch Druck einer Taste.
Puntk 4. bewirkt den Rücksprung in das Hauptmenü.

*Bitte lesen Sie weiter auf Seite 84*

```
10 rem diagramm================p4 <pl>
20 rem (p) commodore welt team  <ho>
30 rem ========================= <ng>
40 rem (c) by henning koglin    <ia>
50 rem                          <pd>
60 rem basic v3.5               <od>
70 rem plus4 (c16/116 + 64 kb)  <cd>
80 rem floppy                   <ak>
90 rem ========================= <jg>
100 gosub2390:rem maschinencode <nh>
110 rem ----------              <ip>
120 rem hauptmenue              <po>
130 rem ----------              <lj>
140 scnclr:clr:gosub2550:color0,1:
color4,1:color1,7,5             <no>
150 dimp(12):dimpr(12)          <en>
160 char1,11,0,rn$+"d i a g r a m
m e"+rf$                        <be>
170 char1,11,3,"1.demos........."
:char1,11,5,"2.balkendiagramm." <od>
180 char1,11,7,"3.kreisdiagramm.."
:char1,11,9,"4.x-y-diagramm..." <jn>
190 char1,11,11,"5.laden/speichern
":char1,11,13,"6.programmende..." <eh>
200 char1,7,17,"bitte waehlen sie
(1-6) !"                        <ei>
210 geta$:ifa$>"0"anda$<"7"then220
:else210                        <fk>
220 onval(a$)goto260,550,830,1250,
1960,1670                       <km>
230 rem ---------------         <hl>
240 rem demo-untermenue         <ik>
250 rem ---------------         <ff>
260 scnclr:char1,11,0,rn$+b3$+"d e
 m o s"+b4$+rf$:char1,15,2,b3$+"zu
"                               <fi>
270 char1,11,4,"1.balkendiagramm":
char1,11,6,"2.kreisdiagramm."   <ji>
280 char1,11,8,"3.x-y-diagramm..":
char1,7,15,"bitte waehlen sie (1-3
) !"                            <gp>
290 geta$:ifa$>"0"anda$<"4"then300
:else290                        <dk>
300 onval(a$)goto340,380,450    <dh>
310 rem ------------------      <mg>
320 rem demo balkendiagramm     <gn>
330 rem ------------------      <ob>
340 c=6:d$="ausgaben im monat deze
mber":me$="dm"                  <ho>
350 p$(1)="miete":p(1)=825.30:p$(2
)="strom":p(2)=70.00:p$(5)="auto":
p$(6)="hobby"                   <mc>
360 p$(3)="haushalt":p(3)=552.28:p
$(4)="kinder":p(4)=239.10:p(5)=83.
20:p(6)=120                     <hl>
370 goto690                     <pc>
380 rem ------------------      <kn>
390 rem demo kreisdiagramm      <ih>
```

```
400 rem ------------------          <fc>
410 c=6:d$="ausgaben im 1.halbjahr
 1988"                              <da>
420 p$(1)="kleidung":p(1)=6.8:p$(2
)="auto":p(2)=14.4:p$(3)="moebelka
uf":p(3)=21.5                       <pc>
430 p$(4)="hobby":p(4)=15.3:p$(5)=
"haushalt":p(5)=22.5:p$(6)="sonsti
ges":p(6)=19.5                      <gm>
440 goto990                         <gl>
450 rem ------------------          <cg>
460 rem demo x-y-diagramm           <gp>
470 rem ------------------          <kj>
480 c=12:d$="krankheitskosten 1988
":me$="dm"                          <hl>
490 p(1)=76:p(2)=54:p(3)=20:p(4)=2
1:p(5)=18:p(6)=5                    <hb>
500 p(7)=66:p(8)=19:p(9)=12:p(10)=
7:p(11)=19:p(12)=32                 <ne>
510 goto1430                        <hj>
520 rem --------------              <gg>
530 rem balkendiagramm              <da>
540 rem --------------              <al>
550 scnclr:printrn$"b a l k e n d
i a g r a m m"rf$                   <hp>
560 printc4$"das balkendiagramm st
ellt bis zu sechs"                  <im>
570 print"verschiedene zahlenwerte
 grafisch dar."                     <bp>
580 print"die eingabe kann in beli
ebiger massein- "                   <jb>
590 print"heit erfolgen."           <dj>
600 printc4$c4$c4$"bitte ueberschr
ift der grafik eingeben !"          <bj>
610 print"(nicht laenger als 40 ze
ichen) !":inputd$                   <pn>
620 iflen(d$)>40then550             <ki>
630 printc4$c4$"in welcher massein
heit soll dargestellt":print"werde
n";:inputme$                        <jm>
640 char1,0,18,"wieviele balken (1
-6)":inputc$                        <nk>
650 ifc$<"7"andc$>"0"then660:else6
40                                  <dp>
660 c=val(c$):scnclr                <ae>
670 fori=1toc:print"eingabe ";i;".
 wert"b3$b4$;:inputp(i)             <ld>
680 print"fuer (bezeichnung)"b3$b3
$;:inputp$(i):nexti                 <io>
690 gosub1720                       <ga>
700 graphic1,1                      <ci>
710 fori=1toc:color1,2+i,4          <am>
720 box1,i*48-48,168-pr(i),i*48-24
,168                                <mb>
730 draw1,i*48-24,168toi*48-12,160
toi*48-12,160-pr(i)toi*48-24,168-p
r(i)                                <co>
740 draw1,i*48-24,168-pr(i)toi*48-
12,160-pr(i)toi*48-36,160-pr(i)toi
*48-48,168-pr(i)                    <pc>
750 paint1,i*48-45,167,1            <bh>
760 nexti:color1,2:z=1              <mo>
770 fori=1toc:char1,(i-1)*6,5+z,st
r$(p(i)):char1,(i-1)*6,23+z,p$(i):
z=-z:nexti                          <co>
780 gosub1810:gosub1870             <bc>
790 getkeyw$:graphic0:goto140       <ci>
800 rem -------------               <cp>
810 rem kreisdiagramm               <hd>
820 rem -------------               <oc>
830 scnclr:char1,0,0,rn$+"k r e i
s d i a g r a m m "+rf$:print       <ia>
840 printc4$c4$"das kreisdiagramm
stellt bis zu sechs"                <bg>
850 print"prozentwerte als teile e
ines vollkreises"                   <li>
860 print"dar. die summe der werte
 darf 100% nicht"                   <df>
870 print"ueberschreiten !"         <ek>
880 print"eingaben unter 5% sollte
n vermieden":print"werden !!!"      <eh>
890 char1,0,10,"bitte ueberschrift
 der grafik eingeben !"             <lb>
900 char1,0,11,"(nicht laenger als
 40 zeichen) !":print:inputd$       <hf>
910 iflen(d$)>40then830             <dp>
920 char1,0,14,"wieviele prozentwe
rte (1-6)":inputc$                  <nb>
930 ifc$>"0"andc$<"7"then940:else9
20                                  <mi>
940 c=val(c$):scnclr                <ie>
950 fori=1toc:print"eingabe ";i;".
 prozentwert ";:inputp(i)           <ea>
960 print"fuer (bezeichnung)"b3$b4
$;:inputp$(i):nexti:su=0            <lf>
970 fori=1toc:su=su+p(i):nexti      <ag>
980 ifsu>100then940                 <pi>
990 fori=1toc:pr(i)=int((p(i)*3.6)
+.5):nexti                          <di>
1000 graphic1,1:color1,8,5          <al>
1010 r=60:su=0:x=160:y=100:locatex
,y                                  <ef>
1020 fori=1toc                      <ib>
1030 locate9,(su+pr(i)/2):xx=rdot(
0):yy=rdot(1):circle1,xx,yy,r,r,su
,su+pr(i)                           <ip>
1040 drawtoxx,yy:drawtor:su:su=su+
pr(i)                               <lm>
1050 locatex,y:locate30,(su-pr(i)/
2):paint                            <pj>
1060 locatex,y:nexti                <fl>
1070 color1,2:gosub1810             <ih>
1080 su=0                           <eh>
1090 fori=1toc:locatex,y:locater+2
5,(su+pr(i)/2)                      <di>
1100 xz=int(rdot(0)/8):yz=int(rdot
```

```
(1)/8):su=su+pr(i):ifyz<2thenyz=2  <pk>
1110 ifyz>21thenyz=21              <nc>
1120 ifyz<21andxz<19thenxz=xz-len(
str$(p(i)))-3                      <on>
1130 color1,3,6                    <eg>
1140 char1,xz,yz,str$(i)+":"+str$(
p(i))+"%":nexti                    <gh>
1150 color1,2:fori=0to39:char1,i,2
2,"C":nexti                        <co>
1160 color1,7,6                    <np>
1170 ifc>3thenfori=1to3:elsefori=1
toc                                <jk>
1180 char1,(i-1)*13,23,str$(i)+":"
+p$(i):nexti                       <dp>
1190 ifc>3thenfori=4toc:else1210   <kj>
1200 char1,(i-4)*13,24,str$(i)+":"
+p$(i):nexti                       <ef>
1210 getkeyw$:graphic0:goto140     <kh>
1220 rem ------------              <gb>
1230 rem x-y-diagramm              <ok>
1240 rem ------------              <oi>
1250 scnclr:char1,0,0,rn$+"x - y -
 d i a g r a m m "+rf$             <bo>
1260 print:printc4$c4$"das x-y-dia
gramm stellt bis zu zwoelf"        <ib>
1270 print"zahlenwerte dar. die y-
achse ist die"                     <ol>
1280 print"zeitachse, die in die z
woelf monate"                      <gp>
1290 print"unterteilt ist. "       <bg>
1300 print"die x-achse kann mit ei
ner beliebigen"                    <lm>
1310 print"masseinheit belegt werd
en. sie wird"                      <ej>
1320 print"grundsaetzlich in sechs
 abschnitte auf-"                  <on>
1330 print"geteilt, wobei der maxi
malwert zwischen"                  <co>
1340 print"0 und 99999 liegen kann
."                                 <fm>
1350 char1,0,14,"bitte ueberschrif
t der grafik eingeben !"           <nb>
1360 char1,0,15,"(nicht laenger al
s 40 zeichen) !":print:inputd$     <jh>
1370 iflen(d$)>40then1250          <eh>
1380 printc4$c4$"in welcher massei
nheit soll dargestellt":print"werd
en";:inputme$                      <jf>
1390 char1,0,23,"wieviele werte (2
-12)":inputc$                      <ki>
1400 ifval(c$)>1andval(c$)<13then1
410:else1390                       <ma>
1410 c=val(c$):scnclr              <jh>
1420 fori=1toc:print"eingabe ";i;"
. wert";:inputp(i):print:nexti     <ic>
1430 gosub1720                     <eh>
1440 z=1:ux=40:uy=160:restore1950  <ki>
1450 graphic1,1:locateux,uy:color1
,8,5:drawto320,uy:locateux,160:dra
wtoux,56                           <fn>
1460 fori=1to12:locate28+i*24,160:
drawto2;0to4;180                   <km>
1470 reada$:char1,((rdot(0)-24)/8)
+1,22+z,a$:z=-z:nexti              <ae>
1480 fori=1to6:locateux,uy-(i*16):
drawto2;270to4;90:nexti            <kp>
1490 pm=pm*100                     <bl>
1500 ifpm<1000andpm>0thenmu=1      <bp>
1510 ifpm<10000andpm>=1000thenmu=.
1:me$=me$+" * 10"                  <cc>
1520 ifpm>=10000thenmu=.01:me$=me$
+" * 100"                          <fc>
1530 ps=int(pm*mu/6)               <dk>
1540 fori=1to6:pa=(7-i)*ps:char1,0
,i*2+6,str$(pa)                    <hf>
1550 nexti                         <jo>
1560 color1,3,5                    <ck>
1570 gosub1600:r=1:gosub1640:gosub
1600:r=-2:gosub1640:gosub1600      <bh>
1580 color1,2:gosub1810:gosub1870  <pg>
1590 getkeyw$:graphic0:goto140     <no>
1600 locate52,160-pr(1)            <dk>
1610 fori=2toc                     <kb>
1620 xp=(28+i*24):drawtoxp,160-pr(
i):nexti                           <kn>
1630 return                        <fg>
1640 fori=1toc:pr(i)=pr(i)-r       <oa>
1650 ifpr(i)<0thenpr(i)=0          <ej>
1660 nexti:return                  <gd>
1670 rem ------------              <ck>
1680 rem programmende              <bd>
1690 rem ------------              <lc>
1700 scnclr:color1,8,7:char1,2,12,
"programm wirklich beenden (j/n) ?
?"                                 <gk>
1710 getkeyw$:ifw$="j"thensys65529
:else140                           <dn>
1720 rem ------------------------  <gg>
1730 rem maximalwert aussortieren  <pi>
1740 rem ------------------------  <ck>
1750 pm=0                          <ba>
1760 fori=1toc:ifp(i)>pmthenpm=p(i
)                                  <ma>
1770 nexti                         <ap>
1780 pm=pm/100                     <nb>
1790 fori=1toc:pr(i)=int((p(i)/pm)
+.5):nexti                         <pd>
1800 return                        <km>
1810 rem ------------------------
-                                  <io>
1820 rem ueberschrift positioniere
n                                  <of>
1830 rem ------------------------
-                                  <jl>
1840 d=int((40-len(d$))/2):char1,d
,0,d$                              <mf>
```

```
1850 fori=0to39:char1,i,1,"C":next
i                                  <ib>
1860 return                        <ce>
1870 rem ------------------------- <db>
1880 rem masseinheit positionieren <ld>
1890 rem ------------------------- <jm>
1900 ma$="angaben in "+me$:me=int(
(40-len(ma$))/2):char1,me,2,ma$    <ko>
1910 return                        <ii>
1920 rem -----                     <ee>
1930 rem datas                     <mj>
1940 rem -----                     <ep>
1950 data jan,feb,mae,apr,mai,jun,
jul,aug,sep,okt,nov,dez            <ag>
1960 rem -------------------------
-                                  <be>
1970 rem untermenue laden/speicher
n                                  <dc>
1980 rem -------------------------
-                                  <ni>
1990 scnclr:char1,12,0,rn$+"laden/
speichern"+rf$                     <do>
2000 char1,12,3,"1.directory....":
char1,12,5,"2.speichern...."       <ah>
2010 char1,12,7,"3.laden........":
char1,12,9,"4.hauptmenue..."       <dd>
2020 char1,7,17,"bitte waehlen sie
 (1-4) !"                          <kl>
2030 geta$:ifa$>"0"anda$<"5"then20
40:else2030                        <bf>
2040 onval(a$)goto2050,2090,2220,1
40                                 <ac>
2050 rem ---------                 <op>
2060 rem directory                 <kf>
2070 rem ---------                 <lm>
2080 scnclr:directory:printc4$"wei
ter mit beliebiger taste !":getkey
w$:goto1960                        <ji>
2090 rem ---------                 <ap>
2100 rem speichern                 <nj>
2110 rem ---------                 <cc>
2120 scnclr:print"zur zeit befinde
t sich die folgende":print"grafik
im speicher:"                      <cn>
2130 fori=1to2000:nexti            <ec>
2140 graphic1:fori=1to3000:nexti:g
raphic0                            <ok>
2150 printc4$c4$c4$"speichern (j/n
) ?";:getkeyw$:print" "+w$         <le>
2160 ifw$="j"then2170:else1960     <jk>
2170 char1,0,12,"name der grafik "
:inputna$                          <ln>
2180 iflen(na$)>15then2170         <jh>
2190 iflen(na$)<15thenfort=len(na$
)to15:na$=na$+" ":nextt            <jf>
2200 fori=1015to1030:pokei,asc(mid
$(na$,i-1014,1)):nexti             <bi>
2210 sys818:goto1960               <fo>
2220 rem -----                     <if>
2230 rem laden                     <db>
2240 rem -----                     <he>
2250 scnclr:print"mit dieser routi
ne koennen sie grafiken"           <eh>
2260 print"laden, die auf der eing
elegten diskette"                  <oc>
2270 print"gespeichert sind.":prin
t"zunaechst erscheint die director
y."                                <pm>
2280 print"anschliessend ist der n
ame der ge-":print"wuenschten graf
ik einzugeben."                    <pk>
2290 print"danach wird die grafik
eingelesen. sie"                   <pl>
2300 print"kann nun betrachtet wer
den."                              <im>
2310 print"durch druck einer taste
 gelangt man wie-"                 <gb>
2320 print"der in das untermenue '
laden/speichern'."                 <gl>
2330 char1,09,24,"bitte taste drue
cken !":getkeyw$                   <og>
2340 scnclr:directory:print:input"
name der grafik";na$               <mf>
2350 iflen(na$)>15then2340         <ok>
2360 iflen(na$)<15thenfort=len(na$
)to15:na$=na$+" ":nextt            <kf>
2370 fori=1015to1030:pokei,asc(mid
$(na$,i-1014,1)):nexti             <fn>
2380 graphic1:poke65286,11:sys854:
poke65286,27:getkeyw$:graphic0:got
o1960                              <ch>
2390 rem ------------------------- <kl>
2400 rem ms-unterprogramm einlesen <ni>
2410 rem ------------------------- <gl>
2420 restore2480                   <gl>
2430 fori=818to877                 <jo>
2440 readms$                       <ff>
2450 pokei,dec(ms$)                <ee>
2460 nexti                         <oe>
2470 return                        <on>
2480 data a9,01,a2,08,a0,08,20,ba,
ff,a9                              <bf>
2490 data 0f,a2,f7,a0,03,20,bd,ff,
a2,00                              <ek>
2500 data a0,18,86,d8,84,d9,a9,d8,
a2,ff                              <hm>
2510 data a0,3f,20,d8,ff,60        <ac>
2520 data a9,01,a2,08,a0,01,20,ba,
ff,a9                              <hf>
2530 data 0f,a2,f7,a0,03,20,bd,ff,
a9,00                              <hb>
2540 data 20,d5,ff,60              <jf>
2550 rem nachspann ================ <hl>
2560 rem * farbcodes/steuercodes * <ao>
2570 c4$=chr$(017):rn$=chr$(018)   <kd>
2580 rf$=chr$(146):b$=chr$(32)     <eh>
```

```
2590 b3$=b$+b$+b$:b4$=b3$+b$          <gg>
2600 return                           <pc>
2610 rem diagramm==============p4     <nd>
2620 rem 60671 bytes memory           <la>
2630 rem 08408 bytes program          <jk>
2640 rem 00049 bytes variables        <be>
2650 rem 00144 bytes arrays           <fb>
2660 rem 00338 bytes strings          <li>
2670 rem 12288 bytes graphic          <oh>
2680 rem 39444 bytes free (0)         <jc>
2690 rem ========================== <jd>
```

# DIAGRAMM

*Fortsetzung von Seite 80*

## 6. Programmende

Nach einer Sicherheitsabfrage wird das Programm durch einen Software-Reset gelöscht. Der Grafikspeicher bleibt erhalten.

Unter Verwendung eines Druckers und einer Hardcopy-Routine (wie die Hires-Hardcopy aus C16-P4 SPECIAL Nr. 1) lassen sich die erstellten Diagramme natürlich auch zu Papier bringen.

*Henning Koglin* □

INTER-COPY

# Von Kassette auf Diskette

**Wer sich eine Diskettenstation angeschafft hat, würde seine Software, die sich noch auf Kassetten befindet, gerne auf das neue Speichermedium kopieren. Autostart und Kopierschutz vereiteln jedoch dieses Vorhaben. In den meisten Fällen kann Inter-Copy helfen.**

Wenn das zu kopierende Programm sich mit dem Speicherplatz eines nicht erweiterten C16 begnügt, nach dem Laden den Interrupt wieder zuläßt und beim Laden weder den Interruptvektor verbiegt noch das ROM einschaltet, steht dem Kopieren von Kassette auf Diskette nichts mehr im Wege.
Das Programm Inter-Copy wartet darauf, daß nach dem Ladevorgang der Interrupt wieder zugelassen wird. Sobald das geschehen ist, werden sofort innerhalb eines Interruptes der ganze Bereich von $0000 bis $0FFF sowie die Ted-Register in einen anderen Speicherbereich verschoben und ein Reset ausgeführt. Das Programm startet also nicht automatisch. Um ein so abgebrochenes Programm wieder zum Laufen zu bringen, wird nicht die Startadresse gebraucht, sondern der Stackpointer. Man muß also außer dem ei-

*Bitte lesen Sie weiter auf Seite 86*

```
10 rem intercopy==============p4 <oj>
20 rem (p) commodore welt team    <ho>
30 rem ========================== <ng>
40 rem (c) by martin schulz       <me>
50 rem                            <pd>
60 rem basic v3.5                 <od>
70 rem plus4 (c16/116 + 64kb)     <cd>
80 rem floppy und datasette       <hh>
90 rem ========================== <jg>
100 fori=312to348                 <bi>
110 reada:pokei,a:next            <ne>
120 data 201,013,208,003,076,075  <oc>
130 data 236,134,208,041,015,170  <im>
140 data 165,209,072,165,210,133  <ko>
150 data 209,104,133,210,032,075  <co>
160 data 236,202,208,250,166,208  <ba>
170 data 024,096,024,096,076,075  <dm>
180 data 236                      <mi>
190 color4,6,4:color0,7,6         <ia>
200 scnclr:print:print:print      <fm>
210 poke209,32:poke210,166        <ei>
220 poke804,56:poke805,1          <mk>
230 print"adcbbbbfbfbe"           <bf>
240 print"bbdcabdbdbfbbb"         <ah>
250 print"bbdfdbdbfbbb"           <bh>
260 print"bbdfdbddde"             <hg>
270 print"bbdbacdbdbfd"           <eh>
280 print"bbdbbbdbdbfbab"         <di>
290 print"adcbbbdbdfbbbb"         <dm>
300 print:print:print             <hm>
310 print"ddddcecbbb"             <gg>
320 print"cbbbbbbbbbbbbbbb"      <lk>
330 print"cbfbbbbbbbbbbb"        <dh>
340 print"cbfbbbbedd"            <np>
350 print"cbfbbbbbhb"            <bd>
360 print"cbbbbbbbbbhb"          <al>
370 print"ddddcbhb"              <cm>
380 poke804,75:poke805,236        <cl>
390 print:print:print:print       <fa>
400 print"     druecken sie die sp
ace-taste !"                      <kh>
410 getkey a$:print:print:print   <gg>
420 print" inter-copy ist ein pro
gramm, das es":print              <pc>
430 print" ermoeglicht, c16-progr
amme mit auto-":print             <di>
440 print" start von kassette auf
 diskette zu":print               <ha>
450 print" uebertragen. hierzu er
forderlich ist":print             <nn>
460 print" ein c16 mit 64 kb.-erw
eiterung oder":print              <gc>
470 print" ein plus 4. da es vers
chiedene":print                   <dn>
480 print" moeglichkeiten des kop
ierschutzes":print                <ob>
490 print" gibt, gelingt diese ar
t des kopierens":print            <cg>
```

```
500 print"  nicht bei jedem progra
mm. auch dort,":print                <hc>
510 print"  wo programmteile von k
assette nachge-":print               <pd>
520 print"  laden werden, ist inte
r-copy nur":print                    <mj>
530 getkey a$                        <ni>
540 print"  bedingt anwendbar. der
 vorteil bei":print                  <cp>
550 print"  inter-copy liegt darin
, dass die":print                    <lk>
560 print"  startadresse eines pro
grammes nicht":print                 <do>
570 print"  mehr gebraucht wird. w
enn sie gleich":print                <mb>
580 print"  die space-taste drueck
en, wird der":print                  <fl>
590 print"  bildschirm geloescht u
nd die ein-":print                   <be>
600 print"  schaltmeldung erschein
t:":print                            <lp>
610 print"     inter-copy     1227
7 bytes free":print                  <fg>
620 print"  inter-copy ist dann ak
tiviert.":print                      <fj>
630 print"  bevor sie nun ein prog
ramm kopieren,":print                <el>
640 print"  laden sie es einmal zu
r probe und":print                   <db>
650 getkey a$                        <fo>
660 print"  merken sie sich den st
and des band-":print                 <cd>
670 print"  zaehlwerkes, nachdem e
s geladen ist!":print                <ek>
680 print"  nun koennen sie die fo
lgenden":print                       <bd>
690 print"  programmschritte zum k
opieren eines":print                 <jj>
700 print"  programmes anwenden. b
evor sie das":print                  <bi>
710 print"  tun, notieren sie sich
 diese bitte !":print                <ce>
720 print                            <do>
730 print"1.  laden sie inter-copy
 und starten es.":getkey a$:print    <di>
740 print"2.     laden sie das zu k
opierende"                           <eg>
750 print"       programm von kasse
tte.":print                          <on>
760 print"3.     kurz bevor der lad
evorgang"                            <ge>
770 print"  abgeschlossen ist, (1-
2 nummern des"                       <lp>
780 print"  bandzaehlwerkes vorher
), druecken sie"                     <kh>
790 print"  die tasten shift und a
 gleichzeitig"                       <jl>
800 print"  und halten diese gedru
eckt, bis der"                       <oo>
810 print"  ladevorgang beendet is
t.":print                            <hg>
820 print"4.  geben sie nun den be
fehl  monitor"                       <eh>
830 print"  ein und druecken sie d
ie return-taste."                    <ln>
840 print"  speichern sie nun das
programm mit"                        <kg>
850 print"  s"chr$(34)"programmnam
e"chr$(34)",8,1000,5060"             <lb>
860 print"  auf diskette ab.":prin
t                                    <pa>
870 print"5.     schalten sie nun d
en computer aus"                     <dl>
880 print"  und wieder an. geben s
ie den befehl"                       <ke>
890 print"  monitor   ein und drue
cken die return-"                    <dm>
900 print"  taste. laden sie das s
oeben abge-"                         <fj>
910 print"  speicherte programm mi
t"                                   <pb>
920 print"  l"chr$(34)" programmna
me"chr$(34)",8   und starten sie"    <ma>
930 print"  es mit     g 5000"       <hm>
940 getkey a$:scnclr:char,13,4,"mo
ment bitte !"                        <ec>
950 char,7,6,"inter-copy wird akti
viert !"                             <hk>
960 for i=0 to 433                   <bg>
970 read x:poke 20480+i,x            <hm>
980 next:sys dec("503c")             <lc>
990 data 120,160,0,185,0,64,153,0    <ej>
1000 data 0,200,208,247,238,5,80,2
38                                   <ng>
1010 data 8,80,173,8,80,201,16,208   <ho>
1020 data 234,185,64,80,153,0,255,
200                                  <cg>
1030 data 192,32,208,245,173,60,80
,141                                 <hf>
1040 data 0,16,173,61,80,141,1,16    <dc>
1050 data 173,62,80,141,2,16,162,2
48                                   <cn>
1060 data 154,76,195,252,120,160,0
,185                                 <no>
1070 data 0,128,153,0,128,200,208,
247                                  <pd>
1080 data 238,65,80,238,68,80,173,
68                                   <mj>
1090 data 80,201,253,208,234,160,1
92,185                               <cj>
1100 data 63,255,153,63,255,136,20
8,247                                <gh>
1110 data 169,63,32,180,80,141,63,
255                                  <fa>
1120 data 169,76,141,14,206,169,23
3,141                                <hc>
```

```
1130 data 15,206,169,80,141,16,206
,160                                <gn>
1140 data 10,185,245,255,153,245,6
3,136                               <do>
1150 data 208,247,76,143,81,162,12
6,232                               <pk>
1160 data 240,24,189,0,68,201,177,
208                                 <mo>
1170 data 246,232,232,232,16,12,18
9,0                                 <mj>
1180 data 68,201,63,208,234,222,0,
68                                  <ea>
1190 data 208,229,173,222,71,201,1
77,208                              <eo>
1200 data 10,173,225,71,201,63,208
,3                                  <pl>
1210 data 206,225,71,96,141,50,129
,141                                <jf>
1220 data 78,129,141,187,207,141,2
47,255                              <kl>
1230 data 141,130,4,141,158,4,141,
172                                 <jk>
1240 data 4,141,183,4,141,194,4,14
1                                   <ij>
1250 data 205,4,141,216,4,141,227,
4                                   <hi>
1260 data 141,225,7,96,141,63,255,
76                                  <bl>
1270 data 66,206,141,63,255,76,14,
206                                 <jn>
1280 data 3,169,0,141,48,253,141,8 <ed>
1290 data 255,173,8,255,201,123,20
8,111                               <ef>
1300 data 173,232,80,240,106,186,1
42,55                               <mg>
1310 data 80,160,0,185,0,0,153,0    <mf>
1320 data 64,200,208,247,238,5,81,
238                                 <ef>
1330 data 8,81,173,5,81,201,16,208 <mp>
1340 data 234,185,0,255,153,64,80,
200                                 <kd>
1350 data 192,32,208,245,169,0,141
,232                                <of>
1360 data 80,169,62,32,180,80,169,
206                                 <lo>
1370 data 141,19,3,141,21,3,141,19 <md>
1380 data 67,141,21,67,169,66,141,
18                                  <ao>
1390 data 3,141,18,67,169,14,141,2
0                                   <ho>
1400 data 3,141,20,67,173,0,16,141 <md>
1410 data 60,80,173,1,16,141,61,80 <po>
1420 data 173,2,16,141,62,80,32,13
3                                   <ij>
1430 data 80,141,62,255,76,164,242
,173                                <ae>
1440 data 19,3,201,206,208,10,169,
80                                  <lo>
1450 data 141,19,3,169,220,141,18,
3                                   <ik>
1460 data 173,21,3,201,206,208,10,
169                                 <gk>
1470 data 80,141,21,3,169,226,141,
20                                  <je>
1480 data 3,173,9,255,76,17,206,16
0                                   <na>
1490 data 20,185,157,81,153,207,12
8,136                               <ci>
1500 data 16,247,76,164,242,32,32,
32                                  <mj>
1510 data 32,73,78,84,69,82,45,67   <mf>
1520 data 79,80,89,32,32,32,32,32,
32,32                               <fk>
1530 rem ========================== <ge>
1540 rem  p r o g r a m m e n d e   <ie>
1550 rem ========================== <op>
```

# Inter-Copy

*Fortsetzung von Seite 84*

gentlichen Programm den kopierten Bereich $0000 bis $0FFF (jetzt $4000 bis $4FFF) und das Startprogramm mit dem Stackpointer auf Diskette abspeichern.
Man kann das Programm nun von Diskette laden und vom Monitor aus mit G5000 starten. Als Beweis, daß dies funktioniert, habe ich das Programm „Atlantis" auf diese Weise auf Diskette abgespeichert.
Es ist durch einen Reset nicht zu stoppen. Da ich die Interruptroutine erweitern mußte, habe ich das ganze ROM ins RAM kopiert. Die Erklärung zu Inter-Copy habe ich mit ins Programm eingebaut. Das eigentliche Programm beginnt mit Zeile 960.

**Zur Anwendung:**

1. Inter-Copy laden und starten. Programm von Kassette laden.
2. Kurz vor Abschluß des Ladevorganges die Tasten Shift und A drücken und gedrückt halten, bis ein Reset ausgeführt wird.
3. Programm vom Monitor abspeichern mit S"Programmname",8,1000,5060.
4. Programm vom Monitor aus laden mit L"Programmname",8 und starten mit G5000. □

# Vom Joystick bis Script Plus: Tips & Tricks für C16/P4

**Wie gewohnt, bringt die Seite 16 wieder Tips und Tricks, wie Sie Ihren Computer noch besser nutzen können. Diesmal zeigen wir Ihnen, wie man mit Hilfe der beiden Joystick-Ports eine Alarmanlage realisieren kann. Außerdem finden Sie das Hilfsprogramm Silbentrennung zur Bearbeitung von SCRIPT/PLUS-Texten.**

Die Funktion eines Joysticks ist denkbar einfach. Die Pin-Belegung ist wie folgt:

| Pin | Belegung | |
|---|---|---|
| 1 | oben | 8 7 6 |
| 2 | rechts | |
| 3 | unten | |
| 4 | links | 5 4 3 |
| 5 | + 5V | |
| 6 | Feuerknopf | |
| 7 | Masse | 2 1 |

Wird nun zum Beispiel Pin 1 (oben) mit Pin 7 (Masse) verbunden, ergibt PRINT JOY(1) die Zahl 1. Im Fachhandel gibt es konfektionierte Kabel mit entsprechendem Stecker. Sicherheitshalber sollte

## PIN-BELEGUNG DES JOYSTICK-PORTS – PORT 1 UND 2 ALS KONTROLL-STATION

der Anschluß für +5V isoliert werden, damit kein Kurzschluß entsteht. Die anderen Anschlüsse sind in diesem Zusammenhang unproblematisch. Wenn das Geschäft keine passenden Kabel vorrätig hat, kann auch ein Joystick-Adapter verwendet werden, der normalerweise benutzt wird, um Joysticks mit neunpoliger Sub-D-Buchse (C64) an den C16/Plus4 anzuschließen. Hier sind in der Regel nur die sechs für den Joystick notwendigen Kabel angeschlossen. Ein einfacher Test kann mit einem Miniprogramm durchgeführt werden:

```
0 print joy(1):goto
```

Wird nach dem Starten dieser Zeile eines der Kabel von 1 bis 4 oder 6 mit Pin 7 verbunden, ändert sich der ausgedruckte Wert. Übrigens springt ein GOTO ohne Zeilennummer zu Zeile 0.

## DER C16 ALS ALARMANLAGE

Die Verbindung der entsprechenden Pins mit Masse (Pin 7) kann auf vielfältige Weise geschehen. Die einfachste Methode wird im Innern des Joystick angewandt. Hier sind einfach kleine Schalter oder Metallkontakte, die die einzelnen Pins auf Masse legen. Genausogut kann man auch sogenannte Reed-Relais verwenden. Diese werden bevorzugt in Alarmanlagen eingesetzt, um das Öffnen und Schließen von Fenstern und Türen zu kontrollieren. Ein Reed-Relais besitzt zwei metallische Kontakte, die sich schließen, wenn ein Magnet in die Nähe gebracht wird und sich öffnet, wenn er wieder entfernt wird. Diese Schaltkontakte sind sehr klein und zudem preiswert zu erhalten. Genauso kann mit einem Quecksilber-Neigungsschalter das Anheben oder Erschüttern eines Gegenstandes kontrolliert werden. In gleicher Weise kann eine Matte mit Trittkontakten eingesetzt werden. Soll die Temperatur eines Gerätes überwacht werden, so kann dafür ein Temperaturschalter verwendet werden. Dieser betätigt sich bei einer bestimmten Temperatur (zum Beispiel 100 Grad) und man kann dies mit dem Joystickport leicht überwachen. In der gleichen Weise kann mit einer Reflexlichtschranke oder einer Gabellichtschranke das Vorhandensein oder Fehler eines Metall- oder Papierbandes überwacht werden. Mit Hilfe einer Taschenlampenbirne oder einer infraroten Leuchtdiode und eines Fototransistors kann eine einfache Lichtschranke gebaut werden. Dazu eignet sich zum Beispiel als Sender der Typ LD261 und als Empfänger BPX81. Der Kontakt ist geschlossen, wenn der Fototransistor beleuchtet wird. Er ist offen, wenn Dunkelheit herrscht. Für eine solche Anwendung haben wir das Beispielprogramm **SCHALTER** geschrieben.

Eine typische Anwendung wäre ein Physikerversuch mit der sogenannten schiefen Ebene. Entlang der Fahrbahn eines kleinen Modellautos befinden sich zehn Fototransistoren, die mit den entsprechenden Pins an den Ports 1 und 2 (mit gemeinsamer Masse) verbunden werden. Auf dem Wagen befindet sich eine kleine Lampe, die die Transistoren beim Vorbeifahren schaltet. Die Schalterstellung wird angezeigt. Die Zeitdifferenz zwischen den Schaltern wird in Zeile 80 bestimmt. Da es nur zehn Werte sind, braucht keine DIM-Anweisung zu erfolgen.

Bei diesem Programm wird vorausgesetzt, daß nur jeweils ein Kontakt geschlossen wird. Für Anwendungen, bei denen zunächst alle Schalter geschlossen sind und nur einer geöffnet wird, muß auf die Speicherstelle $FF08 zurückgegriffen werden.

## JOYSTICK-ABFRAGE OHNE JOY

Der entscheidende Nachteil der JOY-Funktion für Meßzwecke ist die Tatsache, daß die Funktion nur vernünftige Werte liefert, wenn die Kontakte eine der mit dem Joystick möglichen Stellungen entspricht. Was vernünftig ist, da zum Beispiel „oben" und „unten" gleichzeitig nicht möglich ist. Die Abfrage der Joystickpositionen ohne die JOY-Funktion erfordert leider etwas Übung mit Maschinensprache und Kenntnis der Hardware des C16. Die vier Leitungen für „oben", „rechts", „unten" und „links" haben bei Port 1 und Port 2 die gleichen Leitungen und sind mit der Tastaturmatrix verbunden. Die beiden Leitungen für die Feuerknöpfe sind getrennt. Die Verbindung mit der Tastaturmatrix wird offensichtlich, wenn der Joystick mit GETa$ abgefragt wird.

Hierzu ein kleines Beispiel:

```
10 v$="56rdt":rem port 1
20 v$=v$+"34wa":rem port 2
30 geta$:printinstr(v$,a$):goto30
```

Hier entspricht „1" der Position „oben" bei Port 1 und „6" der Position „oben" bei Port 2. Die Feuertaste von Port 2 kann so nicht abgefragt werden.
Umgekehrt erhält man die richtigen Werte mit JOY(1) und JOY(2) durch die Tasten 1, 2, HOME, CTRL und SPACE(=Feuer). Ein Joystickspiel, das mit der JOY-Funktion arbeitet, kann also ohne Joystick gespielt werden. Nicht so Maschinenprogramme. Hier erfolgt die Abfrage komplizierter. Wir haben für unsere Leser daher ein kleines Maschinenprogramm geschrieben, das die Joystickposition abfragt:

```
. 065e  78       sei
. 065f  a2 fa    ldx #$fa
. 0661  8e 08 ff stx $ff08
. 0664  ad 08 ff lda $ff08
. 0667  8e 08 ff stx $ff08
. 066a  cd 08 ff cmp $ff08
. 066d  d0 f2    bne $0661
. 066f  49 ff    eor #$ff
. 0671  58       cli
. 0672  60       rts
. 0673  78       sei
. 0674  a2 fd    ldx #$fd
. 0676  8e 08 ff stx $ff08
. 0679  ad 08 ff lda $ff08
. 067c  8e 08 ff stx $ff08
. 067f  cd 08 ff cmp $ff08
. 0682  d0 f2    bne $0676
. 0684  49 ff    eor #$ff
. 0686  58       cli
. 0687  60       rts
```

Dies sind zwei Routinen. Die erste (für Port 1) beginnt bei $065e und wird mit SYS1630 aufgerufen, die zweite beginnt bei $0673 und wird mit SYS1651 gestartet. Man kann sich die Tipparbeit erleichtern, indem man mit dem Transfer-Befehl einen Teil der JOY-Routine im Betriebssystem (ab $BFC1) kopiert:

T BFCD BFDD 0660

Nach dem Aufruf der Routinen steht das jeweilige Ergebnis im Akku-Register und kann durch PEEK(2034) abgefragt werden.

Mit Hilfe von IF-Abfragen und der AND-Funktion kann jede Schalterstellung erfragt werden:

Beispiel:

```
100 sys1630:b=peek(2034)
110 if band1 then print"oben"
120 if band2 then print"unten"
130 if band4 then print"links"
140 if band8 then print"rechts"
150 goto100
```

## PORT-AUSWAHL DURCH SCHREIBZUGRIFF

Wie weiter oben erwähnt wurde, haben die Pins von Port 1 und Port 2 teilweise die gleichen Leitungen. Wie kann dann unterschieden werden, von welchem Port das Signal kommt? Dies geschieht durch einen Schreibzugriff (STX $....) auf $FF08. Je nachdem, welcher Wert gePOKEd wird, liegt die Masseleitung von Port 1 oder Port 2 an Masse. Die beiden möglichen Werte sind:

Port 1:
$FA = 250 = 11111010
Port 2:
$FD = 253 = 11111101

Der Schreibzugriff wird zur Sicherheit wiederholt, um kurzfristige Schwankungen auszugleichen. Die Abfrage geht auch in BASIC:

```
10 a=65288
20 pokea,250:b=peek(a):rem port 1
30 print
band1;band2;band4;band8;band64;band128:
goto 20
```

Die Ergebnisse sind jedoch fehlerhaft, da manchmal der Interrupt zwischen POKE und PEEK den Wert verändert.
Wenn der C16/Plus4 ein Ereignis an einem der Joystickports feststellt, muß dieses Ergebnis in der Regel zu einem Alarmsignal verarbeitet werden. Die einfachste Möglichkeit besteht in der Verwendung des Sound-Befehls und der entsprechenden Bildschirmanzeige. Jedoch kann meist der Fernseher nicht die ganze Zeit eingeschaltet bleiben. Daher hier ein einfacher Tip, wie ein Alarm ausgelöst werden kann. Am Kassettenport stehen fünf Volt zur Verfügung, mit denen normalerweise der Recordermotor betrieben wird. Wird hier ein Piezosummer (verbraucht sehr wenig Strom) angeschlossen, so kann durch Einschalten des Kassettenrecorders dieser Summer betätigt werden. Genauso kann ein Relais gesteuert werden, das eine Sirene in Betrieb setzt.

## AUSLÖSEN DES ALARMS

Wir hoffen, daß Sie durch diese Beispiele angeregt wurden, Ihren C16/Plus4 zu Meßzwecken einzusetzen. Wir würden uns freuen, wenn Sie uns Ihre Anwendungen mitteilen würden.

## TEXTE RECHTSBÜNDIG AUSDRUCKEN

Mit SCRIPT/PLUS kann man auf einfache Weise beliebige Texte erstellen. Besonders schön werden diese mit der Formatan-

Beispiel:

```
10 sys1630:b=peek(2034):rem port1
20 print
band1;band2;band4;band8;band64;
30 sys1651:b=peek(2034):rem port2
40 print chr$(18)band1;band2;band4;band
8;band128:goto10
```

weisung **jul** (justification ein). Diese bewirkt, daß alle Zeilen gleich lang werden. Fehlende Buchstaben werden durch Leerzeichen zwischen den Worten aufgefüllt. Damit dabei nicht zu große Lükken entstehen, muß am Zeilenende richtig getrennt werden.
Hierzu hat SCRIPT/PLUS die Möglichkeit, sogenannte Soft Hyphen (weiche Trennungszeichen) einzufügen. Dies geschieht mit ESC und COMMODORE/–. Im Handbuch ist

## TRENNVORSCHLÄGE

hier ein Druckfehler und es wird die Ctrl-Taste angegeben. Steht ein solches Zeichen im Wort, so wird an dieser Stelle getrennt, wenn sie am Zeilenende zu stehen kommt. Der Nachteil liegt darin, daß nur ein solches Trennungszeichen eingesetzt werden kann. Mehrere Trennvorschläge schaden zwar nichts, da sie normalerweise nicht ausgedruckt werden, aber es wird prinzipiell das erste verwendet, auch wenn ein anderes eine günstigere Trennung ergäbe. Soll ein wichtiger Text im Blocksatz (beispielsweise rechtsbündig) gedruckt werden, ist es unumgänglich, die Trennungen von Hand einzufügen. Dazu muß die Zeilenlänge im Eingangsmenü auf die gewünschte Länge eingestellt werden und nach jeder Silbentrennung der Abschnitt neu formatiert werden. Dies ist sehr aufwendig. Daher haben wir Ihnen das Hilfsprogramm **SILBENTRENNUNG** geschrieben, mit dem die optimale Trennung auf einfache Weise möglich ist. Das Programm ist auf Diskettenbetrieb ausgelegt, kann aber leicht mit Hilfe des Handbuchs für Kassettenbetrieb umgeschrieben werden. Der mit SCRIPT/PLUS (oder einem anderen Programm das mit ASCII-Zeichen arbeitet) erstellte Text wird in den Speicher geladen und Zeile für Zeile abgearbeitet. Immer, wenn ein Leerzeichen oder Trennungszeichen (auch Soft Hyphen) gefunden wird, überprüft unser Programm, ob die erforderliche Zeilenlänge erreicht ist. Bei einer zu langen Zeile wird diese am Schirm angezeigt und mit Hilfe der Cursortasten (links und rechts) das Trennungszeichen an die gewünschte Position gebracht. Mit Return wird die Zeile abgeschlossen und das Programm fügt automatisch den Return-Code 13 in den Text ein.

## PROGRAMM-ERKLÄRUNG

**Zeile 100:** Der Speicher wird begrenzt, um einen geschützten Textspeicher zu schaffen.
**Zeile 160–180:** Die Zeilenparameter und der Filename werden abgefragt. Die Variable DM gibt an, wieviele Zeichen noch bis zum Zeilenende fehlen dürfen, damit das Programm automatisch ein Return setzt.
**Zeile 200–250:** Einlesen des ASCII-Textes in den Speicher. Die Print-Anweisung in 230 kann weggelassen werden (der Ablauf wird dadurch schneller).
**Zeile 280:** Das alte Textfile wird in eine Sicherheitskopie umbenannt. Hier kann der Rename-Befehl nicht verwendet werden, da dieser keine Variablen verarbeitet.
*(Anmerkung der Redaktion:* Der Rename-Befehl kann Variablen verarbeiten, wenn diese in Klammern eingeschlossen werden.)
**Zeile 300:** Seq-File zum Schreiben öffnen.
**Zeile 330–370:** Überprüfung, ob ein Trennungszeichen gefunden wurde.
**Zeile 380–390:** Prüfung auf Zeilenende. Ist die maximale Zeilenlänge erreicht und das letzte Trennungszeichen zu weit ent-

## SILBENTRENNUNG

```
10 rem silbentrennung =========c16 <kg>
20 rem (p) commodore welt team    <ho>
30 rem ============================ <ng>
40 rem (c) by  r. schmid-fabian   <jp>
50 rem              heidelberg    <kp>
60 rem                            <ah>
70 rem basic v3.5                 <nl>
80 rem c16/116/p4   floppy        <ba>
90 rem ============================ <jg>
100 poke55,0:poke56,39:clr        <bi>
110 l$=" ":b$="-":rem trennzeichen <le>
120 sh$=chr$(192):rem soft-trennze
ichen                             <li>
130 cr$=chr$(013):cl$=chr$(147)   <bm>
140 li$=chr$(157):re$=chr$(29)    <el>
150 ro$=chr$(146):rv$=chr$(18):d$=
l$+l$+l$+l$                       <pb>
160 input"max. Zeilenlaenge";ml   <df>
170 input"max. Abweichung";dm     <ae>
180 input"file-Name";p$           <nn>
190 rem ****** einlesen ********** <la>
200 trap250                       <kc>
210 open2,8,2,p$+",s,r"           <de>
220 fori=12000to60000             <kj>
230 get#2,a$:pokei,asc(a$):printa$
;                                 <bo>
240 if st=0 then next             <mi>
250 close2:hlt=i                  <hd>
260 rem **** rename v. p$  ******* <el>
270 printchr$(147)"Jetzt wird "+ch
r$(34)+p$+chr$(34)+" umbenannt (Ta
ste)":getkeya$                    <na>
280 open1,8,15,"r:"+left$(p$+d$+d$
+d$,12)+".bak+"+"="+p$:close1     <dl>
290 rem ****** bearbeiten ******** <fc>
300 open2,8,2,p$+",s,w"           <kp>
310 for i=12000 to hlt            <gm>
320 a$=chr$(peek(i))              <gp>
330 if a$=sh$ then tp=zl:en$=b$:go
to380                             <nb>
340 if a$=cr$ then printrv$w$:prin
t#2,w$:zl=0:w$="":goto440         <jh>
350 w$=w$+a$:zl=zl+1              <gk>
360 if a$=1$ then tp=zl:en$=""    <bf>
370 if a$=b$ then tp=zl:en$=b$    <je>
380 if zl<ml then 440             <oa>
390 if(ml-tp)>dm then gosub 480   <na>
400 pr$=left$(w$,tp)+en$:printrv$p
r$                                <ph>
410 print#2,pr$                   <ig>
420 w$=right$(w$,ml-tp)           <gk>
430 tp=0:zl=len(w$)               <na>
440 next                          <pf>
450 close2:end                    <pg>
460 rem ****** trennung ********** <oo>
470 rem ****** von hand ********** <og>
```

```
480 d=0:x$="":forv=i+1toi+6:x$=x$+
chr$(peek(v)):nextv                    <gh>
490 if asc(x$)=32 then i=i+1:tp=le
n(w$):en$="":return                    <np>
500 x$=rv$+x$                          <pn>
510 pr$=right$(w$,ml):z$=left$(pr$
,ml-d):t$=right$(pr$,d)                <lo>
520 en$=b$:p=peek(i-d):if p=32 the
n en$=l$                               <le>
530 printchr$(147)+z$+rv$+en$+ro$+
t$+x$                                  <al>
540 getkeyq$                           <jj>
550 if q$=li$ then d=d+1               <gk>
560 if q$=re$ then d=d-1:if d<0 th
en d=0                                 <kb>
570 if q$=cr$ then tp=len(w$)-d:en
$="-":printcl$:return                  <ga>
580 goto510                            <ap>
590 rem ========================== <ce>
600 rem  p r o g r a m m e n d e   <gm>
610 rem ========================== <id>
```

## SCHALTER.C16

```
10 rem schalter ==============c16 <im>
20 rem *************************** <cb>
30 rem ** abfrage der optischen ** <gg>
40 rem ******* schalter ********** <ah>
50 scnclr:for i=1 to 10:char,1,i,"
schalter"+str$(i-1)+":":next           <gb>
60 t0=ti                               <ag>
70 do:a=joy(1):b=joy(2):loop until
 a+b                                   <ij>
80 dt=ti-t0                            <kc>
90 if(a+b)=128 then a=-9*(a=128):b
=-9*(b=128)                            <mb>
100 sw=(a+b+1)/2-5*(b>0)               <ga>
110 for i=1 to 10:char,15,i,str$(-
(i=sw)):next                           <fo>
120 goto60                             <ia>
130 rem ========================== <np>
140 rem  p r o g r a m m e n d e   <fe>
150 rem ========================== <nj>
```

fernt, wird in das Unterprogramm ab Zeile 480 verzweigt.
Zeile 480: Die nächsten fünf Zeichen werden aus dem Speicher geholt, damit man beim Trennen die Fortsetzung des Wortes sehen kann.
Zeile 490: Ist das Folgezeichen ein Leerzeichen, so wird die Zeile selbständig abgeschlossen.
Zeile 500–550: Die überlange Zeile wird angezeigt. Der Wort- oder Satzteil, der über die maximale Zeilenlänge hinausgeht, erscheint revers.

### ZWEISPALTIG DRUCKEN

SCRIPT/PLUS ist zwar sehr komfortabel und auch das im Plus4 eingebaute Textsystem erlaubt (in gewissen Grenzen) die Erstellung von größeren Texten. Natürlich kann man für diesen Preis kein Desktop-Publishing-Programm verlangen. Aber es können auf einfache Weise mit SCRIPT/PLUS auch zweispaltige Texte erstellt werden. Dazu muß der Text zuerst vollkommen fertig eingegeben sein. Dann begrenzt man die Zeilenlänge zum Beispiel auf 40 Zeichen mit dem Programm **SILBENTRENNUNG**. Im Eingangsmenü von SCRIPT/PLUS wird die gesamte Zeilenlänge (zwei Spalten + Abstand + Rand) eingestellt. Nachdem der Text eingeladen wurde (Wordwrap ausschalten) ist der Text auf die linke Hälfte der verfügbaren Breite beschränkt. Jetzt kommt eine außergewöhnliche Eigenschaft von SCRIPT/PLUS zur Anwendung – nur wenige Textsysteme haben diese. Mit Esc und Shift R kann ein kompletter Textblock markiert werden (es erscheint **SET COLUMN RANGE**) und nach Drücken von Return kann dieser mit den Cursortasten verschoben werden. Angenommen, es sollen 50 Zeilen auf eine Seite gedruckt werden, so muß der Text ab Zeile 51 bis 100 als Block markiert und zuerst nach rechts und dann nach oben verschoben werden, bis die beiden Spalten nebeneinander stehen. Vorsicht! Der markierte Block kann den stehenden Text überschreiben, wenn der Block über diesen Text geschoben wird. Dieser Vorgang wiederholt sich, bis alle Seiten zweispaltig im Speicher stehen. Dabei kann auch Platz für Bilder gelassen werden. Nach Einfügen von Formatzeichen und Seite-Ende-Kennung kann jetzt der zweispaltige Text ausgedruckt werden. □

DER COMPUTER SPIELT NACH NOTEN

# Haste Töne?

**Vom Blatt zu singen oder zu spielen, ist nicht jedermanns Sache. Schreiben Sie die Noten doch einfach ab, und zwar mit dem Computer. Dann können Sie hören, wie es richtig klingt. Außerdem läßt sich die Melodie in eigene Programme einbinden. Gleich zwei Programme sorgen für die richtige Musik.**

Beide Programme gehören zusammen. Das erste Programm heißt „Notenblatt", das zweite „Datamaker Musik".

### NOTENBLATT

Nach dem Start können Erklärungen abgerufen werden. Danach wird das Notenblatt mit vier Notenzeilen und Tastenerklärungen gezeichnet.
Jeder, der mit der Musik nicht allzuviel am Hut hat, der auch nicht Klavierspielen kann und somit mit

professionellen Programmen nicht zurechtkommt, kann mit diesem Programm die Noten aus einem Liederbuch einfach abschreiben.
Das Programm verarbeitet 18 Noten mit den dazugehörigen Halbtönen. Eingangs gesetzte # oder b werden über das ganze Stück beibehalten. Auflösungen von = oder b im Takt werden nach Setzen des nächsten Taktstriches für weitere Eingaben wieder aufgehoben. Ebenso gelten im Takt geschriebene # oder b nur für den betreffenden Takt.
Es ist auch möglich, die im Stück vorgesehenen Pausen mit einzugeben.
Das Stück kann jederzeit abgespielt werden, danach lassen sich weitere Eingaben vornehmen. Insgesamt sind 500 Töne und Pausen möglich.
Das Stück kann nach Beendigung der Eingabe auf Diskette gespeichert werden.

## DATAMAKER MUSIK

Dieses Programm wandelt ein sequentielles File (Musikstück) in Datazeilen zur Verwendung in eigenen Programmen um. Um diese Data auch verwenden zu können, ist es am besten, das in den Zeilen 500 bis 600 des Programms unter REM angegebene Programm zu verwenden, das auch einen Ausstieg aus dem Musikstück ermöglicht.

### 1. Notenblatt

**Dokumentation:**

| | |
|---|---|
| Zeilen 100– 590 | Notenblatt und SHAPES zeichnen. |
| Zeilen 610– 810 | Hauptteil der Eingabe. |
| Zeilen 830– 900 | Erkennen der Tonhöhe und Tondauer. |
| Zeilen 920–1020 | Unterprogramm "Ende". |
| Zeilen 1040–1160 | Laden einer Melodie von Diskette. |
| Zeilen 1180–1310 | Speichern einer Melodie (Diskette). |
| Zeilen 1320–1520 | Setzen der Tonerhöhungs- (#) und Tonerniedrigungszeichen (b). |
| Zeilen 1540–1580 | Spielen einer eingegebenen oder geladenen Melodie. |
| Zeilen 1600–1660 | Schreiben des Auflösungszeichens (# und b werden für einen Takt aufgehoben). |
| Zeilen 1680–1800 | Setzen des Taktstriches. |
| Zeilen 1820–1870 | Notenzahl (500 Noten) überschritten. |
| Zeilen 1890–2030 | Löschen einer Eingabe. Nur die jeweils letzte Eingabe kann gelöscht werden. |
| Zeilen 2050–2160 | Schreiben des Wiederholungszeichens. Die Noten, die zwischen zwei Wiederholungszeichen stehen, werden nochmals übernommen. |
| Zeilen 2180–2260 | Setzen des Eingabepfeiles. |
| Zeilen 2280–2340 | Schreiben der Pausen. |
| Zeilen 2360–2860 | Variablen, Einleitung. |
| Zeilen 2870–2980 | Fehlerkanalabfrage. |
| Zeilen 3000–3170 | Notendata. |

**Erläuterungen zum Programmablauf:**

Nachdem das Programm mit RUN gestartet wurde, können Bedienungsinstruktionen abgefragt werden. Danach wird das Notenblatt gezeichnet sowie am unteren Bildschirmrand die Tastenbedeutung angezeigt. Im einzelnen:

0 = Wiederholung Anfang.
g = Wiederholung Ende.
1 = Ganze Note, 2 = halbe Note, 3 = viertel Note, 4 = achtel Note, 5 = sechzehntel Note, 6 = zweiunddreißigstel Note.
7 = #, 8 = b.
A = Auflösungszeichen, hebt # oder b für diesen Takt auf.
B = Zeichnet den Taktstrich (wichtig, wenn innerhalb eines Takts ein Auflösungszeichen oder ein # oder ein b geschrieben wurde).
C = Ganze Pause, D = halbe Pause, E = viertel Pause.
F = Achtel Pause, G = sechzehntel Pause.
H = Ende des Programms mit Sicherheitsabfrage, ob Melodie gespeichert wurde.
L = Laden einer Melodie von Diskette.
S = Speichern einer Melodie auf Diskette.
P = Spielen einer eingegebenen Melodie zu jedem Zeitpunkt der Eingabe oder einer von Diskette geladenen Melodie.
N = Löschen aller Eingaben und Neustart. Vorsicht: Keine Sicherheitsabfrage!
+ = Tondauer des zuletzt eingegebenen Tones oder einer Pause wird um die Hälfte des Wertes verlängert.
– = Löscht die letzte Eingabe (Note oder Taktstrich oder # oder b oder Auflösungszeichen oder Pause oder Wiederholungszeichen).

**Zur Eingabe:**

Nachdem das Notenblatt und der Eingabepfeil oben links gezeichnet sind, können die Noten von einem Musikstück abgeschrieben werden.
Zu Anfang – wie beim Musikstück auch – wird die Tonart durch # oder b bestimmt. Dazu mit „Cursor down" und „Cursor up" auf oder zwischen die entsprechenden Linien der ersten Notenzeile fahren und 7 oder 8 drücken. Danach auf die entsprechende Notenlinie fahren und das nächste # oder b schreiben. Diese Eingaben gelten für das ganze Stück.
Jede Notenzeile besteht aus fünf Notenlinien und vier Hilfslinien. Die Hilfslinien sind auf einem normalen Notenblatt nicht gezeichnet und hier nur gestrichelt. Im Bereich der unteren Hilfslinie bis zur oberen Hilfslinie jeder Notenzeile können Noten plaziert werden.
Kommt im Musikstück ein Taktstrich, ist dieser mit B zu schreiben. Die richtige Spalte sucht sich der Computer selbst für alle Eingaben. Die Wiederholungszeichen sehen wie folgt aus:

```
Anfang   ||: 0
Ende     :|| g
```

Wenn im Musikstück von Anfang bis Ende wiederholt werden soll, genügt die Eingabe g am Ende des Musikstückes, und das ganze Stück wird wiederholt (nicht nochmal geschrieben, aber die Noten werden zweimal hintereinander gespeichert).
Kommt im Musikstück zwischendurch ein # oder b, so ist unbedingt auf die Eingabe des Taktstriches zu achten, denn # oder b gelten nur bis zum nächsten Taktstrich. Das gleiche gilt für ein Auflösungszeichen. Wird der Taktstrich vergessen, wird ab dieser Stelle das ganze Stück falsch, da dann jede entsprechende Note mit Erhöhung, Erniedrigung oder beim Auflösungszeichen ohne diese gespeichert wird.
Während der Eingabe kann jederzeit das Musikstück mit P gespielt und die Eingabe dann fortgesetzt werden.

*Bitte lesen Sie weiter auf Seite 95*

```
10 rem notenblatt ==============p4 <li>
20 rem (p) commodore welt team    <ho>
30 rem ============================ <ng>
40 rem (c) by reiner hickel       <hg>
50 rem                            <pd>
60 rem                            <ah>
70 rem basic v3.5                 <nl>
80 rem plus4 (c16/116 + 64kb),disk <gc>
90 rem ============================ <jg>
100 graphic1,1:clr:gosub3250:color
1,2,4:color0,1:scnclr             <gl>
110 gosub2360:volla               <kn>
120 scnclr:forj=16to136step40     <mk>
130 fori=0to4:sound3,750+i*50,5   <lm>
140 draw1,10,j+i*4to319,j+i*4     <ng>
150 nexti,j                       <ee>
160 forj=8to128step40             <jg>
170 fori=jtoj+4step4              <dp>
180 fork=10to314step8:sound3,700+k
,2                                <em>
190 draw1,k,itok+4,i              <go>
200 draw1,k,i+28tok+4,i+28        <no>
210 nextk,i,j                     <lo>
220 draw1,10,0to10,168:sound3,900,
1                                 <ln>
230 draw1,0,168to319,168to319,169t
o0,169:sound3,900,2               <dg>
240 char,11,22,"tastenbedeutungen" <ic>
250 char1,0,22,"loesch= -":char1,0
,23,"play"+b2$+"= p"              <fp>
260 char1,0,24,"save"+b2$+"= s"   <mk>
270 char1,30,22,"punkt = +":char1,
31,23,"ende = h"                  <og>
280 char1,31,24,"load = l"        <jb>
290 z=0:fori=11to20               <pa>
300 char,11,23,"0123456789abcdefg" <kh>
310 draw1,89,192to89,198:draw1,91,
192to91,198:draw1,93,194:draw1,93,
196:sound3,900,4                  <mo>
320 fori=99to139step8:sound3,850+i
,45                               <jh>
330 circle1,i,197,2,1             <id>
340 ifi>107thenpaint1,i,197,1:soun
d3,650,5                          <aa>
350 nexti                         <mj>
360 fori=109to141step8:sound3,850+
i,1                               <io>
370 draw1,i,192toi,197            <hg>
380 nexti:sound3,950,10           <cc>
390 draw1,125,192to127,192        <dp>
400 draw1,133,192to135,192:draw1,1
33,193to135,193                   <nh>
410 draw1,141,192to143,192:draw1,1
41,193to143,193:draw1,141,194to143
,194                              <of>
420 draw1,146,194to146,199:draw1,1
48,193to148,198:draw1,145,195to150
,192                              <pj>
430 draw1,145,199to150,196        <hk>
440 draw1,154,192to154,198to157,19
6to157,194to154,194               <og>
450 draw1,161,194:draw1,161,196:dr
aw1,163,192to163,198:draw1,165,192
to165,198                         <fg>
460 draw1,169,192to169,197to172,19
6                                 <oi>
470 draw1,169,194to172,193to172,19
9                                 <ad>
480 draw1,179,192to179,199        <oh>
490 draw1,186,193to189,193to189,19
4to186,194:draw1,184,192to191,192 <nc>
500 draw1,192,195to199,195:draw1,1
94,193to197,193to197,194to194,194 <ek>
510 draw1,203,192to204,194to203,19
6to204,197to203,198               <mk>
520 draw1,210,194to214,194to211,19
9                                 <oh>
530 draw1,218,194to222,194to217,19
9:draw1,218,195to221,195          <ge>
540 draw1,3,6to6,8to3,10:ifoo=1the
noo=0:ns=an+10:z=1:zp=6:return    <df>
550 fori=88to216step8             <dj>
560 sshapen$((i-88)/8),i,192,i+7,1
99                                <pi>
570 nexti:sound1,500,3            <jg>
580 sshapep$,0,6,6,10:sshapecl$,0,
11,6,19                           <em>
590 sshapecn$,11,0,18,42          <jh>
600 rem =hauptschleife=========== <lm>
610 z=1                           <bo>
620 do                            <mh>
630 getq$:ifq$=""then610          <hn>
640 sound3,934,1                  <kf>
650 ifq$=cu$thenh=zp:zp=zp+2:gosub
2180                              <mj>
660 ifq$=co$thenh=zp:zp=zp-2:gosub
2180                              <nd>
670 ifq$="0"thenwa=nz:gosub2050   <no>
680 ifq$="9"thenwe=nz:gosub2050   <gd>
690 ifq$="+"thensound3,750,5:l(nz)
=1.5*l(nz)                        <pl>
700 ifq$="-"thengosub1890         <pj>
710 ifq$="8"orq$="7"thengosub1330 <mm>
720 ifq$>"0"andq$<"7"thengosub830 <bk>
730 ifq$="a"thengosub1600         <cn>
740 ifq$="b"thengosub1680         <ib>
750 ifq$="h"then920               <jb>
760 ifq$="l"thengosub1040         <nh>
770 ifq$="s"thengosub1180         <em>
780 ifq$="p"thengosub1540         <io>
790 ifq$="n"thenrun               <jc>
800 ifq$>"b"andq$<"h"thengosub2280 <ka>
810 loop                          <eg>
820 rem =noten erkennen=========== <nj>
830 gosub2180:w=(zp-(z-1)*40-4)/2 <lk>
840 a=h(w)+a(w)+1:q=val(q$):h=zp-3 <el>
```

```
850 nz=nz+1:ifnz>500thengosub1820  <oe>
860 s(nz)=t(w,a)                   <ea>
870 l(nz)=64/(2↑(q-1))             <al>
880 sound1,s(nz),l(nz)             <ah>
890 gosub2240                      <gl>
900 lo=0:return                    <kj>
910 rem =ende=====================  <ni>
920 graphic0,1                     <pc>
930 printcu$cu$tab(3)"ist die melo
die gespeichert? (j/n)"            <id>
940 getkeyx$                       <lk>
950 ifx$="j"thengraphicclr:end     <ph>
960 printcu$tab(3)"soll sie gespei
chert werden? (j/n)"               <mo>
970 getkeyx$                       <ji>
980 ifx$="j"thengosub1180          <em>
990 printcu$tab(3)"weitere melodie
n eingeben (j/n)"                  <lk>
1000 getkeyx$:ifx$="j"thengraphic1
,1:goto120                         <gm>
1010 graphicclr:end                <lb>
1020 goto920                       <ln>
1030 rem =melodie laden===========  <ia>
1040 graphic0,1                    <hk>
1050 printcu$cu$tab(3)"mit <d> wer
den alle musiktitel"               <hm>
1060 printcu$tab(3)"ausgedruckt, s
onst andere taste"                 <bc>
1070 getkeyx$:ifx$="d"thendirector
y"*=s"                             <ce>
1080 printcu$tab(3);:input"titel d
er melodie";na$:na$=left$(na$,12)  <pl>
1090 open2,8,2,na$+",s,r"          <an>
1100 input#2,nz                    <hh>
1110 fori=1tonz                    <jo>
1120 input#2,s(i)                  <bc>
1130 input#2,l(i)                  <ml>
1140 nexti                         <cn>
1150 close2                        <hn>
1160 graphic1,0:return             <cc>
1170 rem =melodie speichern=======  <ml>
1180 graphic0,1                    <eb>
1190 printcu$cu$tab(3);:input"tite
l der melodie";na$:na$=left$(na$,9
)                                  <ok>
1200 open15,8,15                   <lp>
1210 open2,8,2,na$+",s,w"          <if>
1220 gosub2870:ifu=1thenu=0:goto11
90                                 <jp>
1230 ifu=2thenu=0:goto1200         <ok>
1240 ifu=3thenu=0:return           <nm>
1250 print#2,nz                    <il>
1260 fori=1tonz                    <nl>
1270 print#2,s(i)                  <gb>
1280 print#2,l(i)                  <ga>
1290 nexti                         <ik>
1300 close2:close15                <gd>
1310 graphic1,0:return             <lc>
1320 rem =setzen von # oder b=====  <ni>
1330 ifnz>0then1440                <on>
1340 w=(zp-(z-1)*40-4)/2:ifval(q$)
=7thenh(w)=1:elseh(w)=-1           <pl>
1350 ifw<5thenh(w+7)=h(w):h(w+14)=
h(w)                               <hn>
1360 ifw>4andw<12thenh(w+7)=h(w)   <cp>
1370 ifw>7andw<15thenh(w-7)=h(w)   <be>
1380 ifw>14thenh(w-14)=h(w):h(w-7)
=h(w)                              <fb>
1390 fori=zp-2to120+zp-2step40     <kb>
1400 gshapen$(val(q$)),ns,i,2      <gm>
1410 nexti                         <gm>
1420 ns=ns+10:an=ns:xy=xy+1        <ad>
1430 lo=0:return                   <le>
1440 gshapen$(val(q$)),ns,zp-2,2:x
y=1                                <ae>
1450 w=(zp-(z-1)*40-4)/2:ifval(q$)
=7thenh(w)=1:elseh(w)=-1:r=w       <nb>
1460 ifw<5thenh(w+7)=h(w):h(w+14)=
h(w):q=w+7:t=w+14                  <hm>
1470 ifw>4andw<12thenh(w+7)=h(w):s
=w+7                               <ga>
1480 ifw>7andw<15thenh(w-7)=h(w):u
=w-7                               <bl>
1490 ifw>14thenh(w-14)=h(w):h(w-7)
=h(w):v=w-14:o=w-7                 <gm>
1500 ns=ns+10:ifns>312thenns=an:z=
z+1                                <mo>
1510 ifz=5thenz=1:oo=1:gosub120    <mh>
1520 lo=0:return                   <cm>
1530 rem =lied spielen============  <nd>
1540 fori=1tonz                    <lg>
1550 wait65297,8:ifs(i)=0thenforp=
1tol(i):nextp:goto1570             <ef>
1560 sound1,s(i),l(i):sound2,s(i)+
2,l(i)                             <np>
1570 nexti                         <oo>
1580 return                        <pc>
1590 rem =aufloesungszeichen======  <ll>
1600 q=dec(q$):h=zp-2:xy=1         <fb>
1610 w=(zp-(z-1)*40-4)/2:a(w)=-h(w
):gosub2240                        <df>
1620 ifw<5thena(w+7)=-h(w):a(w+14)
=-h(w)                             <fe>
1630 ifw>4andw<12thena(w+7)=-h(w)  <nd>
1640 ifw>7andw<15thena(w-7)=-h(w)  <kp>
1650 ifw>14thena(w-7)=-h(w):a(w-14
)=-h(w)                            <pm>
1660 lo=0:return                   <gp>
1670 rem =taktstrich==============  <fl>
1680 h=(z-1)*40+20:q=dec(q$)       <kd>
1690 gosub2240:xy=1                <fe>
1700 fori=1to18                    <km>
1710 a(i)=0                        <fc>
1720 nexti                         <ei>
1730 ifq<>0thenh(q)=0:q=0          <ip>
1740 ifr<>0thenh(r)=0:r=0          <oh>
```

```
1750 ifs<>0thenh(s)=0:s=0                <cm>
1760 ift<>0thenh(t)=0:t=0                <dp>
1770 ifo<>0thenh(o)=0:o=0                <lm>
1780 ifu<>0thenh(u)=0:u=0                <km>
1790 ifv<>0thenh(v)=0:v=0                <ac>
1800 lo=0:return                         <ig>
1810 rem =notenzahl ueberschritten       <hp>
1820 graphic0,1                          <gj>
1830 printcu$cu$cu$tab(3)"mehr als
 500 toene gehen nicht."                 <mc>
1840 printtab(3)cu$"bitte melodie
speichern und neu"                       <ge>
1850 printtab(3)cu$"dimensionieren
 (zeile 2300)"                           <fp>
1860 printtab(3)cu$cu$cu$"taste dr
uecken"                                  <on>
1870 getkeyx$:graphic1,0:return          <ig>
1880 rem =note loeschen==========        <ek>
1890 iflo=1thenreturn                    <ob>
1900 ifxy>0andnz=0then1970               <of>
1910 ifxy>0andnz>0thenxy=0:goto193
0                                        <ho>
1920 nz=nz-1:ifnz<0thennz=0              <ch>
1930 ns=ns-10:ifns<anthenns=312:z=
z-1:ifz<1thenz=1:ns=an                   <dj>
1940 h=(z-1)*40+8                        <oe>
1950 gosub2180:gshapecn$,ns,pa-6         <jo>
1960 lo=1:return                         <md>
1970 h=(z-1)*40                          <io>
1980 ns=ns-10:an=an-10:ifns<12then
ns=12:an=12:xy=xy-1                      <hd>
1990 ifxy<0thenxy=0                      <fc>
2000 fori=htoh+120step40                 <mm>
2010 gshapecn$,ns,i                      <ge>
2020 nexti                               <pl>
2030 lo=1:return                         <da>
2040 rem =wiederholungszeichen====       <md>
2050 ifnz>500thengosub1820:return        <of>
2060 h=int(zp/42)                        <gj>
2070 h=40*h+20:q=val(q$)                 <dd>
2080 gosub2240:xy=1                      <pf>
2090 ifq$="0"thenreturn                  <ho>
2100 g=we-wa                             <ga>
2110 ifg+nz>500thengosub1820:retur
n                                        <bm>
2120 fori=we+1towe+g                     <ic>
2130 s(i)=s(i-g):l(i)=l(i-g)             <dd>
2140 nexti:nz=nz+g                       <ci>
2150 wa=0:we=0                           <en>
2160 return                              <ho>
2170 rem =pfeil positionieren=====       <jf>
2180 pa=(z-1)*40+6                       <ac>
2190 pe=z*40                             <bm>
2200 ifzp>pethenh=zp:zp=pe               <mj>
2210 ifzp<pathenh=zp:zp=pa               <bc>
2220 gshapecl$,0,h-4:gshapep$,0,zp       <fi>
2230 return                              <al>
2240 gshapen$(q),ns,h,2:ns=ns+10:i
fns>312thenns=an:z=z+1                   <of>
2250 ifz=5thenz=1:oo=1:gosub120          <ck>
2260 xy=0:return                         <ai>
2270 rem =pausen setzen==========        <kj>
2280 q=asc(q$)-55                        <dn>
2290 pp=2↑(16-(q-4))                     <ln>
2300 h=(z-1)*40-20*(pp=256)-21*(pp
=128)-20*(pp=64)-20*(pp=32)-20*(pp
=16)                                     <gb>
2310 gosub2240                           <kn>
2320 nz=nz+1:ifnz>500thengosub1820       <gf>
2330 s(nz)=0:l(nz)=3.5*pp+4*(q-8)+
20                                       <pa>
2340 lo=0:return                         <bo>
2350 rem =variablen==============        <ng>
2360 zp=6:nz=0:wa=0:we=0:ne=312:an
=12:ns=12:la=8                           <ae>
2370 dim s(500),l(500),n$(16),h(18
),a(18),t(18,2)                          <hm>
2380 cu$=chr$(17):co$=chr$(145)          <ko>
2390 fori=1to18                          <mg>
2400 forj=0to2                           <ng>
2410 readt(i,j)                          <aa>
2420 nextj,i                             <kk>
2430 char,1,1,"werden instruktione
n gewuenscht?(j/n)"                      <hm>
2440 getkeyx$                            <no>
2450 ifx$="n"thenreturn                  <bg>
2460 ifx$<>"j"then2430                   <bg>
2470 scnclr:char,1,1,"mit diesem p
rogramm koennen musik-"                  <kk>
2480 char,1,3,"stuecke von einem n
otenblatt abge-"                         <ef>
2490 char,1,5,"schrieben werden. m
it cursor up/down"                       <kf>
2500 char,1,7,"wird der pfeil (lin
ker rand) auf die"                       <cf>
2510 char,1,9,"entsprechende noten
linie gestellt."                         <dn>
2520 char,1,11,"mit '7' oder '8' w
erden die # oder b,"                     <id>
2530 char,1,13,"mit '1' - '6' die
notenwerte,"                             <kf>
2540 char,1,15,"mit '0' oder '9' d
ie wiederholungen"                       <in>
2550 char,1,17,"mit 'a' das aufloe
sungszeichen"                            <hh>
2560 char,1,19,"mit 'b' die taktst
riche und mit"                           <hb>
2570 char,1,21,"'c' - 'g' die paus
en eingegeben."                          <nm>
2580 char,10,24,"taste druecken"         <lg>
2590 getkeyyx$                           <bm>
2600 scnclr:char,1,1,"weitere tast
enbedeutungen:"                          <cm>
2610 char,1,3,"'l'= lied laden; 's
'= lied speichern"                       <ih>
2620 char,1,5,"'h'= programm beend
```

```
en"                                          <pc>
2630 char,1,7,"'n'= speicher loesc
hen, neuanfang"                              <hp>
2640 char,1,9,"'p'= lied spielen"            <bp>
2650 char,1,11,"'+' verlaengert di
e tondauer um die"                           <kg>
2660 char,5,13,"haelfte"                     <jb>
2670 char,1,15,"'-' loescht die le
tzte eingabe"                                <kh>
2680 char,1,17,"der taktstrich ist
 stets zu setzen,"                           <ak>
2690 char,1,19,"wenn auch im musik
stueck ein solcher"                          <gg>
2700 char,1,21,"erscheint (senkrec
hter strich ueber"                           <kk>
2710 char,1,23,"5 notenlinien)"              <ij>
2720 char,10,24,"taste druecken"             <cc>
2730 getkeyyx$                               <ka>
2740 scnclr:char,1,1,"die noten, d
ie zwischen den wiederho-"                   <af>
2750 char,1,3,"lungszeichen stehen
, werden doppelt"                            <hd>
2760 char,1,5,"gespeichert. wenn n
ur das wiederho-"                            <hb>
2770 char,1,7,"lungszeichen '9' ge
setzt ist, wird"                             <mc>
2780 char,1,9,"das stueck vom anfa
ng bis zu diesem"                            <hb>
2790 char,1,11,"wiederholungszeich
en doppelt ge-"                              <kj>
2800 char,1,13,"speichert."                  <be>
2810 char,1,15,"zusaetzliche # ode
r b gelten nur im"                           <bj>
2820 char,1,17,"takt, das aufloesu
ngszeichen 'a'"                              <al>
2830 char,1,19,"ebenfalls. # oder
b, die zu beginn"                            <bk>
2840 char,1,21,"geschrieben wurden
,gelten fuer das"                            <hn>
2850 char,1,23,"ganze stueck."+b$+
" taste druecken"                            <ic>
2860 getkeyx$:return                         <mh>
2870 input#15,aa,bb$,cc,dd                   <el>
2880 ifaa<>63then2950                        <ag>
2890 ifaa=63thenprintcu$tab(3)na$"
 existiert"                                  <dd>
2900 printcu$tab(3)"soll ueberschr
ieben werden? (j/n)                          <jn>
2910 getkeyx$:ifx$="n"thenu=1:clos
e15:close2:return                            <mn>
2920 ifx$<>"j"then2910                       <ef>
2930 na$="@0:"+na$:close15:close2            <co>
2940 u=2:return                              <jl>
2950 ifaa=0thenreturn                        <bd>
2960 printaa;bb$,cc,dd:close2:clos
e15:u=3                                      <bb>
2970 print"taste druecken"                    <lc>
2980 getkeyx$:return                         <on>
```

```
2990 rem =notendatas============== <ee>
3000 data911,917,923                 <jm>
3010 data904,911,917                 <kb>
3020 data889,897,904                 <jc>
3030 data872,881,889                 <mj>
3040 data854,864,872                 <fn>
3050 data844,854,864                 <fj>
3060 data822,834,844                 <dg>
3070 data798,810,822                 <ej>
3080 data784,798,810                 <ab>
3090 data754,770,784                 <ib>
3100 data721,739,754                 <mg>
3110 data685,704,721                 <jc>
3120 data664,685,704                 <bf>
3130 data620,643,664                 <mn>
3140 data571,596,620                 <eg>
3150 data544,571,596                 <fm>
3160 data505,516,544                 <ic>
3170 data418,453,485                 <oi>
3180 rem aenderungen fuer datasett
enbetrieb                            <li>
3190 rem streiche 1050 - 1070        <on>
3200 rem streiche 1200 - 1240,2870
 - 2980                              <bd>
3210 rem aendere: 1090 open2,1,0,n
a$                                   <pf>
3220 rem aendere: 1200 open2,1,1,n
a$                                   <pa>
3230 rem aendere: 1300 close2        <mk>
3240 rem nachspann================== <pd>
3250 b$=chr$(32):b2$=b$+b$           <li>
3260 b$=b2$+b2$:b$=b$+b$+b2$         <ji>
3270 return                          <dd>
3280 rem ========================== <jp>
3290 rem 60671 bytes memory          <oe>
3300 rem 08852 bytes program         <eg>
3310 rem 00203 bytes variables       <ji>
3320 rem 05580 bytes arrays          <ka>
3330 rem 00659 bytes strings         <bl>
3340 rem 12288 bytes graphic         <ha>
3350 rem 33089 bytes free (0)        <eh>
3360 rem ========================== <di>
```

## Musik

*Fortsetzung von Seite 91*

Wenn zur Speicherung keine Diskette vorhanden ist, kann mit Datasette mit folgenden Programmänderungen gearbeitet werden:
Streiche Zeilen 1050–1070
Ändere: 1090 open2,1,0,na$

Streiche Zeilen 1200–1240 und 2870–2980
Ändere: 1200 open2,1,1,na$
Ändere: 1300 close2

Grundsätzliches zur Eingabe: Notenlinien abzählen, Pfeil am linken Rand auf die entsprechende Notenlinie setzen und eine der Tasten 1 bis 6, 7, 8, A oder C bis G drücken.
Für die Eingabe 0, 9, B, „+" oder „–" ist die Stellung des Pfeiles gleichgültig, ebenso für F, H, L, S, P, N.

```
10 rem datamaker musik==========p4 <mp>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by reiner hickel        <hg>
50 rem                             <pd>
60 rem basic v3.5                  <od>
70 rem plus4 (c16/116 + 64 kb)     <cd>
80 rem floppy (kassette anpassbar) <nb>
90 rem ========================== <jg>
100 graphic1,1:graphic0            <oa>
110 scnclr:input"name des liedes";
a$                                 <ka>
120 open2,8,2,a$+",s,r"            <jf>
130 input#2,nz                     <nb>
140 x=int(nz/3)                    <kk>
150 ifnz-3*x=0thenh=0:goto170      <bo>
160 if(nz+1)/3=int((nz+1)/3)thenh=
1:elseh=2                          <hm>
170 dims(nz+h),l(nz+h)             <jb>
180 fori=1tonz                     <pj>
190 input#2,s(i),l(i)              <ai>
200 nexti                          <gp>
210 close2:ifh=0then250            <bm>
220 fori=nz+1tonz+h                <bf>
230 s(i)=0:l(i)=0                  <bm>
240 nexti                          <ap>
250 nz=nz+h                        <kg>
260 ad=4100                        <gf>
270 fori=1tonz                     <ek>
280 s1=int(s(i)/256):s2=s(i)-s1*25
6                                  <kf>
290 l1=int(l(i)/256):l2=l(i)-l1*25
6                                  <ed>
300 pokead,s1:pokead+1,s2:pokead+2
,l1:pokead+3,l2                    <lj>
310 ad=ad+4                        <dk>
320 nexti:ed=4100+nz*4-1:ad=4100   <km>
330 input"beginn der datazeilen";d
a                                  <df>
340 input"schrittweite";sw         <ep>
350 scnclr:print"370data";ad;",";e
d;",";da;",";sw:print"goto380"     <bm>
360 poke1319,19:poke1320,13:poke13
21,13:poke239,3                    <nl>
370 rem =reservierte zeile======== <he>
380 restore370:x$=""               <pc>
390 readad,ed,da,sw                <pn>
400 ifad-1=edthenscnclr:printda"da
ta0,0":poke1319,19:poke1320,13:pok
e239,2:end                         <bn>
410 forj=1to3                      <ik>
420 s=peek(ad)*256+peek(ad+1)      <mo>
430 l=peek(ad+2)*256+peek(ad+3)    <hn>
440 x$=x$+str$(s)+","+str$(l)      <mj>
450 ifj<3thenx$=x$+","             <mo>
460 ad=ad+4                        <hn>
470 nextj:nd=da+sw                 <la>
480 scnclr:print"370data"ad","ed",
"nd","sw:printstr$(da)"data"x$:pri
nt"goto380"                        <lm>
490 poke1319,19:poke1320,13:poke13
21,13:poke1322,13:poke239,4:end    <ai>
500 rem um die datas dann zu verwe
nden                               <hg>
510 rem ist am besten folgendes pr
ogramm                             <jb>
520 rem zu verwenden wegen der pau
sen:                               <nf>
530 rem 10 la=8: rem (lautstaerke) <km>
540 rem 20 restore:rem dataanfang
           lied                    <oi>
550 rem 30 volla:do:reads,l:getx$:
           ifx$<>""thenexit        <ii>
560 rem 40 ifs=0andl>0thenfori=1to
           l:nexti:goto70          <of>
570 rem 50 ifs=0andl=0then exit    <pb>
580 rem 60 sound1,s,l              <pj>
590 rem 70 wait65297,la:loop:retur
           n(hauptprogramm)        <ed>
600 rem la im waitbefehl entsprich
t der vol-einstellung              <ig>
610 rem die anweisung wartet, bis
der ton zu ende gespielt ist       <io>
620 rem zeile 370 ist die fuer das
 programm wichtige datazeile       <di>
630 rem aenderungen fuer datasette
nbetrieb                           <li>
640 rem aendere: 110 open2,1,0,na$ <pm>
650 rem ========================== <ge>
660 rem = achtung !!!            = <al>
670 rem = programm darf nicht    = <db>
680 rem = umnummeriert werden!   = <jg>
690 rem ========================== <cn>
700 rem  p r o g r a m m e n d e   <ao>
710 rem ========================== <ne>
```

## 2. Datamaker Musik

**Dokumentation:**

Zeilen 100–210 Laden des Liedes von D.

Zeilen 220–320 Schreiben der Noten mit Tondauer in den Speicher DEC 4100–6100

Zeilen 330–360 Eingabe der ersten Zeilennnummer der Data und Schrittweite. (Diese Daten werden in Zeile 370 als Arbeitsdatazeile übernommen.)

Zeilen 380–490 Umwandeln der Lieddaten in Datazeilen.

Zeilen 500–640 Anmerkungen mit kurzem Programm zum Abspielen der Melodie mit Ausstiegmöglichkeit.

Pausen erscheinen in den Data mit Notenwert 0 und Tondauer größer als 96.

Nachdem die Datas geschrieben sind, DELETE –710 eingeben, dann stehen nur noch die Datazeilen im Speicher.

Bitte beachten, daß der Grafikmodus eingeschaltet ist und der BASIC-Anfang bei Hex $4000 liegt. □

# Die Schildkröte in deutsch

**Ein ganz besonderer Leckerbissen ist das Programm Igel-Grafik, eine Programmiersprache zur Erstellung von Grafiken.**

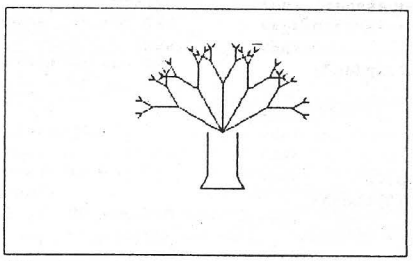
Der Igel ist ein kybernetisches Tier, das auf dem Bildschirm lebt und Befehle ausführt, die ihn zum Laufen (vorwärts oder rückwärts) oder zum Drehen um die eigene Achse (rechts oder links) veranlassen. Bei der Bewegung zeichnet der Igel seine Spur auf den Bildschirm. Er kann also dazu benutzt werden, Zeichnungen darauf zu erzeugen.
Igelprogramme werden gebildet durch Zusammenfassen von Grundbefehlen zu sogenannten Prozeduren; diese Programme können wieder in anderen Prozeduren verwendet werden, so daß schrittweise bis zu einem beliebigen Grad an Komplexität aufgebaut werden kann. Jeder einzelne Schritt kann aus einem Grundbefehl oder aus einer vom Benutzer entwickelten Prozedur bestehen.
Alle Prozeduren, ob als Grundbefehl vorhanden oder vom Benutzer definiert, können ausgeführt werden, indem ihre Namen einfach mit der Tastatur eingegeben werden. Der im Programm integrierte Editor macht es leicht, Prozeduren zu definieren, zu verändern und ausfzuführen.

## PROGRAMMSTART

Etwas Geduld ist nach dem Programmstart erforderlich, bis die Datazeilen der Maschinenroutine eingelesen sind. Alle für das Hauptprogramm später nicht mehr benötigten BASIC-Zeilen werden durch eine Maschinenroutine gelöscht. Das schafft Platz im Speicher.

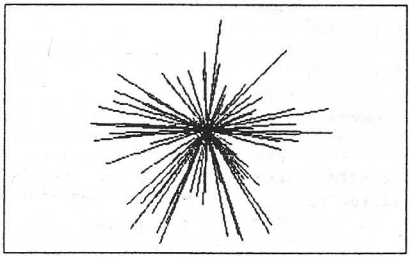
## HARDWARE

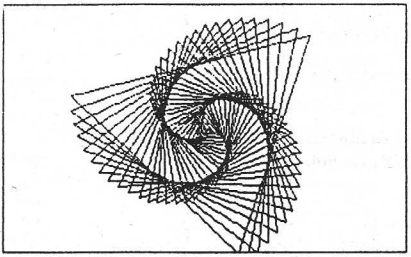
- Computer Plus/4 oder C16/116 mit 64 K-Byte.
- Floppy.



```
dra

pr zufall
var.laenge+
var.winkel+
wh 100
zz-1 100
vw
rw
zz-w 360
re
endwh
ende 1
```





```
wiederhole 100   laenge+ 1
vi               poly
winkel+ 88       db
```

- Drucker Epson-kompatibel für den Ausdruck von Grafiken; für den Ausdruck von Prozeduren reicht auch ein anderer Drucker. Durch Benützung einer geeigneten Hardcopy-Routine können die Grafiken aber auch mit jedem sonstigen Drucker zu Papier gebracht werden.

Auch wer nur eine Datasette besitzt, kann mit der Igel-Grafik experimentieren. Allerdings sind die Möglichkeiten sehr beschränkt, wenn auf das Laden und Speichern von Prozeduren und Bildern verzichtet werden muß.

## BEFEHLSLISTE IM DIREKTMODUS

**Funktionstasten**
F1 voller Grafikbildschirm;
F2 neun Zeilen Text;
F3 fünf Zeilen Text;
F5 Neustart Igelprogramm;
F8 voller Textbildschirm.
Die Funktionstasten F1, F2, F3 und F8 sind interruptprogrammiert. Daher ist ihre Bedienung jederzeit im laufenden Programm möglich.

**Grundbefehle**
**ade**
Alle in das Programm aufgenommenen Prozeduren werden gelöscht. Hierzu muß das Programm DELETE auf der Diskette vorhanden sein.

**basic**
Prozeduren werden im BASIC-Editor gezeigt. Zu sehen ist nicht die eingegebene Form, sondern wie diese in BASIC übersetzt wurde. Editieren ist möglich.

**zeige.titel (zt)**
Alle Namen der aufgenommenen Prozeduren werden gelistet.

**versteckigel (vi)**
Der als Dreieck auf dem

Bildschirm sichtbare Igel wird unsichtbar. Programme können dadurch schneller abgearbeitet werden.

**zeigigel**
Der unsichtbar gemachte Igel wird wieder sichtbar.

**igeltext [1]**
Der danach eingegebene Text wird auf den Grafikbildschirm ausgegeben. Folgt dem Befehl igeltext der Parameter 1, so wird der auszugebende Text revers dargestellt.

**wiederhole (wh) <Anzahl>**
Durch diesen Befehl, dem eine Zahl nachfolgen muß, wird die Anzahl der Wiederholungen bestimmt.

**fuelle**
Eine geschlossene Fläche, innerhalb derer sich der Igel befindet, wird mit der Zeichenfarbe gefüllt.

**inhalt (ih)**
Das Disketten-Directory wird aufgelistet.

**vorwaertssh (vwsh) <Länge>**
Vorwärts ohne Spur.

**rueckwaertssh (rwsh) <Länge>**
Rückwärts ohne Spur.

**ls**
Textbildschirm wird gelöscht.

**rueckwaerts (rw) <Länge>**
(selbsterklärend)

**vorwaerts (vw) <Länge>**
(selbsterklärend)

**rechts (re) <Winkel>**
(selbsterklärend)

**links (li) <Winkel>**
(selbsterklärend)

**invert (i)**
Grafikbildschirm wird invertiert.

**xko**
Ausgabe der X-Koordinate der Igelposition.

**yko**
Ausgabe der Y-Koordinate der Igelposition.

**schiebbild <Schirmnummer>**
Verschieben eines Bildes nach Schirm zwei oder drei (insgesamt drei Grafikbildschirme).

**holbild <Schirmnummer>**
Holen eines Bildes von Schirm zwei oder drei (Weiterbearbeitung möglich).

**zeigbild <Schirmnummer>**
Bild zeigen auf Schirm eins, zwei oder drei.

**winkel= <Größe>**
Winkelgröße setzen.

**winkel+ <Zuwachs>**
Zuwachs für den Winkel setzen.

**laenge+ <Zuwachs>**
Zuwachs für die Länge setzen.

**A. Befehlsliste**

| | |
|---|---|
| warte <zahl> | Programmverzögerung |
| stifthoch (sh) | |
| stiftab (sa) | |
| zeigigel (zi) | wie Direktmodus |
| versteckigel (vi) | wie Direktmodus |
| zufallszahl–l (zz-l) <Bereich> | zufällige Igelschrittlänge erzeugen |
| zufallszahl–w (zz-w)<Bereich> | zufälligen Igeldrehwinkel erzeugen |
| aufkurs (ak) | wie Direktmodus |
| igeltext | wie Direktmodus |
| fuelle | wie Direktmodus |
| ls | Textschirm löschen, Prozedureneingabe |
| vorwaerts (vw) <Länge> | wie Direktmodus |
| rueckwaerts (rw) <Länge> | wie Direktmodus |
| links (li) <Winkel> | wie Direktmodus |
| rechts (re) <Winkel> | wie Direktmodus |
| stop.wenn.winkel> (sww>) | Abbruchbedingung |
| stop.wenn.winkel< (sww<) | Abbruchbedingung |
| stop.wenn.winkel= (sww=) | Abbruchbedingung |
| stop.wenn.laenge> (swl>) | Abbruchbedingung |
| stop.wenn.laenge< (swl<) | Abbruchbedingung |
| stop.wenn.laenge= (swl=) | Abbruchbedingung |
| var.laenge+ | Länge Igelschritt variabel wählen |
| var.laenge* | Länge Igelschritt variabel wählen |
| var.winkel+ | Drehwinkel Igel variabel wählen |
| var.winkel* | Drehwinkel Igel variabel wählen |
| winkel+ | Winkelzuwachs wählen |
| winkel* | Winkelzuwachs wählen |
| laenge+ | Längenzuwachs wählen |
| laenge* | Längenzuwachs wählen |
| wiederhole <Anzahl> | Anzahl der Wiederholungen bestimmen. Ist keine Anzahl angegeben, so wird diese vor dem Prozeduraufruf bestimmt. |
| ende.wiederhole | Schleifenabschluß |
| mitte | Igel geht zur Mitte und hinterläßt eine Spur |
| loeschebild | wie Direktmodus |
| pause | Programmunterbrechung. Fortsetzung auf Tastendruck |
| laenge= <Zahl> | Schrittlänge setzen |
| winkel= <Winkel> | Drehwinkel setzen |
| aufx <Koordinate> | Igel auf X-Position setzen |
| aufy <Koordinate> | Igel auf Y-Position setzen |
| radiere | Igel löscht Spur, die er überfährt |
| farbe <Nummer> | Spurfarbe wählen |
| luminanz <Zahl> | wie Direktmodus |
| rahmen <Zahl> | wie Direktmodus |
| ende <Zahl> | Prozedurennummer wählen |

radiere
Pixel löschen.

luminanz (lu) <Größe>
Farbhelligkeit setzen.

farbe <Code>
Zeichenfarbe wählen.

rahmen <Code>
Rahmenfarbe wählen.

aufxy
X- und Y-Koordinaten für Igelposition setzen. Der Rechner fragt anschliessend nach den Koordinatenwerten. Es ändert sich nur die Position, nicht aber der Kurs des Igels.

kurs
Ausgabe des Igelkurses (Winkel 0 bis 360).

aufkurs (ak) <Winkel>
Igelkurs setzen.

vergiss (vg) <Prozedurnummer>
Prozedur aus dem Arbeitsspeicher löschen.

loeschefile <Name>
File auf Diskette löschen.

druckebild (db)
Grafik drucken von Schirm eins. Nur mit Epson-kompatiblem Drucker möglich.

druckerein (dre)
Protokoll ein. Mit jedem Drucker möglich.

druckeraus (dra)
Protokoll aus.

ladebild [1] <Name>
Grafik von Diskette laden. Folgt dem Befehl der Parameter 1, so werden nur 33 Blocks für die reine Bildinformation geladen. Wird der Parameter 1 weggelassen, so werden 41 Blocks geladen, die auch die Farbinformation enthalten.

bewahrebild [1] <Name>
Grafik auf Diskette speichern. (33 Blocks bei Parameter 1, sonst 41 Blocks.)

lade <name>
Prozeduren laden (Paket). Die Prozeduren müssen bereits in BASIC übersetzt vorliegen.

bewahre <name>
Prozeduren auf Diskette sichern. Alle im Arbeitsspeicher vorliegenden Prozeduren werden als Prozedurenpaket gespeichert und können zu einem späteren Zeitpunkt mit dem Lade-Befehl wieder eingeladen werden.

loeschebild (lb)
Grafikbildschirm löschen. Die Igelposition bleibt erhalten.

mitte
Igel in Bildschirmmitte setzen (ohne Spur).

bild
Grafikbildschirm löschen (Igel in Bildschirmmitte).

edit
Übergang in den Prozeduren-Editor.

ende
Programmabbruch.

**1. Im Direktmodus edit eingeben**

**2. Prozedurnamen wählen <RETURN>**
<ESC>-Taste bei Redigierwunsch bereits erstellter und auf Diskette gespeicherter Prozeduren, sonst <RETURN>. Prozeduren werden als sequentielle Files gespeichert und müssen in den Prozedurenspeicher, wie weiter unten beschrieben, eingelesen werden. Im Unterschied dazu werden Prozedurenpakete, die aus einer Vielzahl fertig eingespeicherter Prozeduren bestehen, durch den Befehl „lade" aus dem Direktmodus heraus geladen und mit „bewahre" gesichert.

**3. Befehle eingeben**
- Ein Befehl pro Zeile.
- Befehl und Parameterzahl immer durch ein Leerzeichen trennen, sonst erfolgt Fehlermeldung.
- Maximal neun Zeilen sind möglich, sonst erfolgt Fehlermeldung.
- Beliebiges Umherwandern und Verändern der Eingabe am Bildschirm ist möglich.
- Einfügen von Prozedurenzeilen mit <ESC> + <i>, löschen von Programmzeilen mit <ESC>+<d>

**4. ende <Prozedurennummer> eingeben (zum Beispiel ende 24)**
- Verschiedene Prozeduren müssen verschiedene Nummern haben, im Zweifelsfall durch zt im Direktmodus Nummern feststellen. Bei Nummerngleichheit wird das alte Programm überschrieben.
- Mit <RETURN> Menü aufrufen.

**5. Menü**
<CONTROL>+
<r> – Redigieren – unbedingt nötig, falls Prozedur nicht mit <CONTROL> + <b> abgespeichert wird.
<b> – Sichern – Prozedur wird als sequentielles File auf Diskette gesichert. Der Prozedurname wird übernommen, ein File des gleichen Namens überschrieben.
<c> – Einlesen – Prozedur wird in BASIC übersetzt und in den Prozedurenspeicher übernommen. Sie kann danach im Direktmodus zusammen mit weiteren Prozeduren als Prozedurenpaket abgespeichert werden.
<d> – Drucken – Prozedur wird auf den Drucker ausgegeben.
<p> – Plotten – Prozedur wird auf den Plotter VC 1520 ausgegeben.
<g> – Abbruch – Ausstieg aus dem Editor und Rückkehr in den Direktmodus.

Nach <CONTROL> + <c> und der Lernmeldung des Rechners kann die neue Prozedur mit ihrem Namen (und mit einer Längenangabe, falls die Länge variabel gewählt wurde) aufgerufen werden. Mit der STOP-Taste kann jederzeit unterbrochen werden.

**6. Drei Möglichkeiten, den Editor zu verlassen:**
a) vor Eingabe des Prozedurnamens: zweimal <RETURN> betätigen;
b) während des Schreibens der Prozedur: ls eingeben und zweimal <RETURN> drücken;
c) während des Schreibens der Prozedur: „ende" eingeben und <CONTROL> + <c> betätigen.

## TIPS ZU IGEL

1. Sollten Sie nach dem Befehl „lade" einmal ein Prozedurenpaket einlesen wollen, das sich nicht auf der eingelegten Diskette befindet, so werden Sie aus dem Programm hinauskatapultiert. Legen Sie dann die Diskette ein, auf der das File „delete" gespeichert ist und betätigen Sie die Funktionstaste F5.
2. Hindern Sie den Igel daran, eine Prozedur zu Ende zu bringen (mit der <STOP>-Taste jederzeit möglich), so ist manchmal ein zweimaliges Befehlen von „bild" nötig, um den Igel in die Ausgangsposition zu bringen.
3. Es ist nicht möglich, alle Fähigkeiten von Igel zu demonstrieren. Eigenes Experimentieren ist erwünscht und wird den Anwender bald perfekt machen. Außerdem muß man sich intensiv mit jedem Befehl beschäftigen.
4. Beim Sichern von Prozedurenpaketen empfiehlt sich die Kennung „.pkt", damit keine Verwechslungen stattfinden.
5. Wählen Sie das Programmpaket „demo.pkt"

□

```
10 rem igel=====================p4 <bp>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by k.-d. mueller        <ig>
50 rem                             <pd>
60 rem basic v3.5                  <od>
70 rem plus4 (c16/116 + 64 kb)     <cd>
80 rem disk + drucker (epson-komp. <mn>
90 rem ========================== <jg>
100 graphic1,1:poke55,0:poke56,160
:clr:graphic0:gosub2290            <he>
110 gosub61830:sys4795             <kf>
120 poke4796,112:poke4800,23:poke
4807,252:poke4811,83:sys4293:sys46
39:run1520                         <bn>
121 run1520                        <lc>
130 ed=1:iffa=0thengosub1390:retur
n                                  <gn>
140 dowhileb>32040:b=b-32040:loop:
return                             <eo>
150 x=rdot(0):y=rdot(1):sys50559:b
ox1,0,0,319,199:locate x,y: return <ol>
160 open 1,8,15:input#1,f,f$:iff<>
0thenprint:printf$:forf=1to1000:ne
xt:close1:run2050                  <je>
170 close1:return                  <hc>
180 printchr$(147)                 <on>
190 gosub2290:i=3:printrn$he$c4$"P
rozedurname"rf$;:inputc$           <mh>
200 poke 205,1:print:poke202,20:pr
intrn$fl$"<esc> zum Laden!!! "rf$f
o$;                                <kc>
210 getkeyv$:ifv$=chr$(27)thensys4
477,""+c$,3072,0,8:gosub160:printh
e$c4$c4$c4$:goto230                <hh>
220 poke205,1:print:poke202,20:pri
ntb$b5$b4$he$c4$c4$c4$"pr "+c$:ifc
$=""thenrun1520                    <ab>
230 dimle$(20):dimw1$(20):dimw$(20
):n=2:sn$="draw m to"              <el>
240 tu$=":go5130":ko$=":go5140"    <od>
250 va$="=c:":vc$="c=":gosub320    <ce>
260 va$="=z:":vc$=":z=":gosub320   <mj>
270 va$="=iw:":vc$=":iw=":gosub320 <fh>
280 va$="=zl:":vc$=":zl=":gosub320 <gd>
290 va$="=r:":vc$=":r=":gosub320   <op>
300 va$="=zw":vc$=":zw=":gosub320  <ae>
310 forla=1to6:bh$=bh$+bh$(la):br$
=br$+br$(la):next:goto360          <hf>
320 la=la+1:s3=int(rnd(0)*25)+65:s
4=int(rnd(0)*9)                    <bi>
330 bw$(la)=chr$(s3)+str$(s4)      <nc>
340 bh$(la)=bw$(la)+va$:br$(la)=vc
$+bw$(la)                          <ag>
350 return                         <em>
360 w$(i)=str$(i)+bh$:i=i+1:n=n+1  <gb>
370 trap2070:do:gosub1440:gosub599
0                                  <dh>
380 ifa$="warte"thenw1$(i)="for v=
1 to"+str$(a)+tu$+":next":goto840  <ee>
390 ifa$="stifthoch"or a$="sh"then
w1$(i)="rem**pu**":sn$="locate":e$
=a$:goto840                        <nk>
400 ifa$="stiftab"ora$="sa"thenw1$
(i)="rem**pd**":sn$="draw m to":e$
=a$:goto840                        <hm>
410 ifa$="zi"ora$="zeigigel"thenw1
$(i)="fa=0:ed=0":goto840           <km>
420 ifa$="zufallszahl-1"ora$="zz-1
"thenw1$(i)="c=int(rnd(1)*"+str$(a
)+")+1":goto840                    <jk>
430 ifa$="zufallszahl-w"ora$="zz-w
"then w1$(i)="iw=int(rnd(1)*"+str$
(a)+")+1":goto 840                 <no>
440 ifa$="aufkurs"ora$="ak"thenw1$
(i)="b="+str$(a):goto840:elseifa$=
"G"thenrun1520                     <nm>
450 ifa$="vi"ora$="versteckigel"th
enw1$(i)="fa=1":goto840            <nk>
460 ifa$="igeltext"thengosub 1360:
goto840                            <ah>
470 ifa$="fuelle"thenw1$(i)="locat
e4;5:paint,rdot(0),rdot(1)":goto84
0                                  <ho>
480 ifa$="invert"ora$="i"thenw1$(i
)="m=-(m=0):sys 4388":goto 840     <cm>
490 ifa$="ls"thenrun180            <jj>
500 ifa$="vorwaerts"ora$="vw"thenw
1$(i)=sn$+str$(a)+f$+";b"+tu$:goto
840                                <dc>
510 ifa$="rueckwaerts"ora$="rw"the
nw1$(i)=sn$+"-("+str$(a)+f$+");b"+
tu$:goto840                        <oj>
520 ifa$="stop.wenn.winkel="ora$="
sww="thenw1$(i)=":do until iw="+st
r$(a):goto840                      <mj>
530 ifa$="stop.wenn.winkel>"ora$="
sww>"thenw1$(i)=":do until iw>"+st
r$(a):goto840                      <aa>
540 ifa$="stop.wenn.winkel<"ora$="
sww<"thenw1$(i)=":do until iw<"+str
r$(a):goto840                      <bn>
550 ifa$="winkel="thenw1$(i)="iw="
+str$(a):goto 840                  <oi>
560 ifa$="var.winkel+"thenw1$(i)="
:":iw$="+iw":goto840               <fa>
570 ifa$="var.winkel*"thenw1$(i)="
:":iw$="*iw":goto840               <mb>
580 ifa$="rechts"ora$="re"thenw1$(
i)="w="+str$(a)+iw$+":b=b+w"+ko$+t
u$:goto840                         <le>
590 ifa$="links"ora$="li"thenw1$(i
)="w=360-("+str$(a)+iw$+"):b=b+w"+
ko$+tu$:goto840                    <ik>
600 ifa$="winkel+"thenw1$(i)="iw=i
w+"+str$(a)+"+zw":goto840          <af>
```

```
610 ifa$="winkel*"thenw1$(i)="iw=i
w*("+str$(a)+"+zw)":goto840          <ok>
620 ifa$="wiederhole"ora$="wh"then
w1$(i)="s1=r:s2=z:z=0:r="+str$(a)+
"+vr:do":goto840                     <om>
630 ifa$="stop.wenn.laenge="ora$="
swl="thenw1$(i)=":do until c="+str
$(a):goto840                         <gg>
640 ifa$="stop.wenn.laenge>"ora$="
swl>"thenw1$(i)=":do until c>"+str
$(a):goto840                         <aj>
650 ifa$="stop.wenn.laenge<"ora$="
swl<"thenw1$(i)=":do until c<"+str
$(a):goto840                         <cb>
660 if a$="ende.wiederhole"ora$="e
ndwh"then1070                        <bb>
670 ifa$=c$thenw1$(i)="loop":goto8
40                                   <ea>
680 ifa$="laenge+"thenw1$(i)="c=c+
"+str$(a)+"+zl":goto840              <no>
690 ifa$="laenge*"thenw1$(i)="c=c*
("+str$(a)+"+zl)":goto840            <ph>
700 ifa$="var.laenge+"thenw1$(i)="
:":f$="+c":goto840                   <la>
710 ifa$="var.laenge*"thenw1$(i)="
:":f$="*c":goto840                   <af>
720 if a$="mitte"thenw1$(i)=sn$+"1
60,100:b=0:w=0":goto840              <ki>
730 ifa$="loeschebild"ora$="lb"the
nw1$(i)="goS150":goto840             <db>
740 ifa$="pause"thenw1$(i)="getkey
 x$":goto840                         <jh>
750 ifa$="laenge="thenw1$(i)="c="+
str$(a):goto840                      <pj>
760 ifa$="aufx"thenw1$(i)=sn$+"160
+"+str$(a)+",rdot(1)"+tu$:goto840    <ie>
770 ifa$="aufy"then w1$(i)=sn$+" r
dot(0),100-"+str$(a)+tu$:goto840     <il>
780 ifa$="radiere"thenw1$(i)="m=0"
:goto840                             <lf>
790 ifa$="farbe"thenw1$(i)="m=1:co
lor1,"+str$(a)+",lu":goto840         <ng>
800 ifa$="rahmen"thenw1$(i)="colo"
+z3$+"4,"+str$(a)+",lu":goto840      <bi>
810 ifa$="luminanz"ora$="lu"thenw1
$(i)="lu="+str$(a):goto840           <mc>
820 ifa$="ende"theni=i+1:n=n+1:got
o 1230                               <am>
830 gosub1200:goto850                <nn>
840 w$(i)=str$(i)+w1$(i):i=i+1:n=n
+1                                   <ba>
850 loop                             <jg>
860 printcl$:sys4940                 <nm>
870 ifa<=0ora>50then2050             <pm>
880 k=a:a=10000+(k-1)*20:d=a-4000:
e=a-3000                             <lc>
890 iffl=0thengoto930                <bd>
900 printc4$c4$"delete"d:printc4$c
4$"delete"e:printc4$c4$"delete"d+1
4000                                 <de>
910 printc4$c4$"delete"a"-"a+19:pr
intc4$c4$"rU1520":printhe$           <co>
920 poke 239,5:fori=0to4:poke1319+
i,13:next:end                        <ce>
930 fori=3ton-1:printstr$(a+i-3)+m
id$(w$(i),-(i>9)+3,len(w$(i))):nex
t                                    <mf>
940 printstr$(a-3+n)+br$+":reT:rem
*"+c$+"*"                            <ng>
950 printstr$(e)+"ifa$="+chr$(34)+
c$+chr$(34)+"thEc=a:goS"+str$(a)+"
:w$(i)=str$(i)+";                    <ln>
960 printchr$(34)+"gosub"+str$(a)+
chr$(34)+":i=i+1:goto2120"           <ng>
970 printstr$(d)"ifa$="chr$(34)c$c
hr$(34)"thEw1$(i)="chr$(34)":goS"s
tr$(a)chr$(34);                      <ke>
980 print":g0840"                    <no>
990 print"1050 df$="chr$(34)c$chr$
(34)                                 <ea>
1000 printstr$(d+14000)":"k"."b2$c
$:print"goto1040"                    <mn>
1010 if peek(205)<23then1030         <nf>
1020 sys4949:printcl$left$(qr$,5)l
eft$(qd$,7)rn$"zu viele Zeilen!!!"
:getkeyze$:goto2050                  <hb>
1030 poke1319,19:poke1320,17:fori=
1ton+3:poke1320+i,13:next:poke239,
n+5:end                              <nh>
1040 gosub2290:printcl$:sys4949      <gn>
1050 df$="chaos"                     <pm>
1060 printleft$(qr$,4)left$(qd$,4)
rn$df$rf$b3$"gelernt":fori=1to600:
next:run 1520                        <jf>
1070 w1$(i)=+"z=z+1:loOuNz>=r:r=s1
:z=s2":goto840                       <id>
1080 printrn$"x"rf$;:inputx3:print
rn$"y"rf$;:inputy3                   <db>
1090 printrn$"sa (0) oder sh (1)"r
f$;:inputpu                          <ah>
1100 ifx3=1111thenx3=rdot(0):elsex
3=160+x3                             <bb>
1110 ify3=1111theny3=rdot(1):elsey
3=100-y3                             <en>
1120 ifputhenlocatex3,y3:elsedrawm
tox3,y3                              <cb>
1130 goto 2120                       <dn>
1140 :                               <ck>
1150 p%=bl                           <bf>
1160 ifa=8thenscratch""+c$:sys4462
,""+c$,3072,bl*40+3072,a             <ji>
1170 ifa=8thenfory=0tobl:poke1319+
y,13:next:poke239,bl:run 190         <ee>
1180 if a=4thenpoke 4999,7:p%=p%-3
:sys4958,12,4,15,p%:a=s1:goto 1280   <jc>
1190 if a=6thenopen 6,6,6:print#6,
```

```
1:p%=p%-3:poke 4999,0:sys4958,12,6
,15,p%:a=s1:goto1240                    <he>
1200 iffa=0thengshapes$(b2),x1-5,y
1-5,4                                   <jk>
1210 printc2$rn$"unbekannte Prozed
ur"rf$;:forf=1to1000:next:printchr
$(27)+"p";                              <im>
1220 printchr$(27)+"j";:return          <pk>
1230 s1=a:b1=peek(205):printchr$(2
7)+"c";                                 <fb>
1240 forzz=1to37:printzv$;:next:pr
intzv$                                  <hn>
1250 print "Ihre Wahl: <ctrl>+....
.............":print                   <pc>
1260 printrn$"<c>:'Lernen'<g>:'Sto
ppen'<b>:'Sichern'"rf$:print            <ej>
1270 printrn$"<r>:'Redig.'<d>:'Dru
cker'<p>:'Plotter'"rf$                  <ac>
1280 getkeyct$:p1=asc(ct$)              <ig>
1290 ifp1=3thengoto860                  <ed>
1300 ifp1=7thenrun1520                  <np>
1310 ifp1=2thena=8:goto1140             <al>
1320 ifp1=4thena=4:goto1140             <mp>
1330 ifp1=16thena=6:goto1140            <jf>
1340 ifp1=18thenfory=0tob1:poke131
9+y,13:next:poke239,b1:run190           <db>
1350 goto 1280                          <oa>
1360 inputz$:w1$(i)="char,rdot(0)/
8+1,rdot(1)/8,"+chr$(34)+z$+chr$(3
4)+",#"+str$(a)                         <pf>
1370 return                             <em>
1380 graphichk,0:sys5376:return         <of>
1390 x1=rdot(0):y1=rdot(1)              <fo>
1400 ifb>=360thenb=b-360:goto1400       <of>
1410 b2=int(b/10):gshapes$(b2),x1-
5,y1-5,4                                <op>
1420 ifed=1thened=0:gshapes$(b2),x
1-5,y1-5,4:locatex1,y1                  <dl>
1430 return                             <mf>
1440 poke 19,1:inputa$:poke19,0         <ee>
1450 le=1:me$=a$                        <ao>
1460 if le>20 thena$=me$:a=0:goto1
490:elsele$(le)=mid$(a$,le,1)           <ch>
1470 ifle$(le)<>" "then le=le+1:go
to1460                                  <ib>
1480 a1$=right$(a$,len(a$)-le):a=v
al(a1$):a$=left$(a$,(le-1))             <gf>
1490 print:if dz=0thenreturn            <fn>
1500 ifa=0thenprint#3,a$:return         <ij>
1510 ifa<>0thenprint#3,a$;a:return      <pj>
1520 graphic2,1                         <ka>
1530 box0,0,0,319,199:dims$(36):sy
s4940:fornr=0to360step10:circle1,5
,5,4,5,,,nr,120                         <jl>
1540 gosub2290:sshapes$(nr/10),0,0
,10,10:sys4922:next:n=3:s=3:b=0:m=
1                                       <ib>
1550 dimw$(30):dimle$(20):trap2170      <ef>
1560 box 1,0,0,319,199:x1=160:y1=1
00:locatex1,y1:gshapes$(b2),x1-5,y
1-5,4:sys 4949                          <op>
1570 do:gosub1440:iffa=0thengshape
s$(b2),x1-5,y1-5,4:locate x1,y1         <fd>
1580 printbk$;:gosub6990                <oh>
1590 ifa$="ade"thensys4795:sys4660
:clr:pn$="delete":goto2220              <ki>
1600 if a$="basic"thengraphic0:got
o2080:elseifa$="ende"thengraphic0:
end                                     <dh>
1610 if a$="zeige.titel"ora$="zt"t
henpoke 22,35:list 19990-:poke22,2
5:goto2120                              <ao>
1620 ifa$="versteckigel"ora$="vi"t
hen:fa=1:goto2120                       <mm>
1630 ifa$="zeigigel"ora$="zi"thenf
a=0:ed=0:goto2120                       <ah>
1640 if a$<>"igeltext"then1660          <ef>
1650 printrn$+"Text"+rf$;:inputz$:
char,rdot(0)/8+1,rdot(1)/8,z$,a:go
to2120                                  <eb>
1660 ifa$="wiederhole"or a$="wh"th
envr=a:goto2120                         <fn>
1670 if a$="fuelle"thenlocate 4;5:
paint,rdot(0),rdot(1):goto 2120         <ob>
1680 ifnot(a$="inhalt"ora$="ih")th
en1700                                  <ak>
1690 graphic0:printcl$:directory:g
etkeyw$:printcl$:graphic2,0:poke20
5,19:print:goto2120                     <mg>
1700 ifa$="vorwaertssh"ora$="vwsh"
then locate a;b:goto2120                <fc>
1710 ifa$="rueckwaertssh"ora$="rws
h"thenlocate-a;b:goto2120               <nm>
1720 if a$="ls"thenprintcl$:poke20
5,19:print:goto 2120                    <li>
1730 ifa$="rueckwaerts"ora$="rw"th
endrawmto-a;b:goto2120                  <hd>
1740 ifa$="vorwaerts"ora$="vw"then
drawmtoa;b:goto2120                     <ej>
1750 ifa$="rechts"ora$="re"thenw=a
:b=b+w:goto2120                         <jf>
1760 ifa$="links"ora$="li"thenw=36
0-a:b=b+w:goto2120                      <og>
1770 if a$="invert"ora$="i"then m=
-(m=0):sys 4388:goto 2120               <mg>
1780 ifa$="xko"thengosub2160:print
b5$b4$c2$x1-160:goto 2120               <ki>
1790 ifa$="yko"thengosub2160:print
b5$b4$c2$100-y1:goto 2120               <nj>
1800 ifa$="schiebbild"thenifa>1the
nifa<4thenq=8192*(a+3):sys(5616)q,
32:goto 2120                            <ie>
1810 ifa$="holbild"thenifa>1thenif
a<4thenq=8192*(a+3):sys(5651)q,32:
goto 2120                               <ao>
1820 ifa$="zeigbild"thenifa>0theni
```

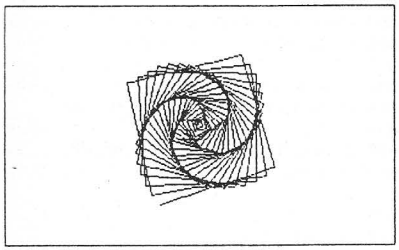
```
fa<4thenq=8*(a+3)+(a=1)*24:poke652
38,q:goto2120                          <bf>
1830 if a$="winkel=" then iw=a:got
o 2120                                 <da>
1840 ifa$="winkel+"thenzw=a:goto21
20:elseifa$="laenge+"thenzl=a:goto
2120                                   <pk>
1850 ifa$="radiere"thenm=0:goto212
0                                      <cd>
1860 ifa$="luminanz"ora$="lu"thenl
u=a:goto2120                           <el>
1870 ifa$="farbe"then m=1:color1,a
,lu:printbk$;:goto2120                 <dp>
1880 ifa$="rahmen"thencolor4,a,lu:
goto2120                               <oi>
1890 ifa$="aufxy"then1080              <gn>
1900 ifa$="kurs"thengosub2150:goto
2120:elseifa$="aufkurs"ora$="ak"th
enb=a:goto2120                         <gk>
1910 ifa$="vergiss"ora$="vg"thenif
a<>0thenfl=1:goto860                   <do>
1920 ifa$="loeschefile"theninputlf
$:sys 4940:scratch""+lf$:sys4949:g
oto2120                                <db>
1930 if a$="druckebild"ora$="db"th
enhk=2:sys4940:gosub1380:sys4949:g
oto2120                                <if>
1940 ifa$="druckerein"ora$="dre"th
enopen3,4,7:dz=1:goto2120              <oe>
1950 if a$="druckeraus"ora$="dra"t
hendz=0:close3:goto2120                <nl>
1960 ifa$="ladebild"ora$="bewahreb
ild"then goto 2240                     <ei>
1970 ifa$="lade"ora$="bewahre"then
2190:goto2120                          <hi>
1980 ifnot(a$="loeschebild"ora$="l
b")then2000                            <di>
1990 gosub2160:sys50559:box1,0,0,3
19,199:locatex1,y1:goto2120            <em>
2000 ifa$="mitte"thenlocate160,100
:b=0:w=0:goto2120                      <ne>
2010 ifa$="bild"thensys50559:b2=0:
b=0:box1,0,0,319,199:locate 160,10
0:goto2120                             <cb>
2020 ifa$="edit"thengraphic0:run 1
80                                     <mk>
2030 gosub1200                         <ga>
2040 loop                              <ol>
2050 color0,4:color4,4:color1,1:tr
ap2050:sys4949                         <io>
2060 run1520                           <bn>
2070 hk=1:gosub1380:goto2050           <eg>
2080 printcl$"prozedurnummer";:inp
utu:u1=9980+u*20:u2=10000+u*20-1:p
rintcl$"list"u1"-"u2                   <ln>
2090 poke 1319,19:poke 1320,13:pok
e 239,2:end                            <ea>
2100 run1520                           <bc>
2110 end                               <am>
2120 iffa=0thengosub1390:elsegosub
2160                                   <gh>
2130 n=n+1                             <hd>
2140 loop                              <le>
2150 rb=b:dountilrb<360:rb=rb-360:
loop:printb$c2$rb:return               <ho>
2160 x1=rdot(0):y1=rdot(1):return      <in>
2170 iffa=0thenifed=0thengshapes$(
b2),x1-5,y1-5,4                        <fm>
2180 resume1570                        <pd>
2190 ifa$="lade"thensys4795:sys466
0:clr:a$="lade"                        <gm>
2200 inputpn$:sys4940:ifa$="bewahr
e"thenscratch""+pn$                    <al>
2210 ifa$="bewahre"thensys4672,""+
pn$,5990,0,21500,8:run1520             <jm>
2220 sys4742,""+pn$,5990,8:sys4660
:poke1319,asc("z"):poke1320,asc("t
")                                     <gj>
2230 poke 1321,13:poke239,3:run152
0                                      <hd>
2240 inputpi$:sys 4940                 <pj>
2250 ifa$="ladebild"thensys4477,""
+pi$,6144-((a=1)*2048),0,8:sys4949
:gosub160:run1530                      <ln>
2260 sys 4462,""+pi$,6144-((a=1)*2
048),16384,8:sys4949:run1530           <do>
2270 rem nachspann ================    <nl>
2280 rem * farbcodes/steuercodes *     <lo>
2290 c4$=chr$(017):rn$=chr$(018)       <pe>
2300 he$=chr$(019):c3$=chr$(029)       <ol>
2310 f1$=chr$(130):fo$=chr$(132)       <mp>
2320 bk$=chr$(144):c2$=chr$(145)       <cc>
2330 rf$=chr$(146):c1$=chr$(147)       <ek>
2340 rem *** zeichensatz/graphik *     <fg>
2350 z3$=chr$(164):zv$=chr$(192)       <ck>
2360 rem ********* zeichenfolgen *     <hi>
2370 qd$="":qr$="":forzz=1 to 40       <al>
2380 qd$=qd$+c4$:qr$=qr$+c3$           <bj>
2390 nextzz                            <ap>
2400 b$=chr$(32):b2$=b$+b$             <fk>
2410 b3$=b2$+b$:b4$=b3$+b$             <pi>
2420 b5$=b4$+b$:b$=b5$+b5$             <ob>
2430 return                            <jn>
5990 rem**neue befehle(programmmod
us)***                                 <ba>
6989 return                            <fo>
6990 rem***neue befehle (direktmod
us)***                                 <oc>
7989 return                            <dg>
19990 rem*****befehlssatz*****        <fl>
60000 data00,00,00,58,00,00,00,00,
 88                                    <bp>
60010 data00,00,53,00,00,00,ad,09,
 265                                   <bm>
60020 dataff,29,02,f0,03,20,60,10,
 685                                   <oa>
```

```
60030 data2c,d8,07,10,0e,ad,01,fd,
 724                                <ce>
60040 data8d,d4,07,10,06,20,95,ea,
 797                                <ok>
60050 data20,5b,ea,20,e4,e3,ad,09,
1026                                <ok>
60060 dataff,29,02,f0,28,8d,09,ff,
 983                                <bl>
60070 data2c,0b,ff,a9,cc,50,1b,6c,
 898                                <hm>
60080 data12,03,20,bf,cf,20,cd,ce,
 894                                <nm>
60090 dataa5,fb,48,a9,00,85,fb,08,
1049                                <bm>
60100 data58,20,11,db,28,68,85,fb,
 884                                <he>
60110 dataa9,a1,8d,0b,ff,4c,be,fc,
1255                                <jd>
60120 dataad,1c,ff,29,01,d0,39,ad,
 936                                <kd>
60130 data1d,ff,c9,a4,b0,2e,24,83,
1038                                <pc>
60140 data50,52,a9,08,8d,14,ff,ad,
 928                                <oh>
60150 data06,ff,29,df,a8,ad,07,ff,
1128                                <pa>
60160 data29,ef,aa,ad,12,ff,0d,fa,
1159                                <di>
60170 data07,48,ad,1d,ff,c9,a4,90,
1045                                <mk>
60180 dataf9,68,8d,12,ff,8c,06,ff,
1168                                <el>
60190 data8e,07,ff,60,c9,cc,90,24,
1085                                <la>
60200 dataa6,83,f0,20,10,08,ad,07,
 773                                <kl>
60210 dataff,09,10,8d,07,ff,ad,06,
 862                                <jg>
60220 dataff,09,20,8d,06,ff,ad,12,
 889                                <kj>
60230 dataff,29,fb,8d,12,ff,ad,fb,
1385                                <ml>
60240 data07,8d,14,ff,60,78,a9,10,
 824                                <ek>
60250 data8d,13,03,8d,15,03,a9,d5,
 710                                <oh>
60260 data8d,14,03,58,60,a5,c6,c9,
 912                                <lc>
60270 data04,d0,0d,a9,06,8d,59,10,
 646                                <hf>
60280 dataa9,07,8d,6b,10,8d,8e,10,
 739                                <ml>
60290 datac9,05,d0,0d,a9,81,8d,59,
 955                                <je>
60300 data10,a9,84,8d,6b,10,8d,8e,
 864                                <fh>
60310 data10,c9,06,d0,0d,a9,a1,8d,
 915                                <cb>
60320 data59,10,a9,a4,8d,6b,10,8d,
 843                                <jb>
60330 data8e,10,c9,03,d0,11,a9,79,
 877                                <kn>
60340 data8d,59,10,a9,7c,8d,6b,10,
 803                                <bc>
60350 data8d,8e,10,a9,40,85,c6,4c,
 939                                <al>
60360 data0e,10,11,78,a9,00,85,e0,
 693                                <pm>
60370 dataa9,20,85,e1,a0,00,b1,e0,
1120                                <mp>
60380 data49,ff,91,e0,c8,d0,f7,e6,
1582                                <ej>
60390 datae1,a5,e1,c9,40,d0,ef,60,
1423                                <aj>
60400 data49,29,20,91,94,20,2c,93,
 662                                <nn>
60410 dataa5,62,85,af,a5,63,85,b0,
1144                                <if>
60420 dataa5,61,85,ab,20,91,94,20,
 923                                <al>
60430 data14,93,20,e4,9d,84,da,85,
1067                                <jf>
60440 datadb,20,91,94,20,d2,9d,a9,
1112                                <mb>
60450 data02,85,ac,86,ae,60,20,42,
 809                                <mp>
60460 data11,85,ad,a9,da,a6,14,a4,
1060                                <ei>
60470 data15,20,d8,ff,60,20,42,11,
 735                                <jj>
60480 dataa9,00,85,ad,a6,da,a4,db,
1242                                <ae>
60490 data20,d5,ff,60,20,42,11,a9,
 880                                <mc>
60500 data01,85,ad,a9,00,20,d5,ff,
 976                                <ep>
60510 data60,ea,20,91,94,20,84,9d,
 976                                <bb>
60520 dataa9,00,a8,85,d2,85,22,86,
 981                                <oj>
60530 datad3,20,91,94,20,84,9d,86,
 991                                <ge>
60540 data23,20,91,94,20,84,9d,ea,
 915                                <mg>
60550 dataea,20,b0,04,91,d2,c8,d0,
1209                                <ji>
60560 dataf6,e6,23,e6,d3,ca,d0,ef,
1601                                <lg>
60570 data60,20,91,94,20,84,9d,e0,
 966                                <cm>
60580 data00,d0,0a,20,91,94,20,84,
 707                                <hp>
60590 data9d,8e,14,ff,60,e0,01,d0,
1103                                <mi>
60600 data0a,20,91,94,20,84,9d,8e,
 798                                <if>
```

```
60610 data12,ff,60,4c,a1,94,a5,2d,
 964                                <ca>
60620 data85,dc,a5,2e,85,dd,60,a5,
1179                                <ld>
60630 datadc,85,2d,a5,dd,85,2e,60,
1059                                <fm>
60640 dataa5,2b,85,e8,a5,2c,85,e9,
1148                                <oo>
60650 data38,a5,2d,e9,02,85,2b,a5,
 842                                <ee>
60660 data2e,e9,00,85,2c,60,a5,e8,
 949                                <pe>
60670 data85,2b,a5,e9,85,2c,60,a9,
1016                                <kf>
60680 data2a,8d,06,03,a9,12,8d,07,
 527                                <kh>
60690 data03,60,48,ad,43,05,d0,fb,
 875                                <dk>
60700 data68,4c,6e,8b,a9,01,8d,02,
 742                                <ef>
60710 data40,20,18,88,20,4c,88,60,
 596                                <pa>
60720 data20,91,94,20,2c,93,a5,62,
 811                                <id>
60730 data85,af,a5,63,85,b0,a5,61,
1143                                <je>
60740 data85,ab,20,91,94,20,d2,9d,
1028                                <an>
60750 data20,3d,8a,a5,5f,85,da,a5,
1007                                <dn>
60760 data60,85,db,20,91,94,20,d2,
1015                                <gb>
60770 data9d,86,ae,20,3d,8a,18,a5,
 885                                <gg>
60780 data5f,69,0a,85,dc,a5,60,69,
 929                                <je>
60790 data00,85,dd,a9,02,85,ac,85,
 963                                <bn>
60800 dataad,a9,da,4c,32,13,20,91,
 882                                <me>
60810 data94,20,2c,93,a5,62,85,af,
 942                                <ao>
60820 dataa5,63,85,b0,a5,61,85,ab,
1139                                <kf>
60830 data20,91,94,20,d2,9d,86,ae,
1032                                <nl>
60840 data20,3d,8a,a5,5f,85,da,a5,
1007                                <ha>
60850 data60,85,db,a9,02,85,ac,a9,
1093                                <nc>
60860 data00,85,ad,a6,da,a4,db,20,
1105                                <gd>
60870 datad5,ff,60,a9,70,85,14,a9,
1167                                <oj>
60880 data17,85,15,20,3d,8a,a9,fc,
 829                                <jb>
60890 data85,14,a9,53,85,15,20,d2,
 801                                <en>
60900 data12,60,a5,5f,a6,60,85,24,
 805                                <ee>
60910 data86,25,20,3d,8a,90,15,a0,
 727                                <on>
60920 data01,20,d1,04,88,aa,d0,05,
 765                                <nc>
60930 data20,d1,04,f0,07,20,d1,04,
 737                                <nb>
60940 data85,5f,86,60,a5,24,38,e5,
 944                                <ko>
60950 data5f,aa,a5,25,e5,60,a8,b0,
1136                                <gl>
60960 data1f,8a,18,65,2d,85,2d,98,
 669                                <hp>
60970 data65,2e,85,2e,a0,00,20,d1,
 727                                <el>
60980 data04,91,24,c8,d0,f8,e6,60,
1167                                <hj>
60990 datae6,25,a5,2e,c5,25,b0,ee,
1126                                <eh>
61000 data20,18,88,a5,22,a6,23,18,
 616                                <ki>
61010 data69,02,85,2d,90,01,e8,86,
 796                                <le>
61020 data2e,60,a6,dc,a4,dd,20,d8,
1161                                <np>
61030 dataff,60,a0,00,98,99,00,20,
 848                                <fn>
61040 data99,40,21,99,80,22,c8,c0,
 957                                <po>
61050 data18,d0,f2,60,ad,06,ff,29,
1045                                <bh>
61060 dataef,8d,06,ff,60,ad,06,ff,
1171                                <bg>
61070 data09,10,8d,06,ff,60,20,91,
 700                                <hj>
61080 data94,20,84,9d,86,23,20,91,
 815                                <em>
61090 data94,20,84,9d,86,ae,20,91,
 954                                <ak>
61100 data94,20,84,9d,86,d3,20,91,
 991                                <el>
61110 data94,20,84,9d,86,d4,18,d8,
1055                                <nh>
61120 dataa9,01,85,ac,85,ab,a9,07,
 955                                <fm>
61130 data85,ad,20,c0,ff,a0,00,a2,
1107                                <of>
61140 data01,20,c9,ff,a9,0d,20,d2,
 913                                <kc>
61150 dataff,a6,d4,a0,78,98,85,22,
1232                                <jd>
61160 dataa0,00,a5,d3,20,d2,ff,20,
1065                                <go>
61170 datab0,04,29,3f,85,d0,20,b0,
 833                                <mp>
61180 data04,29,40,85,d1,18,65,d1,
 785                                <ea>
```

```
61190 data65,d0,85,d0,20,b0,04,49,
 935                                <ba>
61200 data7f,29,20,85,d1,18,65,d1,
 876                                <ke>
61210 data65,d0,85,d2,20,b0,04,29,
 905                                <in>
61220 data80,c9,80,d0,17,a9,12,20,
 907                                <ej>
61230 datad2,ff,ea,ea,ea,ea,ea,a5,
 1800                               <ch>
61240 datad2,20,d2,ff,a9,92,20,d2,
 1264                               <gg>
61250 dataff,18,90,05,a5,d2,20,d2,
 1045                               <ne>
61260 dataff,c8,c0,28,30,b1,a9,0d,
 1094                               <mo>
61270 data20,d2,ff,18,a5,22,69,28,
 865                                <nh>
61280 data85,22,a5,23,69,00,85,23,
 640                                <hm>
61290 dataca,d0,95,a9,0d,20,d2,ff,
 1238                               <dl>
61300 data20,cc,ff,a9,01,20,c3,ff,
 1143                               <bb>
61310 data60,ff,c4,28,49,29,aa,22,
 905                                <ha>
61320 fort= 4096 to 5145 step8:p=0 <md>
61330 fori=0to7:reada$:a=dec(a$):p
=p+a:poket+i,a:nexti               <ij>
61340 readr:ifp=rthen61360         <fd>
61350 print"Pruefsummenfehler in z
eile"peek(63)+peek(64)*256:gosub61
910:end                            <jn>
61360 next                         <mb>
61370 dataea,ea,ea,ea,ea,ea,ea,ea,
 1872                               <em>
61380 dataea,ea,ea,ea,ea,ea,ea,a2,
 1800                               <fh>
61390 data18,8e,60,15,a9,05,48,a2,
 691                                <jf>
61400 data04,ea,ea,a9,00,a0,20,85,
 966                                <pk>
61410 datafa,84,fb,68,48,a0,01,20,
 1002                               <lk>
61420 databa,ff,a9,00,20,bd,ff,20,
 1118                               <aj>
61430 datac0,ff,68,48,aa,20,c9,ff,
 1281                               <ae>
61440 dataa9,1b,20,d2,ff,a9,31,20,
 943                                <pm>
61450 datad2,ff,20,5f,15,a9,1b,20,
 841                                <ih>
61460 datad2,ff,a9,32,20,d2,ff,20,
 1213                               <ea>
61470 datacc,ff,68,4c,c3,ff,00,00,
 1089                               <gc>
61480 data00,00,00,00,00,00,16,a2,
 184                                <ma>
61490 data18,a9,0d,20,d2,ff,a0,05,
 868                                <ml>
61500 datab9,8f,15,20,d2,ff,88,10,
 998                                <ac>
61510 dataf7,a9,27,8d,5e,15,a0,00,
 871                                <ic>
61520 data20,95,15,a5,fa,18,69,08,
 754                                <bn>
61530 data90,02,e6,fb,85,fa,ce,5e,
 1310                               <cm>
61540 data15,10,eb,ca,10,d3,60,01,
 798                                <gl>
61550 data40,04,2a,1b,09,8a,48,98,
 508                                <jf>
61560 data48,a9,80,8d,d2,15,a2,07,
 910                                <cp>
61570 dataa0,00,b1,fa,2d,d2,15,38,
 919                                <cj>
61580 datad0,01,18,3e,56,15,c8,c0,
 794                                <li>
61590 data08,d0,ef,4e,d2,15,ca,10,
 982                                <kg>
61600 datae7,a2,07,bd,56,15,20,d2,
 938                                <ia>
61610 dataff,ea,ea,ea,a9,00,9d,56,
 1369                               <ci>
61620 data15,ca,10,ef,68,a8,68,aa,
 1024                               <je>
61630 data60,00,ea,ea,ea,ea,ea,ea,
 1500                               <ak>
61640 dataea,ea,ea,ea,ea,ea,ea,ea,
 1872                               <jk>
61650 dataea,ea,ea,ea,ea,ea,ea,ea,
 1872                               <la>
61660 dataea,ea,ea,ea,ea,ea,ea,ea,
 1872                               <kh>
61670 data20,d2,9d,a0,00,98,85,d0,
 1052                               <hp>
61680 dataa9,20,85,d1,a5,14,85,d2,
 1071                               <ic>
61690 dataa5,15,85,d3,b1,d0,91,d2,
 1270                               <mj>
61700 datac8,d0,f9,e6,d1,e6,d3,ca,
 1739                               <of>
61710 datad0,f2,60,20,d2,9d,a9,00,
 1114                               <ei>
61720 data85,d2,a9,20,85,d3,a5,14,
 1073                               <jn>
61730 data85,22,a5,15,85,23,ea,a0,
 915                                <mc>
61740 data00,b1,22,20,b0,04,91,d2,
 778                                <hi>
61750 datac8,d0,f6,e6,23,e6,d3,ca,
 1562                               <fm>
61760 datad0,ef,60,a2,01,bd,00,0f,
 910                                <id>
61770 fort= 5376 to 5695 step8:p=0 <bn>
61780 fori=0to7:reada$:a=dec(a$):p
```

```
=p+a:poket+i,a:nexti                                    <fe>
61790 readr:ifp=rthen61810                              <mm>
61800 print"Pruefsummenfehler in z
eile"peek(63)+peek(64)*256:gosub61
910:end                                                 <po>
61810 next                                              <ei>
61820 return                                            <gi>
61830 gosub 61900                                       <mb>
61840 color0,4:color4,4:color1,1:p
rintchr$(14)                                            <lg>
61850 printcl$                                          <ep>
61860 fori=1to8:keyi,"":next:key4,
"run1520"+chr$(13)                                      <lo>
61870 key5,"pn$="+chr$(34)+"delete
"+chr$(34)+":goto2220"+chr$(13)                         <fo>
61880 gosub60000:poke4796,96:poke4
800,234:poke4807,48:poke 4811,242                       <ik>
61890 return                                            <pf>
61900 poke 65286,peek(65286) and 2
39:return                                               <dd>
61910 poke 65286,peek(65286) or 16
:return                                                 <nl>
61920 rem igel=================p4                       <gn>
61930 rem 60671 bytes memory                            <ek>
61940 rem 15907 bytes programm                          <an>
61950 rem (im urzustand)                                <ln>
61960 rem                                               <ie>
61970 rem achtung !!!                                   <on>
61980 rem programm darf nicht                           <mj>
61990 rem umnummeriert werden                           <ji>
62000 rem ========================                      <mo>
```



## DELETE FUER IGELGRAFIK

```
5990 rem *neue befehle (programmmo
dus)*                                                   <dm>
6989 return                                             <fo>
6990 rem *neue befehle (direktmodu
s)*                                                     <pe>
7989 return                                             <dg>
19990 *****befehlssatz*****                             <kn>
```

---

((BU:)) Programm DELETE fuer IGEL-Grafik

```
5990 rem demo.pkt==============p4                       <nj>
6000 ifa$="quadrat"thenw$(i)=str$(
i)+":goS 10000"   :goto840                              <nc>
6020 ifa$="kiste5"thenw$(i)=str$(i
)+":goS 10020"    :goto840                              <ol>
6040 ifa$="stern5"thenw$(i)=str$(i
)+":goS 10040"    :goto840                              <pd>
6060 ifa$="dreieck"thenw$(i)=str$(
i)+":goS 10060"   :goto840                              <pe>
6080 ifa$="drehquadrat"thenw$(i)=s
tr$(i)+":goS 10080"          :goto84
0                                                       <gp>
6100 ifa$="mitte"thenw$(i)=str$(i)
+":goS 10100"     :goto840                              <oo>
6120 ifa$="demodrehquadrat"thenw$(
i)=str$(i)+":goS 10120"      :goto84
0                                                       <jl>
6140 ifa$="stern"thenw$(i)=str$(i)
+":goS 10140"     :goto840                              <ko>
6160 ifa$="demostern"thenw$(i)=str
$(i)+":goS 10160"            :goto84
0                                                       <ab>
6180 ifa$="stern2"thenw$(i)=str$(i
)+":goS 10180"    :goto840                              <ed>
6200 ifa$="demostern2"thenw$(i)=st
r$(i)+":goS 10200"           :goto84
0                                                       <kp>
6220 ifa$="rechteck1-5"thenw$(i)=s
tr$(i)+":goS 10220"          :goto84
0                                                       <bn>
6240 ifa$="rechteck1-3"thenw$(i)=s
tr$(i)+":goS 10240"          :goto84
0                                                       <kp>
6260 ifa$="rechteckrot."thenw$(i)=
str$(i)+":goS 10260"         :goto84
0                                                       <bj>
6280 ifa$="drehquadrat2"thenw$(i)=
str$(i)+":goS 10280"         :goto84
0                                                       <ho>
6300 ifa$="fassade"thenw$(i)=str$(
i)+":goS 10300"   :goto840                              <lf>
6320 ifa$="gehvor"thenw$(i)=str$(i
)+":goS 10320"    :goto840                              <bg>
6340 ifa$="fassaden"thenw$(i)=str$
(i)+":goS 10340" :goto840                               <lo>
6360 ifa$="rbogen"thenw$(i)=str$(i
)+":goS 10360"    :goto840                              <dh>
6380 ifa$="blatt"thenw$(i)=str$(i)
+":goS 10380"     :goto840                              <fo>
6400 ifa$="dolde"thenw$(i)=str$(i)
+":goS 10400"     :goto840                              <fj>
6420 ifa$="stiel"thenw$(i)=str$(i)
+":goS 10420"     :goto840                              <lm>
6440 ifa$="blume"thenw$(i)=str$(i)
+":goS 10440"     :goto840                              <kf>
6460 ifa$="garten"thenw$(i)=str$(i
)+":goS 10460"    :goto840                              <bo>
6480 ifa$="blumenmotiv"thenw$(i)=s
```

```
tr$(i)+":goS 10480"          :goto84
0                                    <gl>
6500 ifa$="demo1"thenw$(i)=str$(i)
+":goS 10500"      :goto840          <nc>
6520 ifa$="demo2"thenw$(i)=str$(i)
+":goS 10520"      :goto840          <ih>
6540 ifa$="demo3"thenw$(i)=str$(i)
+":goS 10540"      :goto840          <ap>
6560 ifa$="demo4"thenw$(i)=str$(i)
+":goS 10560"      :goto840          <fk>
6580 ifa$="demo"thenw$(i)=str$(i)+
":goS 10580"       :goto840          <jm>
6600 ifa$="wachsquadrat"thenw$(i)=
str$(i)+":goS 10600"         :goto84
0                                    <nf>
6620 ifa$="wachsfassade"thenw$(i)=
str$(i)+":goS 10620"         :goto84
0                                    <ah>
6989 return                          <fo>
6990 rem***neue befehle (direktmod
us)***                               <oc>
7000 ifa$="quadrat"thenc=a:gosub 1
0000:goto2120                        <mg>
7020 ifa$="kiste5"thenc=a:gosub 10
020:goto2120                         <fg>
7040 ifa$="stern5"thenc=a:gosub 10
040:goto2120                         <cj>
7060 ifa$="dreieck"thenc=a:gosub 1
0060:goto2120                        <dc>
7080 ifa$="drehquadrat"thenc=a:gos
ub 10080:goto2120                    <jo>
7100 ifa$="mitte"thenc=a:gosub 101
00:goto2120                          <lg>
7120 ifa$="demodrehquadrat"thenc=a
:gosub 10120:goto2120                <ag>
7140 ifa$="stern"thenc=a:gosub 101
40:goto2120                          <mj>
7160 ifa$="demostern"thenc=a:gosub
 10160:goto2120                      <pj>
7180 ifa$="stern2"thenc=a:gosub 10
180:goto2120                         <ad>
7200 ifa$="demostern2"thenc=a:gosu
b 10200:goto2120                     <nc>
7220 ifa$="rechteck1-5"thenc=a:gos
ub 10220:goto2120                    <dn>
7240 ifa$="rechteck1-3"thenc=a:gos
ub 10240:goto2120                    <fa>
7260 ifa$="rechteckrot."thenc=a:go
sub 10260:goto2120                   <op>
7280 ifa$="drehquadrat2"thenc=a:go
sub 10280:goto2120                   <jd>
7300 ifa$="fassade"thenc=a:gosub 1
0300:goto2120                        <ae>
7320 ifa$="gehvor"thenc=a:gosub 10
320:goto2120                         <bf>
7340 ifa$="fassaden"thenc=a:gosub
10340:goto2120                       <kd>
7360 ifa$="rbogen"thenc=a:gosub 10
360:goto2120                         <hi>
7380 ifa$="blatt"thenc=a:gosub 103
80:goto2120                          <ej>
7400 ifa$="dolde"thenc=a:gosub 104
00:goto2120                          <fc>
7420 ifa$="stiel"thenc=a:gosub 104
20:goto2120                          <ng>
7440 ifa$="blume"thenc=a:gosub 104
40:goto2120                          <ml>
7460 ifa$="garten"thenc=a:gosub 10
460:goto2120                         <kl>
7480 ifa$="blumenmotiv"thenc=a:gos
ub 10480:goto2120                    <hp>
7500 ifa$="demo1"thenc=a:gosub 105
00:goto2120                          <hb>
7520 ifa$="demo2"thenc=a:gosub 105
20:goto2120                          <dg>
7540 ifa$="demo3"thenc=a:gosub 105
40:goto2120                          <de>
7560 ifa$="demo4"thenc=a:gosub 105
60:goto2120                          <gc>
7580 ifa$="demo"thenc=a:gosub 1058
0:goto2120                           <bd>
7600 ifa$="wachsquadrat"thenc=a:go
sub 10600:goto2120                   <if>
7620 ifa$="wachsfassade"thenc=a:go
sub 10620:goto2120                   <bi>
7989 return                          <dg>
10000 :                              <ki>
10001 draw m to 0+c;b                <ig>
10002 w= 90:b=b+w:if b>360thenb=b-
360                                  <fh>
10003 draw m to 0+c;b                <ii>
10004 w= 90:b=b+w:if b>360thenb=b-
360                                  <dd>
10005 draw m to 0+c;b                <hc>
10006 w= 90:b=b+w:if b>360thenb=b-
360                                  <dp>
10007 draw m to 0+c;b                <gm>
10008 w= 90:b=b+w:if b>360thenb=b-
360                                  <ik>
10009 return                         <ao>
10020 c= 10                          <ch>
10021 :gosub 10000                   <cl>
10022 c= 20                          <ei>
10023 :gosub 10000                   <ci>
10024 c= 30                          <gk>
10025 :gosub 10000                   <cn>
10026 c= 40                          <im>
10027 :gosub 10000                   <co>
10028 c= 50                          <la>
10029 :gosub 10000                   <cp>
10030 return                         <di>
10040 :                              <pi>
10041 r1= 5                          <on>
10042 z 1=0:do                       <ee>
10043 draw m to 0+c;b                <nc>
10044 w= 144:b=b+w:if b>360thenb=b
```

```
-360                                    <mg>
10045 z 1=z 1+1:loopuntilz 1>=r 1       <oc>
10046 return                            <fi>
10060 :                                 <ca>
10061 r1= 3                             <co>
10062 z 1=0:do                          <eh>
10063 draw m to 0+c;b                   <mg>
10064 w= 120:b=b+w:if b>360thenb=b
-360                                    <bn>
10065 z 1=z 1+1:loopuntilz 1>=r 1       <io>
10066 return                            <ia>
10080 :                                 <ej>
10081 do until b> 359                   <mi>
10082 c= 50                             <bd>
10083 :gosub 10000                      <el>
10084 w= 0+iw:b=b+w:if b>360thenb=
b-360                                   <jp>
10085 loop:c=0                          <mc>
10086 return                            <kj>
10100 rem**pu**                         <op>
10101 locate160+((rgr(3)=3orrgr(4)
=4)*80),100:b=0:w=0                     <nh>
10102 rem**pd**                         <km>
10103 for v=1 to 2000:next              <dm>
10104 x=rdot(0):y=rdot(1):scnclr:l
ocate x,y                               <kc>
10105 return                            <mp>
10120 iw= 45                            <kj>
10121 :gosub 10080                      <im>
10122 :gosub 10100                      <ac>
10123 iw= 90                            <ao>
10124 :gosub 10080                      <ap>
10125 :gosub 10100                      <ic>
10126 iw= 15                            <ge>
10127 :gosub 10080                      <ij>
10128 :gosub 10100                      <pn>
10129 iw= 50                            <mh>
10130 :gosub 10080                      <ac>
10131 return                            <ad>
10140 :                                 <mb>
10141 do until b> 359                   <ko>
10142 draw m to 50;b                    <ak>
10143 w= 0+iw:b=b+w:if b>360thenb=
b-360                                   <ko>
10144 loop:c=0                          <cd>
10145 return                            <bp>
10160 iw= 135                           <hc>
10161 :gosub 10140                      <he>
10162 :gosub 10100                      <og>
10163 iw= 144                           <dd>
10164 :gosub 10140                      <op>
10165 :gosub 10100                      <gg>
10166 iw= 150                           <no>
10167 :gosub 10140                      <hb>
10168 :gosub 10100                      <pj>
10169 iw= 160                           <kf>
10170 :gosub 10140                      <ac>
10171 :gosub 10100                      <hl>
10172 return                            <ff>
10180 :                                 <bb>
10181 :                                 <bd>
10182 do until b> 359                   <ej>
10183 draw m to 0+c;b                   <do>
10184 w= 0+iw:b=b+w:if b>360thenb=
b-360                                   <lm>
10185 loop:c=0                          <ep>
10186 return                            <hb>
10200 iw= 144                           <gn>
10201 c= 100                            <bg>
10202 :gosub 10180                      <ji>
10203 :gosub 10100                      <ak>
10204 iw= 135                           <el>
10205 c= 50                             <ll>
10206 :gosub 10180                      <jk>
10207 :gosub 10100                      <ai>
10208 iw= 160                           <eh>
10209 c= 20                             <pf>
10210 :gosub 10180                      <je>
10211 return                            <kc>
10220 :                                 <ga>
10221 r1= 2                             <cl>
10222 z 1=0:do                          <ep>
10223 draw m to 1*c;b                   <ie>
10224 w= 90:b=b+w:if b>360thenb=b-
360                                     <bl>
10225 draw m to 5*c;b                   <me>
10226 w= 90:b=b+w:if b>360thenb=b-
360                                     <oh>
10227 z 1=z 1+1:loopuntilz 1>=r 1       <la>
10228 return                            <me>
10240 :                                 <ik>
10241 r1= 2                             <hp>
10242 z 1=0:do                          <jo>
10243 draw m to 1*c;b                   <kd>
10244 w= 90:b=b+w:if b>360thenb=b-
360                                     <bc>
10245 draw m to 3*c;b                   <mb>
10246 w= 90:b=b+w:if b>360thenb=b-
360                                     <oo>
10247 z 1=z 1+1:loopuntilz 1>=r 1       <lg>
10248 return                            <oo>
10260 r2= 6                             <li>
10261 z 2=0:do                          <gh>
10262 :gosub 10220                      <dn>
10263 w= 60:b=b+w:if b>360thenb=b-
360                                     <ep>
10264 z 2=z 2+1:loopuntilz 2>=r 2       <ba>
10265 return                            <ba>
10280 :                                 <nk>
10281 z 2=0:do                          <ga>
10282 :gosub 10000                      <co>
10283 w= 0+iw:b=b+w:if b>360thenb=
b-360                                   <dl>
10284 z 2=z 2+1:loopuntilz 2>=r 2       <me>
10285 return                            <di>
10300 :                                 <ac>
```

```
10301 :gosub 10000                          <je>
10302 draw m to 0+c;b                       <le>
10303 w= 30:b=b+w:if b>360thenb=b-
360                                         <di>
10304 :gosub 10060                          <bn>
10305 w=360-( 30):b=b+w:ifb>360the
nb=b-360                                    <po>
10306 draw m to-( 0+c);b                    <ni>
10307 return                                <ge>
10320 :                                     <ck>
10321 rem**pu**                             <jk>
10322 w= 90:b=b+w:if b>360thenb=b-
360                                         <ko>
10323 locate 10+c;b                         <gl>
10324 w=360-( 90):b=b+w:ifb>360the
nb=b-360                                    <dd>
10325 rem**pd**                             <fk>
10326 return                                <ik>
10340 c=-100                                <en>
10341 :gosub 10320                          <ni>
10342 c= 10                                 <fk>
10343 :gosub 10300                          <nj>
10344 :gosub 10320                          <fm>
10345 c= 20                                 <im>
10346 :gosub 10300                          <fl>
10347 :gosub 10320                          <np>
10348 c= 30                                 <ma>
10349 :gosub 10300                          <ni>
10350 :gosub 10320                          <fn>
10351 return                                <ln>
10360 :                                     <hl>
10361 r1= 9                                 <an>
10362 z 1=0:do                              <le>
10363 draw m to 0+c;b                       <al>
10364 w= 10:b=b+w:if b>360thenb=b-
360                                         <cm>
10365 z 1=z 1+1:loopuntilz 1>=r 1           <ed>
10366 return                                <nl>
10380 :gosub 10360                          <jg>
10381 w= 90:b=b+w:if b>360thenb=b-
360                                         <ij>
10382 :gosub 10360                          <jb>
10383 w= 90:b=b+w:if b>360thenb=b-
360                                         <cn>
10384 return                                <pp>
10400 r2= 3                                 <ho>
10401 z 2=0:do                              <ig>
10402 :gosub 10380                          <ih>
10403 w= 60:b=b+w:if b>360thenb=b-
360                                         <pl>
10404 z 2=z 2+1:loopuntilz 2>=r 2           <pe>
10405 return                                <cj>
10420 :                                     <pd>
10421 draw m to 7*c;b                       <mo>
10422 :gosub 10380                          <ib>
10423 draw m to 14*c;b                      <pm>
10424 :gosub 10400                          <ie>
10425 return                                <fb>
10440 :                                     <bl>
10441 :gosub 10420                          <pc>
10442 :gosub 10400                          <gp>
10443 draw m to-(21*c);b                    <im>
10444 return                                <hh>
10460 c=-60                                 <ij>
10461 :gosub 10320                          <je>
10462 c= 1                                  <nb>
10463 :gosub 10440                          <jj>
10464 c= 30                                 <ph>
10465 :gosub 10320                          <ja>
10466 c= 2                                  <pc>
10467 :gosub 10440                          <kd>
10468 c= 30                                 <dg>
10469 :gosub 10320                          <ji>
10470 c= 3                                  <bd>
10471 :gosub 10440                          <kf>
10472 return                                <ko>
10480 :                                     <gk>
10481 b=0:   do until b> 359                <io>
10482 :gosub 10440                          <bk>
10483 w= 0+iw:b=b+w:if b>360thenb=
b-360                                       <ab>
10484 loop                                  <ce>
10485 return                                <mi>
10500 c= 50                                 <fa>
10501 :gosub 10000                          <dj>
10502 :gosub 10100                          <mb>
10503 :gosub 10020                          <ea>
10504 :gosub 10100                          <mc>
10505 c= 80                                 <ke>
10506 :gosub 10040                          <mh>
10507 :gosub 10100                          <ec>
10508 c= 40                                 <mm>
10509 :gosub 10060                          <ej>
10510 :gosub 10100                          <ln>
10511 return                                <po>
10520 :gosub 10120                          <ko>
10521 :gosub 10100                          <ck>
10522 :gosub 10160                          <lh>
10523 :gosub 10100                          <cl>
10524 c= 20                                 <mj>
10525 :gosub 10220                          <db>
10526 :gosub 10100                          <kf>
10527 c= 30                                 <pl>
10528 :gosub 10240                          <lc>
10529 :gosub 10100                          <co>
10530 c= 20                                 <cj>
10531 :gosub 10260                          <do>
10532 :gosub 10100                          <km>
10533 return                                <ck>
10540 iw= 72                                <fj>
10541 c= 50                                 <oa>
10542 :gosub 10280                          <mb>
10543 :gosub 10100                          <cn>
10544 :gosub 10340                          <lo>
10545 :gosub 10100                          <cg>
10546 c= 10                                 <ci>
```

## DEMO ZUR IGELGRAFIK

```
10547 :gosub 10360                    <dk>
10548 :gosub 10100                    <ke>
10549 :gosub 10380                    <dn>
10550 :gosub 10100                    <kj>
10551 return                          <eo>
10560 c= 2                            <ol>
10561 :gosub 10400                    <ck>
10562 :gosub 10400                    <kl>
10563 :gosub 10100                    <bp>
10564 :gosub 10420                    <km>
10565 :gosub 10100                    <bm>
10566 :gosub 10440                    <lf>
10567 :gosub 10100                    <cb>
10568 :gosub 10460                    <lk>
10569 :gosub 10100                    <cc>
10570 iw= 30                          <ak>
10571 c= 3                            <ed>
10572 :gosub 10480                    <lm>
10573 return                          <hk>
10580 :gosub 10500                    <ke>
10581 :gosub 10520                    <ci>
10582 :gosub 10540                    <lb>
10583 :gosub 10560                    <ge>
10584 return                          <jb>
10600 zl= 10                          <il>
10601 :do until c> 100                <pe>
10602 :gosub 10000                    <mp>
10603 c=c+zl                          <gi>
10604 loop:c=0                        <ob>
10605 return                          <ll>
10620 zl= 10                          <bj>
10621 :do until c> 60                 <jc>
10622 :gosub 10300                    <pn>
10623 :gosub 10320                    <ia>
10624 c=c+zl                          <ll>
10625 loop:c=0                        <ig>
10626 return                          <of>
10641 c= 1                            <hh>
10642 :gosub 10480                    <bd>
10643 c= 10                           <dp>
10644 :gosub 10360                    <ai>
10645 return                          <al>
10646 rem demo.pkt==============p4   <gf>
10647 rem nur in verbindung           <mb>
10648 rem mit igel-programm           <pi>
10649 rem                            <be>
10650 rem achtung !!!                 <od>
10651 rem programm darf nicht         <ig>
10652 rem umnummeriert werden.        <ih>
10653 rem ========================    <hl>
```

AUF SCHATZSUCHE IM LABYRINTH

# Räuber

**Suchen Sie den Schatz, der in diesem 3-D-Labyrinth verborgen ist. In der linken oberen Ecke wird die Entfernung angezeigt. Die Zeit läuft mit. Wer ist der Schnellste?**

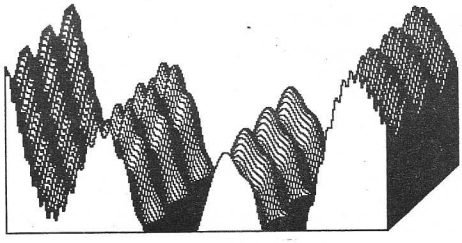
Sie stehen in einem Labyrinth. Wohin Sie auch schauen, überall nur Gänge und Wände. Doch halt, etwas gibt Ihnen Aufschluß darüber, wo Sie sich ungefähr befinden. In der oberen Ecke sehen Sie eine Zahl. Das ist die Entfernung zum verborgenen Schatz. Ihn zu finden, ist nur eine Frage der Zeit, denn keine Monster oder sonstige Feinde bedrohen Sie. Sie sind mutterseelenallein.
Oft glauben Sie, gleich müßten Sie den Schatz erreicht haben. Doch dies ist ein Trugschluß, denn urplötzlich ist der Weg zu Ende und eine undurchdringliche Wand behindert Ihr Fortkommen. Es gibt dann nur eines: wieder zurückzugehen und in einem der anderen zahlreichen Gänge Ihr Glück zu versuchen.
Haben Sie endlich den Schatz erreicht, wird Ihnen mitgeteilt, wie lange Sie dazu gebraucht haben. Am Anfang dürften Sie es schwer haben. Später, mit etwas Übung, wird es schneller gehen. Sie brauchen jedoch nicht zu glauben, daß Sie dann das Labyrinth schon auswendig kennen, denn jedesmal wird es neu aufgebaut.
Nach erfolgreicher Schatzsuche dürfen Sie das Labyrinth von oben betrachten, um einen Überblick zu bekommen. Sichtbar sind der Lageplan, Ihr Startpunkt, der Schatz und der zurückgelegte Weg. Wenn Sie wollen, dürfen Sie es jetzt von Neuem versuchen. □

```
10 rem raeuber----------------c16 <kn>
20 rem (p) commodore welt team <ho>
30 rem ========================== <ng>
40 rem (c) by eckhard schulz <no>
50 rem <pd>
60 rem <ah>
70 rem basic v3.5 <nl>
80 rem c16/116/plus4 <ki>
90 rem ========================== <jg>
100 gosub2860 <gc>
110 key8,"?":color0,1:color4,1:pri
ntwh$ <pp>
120 printchr$(142)chr$(8):gosub274
0 <di>
130 h=12:v=7:h1=h+1:w=818:v1=1630 <pn>
140 i=rnd(-ti) <nj>
150 dimxl(4),n2(4),yl(4),xr(4) <fo>
160 fori=0to4:readxl(i),n2(i),yl(i
),xr(i):next <ji>
170 data8,20,-4,31,8,12,1,27,12,6,
5,24,15,2,8,22,17,0,10,21 <ji>
180 cx=int(rnd(1)*h)+1:cy=int(rnd(
1)*v)+1 <ci>
190 c=cx:r=cy:s=0 <mf>
200 fori=0to(v+1)*h1:pokew+i,0:pok
ev1+i,0:next <gh>
210 printcl$c4$c4$"aufbau des laby
rinthes" <hf>
220 da=0:n=0:av=v1+c+r*h1:aw=w+c+r
*h1:pokeaw,1:s=s+1:ifs>=h*vthen480 <jo>
230 printmid$(rf$,(sand1)+1,1)"V"; <om>
240 ifc>1andpeek(aw-1)-0thenda=da+
2:n=n+1 <hh>
250 ifc<handpeek(aw+1)=0thenda=da+
1:n=n+1 <ei>
260 ifr>1andpeek(aw-h1)=0thenda=da
+8:n=n+1 <pd>
270 if(r<vandpeek(aw+h1)=0)thenda=
da+4:n=n+1 <nj>
280 n=int(rnd(1)*n)+1 <mi>
290 onda+1goto300,470,460,330,450,
340,350,360,440,370,380,390,400,41
0,420,430 <fm>
300 s=s-1 <dm>
310 c=c+1:ifc>hthenc=1:r=r+1:ifr>v
thenr=1 <bn>
320 onpeek(w+c+r*h1)+1goto310,220 <mo>
330 onngoto460,470 <ai>
340 onngoto450,470 <ig>
350 onngoto450,460 <im>
360 onngoto450,460,470 <bf>
370 onngoto440,470 <np>
380 onngoto440,460 <po>
390 onngoto440,460,470 <gk>
400 onngoto440,450 <ip>
410 onngoto440,450,470 <ej>
420 onngoto440,450,460 <go>
430 onngoto440,450,460,470 <mh>
440 r=r-1:av=av-h1:pokeav,peek(av)
or1:goto220 <fg>
450 pokeav,peek(av)or1:r=r+1:av=av
+h1:goto220 <fd>
460 c=c-1:av=av-1:pokeav,peek(av)o
r2:goto220 <ab>
470 pokeav,peek(av)or2:c=c+1:av=av
+1:goto220 <eg>
480 i=rnd(1)>.5:j=rnd(1):k=rnd(1)>
.5 <od>
490 ifithenx=int(j*h)+1:y=v+(v-1)*
k <oj>
500 ifi=0theny=int(j*v)+1:x=h+(h-1
)*k <mj>
510 if(cx-x)^2+(cy-y)^2<(h^2+v^2)/
9then480 <ip>
520 sx=x:sy=y <ae>
530 ti$="000000" <ei>
540 : <hg>
550 printcl$ <be>
560 fori=wtow+(v+1)*h1:pokei,0:nex
t <fj>
570 d=int(rnd(1)*4) <gn>
580 t4=ti:poke(w+x+y*h1),1 <fl>
590 iff-0thengosub950:printhe$;int
(sqr((x-cx)^2+(y-cy)^2)*10+.5)/10 <ok>
600 ifcx=xandcy=ythen2590 <pi>
610 getq$:ifq$=""goto610 <be>
620 ifq$="?"thenk=ti:goto2620 <pa>
630 jq=joy(2) <hi>
640 ifq$=c4$orjq=5thend=d+2 <nm>
650 ifq$=c1$orjq=7thend=d-1 <cf>
660 ifq$=c3$orjq=3thend=d+1 <nk>
670 ifd<0thend=d+4 <gc>
680 ifd>3thend=d-4 <pn>
690 ifq$=c2$orjq=1then700:else590 <eg>
700 av=v1+x+y*h1:aw=w+x+y*h1 <gb>
710 ondgoto740,760,780 <na>
720 ify>1and(peek(av-h1)and1)theny
=y-1:pokeaw-h1,1:goto590 <ek>
730 goto790 <jb>
740 ifx<hand(peek(av)and2)thenx=x+
1:pokeaw+1,1:goto590 <ga>
750 goto790 <mo>
760 ify<vand(peek(av)and1)theny=y+
1:pokeaw+h1,1:goto590 <li>
770 goto790 <bl>
780 ifx>1and(peek(av-1)and2)thenx=
x-1:pokeaw-1,1:goto590 <jm>
790 dn=23:gosub2710 <om>
800 printtab(16)zz$ze$ze$ze$ze$lef
t$(q1$,14)c4$rn$"kein durchgang"c2
$:fori=1to1000:next <gk>
810 printtab(16)c2$spc(14)left$(q1
$,14)c4$spc(14)c2$:goto600 <an>
820 printcl$" "; <ap>
830 fori=1toh:printze$ze$ze$;:next
:printc4$c4$ <om>
```

```
840 forj=1tov:printz9$c2$c1$z9$c2$
c1$z9$c4$c4$;:fori=1toh                <ja>
850 k=peek(v1+i+j*h1)                  <dd>
860 printmid$(ze$+ze$+zp$+b2$+z9$+
ze$+ze$+ze$+b3$,1+3*k,3);              <gi>
870 ifk<2thenprintc2$c1$z9$c2$c1$z
9$c4$c4$;                              <ip>
880 printc2$c1$;                       <dk>
890 ifpeek(w+i+j*h1)=1thenprintc1$
+".";                                  <oh>
900 ifi=sxandj=sythenprintc1$+rn$+
" "+rf$;                               <lb>
910 ifi=cxandj=cythenprintc1$+z5$;     <ae>
920 ifi=xandj=ythenprintc1$;mid$("
^>v<",d+1,1);                          <oj>
930 printc3$c4$;                       <bj>
940 next:printc4$c4$:next:return       <in>
950 s=0:printc1$                       <ap>
960 av=v1+x+y*h1                       <ca>
970 ondgoto1120,1260,1400              <al>
980 ifsy=y-sandsx=xthengosub2520       <bb>
990 ifcy=y-sandcx=xthengosub2440       <ec>
1000 k=peek(av-(s+1)*h1):ifk=0ork=
2goto1070                              <np>
1010 ifpeek(av-s*h1)and2thengosub1
550:goto1030                           <gf>
1020 gosub1670                         <mj>
1030 ifpeek(av-1-s*h1)and2thengosu
b2020:goto1050                         <oo>
1040 gosub2140                         <ei>
1050 s=s+1:ifs>4goto1540               <fk>
1060 goto980                           <fc>
1070 ifpeek(av-s*h1)and2thengosub1
750:goto1090                           <eo>
1080 gosub2220                         <dp>
1090 ifpeek(av-1-s*h1)and2thengosu
b1890:goto1110                         <kc>
1100 gosub2320                         <mf>
1110 return                            <ec>
1120 ifcy=yandcx=x+sthengosub2440      <ap>
1130 ifsy=yandsx=x+sthengosub2520      <lh>
1140 k=peek(av+s):ifk=0ork=1goto12
10                                     <he>
1150 ifpeek(av+s)=2thengosub1670:g
oto1170                                <bi>
1160 gosub1550                         <fp>
1170 ifpeek(av+s-h1)and1thengosub2
020:goto1190                           <pn>
1180 gosub2140                         <nl>
1190 s=s+1:ifs>4goto1540               <lb>
1200 goto1120                          <aj>
1210 ifpeek(av+s)=0thengosub2220:g
oto1230                                <fb>
1220 gosub1750                         <no>
1230 ifpeek(av+s-h1)and1thengosub1
890:goto1250                           <ln>
1240 gosub2320                         <fh>
1250 return                            <fl>
1260 ifcy=y+sandcx=xthengosub2440      <ah>
1270 ifsy=y+sandsx=xthengosub2520      <mn>
1280 k=peek(av+s*h1):ifk=0ork=2the
ngoto1350                              <ie>
1290 ifpeek(av-1+s*h1)and2thengosu
b1550:goto1310                         <nl>
1300 gosub1670                         <pm>
1310 ifpeek(av+s*h1)=1thengosub214
0:goto1330                             <oc>
1320 gosub2020                         <fn>
1330 s=s+1:ifs>4goto1540               <eh>
1340 goto1260                          <jp>
1350 ifpeek(av-1+s*h1)and2thengosu
b1750:goto1370                         <hl>
1360 gosub2220                         <ga>
1370 ifpeek(av+s*h1)=0thengosub232
0:goto1390                             <hi>
1380 gosub1890                         <po>
1390 return                            <he>
1400 ifcx=x-sandcy=ythengosub2440      <hb>
1410 ifsx=x-sandsy=ythengosub2520      <ck>
1420 k=peek(av-(s+1)):ifk=0ork=1th
engoto1490                             <jo>
1430 ifpeek(av-s-h1)and1thengosub1
550:goto1450                           <eb>
1440 gosub1670                         <im>
1450 ifpeek(av-s)and1thengosub2020
:goto1470                              <md>
1460 gosub2140                         <oo>
1470 s=s+1:ifs>4then1540               <ba>
1480 goto1400                          <ci>
1490 ifpeek(av-s-h1)and1thengosub1
750:goto1510                           <pa>
1500 gosub2220                         <pd>
1510 ifpeek(av-s)and1thengosub1890
:goto1530                              <af>
1520 gosub2320                         <ha>
1530 return                            <in>
1540 dn=11:gosub2710:printtab(19)"
MN"c4$c1$c1$"NM"he$:return             <mm>
1550 dn=y1(s):gosub2710                <cn>
1560 ifs=0thenprinttab(xr(s))ze$       <kl>
1570 ifs>0ands<4thenfori=1to4-s:pr
inttab(xr(s)+5-s)zj$:next              <ol>
1580 ifs>0thenprinttab(xr(s))mid$(
ze$+ze$+ze$+ze$+zj$,s)                 <kc>
1590 ifs=4thenprinttab(xr(s))zj$+z
j$+c4$+c1$+c1$+zj$+zj$                 <fh>
1600 q$="":ifs>0thenq$=mid$(c3$+c3
$+c3$+zj$,s)                           <mo>
1610 ifs<4thenfori=0ton2(s)+1:prin
ttab(xr(s))zj$q$:next                  <pk>
1620 ifs>0thenprinttab(xr(s))mid$(
zm$+zm$+zm$+zm$+zj$,s)                 <ha>
1630 ifs>0ands<4thenfori=1to4-s:pr
inttab(xr(s)+5-s)zj$:next              <fm>
1640 ifs=0thenprinttab(xr(s))zm$       <cg>
1650 printhe$                          <bl>
```

```
1660 return                                    <jd>
1670 dn=yl(s):gosub2710                        <el>
1680 ifs=0thenprinttab(xr(s))"N":g
oto1700                                        <be>
1690 fori=1to5-s:printtab(xr(s)+5-
s-i)"N":next                                   <ek>
1700 dn=n2(s)+2:gosub2720                      <ho>
1710 ifs=0thenprinttab(xr(s))"M":g
oto1730                                        <kc>
1720 fori=0to4-s:printtab(xr(s)+i)
"M":next                                       <jh>
1730 printhe$                                  <ca>
1740 return                                    <dd>
1750 j=3-s:ifs=0thenj=0                        <de>
1760 printhe$;                                 <di>
1770 ifs>0goto1810                             <en>
1780 printtab(20);:printc4$;:fori=
1to18:printzm$;:next:printzm$                  <lb>
1790 dn=n2(0):gosub2720                        <dd>
1800 printtab(20);:fori=1to19:prin
tze$;:next:goto1870                            <fm>
1810 dn=yl(s):gosub2720                        <on>
1820 fori=1to5-s:printtab(j+xr(s)+
2)zj$:next                                     <hp>
1830 printtab(20);:fori=19toj++xr(
s):printzm$;:next:printzj$                     <gk>
1840 ifs<4thenfori=1ton2(s):printt
ab(j+xr(s)+2)zj$:next                          <bo>
1850 printtab(20);:fori=19toj+xr(s
):printze$;:next:printzj$                      <gf>
1860 fori=1to5-s:printtab(j+xr(s)+
2)zj$:next                                     <fg>
1870 printhe$                                  <oj>
1880 return                                    <em>
1890 printhe$;                                 <ij>
1900 ifs>0goto1940                             <fk>
1910 printc4$;:fori=1to11+xl(0):pr
intzm$;:next:printzm$                          <ml>
1920 dn=n2(0):gosub2720                        <nh>
1930 fori=1to12+xl(0):printze$;:ne
xt:goto1870                                    <of>
1940 dn=yl(s):gosub2720                        <im>
1950 fori=1to5-s:printtab(xl(s))z9
$:next                                         <ba>
1960 printtab(xl(s));:printz9$;:fo
ri=xl(s)to17:printzm$;:next:printz
m$                                             <lf>
1970 ifs<4thenfori=1ton2(s):printt
ab(xl(s))z9$:next                              <ca>
1980 printtab(xl(s));:printz9$;:fo
ri=xl(s)to17:printze$;:next:printz
e$                                             <li>
1990 fori=1to5-s:printtab(xl(s))z9
$:next                                         <nb>
2000 printhe$                                  <bb>
2010 return                                    <fb>
2020 dn=yl(s):gosub2710                        <pp>
2030 ifs=0thenprinttab(xl(s))ze$               <lf>
2040 ifs>0ands<4thenfori=1to4-s:pr
inttab(xl(s))z9$:next                          <kd>
2050 ifs>0thenprinttab(xl(s))left$
(z9$+ze$+ze$+ze$+ze$,6-s)                      <ld>
2060 ifs=4thenprinttab(xl(s))z9$+z
9$+c1$+c1$+c4$+z9$+z9$                         <pp>
2070 q$="":ifs>0thenq$=mid$(c3$+c3
$+c3$+z9$,s)                                   <bf>
2080 ifs<4thenfori=0ton2(s)+1:prin
ttab(xl(s))z9$q$:next                          <fl>
2090 ifs>0thenprinttab(xl(s))left$
(z9$+zm$+zm$+zm$+zm$,6-s)                      <pk>
2100 ifs>0ands<4thenfori=1to4-s:pr
inttab(xl(s))z9$:next                          <mg>
2110 ifs=0thenprinttab(xl(s))zm$               <be>
2120 printhe$                                  <jj>
2130 return                                    <ec>
2140 dn=yl(s):gosub2710                        <oj>
2150 ifs=0thenprinttab(xl(s))"M":g
oto2170                                        <ak>
2160 fori=1to5-s:printtab(xl(s)+i)
"M":next                                       <am>
2170 dn=n2(s)+2:gosub2720                      <lh>
2180 ifs=0thenprinttab(xl(s))"N":g
oto2200                                        <mn>
2190 fori=0to4-s:printtab(xl(s)+5-
s-i)"N":next                                   <je>
2200 printhe$                                  <ke>
2210 return                                    <od>
2220 gosub1670                                 <gg>
2230 dn=yl(s)+5-s:gosub2710                    <dm>
2240 printtab(20);                             <ff>
2250 ifs<4thenfori=1toxr(s)-21:pri
ntzm$;:next                                    <ma>
2260 print"P"c1$c4$;                           <on>
2270 ifs<4thenfori=1ton2(s):printz
9$c4$c1$;:next                                 <hj>
2280 printzp$c1$c1$;                           <fc>
2290 ifs<4thenfori=1toxr(s)-20-1:p
rintze$c1$c1$;:next                            <an>
2300 printhe$                                  <nm>
2310 return                                    <km>
2320 ifs=0thenj=5:goto2340                     <kp>
2330 j=0                                       <nh>
2340 gosub2140                                 <gf>
2350 dn=yl(s)+5-s:gosub2710                    <nl>
2360 printtab(19);                             <jf>
2370 ifs<4thenfori=1to18-xl(s)-5+s
+j:printzm$c1$c1$;:next                        <pm>
2380 print"O"c1$c4$;                           <ji>
2390 ifs<4thenfori=1ton2(s):printz
j$c4$c1$;:next                                 <cp>
2400 print"L";                                 <hg>
2410 ifs<4thenfori=1to18-xl(s)-5+s
+j:printze$;:next                              <ca>
2420 printhe$                                  <gk>
2430 return                                    <jn>
2440 ifs=0ors=4thenreturn                      <pc>
```

```
2450 dn=18:gosub2710                                    <pb>
2460 ifs=3thenprinttab(20)c2$+c2$+
c2$+"."                                                 <ec>
2470 ifs=2thenprinttab(19)rn$+c2$+
z8$+rf$+z8$+c1$+c1$+c4$+z5$+"N"+c2
$+zj$                                                   <ln>
2480 ifs<>1thenreturn                                   <fk>
2490 printtab(18)c4$" "ze$ze$ze$le
ft$(ql$,4)c4$"N"b2$"N";                                 <dm>
2500 printzj$left$(ql$,5)c4$z5$z5$
z5$" "zj$left$(ql$,5)c4$z5$z5$z5$"
"                                                       <gd>
2510 return                                             <dn>
2520 ifs=0ors=4thenreturn                               <dl>
2530 dn=20:gosub2710                                    <pb>
2540 ifs=3thenprinttab(19)left$(qu
$,5)+rn$+z1$+z1$+c2$+c1$+c1$+rf$+z
1$+z1$                                                  <ek>
2550 ifs=2thenprinttab(16)c2$+c2$+
c2$+rn$+z8$b3$b3$yq$                                    <lm>
2560 ifs=1thenprinttab(13)rn$+z8$b
$b2$yq$                                                 <ig>
2570 ifs=1thenprinttab(12)rn$+z8$b
$b4$yq$                                                 <mh>
2580 return                                             <mk>
2590 printhe$b$"deine zeit: "left$
(ti$,2)":"mid$(ti$,3,2)"'"right$(t
i$,2)b5$b4$                                             <ek>
2600 k=ti:fori=1to25:poke2035,23:p
oke2036,15:sys65520:print"gefunden
":forj=1to50:nextj                                      <kb>
2610 poke2035,23:poke2036,15:sys65
520:printb$:forj=1to50:nextj,i                          <el>
2620 gosub820                                           <dm>
2630 printtab(10)c2$"noch ein spie
l";                                                     <ml>
2640 sw=1:tm=0                                          <da>
2650 gett$:ift$<>""then2680                             <oa>
2660 ifti>tmthenprintmid$("? ",sw,
1)c1$;:tm=ti+15:sw=3-sw                                 <oo>
2670 goto2650                                           <jc>
2680 ift$="j"thenprint" ja.":goto1
80                                                      <el>
2690 ift$="n"thenprint" nein"+c4$:
end                                                     <gi>
2700 goto2650                                           <go>
2710 printhe$;                                          <ie>
2720 ifdn>0thenforz=1todn:printc4$
;:next                                                  <pk>
2730 return                                             <ph>
2740 printc1$c4$spc(13)"schatzraeu
ber"                                                    <dl>
2750 printc4$spc(7)"(c) 1987 by ec
khard schulz"                                           <cc>
2760 printc4$c4$c4$" finde den sch
atz der in dem labyrinth                                <nl>
2770 printc4$" verborgen ist.                           <ml>
2780 printc4$" cursor up"b4$"- 1 s
chritt vorwaerts"                                       <em>
2790 print" cursor left"b2$"- dreh
ung nach links"                                         <ob>
2800 print" cursor right = drehung
 nach rechts                                            <hf>
2810 print" cursor down"b2$"= dreh
ung um 180 grad"                                        <hb>
2820 print" help"spc(9)"= hilfe"                        <je>
2830 printc4$" analog dazu steueru
ng mit dem joystick"b2$c4$"in port
 ii."                                                   <nf>
2840 printc4$c4$spc(7)"druecke 'ta
ste' zum start"                                         <ji>
2850 getkeya$:return                                    <kc>
2860 b$=chr$(32):b2$=b$+b$                              <ja>
2870 b3$=b2$+b$:b4$=b3$+b$                              <kf>
2880 b5$=b4$+b$:b$=b5$+b5$                              <io>
2890 rem nachspann ===============                      <kh>
2900 rem * farbcodes/steuercodes *                      <ad>
2910 wh$=chr$(005):c4$=chr$(017)                        <gd>
2920 rn$=chr$(018):he$=chr$(019)                        <fm>
2930 c3$=chr$(029):c2$=chr$(145)                        <ma>
2940 rf$=chr$(146):c1$=chr$(147)                        <he>
2950 c1$=chr$(157)                                      <id>
2960 rem *** zeichensatz/graphik *                      <gb>
2970 z1$=chr$(162):z5$=chr$(166)                        <km>
2980 z8$=chr$(169):z9$=chr$(170)                        <mm>
2990 ze$=chr$(175):zj$=chr$(180)                        <fb>
3000 zm$=chr$(183):zp$=chr$(186)                        <me>
3010 yq$=chr$(223)                                      <do>
3020 rem ********* zeichenfolgen *                      <ca>
3030 forq=1to10:zz$=zz$+ze$:next                        <kn>
3040 for q=1 to 40                                      <pp>
3050 qu$=qu$+c2$:ql$=ql$+c1$                            <ib>
3060 next q                                             <fe>
3070 return                                             <ka>
3080 rem -------------------------                      <op>
3090 rem 12277 bytes memory                             <ae>
3100 rem 07955 bytes program                            <pe>
3110 rem 00399 bytes variables                          <dj>
3120 rem 00128 bytes arrays                             <if>
3130 rem 00505 bytes strings                            <ih>
3140 rem 03290 bytes free (0)                           <eh>
3150 rem =========================                      <lh>
```



# Sprites und Animation

**Hier ist sie, die BASIC-Erweiterung, mit der Sie, sofern Ihr Rechner mit mehr als 16 KByte Hauptspeicher ausgerüstet ist, richtige Sprites erzeugen können. 26 Befehle bringen eine Menge Bewegung ins Spiel.**

Jetzt gibt es auch für den C16 Sprites, die den Hintergrund nicht überschreiben und pixelweise positioniert werden können. Außerdem stellen wir jede Menge Befehle vor, die das Programmieren von Animation, Scrolling und vielem mehr ermöglichen.

**Vorab noch einige Hinweise:**

1. Alle Befehlsworte beginnen mit einer Raute (#).
2. Im Direktmodus und nach den BASIC-Befehlen THEN und ELSE muß zusätzlich ein Doppelpunkt eingegeben werden:
   IFX=0THEN:#SPRITE0FF0
3. Der Befehl GRAPHIC CLR darf nicht mehr benutzt werden, da sonst S&A BASIC zerstört würde.
4. Bevor man die Sprites benutzt, muß man :#ZTRANS120 :#CHN120 : POKE65287,136 eingeben.
5. Aufbau eines Sprites:
   Ein Sprite ist ein aus neun Zeichen zusammengesetzter Block.
   Es werden aber nur zwei mal zwei Zeichen als Sprite dargestellt.
6. Sprites dürfen sich nicht überlagern, da es sonst zu Grafikverfälschungen kommt.

## DIE S&A-BASIC-BEFEHLE

1. **#SETSPRITE Nr, XG, YG, XF, YF**
   **[Nr=0–7; XG=0–37; YG=0–22; XF, YF=0–8]**

Setzt das Sprite Nr auf die Bildschirmposition XG; YG. XF und YF bieten die Möglichkeit, das Sprite nach der groben Positionierung auf die Feinposition XF; YF zu setzen.

2. **#SPRITEOFF NR**
   **[NR=0–7]**

Schaltet das Sprite Nr ab.

3. **#ANIMATE BC, R**
   **[BC=0–255; R=0–3]**

Das Zeichen mit dem Bildschirmcode BC wird um ein Pixel in die Richtung R verschoben. Was die einzelnen R-Werte bewirken, ist unter „Richtungswerte" beschrieben.

4. **#SPIEGEL BC, RH**
   **[BC=0–255; RH=0–1]**

Das Zeichen BC wird horizontal (RH=0) beziehungsweise vertikal (RH=1) gespiegelt.

5. **#FILL XO, YO, XU, YU, Z**
   **[XO, XU=0–39; YO,YU=0–24; Z=0–255]**

Füllt den Bildschirmausschnitt mit den Koordinaten XO; YO und XU; YU mit den CHR$-Code des Zeichens Z.

6. **#SPALTE Flag1**
   **[Flag1=0–1]**

Der Bildschirm wird abhängig von Flag1 auf 38 beziehungsweise 40 Spalten gesetzt (siehe auch unter Flag1).

7. **#ZEILE Flag1**
   **[Flag1=0–1]**

Der Bildschirm wird, abhängig von Flag1, auf 24- oder 25-Zeilen-Format gebracht (siehe auch unter Flag1).

8. **#SCROLL R, XO, YO, XU, YU, Flag2**
   **[R=0–3; XO, XU=0–39; YO,YU=0–24; Flag2=0–1]**

Scrollt oder rollt einen Bildschirmausschnitt mit den Koordinaten XO; YO und XU; YU in die Richtung R (siehe auch unter „Richtung", Flag2).

9. **#SOFTV Pos**
   **[Pos=0–7]**

Setzt den Bildschirm auf die vertikale Smooth-Scrolling-Position pos.

10. **#SOFTH Pos**
    **[Pos=0–7]**

Hat dieselbe Wirkung wie SoftV, nur in horizontaler Richtung.

# mit BASIC für C16/P4

11. #CHN ADR
[ADR=0–255]

Schaltet den selbstdefinierten Zeichensatz mit der High-Adresse ADR an (siehe auch ADR).

12. #CHOFF

Schaltet denselben aus und gleichzeitig den Original-Zeichensatz an.

13. #ZTRANS ADR
[ADR=0–255]

Kopiert den Original-Zeichensatz an die High-Adresse ADR ins RAM (siehe auch ADR).

14. #ZLINK BC, MUSTER
[BC=0–255; Muster=0–255]

Das Zeichen mit dem Bildschirmcode BC wird mit dem Muster EOR-verknüpft.

15. #MCN

Schaltet den Multicolor-Modus an.

16. #MCOFF

Schaltet ihn aus.

17. #ECMN

Schaltet den erweiterten Hintergrund-Modus (=Extended-Color-Mode) an.

18. #ECMOFF
Schaltet ihn aus.

19. #ZDESIGN BC, x, x, x, x, x, x, x, x
[BC,x=0–255]
Definiert ein neues Zeichen mit dem Bildschirmcode BC und den acht Byte x.

20. #SDESIGN Nr, Bnr, x, x, x, x, x, x, x, x
[Nr=0–7; Bnr=0–3; x=0–255]
Definiert den Block Bnr des Sprites Nr durch die acht Byte x.
Bnr=0 linke obere Sprite-Ecke
Bnr=1 linke untere Sprite-Ecke
Bnr=2 rechte obere Sprite-Ecke
Bnr=3 rechte untere Sprite-Ecke

21. #SREVERS Nr, Bnr
[Nr=0–7; Bnr=0–3]

Reversiert den Block Bnr des Sprites (siehe auch #SDESIGN BNR).

22. #ECMCOL F1, F2, F3
[F1, F2, F3=0–255]

Legt die drei zusätzlichen Farben für den ECM-MODE fest.

23. #MCCOL F1, F2
[F1, F2=0–255]

Legt die zwei zusätzlichen Farben des MC-Modes fest.

24. #MELODY Mnr, Tg, L, TH, TL, . . .
[Mnr=0–4; Tg=0–2; L=0–8; TH=0–1023; TL=0–255]

Definiert die Melodie Mnr auf dem Tongenerator Tg mit der Länge L und den dazugehörigen Noten (TH=Tonhöhe, TL=Tonlänge).

25. #PLAY Mnr
[Mnr=0–4]

Spielt die Melodie Mnr, die zuvor definiert wurde.

26. #SHOW

Zeigt alle S&A-BASIC-Befehle auf dem Bildschirm an.

## RICHTUNGSWERTE

Bei R=0 wird nach links, bei R=1 nach rechts verschoben. Nach oben wird bei R=2 verlagert, bei R=3 folglich nach unten

**Flag 1**
Flag 1=0 – Der Bildschirm wird verkleindert.
Flag 1=1 – Der Bildschirm hat seine normale Größe.

**Flag 2**
Flag 2 bestimmt, ob der Bildschirm gerollt:
Flag 2 < 128 oder gescrollt wird:
Flag 2 ≥128

*Bitte lesen Sie weiter auf Seite 122*

```
S&A-BASIC=======================P4
(P) COMMODORE WELT TEAM
==================================
(C) BY MICHAEL INDEN

PLUS4 (C16/116 + 64 KB)
==================================
Das Programm ist mit dem Maschinen-
sprachmonitor TEDMON unter Zuhilfe-
nahme des Pruefsummenprogrammes
CHECKMON einzugeben und auf Disket-
te mit S"S&A BASIC",8,1001,2005 ab-
zuspeichern, auf Kassette mit
S"S&A BASIC",1,1001,2005.
Danach kann das Programm wie ein
BASIC-Programm gehandhabt werden.
==================================
>1000 00 23 10 0a 00 8f 20 53 :<46>
>1008 26 41 20 42 41 53 49 43 :<bd>
>1010 3d 3d 3d 3d 3d 3d 3d 3d :<0b>
>1018 3d 3d 3d 3d 3d 3d 3d 3d :<3b>
>1020 50 34 00 44 10 14 00 8f :<f5>
>1028 20 28 50 29 20 42 59 20 :<bb>
>1030 43 4f 4d 4d 4f 44 4f 52 :<eb>
>1038 45 20 57 45 4c 54 20 54 :<fe>
>1040 45 41 4d 00 66 10 1e 00 :<6b>
>1048 8f 20 3d 3d 3d 3d 3d 3d :<73>
>1050 3d 3d 3d 3d 3d 3d 3d 3d :<8c>
>1058 3d 3d 3d 3d 3d 3d 3d 3d :<c0>
>1060 3d 3d 3d 3d 3d 00 81 10 :<a1>
>1068 28 00 8f 20 28 43 29 20 :<69>
>1070 42 59 20 4d 49 43 48 41 :<ef>
>1078 45 4c 20 49 4e 44 45 4e :<66>
>1080 00 87 10 32 00 8f 00 a5 :<b4>
>1088 10 3c 00 8f 20 50 4c 55 :<cf>
>1090 53 34 20 28 43 31 36 2f :<1d>
>1098 31 31 36 20 2b 20 36 34 :<04>
>10a0 20 4b 42 29 00 c7 10 46 :<58>
>10a8 00 8f 20 3d 3d 3d 3d 3d :<12>
>10b0 3d 3d 3d 3d 3d 3d 3d 3d :<d0>
>10b8 3d 3d 3d 3d 3d 3d 3d 00 :<7f>
>10c0 3d 3d 3d 3d 3d 3d 00 e2 :<90>
>10c8 10 50 00 9e 20 34 33 35 :<e6>
>10d0 38 3a 8f 20 4d 41 53 43 :<0c>
>10d8 48 49 4e 45 4e 43 4f 44 :<1e>
>10e0 45 00 04 11 5a 00 8f 20 :<08>
>10e8 3d 3d 3d 3d 3d 3d 3d 3d :<5f>
>10f0 3d 3d 3d 3d 3d 3d 3d 3d :<4f>
>10f8 3d 3d 3d 3d 3d 3d 3d 3d :<00>
>1100 3d 3d 3d 00 00 00 a9 4f :<8d>
>1108 85 2c 85 2e 20 9a 8a c6 :<60>
>1110 75 a9 4c 85 d0 a9 11 85 :<6d>
>1118 d1 a9 b9 85 d2 a9 0e 85 :<8c>
>1120 d3 a9 00 85 d4 a9 40 85 :<60>
>1128 d5 a5 d2 aa a0 00 b1 d0 :<74>
>1130 91 d4 c8 d0 f9 e6 d1 e6 :<bb>
>1138 d5 ca d0 f2 a6 d2 b1 d0 :<f2>
>1140 91 d4 c8 ca d0 f8 8e 00 :<df>
>1148 4f 4c 00 40 a9 fd a2 40 :<7e>
>1150 8d 08 03 8e 09 03 20 67 :<73>
>1158 c5 a2 00 bd 33 40 20 d2 :<dc>
>1160 ff e8 e0 cb d0 f5 a9 00 :<7c>
>1168 8d b9 4e a9 4e 85 2c 85 :<31>
>1170 2e a9 ba 85 2b a9 a6 85 :<f1>
>1178 2d 20 7b 8a 4c 03 87 23 :<47>
>1180 23 23 23 23 23 23 23 23 :<11>
>1188 23 23 23 23 23 23 23 23 :<01>
>1190 23 23 23 23 23 23 23 23 :<32>
>1198 23 23 23 23 23 23 23 23 :<21>
>11a0 23 23 23 23 23 23 23 23 :<52>
>11a8 23 20 53 26 41 20 42 20 :<bc>
>11b0 41 20 53 20 49 20 43 20 :<3a>
>11b8 20 46 4f 52 20 50 4c 55 :<86>
>11c0 53 34 2f 43 31 36 20 28 :<1a>
>11c8 36 34 4b 29 20 23 23 23 :<56>
>11d0 23 20 20 20 20 20 20 20 :<a5>
>11d8 20 20 20 20 20 20 20 20 :<b2>
>11e0 20 20 20 20 20 20 20 20 :<c2>
>11e8 20 20 20 20 20 20 20 20 :<d2>
>11f0 20 20 20 20 20 23 23 23 :<14>
>11f8 23 20 57 52 49 54 54 45 :<07>
>1200 4e 20 42 59 20 12 4d 49 :<66>
>1208 43 48 41 45 4c 20 20 49 :<92>
>1210 4e 44 45 4e 92 20 49 4e :<22>
>1218 20 38 37 2f 38 38 20 23 :<29>
>1220 23 23 23 23 23 23 23 23 :<52>
>1228 23 23 23 23 23 23 23 23 :<42>
>1230 23 23 23 23 23 23 23 23 :<ed>
>1238 23 23 23 23 23 23 23 23 :<62>
>1240 23 23 23 23 23 23 23 23 :<92>
>1248 23 20 73 04 c9 23 f0 06 :<df>
>1250 20 79 04 4c d9 8b 20 73 :<d6>
>1258 04 a2 00 86 19 a2 00 a0 :<d5>
>1260 00 bd 12 4a c9 80 d0 03 :<f8>
>1268 4c f6 49 d1 3b d0 26 e8 :<11>
>1270 c8 bd 12 4a d0 f5 18 98 :<fc>
>1278 65 3b 85 3b 90 03 e6 3c :<be>
>1280 18 a5 19 0a aa bd bf 4a :<50>
>1288 a8 bd c0 4a 8d 48 41 8c :<ee>
>1290 47 41 4c 90 47 e8 bd 12 :<54>
>1298 4a d0 fa e8 e6 19 4c 13 :<52>
>12a0 41 d7 4b 1f 4c 67 4c af :<5c>
>12a8 4c f7 4c 3f 4d 87 4d cf :<18>
>12b0 4d 20 cb 41 20 ee 49 c9 :<ac>
>12b8 26 b0 42 8d b5 41 20 ee :<29>
>12c0 49 c9 17 b0 38 8d b6 41 :<8b>
>12c8 20 ee 49 c9 09 b0 2e 8d :<e3>
>12d0 b7 41 20 ee 49 c9 09 b0 :<a4>
>12d8 24 8d b8 41 20 d9 41 20 :<79>
>12e0 f2 41 20 ff 41 20 4a 42 :<4a>
>12e8 20 78 42 20 0b 43 20 27 :<93>
>12f0 43 20 db 43 20 33 43 20 :<c5>
>12f8 94 43 4c 04 41 4c 0d 4a :<74>
>1300 00 00 00 00 00 20 cb 41 :<6a>
>1308 20 d9 41 20 ff 41 20 d3 :<84>
>1310 42 20 bf 43 4c 04 41 20 :<6e>
```

```
>1318 84 9d e0 08 90 03 4c 0d :<b3>
>1320 4a 8e b4 41 60 a9 4e a2 :<04>
>1328 5f 8e e9 41 85 1a ad b4 :<1b>
>1330 41 0a 0a 18 69 5f 90 03 :<75>
>1338 e6 1a 18 85 19 60 a0 00 :<ee>
>1340 b9 b5 41 91 19 c8 c0 04 :<d4>
>1348 d0 f6 60 a9 4e 8d 13 43 :<1b>
>1350 8d 19 43 8d 1f 43 8d c5 :<e4>
>1358 43 8d cb 43 20 a4 4e ad :<c5>
>1360 b4 41 8d 9f 4e 0a 0a 0a :<d7>
>1368 18 6d 9f 4e 69 17 90 13 :<ac>
>1370 ee 13 43 ee 19 43 ee 1f :<29>
>1378 43 ee c5 43 ee cb 43 20 :<88>
>1380 ab 4e 18 8d 12 43 8d 18 :<d7>
>1388 43 8d 1e 43 8d c4 43 8d :<fb>
>1390 ca 43 20 b2 4e 60 ad b4 :<ac>
>1398 41 0a aa bd 55 41 85 e2 :<a5>
>13a0 bd 56 41 85 e3 ae b4 41 :<78>
>13a8 86 28 a9 00 85 29 85 72 :<54>
>13b0 a9 48 85 71 20 3c 9a 86 :<d6>
>13b8 d0 a9 7c c0 00 f0 02 a9 :<e9>
>13c0 7d 85 d1 60 a2 00 8e a0 :<30>
>13c8 4e a5 e2 48 a2 01 8e 9f :<77>
>13d0 4e 8a 0a aa a5 d1 95 d1 :<9f>
>13d8 ae 9f 4e e8 e0 09 d0 ee :<42>
>13e0 a2 01 8e 9f 4e 18 a5 d0 :<9c>
>13e8 7d c9 42 90 06 a0 01 8c :<ef>
>13f0 a0 4e 18 48 8a 0a aa 68 :<da>
>13f8 95 d0 ad a0 4e f0 08 f6 :<1c>
>1400 d1 a0 00 8c a0 4e 18 ae :<a1>
>1408 9f 4e e8 e0 0a d0 d3 68 :<81>
>1410 85 e2 4c d3 42 00 08 10 :<64>
>1418 18 20 28 30 38 40 00 a0 :<11>
>1420 01 b1 19 0a aa bd 7c 4b :<32>
>1428 85 e5 bd 7b 4b a0 00 71 :<5c>
>1430 19 85 e4 90 03 18 e6 e5 :<c5>
>1438 a5 e5 85 e7 a5 e4 69 28 :<75>
>1440 85 e6 90 03 18 e6 e7 a5 :<26>
>1448 e7 85 e9 a5 e6 69 28 85 :<b5>
>1450 e8 90 03 18 e6 e9 60 a0 :<ec>
>1458 00 a2 00 b1 e4 9d 29 4e :<d8>
>1460 e8 b1 e6 9d 29 4e e8 b1 :<4b>
>1468 e8 9d 29 4e e8 c8 c0 03 :<fe>
>1470 d0 e9 60 a0 00 b1 e2 91 :<ec>
>1478 d0 c8 c0 72 d0 f7 60 a0 :<3a>
>1480 00 a5 d1 8d 80 43 a5 d0 :<f9>
>1488 8d 7f 43 8c 9f 4e 98 0a :<b7>
>1490 aa b5 d1 8d 83 43 8d 80 :<dc>
>1498 43 b5 d0 8d 82 43 8d 7f :<d2>
>14a0 43 8d 7f 43 ac 9f 4e a9 :<50>
>14a8 78 8d 7d 43 b9 29 4e 18 :<12>
>14b0 0a 90 03 20 8f 43 0a 90 :<27>
>14b8 03 20 8f 43 0a 90 03 20 :<80>
>14c0 8f 43 8d 7c 43 a2 00 bd :<04>
>14c8 08 78 1d d0 7c 9d d0 7c :<0f>
>14d0 e8 e0 08 d0 f2 c8 c0 09 :<88>
>14d8 d0 b1 60 ee 7d 43 18 60 :<3a>
>14e0 ae b4 41 8a 0a 0a 0a 6d :<cf>
>14e8 b4 41 69 80 8d a1 4e 69 :<76>
>14f0 03 8d a2 4e ad a1 4e a0 :<4d>
>14f8 00 91 e4 69 01 91 e6 69 :<4b>
>1500 01 91 e8 69 01 c8 c0 03 :<b6>
>1508 d0 ef 60 a0 00 a2 00 bd :<04>
>1510 29 4e 91 e4 e8 bd 29 4e :<03>
>1518 91 e6 e8 bd 29 4e 91 e8 :<83>
>1520 e8 c8 c0 03 d0 e9 60 ae :<bc>
>1528 b7 41 e0 00 f0 0f ea ea :<6e>
>1530 ea a2 00 20 2b 44 e8 8a :<97>
>1538 cd b7 41 d0 f6 ae b8 41 :<eb>
>1540 e0 00 f0 0f ea ea ea a2 :<75>
>1548 00 20 08 44 e8 8a cd b8 :<27>
>1550 41 d0 f6 60 a0 01 b1 d0 :<9f>
>1558 48 b1 d6 48 b1 dc 88 91 :<f3>
>1560 dc 68 91 d6 68 91 d0 c8 :<f6>
>1568 c8 c0 18 d0 e9 88 a9 00 :<c7>
>1570 91 d0 91 d6 91 dc 60 a0 :<87>
>1578 00 b1 e0 4a 91 e0 b1 de :<f4>
>1580 4a 91 de b1 dc 4a 91 dc :<e9>
>1588 18 b1 da 4a 91 da 90 07 :<08>
>1590 a9 7f 71 e0 91 e0 18 b1 :<16>
>1598 d8 4a 91 d8 90 07 a9 7f :<4b>
>15a0 71 de 91 de 18 b1 d6 4a :<ee>
>15a8 91 d6 90 07 a9 7f 71 dc :<ce>
>15b0 91 dc 18 b1 d4 4a 91 d4 :<39>
>15b8 90 07 a9 7f 71 da 91 da :<a7>
>15c0 18 b1 d2 4a 91 d2 90 07 :<94>
>15c8 a9 7f 71 d8 91 d8 18 b1 :<83>
>15d0 d0 4a 91 d0 90 07 a9 7f :<b3>
>15d8 71 d6 91 d6 18 c8 c0 08 :<87>
>15e0 d0 97 60 20 a4 49 20 bc :<81>
>15e8 49 20 ee 49 e0 00 f0 0f :<f3>
>15f0 e0 01 f0 1c e0 02 f0 2b :<c8>
>15f8 e0 03 f0 41 4c 0d 4a a0 :<f7>
>1600 00 b1 19 0a 69 00 91 19 :<75>
>1608 c8 c0 08 d0 f4 4c 04 41 :<7e>
>1610 a0 00 b1 19 4a 90 02 69 :<3d>
>1618 7f 91 19 c8 c0 08 d0 f2 :<7c>
>1620 4c 04 41 a0 00 b1 19 48 :<08>
>1628 a0 01 b1 19 88 91 19 c8 :<58>
>1630 c8 c0 08 d0 f5 68 a0 07 :<24>
>1638 91 19 4c 04 41 a0 07 b1 :<5d>
>1640 19 48 a0 06 b1 19 c8 91 :<e7>
>1648 19 88 88 c0 ff d0 f5 68 :<4c>
>1650 a0 00 91 19 4c 04 41 20 :<0b>
>1658 a4 49 20 bc 49 20 ee 49 :<12>
>1660 e0 02 90 03 4c 0d 4a e0 :<fa>
>1668 01 f0 29 a5 19 8d 34 45 :<eb>
>1670 8d 3a 45 a5 1a 8d 35 45 :<fb>
>1678 8d 3b 45 a2 00 a0 07 bd :<e4>
>1680 00 0c 48 b1 19 9d 00 0c :<05>
>1688 68 91 19 88 e8 e0 04 d0 :<bc>
>1690 ee 4c 04 41 a0 00 b1 19 :<da>
>1698 29 0f 20 6d 45 0a 0a 0a :<93>
>16a0 0a 85 1f b1 19 4a 4a 4a :<c6>
>16a8 4a 20 6d 45 18 65 1f 91 :<eb>
>16b0 19 c8 c0 08 d0 e0 4c 04 :<2d>
```

```
>16b8 41 a2 00 dd 7d 45 f0 05 :<e1>
>16c0 e8 e0 10 d0 f6 bd 8d 45 :<ba>
>16c8 60 00 01 02 03 04 05 06 :<10>
>16d0 07 08 09 0a 0b 0c 0d 0e :<7f>
>16d8 0f 00 08 04 0c 02 0a 06 :<9d>
>16e0 0e 01 09 05 0d 03 0b 07 :<ab>
>16e8 0f 20 84 9d 8a cd f2 45 :<15>
>16f0 90 03 4c 0d 4a 8d f7 45 :<3d>
>16f8 a0 01 98 48 20 ee 49 68 :<4f>
>1700 a8 8a d9 f2 45 90 03 4c :<e2>
>1708 0d 4a 99 f7 45 c8 c0 05 :<a2>
>1710 d0 e8 ae f8 45 ac f7 45 :<6b>
>1718 18 20 f0 ff ae f7 45 86 :<25>
>1720 21 ad fb 45 20 d2 ff a5 :<36>
>1728 21 e6 21 cd f9 45 90 f1 :<d3>
>1730 ad f8 45 ee f8 45 cd fa :<fd>
>1738 45 90 d7 4c 04 41 28 19 :<82>
>1740 28 19 c0 00 0b 1d 0a 41 :<68>
>1748 20 84 9d 8a f0 07 c9 01 :<28>
>1750 f0 0b 4c 0d 4a ad 07 ff :<be>
>1758 29 f7 4c 16 46 ad 07 ff :<07>
>1760 09 08 8d 07 ff 4c 04 41 :<56>
>1768 20 84 9d 8a f0 07 c9 01 :<68>
>1770 f0 0b 4c 0d 4a ad 06 ff :<be>
>1778 29 f7 4c 36 46 ad 06 ff :<07>
>1780 09 08 8d 06 ff 4c 04 41 :<8e>
>1788 20 84 9d 8a 85 20 20 ee :<63>
>1790 49 c9 28 b0 2a 85 1b 20 :<4f>
>1798 ee 49 c9 19 b0 21 85 1c :<43>
>17a0 20 ee 49 c9 28 b0 18 85 :<a4>
>17a8 1d 20 ee 49 c9 19 b0 0f :<ea>
>17b0 85 1e 20 ee 49 85 21 e6 :<8a>
>17b8 1e a5 20 c9 04 90 03 4c :<d5>
>17c0 0d 4a c9 03 d0 05 c6 1e :<91>
>17c8 4c 2a 47 c9 02 d0 05 c6 :<ea>
>17d0 1e 4c ef 46 c9 00 f0 05 :<66>
>17d8 e6 1d 4c 93 46 c6 1b a4 :<80>
>17e0 1c 20 e0 49 a5 20 f0 06 :<46>
>17e8 20 b1 46 4c a5 46 20 d0 :<d8>
>17f0 46 a4 1c c8 84 1c c4 1e :<b0>
>17f8 90 e5 4c 04 41 c6 1d a4 :<39>
>1800 1d e6 1d b1 19 aa 24 21 :<63>
>1808 10 02 a2 20 a4 1b b1 19 :<2a>
>1810 48 8a 91 19 68 aa c8 c4 :<2b>
>1818 1d d0 f3 60 e6 1b a4 1b :<2c>
>1820 c6 1b b1 19 aa 24 21 10 :<4b>
>1828 02 a2 20 a4 1d b1 19 48 :<25>
>1830 8a 91 19 68 aa 88 c4 1b :<95>
>1838 d0 f3 60 a4 1c 20 e0 49 :<3f>
>1840 e6 1d a4 1b b1 19 99 af :<4c>
>1848 4b c8 c4 1d d0 f6 20 c7 :<ae>
>1850 49 a9 de 8d 0c 47 20 f6 :<bb>
>1858 00 24 21 30 13 a4 1e 20 :<70>
>1860 e0 49 a4 1b a4 1b b9 af :<c9>
>1868 4b 91 19 c8 c4 1d d0 f6 :<c8>
>1870 20 70 de 4c 04 41 a4 1e :<f3>
>1878 20 e0 49 e6 1d a4 1b b1 :<42>
>1880 19 99 af 4b c8 c4 1d d0 :<ae>
>1888 f6 20 c7 49 20 04 df 24 :<11>
>1890 21 30 11 a4 1c 20 e0 49 :<7e>
>1898 a4 1b b9 af 4b 91 19 c8 :<74>
>18a0 c4 1d d0 f6 20 70 de 4c :<d2>
>18a8 04 41 20 84 9d e0 08 90 :<22>
>18b0 03 4c 0d 4a 86 19 ad 06 :<43>
>18b8 ff 29 f8 05 19 8d 06 ff :<25>
>18c0 4c 04 41 20 84 9d e0 08 :<bb>
>18c8 90 03 4c 0d 4a 86 19 ad :<eb>
>18d0 07 ff 29 f8 05 19 8d 07 :<39>
>18d8 ff 4c 04 41 20 84 9d 86 :<10>
>18e0 21 8e 13 ff a9 c0 8d 12 :<29>
>18e8 ff c6 21 a5 21 85 38 a9 :<ff>
>18f0 ff 85 37 4c 04 41 a9 d0 :<bb>
>18f8 8d 13 ff a9 c4 8d 12 ff :<ef>
>1900 4c 04 41 20 84 9d 86 1a :<cd>
>1908 a2 d0 86 1c a2 00 86 19 :<6b>
>1910 86 1b a2 08 a0 00 b1 1b :<c0>
>1918 91 19 88 c0 00 d0 f7 e6 :<d3>
>1920 1a e6 1c ca e0 00 d0 ec :<eb>
>1928 4c 04 41 20 a4 49 20 bc :<08>
>1930 49 20 ee 49 8d f0 47 a0 :<6f>
>1938 00 b1 19 49 ff 91 19 c8 :<3e>
>1940 c0 08 d0 f5 4c 04 41 ad :<7c>
>1948 07 ff 09 10 8d 07 ff 4c :<f2>
>1950 04 41 ad 07 ff 29 ef 8d :<64>
>1958 07 ff 4c 04 41 ad 06 ff :<ca>
>1960 09 40 8d 06 ff 4c 04 41 :<c4>
>1968 ad 06 ff 29 bf 8d 06 ff :<86>
>1970 4c 04 41 20 a4 49 20 bc :<7d>
>1978 49 a2 00 86 21 20 ee 49 :<f7>
>1980 a4 21 91 19 a6 21 e8 e0 :<73>
>1988 08 d0 f0 4c 04 41 20 48 :<80>
>1990 48 4c 75 48 20 84 9d e0 :<ba>
>1998 08 90 03 4c 0d 4a 8e b4 :<39>
>19a0 41 20 4a 42 20 78 42 20 :<a8>
>19a8 ee 49 aa e0 04 90 03 4c :<29>
>19b0 0d 4a a5 e2 18 7d 8a 48 :<f8>
>19b8 85 19 a9 00 65 e3 85 1a :<05>
>19c0 60 a0 00 8c 9f 4e 20 ee :<38>
>19c8 49 ac 9f 4e 91 19 c8 c0 :<76>
>19d0 08 d0 f0 4c 04 41 08 10 :<f6>
>19d8 20 28 20 48 48 a0 00 b1 :<ee>
>19e0 19 49 ff 91 19 c8 c0 08 :<8b>
>19e8 d0 f5 4c 04 41 20 84 9d :<2d>
>19f0 8e 16 ff 20 ee 49 8d 17 :<28>
>19f8 ff 20 ee 49 8d 18 ff 4c :<0b>
>1a00 04 41 20 84 9d 8e 17 ff :<14>
>1a08 20 ee 49 8d 18 ff 4c 04 :<1f>
>1a10 41 20 84 9d 8a c9 05 90 :<89>
>1a18 03 4c 0d 4a 0a 85 1b 0a :<5f>
>1a20 0a 85 1d 0a 65 1b 65 1d :<95>
>1a28 69 f3 85 19 a9 4a 85 1a :<8a>
>1a30 20 ee 49 c9 03 90 03 4c :<8a>
>1a38 0d 4a a0 00 91 19 20 ee :<d5>
>1a40 49 c9 09 90 03 4c 0d 4a :<f7>
>1a48 85 21 0a 69 02 65 21 85 :<9b>
>1a50 21 a0 01 91 19 c8 84 20 :<5f>
```

```
>1a58 20 91 94 20 e1 9d a5 15 :<f4>
>1a60 c9 04 90 03 4c 0d 4a a4 :<b0>
>1a68 20 a5 14 91 19 c8 a5 15 :<71>
>1a70 91 19 c8 84 20 20 ee 49 :<99>
>1a78 a4 20 91 19 c8 c4 21 d0 :<ab>
>1a80 d5 4c 04 41 20 84 9d 8a :<a1>
>1a88 c9 05 90 03 4c 0d 4a 0a :<16>
>1a90 85 1b 0a 0a 85 1d 0a 65 :<a0>
>1a98 1b 65 1d 69 f3 85 19 a9 :<61>
>1aa0 4a 85 1a a0 01 b1 19 85 :<ea>
>1aa8 21 a2 08 20 c0 b8 a0 02 :<c6>
>1ab0 84 20 a0 00 b1 19 85 80 :<cd>
>1ab8 a4 20 b1 19 85 7e c8 b1 :<cc>
>1ac0 19 85 7f c8 84 20 b1 19 :<f4>
>1ac8 a8 a9 00 20 61 b8 a4 20 :<bc>
>1ad0 c8 c4 21 d0 db 4c 04 41 :<c6>
>1ad8 a2 00 a0 00 bd 12 4a 30 :<a9>
>1ae0 0c d0 03 a9 20 c8 20 d2 :<c4>
>1ae8 ff e8 4c 90 49 4c 04 41 :<ab>
>1af0 20 84 9d 86 28 a2 08 86 :<1c>
>1af8 71 a2 00 86 29 86 72 18 :<d7>
>1b00 20 3c 9a 84 1a 86 19 60 :<02>
>1b08 ad 13 ff 18 e9 01 65 1a :<8f>
>1b10 85 1a 60 a5 1b 8d e7 07 :<1b>
>1b18 c6 1d a5 1d 8d e8 07 e6 :<41>
>1b20 1d a5 1c 8d e6 07 a5 1e :<15>
>1b28 8d e5 07 60 98 0a a8 b9 :<32>
>1b30 7b 4b 85 19 b9 7c 4b 85 :<5b>
>1b38 1a 60 20 91 94 20 84 9d :<9e>
>1b40 8a 60 a9 01 85 24 a9 4a :<ed>
>1b48 85 25 4c da 86 4e 4f 20 :<40>
>1b50 53 26 41 20 42 41 53 49 :<16>
>1b58 c3 a2 0e 4c 86 86 53 45 :<dd>
>1b60 54 53 50 52 49 54 45 00 :<16>
>1b68 53 50 52 49 54 45 4f 46 :<c5>
>1b70 46 00 41 4e 49 4d 41 54 :<e1>
>1b78 45 00 53 50 49 45 47 45 :<ad>
>1b80 4c 00 46 49 4c 4c 00 53 :<b3>
>1b88 50 41 4c 54 45 00 5a 45 :<5e>
>1b90 49 4c 45 00 53 43 52 4f :<ec>
>1b98 4c 4c 00 53 4f 46 54 56 :<89>
>1ba0 00 53 4f 46 54 48 00 43 :<40>
>1ba8 48 4e 00 43 48 4f 46 46 :<05>
>1bb0 00 5a 54 52 41 4e 53 00 :<7f>
>1bb8 5a 4c 49 4e 4b 00 4d 43 :<76>
>1bc0 4e 00 4d 43 4f 46 46 00 :<89>
>1bc8 45 43 4d 4e 00 45 43 4d :<3b>
>1bd0 4f 46 46 00 5a 44 45 53 :<ba>
>1bd8 49 47 4e 00 53 44 45 53 :<56>
>1be0 49 47 4e 00 53 52 45 56 :<48>
>1be8 45 52 53 00 45 43 4d 43 :<53>
>1bf0 4f 4c 00 4d 43 43 4f 4c :<53>
>1bf8 00 4d 45 4c 4f 44 59 00 :<e9>
>1c00 50 4c 41 59 00 53 48 4f :<d6>
>1c08 57 00 80 65 41 b9 41 97 :<c3>
>1c10 44 0b 45 9d 45 fc 45 1c :<c3>
>1c18 46 3c 46 5e 47 77 47 90 :<13>
>1c20 47 aa 47 b7 47 df 47 fb :<69>

>1c28 47 06 48 11 48 1c 48 27 :<d6>
>1c30 48 42 48 8e 48 a1 48 b6 :<72>
>1c38 48 c5 48 38 49 8c 49 00 :<99>
>1c40 00 00 00 00 00 00 00 00 :<9c>
>1c48 00 00 00 00 00 00 00 00 :<ac>
>1c50 00 00 00 00 00 00 00 00 :<bc>
>1c58 00 00 00 00 00 00 00 00 :<cc>
>1c60 00 00 00 00 00 00 00 00 :<dc>
>1c68 00 00 00 00 00 00 00 00 :<ec>
>1c70 00 00 00 00 00 00 00 00 :<fc>
>1c78 00 00 00 00 00 00 00 00 :<0c>
>1c80 00 00 00 00 00 00 00 00 :<1d>
>1c88 00 00 00 00 00 00 00 00 :<2d>
>1c90 00 00 00 00 00 00 00 00 :<3d>
>1c98 00 00 00 00 00 00 00 00 :<4d>
>1ca0 00 00 00 00 00 00 00 00 :<5d>
>1ca8 00 00 00 00 00 00 00 00 :<6d>
>1cb0 00 00 00 00 00 00 00 00 :<7d>
>1cb8 00 00 00 00 00 00 00 00 :<8d>
>1cc0 00 00 00 00 00 00 00 00 :<9d>
>1cc8 0c 28 0c 50 0c 78 0c a0 :<8f>
>1cd0 0c c8 0c f0 0c 18 0d 40 :<30>
>1cd8 0d 68 0d 90 0d b8 0d e0 :<3d>
>1ce0 0d 08 0e 30 0e 58 0e 80 :<e3>
>1ce8 0e a8 0e d0 0e f8 0e 20 :<3d>
>1cf0 0f 48 0f 70 0f 98 0f c0 :<fc>
>1cf8 0f e8 0f 00 00 00 00 00 :<28>
>1d00 00 00 00 00 00 00 00 00 :<1d>
>1d08 00 00 00 00 00 00 00 00 :<2d>
>1d10 00 00 00 00 00 00 00 00 :<3d>
>1d18 00 00 00 00 00 00 00 00 :<4d>
>1d20 00 00 00 00 00 00 00 00 :<5d>
>1d28 00 00 00 ff ff ff ff ff :<74>
>1d30 ff ff ff ff ff ff ff ff :<7d>
>1d38 ff ff ff 00 00 00 00 00 :<86>
>1d40 00 00 00 ff ff ff ff ff :<a4>
>1d48 ff ff ff ff ff ff ff ff :<ad>
>1d50 ff ff ff 00 00 00 00 00 :<b6>
>1d58 00 00 00 00 00 00 00 00 :<cd>
>1d60 00 00 00 00 00 00 00 00 :<dd>
>1d68 00 00 00 00 00 00 00 00 :<ed>
>1d70 00 00 00 00 08 1c 1c 1c 3e :<29>
>1d78 3e be ff ff be 3e 3e 1c :<0d>
>1d80 1c 1c 08 00 00 00 00 00 :<92>
>1d88 00 00 00 00 00 00 00 04 :<30>
>1d90 0e 3e ff ff 3e 0e 04 00 :<43>
>1d98 00 00 00 00 00 00 00 00 :<4e>
>1da0 00 00 00 00 00 00 00 00 :<5e>
>1da8 00 00 00 00 00 00 00 00 :<6e>
>1db0 00 00 00 00 00 00 00 00 :<7e>
>1db8 00 00 00 01 02 03 04 05 :<9a>
>1dc0 06 07 08 00 00 00 00 00 :<d2>
>1dc8 00 00 00 00 00 00 00 00 :<ae>
>1dd0 00 00 00 00 00 00 00 00 :<be>
>1dd8 00 00 00 00 00 00 00 00 :<ce>
>1de0 00 00 00 00 00 00 00 00 :<de>
>1de8 00 00 00 00 00 00 00 00 :<ee>
>1df0 00 00 00 00 00 00 00 00 :<fe>
```

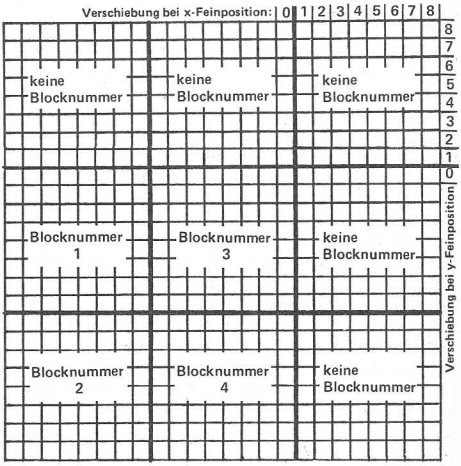
```
>1df8 00 00 00 00 00 00 00 00 :<0e>
>1e00 00 00 00 00 00 00 00 00 :<1e>
>1e08 00 00 00 00 00 00 00 00 :<2e>
>1e10 00 00 00 00 00 00 00 00 :<3e>
>1e18 00 00 00 00 00 00 00 00 :<4e>
>1e20 00 00 00 00 00 00 00 00 :<5e>
>1e28 00 00 00 00 00 00 00 00 :<6e>
>1e30 00 00 00 00 00 00 00 00 :<7e>
>1e38 00 00 00 00 00 00 00 00 :<8e>
>1e40 00 00 00 00 00 00 00 00 :<9e>
>1e48 00 00 00 00 00 00 00 00 :<ae>
>1e50 00 00 00 00 00 00 00 00 :<be>
>1e58 00 00 00 00 00 00 00 00 :<ce>
>1e60 00 00 00 00 00 00 00 00 :<de>
>1e68 00 00 00 00 00 00 00 00 :<ee>
>1e70 00 00 00 00 00 00 00 00 :<fe>
>1e78 00 00 00 00 00 00 00 00 :<0e>
>1e80 00 00 00 00 00 00 00 00 :<1f>
>1e88 00 00 00 00 00 00 00 00 :<2f>
>1e90 00 00 00 00 00 00 00 00 :<3f>
>1e98 00 00 00 00 00 00 00 00 :<4f>
>1ea0 00 00 00 00 00 00 00 00 :<5f>
>1ea8 00 00 00 00 00 00 00 00 :<6f>
>1eb0 00 00 00 00 00 00 00 00 :<7f>
>1eb8 00 00 00 00 00 00 00 00 :<8f>
>1ec0 00 00 00 00 00 00 00 00 :<9f>
>1ec8 00 00 00 00 00 00 00 00 :<af>
>1ed0 00 00 00 00 00 00 00 00 :<bf>
>1ed8 00 00 00 00 00 00 00 00 :<cf>
>1ee0 00 00 00 00 00 00 00 00 :<df>
>1ee8 00 00 00 00 00 00 00 00 :<ef>
>1ef0 00 00 00 00 00 00 00 00 :<ff>
>1ef8 00 00 00 00 00 00 00 00 :<0f>
>1f00 00 00 00 00 00 00 00 00 :<1f>
>1f08 00 00 00 00 00 00 00 00 :<2f>
>1f10 00 00 00 00 00 00 00 00 :<3f>
>1f18 00 00 00 00 00 00 00 00 :<4f>
>1f20 00 00 00 00 00 00 00 00 :<5f>
>1f28 00 00 00 00 00 00 00 00 :<6f>
>1f30 00 00 00 00 00 00 00 00 :<7f>
>1f38 00 00 00 00 00 00 00 00 :<8f>
>1f40 00 00 00 00 00 00 00 00 :<9f>
>1f48 00 00 00 00 00 00 00 00 :<af>
>1f50 00 00 00 00 00 00 00 00 :<bf>
>1f58 00 00 00 00 00 00 00 00 :<cf>
>1f60 00 00 00 89 8a 8b 8c 8d :<5f>
>1f68 8e 8f 90 91 01 01 01 01 :<5a>
>1f70 01 01 01 01 01 01 01 01 :<7f>
>1f78 01 01 01 01 01 01 00 00 :<4e>
>1f80 00 00 00 00 00 00 00 00 :<20>
>1f88 00 00 00 00 00 00 00 00 :<30>
>1f90 00 00 00 00 00 00 00 00 :<40>
>1f98 00 00 00 00 00 00 00 00 :<50>
>1fa0 00 00 00 00 00 00 00 00 :<60>
>1fa8 00 00 00 05 05 03 03 00 :<0a>
>1fb0 05 00 00 00 05 00 00 00 :<d5>
>1fb8 00 00 00 00 00 00 00 00 :<90>
>1fc0 00 00 00 00 00 00 00 00 :<a0>
>1fc8 00 00 00 00 00 00 00 00 :<b0>
>1fd0 00 00 00 00 00 00 00 00 :<c0>
>1fd8 00 00 00 00 00 00 00 00 :<d0>
>1fe0 00 00 00 00 00 00 00 00 :<e0>
>1fe8 00 00 00 07 00 92 95 00 :<80>
>1ff0 8d d1 43 8d 62 43 60 ee :<7d>
>1ff8 d1 43 ee 62 43 60 8d d0 :<de>
>2000 43 8d 61 43 60 ff ff ff :<2e>
==========
PROGRAMMENDE
==========
```

# SCA-BASIC

*Fortsetzung von Seite 117*

## ADR

ADR kann zwar theoretisch Werte von 0 bis 255 annehmen. Damit der Rechner nicht abstürzt, sollten aber nur Werte zwischen 16 bis 56 und 79 bis 120 verwendet werden.




Die BASIC-Erweiterung geben Sie ein, indem Sie das Programm CHECKMON laden und starten. Anschließend gehen Sie mit MONITOR in den Maschinenmonitor Ihres Rechners. Tippen Sie nur die Zeilen ab, die am Anfang das Zeichen „>“ enthalten. Was rechts ab dem Doppelpunkt erscheint, brauchen Sie nicht einzugeben. Die erscheinenden zwei Buchstaben zwischen dem Größer- und Kleinerzeichen dienen zur Kontrolle Ihrer Eingabe. Zum Abspeichern richten Sie sich nach den Anweisungen im Programmvorspann. Nach dem Starten mit RUN kopiert sich das Maschinenprogramm in den Bereich von $4000 bis $4EB8. Die Speicheraufteilung ist die, die sich nach einem Aufruf der hochauflösenden Grafik ergibt. Der BASIC-Anfang wurde vorverlegt auf $4EBA. Nach dem Starten von S&A-BASIC können Sie sich von den neuen Möglichkeiten ein Bild machen, indem Sie das Programm S&A-DEMO ausprobieren, das einige der neuen BASIC-Befehle nützt. □

# Jetzt perfekt: Unser Checksummer

Hatte bisher unser Checksummer an Buchstabenvertauschungen nichts auszusetzen, so zeigt er sich nun nicht mehr so kulant.

Ob Sie mit der alten Version nun eingegeben hatten:

```
10 print"ab"
oder
10 print "ba",
```

der Checksummer brachte in beiden Fällen die Prüfsumme < gk > . Leicht kann es vorkommen, daß beim schnellen Tippen, besonders im Zehnfingersystem, die Taste, die eigentlich erst als übernächste drankommen sollte, ein wenig zu früh erwischt wird. Dem Checksummer, der lediglich die Ascii-Werte der Buchstaben addierte, konnte dieses natürlich nicht auffallen. Was also tun? Ob etwas früher oder später addiert wird, ändert nichts am Resultat der Summe. Anders ist es, wenn man zwei Verknüpfungsarten kombiniert. So ist z.B. 2*30+40 etwas anderes als 2*40+30. Und genau dieses war dann die Lösung. Die Summe wird nun einfach durch eine Linksverschiebung vor jeder Addition verdoppelt. Dadurch, daß im Falle, wenn das Ergebnis größer als 255 ist, der dabei entstehende Übertrag als Wert 1 zusätzlich addiert wird, verflüchtigen die Werte der am Anfang der Zeile gefundenen Codes sich nicht nach 8 weiteren Zeichen. Damit bleibt nicht nur die Aussagekraft der Prüfsumme voll erhalten, sondern erfährt sogar eine erhebliche Steigerung. Und vor allen Dingen wird nur eine klitzekleine Änderung erforderlich, die dieses zu vollbringen, in der Lage ist. Ein einziges Byte ist nur zu ändern. Wir tun dieses mit "poke 345,10" in der Zeile 470. Dadurch wird das hier ursprünglich ansässige CLC (Clear Carry) durch ASL (Arithmetik Shift Left) ersetzt. Die nachfolgende Addition mit ADC (Addiere mit Carry) addiert den Ascii-Code des gefundenen Zeichens und den nach links herausgeshifteten Übertrag. Da einige unserer Leser beklagten, daß das Checksummerlisting nachher noch im Programmspeicher stehen würde, haben wir diesem noch mit einem "new" abgeholfen. New bzw. neu ist nun folgendes.

```
10 print"ab" ergibt die Prüfsumme<jd>
10 print"ba" die Prüfsumme<jf>
```

Sie brauchen den Checksummer nicht neu einzutippen. Alles, was Sie tun müssen, ist, die Zeile 470 anzufügen. An der Bedienung des Checksummers hat sich nichts geändert. Die Eingabehinweise bleiben daher wie gehabt.

## EINGABEHINWEISE

Am rechten Rand jedes Listings, jeweils am Ende einer Eingabezeile, finden Sie zwei Buchstaben zwischen einem Kleiner- und einem Größerzeichen eingeschlossen. Diese dürfen Sie nicht mit in Ihr Listing eintippen, sondern sie dienen Ihnen zur Überprüfung Ihrer Eingabe.

Zwischen dem Kleiner- und dem Größerzeichen am rechten Rand befinden sich zwei Buchstaben. Mit einem speziellen Programm können Sie beim Eintippen Ihre Eingabe auf ihre Richtigkeit überprüfen. Dieses Programm, der Checksummer, sorgt nämlich dafür, daß nach erfolgter Zeileneingabe am linken oberen Bildschirmeck zwei Buchstaben ausgegeben werden. Wenn diese Buchstaben nicht mit den vorher erwähnten Buchstaben in unserem Listing übereinstimmen, so können Sie davon ausgehen, daß Sie sich vertippt haben und können sich so die Zeile nochmals näher ansehen, ob Sie Ihren Eingabefehler finden. Wenn Sie dann alles richtig getippt haben, so stimmen die Buchstaben überein und Sie können sich getrost der nächsten Zeile zuwenden.

## ERST SICHERN, DANN AUSPROBIEREN

Das Checksummerlisting hat noch keine Prüfsummen. Seien Sie deshalb besonders aufmerksam, daß alles paßt und speichern Sie dieses Programm unbedingt ab, bevor Sie es starten! Bei einem Tippfehler würde es sich wahrscheinlich auf Nimmerwiedersehen verabschieden und Sie müßten die ganze Arbeit vermutlich nochmals machen. Wenn Sie es gestartet haben, so geschieht nichts Besonderes. Der Computer meldet sich einfach kurz darauf mit „READY", und das war auch schon alles. Alles sollte nun wie immer funktionieren, mit der kleinen Ausnahme, daß nunmehr nach jeder Eingabe im Direktmodus eine Prüfsumme erscheint. Nehmen Sie zum Testen irgendeine kurze Basiczeile aus unserem Heft her und testen sie aus. Wenn die Summen übereinstimmen, so können Sie sich freuen, denn Fehler beim Abtippen werden Ihnen nun in Zukunft viel weniger passieren, als vorher.

## EINER FÜR ALLE, EIN ECHTES UNIVERSAL-PROGRAMM

Unseren Checksummer können Sie verwenden, ob Sie einen C16/116/Plus4 oder ob Sie einen C64 oder gar einen C128 haben. Nur müssen Sie beim letzteren beachten, ob Sie auch wirklich im 40-Zeichenmodus sind. Nachdem Sie den Checksummer geladen und gestartet haben, können Sie Ihr Basicprogramm eingeben wie gewohnt, Sie können es abspeichern, Sie können auch laden, Sie können Kürzel verwenden und, ob Sie ein paar Leerzeichen mehr oder weniger verwenden, der Checksummer läßt sich dadurch nicht aus der Fassung bringen. Ein bißchen Vorsicht sollte man allerdings walten lassen, wenn man Programme eingetippt hat, in denen Peeks und Pokes vorkommen. Es wird zwar nicht besonders häufig vorkommen, aber es könnte bisweilen ge-

```
10 rem =checksummer==c16 c64 c128==
20 rem (p) 05/87 commodore welt   ==
30 rem ============================
40 rem (c) alfons mittelmeyer     ==
50 rem                            ==
60 rem c16/116/plus4              ==
70 rem c64                        ==
80 rem c128 (40-zeichen)          ==
90 rem ============================
100 rem ---------------------------
110 rem    grundroutine (c16)
120 rem ---------------------------
130 data165,059,072,165,060,072,032
140 data086,137,104,133,060,104,133
150 data059,152,072,160,000,165,020
160 data024,101,021,170,024,144,011
170 data201,032,240,006,138,024,113
180 data059,234,170,200,177,059,234
190 data208,240,169,031,072,138,074
200 data074,074,074,072,138,041,015
205 data072,169,031,072,162,003,104
210 data024,105,129,157,000,012,202
220 data016,246,104,168,096
230 lt=peek(772):ht=peek(773)
240 fori=312to386:readx:pokei,x:nex
t
250 iflt<>124then350
260 rem ---------------------------
270 rem        anpassung c64
280 rem ---------------------------
290 fori=312to317:pokei,234:next
300 fori=321to326:pokei,234:next
310 fori=1to6:readad:readx:pokead,x
:next
320 poke380,4:poke319,lt:poke320,ht
:goto430
330 data346,121,347,000,348,002
340 data351,185,352,000,353,002
350 iflt<>13then430
360 rem ---------------------------
370 rem anpassung c128 (40 zeichen)
380 rem ---------------------------
390 restore410:poke332,22
400 poke335,23:goto310
410 data313,061,316,062,323,062
420 data326,061,347,061,352,061
430 poke772,056:poke 773,1
440 rem ---------------------------
450 rem ergaenzung 10/87
460 rem ---------------------------
470 poke 345,10:new
480 rem ============================
490 rem = fuer hefte cw 7/87 bis  =
500 rem = cw 9/87 sowie cw128 5/87=
510 rem = und c16 6/87 ist die    =
520 rem = poke-anweisung in zeile =
530 rem = 470 wegzulassen         =
540 rem ============================
```

schehen, daß nach dem Laufenlassen eines Programmes weder der Checksummer noch sonst etwas mehr funktioniert, auch wenn dies bisher ohne Checksummer nicht der Fall gewesen sein sollte. Also bitte sichern Sie in jedem Falle Ihre Programme, bevor Sie sie ausprobieren.

Ein paar Dinge sollten Sie noch wissen. Wir drucken in unseren Listings des öfteren Punkte statt Leerzeichen. Wenn Ihnen nun aber Leerzeichen besser gefallen, so liefert der Checksummer natürlich eine falsche Summe. Wenn Sie diese Richtigkeit überprüfen wollen, so können Sie dies tun, indem Sie sie zuerst einmal so wie im Heft abtippen, und nachher, nachdem Sie sie nachgeprüft haben, einfach wieder die Punkte durch Leerzeichen ersetzen.

*A. Mittelmeyer*

# MONITOR

## CHECKMON

```
40 rem checkmon ==============c16 <cn>
50 rem (p) commodore welt team    <ke>
60 rem ========================== <nk>
70 rem (c) by  a.mittelmeyer      <ag>
80 rem c16/116/plus4              <ki>
90 rem ========================== <jg>
100 fori=312to398:readа           <ei>
110 pokei,a:next                  <ep>
120 data 132,218,108,219,0,132,219 <oe>
130 data 164,218,76,75,236,201,62 <nk>
140 data 208,249,165,161,10,101   <jc>
150 data 162,160,7,10,113,161,136 <ej>
160 data 16,250,133,216,169,30,133 <oh>
170 data 217,169,62,160,97,208,220 <mk>
180 data 198,217,208,218,160,105  <ai>
190 data 208,212,201,13,240,4,164 <ha>
200 data 218,24,96,169,60,160,68  <lh>
210 data 32,61,1,165,216,32,16,251 <ec>
220 data 169,62,160,5,208,2,169,32 <om>
230 data 32,75,236,136,208,248,169 <ol>
240 data 13,208,176,219,68,220,1  <go>
250 data 804,56,805,1             <hn>
260 fori=1to4:readа:readb:pokea,b <lm>
270 next:new                      <ji>
280 rem ========e=n=d=e=========== <cc>
```

"CHECKMON" ist eine unerlaessliche Hilfe zur Eingabe von Maschinenprogrammen. Laden und starten Sie "Checkmon" und gehen dann mittels MONITOR in denselben. Wenn Sie sich nun z.B. mit 'M1000' einen Speicherbereich ansehen, oder Hexzahlen eingeben, so erscheint rechts die Pruefsumme anstatt der Asciicodes.

ZEICHENCREATOR

# Eigene Zeichen entwickeln auf dem C16

**Mit dem Joystick und menüunterstützt entfällt die lästige Rechnerei. Zeichen editieren wird komfortabel. Besonders wenn Drehungen, Spiegelungen und weitere Funktionen zur Verfügung stehen.**

Mit dem Zeichencreator lassen sich sowohl einfarbige als auch Multicolorzeichen komfortabel erstellen und editieren. Sie können Zeichen invertieren, in andere Zeichen kopieren, scrollen, horizontal und vertikal spiegeln, drehen und vieles mehr.
Der Cursor wird mit den Cursortasten und mit dem Joystick in Port 1 beziehungsweise 2 (wählbar, siehe Menüpunkt 1) gesteuert.
Den Zeicheneditor rufen Sie durch Aktivieren des zweiten Menüpunktes auf. Alle Funktionen und Tasten werden rechts auf dem Bildschirm angezeigt.
Den Zeichenmodus können Sie durch Drücken der SHIFT- und „±"-Taste verändern:
Modus 0 = Löschen;
Modus 1 bis 3 = Punktfarbe 1 bis 3.
Um ein Zeichen in ein anderes zu kopieren, drücken Sie c (1), suchen sich mit „±" ein Zeichen aus (2) und drücken Sie Return (3). Editieren Sie das Zeichen (4), drücken Sie erneut c (5) und suchen Sie das Zeichen mit „±" aus, in das das andere Zeichen kopiert werden soll (6). Jetzt drücken Sie „F" (7).
Falls Sie das Zeichen schon editiert haben, führen Sie nur noch Schritt 5 bis 7 aus.
Wenn Sie Ihre Zeichen in eigenen Programmen benutzen wollen, so wählen Sie den Menüpunkt „Zeichendata". Er sollte aber erst dann angewählt werden, nachdem alle Zeichen editiert sind.
Ab Zeile 8000 entstehen Zeichendata mit Einleseprogramm. Jetzt geben Sie Delete –7000 ein, um nur noch das Einleseprogramm und die Data im Speicher zu haben.
Alles weitere wird im Programm erklärt. Vergessen Sie nicht, vor dem ersten RUN abzuspeichern. □

```
10 rem zeichencreator==========c16 <gj>
20 rem (p) commodore welt team    <ho>
30 rem ========================== <ng>
40 rem (c) by michael inden       <gh>
50 rem                            <pd>
60 rem                            <ah>
70 rem basic v3.5                 <nl>
80 rem c16/116/plus4              <ki>
90 rem ========================== <jg>
100 poke55,0:poke56,56:clr        <bm>
110 rem mit 64 kb zeile 100 ersetz
en durch 'graphic1,1:clr:graphic0' <ka>
120 rem farben                    <am>
130 :                             <dp>
140 color1,1:color0,2:color4,7,5  <cn>
150 fort=1375to1382:poket,0:next  <nk>
160 :                             <hl>
170 rem zeichenkopierroutine      <ga>
180 :                             <kd>
190 fort=0to32:readp:poke1383+t,p:
next                              <am>
200 data169,208,133,209,169,0,133,
208,133,210,169,56,133,211,162,8,1
60,0,177                          <bj>
210 data208,145,210,136,208,249,23
0,209,230,211,202,208,240,96      <gn>
220 sys1383:gosub3340             <fi>
230 :                             <ah>
240 rem variablen                 <mc>
250 :                             <cp>
260 mf=0:mo=1:po=1:g=1:zd=8060:b=1 <la>
270 p(0)=46:p(1)=81:p(2)=87:p(3)=4
2                                 <fc>
280 dimz%(127),d(63)              <fb>
290 mf$(0)=rn$+"normal"+rf$:mf$(1)
=rn$+"multi"+rf$+" "              <hi>
300 :                             <je>
310 poke65287,peek(65287)and239   <be>
320 :                             <lm>
330 rem zeichensatzeditor         <nn>
340 :                             <oe>
350 rem menue                     <mh>
360 :                             <am>
370 printcl$c4$b6$"***> zeichencre
ator c-16 <***"b6$                <ig>
380 printc4$b$" ==================
="                                <fe>
390 printc4$c4$                   <im>
400 printb9$"steuerung............
1"c4$                             <ci>
410 printb9$"zeichensatzeditor....
2"c4$                             <je>
420 printb9$"zeichendatas.........
3"c4$                             <gk>
430 printb9$"normal/multicolor....
4"c4$                             <ii>
440 printb9$"ende.................
5"                                <dh>
450 :                             <mb>
460 printc4$c4$b9$"bitte waehlen s
ie [1-5]"                         <lk>
470 a$="":getkeya$                <fm>
480 a=val(a$)                     <fj>
490 ifa<1ora>5thenprintleft$(qu$,4
):goto460                         <kd>
500 :                             <cf>
510 onagoto540,650,2910,3130,3280 <md>
520 :                             <eo>
530 :                             <gc>
540 rem steuerung                 <km>
550 scnclr                        <di>
560 printc4$c4$c4$" cursosteuerung
 : normal"                        <dj>
570 printc4$c4$" joysticksteuerung
 : port"po" [+]"                  <oh>
580 printc4$c4$" m = "rn$"back to
menue"rf$                         <lf>
590 geta$                         <kl>
600 ifa$="+"thenpo=po+1:ifpo>2then
po=1                              <pp>
610 ifa$="m"thengoto370           <ek>
620 char1,25,6,str$(po)           <ao>
630 goto590                       <dp>
640 :                             <dp>
650 rem zeichensatzeditor         <lm>
660 :                             <gh>
670 printcl$bk$;                  <ki>
680 fort=0to7:print".....B"t:ne
xt                                <ip>
690 print"CCCCCCCC"zs$            <mc>
700 printc4$"76543210"            <ep>
710 printc4$" code:"using"###";zc:
poke3562,zc                       <pe>
720 print" mode: "mo              <bf>
730 char1,15,0,"funktionstasten:" <io>
740 char1,15,1,"================" <hh>
750 char1,15,2,"code = "+re$+"c"+b
k$                                <bo>
760 char1,15,3,"invert = "+re$+"i"
+bk$                              <ce>
770 char1,15,4,"back to menue = "+
re$+"esc"+bk$                     <ap>
780 char1,15,5,"vertikalspiegeln =
 "+re$+"v"+bk$                    <cm>
790 char1,15,6,"hori.-spiegeln = "
+re$+"h"+bk$                      <bp>
800 char1,15,7,"breite 1/2 = "+re$
+"b"+bk$                          <pp>
810 char1,15,8,"punkt s/l = "+re$+
"space/fire"+bk$                  <bk>
820 char1,15,9,"scroll = "+re$+"s"
+bk$                              <fg>
830 char1,15,10,"home = "+re$+"hom
e"+bk$                           <km>
840 char1,15,11,"clear = "+re$+"cl
ear"+bk$                         <kp>
```

```
850 char1,15,12,"drehung = "+re$+"
u"+bk$                                <me>
860 char1,15,13,"alt. zeichen zuru
eck = "+re$+"n"+bk$                   <fa>
870 char1,15,14,"zeichen uebernehm
en = "+re$+"f"+bk$                    <go>
880 char1,0,15,"CCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCC"                <jh>
890 char1,0,18,"scroll = s + curso
r up/down/right/left"                 <od>
900 char1,0,20,"drehung 90/180/270
 = d + 1/2/3"                         <cb>
910 char1,0,22,"return zur ausfueh
rung von drehung und"+b2$             <be>
920 char1,0,23,"scroll druecken"      <oi>
930 gosub2540                         <ik>
940 :                                 <jj>
950 rem bewegung                      <bg>
960 geta$:j=joy(po)                   <ga>
970 poke65299,56:poke65298,0          <gm>
980 ifa$=c3$orj=3thenx=x+b:ifx>8-b
thenx=0                               <co>
990 ifa$=c1$orj=7thenx=x-b:ifx<0th
enx=8-b                               <gi>
1000 ifa$=c2$orj=1theny=y-1:ify<0t
heny=7                                <lj>
1010 ifa$=c4$orj=5theny=y+1:ify>7t
heny=0                                <ef>
1020 ifa$=he$thenx=0:y=0              <jh>
1030 ifa$=cl$thenx=0:y=0:printhe$;
:fort=0to7:print".......":next        <ke>
1040 ifa$="h"thengosub1370            <hi>
1050 ifa$="v"thengosub1450            <oo>
1060 ifa$="i"thengosub1290            <dh>
1070 ifa$="s"thengosub1530            <lg>
1080 ifa$="u"thengosub1980            <ap>
1090 ifmf=1then1100:elseifa$="b"th
enb=b+1:ifb>2thenb=1                  <nl>
1100 ifa$="c"thengosub2400            <jn>
1110 ifa$="n"thengosub2680            <ej>
1120 ifa$="f"thengosub2800            <ij>
1130 ifa$=ym$thenmo=mo+1:ifmo>gthe
nmo=0                                 <ad>
1140 ifa$=yo$thenmo=mo-1:ifmo<0the
nmo=g                                 <af>
1150 ifa$=chr$(27)thengoto370         <pl>
1160 ifa$=" "orj>=127thengoto1230     <me>
1170 poke3072+y*40+x,peek(3072+y*4
0+x)or128                             <mc>
1180 fort=1to30:next:poke3072+y*40
+x,peek(3072+y*40+x)and127            <ch>
1190 char1,0,16,"breite:":printstr
$(b):poke3562,zc                      <kb>
1200 printhe$left$(qd$,12)" code:"
using"###";zc                         <oo>
1210 print" mode: "mo:char1,1,14,"
<shft.[+/-]>"                         <dj>
1220 goto960                          <fa>
1230 p1=3072+y*40+x:p2=3072+y*40+x
+1                                    <kj>
1240 ifb=2andx=7thengoto960           <oh>
1250 ifb=1thenpokep1,p(mo):goto960    <il>
1260 pokep1,p(mo):pokep2,p(mo)        <co>
1270 goto960                          <hb>
1280 :                                <ee>
1290 rem invert                       <dj>
1300 fors=0to7                        <hi>
1310 forz=0to7                        <gl>
1320 ifpeek(3072+s*40+z)=46thenpok
e3072+s*40+z,81:goto1340              <hj>
1330 ifpeek(3072+s*40+z)=81thenpok
e3072+s*40+z,46                       <im>
1340 nextz:nexts                      <fb>
1350 return                           <ce>
1360 :                                <oe>
1370 rem horizontalspiegeln           <hi>
1380 fors=0to3                        <gi>
1390 forz=0to7                        <ep>
1400 p1=3072+s*40+z:p2=3072+(7-s)*
40+z                                  <ce>
1410 pp1=peek(p1):pokep1,peek(p2):
pokep2,pp1                            <ce>
1420 nextz:nexts                      <go>
1430 return                           <mf>
1440 :                                <if>
1450 rem vertikalspiegeln             <ap>
1460 fors=0to7                        <jc>
1470 forz=0to3                        <gp>
1480 p1=3072+s*40+z:p2=3072+s*40+7
-z                                    <hm>
1490 pp1=peek(p1):pokep1,peek(p2):
pokep2,pp1                            <mo>
1500 nextz:nexts                      <ge>
1510 return                           <gf>
1520 :                                <cf>
1530 rem scroll                       <og>
1540 char1,25,16,"scroll:"            <bj>
1550 geta$                            <mn>
1560 ifa$=c2$thenchar1,33,16,b6$+l
eft$(ql$,6)+"hoch":sc=1               <gd>
1570 ifa$=c4$thenchar1,33,16,b6$+l
eft$(ql$,6)+"runter":sc=2             <cm>
1580 ifa$=c3$thenchar1,33,16,b6$+l
eft$(ql$,6)+"rechts":sc=3             <md>
1590 ifa$=c1$thenchar1,33,16,b6$+l
eft$(ql$,6)+"links":sc=4              <gg>
1600 ifa$=chr$(13)thengoto1630        <ik>
1610 goto1550                         <il>
1620 :                                <oo>
1630 onscgosub1660,1740,1820,1900     <gm>
1640 return                           <gk>
1650 :                                <ck>
1660 rem hoch-scroll                  <fg>
1670 fors=1to7                        <gb>
1680 forz=0to7                        <ei>
1690 fort=0tog                        <hk>
```

```
1700 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z-40,p(t):poke3072+s*
40+z,46                          <am>
1710 nextt:nextz:nexts           <el>
1720 return                      <al>
1730 :                           <ml>
1740 rem runter-scroll           <li>
1750 fors=6to0step-1             <pl>
1760 forz=0to7                   <dm>
1770 fort=0to6                   <ff>
1780 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z+40,p(t):poke3072+s*
40+z,46                          <cm>
1790 nextt:nextz:nexts           <ll>
1800 return                      <km>
1810 :                           <gm>
1820 rem rechts-scroll           <ld>
1830 ford=0tomf                  <ip>
1840 fors=0to7:forz=6to0step-1   <ak>
1850 fort=1tog                   <ig>
1860 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z+1,p(t):poke3072+s*4
0+z,46                           <lp>
1870 nextt,z,s,d                 <jn>
1880 return                      <em>
1890 :                           <am>
1900 rem links-scroll            <pg>
1910 ford=0tomf                  <fd>
1920 fors=0to7:forz=1to7step1    <pg>
1930 fort=0tog                   <lm>
1940 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z-1,p(t):poke3072+s*4
0+z,46                           <ko>
1950 nextt,z,s,d                 <ei>
1960 return                      <on>
1970 :                           <kn>
1980 rem drehung                 <bf>
1990 char1,11,16,"drehung:"      <cg>
2000 geta$                       <oh>
2010 ifa$="1"thenchar1,20,16,b3$+c
1$+c1$+c1$+" 90":dr=1            <ph>
2020 ifa$="2"thenchar1,20,16,b3$+c
1$+c1$+c1$+"180":dr=2            <mh>
2030 ifa$="3"thenchar1,20,16,b3$+c
1$+c1$+c1$+"270":dr=3            <id>
2040 ifa$=chr$(13)thengoto2070   <fp>
2050 goto2000                    <li>
2060 :                           <gc>
2070 fors=0to7                   <lj>
2080 forz=0to7                   <ko>
2090 d(s*8+z)=peek(3072+s*40+z)  <np>
2100 poke3072+s*40+z,46          <pl>
2110 nextz:nexts                 <pa>
2120 :                           <nk>
2130 ondrgosub2150,2230,2310     <oa>
2140 return                      <fg>
2150 rem  90 grad                <ck>
2160 fors=0to7                   <bp>
2170 forz=0to7                   <ed>
2180 p2=3072+(z)*40+7-s          <fm>
2190 fort=0tog                   <ph>
2200 ifd(s*8+z)=p(t)thenpokep2,p(t
)                                <cf>
2210 nextt:nextz:nexts           <db>
2220 return                      <ph>
2230 rem 180 grad                <ke>
2240 fors=0to3                   <ea>
2250 forz=0to7                   <dl>
2260 fort=0tog                   <gj>
2270 ifd(s*8+z)=p(t)thenpoke3072+(
7-s)*40+7-z,p(t)                 <pl>
2280 ifd((7-s)*8+7-z)=p(t)thenpoke
3072+s*40+z,p(t)                 <il>
2290 nextt:nextz:nexts           <aj>
2300 return                      <jg>
2310 rem 270 grad                <jd>
2320 fors=0to7                   <hi>
2330 forz=0to7                   <gl>
2340 p2=3072+(7-z)*40+s          <hp>
2350 fort=0tog                   <ap>
2360 ifd(s*8+z)=p(t)thenpokep2,p(t
)                                <oa>
2370 nextt:nextz:nexts           <ja>
2380 return                      <di>
2390 :                           <pi>
2400 rem codewahl                <lp>
2410 :                           <ca>
2420 geta$                       <be>
2430 ifa$="+"thenzc=zc+1:ifzc>127t
henzc=0                          <lh>
2440 ifa$="-"thenzc=zc-1:ifzc<0the
nzc=127                          <mf>
2450 ifa$="f"thengoto2800        <fg>
2460 ifa$=chr$(13)then2520       <ig>
2470 printhe$left$(qd$,12)" "rn$"c
ode"rf$":"using"###";zc;         <pj>
2480 char1,1,14,b6$+" "+rn$+"[+/-]
"+rf$+" ":poke3562,zc            <jb>
2490 goto2420                    <ph>
2500 return                      <cj>
2510 :                           <oj>
2520 rem zeichen vergroessern    <gd>
2530 :                           <bb>
2540 ad=dec("3800")              <jg>
2550 fors=0to7                   <bb>
2560 forz=mfto7+mfstep1+mf       <cj>
2570 p1=(peek(ad+zc*8+s)and2^(7-z)
):ifmf=1thenp2=(peek(ad+zc*8+s)and
2^(7-z+1)):elsep2=p1             <fe>
2580 p=3072+z+s*40               <bn>
2590 ifp1<>2^(7-z)thenifmf=0thengo
to2640:elseifmf=1then2610        <ll>
2600 ifp1=2^(7-z)thenifmf=0thenpok
ep,81:goto2650:else2610          <nd>
2610 ifp1=1*2^(7-z)andp2=1*2^(7-z+
1)thenpokep,81:pokep-1,81:goto2650 <ee>
```

```
2620 ifp1=1*2^(7-z)andp2=0thenpoke
p,87:pokep-1,87:goto2650                <kp>
2630 ifp2=1*2^(7-z+1)andp1=0thenpo
kep,42:pokep-1,42:goto2650              <gp>
2640 pokep,46:ifmf=1thenpokep-1,46      <am>
2650 next:next                          <bp>
2660 goto2500                           <ej>
2670 :                                  <ck>
2680 rem altes zeichen zurueck          <en>
2690 :                                  <fd>
2700 ad=dec("d000")                     <ig>
2710 poke1177,62                        <eh>
2720 fors=0to7                          <ng>
2730 forz=0to7                          <lp>
2740 p=(peek(ad+zc*8+s)and2^(7-z))      <df>
2750 ifp=0thenchar1,z,s,".":elsech
ar1,z,s,"Q"                             <gl>
2760 next:next                          <jd>
2770 poke1177,63                        <bd>
2780 return                             <fl>
2790 :                                  <bl>
2800 rem zeichen uebernehmen            <me>
2810 :                                  <ec>
2820 fors=0to7:p=0                      <ho>
2830 forz=7tomfstep-1-mf                <ho>
2840 p1=3072+z+s*40:ifmf=1thenp2=3
072+z+s*40-1:elsep2=p1                  <jo>
2850 ifpeek(p1)=81orpeek(p2)=81the
np=p+2^(7-z):ifmf=1thenp=p+2^(7-z+
1)                                      <pc>
2860 ifpeek(p1)=87orpeek(p2)=87the
np=p+2^(7-z)                            <hj>
2870 ifpeek(p1)=42orpeek(p2)=42the
np=p+2^(7-z+1)                          <cj>
2880 next:pokedec("3800")+zc*8+s,p
:next                                   <fe>
2890 z%(zc)=1:flag=1:return             <oc>
2900 :                                  <pi>
2910 rem zeichen in datas ablegen       <ih>
2920 :                                  <ca>
2930 ifflag=0thengoto360                <jk>
2940 k=0:fort=0to127:poke1383+t,z%
(t):next                                <kh>
2950 ifpeek(1383+k)=1thenad=dec("3
800"):zd=zd+10:goto3000:elsek=k+1       <ld>
2960 ifk<127thengoto2950                <bk>
2970 goto3080                           <eg>
2980 gosub3340:goto360                  <aj>
2990 :                                  <kn>
3000 gosub3340:printcl$c4$c4$c4$zd
"data"k",";                             <hl>
3010 fort=0to7                          <jn>
3020 d=peek(ad+k*8+t):d$=str$(d)        <ef>
3030 printright$(d$,len(d$)-1)",";
:next:printc1$" "                       <ge>
3040 gosub3120:print"goto2950"chr$
(19)                                    <ce>
3050 :                                  <cf>
3060 poke1319,13:poke1320,13:poke2
39,2:end                                <kj>
3070 :                                  <em>
3080 gosub3340:printcl$c4$c4$c4$zd
+10"data -1"                            <ih>
3090 gosub3120:print"goto2980"chr$
(19)                                    <ab>
3100 :                                  <ik>
3110 poke1319,13:poke1320,13:poke2
39,2:end                                <jp>
3120 print"zd="zd":po="po":k="k+1"
:dimz%(127):";:return                   <gk>
3130 rem normal/multicolor-zeichen      <il>
3140 :                                  <nk>
3150 scnclr                             <jm>
3160 print" aktueller modus : "mf$
(mf)                                    <fj>
3170 print:print                        <ob>
3180 print" soll das zeichen in "r
n$"m"rf$"ulticolor oder "               <ip>
3190 print" "rn$"n"rf$"ormal erste
llt werden ?"                           <oj>
3200 print:print:print                  <ha>
3210 print" QQ = zeichenfarbe 1         <bf>
3220 print" WW = zeichenfarbe 2         <gb>
3230 print" ** = zeichenfarbe 3         <de>
3240 getkeym$                           <pc>
3250 ifm$="m"thenmf=1:g=3:x=2*int(
x/2):b=2:goto370                        <oc>
3260 ifmf=1then370:else:mf=0:b=1:g
=1:goto370                              <hi>
3270 :                                  <np>
3280 rem ende                           <ld>
3290 :                                  <ah>
3300 end                                <gb>
3310 :                                  <cp>
3320 rem nachspann ==============       <do>
3330 rem * farbcodes/steuercodes *      <hb>
3340 c4$=chr$(017):rn$=chr$(018)        <ac>
3350 he$=chr$(019):re$=chr$(028)        <jk>
3360 c3$=chr$(029):bk$=chr$(144)        <ai>
3370 c2$=chr$(145):rf$=chr$(146)        <dc>
3380 c1$=chr$(147):c1$=chr$(157)        <ph>
3390 rem *** zeichensatz/graphik *      <kh>
3400 zs$=chr$(189):ym$=chr$(219)        <lo>
3410 yo$=chr$(221)                      <pe>
3420 rem ********* zeichenfolgen *      <ka>
3430 qd$="":qu$="":ql$="":forq=1to
40                                      <ep>
3440 qd$=qd$+c4$:qu$=qu$+c2$            <lp>
3450 ql$=ql$+c1$                        <bf>
3460 nextq:b$=chr$(32):b2$=b$+b$        <cd>
3470 b3$=b2$+b$:b6$=b3$+b3$             <en>
3480 b9$=b6$+b3$:b$=b9$+b$              <dn>
3490 return                             <on>
3500 rem ========================       <de>
8000 rem einleseprogramm                <lk>
8010 :                                  <am>
```

```
8020 readz:ifz=-1thenpoke65298,0:p
oke65299,56:goto65535                <op>
8030 fort=0to7:readd:poke14336+z*8
+t,d:next:goto65535                  <nb>
8040 :                               <ej>
8050 rem datas fuer neue zeichen     <pg>
8060 :                               <hb>
8070 =p=r=o=g=r=a=m=m=e=n=d=e===== <fg>
```

# BÖRSE

SUCHE COMPUTE Mit u. Sonderheft f. C16/ Plus4, auch C16/Plus4-SPECIAL-Hefte u. andere Lit. Schickt Eure Listen u. Preise bitte schnell an: Wolfgang Polenda, Brückenstr. 15, 5350 Eu.-Roitzheim

Computerzeitschriften RUN, 64er, Data Welt, Input 64 jahrgangsweise zu verkaufen. Außerdem Spektrum, Bild der Wissenschaft, Physik in unserer Zeit, mehrere Jahrgänge, preiswert zu verkaufen. Tel. 07231/ 481493

C16/PLUS4! Direkt aus USA. Freesoft, ca. 2.500 Block z.B. Druckersteuerung, Biorhythmus, Black Jack, Kalenderdruck. Zusätzlich: Lotto, Vokabelprog., monatl. Kosten, Etikettendruck. Schickt 20,– DM an: Wolfgang Koß, Beringstr. 29, 2215 Hademarschen

RELATIVE Dateiverw. der Spitzenklasse f. +4/ C16/116. Freie Maskenerstellung. Bis zu 1500 Datensätze (160 kB). Mit Handbuch 19,50 DM + Porto u. Nachnahme. Jürgen Vowerk, Dörener Weg 29a, 4790 Paderborn, Tel. 05251/58423

+4-USER aus der DDR sucht Superbase f. +4 u. Alles über Superbase u. Script/Plus; bin auch an Software u. Erf.-austausch interessiert. Karbe, Th.-Brugsch-Str. 50, DDR-1115 Berlin

C16/+4-SUPERPROGRAMME, Spiele, Anwender, Kopierprogr. Gratis-Info m. Tips u. Tricks. Liste anfordern. Th. Görtz, Friedr.-Ebert-Str. 113, 6103 Griesheim

hrc e.V. – Plus4 * C16 * C116 * Anwender: Der Verein speziell für uns! Mit Zeitschrift u. Software-Bibliothek. Info beim hrc e.V., Bauerland 15, 4800 Bielefeld 1

C16/64K, verstärktes Netzteil, Floppy 1541 II (gara.); Joystick, Datas, I/O-Port u. Turbo + Steckpl., Bücher, Zeitschriften, ROM-Listing, Basic-Kurs, Ace, Spitzensoftware, NP 1600,– DM, VB 800,– DM. H. Bosshammer, Heckenweg 2, 4920 Lemgo 1, Tel. 05261/87261

SUCHE Tauschpartner f. +4. Nur Top-Software gesucht. Suche auch Bücher u. Hefte f. +4. Listen an Lars Weißbrodt, Erpener Weg 14, 4503 Dissen

VERKAUF/TAUSCH/ KAUF – C16/Plus4. Anwender, Games, Freesoft, Info bei Jürgen Cissarek, Giebelstr. 5, 4650 Gelsenkirchen. 100% Antwort.

C16 + 64 K, 1531, Drukker, viele orig. Anwender- u. Spiel-Progr. (64 K, überw. Kingsoft), Joysticks sowie ca. 3 kg Literatur; NP über 800,– DM, VB 300,– DM. Bernd Mackeldanz, Kanzlerstr. 12, 4000 Düsseldorf 30

30 GAMES für C16/116/ +4 auf Tape für 10,– DM (incl. Porto). Dietmar Neumann, Trierer Str. 398, 5100 Aachen

SUCHE Datasette 1531 oder 1530 m. Cassettenportadapter. M. Greifenhagen, Stöcklerstr. 8, CH-8610 Uster




## IMPRESSUM C16/P4 SPECIAL

erscheint zweimonatlich in der CA-Verlags GmbH, einer Gesellschaft der Aktuell-Gruppe




VERANTWORTLICH FÜR DEN INHALT:
Anton Kult
Alfons Mittelmeyer

REDAKTION UND STÄNDIGE MITARBEITER:
Peter Basch, Harald Beiler, Rosemarie Huber, Lothar Miedel, Michael Reppisch, Rudolf Schmid-Fabian, Torsten Seibt, Hermann Wellesen, Bernd Welte

GESCHÄFTSFÜHRER:
Werner E. Seibt

ANSCHRIFT FÜR ALLE VERANTWORTLICHEN:
Postfach 1161,
8044 Unterschleißheim
Tel.: 089/1298011
Telex: 5214428 cav-d

ANZEIGENVERWALTUNG:
ADV-Mediendienste,
Aindlingerstr. 17–19,
8900 Augsburg 1

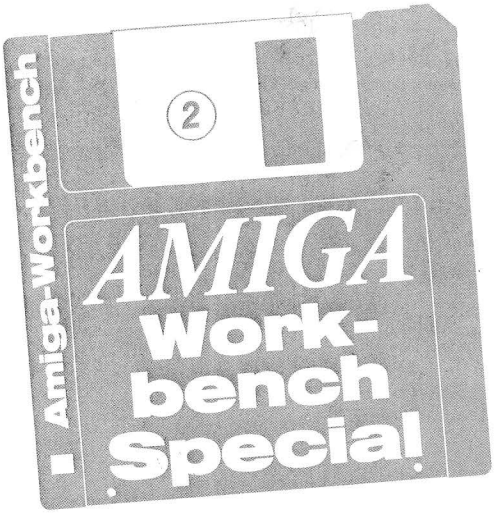
Tel.: 08121/7904-227
Telex: 533502
Teletex: 821887
Telefax: 0821/7904-243

VERANTWORTLICH FÜR DEN ANZEIGENINHALT:
Brigitte Kostić
Es gilt Preisliste Nr. 9 vom 1.6.1988
Media-Unterlagen bitte anfordern.

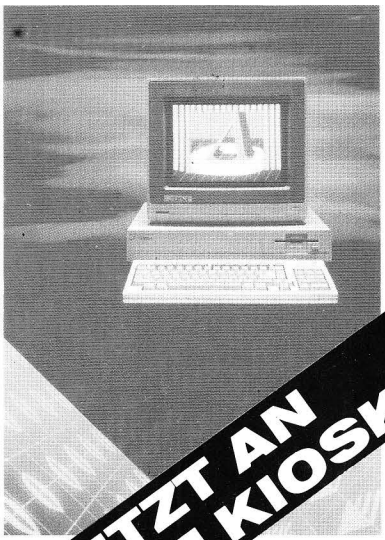
VERTRIEB:
Verlagsunion Wiesbaden

Printed in Germany by ADV, Aindlingerstr. 17–19, 8900 Augsburg 1

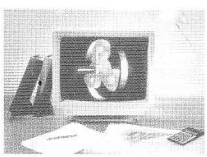
Veröffentlichung gem. §8,3 Bayr.PrG: Gesellschafter der CAV GmbH sind Henny Rose Seibt, Kauffrau, zu 26 v. H. und Werner E. Seibt, Journalist, beide Elisabethstraße 1, 8044 Unterschleißheim, zu 74 v. H.




# AMIGA TEST JAHRBUCH 1988



JETZT AN IHREM KIOSK



**Kaufberatung:**
**Alles, was Sie über Ihren Amiga wissen müssen.**

**Getestet:**
**Hardware, Software, Peripherie**

**Specials:**
**Archimedes, Assembler, Modula**

NEU!

# COMPUTERN LEICHT GEMACHT

## Das PC-Magazin


NEU

**Jetzt an ausgewählten Kiosken und im Bahnhofs-Buchhandel**